Machine Learning Solution Methods for Multistage Stochastic Programming

Thesis by
Boris Defourny

Systems and Modeling Research Unit
Department of Electrical Engineering and Computer Science
University of Liège, Belgium

2010

Abstract

This thesis investigates the following question: *Can supervised learning techniques be successfully used for finding better solutions to multistage stochastic programs?* A similar question had already been posed in the context of reinforcement learning, and had led to algorithmic and conceptual advances in the field of approximate value function methods over the years (Lagoudakis and Parr, 2003; Ernst, Geurts, and Wehenkel, 2005; Langford and Zadrozny, 2005; Antos, Munos, and Szepesvári, 2008). This thesis identifies several ways to exploit the combination "multistage stochastic programming/supervised learning" for sequential decision making under uncertainty.

Multistage stochastic programming is essentially the extension of stochastic programming (Dantzig, 1955; Beale, 1955) to several recourse stages. After an introduction to multistage stochastic programming and a summary of existing approximation approaches based on scenario trees, this thesis mainly focusses on the use of supervised learning for building decision policies from scenario-tree approximations.

Two ways of exploiting learned policies in the context of the practical issues posed by the multistage stochastic programming framework are explored: the fast evaluation of performance guarantees for a given approximation, and the selection of good scenario trees. The computational efficiency of the approach allows novel investigations relative to the construction of scenario trees, from which novel insights, solution approaches and algorithms are derived. For instance, we generate and select scenario trees with random branching structures for problems over large planning horizons. Our experiments on the empirical performances of learned policies, compared to golden-standard policies, suggest that the combination of stochastic programming and machine learning techniques could also constitute a method per se for sequential decision making under uncertainty, inasmuch as learned policies are simple to use, and come with performance guarantees that can actually be quite good.

Finally, limitations of approaches that build an explicit model to represent an optimal solution mapping are studied in a simple parametric programming setting, and various insights regarding this issue are obtained.

Acknowledgments

Warm thanks are addressed to my family for their love and constant support over the years.

I express my deepest gratitude to my advisor, Louis Wehenkel. In addition to his guidance, to his questioning, and to the many discussions we had together, from technical subjects to scientific research in general, Louis has provided me with an outstanding environment for doing research and communicating it, while demonstrating, with style, his large culture, his high ethic, and his formidable ability to be present in times of need. This thesis would not have been written without Louis' interests for supervised learning, ensemble methods, and optimal control.

I am very grateful to Damien Ernst. Together, we had many discussions, for instance on approximate dynamic programming and reinforcement learning, the cross-entropy method, the uses of scenario-tree methods, and the ways to communicate and publish results. Damien has repeatedly provided support and encouragements regarding the present work. Collaborating with Louis and Damien has been extremely energizing, productive, and life-enriching.

I wish to express my gratitude to Rodolphe Sepulchre. Rodolphe too deserves credit for setting up an excellent research environment, for instance by organizing stimulating weekly seminars, group lunches, and offering me the opportunity to participate to activities of the Belgian network DYSCO or to activities hosted by CESAME/INMA(UCL).

Warm thanks to Jean-Louis Lilien for his good advice and support over the years. With hindsight, I owe to Jean-Louis many important choices that I am happy to have made.

I would like to thank Yves Crama for setting up discussions on multistage stochastic programming and models in logistics, and inviting me to presentations given by his group.

I would like to thank Quentin Louveaux for several stimulating discussions.

I express my gratitude to Mania Pavella for her interest and her repeated encouragements.

I am grateful to Jacek Gondzio (University of Edinburgh), Rüdiger Schultz (University of Duisburg-Hessen), Werner Römisch (Humboldt University), and Alexander Shapiro (Georgia Tech) for stimulating discussions on stochastic programming, and expressions of interest in the general orientation of this research. The scientific responsibility of the present work and the views expressed in the thesis rest with us.

I am grateful to Rémi Munos (INRIA Lille) and Olivier Teytaud (INRIA Saclay) for stimulating discussions related to machine learning and sequential decision making.

I am grateful to Roman Belavkin (Middlesex University) for stimulating discussions on cognitive sciences, on finance, and for inviting me to give a seminar in his group.

I am grateful to Jovan Ilic and Marija Ilic (Carnegie Mellon University) for meetings and discussions from which my interest in risk-aware decision making dates back.

My interest in multistage stochastic programming originates from meetings with members of the department OSIRIS (Optimisation, Simulation, Risque et Statistiques) at Electricité de France. I would like to thank especially Yannick Jacquemart, Kengy Barty, Pascal Martinetto, Jean-Sébastien Roy, Cyrille Strugarek, and Gérald Vignal for inspiring discussions on sequential decision making models and current challenges in the electricity industry. The scientific responsibility of the present work and the views expressed in the thesis rest with us.

Warm thanks to my friends, colleagues and past post-docs from the Systems and Modeling research unit and beyond. I had innumerable scientific and personal discussions with Alain Sarlette and Michel Journée, and during their post-doc time with Emre Tuna and Silvère Bonnabel. I had wonderful time and discussions with Pierre Geurts, Vincent Auvray, Guy-Bart Stan, Renaud Ronsse, Christophe Germay, Maxime Bonjean, Luca Scardovi, Denis Efimov, Christian Lageman, Alexandre Mauroy, Gilles Meyer, Marie Dumont, Pierre Sacré, Christian Bastin, Guillaume Drion, Anne Collard, Laura Trotta, Bertrand Cornélusse, Raphaël Fonteneau, Florence Belmudes, François Schnitzler, François Van Lishout, Gérard Dethier, Olivier Barnich, Philippe Ries, and Axel Bodart. I extend my acknowledgements to Thierry Pironet and Yasemin Arda (HEC-Ulg). Thanks also to Patricia Rousseaux, Thierry Van Cutsem and Mevludin Glavic. Many thanks to my friends who encouraged me to pursue this work, and in particular to Estelle Derclaye (University of Nottingham).

I am also indebted to people who helped me when I was abroad for conferences and made my time there even more enjoyable: Marija Prica, Guillaume Obozinski, Janne Kettunen, Aude Piette, Diana Chan, Daniel Bartz, Kazuya Haraguchi.

I had the opportunity to be invited to be a coauthor to papers by Sourour Ammar, Philippe Leray (INSA Rouen) and Louis Wehenkel, and to a paper by Bertrand Cornélusse, Gérald Vignal and Louis Wehenkel, and I wish to thank these authors for these collaborations.

I gratefully acknowledge the financial support of the Belgian network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with us. This work was supported in part by the IST Programme of the European Union, under the PASCAL2 Network of Excellence, IST-2007-216886. This thesis only reflects our views.

Finally, I would like to express my extreme gratitude to the members of my thesis defense committee: Rodolphe Sepulchre (Chair), Louis Wehenkel (Advisor), Yves Crama, Quentin Louveaux, Shie Mannor (Technion), Werner Römisch (Humboldt University), Alexander Shapiro (Georgia Tech), and Olivier Teytaud (INRIA Saclay/Université Paris-Sud).

Contents

Chapter 1

Introduction

Multistage stochastic programming has attracted a lot of interest over the last years, as a promising framework for formulating sequential decision making problems under uncertainty. Several potential applications of the framework are often cited:

- Capacity planning: finding the location and size of units or equipment, such as power plants or telecommunication relays.
- Production planning: selecting components to produce, allocating components to machines, managing stocks.
- Transportation and logistics: sending trucks and deliver goods.
- Financial management: balancing a portfolio of assets and liabilities according to market conditions and subject to regulatory constraints.

In these applications, uncertainty may refer to the evolution of the demand for goods or services, temperature and rainfall patterns affecting consumption or production, interest rates affecting the burden of debt, . . . Under growing environmental stress, resource limitations, concentration of populations in cities, many believe that these applications can only get a higher societal impact in the future, and that even better quantitative methods for tackling them are needed, especially methods able to take into account a large number of constraints.

In general, problems where a flexible plan of successive decisions has to be implemented, under uncertainties described by a probabilistic model, can be formulated as a multistage stochastic program (Chapter 2). However, scalable numerical solution algorithms are not always available, so that restrictions to certain classes of programs and then further approximations are needed.

Interestingly, the approximations affect primarily the representation of the uncertainty, rather than the space of possible decisions or the space of possible states reachable by the controlled system. Thus, the limitations with the problem dimensions suffered by the multistage stochastic framework are of a different nature than those found in dynamic programming (the so-called curse of dimensionality). The multistage stochastic programming framework is very attractive for settings where decisions in high-dimensional spaces must be found, but suffers quickly from the dimensions of the uncertainty, and from the extension of the planning horizon.

This thesis deals with some aspects related to multistage stochastic programming. Our research was initially motivated by finding ways to incorporate to the multistage stochastic programming framework recent advances in statistics and machine learning, especially perturb-and-combine estimation methods, and value function approximation methods from approximate dynamic programming.

In this thesis, we propose and implement on a series of test problems a fast approach, based on supervised learning, for estimating the quality of an approximate solution. We show that this approach is flexible and tractable enough to foster advances in the ways multistage stochastic programming problems are approximated, by explicitly proposing and evaluating novel approximation procedures, according to two criteria: the quality of the approximate solution for the true problem, and the overall computational complexity of the procedure.

A detailed account of the contributions presented in the thesis can be found in Section 7.1.

1.1 Outline of the Thesis

The thesis is organized as follows.

Chapter 2 introduces the multistage stochastic programming approach to sequential decision making under uncertainty, and several notions used throughout the thesis. It discusses the value of multistage stochastic programming with respect to related approaches, and presents the main challenges and limitations of the multistage stochastic programming approach.

Chapter 3 reviews some approaches to statistical estimation investigated in machine learning. Then, it explores the idea of aggregating in a certain sense the solutions to various approximations of the same multistage problem.

Chapter 4 develops the principles of a solution validation approach, based on supervised learning, and shows how it can be exploited so as to identify good approximations to multistage programs under tight complexity limitations. Then, it proposes and evaluates on a family of test problems a new approximate solution approach, based on the generation of several approximations (scenario trees) rather than a single one.

Chapter 5 investigates further methods for estimating the value of a single approximation to a multistage program, in the practical context of a test problem.

Chapter 6 develops an efficient procedure for predicting the optimal solution of a certain class of parametric programs, with the aim of better characterizing the potential limitations of approaches based on learning.

Chapter 7 concludes by a summary of contributions, a discussion on future research directions, and some thoughts about the possible impacts on machine learning and artificial intelligence of the research in stochastic programming.

Some mathematical background, deemed not essential to be imposed as a preliminary reading, has been put in a series of appendices. The reasons for including in the thesis the material of a given appendix are detailed at the beginning of the appendix. The appendices could also be handy to clarify some statements in the main body of the thesis, and to this end, the content of the appendices has been referenced in an index placed at the end of the thesis.

The appendices are organized as follows.

Appendix A defines notions from optimization and variational analysis.

Appendix B defines notions from measure and probability theory.

Appendix C defines notions from functional analysis related to kernel methods.

Appendix D summarizes some results from two-stage stochastic programming.

1.2 Published Work

Whereas the material of Chapters 5 and 6 is still unpublished, most of the material of Chapters 2, 3, 4 has been published in the following papers.

• B. Defourny, L. Wehenkel. 2007. Projecting generation decisions induced by a stochastic program on a family of supply curve functions. *Third Carnegie Mellon Conference on the Electricity Industry.* Pittsburgh PA. 6 pages.

• B. Defourny, D. Ernst, L. Wehenkel. 2008. Lazy planning under uncertainty by optimizing decisions on an ensemble of incomplete disturbance trees. S. Girgin, M. Loth, R. Munos, editors, *Recent Advances in Reinforcement Learning, Eighth European Workshop (EWRL-2008).* LNCS (LNAI) 5323, Springer, 1–14.

• B. Defourny, D. Ernst, L. Wehenkel. 2009. Planning under uncertainty, ensembles of disturbance trees and kernelized discrete action spaces. *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL-2009).* 145–152.

• B. Defourny, D. Ernst, L. Wehenkel. 2009. Bounds for multistage stochastic programs using supervised learning strategies. O. Watanabe, T. Zeugmann, editors, *Stochastic Algorithms: Foundations and Applications. Fifth International Symposium, SAGA 2009.* LNCS 5792, Springer, 61–73.

• B. Defourny, D. Ernst, L. Wehenkel. 2010. Multistage stochastic programming: A scenario tree based approach to planning under uncertainty. Accepted as a contributing chapter to L.E. Sucar, E.F. Morales, and J. Hoey, editors, *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions.* To be published by IGI Global.

Work that uses concepts or algorithms from stochastic programming, and addresses specific topics in machine learning, has also been presented in the following papers.

• B. Defourny, D. Ernst, L. Wehenkel. 2008. Risk-aware decision making and dynamic programming. Y. Engel, M. Ghavamzadeh, S. Mannor, P. Poupart, editors, *NIPS-08 workshop on model uncertainty and risk in reinforcement learning.* 8 pages.

• B. Defourny, L. Wehenkel. 2009. Large margin classification with the progressive hedging algorithm. S. Nowozin, S. Sra, S. Vishwanathan, S. Wright, editors, *Second NIPS workshop on optimization for machine learning.* 6 pages.

Finally, some work to which we collaborated can be recorded.

• S. Ammar, P. Leray, B. Defourny, L. Wehenkel. 2008. High-dimensional probability density estimation with randomized ensembles of tree structured Bayesian networks. *Fourth European Workshop on Probabilistic Graphical Models (PGM 2008).* 9–16.

• S. Ammar, P. Leray, B. Defourny, L. Wehenkel. 2009. Probability density estimation by perturbing and combining tree structured Markov networks. *Tenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2009).* 158–167.

- B. Cornélusse, G. Vignal, B. Defourny, L. Wehenkel. 2009. Supervised learning of intra-daily recourse strategies for generation management under uncertainties. *IEEE Power Tech Conference.* 8 pages.

Chapter 2

The Multistage Stochastic Programming Framework

This chapter presents the multistage stochastic programming approach to sequential decision making under uncertainty. It points out important issues posed by the approach, and discusses the value of the framework with respect to related frameworks.

The chapter is organized as follows. Section 2.1 presents the multistage stochastic programming framework, the discretization techniques, and the considerations on numerical optimization methods that have an influence on the way problems are modeled. Section 2.2 compares the approach to Markov Decision Processes, discusses the curse of dimensionality, and puts in perspective simpler decision making models based on numerical optimization, such as two-stage stochastic programming with recourse or Model Predictive Control. Section 2.3 explains the issues posed by the dominant approximation/discretization approach for solving multistage programs (which is suitable for handling both discrete and continuous random variables). Section 2.4 provides some background information on existing approximation methods. Finally, Section 2.5 concludes by our summary of today's perception of multistage stochastic programming among researchers.

2.1 *Description of the Framework*

In this section, we describe an attitude towards risk and uncertainty that can motivate decision makers to employ multistage stochastic programming. Then, we detail the elements of the decision model and the approximations that can make the model tractable.

2.1.1 *From Nominal Plans to Decision Processes*

In their first attempt towards planning under uncertainty, decision makers often set up a course of actions, or *nominal plan* (reference plan), deemed to be robust to uncertainties in some sense, or to be a wise bet on future events. Then, they apply the decisions, often departing from the nominal plan to better take account of actual events. To further improve the plan, decision makers are then led to consider (i) in which parts of the plan flexibility in the decisions may help to better fulfill the objectives, and (ii) whether the process by which they make themselves (or the system) "ready to react" impacts the initial decisions of the plan and the overall objectives. If the answer to (ii) is positive, then it becomes valuable to cast the decision problem as a sequential decision making problem, even if the net added value of doing so (benefits minus increased complexity) is unknown at this stage. During the planning process, the adaptations (or *recourse decisions*) that may be needed are clarified, their influence on prior decisions is quantified. The notion of nominal plan is replaced by the notion of *decision process*, defined as a course of actions driven by observable events. As distinct outcomes have usually antagonist effects on ideal prior decisions, it becomes crucial to determine which outcomes should be considered, and what importance weights should be put on these outcomes, in the perspective of selecting decisions under uncertainty that are not regretted too much after the dissipation of the uncertainty by the course of real-life events.

2.1.2 Incorporating Probabilistic Reasoning

In the *robust optimization* approach to decision making under uncertainty, decision makers are concerned by worst-case outcomes. Describing the uncertainty is then essentially reduced to drawing the frontier between events that should be considered and events that should be excluded from consideration (for instance, because they would paralyze any action). In that context, outcomes under consideration form the *uncertainty set*, and decision making becomes a game against some hostile opponent that selects the worst outcome from the uncertainty set. Ben-Tal et al. (2004) provide arguments in favor of robust approaches.

In a *stochastic programming* approach, decision makers use a softer frontier between possible outcomes, by assigning weights to outcomes and optimizing some aggregated measure of performance that takes into account all these possible outcomes. In that context, the weights are often interpreted as a probability measure over the events, and a typical way of aggregating the events is to consider the expected performance under that probability measure.

Furthermore, interpreting weights as probabilities allows *reasoning under uncertainty.* Essentially, probability distributions are conditioned on observations, and Bayes' rule from probability theory quantifies how decision makers' initial beliefs about the likelihood of future events — be it from historical data or from bets — should be updated on the basis of new observations.

Technically, it turns out that the optimization of a decision process contingent to future events is more tractable (read: suitable to large-scale operations) when the "reasoning under uncertainty" part can be decoupled from the optimization process itself. In particular, such a decoupling occurs when the probability distributions describing future events are not influenced in any way by the decisions selected by the agent, that is, when the uncertainty is exogenous to the decision process.

2.1.3 The Elements of the General Decision Model

We can now describe the main elements of a multistage stochastic programming decision model. These elements are:

i. A sequence of random variables $\xi_1, \xi_2, \ldots, \xi_T$ defined on a probability space $(\Omega, \mathcal{B}, \mathbb{P})$. The random variables represent the uncertainty in the decision problem, and their possible values represent the possible observations to which the decision maker will react. The probability measure $\mathbb{P}$ serves to quantify the prior beliefs about the uncertainty. There is no restriction on the structure of the random variables; in particular, the random variables may be dependent. When the realization of $\xi_1, \dots, \xi_{t-1}$ is known, there is a residual uncertainty represented by the random variables $\xi_t, \dots, \xi_T$, the distribution of which in now conditioned on the realization of $\xi_1, \dots, \xi_{t-1}$.

ii. A sequence of decisions u_1, u_2, …, u_T defining the decision process for the problem. Many models also use a terminal decision u_{T+1}. We will assume that u_t is valued in a Euclidian space $\mathbb{R}^m$ (the space dimension m, corresponding to a number of scalar decisions, could vary with the index t, but we will not stress that in the notation).

iii. A convention specifying when decisions should actually be taken and when the realizations of the random variables are actually revealed. This means that if ξ_{t-1} is observed before taking a decision u_t, we can actually adapt u_t to the realization of ξ_{t-1}. To this end, we identify *decision stages*: see Table 2.1. A row of the table is read as follows: at decision stage $t > 1$, the decisions $u_1, \dots, u_{t-1}$ are already implemented (no modification is possible), the realization of the random variables $\xi_1, \dots, \xi_{t-1}$ is known, the realization of the random variables $\xi_t, \dots, \xi_T$ is still unknown but a density $\mathbb{P}(\xi_t, \dots, \xi_T \mid \xi_1, \dots, \xi_{t-1})$ conditioned on the realized value of $\xi_1, \dots, \xi_{t-1}$ is available, and the current decision to take concerns the value of u_t. Once such a convention holds, we need not stress in the notation the difference between random variables ξ_t and their realized value, or decisions as functions of uncertain events and the actual value for these decisions: the correct interpretation is clear from the context of the current decision stage.

   The adaptation of a decision u_t to prior observations $\xi_1, \dots, \xi_{t-1}$ will always be made in a deterministic fashion, in the sense that u_t is uniquely determined by the value of $(\xi_1, \dots, \xi_{t-1})$.

   A sequential decision making problem has more than two decision stages inasmuch as the realizations of the random variables are not revealed simultaneously: the choice of the decisions taken between successive observations has to take into account some residual uncertainty on future observations. If the realization of several random variables is revealed before actually taking a decision, then the corresponding random variables should be merged into a single random vector; if several decisions are taken without intermediary observations, then the corresponding decisions should be merged into a single decision vector (Gassmann and Prékopa, 2005). This is how a problem concerning several time periods could actually be a two-stage stochastic program, involving two large decision vectors u_1 (first-stage decision, constant), u_2 (recourse decision, adapted to the observation of ξ_1). What is called a *decision* in a stochastic programming model may thus actually correspond to several actions implemented over a certain number of discrete time periods.

iv. A sequence of feasibility sets $\mathcal{U}_1, \dots, \mathcal{U}_T$ describing which decisions $u_1, \dots, u_T$ are admissible. When $u_t \in \mathcal{U}_t$, one says that u_t is feasible. The feasibility sets $\mathcal{U}_2, \dots, \mathcal{U}_T$ may depend, in a deterministic fashion, on available observations and prior decisions. Thus, following Table 2.1, $\mathcal{U}_t$ may depend on ξ_1, u_1, ξ_2, u_2, …, ξ_{t-1} in a deterministic fashion. Note that prior decisions are uniquely deter-

Tab. 2.1: Decision stages, setting the order of observations and decisions.

Stage	Available information for taking decisions			Decision
	Prior decisions	Observed outcomes	Residual uncertainty	
1	none	none	$\mathbb{P}(\xi_1,\dots,\xi_T)$	u_1
2	u_1	ξ_1	$\mathbb{P}(\xi_2,\dots,\xi_T \mid \xi_1)$	u_2
3	u_1, u_2	ξ_1, ξ_2	$\mathbb{P}(\xi_3,\dots,\xi_T \mid \xi_1, \xi_2)$	u_3
$\vdots$				$\vdots$
T	$u_1,\dots,u_{T-1}$	$\xi_1,\dots,\xi_{T-1}$	$\mathbb{P}(\xi_T \mid \xi_1,\dots,\xi_{T-1})$	u_T
optional:				
$T{+}1$	$u_1,\dots,u_T$	$\xi_1,\dots,\xi_T$	none	(u_{T+1})

mined by prior observations, but for convenience we keep track of prior decisions to parametrize the feasibility sets.

An important role of the feasibility sets is to model how decisions are affected by prior decisions and prior events. In particular, a situation with no possible recourse decision ($\mathcal{U}_t$ empty at stage t, meaning that no feasible decision $u_t \in \mathcal{U}_t$ exists) is interpreted as a catastrophic situation to be avoided at any cost.

We will always assume that the planning agent knows the set-valued mapping from the random variables $\xi_1,\dots,\xi_{t-1}$ and the decisions $u_1,\dots,u_{t-1}$ to the set $\mathcal{U}_t$ of feasible decisions u_t.

We will also assume that the feasibility sets are such that a feasible sequence of decisions $u_1 \in \mathcal{U}_1,\dots,u_T \in \mathcal{U}_T$ exists for all possible joint realizations of $\xi_1,\dots,\xi_T$. In particular, the fixed set $\mathcal{U}_1$ must be nonempty. A feasibility set $\mathcal{U}_t$ parametrized only by variables in a subset of $\{\xi_1,\dots,\xi_{t-1}\}$ must be nonempty for any possible joint realization of those variables. A feasibility set $\mathcal{U}_t$ also parametrized by variables in a subset of $\{u_1,\dots,u_{t-1}\}$ must be implicitly taken into account in the definition of the prior feasibility sets, so as to prevent immediately a decision maker from taking a decision at some earlier stage that could lead to a situation at stage t with no possible recourse decision ($\mathcal{U}_t$ empty), be it for all possible joint realizations of the subset of $\{\xi_1,\dots,\xi_{t-1}\}$ on which $\mathcal{U}_t$ depends, or for some possible joint realization only. These implicit requirements will affect in particular the definition of $\mathcal{U}_1$.

For example, assume that $u_{t-1},\ u_t \in \mathbb{R}^m$, and take $\mathcal{U}_t = \{u_t \in \mathbb{R}^m : u_t \succeq 0,\ A_{t-1}u_{t-1} + B_t u_t = h_t(\xi_{t-1})\}$ with $A_{t-1},\ B_t \in \mathbb{R}^{q\times m}$ fixed matrices, and h_t an affine function of ξ_{t-1} with values in $\mathbb{R}^q$. If B_t is such that $\{B_t u_t : u_t \geq 0\} = \mathbb{R}^q$, meaning that for any $v \in \mathbb{R}^q$, there exists some $u_t \succeq 0$ with $B_t u_t = v$, then this is true in particular for $v = h_t(\xi_{t-1}) - A_{t-1}u_{t-1}$, so that $\mathcal{U}_t$ is never empty. More details on such conditions can be found in Appendix D.

v. A performance measure, summarizing the overall objectives of the decision maker, that should be optimized. It is assumed that the decision maker knows the perfor-

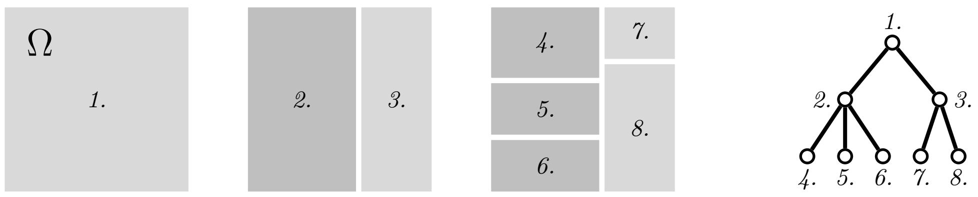

Fig. 2.1: (From left to right) Nested partitioning of the event space Ω, starting from a trivial partition representing the absence of observations. (Rightmost) Scenario tree corresponding to the partitioning process.

mance measure. In this chapter, we write the performance measure as the expectation of a function f that assigns some scalar value to each realization of $\xi_1, \dots, \xi_T$ and $u_1, \dots, u_T$, assuming the integrability of f with respect to the joint distribution of $\xi_1, \dots, \xi_T$.

For example, one could take for f a sum of scalar products $\sum_{t=1}^{T} c_t \cdot u_t$, where c_1 is fixed and where c_t depends affinely on $\xi_1, \dots, \xi_{t-1}$. The function f would represent a sum of instantaneous costs over the planning horizon. The decision maker would be assumed to know the vector-valued mapping from the random variables $\xi_1, \dots, \xi_{t-1}$ to the vector c_t, for each t.

Besides the expectation, more sophisticated ways to aggregate the distribution of f into a single measure of performance have been investigated (Ruszczyński and Shapiro, 2006; Pflug and Römisch, 2007). An important element considered in the choice of the performance measure is the tractability of the resulting optimization problem.

The planning problem is then formalized as a mathematical programming problem. The formulation relies on a particular representation of the random process $\xi_1, \dots, \xi_T$ in relation with the decision stages, referred to as a *scenario tree* in the stochastic programming literature, and described in the next section.

2.1.4 The Tree Representation of Gradually Revealed Scenarios

Let us call *scenario* an outcome of the random process $\xi_1, \dots, \xi_T$. A *scenario tree* is an explicit representation of the branching process induced by the gradual observation of $\xi_1, \dots, \xi_T$, under the assumption that the random variables have a finite discrete support. It is built as follows. A *root node* is associated to the first decision stage and to the initial absence of observations. To the root node are connected children nodes associated to stage 2, one child node for each possible outcome of the random variable ξ_1. Then, to each node of stage 2 are connected children nodes associated to stage 3, one for each outcome of ξ_2 given the observation of ξ_1 relative to the parent node. The branching process construction goes on until the last stage is reached; at this point, the outcomes associated to the nodes on the unique path from the root to a leaf define together a particular scenario, that can be associated to the leaf.

The probability distribution of the random variables is also taken into account. Probability masses are associated to the nodes of the scenario tree. The root node has probability 1, whereas children nodes are weighted by probabilities that represent the probability of the value to which they are associated, conditioned on the value associated to their ancestor node. Multiplying the probabilities of the nodes of the path from the root to a leaf gives the probability of a scenario.

Clearly, an exact construction of the scenario tree would require an infinite number of nodes if the support of $(\xi_1, \dots, \xi_T)$ is discrete but not finite. A random process involving continuous random variables cannot be represented as a scenario tree; nevertheless, the scenario tree construction turns out to be instrumental in the construction of approximations to nested continuous conditional distributions.

Branchings are essential to represent residual uncertainty beyond the first decision stage. At the planning time, the decision makers may contemplate as many hypothetical scenarios as desired, but when decisions are actually implemented, the decisions cannot depend on observations that are not yet available. We have seen that the decision model specifies, with decision stages, how the scenario actually realized will be gradually revealed. No branchings in the representation of the outcomes of the random process would mean that after conditioning on the observation of ξ_1, the outcome of $\xi_2, \dots, \xi_T$ could be predicted (anticipated) exactly. Under such a representation, decisions spanning stages 2 to T would be optimized on the anticipated outcome. This would be equivalent to optimizing a nominal plan for $u_2, \dots, u_T$ that fully bets on some scenario anticipated at stage 2.

To visualize how information on the realization of the random variables becomes gradually available, it is convenient to imagine nested partitions of the event space (Figure 2.1): refinements of the partitions appear gradually at each decision stage in correspondence with the possible realizations of the new observations. To each subregion induced by the partitioning of the event space can be associated a constant recourse decision, as if decisions were chosen according to a piecewise constant decision policy. On Figure 2.1, the surface of each subregion could also represent probabilities (then by convention the initial square has a unit surface and the thin space between subregions is for visual separation only). The dynamical evolution of the partitioning can be represented by a scenario tree: the nodes of the tree corresponds to the subregions of the event space, and the edges between subregions connect a parent subregion to its refined subregions obtained by one step of the recursive partitioning process.

Ideally a scenario tree should cover the totality of possible outcomes of a random process. But unless the support of the distribution of the random variables is finite, no scenario tree with a finite number of nodes can represent exactly the random process and the probability measure, as we already mentioned, while even if the support is finite, the number of scenarios grows exponentially with the number of stages.

2.1.5 *Approximating Random Processes with Scenario Trees*

In the general decision model, the agent is assumed to have access to the joint probability distributions, and is able to derive from it the conditional distributions listed in Table 2.1. In practice, computational limitations will restrict the quality of the representation of $\mathbb{P}$. Let us however reason at first at an abstract and ideal level to establish the program that an agent would solve for planning under uncertainty.

For brevity, let ξ denote $(\xi_1, \dots, \xi_T)$, and let $\pi(\xi)$ denote a *decision policy* mapping realizations of ξ to realizations of the decision process $u_1, \dots, u_T$. Let $\pi_t(\xi)$ denote

u_t viewed as a function of ξ. To be consistent with the decision stages, the policy must be *non-anticipative*, in the sense that u_t cannot depend on observations relative to subsequent stages. Equivalently one can say that π_1 must be a constant-valued function, π_2 a function of ξ_1, and in general π_t a function of $\xi_1, \dots, \xi_{t-1}$ for $t = 2, \dots, T$.

The planning problem can then be stated as the search for a non-anticipative policy π, restricted by the feasibility sets $\mathcal{U}_t$, that minimizes an expected total cost f spanning the decision stages and determined by the scenario ξ and the decisions $\pi(\xi)$:

$$\begin{aligned} \mathcal{P} : \quad \text{minimize} \quad & \mathbb{E}\{f(\xi, \pi(\xi))\} \quad \text{subject to} \quad \pi_t(\xi) \in \mathcal{U}_t(\xi) \ ; \\ & \pi(\xi) \text{ non-anticipative.} \end{aligned}$$

Here we used an abstract notation which hides the nested expectations corresponding to the successive random variables, and the possible sum decomposition of the function f among the decision stages. Concrete formulations are presented in Appendix D. Note that it is possible to be more general by replacing the expectation operator by a functional $\Phi\{\cdot\}$ that maps the distribution of f to a single number in $[-\infty, \infty]$. We also stressed the possible dependence of $\mathcal{U}_t$ on $\xi_1, u_1, \xi_2, u_2, \dots, \xi_{t-1}$ by writing $\mathcal{U}_t(\xi)$.

A program more amenable to numerical optimization techniques is obtained by representing $\pi(\cdot)$ by a set of optimization variables for each possible argument of the function — for each possible outcome $\xi^k = (\xi_1^k, \dots, \xi_T^k)$ of ξ, one associates the optimization variables $(u_1^k, \dots, u_T^k)$, written u^k for brevity. The non-anticipativity of the policy can be expressed by a set of equality constraints: for the first decision stage ($t = 1$) we require $u_1^k = u_1^j$ for all (k, j), and for subsequent stages ($t \geq 2$) we require $u_t^k = u_t^j$ for each (k, j) such that $(\xi_1^k, \dots, \xi_{t-1}^k) \equiv (\xi_1^j, \dots, \xi_{t-1}^j)$.

A finite-dimensional approximation to the program $\mathcal{P}$ is obtained by considering a finite number n of outcomes, and assigning to each outcome a probability $p^k > 0$. This yields a formulation on a scenario tree covering the scenarios ξ^k:

$$\begin{aligned} \mathcal{P}' : \quad \text{minimize} \quad & \textstyle\sum_{k=1}^n p^k \, f(\xi^k, u^k) \quad \text{subject to} \quad u_t^k \in \mathcal{U}_t(\xi^k) \quad \forall\, k \ ; \\ & u_1^k = u_1^j \quad \forall\, k, j \ , \\ & u_t^k = u_t^j \quad \text{whenever} \quad (\xi_1^k, \dots, \xi_{t-1}^k) \equiv (\xi_1^j, \dots, \xi_{t-1}^j) \ . \end{aligned}$$

Once again we used a simple notation ξ^k for designating outcomes of the process ξ, which hides the fact that outcomes can share some elements according to the branching structure of the scenario tree.

Non-anticipativity constraints can also be accounted for implicitly. A partial path from the root (depth 0) to some node of depth t of the scenario tree identifies some outcome $(\xi_1^k, \dots, \xi_t^k)$ of $(\xi_1, \dots, \xi_t)$. To the node can be associated the decision u_{t+1}^k, but also all decisions u_{t+1}^j such that $(\xi_1^k, \dots, \xi_t^k) \equiv (\xi_1^j, \dots, \xi_t^j)$. Those decisions are redundant and can be merged into a single decision on the tree, associated to the considered node of depth t.

2.1.6 Simple Example of Formulation

To fix ideas, we illustrate the scenario tree technique on a trajectory tracking problem under uncertainty with control penalization. In the proposed example, the uncertainty is such that the exact problem can be posed on a small finite scenario tree.

Say that a random process $\xi = (\xi_1, \xi_2, \xi_3)$, representing perturbations at time $t = 1, 2, 3$, has 7 possible outcomes (scenarios), denoted by ξ^k, $1 \le k \le 7$, with known probabilities p^k:

k	1	2	3	4	5	6	7
ξ_1^k	-4	-4	-4	3	3	3	3
ξ_2^k	-3	2	2	-3	0	0	2
ξ_3^k	0	-2	1	0	-1	2	1
p^k	0.1	0.2	0.1	0.2	0.1	0.1	0.2

The random process is fully represented by the scenario tree of Figure 2.2 (Left): the first possible outcome is $\xi^1 = (-4, -3, 0)$ with probability $p^1 = 0.1$, and so on. Note that the random variables ξ_1, ξ_2, ξ_3 are not mutually independent.

Assume that an agent can choose actions $v_t \in \mathbb{R}$ at $t = 1, 2, 3$ (the notation v_t instead of u_t is justified in the sequel). The goal of the agent is the minimization of an expected sum of costs $\mathbb{E}\{\sum_{t=1}^{3} c_t(v_t, x_{t+1}) \mid x_1 = 0\}$. Here $x_t \in \mathbb{R}$ is the state of a continuous-state, discrete-time dynamical system, that starts from the initial state $x_1 = 0$ and follows the state transition equation $x_{t+1} = x_t + v_t + \xi_t$. Costs $c_t(v_t, x_{t+1})$, associated to the decision v_t and the transition to the state x_{t+1}, are defined by $c_t = (d_{t+1} + v_t^2/4)$ with $d_{t+1} = |x_{t+1} - \alpha_{t+1}|$ and $\alpha_2 = 2.9$, $\alpha_3 = 0$, $\alpha_4 = 0$ (α_{t+1}: nominal trajectory; d_{t+1}: tracking error; $v_t^2/4$: penalization of control effort).

An optimal policy mapping observations $\xi_1, \dots, \xi_{t-1}$ to decisions v_t can be obtained by solving the following convex quadratic program over variables $v_t^k, x_{t+1}^k, d_{t+1}^k$, where k runs from 1 to 7 and t from 1 to 3, and over x_1^k trivially set to 0:

$$\begin{array}{ll}
\text{minimize} & \sum_{k=1}^{7} p^k \left[\sum_{t=1}^{3} (d_{t+1}^k + (v_t^k)^2/4)\right] \\
\text{subject to} & -d_{t+1}^k \le x_{t+1}^k - \alpha_{t+1} \le d_{t+1}^k \quad \forall\, k, t \\
& x_1^k = 0 \ , \quad x_{t+1}^k = x_t^k + v_t^k + \xi_t^k \quad \forall\, k, t \\
& v_1^1 = v_1^2 = v_1^3 = v_1^4 = v_1^5 = v_1^6 = v_1^7 \\
& v_2^1 = v_2^2 = v_2^3 \ , \quad v_2^4 = v_2^5 = v_2^6 = v_2^7 \\
& v_3^2 = v_3^3 \ , \quad v_3^5 = v_3^6 \ .
\end{array}$$

Here, the vector of optimization variables (v_1^k, x_1^k) plays the role of u_1^k, the vector (v_t^k, x_t^k, d_t^k) plays the role of u_t^k for $t = 2, 3$, and the vector (x_4^k, d_4^k) plays the role of u_4^k, showing that the decision process $u_1, \dots, u_{T+1}$ of the general multistage stochastic programming decision model can in fact include state variables and more generally any element that serves to evaluate costs conveniently.

The optimal objective value is $+7.3148$, and the optimal solution is depicted on Figure 2.2. In this example, the final solution can be recast as a mapping $\tilde{\pi}_t$ from x_t to v_t: $\tilde{\pi}_1(0) = -0.1$, $\tilde{\pi}_2(-4.1) = 2.1$, $\tilde{\pi}_2(2.9) = -1.16$, $\tilde{\pi}_3(-5) = 2$, $\tilde{\pi}_3(-1.26) = 1.26$, $\tilde{\pi}_3(0) = 0.667$, $\tilde{\pi}_3(1.74) = -0.74$, $\tilde{\pi}_3(3.74) = -2$. Hence in this case the modeling assumption of an agent observing ξ_t instead of the system state x_t is not a fundamental restriction.

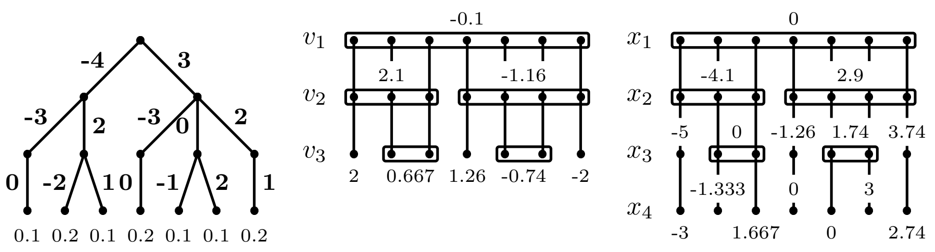

Fig. 2.2: (Left) Scenario tree representing the 7 possible scenarios for a random process $\xi = (\xi_0, \xi_1, \xi_2)$. The outcomes ξ_t^k are written in bold, and the scenario probabilities p^k are reported at the leaf nodes. (Middle) Optimal actions v_t for the agent. (Right) Visited states x_t under the optimal actions, treated as artificial decisions (see text).

2.2 *Comparison to Related Approaches*

This section discusses several modeling and algorithmic complexity issues raised by the multistage stochastic programming framework and scenario-tree based decision making.

2.2.1 *The Exogenous Nature of the Random Process*

A frequent assumption made in the stochastic programming framework is that decision makers do not influence by their decisions the realization of the random process representing the uncertainty. The random process is said to be *exogenous.* This allows to simulate, select and organize in advance possible realizations of the exogenous process, before any observation is actually made, and then optimize jointly (by opposition to individually for each scenario) the decisions contingent to the possible realizations.

The need to decouple the description of uncertainties and the optimization of decisions might appear at first as a strong limitation on the situations that can be modeled and treated by stochastic programming techniques. This impression is in part justified for a large family of problems of control theory in which the uncertainty is identified to some zero-mean noise perturbing the observations or the dynamics of the system, or when the uncertainty is understood as the uncertainty on the value of system parameters. But in another large family of sequential decision making problems under uncertainty, major sources of uncertainty are precisely the ones that are the less influenced by the behavior of the decision makers. We also note that random processes strongly influenced by the behavior of the decision makers can sometimes be handled by incorporating them to the initial decision process and treating them as a virtual decision process.

A probabilistic reasoning based on a subset of possible of scenarios could easily be tricked by an adversarial random process that would exploit one of the scenarios discarded during the planning process. In many practical problems however, the environment is not totally adversarial. In situations where the environment is mildly adversarial, it is often possible to choose measures of performances that are more robust to bad outcomes, and that can still be optimized in a tractable way.

Finally, it is easier in terms of sample complexity to learn a model (find model parameters from finite data sets) for an exogenous process than for an endogenous process. Learning a model for an exogenous process is possible from observations of the process, such as time series, whereas learning a model for an endogenous process forces us to be able to simulate possible state transitions for every possible action, or at least to have at one's disposal a fairly exhaustive data set relating actions to state transitions.

2.2.2 Comparison to Markov Decision Processes

In Markov Decision Processes (MDP) (Bellman, 1954; Howard, 1960), the decision maker seeks to optimize a performance criterion decomposed into a sum of instantaneous rewards. The information state of the decision maker at time t coincides with the state x_t of a dynamical system For simplicity, we do not consider in this discussion partial observability (POMDP) or risk-sensitivity, for which the system state need not be the information state of the agent. Optimal decision policies are often found by a reasoning based on the dynamic programming principle, to which is essential the notion of state as a sufficient statistic for representing the complete history of the system's evolution and agent's beliefs.

Multistage stochastic programming problems could be viewed as a subclass of finite-horizon Markov Decision Processes, by identifying the growing history of observations $(\xi_1, \dots, \xi_{t-1})$ to the agent's state. However, the mathematical assumptions under the MDP and the stochastic programming formulations are in fact quite different. Complexity results suggest that the algorithmic resolution of MDPs is efficient when the decision space is finite and small (Littman et al., 1995; Rust, 1997; Mundhenk et al., 2000; Kearns et al., 2002), while for the scenario-tree based stochastic programming framework, the resolution is efficient when the optimization problem is convex — in particular the decision space is continuous — and the number of decision stages is small (Shapiro, 2006).

One of the main appeals of stochastic programming techniques is their ability to deal efficiently with high-dimensional continuous decision spaces structured by numerous constraints, and with sophisticated, non-memoryless random processes. At the same time, if stochastic programming models have traditionally been used for optimizing long-term decisions that are implemented once and have lasting consequences, for example in network capacity planning (Sen et al., 1994), they are now increasingly used in the context of near-optimal control strategies that Bertsekas (2005a) calls *limited-lookahead* strategies. In this usage, at each decision stage an updated model over the remaining planning horizon is rebuilt and optimized on the fly, from which only the first-stage decisions are actually implemented. Indeed, when a stochastic program is solved on a scenario tree, the initial search for a decision policy degenerates into the search for sequences of decisions relative to the scenarios covered by the tree. The first-stage decision does not depend on observations and can thus always be implemented on any new scenario, whereas the recourse decisions relative to any particular scenario in the tree could be infeasible on a new scenario, especially if the feasibility sets depend on the random process.

2.2.3 The Curse of Dimensionality

The *curse of dimensionality* is an algorithmic-complexity phenomenon by which computing optimal policies on higher dimensional input spaces requires an exponential growth of computational resources, leading to intractable problem formulations. In dynamic programming, the input space is the state space or a reduced parametrization of it. In practice the curse of dimensionality limits attempts to cover inputs to spaces embedded in $\mathbb{R}^d$ with d at most equal to 5-10.

Approximate Dynamic Programming methods (Bertsekas and Tsitsiklis, 1996; Bertsekas, 2005a; Powell, 2007) and Reinforcement Learning approaches (Sutton and Barto, 1998) help to mitigate the curse of dimensionality, for instance by attempting to focus on the part of the state space that is actually reached under an optimal policy. An exploratory phase may be added to the original dynamic programming solution strategy so as to discover the relevant part of the state space. Those approaches work well in several cases:

i. The structure of a near-optimal policy is already known. For example, policy search methods assume that a near-optimal policy can be parametrized a priori by a small number of parameters, and rely on the user's expertise to find such a parametrization.

ii. Value-function based methods assume that there is a finite set of actions (or policy parameters), given a priori, that are the elementary building blocks of a near-optimal policy, and that can be used to drive the exploratory phase. The value function represents or approximates the expected value-to-go from the current state, and can be used to rank candidate actions (or policy parameters).

iii. By the structure of the optimization problem, the decisions and the state space subregions identified as promising early in the exploratory phase are those that are actually relevant to a near-optimal policy. This ensures the success of optimistic exploratory strategies, that refine decisions within promising subregions.

Stochastic programming algorithms do not rely on the covering of the state space of dynamic programming. Instead, they rely on the covering of the random exogenous process, which needs not correspond to the complete state space (see how the auxiliary state x_t is treated in the example of the previous section). The complement of the state space and the decision space are "explored" during the optimization procedure itself. The success of the approach will thus depend on the tractability of the joint optimization in those spaces, and not on insights on the structure of near-optimal policies.

In multistage stochastic programming approaches, the curse of dimensionality is present when the number of decision stages increases, and in face of high-dimensional exogenous processes. Therefore, methods that one could call, by analogy to approximate dynamic programming, *approximate stochastic programming methods*, will attempt to cover only the realizations of the exogenous random process that are truly needed to obtain near-optimal decisions. These methods work with a number of scenarios that does not grow exponentially with the dimension of the exogenous process and the number of stages.

2.3 The Value of Multistage Stochastic Programming

Due to the curse of dimensionality, multistage stochastic programming is in competition with more tractable decision models. At the same time it provides a unifying framework between several simplified decision making paradigms, that we now describe.

2.3.1 Reduction to Model Predictive Control

A radical simplification consists in discarding the detailed probabilistic information on the uncertainty, taking a nominal scenario, and optimizing decisions on the nominal scenario. The common practice for defining a nominal scenario is to replace random variables by their expectation. The resulting problem is called the *expected value problem*, the solution of which constitutes a nominal plan. Even if the nominal plan could be used as an *open-loop decision policy*, that is, implemented over the complete planning horizon, decision makers will usually want to recompute the plan at the next decision stage by solving an updated expected value problem. In control theory, the approach is known as Model Predictive Control (MPC) (Bertsekas, 2005b).

An indicator of the value of multistage programming decisions over model predictive control decisions is given by the *value of the stochastic solution* (VSS). To make the notion precise, let us define successively:

- V^*, the optimal value of the multistage stochastic program $\min_\pi \mathbb{E}\{f(\xi, \pi(\xi))\}$. For notational simplicity, we adopt the convention that $f(\xi, \pi(\xi)) = \infty$ if the policy π is anticipative or yields infeasible decisions.
- $\zeta = (\zeta_1, \ldots, \zeta_T)$, the nominal scenario.
- u^ζ, the optimal solution to the expected value problem $\min_u f(\zeta, u)$. Note that the optimization is over a single fixed sequence of feasible decisions; the problem data is determined by ζ.
- u_1^ζ, the first-stage decision of u^ζ.
- V^ζ, the optimal value of the multistage stochastic program $\min_\pi \mathbb{E}\{f(\xi, \pi(\xi))\}$ subject to the additional constraint $\pi_1(\xi) = u_1^\zeta$ for all ξ. If by a slight abuse of notation, we write π_1, viewed as an optimization variable, for the value of the constant-valued function π_1, then the additional constraint is simply $\pi_1 = u_1^\zeta$. By definition, V^ζ is the value of a policy implementing the first decision from the expected value problem, and then selecting optimal recourse decisions for the subsequent decision stages. The recourse decisions differ in general from those that would be selected by a policy optimal for the original multistage program.

The VSS is then defined as the difference $V^\zeta - V^* \geq 0$. For maximization problems, it would be defined by $V^* - V^\zeta \geq 0$. Birge and Louveaux (1997) describe special cases (with two decision stages, and restrictions on the way randomness affects problem data) for which it is possible to compute bounds on the VSS. They also come to the conclusion, from their survey of works studying the VSS, that there is no rule that can predict a priori whether the VSS is low or high for a given problem instance — for example increasing the variance of random variables may increase or decrease the VSS.

2.3.2 Reduction to Two-Stage Stochastic Programming

A less radical simplification consists in discarding the distinction between recourse stages, keeping in the model a first stage (associated to full uncertainty) and a second stage (associated to full knowledge). A multistage model degenerates into a two-stage model when the scenario tree has branchings only at one stage. The situation arises for instance when scenarios are sampled over the full horizon independently: the tree has then branchings only at the root. In Huang and Ahmed (2009), the *value of multistage stochastic programming* (VMS) is defined as the difference of the optimal values of the multistage model versus the two-stage model. The authors establish bounds on the VMS and describe an application (in the semiconductor industry) where the VMS is high. Note however that a generalization of the notion of VSS would rather quantify how multistage decisions outperform two-stage decisions when those two-stage decisions are implemented with model rebuilding at each stage, in the manner of the Model Predictive Control scheme.

2.3.3 Reduction to Heuristics based on Parametric Optimization

As an intermediate simplification between the expected value problem and the reduction to a two-stage model, it is possible to optimize sequences of decisions separately on each scenario. The decision maker can then use some averaging, consensus or selection strategy to implement a first-stage decision inferred from the so-obtained ensemble of first-stage decisions. Here again, the model should be rebuilt with updated scenarios at each decision stage.

The problem of computing optimal decisions separately for each scenario is known as the *distribution problem*. The problem appears in the definition of the *expected value of perfect information* (EVPI), which quantifies the additional value that a decision maker could reach in expectation if he or she were able to predict the future. To make the notion precise, let V^* denote as before the optimal value of the multistage stochastic program $\min_\pi \mathbb{E}\{f(\xi, \pi(\xi))\}$ over non-anticipative policies π; let $V(\xi)$ denote the optimal value of the deterministic program $\min_u f(\xi, u)$; and let V^A be the expected value of $V(\xi)$, according to the distribution of ξ. Observe that V^A is also the optimal value of the program $\min_{\pi^A} \mathbb{E}\{f(\xi, \pi^A(\xi))\}$ over anticipative policies π^A, the optimization of which is now decomposable among scenario subproblems. The EVPI is then defined as the difference $V^* - V^A \geq 0$. For maximization problems, it is defined by $V^A - V^* \geq 0$. Intuitively, the EVPI is high when having to delay adaptations to final outcomes due to a lack of information results in high costs.

The EVPI is usually interpreted as the price a decision maker would be ready to pay to know the future (Raiffa and Schlaifer, 1961; Birge, 1992). The EVPI also indicates how valuable the dependence of decision sequences is on the particular scenario they are optimized over. Mercier and Van Hentenryck (2007) show on an example with low EVPI how a strategy based on a particular aggregation of decisions optimized separately on deterministic scenarios can be arbitrarily bad. Thus even if the EVPI is low, heuristics based on the decisions of anticipative policies can perform poorly.

This does not mean that the approach cannot perform well in practice. Van Hentenryck and Bent (2006) have studied and refined various aggregation and regret-minimization strategies on a series of stochastic combinatorial problems already hard to solve on a single scenario, as well as schemes that build a bank of pre-computed reference solutions and then adapt them online to accelerate the optimization on new scenarios. They show that their strategies perform well on vehicle routing applications.

Remark 2.1. The progressive hedging algorithm (Rockafellar and Wets, 1991) is a decomposition method that computes the solution to a multistage stochastic program on a scenario tree by solving repeatedly separate subproblems on the scenarios covered by the tree. First-stage decisions and other decisions coupled by non-anticipativity constraints are obtained by aggregating the decisions of the concerned scenarios, in the spirit of the heuristics based on the distribution problem presented above. The algorithm modifies the scenario subproblems at each iteration to make the decisions coupled by non-anticipativity constraints converge towards a common and optimal decision.

As the iterations are carried out, first-stage decisions evolve from decisions hedged by the aggregation strategy to decisions hedged by the multiple recourse decisions computed on the scenario tree. Therefore, the progressive hedging algorithm shows that there can be a smooth conceptual transition between the decision model based on the distribution problem and the decision model based on the multistage stochastic programming problem. □

Example 2.1. We illustrate the computation of the VSS and the EVPI on an artificial multistage problem, with numerical parameters chosen in such a way that the full multistage model is valuable. By valuable we mean that the presented simplified decision-making schemes will output first-stage decisions that are suboptimal. If those decisions were implemented, and subsequently the best possible recourse decisions were applied, the value of the objective over the full horizon would be significantly suboptimal.

Let w_1, w_2, w_3 be mutually independent random variables uniformly distributed on $\{+1, -1\}$. Let $\xi = (\xi_1, \xi_2, \xi_3)$ be a random walk such that $\xi_1 = w_1$, $\xi_2 = w_1 + w_2$, $\xi_3 = w_1 + w_2 + w_3$. Let the 8 equiprobable outcomes of ξ form a scenario tree and induce non-anticipativity constraints (the tree is a binary tree of depth 3). Consider the decision process $u = (u_1, u_2, u_3)$ with $u_2 \in \mathbb{R}$ and $u_t = (u_{t1}, u_{t2}) \in \mathbb{R}^2$ for $t = 1, 3$. Then consider the multistage stochastic program

$$\begin{aligned}
&\text{maximize}\\
&\quad \tfrac{1}{8}\textstyle\sum_{k=1}^{8}\{[0.8u_{11}^k - 0.4(u_2^k/2 + u_{31}^k - \xi_3^k)^2]\\
&\qquad\quad + u_{32}^k\xi_3^k + [1 - u_{11}^k - u_{12}^k]\}\\
&\text{subject to}\\
&\quad u_{11}^k + u_{12}^k \le 1 \quad \forall k\\
&\quad - u_{11}^k \le u_2^k \le u_{11}^k \quad \forall k\\
&\quad - u_{1j}^k \le u_{3j}^k \le u_{1j}^k \quad \forall k \text{ and } j = 1, 2\\
&\quad \text{C1: } u_1^k = u_1^1 \quad \forall k\\
&\quad \text{C2: } u_2^k = u_2^{k+1} = u_2^{k+2} = u_2^{k+3} \quad \text{for } k = 1, 5\\
&\quad \text{C3: } u_3^k = u_3^{k+1} \quad \text{for } k = 1, 3, 5, 7\ .
\end{aligned}$$

The non-anticipativity constraints C1, C2, C3, which are convenient to state the problem, indicate in practice the redundant optimization variables that can be eliminated.

- The optimal value of the multistage stochastic program is $V^* = 0.35$ with optimal first-stage decision $u_1^* = (1, 0)$.
- The *expected value problem* for the mean scenario $\zeta = (0, 0, 0)$ is obtained by setting momentarily $\xi^k = \zeta$ for all k and adding the constraints

$$\begin{aligned} &\text{C2'}:\ u_2^k = u_2^1 \quad \text{for all } k\ , \\ &\text{C3'}:\ u_3^k = u_3^1 \quad \text{for all } k\ . \end{aligned}$$

  Its optimal value is 1 with first-stage decision $u_1^\zeta = (0, 0)$. When equality constraints are made implicit the problem can be formulated using 5 scalar optimization variables only.
- The *two-stage relaxation* is obtained by relaxing the constraints C2, C3. Its optimal value is 0.6361 with $u_1^k \overset{\text{def}}{=} u_1^{\text{II}} = (0.6111, 0.3889)$.
- The *distribution problem* is obtained by relaxing the constraints C1, C2, C3. Its optimal value is $V^A = 0.6444$. The two extreme scenarios $\xi^1 = (1, 2, 3)$ and $\xi^8 = (-1, -2, -3)$ have first-stage decisions $u_1^1 = u_1^8 = (0.7778, 0.2222)$ and value -0.0556. The 6 other scenarios have $u_1^k = (0.5556, 0.3578)$ and value 0.8778, $k = 2, \dots, 7$. Note that in general, (i) scenarios with the same optimal first-stage decision and values may still have different recourse decisions, and (ii) the first-stage decisions can be distinct for all scenarios.
- The EVPI is equal to $V^A - V^* = 0.2944$.
- Solving the multistage stochastic program with the additional constraint

$$\text{C1}^\zeta : u_1^k = u_1^\zeta\ \forall k$$

  yields an upper bound on the optimal value of any scheme using the first-stage decision of the expected value problem. This value is $V^\zeta = -0.2000$.
- The VSS is equal to $V^* - V^\zeta = 0.55$.
- Solving the multistage stochastic program with the additional constraint

$$\text{C1}^{\text{II}}\text{:}\ u_1^k = u_1^{\text{II}}\ \forall k$$

  yields an upper bound on the optimal value of any scheme using the first-stage decision of the two-stage relaxation model. This value is $V^{\text{II}} = 0.2431$. Thus, the value of the multistage model over a two-stage model, in our sense (distinct from the VMS of Huang and Ahmed (2009)), is at least $V^* - V^{\text{II}}$=0.1069.

To summarize, observe the collapse of the optimal value from $V^* = 0.35$ to $V^{\text{II}} = 0.2431$ (with the first-stage decision of the two-stage model) and then to $V^\zeta = -0.2$ (with the first-stage decision of the expected value model).

We can also consider the anticipative decision sequences of the distribution problem, and check if there exist plausible strategies that could exploit the set of first-stage decisions to output a good first-stage decision (with respect to any decision-making scheme for the subsequent stages).

- Selection strategy: Solving the multistage stochastic program with a constraint that enforces one of the first-stage decisions extracted from the distribution problem yields the following results: optimal value 0.3056 if $u_1^k =$

  $(0.7778, 0.2222)$, optimal value 0.2167 if $u_1^k = (0.5556, 0.3578)$. But one has to concede that in contrast to other simplified models, for which we solve multistage programs only to measure the quality of a suboptimal first-stage decision, the selection strategy needs good estimates of the different optimal values to actually output the best decision.

- Consensus strategy: The outcome of a majority vote out of the set of the 8 first-stage decisions would be the decision $(0.5556, 0.3578)$ associated to the scenarios 2 to 7. With value 0.2167, this turns out to be the worst decision between $(0.7778, 0.2222)$ and $(0.5556, 0.3578)$.
- Averaging strategy: The mean first-stage decision of the set of 8 first-stage decisions is $\bar{u}_1 = (0.6111, 0.3239)$. Solving the multistage program with $u_1^k = \bar{u}_1$ for all k yields the optimal value 0.2431.

The best result is the value 0.3056 obtained by the selection strategy. Note that we are here in a situation where the multistage program and its variants could be solved exactly, that is, with a scenario tree representing the possible outcomes of the random process exactly.

□

2.4 Practical Scenario-Tree Approaches

We now focus on a practical question essential to the deployment of a multistage stochastic programming model: if a problem has to be approximately represented by a scenario tree in order to compute a decision strategy, how should a tractable and at the same time representative scenario-tree approximation be selected for a given problem?

After some background on discretization methods for two-stage stochastic programming, we pose the scenario tree building problem in an abstract way and then discuss the antagonist requirements that make its solution very challenging. Then we review the main families of methods proposed in the literature to build tractable scenario-tree approximations for a given problem, and highlight their main properties from a theoretical point of view.

Given the difficulty of determining a priori good scenario-tree approximations for many problems of practical interest (a difficulty which is to some extent surprising, given the practical success of related approximation methods for two-stage stochastic programming), there is a growing consensus on the necessity of being able to test a posteriori the quality of scenario-tree based approximations on a large independent sample of new scenarios. We present in this light a standard strategy based on the so-called shrinking-horizon approach — the term is used, for instance, in Balasubramanian and Grossmann (2003).

2.4.1 Approximation Methods in Two-stage Stochastic Programming

Let $\mathcal{P}$ denote a two-stage stochastic program, where the uncertainty is modeled by a random vector ξ, possibly of high-dimension, following a certain distribution with either a discrete support of large cardinality, or a continuous support. Let $\mathcal{P}'$ be an approximation to $\mathcal{P}$, where ξ is approximated by a random vector ξ' that follows a distribution with a finite discrete support, the cardinality of the support being limited by the fact that to each possible realization of ξ' is associated a number of optimization variables for representing the corresponding recourse decisions. To obtain a good approximation, one would ideally target the problem of finding a finite discrete distribution for ξ' (values for the support and associated probability masses) such that any first-stage decision u_1' optimal for $\mathcal{P}'$ yields on $\mathcal{P}$ a minimal regret, in the sense that with optimal recourse decisions, the value on $\mathcal{P}$ of the solution made of u_1' and of optimal recourse decisions is close to the exact optimal value of $\mathcal{P}$. By analogy to the VSS, we could also say that the distribution for ξ' should minimize the value of the exact program $\mathcal{P}$ with respect to the approximate program $\mathcal{P}'$.

Many authors have found it more convenient to restrict the attention on the problem of finding a finite discrete distribution for ξ' such that the optimal value of $\mathcal{P}'$ is close to the optimal value of $\mathcal{P}$, and the solutions u_1' optimal for $\mathcal{P}'$ are close to solutions optimal for $\mathcal{P}$. For this approach to work, one might want to impose some weak form of continuity of the objective of $\mathcal{P}$ with respect to solutions. One may also want to ensure that small perturbations of the probability measure for ξ have a bounded effect on the perturbation of optimal solutions u_1.

An interesting deterministic approach (Rachev and Römisch, 2002) consists in analyzing the structure of optimal policies for a given problem class, the structure of the objective when the optimal policy is implicitly taken into account, and inferring from it a relevant measure of distance between probability distributions. The distance is based on the worst-case difference among objectives taken from a class of functions with the identified structure (or from a larger class of functions if this is technically convenient for the computation of the distance measure). Finding a good approximation to a two-stage stochastic program is then reformulated as the problem of finding a discrete distribution minimizing the relevant probability distance to the original distribution. Note that probability distance minimization problems can be difficult to solve, especially with high-dimensional distributions. Thus, the approach, which reformulates the approximation problem as the optimal quantization of the initial probability distribution, can have essentially two sources of suboptimality with respect to the ideal approximation problem: (i) in order to get a tractable computation of the distance, the class of functions over which worst-case distances are evaluated has often to be enlarged; (ii) even on the enlarged class, the probability distance minimization problem is often difficult to solve (for instance NP-hard), so that the minimal distance is not necessarily attained, especially when heuristics for finding solutions are used. Despite these limitations, the approach has been shown to work well. Moreover, the reduction of the initial approximation problem to an optimal quantization problem indicates the relevance of existing work on vector quantization and probability density estimation (MacKay, 2003, Chapter 20), and on discretization methods explored in approximate dynamic programming.

Randomized approaches are based on Monte Carlo sampling (Metropolis and Ulam, 1949) and its many extensions, including variance reduction techniques, and quasi Monte Carlo techniques (MacKay et al., 1979). All these techniques have more or less been tried for solving two-stage stochastic programs: Infanger (1992), for instance, investigates importance sampling. They have been shown to work well in practice. Random approximations based on Monte Carlo have been shown to be consistent, in the sense that with an infinite number of samples, optimal solutions to discretized programs can converge to solutions optimal for $\mathcal{P}$. More detailed results can be found in Shapiro (2003b).

2.4.2 Challenges in the Generation of Scenario Trees

In two-stage stochastic programming, the large or infinite set of recourse decisions of the original program is reduced to a finite set of recourse decisions for the approximation. Hence the exact and approximate solutions lie in different spaces and cannot be compared directly. Still, recourse decisions can be treated implicitly, as if they were already incorporated to the objective function, and as if the only remaining element to optimize were the first-stage decision.

In multistage stochastic programming, we face the same issue: one cannot directly compare finite-dimensional solutions obtained from finite scenario-tree approximations to exact optimal solutions lying in a space of functions. But now, using the same technique of treating all recourse decisions implicitly leads to a dilution of the structural properties of the objective function. As these structural properties are weaker, the class of objective functions to consider becomes very general. Worst-case distances between functions in such classes may cease to guide satisfactorily a discretization procedure. In addition, as the random process runs over several stages, the discretization problems are posed over typically larger spaces, making them more difficult to solve, even approximately.

For these reasons, rather than presenting the generation of scenario trees as a natural extension of discretization methods for two-stage stochastic programming, with the incorporation of branchings for representing the nested conditional probability densities, we state the problem in a more open way, which also highlights complexity aspects:

Construct a tractable algorithm $\mathcal{A}$ that,

- *given a multistage stochastic program $\mathcal{P}$: $\min_\pi \mathbb{E}\{f(\xi, \pi(\xi))\}$ defined over a probability space $(\Omega, \mathcal{B}, \mathbb{P})$ with objective f (including by convention the constraints) and non-anticipative policies $\pi(\xi)$,*
- *will produce an approximate finite-dimensional surrogate program of the form $\mathcal{P}'$: $\min_u \sum_{k=1}^n p^k \{g(\xi^k, u^k)\}$ defined over some reduced space $(\Omega', \mathcal{B}', \mathbb{P}')$ and objective g, and from which a surrogate policy $\hat{\pi}(\xi)$ subject to non-anticipativity constraints may be computed in a tractable way,*
- *with the goal of making the regret*

$$\mathcal{R} = \mathbb{E}\{f(\xi, \hat{\pi}(\xi))\} - \min_\pi \mathbb{E}\{f(\xi, \pi(\xi))\} \geq 0$$

  as small as possible.

Notice that we allow, for the sake of generality, that the surrogate program may refer to a function g different from the original objective f, and that we impose that the algorithm $\mathcal{A}$, the solving strategy associated to the problem $\mathcal{P}'$, as well as the evaluation of the induced policy $\hat{\pi}$ on any new scenario, are all tractable. At this stage, we do not specify how $\hat{\pi}$ is inferred or understood; $\hat{\pi}$ needs to be introduced here only to be able to write a valid expression for the regret on the original multistage program.

Depending on situations, the problem $\mathcal{P}$ (random process model and function f) can be described analytically, or be only accessible through sampling and/or simulation. The problem $\mathcal{P}'$ will be described by a scenario tree and the choice of the function g, under limitations intrinsically due to the tractability of the optimization of the approximate program.

As we have seen, there are many derived decision-making schemes and usages of the multistage stochastic programming framework. Also, various classes of optimization programs can be distinguished — with the main distinctions being between two-stage and multistage settings, and among linear, convex, and integer/mixed-integer formulations — and thus several possible families of functions over which one might attempt to minimize a worst-case regret.

In the stochastic programming literature, several scenario tree generation strategies have been studied. The scenario tree generation problem is there often viewed in one or another of two reduced ways with respect to the above definition, namely

(i) as the problem of finding a scenario tree with an associated optimal value

$$\min_u \sum_{k=1}^{n} p^k \{f(\xi^k, u^k)\}$$

close to the exact optimal value $\min_\pi \mathbb{E}\{f(\xi, \pi(\xi))\}$, or

(ii) as the problem of finding a scenario tree with its associated optimal first-stage decision $\hat{u}_1$ close to a first-stage decision π_1 optimal for the exact program.

Indeed, version (i) is useful when the goal is merely to estimate the optimal value of the original program $\mathcal{P}$, while version (ii) is useful when the goal is to extract only the first stage decision, assuming that later on, recourse decisions are recomputed using a similar algorithm, given the new observations.

The generic approximation problem that we have described is more general, since it covers also the case where the scenario tree approach may be exploited offline to extract a complete policy $\hat{\pi}(\xi)$ that may then be used later on, in a stand-alone fashion for decision making over arbitrary scenarios and decision steps, be it in the real world or in the context of Monte Carlo simulations.

To give an idea of theoretical results established in the scenario tree generation literature, we now briefly discuss two representative trends of research: works that study Monte Carlo methods for building the tree, and works that seek to minimize in a deterministic fashion a certain measure of discrepancy between the original process and the approximate process represented by the scenario tree.

Monte Carlo Scenario Tree Sampling Methods.

Monte Carlo methods have several advantages: they are easy to implement and they scale well with the dimension, in the sense that with enough samples, one can get close to the statistical properties of high-dimensional target distributions with high probability. The major drawback of (pure) Monte Carlo methods is the variance of the results (in our case, the optimal value and optimal solutions of the approximate programs) in small-sample conditions.

Let us describe the Sample Average Approximation method (SAA) (Shapiro, 2003b), which uses Monte Carlo for generating the scenario tree. One starts by building the branching structure of the tree. Note that the method does not specify how to carry out that step. Practitioners often use the same branching factor for each node relative to a given decision stage. They also often concentrate the branchings at early stages: the branching factor is high at the root node and then decreases with the index of the decision stage. The next step of the method consists in sampling the node values according to the distributions conditioned on the values of the ancestor nodes. The procedure, referred to as *conditional sampling*, is implemented by sampling the realizations of random variables at stage t before sampling those of stage $t+1$. Distinct realizations are assigned to distinct nodes, which are given a conditional probability equal to the inverse of the branching factor. The last step consists in solving the program on the so-obtained scenario tree and thus, although part of the description of the SAA method, does not concern the generation of the tree itself.

Consider scenario trees obtained by conditional sampling. For simplicity assume a uniform branching factor n_t at each stage t, so that the number of scenarios is $n = \prod_{t=1}^{T} n_t$. Shapiro (2006) shows under some technical assumptions that if we want to guarantee, with a probability at least $1 - \alpha$, that implementing the first-stage decision $\hat{u}_1$ optimized on a scenario tree of size n while implementing subsequently optimal recourse decisions conditionally to the first-stage decision will yield an objective value ϵ-close to the exact optimal value, then the size n of the tree we use for that purpose has to grow exponentially with the number of stages. The result goes against the intuition that by asking for ϵ-optimality with probability $1 - \alpha$ only, one could get moderate sample complexity requirements. Now, as the exponential growth of the number of scenarios is not sustainable, one can only hope solving multistage models in small-sample conditions, and obtain solutions that at least with the SAA method may vary from tree to tree and be of uncertain value for the real problem. Perhaps surprisingly, it is not possible to obtain valid statistical bounds for that uncertain value by imposing as first-stage decision the tested first-stage decision and reoptimizing recourse decisions on several new random trees (Shapiro, 2003a).

Deterministic Scenario Tree Optimization Methods.

There exist various deterministic techniques for selecting jointly the scenarios of the tree. Note that there is a part of numerical experimentation in the development of scenario tree methods, and a risk of overestimating the domain of validity of the proposed methods, since research efforts are oriented by experiments on particular problems.

Moment-matching methods (Høyland and Wallace, 2001; Høyland et al., 2003) attempt to produce discrete distributions with some statistical moments matching those of a target distribution. Moment matching may be done at the expense of other statistics, such as the number and the location of the modes, that might also be important. Hochreiter and Pflug (2007) give an example illustrating that risk.

The theoretical analysis underlying the so-called *probability metrics* methods, that we have briefly evoked in the context of two-stage stochastic programming, was initially believed to be easily extensible to the multistage case (Heitsch and Römisch, 2003); but then it turned out that more elaborated measures of probability distances, integrating the intertemporal aspect of observations, were needed (Heitsch and Römisch, 2009). These elaborated metrics are more difficult to compute and to minimize, so that well-justified discretizations of multistage programs are more difficult to obtain.

We can also mention methods that come with approximation guarantees, such as bounds on the suboptimality of the approximation (Frauendorfer, 1996; Kuhn, 2005). However, they are applicable only under relatively strong assumptions concerning the problem class and the type of randomness. Quasi Monte Carlo techniques are perhaps among the more generally applicable methods (Pennanen, 2009).

Most deterministic methods end up with the formulation of difficult optimization problems, such as nonconvex or NP-hard problems (Høyland et al., 2003; Hochreiter and Pflug, 2007), with computationally demanding tasks (such as multidimensional integrations), especially for high-dimensional random processes.

The field is still in a state where the scope of existing methods is not well defined, and where the algorithmic description of the methods is incomplete, especially concerning the branching structure of the trees. That the domains of applicability are not known or overestimated makes it delicate to select a sophisticated deterministic technique for building a scenario tree on a new problem.

2.4.3 The Need for Testing Scenario-Tree Approximations

Theoretical analyses of scenario tree generation algorithms, often based on worst-case reasonings or large deviation theory, provide guarantees on the quality of approximate solutions that are usually too loose in practice or equivalently call for intractable scenario tree sizes. Hence they do not really solve the basic question of how to build a priori small scenario trees in a generic, scalable, and computationally efficient way, potentially jeopardizing the practical relevance of the multistage extension of stochastic programming for sequential decision making under uncertainty. Now if we are ready to renounce to worst-case guarantees embedded in the scenario tree generation method, new tools are needed for computing, a posteriori, guarantees on the value of a given numerical approximation scheme.

If we want to assess on an independent test set of scenarios the performance of decisions optimized on a scenario tree, a difficulty arises: first-stage decisions can be tested but subsequent recourse decisions are only defined for the scenarios covered by the scenario tree. Therefore, it is necessary to extend the approach so as to allow one to test solutions on new scenarios, at a computational cost low enough to allow the validation on a sufficiently large number of test scenarios.

We have to stress that this extension is not really necessary for two-stage stochastic programming. First, approximations of two-stage models yield constant first-stage decisions, that are implementable on any scenario, while recourse decisions on new scenarios can then often be found analytically, or by running a myopic one-stage optimization procedure for each new scenario, or by implementing a known recourse procedure that the initial two-stage model was only approximating for optimizing the first-stage decisions — a strategy found efficient in capacity planning (Sen et al., 1994). Thus, testing is generally straightforward for two-stage models. Second, finite-dimensional approximations of two-stage stochastic programming models do not use scenario trees. They only use a finite set of outcomes. Theoretical results show that in the two-stage situation, statistical confidence bounds on the quality of an approximate solution can be computed (Norkin et al., 1998b; Mak et al., 1999). These results break down in the multistage case, giving its true interest to guarantees based on testing (Shapiro, 2003a).

2.4.4 The Inference of Shrinking-Horizon Decision Policies

Several authors have proposed to use a generic scheme similar to Model Predictive Control to assess the performances associated to a particular algorithm $\mathcal{A}$ for building the scenario tree (Kouwenberg, 2001; Chiralaksanakul, 2003; Hilli and Pennanen, 2008). The scheme can be sketched as follows.

i. Generate a scenario tree using algorithm $\mathcal{A}$. Solve the resulting program and extract from its solution the value of the first-stage decision u_1, say $\bar{u}_1$.

ii. Generate a test sample, of n'' mutually independent scenarios by drawing i.i.d. realizations ξ^j of the random process ξ.

iii. For each scenario ξ^j of the test sample, set $u_1^j = \bar{u}_1$, and obtain sequentially the recourse decisions $u_2^j, \dots, u_T^j$, as follows: each decision u_t^j is obtained as a first-stage decision computed by taking as an initial condition the past decisions $u_1^j, \dots, u_{t-1}^j$ and the history $\xi_1^j, \dots, \xi_{t-1}^j$ of the test scenario ξ^j, by conditioning the joint distribution of $\xi_t, \dots, \xi_T$ on the history, by using the algorithm $\mathcal{A}$ to build a new scenario tree that approximates the random process $\xi_t, \dots, \xi_T$, by solving the program formulated on this tree over the optimization variables relative to the decisions $u_t, \dots, u_T$, and by discarding all but the decision u_t, the optimal value of which is then assigned to u_t^j.

iv. Estimate the overall performance of the scheme by Monte Carlo simulation. This consists in evaluating on the test sample the empirical average

$$V_{\mathrm{TS}}(\mathcal{A}) = (1/n'') \textstyle\sum_{j=1}^{n''} f(\xi^j, u^j) \ ,$$

where we have denoted by $u^j = (u_1^j, \dots, u_T^j)$ the sequence of decisions associated to the scenario ξ^j, and where TS recalls that the estimate is computed on the test sample.

The Monte Carlo estimate $V_{\mathrm{TS}}(\mathcal{A})$ can provide an unbiased estimation of the value of the scenario tree building algorithm $\mathcal{A}$ in the context of the other approximations involved in the numerical computations of the sequences of decisions, such as for instance simplifications of the objective function, or early stopping at low-accuracy solutions.

The estimator $V_{\mathrm{TS}}(\mathcal{A})$ may have a high variance, but we can expect a high positive correlation between this estimator and an estimator $V_{\mathrm{TS}}(\mathcal{A}')$ using the *same* test sample but relative to another tree generation algorithm $\mathcal{A}'$. This would allow a reliable comparison of the relative performance of the two algorithms $\mathcal{A}$, $\mathcal{A}'$ on the problem instance at hand.

The validation is generic in the sense that it can be applied to any algorithm $\mathcal{A}$, but also in the sense that it addresses the general scenario tree building problem in the larger context of the decision making scheme actually implemented in practice.

2.4.5 Alternatives to the Scenario Tree Approach

We point out that alternative numerical methods for solving infinite-dimensional two-stage stochastic programs exist, based on an incorporation of the discretization procedure to the optimization, for instance by updating the discretization or carrying out importance sampling within the iterations of a given optimization algorithm (Slyke and Wets, 1969; Higle and Sen, 1991; Norkin et al., 1998a), or by using stochastic subgradient methods (Nemirovski et al., 2009). Also, heuristics for finding good policies directly on the infinite-dimensional multistage problem have been suggested: a possible idea, akin to direct policy search procedures in Markov Decision Processes, is to optimize a combination of feasible non-anticipative basis policies $\pi^j(\xi)$ specified beforehand (Koivu and Pennanen, 2010). These methods are nevertheless less general than the standard scenario tree approach, because they seem to be reserved to applications with rather simple feasibility sets.

2.5 Conclusions

Elaborating strategies involving complex quantitative decisions calls for optimization techniques that can avoid a systematic enumeration and evaluation of possible options at each decision stage. Multistage stochastic programming has been recognized by several industries (Dempster et al., 2008; Kallrath et al., 2009) as a promising framework to formulate complex problems under uncertainty, exploit domain knowledge, use risk-averse objectives, incorporate probabilistic and dynamical aspects, and preserve structures that allow to apply large-scale optimization techniques. These techniques for sequential decision making under uncertainty are particularly interesting to study: Puterman (1994), citing Arrow (1958) on the early roots of sequential decision processes, recalls the role of the multi-period inventory models from the industry in the development of the theory of Markov Decision Processes (Bellman, 1954; Howard, 1960). We could also mention the role of applications in finance as a motivation for the early theory of multi-armed bandits and for the theory of sequential prediction (Cesa-Bianchi and Lugosi, 2006), now an important field of research in machine learning (Auer et al., 2002; Coquelin and Munos, 2007).

In the preceding sections, where the problem of inferring a good scenario-tree approximation for a given multistage stochastic programming problem was presented, we have seen that the state of the art does not currently provide strong enough methods with broad enough practical coverage and good enough theoretical guarantees in terms of the quality of the approximate solutions derived in this way.

Researchers in the field were thus led to suggest the use of the shrinking-horizon recursive procedure for exploiting the scenario tree based approach in practice. However, evaluating the resulting performance estimator on an independent sample of scenarios is extremely demanding, as it requires, for each test scenario and at each stage of recourse decisions, the automatic construction of a new scenario tree and the optimization of the resulting program on the tree. Doing this is still beyond the possibility of available computational approaches when considering the solution of large-scale problems.

For these reasons, there is currently no scalable off-the-shelf method for generating and testing scenario-tree based approximations of multistage stochastic programs, and the framework of stochastic programming based on scenario trees has in this way, in spite of its theoretical appeal, lost its practical attractiveness during the last years in many environments dealing with large-scale systems (Powell and Topaloglu, 2003; Van Hentenryck and Bent, 2006).

Chapter 3

Solution Averaging in Structured Spaces

In this chapter, we consider an extension to the multistage stochastic programming framework, based on the simultaneous use of several scenario trees for making decisions.

This work is motivated by the excellent performances of the perturb-and-combine estimation methods from machine learning (P&C) (Breiman, 1998).

In fact, stochastic programs are usually formulated in a way very similar to maximum likelihood estimation problems (Dupacova and Wets, 1988). Yet, since Fisher's contributions to maximum likelihood estimation, considerable methodological advances have been made on the problem of estimating a parameter of interest with a finite amount of data, through a mix of optimization, resampling and averaging techniques.

The chapter is organized as follows. Section 3.1 provides an unified view to a series of estimation methods used in machine learning, that have finally led to ensemble methods. Section 3.2 proposes an approach that seeks to better estimate the first-stage decision of a multistage program by aggregating several first-stage decisions optimal with respect to different scenario-tree approximations. Section 3.3 illustrates and evaluates the proposed approach on a test problem having an interesting discrete decision space. Section 3.4 concludes the chapter with some ideas concerning the regularization of stochastic programming approximations.

3.1 The Perturb-and-Combine Paradigm

Let X be a random vector following some fixed but unknown density $\mathbb{P}_X$, referred to as the data-generating density.

Let $D = \{x_1, \dots, x_n\}$ denote a set of realizations of X drawn from $\mathbb{P}_X$ in some way. Call D the data set. For brevity we write x^n for $x_1, \dots, x_n$. A data set of n samples is a random quantity. Its density is written $\mathbb{P}_D$. There is a wide spectrum of methods from statistics and machine learning that can be used to explain the data, and predict (forecast) future samples. We discuss those methods in the context of the inference of a predictive density $p_{x|D}$ for a new sample x, given the data. One could also condition the density of $x \overset{\text{def}}{=} (y, z)$ on some of its components y and interpret the resulting density $p_{z|y,D}$ as the predictive distribution of output variables z given input variables y and data set D.

The later use of densities is not discussed in this section. We simply recall that the summary of a density through a single value is addressed by decision theory, and is usually done through the choice of a loss function (Robert, 2007, Chapter 2). The quality of the inference can also be quantified through a measure of divergence between the predicted and true densities (Ali and Silvey, 1966; Csizár, 1967). A particular divergence that has been found useful (Clarke and Barron, 1990) is the Kullback-Leibler divergence between two densities g, h, defined by

$$\mathrm{KL}(g||h) = \int g(x) \log \frac{g(x)}{h(x)} \mathrm{d}x \ . \tag{3.1}$$

The KL divergence is also referred to as the cross-entropy distance (Rubinstein and Kroese, 2004).

3.1.1 Maximum Likelihood Estimation

In the simplest frequentist approach to explaining data, one assumes that the samples are drawn independently, and that the data-generating density belongs to a family of densities parametrized by $\theta \in \mathbb{R}^d$. The density at x with parameter θ is written $p(x;\theta)$. As the joint density of independent random variables is the product of the marginal densities, we can write the joint density of the samples as

$$p_{D|\theta}(x^n;\theta) = \prod_{k=1}^{n} p(x_k;\theta) \ .$$

The parameter θ can be inferred (estimated) from the finite data set by maximizing the log-likelihood of the data (Fisher, 1925):

$$\hat{\theta} \in \mathrm{argmax}_\theta \sum_{k=1}^{n} \log p(x_k;\theta) \ , \tag{3.2}$$

where $\mathrm{argmax}\, f$ denotes the set of maximizers of f (often a singleton). The predictive density, defined as the density of a new sample x_{n+1}, conditionally to the data set, is then given by

$$p_{x|D}(x_{n+1};x^n) = p(x_{n+1};\hat{\theta}) \ ,$$

where $\hat{\theta}$ is in this context referred to as a plug-in estimate.

Remark 3.1. If $\log p(x;\theta)$ is a continuously differentiable concave function with respect to $\theta \in \Theta \subset \mathbb{R}^d$, the maximum in (3.2) can be obtained from the first-order optimality condition

$$\nabla_\theta \sum_{k=1}^{n} \log p(x_k;\theta) = \sum_{k=1}^{n} \frac{\nabla_\theta p(x_k;\theta)}{p(x_k;\theta)} = 0 \ , \tag{3.3}$$

provided that the resulting estimate $\hat{\theta}$ is in the interior of Θ. The left-hand side of (3.3) is called the *score function.* The random vector

$$\delta(\theta) = \sum_{k=1}^{n} \frac{\nabla_\theta p(X_k;\theta)}{p(X_k;\theta)} \qquad \text{with } X_k \text{ drawn according to } p(\cdot;\theta)$$

is called the efficient score. The covariance matrix of the efficient score is called the Fisher information matrix, written $I(\theta; n) \in \mathbb{R}^{d \times d}$ — the argument n stresses that we define the Fisher information matrix for n observations. We write $I(\theta)$ for $I(\theta; 1)$. Under suitable conditions on $p(X; \theta)$ allowing the interchange of the expectation and differentiation operators,

$$I_{ij}(\theta; n) = -n\, \mathbb{E}\{\frac{\partial^2 \log p(X;\theta)}{\partial\theta_i \partial\theta_j}\} \ ,$$

which shows that the Fisher information is related to the curvature of p evaluated at θ.

Let $\phi(D) \in \mathbb{R}^d$, with D made of n observations, denote an unbiased estimator of θ, that is, an $\mathbb{R}^d$-valued mapping such that $\mathbb{E}\{\phi(D)\} = \theta$. Let us also assume that the true density $\mathbb{P}_X$ is $p(\cdot; \theta)$. Then under some regularity conditions, the covariance matrix of $\phi(D)$, written Σ_ϕ, satisfies the Cramér-Rao inequality:

$$\Sigma_\phi \succeq I^{-1}(\theta; n) \ ,$$

the inequality referring to the cone of positive semi-definite matrices. If the maximum likelihood estimate $\hat{\theta}$ in (3.2) is unique and "far enough" from the boundary of Θ, then for n "large enough", $\hat{\theta}$ is "approximately" normally distributed with mean θ and covariance $I^{-1}(\theta; n) = n^{-1} I^{-1}(\theta)$. This result is usually expressed by saying that

$$\sqrt{n}(\hat{\theta} - \theta) \text{ converges in distribution to } \mathcal{N}(0, I^{-1}(\theta)) \ .$$

As $\hat{\theta}$ has asymptotically the best possible covariance for unbiased estimators, the maximum likelihood estimator is said to be an efficient estimator. Note, however, that the covariance matrix relative to a *biased* estimator could be smaller.

For asymptotic results in situations where $p(x; \theta)$ is not twice differentiable in a neighborhood of θ, see Dupacova and Wets (1988); for asymptotic results in situations where θ is on the boundary of Θ, see Shapiro (2000). □

The function $\ell(x; \theta) = -\log p(x; \theta)$ is a loss function, called the negative log-likelihood loss function. If the samples are truly drawn independently, the maximization of the log-likelihood of the data in (3.2) is a surrogate program for the minimization of $\mathbb{E}\{\ell(X; \theta)\}$, where the expectation is taken with respect to the true data-generating density $\mathbb{P}_X$. Hence, as the surrogate problem is ill-posed, it may be preferable to penalize the objective when the number n of samples is small, for example (Tikhonov and Arsenin, 1977) by adding a regularization term $-\frac{1}{2}\lambda||\theta||^2$ with $\lambda > 0$ to the log-likelihood:

$$\hat{\theta} \in \text{argmax}_\theta \sum_{k=1}^{n} \left(\log p(x_k; \theta) - \tfrac{1}{2} n^{-1} \lambda ||\theta||^2\right) \ . \tag{3.4}$$

When the objective can be written as a sum $\sum_{k=1}^n \rho(x_k; \theta)$ for some function ρ, the corresponding estimates $\hat{\theta}$ are sometimes referred to as *M-estimates* (Maximum Likelihood Type Estimates) (Huber, 1964).

When the true density of the data set $\mathbb{P}_D$ cannot be identified to $p_{D|\theta}$ for some θ, be it because the data-generating density does not belong to the parametric family of densities, or because the samples are not drawn independently, the probability model is said to be misspecified. This is the most common situation encountered in practice. Using maximum likelihood type estimators with misspecified models does not necessarily lead to inconsistent estimates (estimates with non vanishing bias as the number of samples grows to infinity): what can really harm consistency is rather to omit some of the relevant variables for explaining the data, or to assume wrong constraints between the components of x (White, 1982).

3.1.2 Bayesian Averaging

In the simplest Bayesian approach to explaining data, one assumes that the samples are drawn independently, that the data-generating density belongs to a family of densities parametrized by $\theta \in \mathbb{R}^d$, and in addition that the parameter θ has been drawn from a fixed density p_θ, called the prior. The conditional density of θ given the data, written $p_{\theta|D}$, is called the posterior. It is computed according to the Bayes formula for conditional distributions

$$p_{\theta|D}(\theta; x^n) = \frac{p_{D|\theta}(x^n; \theta) p_\theta(\theta)}{\int_\Theta p_{D|\theta}(x^n; \theta) p_\theta(\theta) \mathrm{d}\theta} \tag{3.5}$$

where Θ denotes the domain of θ, and $\mathrm{d}\theta$ denotes the Lebesgue measure if θ is continuous, or the counting measure if θ is countable. Note that the normalization of $p_{\theta|D}$ by the integral makes it possible to (formally) use improper priors (Jeffrey, 1939), that is, "generalized" densities p_θ such that $\int_\Theta p_\theta(\theta)\mathrm{d}\theta = +\infty$. In particular, using a uniform prior $p_\theta(\theta) = 1$ amounts to identify $p_{\theta|D}$ to the likelihood, $p_{D|\theta}$. On the other hand, for certain families of distributions $p(x; \theta)$, there exists a special choice for the prior, referred to as the conjugate prior, such that the prior and the posterior belong to the same family (Raiffa and Schlaifer, 1961). This is convenient for evaluating $p_{\theta|D}$ in closed-form, but reduces the prior to a mere device for making tractable predictions.

Note that the frequentist approach also makes prior assumptions, for instance through the choice of λ in (3.4), which is formally set to 0 in the maximum likelihood estimate (3.2). The family $p(x; \theta)$ and the type of regularization are also often chosen to facilitate the evaluation of the M-estimate.

If (3.5) cannot be evaluated in closed-form, the simplest approximation is Maximum A Posteriori (MAP) estimation, which consists in approximating $p_{\theta|D}$ by a distribution with all the probability mass concentrated on the mode of $p_{\theta|D}$ (with ties broken arbitrarily). Maximum A Posteriori estimation with a uniform prior coincides with Maximum Likelihood estimation. More advanced approximation techniques include asymptotic methods such as Laplace's method (MacKay, 2003, Chapter 27) which consists in replacing a distribution by a Gaussian approximation, importance sampling, multiple quadrature, and Markov Chain Monte Carlo methods (MCMC) (Metropolis et al., 1953; Neal, 1993, 2010), which essentially consists in approximating the integration over θ by accumulating evaluations at points θ_k generated by a random walk in the parameter space Θ. The relative merit of these methods are discussed in Evans and Swartz (1995) and in MacKay (2003, Chapters 29 & 30). The methods that scale well with the dimension d are the MCMC methods.

The Bayesian approach aims at taking into account (through the prior) the uncertainty associated to the selection of a particular value $\hat{\theta}$ for making predictions after having observed the data. The density of a new sample x_{n+1}, conditionally to the data set, is given by a mixture of all members of the parametric family, obtained by averaging all the members with importance weights given by $p_{\theta|D}$ (Bayesian averaging):

$$p_{x|D}(x_{n+1}; x^n) = \int_\Theta p(x_{n+1}; \theta) p_{\theta|D}(\theta; x^n) \mathrm{d}\theta \ . \tag{3.6}$$

Here again approximations can be employed so as to carry out the integration.

The frequentist approach takes the uncertainty into account through regularization. In many cases, there is a connection between particular prior distributions and particular regularization mechanisms, so that regularization can be reinterpreted as an implicit prior, and vice-versa (Kimeldorf and Wahba, 1970).

Remark 3.2. Under suitable conditions (Walker, 1969), one can show that if the true density $\mathbb{P}_X$ is $p(\cdot; \theta)$ for some θ in the interior of its domain Θ, then for n "large enough", the posterior distribution $p_{x|D}$ given the data is "approximately" a normal distribution with mean $\hat{\theta}$ (maximum likelihood estimator on the data) and covariance matrix $I^{-1}(\theta; n)$ (inverse of the Fisher information matrix for n observations). □

3.1.3 Mixture Models

Bayesian averaging suggests to consider a more expressive class of probability models, formed by combining models $p(x; \theta)$ from the parametric family.

Following this idea, one assumes that the samples of the data set are drawn independently by first drawing a density from a family of densities parametrized by $\theta \in \mathbb{R}^d$, according to some fixed but unknown distribution p_θ, and then by drawing the sample from the selected density $p(x; \theta)$. Note that θ is a latent variable: it is not part of the observed samples collected in the data set. The density for a new sample x_{n+1} will be given by (3.6), except that now $p_{\theta|D}$ loses its Bayesian interpretation and is viewed as a free component of the probability model. When $p_{\theta|D}$ has a finite support of fixed cardinality (finite mixture models), $p_{\theta|D}$ is described by a finite number of parameters (values and probability masses) and can thus be optimized according to the maximum likelihood principle (Hasselblad, 1966), for example through the Expectation-Maximization (EM) iterative procedure (Dempster et al., 1977), that treats a sample x as an incomplete observation of (θ, x).

3.1.4 Model Selection

In a more advanced approach to explaining data, one assumes that the data-generating density $\mathbb{P}_X$ belongs to a space of densities described by a model structure $M \in \mathcal{M}$ with model parameters $\theta_M \in \Theta_M$. The dimension of θ_M can vary with M. One speaks of nested models when there exists a complete ordering $M_1, M_2, \ldots$ of the models such that all the densities representable by M_ν are also representable by $M_{\nu+1}$.

Models of different complexity (flexibility) coexist in the hypothesis space. Loosely speaking, low complexity was originally associated to a small number of parameters for describing a model (Rissanen, 1978), or to a greater smoothness of the model (Good and Gaskins, 1971). It is convenient to view a model through the pair (M, s), where s is a complexity parameter associated to M. For nested models M_ν, we can assume that there exists an increasing function that maps structure indices ν to complexity parameters s.

Model selection methods aim at identifying the model M that best explains the data, often by adapting the complexity s of the selected model to the size n of the data set. Note that the misspecification issue is completely irrelevant here inasmuch as one seeks to explain learnable properties of the data (Vapnik, 1998): assumptions on a hypothetical true distribution $\mathbb{P}_X$ are a matter of pure convenience.

In finite mixture density estimation for instance, the cardinality of the finite support of $p_{\theta|D}$ determines the model structure and induces a model ordering, so that competing models can be ranked according to the log-likelihood of the data penalized by a complexity parameter s (Li and Barron, 2000).

3.1.5 Bayesian Model Averaging

Bayesian Model Averaging is the extension of Bayesian averaging to hypothesis spaces formed of several model structures. Instead of performing model selection, one defines, for nested models M_ν, a prior p_ν on the structure index $\nu \in N \subset \mathbb{N}$ relative to the model M_ν. If $p(x; \nu, \theta_\nu)$ denotes the density associated to M_ν with parameter $\theta_\nu \in \Theta_\nu$, the predictive distribution is given by

$$p_{x|D}(x_{n+1}; x^n) = \sum_{\nu \in N} p_{\nu|D}(\nu; x^n) \int_{\Theta_\nu} p(x_{n+1}; \nu, \theta_\nu) p_{\theta_\nu|D}(\theta_\nu; x^n) \mathrm{d}\theta_\nu \tag{3.7}$$

with $p_{\nu|D}$ interpreted as the importance weight of the model M_ν, determined by updating the prior p_ν using the observed data.

For models M identified by some continuous hyper-parameter $\alpha \in \mathbb{R}^q$, so that x follows $f(x; \alpha, \theta)$, it is common to define a joint prior $p_{\alpha,\theta} = p_\alpha p_{\theta|\alpha}$ on $(\alpha, \theta) \in A \times \Theta$. The predictive distribution is then given by

$$p_{x|D}(x_{n+1}; x^n) = \int_A \left(\int_{\Theta(\alpha)} p(x_{n+1}; \alpha, \theta) p_{\theta|\alpha,D}(\theta; \alpha, x^n) \mathrm{d}\theta \right) p_{\alpha|D}(\alpha; x^n) \mathrm{d}\alpha \ . \tag{3.8}$$

Approximations include MAP-type simplifications (selection of the model that maximizes $p_{\alpha,\theta|D}$), model expansion techniques that restrict the integration to a neighborhood of a good model (Draper, 1995), selective model averaging that restrict the integration to a series of good models, and Markov Chain Monte Carlo techniques that perform a random walk in the space of models (Madigan and York, 1995). There also exist connections between particular model selection criteria and particular priors on models (Zhang et al., 2009).

3.1.6 Ensemble Methods

Ensemble methods build on the principles suggested by Bayesian averaging, Bayesian model averaging, and MCMC or importance sampling Monte Carlo techniques. We will present these methods under a common umbrella by saying that ensemble methods assume a predictive distribution of the form (3.8) with a MCMC approximation already applied to the integral, that is,

$$p_{x|D}(x_{n+1}; x^n) = \sum_{\nu=1}^{m} p(x_{n+1}; \alpha_\nu, \theta_\nu) w_\nu \ , \tag{3.9}$$

where

- α_ν describes the structure of a model M_ν,
- θ_ν refers to the parameters of the model M_ν,
- $p(\cdot; \alpha_\nu, \theta_\nu)$ stands for the predictive distribution according to the model M_ν,
- m is the number, possibly depending on x^n, of models M_ν that are generated sequentially given the data x^n, and possibly given information extracted from previous models — this information would be represented by the state of the Markov Chain in MCMC,
- $w_\nu \geq 0$ is the weight of the model M_ν in the ensemble, with weight updates permitted during the construction of the sequence — in particular, setting an initial sequence of weights to 0 amounts to discard models generated during a first "burn in" period.

Each term in the sum represent the contribution of a weighted sample as if it were drawn from the joint density $p_{\alpha,\theta|D} = p_{\theta|\alpha,D} p_{\alpha|D}$ in (3.8), duplicate samples being permitted.

Bagging.

In bagging methods (bootstrap aggregating methods) (Breiman, 1996), the sequence $M_1, \ldots, M_m$ is built by sampling models M_ν as follows: the parameters of a model M_ν are particularized (plug-in estimate) to a random resampling D_ν of the elements in the data set D with replacement (bootstrap) (Efron and Tibshirani, 1993), each element of D having the same probability to be drawn, repetitions permitted. The number of draws is usually set to a fixed ratio α of the cardinality of D.

The idea of bagging has originally been proposed in the context of prediction, but has then also been applied to density estimation. In the context of prediction, bagging has been shown to reduce the variance of estimators that are unstable with respect to perturbations of the data x^n (assuming that the number n of samples is held fixed), with a beneficial effect on the bias/variance tradeoff provided that the resampling ratio α is large enough (Buja and Stuetzle, 2006) ($\alpha = 1$ in Breiman's original algorithm).

Boosting.

In boosting methods (Schapire, 1990; Freund and Schapire, 1996; Schapire et al., 2002), the sequence $M_1, \ldots, M_m$ is built by sampling models M_ν as follows: the parameters of a model M_ν are particularized to a random resampling D_ν of the elements in a data set D, each element of D having a certain probability to be drawn, determined by assigning to each element k of the data set D an importance weight that is relatively greater if the element k is not well explained (or predicted) by the previous models. The m models are then aggregated by weighted averaging (Littlestone and Warmuth, 1989), the weights reflecting the respective quality of each model at explaining the data. The weighted aggregation scheme depends on generalization bounds proper to the loss function chosen for scoring the models.

Boosting has been shown to induce predictive models with excellent generalization capabilities starting from a family of models M_k having their prediction slightly better than random predictions once their parameters θ_k are adapted to the data (weak models). The reasons for the empirical success of boosting may not still be fully elucidated (Mease and Wyner, 2008). The aggregation schemes used for the online prediction of (bounded) sequences $X_1, X_2, \dots$ *without assuming a probabilistic model* $\mathbb{P}_X$ (Cesa-Bianchi and Lugosi, 1999), as advocated by Dawid (1984), are similar to the aggregation schemes used in boosting (Cesa-Bianchi et al., 1997), and have been analyzed in terms of their generalization ability in the context of online prediction (Cesa-Bianchi et al., 2004).

Other Ensemble Methods.

While bagging and boosting exploit perturbations of the data set based on the temporary presence or absence of particular samples, other ensemble methods use perturbations of the data set based on the temporary presence or absence of certain components of x (features) (Dietterich, 2000; Breiman, 2001; Geurts et al., 2006). Research is still very active in machine learning for finding beneficial ways to perturb data sets by further randomizing the features, be it in the context of ensemble methods stricto sensu (Breiman, 2000), or in the context of kernel methods (Rahimi and Recht, 2008, 2009; Shi et al., 2009).

3.2 Adaptation to Stochastic Programming

Most stochastic programs of practical interest use unbounded objective functions. This is in strong contrast with the usual assumptions made in machine learning and online sequence prediction. A large body of theoretical work based on empirical processes theory (Pollard, 1990), large-deviation theory and concentration inequalities such as Hoeffding's inequality (Hoeffding, 1963), Azuma's inequality (Azuma, 1967), McDiarmid's inequality (McDiarmid, 1989), ultimately relies on a bounded range assumption for establishing the generalization bounds that back the predictions realized by mixtures of experts or boosting-type approaches (Koltchinskii and Panchenko, 2002; Audibert et al., 2007; Shivaswamy and Jebara, 2010). Results and reasonings from those works are thus difficult to adapt to the usual models of stochastic programming. Note that when one accepts to focus on bounded objective functions, theoretical investigations are possible (Nesterov and Vial, 2008).

We follow another path here, and investigate empirically the use of bagging methods for estimating an optimal first-stage decision to a multistage stochastic program. We consider perturbed scenario-tree approximations to multistage stochastic programs with decisions valued in a nonconvex feasible set. The standard averaging rule is not implementable, calling for a more sophisticated aggregation strategy. The first-stage decision plays the role of the parameter θ considered in Section 3.1.

3.2.1 Principle of the Approach

In this section, we outline the principle of the proposed approach, and discuss the main underlying assumptions. We start by describing the class of problems that we address and then provide an overview of the main ingredients of the proposed solution approach, namely, a procedure for generating an ensemble of scenario trees, an algorithm based on the cross-entropy method for computing near-optimal first-stage decisions, and a kernel-based method for aggregating the first-stage decisions derived from the ensemble of scenario trees. Background material on kernel methods can be found in Appendix C.

Problem Formulation and Assumptions.

We consider a system that evolves according to a state transition equation

$$x_{t+1} = f_t(x_t, u_t, w_t) \ ,$$

starting from a fixed initial state $x_0 \in X$. The state trajectory $x_0, x_1, \dots$ is controlled by the decisions $u_t \in U$ and perturbed by disturbances $w_t \in W$ generated by a memoryless, exogenous process, so that w_t is drawn from a fixed probability distribution $\mathbb{P}_{t,w}$. A reward process $r_0, r_1, \dots, r_{T-1}$ is defined by mappings r_t from $X \times U \times W$ to $\mathbb{R}$ with values $r_t(x_t, u_t, w_t)$. The initial state x_0, system dynamics f_t, reward functions r_t, disturbance model $\mathbb{P}_{t,w}$, are assumed to be known by the decision maker, whose goal is to find a non-anticipative decision strategy μ for selecting actions u_t and maximizing the expectation of the cumulated rewards over T stages, written

$$J^*(x_0) = \max_{\mu} \mathbb{E}\{\sum_{t=0}^{T-1} r_t(x_t, u_t, w_t)|x_0\}. \tag{3.10}$$

The candidate strategies μ for selecting the decisions u_t at times $0 \le t < T$ is a sequence of time-indexed deterministic mappings μ_t from the current history $h_t = (w_0, w_1, \dots, w_{t-1})$ of the disturbance process to a fixed decision $u_t = \mu_t(h_t) \in U$.

(To compare this setup to the Markov Decision Process framework, one may assume temporarily that the disturbance process is observable. Then, the mappings from h_t to u_t are as expressive as mappings from states x_t to actions u_t, since the states x_t are ultimately a function of h_t: x_t can be recovered from h_t, given x_0, u_0, the decision rules μ_1,. . . μ_{t-1}, and the state transition functions f_0, . . . , f_{t-1}).

No assumption is made about the dimensionality or the structure of the state space X. The space U of possible actions, and the space W of possible disturbances, are assumed to be made of a finite number of elements.

The notations x_t, u_t, w_t, f_t, r_t, the assumption of a memoryless disturbance process, the initial condition for $t = 0$ rather than $t = 1$, are meant to facilitate the connection with the usual discrete-time optimal control framework (Bertsekas, 2005a). The memoryless assumption may be relaxed, by simply requiring that the probabilities of all future disturbance sequences are known by the decision maker. The temporal decomposition of the performance criterion in Equation (3.10) is fundamental in an optimization procedure based on dynamic programming, but is not essential in the present approach.

Exact Solution Based on a Complete Scenario Tree.

A complete scenario tree of depth T represents all the possible realizations of the process $w_0, w_1, \ldots, w_{T-1}$, together with their probabilities. In such a tree, the root node (depth 0) corresponds to time $t = 0$ and to an empty process history. To each node n of depth $t \in \{1, 2, \ldots, T\}$ in the tree corresponds a possible history $h_n = [w_0, \ldots, w_{t-1}]_n$ of the process, through the unique path from the root to the node n. The disturbance $(w_{t-1})_n$ is directly assigned to the node n together with its probability, while $[w_0, \ldots, w_{t-2}]_n$ and their joint probability can be collected from the disturbances and probabilities associated to the nodes in the path.

Any strategy μ can be represented on the completetree by assigning to each node n of depth $0 \leq t < T$ a fixed value $u_n = \mu(h_n) \in U$. Consequently, searching for an optimal strategy is equivalent to jointly optimizing the values u_n assigned to the internal nodes of the tree.

The performance criterion defined in (3.10) can be evaluated once decisions have been assigned to the nodes. Indeed, given the value of x_0, $u_0 = \mu_0$ and a particular w_0, one can evaluate $r_0 = r_0(x_0, u_0, w_0)$ and $x_1 = f_0(x_0, u_0, w_0)$ by the knowledge of r_t and f_t at $t = 0$. The values r_0 and x_1 can thus be assigned to the node associated to the disturbance process history $[w_0]$. The probability $\mathbb{P}_{0,w}(w_0)$ is determined from the disturbance process model. Given the nodal decision for $u_1 = \mu_1(w_0)$, and using x_1 and a particular w_1, one gets the values of x_2 and r_1 for the corresponding particular value of $[w_0, w_1]$. The value r_1 can be assigned to the node corresponding to $[w_0, w_1]$, to which is also assigned a probability $\mathbb{P}_{0,w}(w_0) \cdot \mathbb{P}_{1,w}(w_1)$, since we assume that w_0, w_1 are independent. The propagation of nodal values is pursued until values are assigned to x_T and r_{T-1}. It can be carried out for each disturbance path in the tree. Therefore, for a given decision strategy μ, all the rewards and probabilities entering the evaluation of the expectation in (3.10) can be computed, given the system model $f_t, r_t, \mathbb{P}_{t,w}$ and the initial state x_0.

Without any particular structure assumed for f_t and r_t, the optimization of the policy μ may be done by a direct search of the decisions u_n assigned to the nodes of the tree. However, the number of possible combinations is of the order of $|U|^{|W|^{T-1}}$, meaning that as soon as the cardinalities $|U|$, $|W|$, or the time horizon T are large, an exact optimization is intractable.

Approximate Solution Based on an Ensemble of Incomplete Scenario Trees.

Conceptually, an incomplete scenario tree is obtained by selecting a subset of the nodes of a complete tree, by removing the arcs leading to these nodes as well as the subtrees emanating from them, and by adjusting the probabilities of successor nodes so that they still sum up to one. Each node of depth inferior to T in the resulting incomplete tree must still have at least one successor node. In practice, incomplete trees can be constructed in a top-down fashion, by subsampling the disturbance process according to its probabilities, in such a way that the resulting incomplete scenario tree is small enough to induce a tractable optimization problem.

While the decisions associated to the nodes of an incomplete scenario tree yield an incomplete decision strategy, such a strategy always provides a value for the first-stage decision u_0. Therefore, building an incomplete tree, optimizing an incomplete decision strategy over it, and extracting the first-stage decision, can be viewed as a simplified estimation procedure for the search of u_0 on the complete tree.

Rather than this usual estimate for an optimal u_0, we propose to consider the model space of all incomplete trees, sample an ensemble of models in that space, and estimate u_0 by aggregating the optimal first-stage decisions associated to these models.

Recalling the conditions for the success of bagging approaches (Section 3.1.6), we can expect that this estimation procedure can be beneficial when the individual incomplete trees are not too small. Working on a small tree or on a single scenario could induce too large a bias on the individual first-stage decisions with respect to the optimal first-stage decision.

The main ingredients of this approach are further discussed below, with an emphasis on decision problems with large discrete action and disturbance spaces.

3.2.2 Generation of an Ensemble of Incomplete Trees

Under our assumptions, the disturbance space can be written as $W = \{w^1, \dots, w^{|W|}\}$, where w^j stands for a specific realization of the disturbance. To each disturbance w^j is associated a probability mass $\mathbb{P}_{t,w}(w_t = w^j) = p^j > 0$ for each $j \in J = \{1, \dots, |W|\}$, with $\sum_{j=1}^{|W|} p^j = 1$.

In our proposal, the generation of an ensemble of incomplete scenario trees is based on the random sampling of a small number of successor nodes, in a top-down fashion. This amounts to replace the discrete distributions $\mathbb{P}_{t,w}$ defined by the pairs (w^j, p^j), $j \in J$, by simpler distributions $\tilde{\mathbb{P}}_{t,w}$ (the approximation can be different at each node), and proceed by developing the nodes recursively. We assume that the simpler distributions are described by the pairs $(w^j, \tilde{p}^j)$ with $j \in \tilde{J} \subset J$ and $\sum_{j=1}^{|\tilde{J}|} \tilde{p}^j = 1$. In fact, as the disturbance space is discrete, there would be no obvious way to define intermediary or averaged values for the disturbances. We write V for the set $\{w^j\}_{j\in\tilde{J}}$. The set $W \setminus V = \{w^j\}_{j\in J\setminus\tilde{J}}$ is the set of disturbances omitted by the approximation.

The probability masses of the disturbances in $W \setminus V$ must be reallocated (transported) to one or several disturbances in V. To do that in a way consistent with approximation methods based on probability metrics (Section 2.4.2), we assume that the user can define distances between the elements in W, ideally in such a way that close disturbances induce similar state transitions and similar rewards. These ideas will be illustrated in Section 3.3. Under this assumption, it makes sense to redistribute the probability mass of a disturbance in $W \setminus V$ among the nearest disturbances in V, so as to reduce (heuristically) approximation errors.

A generic way to induce distances in a discrete space is to introduce a positive-definite kernel $k : W \times W \to \mathbb{R}$ with values $k(w, w') = k(w', w)$ (see Appendix C). The distance between two disturbances w, w' is then given by

$$d(w, w') = [k(w, w) + k(w', w') - 2k(w, w')]^{1/2} = d(w', w) \geq 0 \ .$$

For each $w \in W \setminus V$, let $C(w)$ denote the subset of V of nearest neighbors to w:

$$C(w) = \{v \in V : d(v, w) \leq d(v', w) \text{ for all } v' \in V\} \ .$$

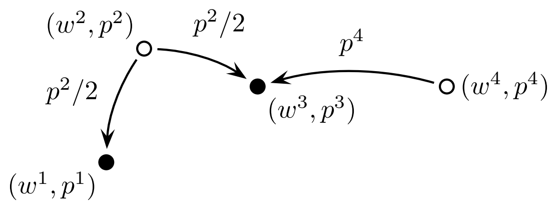

Fig. 3.1: Illustration of the probability redistribution rule (3.11), for $W = \{w^1, w^2, w^3, w^4\}$, $V = \{w^1, w^3\}$, $C(w^2) = \{w^1, w^3\}$, $C(w^4) = \{w^3\}$. The pairs (w^j, p^j) can be embedded in a feature space induced by the choice of the kernel k. The dots represent V (black) and $W \setminus V$ (white) in the feature space where the pairwise distances are evaluated.

For each $v \in V$, let $C^{-1}(v)$ denote the subset of elements in $W \setminus V$ that have v as a nearest neighbor:

$$C^{-1}(v) = \{w \in W \setminus V : w \in C(v)\} \ .$$

The probability mass of a node n to which a disturbance $w^j \in V$ is associated is then given by

$$\tilde{p}^j = p^j + \sum_{w^k \in C^{-1}(w^j)} p^k / |C(w^k)| \ . \tag{3.11}$$

The probability mass redistribution rule is illustrated in Figure 3.1.

3.2.3 Optimization with the Cross-Entropy method

Consider an incomplete scenario tree with N nodes numbered from 1 (root, depth 0) to N (last leaf, depth T). We assume that the leaf nodes (depth T) are numbered from $N - L + 1$ to N, where L is the number of leaf nodes or equivalently, the number of scenarios. Let $\boldsymbol{w}_n$, $\boldsymbol{x}_n$, $\boldsymbol{u}_n$, $\boldsymbol{r}_n$, denote respectively the disturbance w_{t-1}, state x_t, decision u_t, reward r_{t-1} assigned to node n, where t corresponds to the depth of the node, and where x_t, u_t, r_{t-1} are conditioned on the disturbance process history $[w_0, \dots, w_{t-1}]$ induced by the path from the root to the node n. The root node has no disturbance and reward assigned to it. The leaf nodes ($N - L + 1 \leq n \leq N$) have no decision $\boldsymbol{u}_n$ assigned to them. Let $\boldsymbol{p}_n$ be the probability mass assigned to node n (the probabilities $\boldsymbol{p}_n$ of the nodes of depth t sum up to 1). For $n > 1$, let n^- denote the index of the parent node of node n. Let f_n^- and r_n^- denote the functions f_t and r_t for t equal to the depth of the node n^-. The problem (3.10) formulated on the incomplete scenario tree becomes

$$\begin{array}{lll} \text{maximize} & \displaystyle\sum_{n=2}^{N} \boldsymbol{p}_n \boldsymbol{r}_n & \\ \text{subject to} & \boldsymbol{x}_1 = x_0 & \\ & \boldsymbol{x}_n = f_n^-(\boldsymbol{x}_{n^-}, \boldsymbol{u}_{n^-}, \boldsymbol{w}_n) & 2 \leq n \leq N \\ & \boldsymbol{r}_n = r_n^-(\boldsymbol{x}_{n^-}, \boldsymbol{u}_{n^-}, \boldsymbol{w}_n) & 2 \leq n \leq N \end{array}$$

over variables $\boldsymbol{x}_n \in X$, $\boldsymbol{u}_n \in U$, $\boldsymbol{r}_n \in \mathbb{R}$. By the implicit treatment of the equality constraints, the problem can be rewritten as

$$\text{maximize} \quad F(\boldsymbol{u}_1, \dots, \boldsymbol{u}_{N-L}) \quad \text{over } \boldsymbol{u}_1, \dots, \boldsymbol{u}_{N-L} \in U. \tag{3.12}$$

We view F as an arbitrary mapping from $U \times \cdots \times U = U^{N-L}$ to $\mathbb{R}$. In theory, the maximum could be computed by sorting the values of F for the $|U|^{N-L}$ possible inputs. For brevity, we will write $\boldsymbol{u}$ for $[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N-L}]$.

For the estimation of the maximum of F over $\boldsymbol{u} \in U^{N-L}$, we use the Cross-Entropy method (CE method) (Rubinstein and Kroese, 2004). When it is applied to importance sampling, the Cross-Entropy method aims at selecting, from a family of parametrized densities, the density that has the smallest Kullback-Leibler divergence (defined previously by (3.1)) with respect to an ideal importance sampling density. Rubinstein and Kroese (2004) note that when the densities come from an exponential family of distributions (defined in Chapter 5), the use of the Kullback-Leibler divergence (CE-distance) allows analytical calculations, and reduces the complexity of importance sampling algorithms that sequentially update the parameters of the sampling distributions.

When the CE method is applied to an optimization problem, viewed as the search of a particular rare event, the method is based on two components:

- A random generator parametrized by θ for sampling candidate solutions $\boldsymbol{u}$ according to a density $g(\cdot;\theta)$:

$$\boldsymbol{u} \sim g(\cdot;\theta) \ . \tag{3.13}$$

  The parametrization by θ must be chosen in such a way that the distribution g can both be uniform over each possible solution in U^{N-L}, or concentrated on any particular solution in U^{N-L}.

- The procedure that computes the value of F for a candidate solution $\boldsymbol{u}$, where $F(\boldsymbol{u})$ will be interpreted as the *score* assigned to $\boldsymbol{u}$ by F:

$$\boldsymbol{u} \mapsto F(\boldsymbol{u}) \ . \tag{3.14}$$

The method works as follows. Starting with the value of θ that corresponds to the uniform distribution over the space of solutions, one draws N_{CE} samples $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^{N_{\mathrm{CE}}}$ from the density $g(\cdot;\theta)$, scores them using the scoring function F, and tags as *elite solutions* the samples with a score greater or equal to the $\lceil \rho N_{\mathrm{CE}} \rceil$-th best score, written $\hat{\gamma}$. The parameter ρ is set to a small positive value, typically 0.01 (a value for which the elite solutions correspond to the best percentile of the empirical distribution of the score). The parameter θ is then updated so as to decrease the CE distance of $g(\cdot;\theta)$ with respect to the empirical density induced by the elite solutions. The update rule proposed by Rubinstein and Kroese (2004, Equation 4.8) is

$$\theta \leftarrow \hat{\theta} \quad \text{where } \hat{\theta} \in \operatorname{argmax}_{\theta} \sum_{k\colon F(\boldsymbol{u}^k) \geq \hat{\gamma}} \log g(\boldsymbol{u}^k;\theta) \ .$$

Thus in fact $\hat{\theta}$ is just the maximum likelihood estimate of the sampling density $g(\boldsymbol{u};\theta)$ given the data set of elite samples (the parameter update maximizes the probability of generating the elite samples).

After the parameter update step, a new set of N_{CE} samples is redrawn from $g(\cdot;\theta)$. The parameter update/resampling procedure is repeated until the density $g(\cdot;\theta)$ concentrates on a single solution, or until the elite scores have ceased to improve. The best candidate solution with respect to F (at any iteration) is then returned. The authors of the method recommend to choose N_{CE} proportional to the number of parameters in θ (the dimension of θ depending itself on the size of the search space). They also propose to smooth the updates of θ as follows: denoting by θ_j the value of θ at iteration j, they suggest to set

$$\theta_{j+1} = \alpha\hat{\theta} + (1-\alpha)\theta_j \tag{3.15}$$

where $\alpha \in (0,1]$ is the smoothing factor.

3.2.4 Aggregation of First-Stage Decisions

Let $u_0^\nu \in U$ denote a near-optimal first-stage decision (root-node decision) relative to an incomplete tree ν, where the tree and u_0^ν have been obtained by the procedures described in the preceding subsections. Let $S_0 = \{u_0^1, \dots, u_0^m\}$ denote the set of near-optimal first-stage decisions relative to m such trees.

The analysis of bagging methods (see Section 3.1.6) suggests that forming an aggregate first-stage decision from the set S_0 could decrease the influence on u_0^ν of the particular sampled tree ν. However, a difficulty in the present setup, where decisions are valued in a large discrete set U, consists in defining an admissible and useful aggregation rule. By admissible, it is meant that the aggregated decision is valued in U; by useful, it is meant that the aggregation rule should be able to preserve the structure and properties of near-optimal solutions.

To this end, we propose to take as the aggregated first-stage decision, written u_0^a, the decision in S_0 that is nearest to a special point that we call the *centroid* of the decisions in S_0, and that is defined, as we explain next, using the metric induced on U by a kernel k_U that quantifies the similarity between decisions in U. Formally, we may assume momentarily (so as to clarify intermediate calculations) that we have access to the feature map $\varphi : U \to H$ relative to the reproducing kernel Hilbert space H induced by $k_U : U \times U \to \mathbb{R}$. Alternatively, we may assume that we can enumerate a finite number of features for the decisions, from which we induce a kernel by (see Appendix C.4)

$$k_U(u_0, u_0') = \langle \varphi(u_0), \varphi(u_0') \rangle \ ,$$

where $\langle \cdot, \cdot \rangle$ stands for a suitably defined inner product.

The centroid of S_0, written u_0^c, is defined by its coordinate $\phi(u_0^c)$ in the feature space, set to

$$\varphi(u_0^c) = m^{-1} \sum_{\nu=1}^{m} \varphi(u_0^\nu) \ .$$

The squared distance between the centroid and some first-stage decision $u_0 \in U$ is given by

$$\begin{aligned} &||\varphi(u_0) - \varphi(u_0^c)||^2 \\ &= \langle \varphi(u_0), \varphi(u_0) \rangle - 2m^{-1} \sum_{i=1}^{m} \langle \varphi(u_0), \varphi(u_0^i) \rangle + m^{-2} \sum_{i=1}^{m} \sum_{j=1}^{m} \langle \varphi(u_0^i), \varphi(u_0^j) \rangle \ . \end{aligned}$$

The squared distance from some decision $u_0^\nu \in S_0$ to the centroid u_0^c may be expressed in terms of the elements $K_{ij} = \langle \varphi(u_0^i), \varphi(u_0^j) \rangle$ of the Gram matrix $K \in \mathbb{R}^{m \times m}$ by

$$||\varphi(u_0^\nu) - \varphi(u_0^c)||^2 = K_{\nu\nu} - 2m^{-1} \sum_{i=1}^m K_{i\nu} + m^{-2} \sum_{i=1}^m \sum_{j=1}^m K_{ij} \ . \tag{3.16}$$

The aggregated solution

$$u_0^a \in \arg \min_{u_0^\nu \in S_0} ||\varphi(u_0^\nu) - \varphi(u_0^c)||^2 \ , \tag{3.17}$$

$$\text{or equivalently } u_0^a = u_0^j \text{ with } j \in \arg \min_{1 \leq \nu \leq m} \{K_{\nu\nu} - 2m^{-1} \sum_{i=1}^m K_{i\nu}\} \ ,$$

with ties broken arbitrarily, may thus also be computed without the need to refer to the feature map φ once the Gram matrix is given. Therefore, the explicit computation of the centroid in the feature space, which would require the explicit knowledge of the feature map, is not actually needed for evaluating the aggregated solution.

Note that the empirical variance of the ensemble of decisions in the feature space induced by the kernel k_U, defined by

$$\text{var}\{S_0\} = m^{-1} \sum_{\nu=1}^m ||\varphi(u_0^\nu) - \varphi(u_0^c)||_2^2 = m^{-1} \sum_{i=1}^m K_{ii} - m^{-2} \sum_{i=1}^m \sum_{j=1}^m K_{ij} \ ,$$

could also be evaluated even if the feature map is specified only implicitly by the definition of the kernel, and could quantify the discrepancy between candidate decisions in S_0.

Discussion.

First consider the situation where the decision space U only possesses a handful of elements. Thanks to the small cardinality of U, we may expect that optimal first-stage decisions are present among the elements of set S_0. Therefore, a simple majority vote among the elements of S_0 can be taken as the estimate u_a of an optimal first-stage decision. Note that the majority vote can be obtained from the general formulation (3.17) based on kernels by setting $K_{ij} = \delta\{u_0^i = u_0^j\}$, where $\delta\{\cdot\}$ denotes the 0-1 indicator function of the event placed in argument. Indeed, as $K_{\nu\nu} = 1$, the squared distances $||\varphi(u_0^\nu) - \varphi(u^c)||^2$, $1 \leq \nu \leq m$, will only differ by the term $-2m^{-1} \sum_{i=1}^m \delta\{u_0^i = u_0^\nu\}$, proportional to the frequency of u_0^ν in S_0.

Now consider the situation where U is finite but has a cardinality $|U|$ much larger than $|S_0| = m$. It is then very likely that a clear majority will not be attained in S_0, especially if there are many quasi-equivalent decisions in terms of optimality. However, in many situations, U is formed from the combination of several elementary decisions. One could thus combine kernels on the elementary decision spaces, for instance by combining separate majority votes on the elementary decisions.

The kernelization of the decision space enables one to incorporate prior knowledge on the structure of the decision space. Therefore, kernels should be consistent with prior beliefs about the decisions that have similar effects on the problem at hand.

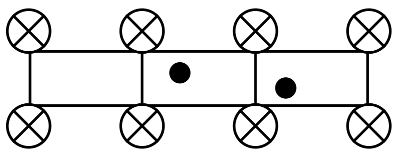

Fig. 3.2: Example of configuration for the sensor network problem, with eight sensors ($\bigotimes$) and two targets ($\bullet$) (figure taken from Dutech et al. (2005)).

3.3 Numerical Experiment

In this section, we illustrate the proposed approach on a test problem problem that has a large, structured, discrete action space. We explain in detail how the action space is kernelized, how the incomplete scenario trees are generated, and how the corresponding optimization problems are solved approximately. We assess the quality of the first-stage decision estimators $\hat{u}_0 = u_0^a$ obtained with the proposed approach by a direct comparison with the optimal strategy, which can be computed exactly in this test problem by dynamic programming (by evaluating recursively the tabular representation of the expected costs-to-go (Q-functions), where the tabular representation of a Q-function has an entry for each combination of state-action pairs).

3.3.1 Description of the Test Problem

The test problem is part of a series of standard benchmark problems proposed for comparing Reinforcement Learning solution approaches (Dutech et al., 2005). The test problem is inspired by a distributed control application from Ali et al. (2005) and named *Sensor Network*. Note that among all the problems selected by Dutech et al. (2005), *Sensor Network* is the only problem with a relatively large discrete decision space.

The problem can be described as follows. Eight sensors bracket an array of three cells, as shown on Figure 3.3.1. Each cell is surrounded by 4 sensors. Two targets float over the cells. Each cell is occupied at most one target.

At time step t, each sensor can be focussed on the cell to its left, on the cell to its right, or be idle. The decision u_t sets the action of the 8 sensors. The decision space is thus a joint action space $U = \{0, 1, 2\}^8$ that encodes the 3 possible actions of the 8 sensors (0: idle, 1: focus left, 2: focus right), totalling $3^8 = 6561$ possible actions. A unit cost is incurred for each focussed sensor; idle sensors have no cost.

The game consists in eliminating the two floating targets from the cells as quickly as possible. Each target start at energy level 3. After sensors have been set according to u_t, the targets move. The leftmost target randomly moves to the left (L), to the right (R), or stay idle (I). A priori the 3 possibilities are equiprobable, but a move is cancelled if the cell where the target intends to go is already occupied or does not exist. After the move performed by the leftmost target, the rightmost target randomly moves according to the same rules.

The intended moves of the targets are viewed as the disturbances in the problem. The disturbance space is $W = \{\mathrm{L}, \mathrm{R}, \mathrm{I}\} \times \{\mathrm{L}, \mathrm{R}, \mathrm{I}\}$, with each of the $3^2 = 9$ possible combinations having probability 1/9. The effective moves may differ inasmuch as intended moves can be blocked as described above.

The sensors are then activated. A target that lies in a cell where 3 sensors or more are focussed loses one energy point. A target is removed from the board when its energy falls to 0. The game ends when the two targets have been eliminated from the board.

The state space $X = \{0,1,2,3\} \times \{0,1,2,3\} \times \{0,1,2,3\}$ encodes the target energy level (0 to 3) of the 3 cells. When a target moves from one cell to another, these cells swap their energy level. The initial state x_0 is either [3 3 0], [3 0 3], [0 3 3], representing 2 targets with energy level 3. The state [0 0 0], corresponding to a board with no remaining targets, is a terminal state. When a target is eliminated, a reward +30 is obtained. Therefore, the instantaneous reward r_t is given by

$$r_t = 30 \sum_{i=1}^{2} \delta\{\text{energy level of target } i \text{ goes from 1 to 0}\} - \sum_{i=1}^{8} \delta\{\text{sensor } i \text{ is not idle}\}$$

The total return is the discounted cumulated reward $\sum_{t=0}^{T} \gamma^t r_t$ with $\gamma = 0.95$, and $T = 10$. The problem is the maximization of the expected total return, starting from some given state, over stochastic programming decision rules $\mu_t : W^{t-1} \to U$ with values $\mu_t(w_0, \dots, w_{t-1}) = u_t$. We concentrate on the estimation of an optimal first-stage decision u_0, given x_0.

3.3.2 Particular Choices

The general approach described in the preceding subsections is adapted to the problem at hand as follows.

- The disturbance space is decomposed as $W = W_a \times W_b$ with $W_a = \{\mathrm{L}, \mathrm{R}, \mathrm{I}\}$ relative to the intended move of the leftmost target, and $W_b = \{\mathrm{L}, \mathrm{R}, \mathrm{I}\}$ relative to the intended move of the rightmost target (if there are still two active targets). A disturbance w is thus decomposed as $w = (w_a, w_b)$, with w_a relative to W_a and w_b relative to W_b.

  We assume (heuristically) that two disturbances w, w' such that $w_a = w'_a$ or $w_b = w'_b$ are similar in terms of induced state transitions and rewards. This assumption is taken into account by defining the kernel on the disturbance space (Section 3.2.2) as

$$k(w, w') = \delta\{w_a = w'_a\} + \delta\{w_b = w'_b\} \ . \tag{3.18}$$

- The incomplete trees are built by sampling, with replacement, N_W disturbances in W at each node. Those N_W disturbances are sampled according to their probabilities. Duplicate samples are then eliminated, and the distinct samples are taken as the children of the node and assigned a probability according to (3.11), with the kernel k defined by (3.18). The initial number of samples N_W is random, so that the branching structure of an incomplete scenario tree is random. For a node of depth $d \geq 0$, $N_W = 3$ samples are drawn with probability $1/(1+d)$, and $N_W = 1$ sample with probability $1 - 1/(1+d)$. Random trees with more than 150 nodes are rejected. Note that the complete tree on the $|W| = 9$ disturbances would have $\sum_{d=0}^{10} 9^d = 3.9 \cdot 10^9$ nodes.

- The sampling distribution for candidate solutions $\boldsymbol{u}$ (Section 3.2.3) is first decomposed into $N-L$ independent components, each component being relative to one internal node of the scenario tree (assuming that the tree has N nodes, including L leaf nodes of depth T). The $N-L$ components are themselves decomposed into 8 independent parts corresponding to the 8 sensors. Each part defines the distribution over $\{0,1,2\}$ of the action of a sensor j at a node i, written a_{ij}. A distribution for a_{ij} is described by the two scalar parameters $p_{ij} = \mathbb{P}\{a_{ij}=0\}, q_{ij} = \mathbb{P}\{a_{ij}=1\}$, with $p_{ij}, q_{ij} \in [0,1]$ and $0 \leq 1-p_{ij}-q_{ij} = \mathbb{P}\{a_{ij}=2\} \leq 1$. A uniform distribution over all possible strategies on the incomplete tree is obtained by setting $p_{ij}=q_{ij}=1/3$ for all i, j, whereas any particular deterministic solution $\boldsymbol{u}$ can be obtained by selecting for each pair (i,j) one of the three configurations $\{p_{ij}=1, q_{ij}=0\}$, $\{p_{ij}=0,\ q_{ij}=1\}$, or $\{p_{ij}=0,\ q_{ij}=0\}$. The distribution for generating a random solution $\boldsymbol{u}$ associated to the incomplete scenario tree is thus specified by $2\cdot 8\cdot (N-L)$ parameters.

  Once the elite samples $\boldsymbol{u}^k$ (Section 3.2.3) have been scored by computing the expected discounted sum of rewards on the incomplete tree with the nodal decisions set to $\boldsymbol{u}^k$, the parameters of the generating distribution for the solutions are updated as follows: given ℓ elite samples, with a_{ij}^k denoting the action a_{ij} from the elite sample $\boldsymbol{u}^k$, $1\leq k\leq \ell$, one first computes the empirical frequencies of the elementary actions in the elite samples,

$$\begin{aligned}\hat{p}_{ij} &\overset{\text{def}}{=} \ell^{-1}\textstyle\sum_{k=1}^{\ell}\delta\{a_{ij}^k=0\}\\ \hat{q}_{ij} &\overset{\text{def}}{=} \ell^{-1}\textstyle\sum_{k=1}^{\ell}\delta\{a_{ij}^k=1\}\ ,\end{aligned}$$

  and then one updates the parameters p_{ij}, q_{ij} of the solution generating distribution by

$$\begin{aligned}p_{ij} &\leftarrow \alpha\,\hat{p}_{ij} + (1-\alpha)\,p_{ij}\\ q_{ij} &\leftarrow \alpha\,\hat{q}_{ij} + (1-\alpha)\,q_{ij}\ ,\end{aligned}$$

  where α is the smoothing factor of Equation 3.15. In the numerical experiments reported in the next section, $\alpha=0.6$.

- The Cross-Entropy optimization can be stopped as soon as the sampling distributions relative to the root node are almost deterministic, since ultimately only the first-stage decision is extracted from a solution $\boldsymbol{u}$ and used in the aggregation step.

  In the numerical experiments reported in the next section, the Cross-Entropy optimization is carried out by generating $N_{\text{CE}} = 32\ (N-L)$ candidate solutions per iteration, that is, twice the number of parameters that describe the solution-generating distribution. The optimization is stopped as soon as the 8 actions at the root have their distribution concentrated on a single action with probability 0.99.

- The aggregation scheme exploits the decomposition of the decision space into separate sensor actions. Recalling that the root node has the node index $i=1$, let a_{1j}^{ν} denote the first-stage action of sensor j from the ν-th solution in the set S_0, $1\leq\nu\leq m$. The action of sensor j in the centroid decision u_0^c (Section 3.2.4) is determined by a majority vote over the action of sensor j in the first-stage decisions

$u_0^\nu = \{a_{1j}^\nu\}_{1\le j\le 8}$ collected in the set S_0:

$$a_{1j}^c = a_{1j}^{\nu(j)} \text{ with } \nu(j) = \min\{\text{argmax}_k \sum_{l=1}^{m} \delta\{a_{1j}^k = a_{1j}^l\}\},\ \ 1 \le j \le 8\ \ ,$$

and then the aggregated decision u_0^a is set to the element $u_0^\nu \in S_0$ sharing the most actions with u_0^a, that is,

$$\nu = \min\{\text{argmax}_k \sum_{j=1}^{8} \delta\{a_{1j}^k = a_{1j}^c\}\}\ \ .$$

A similar effect can be obtained by defining the kernel k_U (Section 3.2.4) between two elements u_0^ν, u_0^σ of S_0 as

$$k_U(u_0^\nu, u_0^\sigma) = \sum_{j=1}^{8} \delta\{a_{1j}^\nu = a_{1j}^\sigma\} = K_{\nu\sigma} = K_{\sigma\nu}$$

and setting

$$u_0^a = u_0^{\nu'} \quad \text{with } \nu' = \min\{\text{argmax}_\nu \sum_{i=1}^{m} K_{i\nu}\}\ \ ,$$

following (3.17) with $K_{\nu\nu}$ constant and ties broken by the lexicographical order on ν.

3.3.3 Simulation Results

Typical outcomes with an ensemble of $m = 5$ incomplete trees are reported in Table 3.1. Three problems corresponding to the 3 initial configurations x_0 of the targets with 3 energy points that float over the 3 cells are considered. Decisions u_0^ν are represented graphically. For instance, the symbol $\substack{-/\backslash-\\ /\backslash/\backslash}$ indicates that 3 sensors are focussed on the leftmost cell, no sensor is focussed on the middle cell, 3 sensors are focussed on the rightmost cell, and the remaining 2 sensors are idle (-). If the targets move onto the leftmost or the rightmost target, they will be hit, so the combined action of the sensors is effective. It would be suboptimal to have 1, 2 or 4 sensors be focussed on a same cell. The table shows that the structure of optimal decisions can be destroyed in the centroid decision u_0^c. In fact, there are several configurations of the sensors that can lead to the same effective targeting of two cells, but these equivalent configurations are made ineffective when they are averaged. The aggregated decision u_0^a reaches a consensus while preserving the structure of effective configurations.

It turns out that for the first and third problem, the aggregated decisions u_0^a are optimal, in the sense that the targeted cells are optimal, according to an exact dynamic programming solution where the decision space is reduced a priori to 6 sensible choices of targeted cells instead of considering the full set of combined actions of the sensors. For the second problem ($x_0 = [3\ 0\ 3]$), the decision u_0^a shown in the table is slightly suboptimal: if subsequently optimal decisions are selected, then u_0^a brings an expected return of 27.78 instead of the optimal return 27.88.

We repeated 10 times the experiment of building an ensemble of 5 trees and computing the aggregated decision. An optimal decision was found: 7 times for $x_0 = [3\ 3\ 0]$, 9 times

Tab. 3.1: Typical result with an ensemble of 5 trees.

x_0	First-stage decision u_0^ν					u_0^c	u_0^a
	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$	$\nu = 5$		
[3 3 0]	`\//-` `//\-`	`\/--` `/---`	`\\/-` `/\\-`	`-/\/` `/\/-`	`\/\-` `/-/\`	`\//-` `/-\-`	`\//-` `//\-`
[3 0 3]	`-/\-` `/\/\`	`\/\/` `-\/-`	`\/\-` `-\/\`	`\-\-` `/\/\`	`\//-` `//\-`	`\/\-` `/\/\`	`\/\-` `-\/\`
[0 3 3]	`\/-/` `/-/\`	`-\\/` `-/\\`	`-\\/` `-/\\`	`\-\/` `/\-\`	`-\//` `-//\`	`-\\/` `-/\\`	`-\\/` `-/\\`

for $x_0 = [0\ 3\ 3]$, and 5 times for $x_0 = [3\ 0\ 3]$ (a second-best decision is found in the 5 other cases).

3.4 Conclusions

This chapter has investigated empirically the estimation of an optimal first-stage decision to a finite-horizon optimal control problem by scenario tree techniques. While we recognize that the solution techniques used in this work might be of limited interest in practice, given that the studied problem class would be more naturally addressed from a Markov Decision Process perspective, we believe that the statistical framework in which the proposed tree-bagging solution technique was presented clarifies the connection between statistical estimation/prediction methods and sequential decision making by stochastic programming.

It is interesting to realize, in particular, that stochastic programming models take seldom into account the intrinsic limitation that only finite-sample approximations can be solved. Usual stochastic programming models are thus close in spirit to maximum likelihood estimation models used on finite data without regularization. Certainly, the appropriate ways to apply regularization to sequential decision making are not clear at this stage, and would call for further research. For instance, we observed — independently of the material presented in this chapter — that the early stopping of the progressive hedging algorithm (Rockafellar and Wets, 1991) (see also Remark 2.1), where decisions are optimized on separate scenarios (with a penalization of the difference with the decisions at the previous iteration) and then averaged if they are relative to a same information state, could provide a kind of regularization without even modifying the formulation of the model. With early stopping however, the objective being optimized is no longer totally explicit, the solution has a dependence on the initial conditions, and therefore the solution algorithm would require a careful tuning of its parameters on the problem at hand.

Unfortunately, we are still lacking at this stage efficient methods for testing the real value of any solution procedure, be it regularized or not. As the right amount of regularization (weight of the regularization in the objective, early stopping criterion, ...)

is usually selected at the light of the results obtained by simulating the model on an independent validation sample or by cross-validation methods (Stone, 1974; Efron and Tibshirani, 1993), it would be vacuous to discuss regularization further if we were ultimately unable to estimate the true quality of the regularized solution. The development of efficient validation methods is the subject of the next chapter.

Chapter 4

Validation of Solutions and Scenario Tree Generation

In this chapter, we propose an approach for solving multistage stochastic programming problems based on the idea of generating in a lazy fashion a large number of random tractable scenario-tree based approximations. The approach is lazy in the sense that instead of recommending a careful analysis of the structure of the problem at hand, and instead of devoting all computational resources to the construction of a single scenario tree, we recommend a multiplication of solution attempts through the generation of several approximations. The method works by extracting, from the solutions of these approximations, data sets that combine realizations of the random process and decision sequences, and by processing these data sets by a supervised learning method, so as to infer policies that can be later on tested efficiently on a large sample of new independent scenarios. The learned policies can be exploited to infer multistage decision strategies that achieve good performances in a very generic way. They can also be used to score and select scenario trees, and thus to guide, in the context of a precise application, the development and fine-tuning of a scenario tree generation algorithm.

The chapter is organized as follows. Section 4.1 motivates the approach investigated in the chapter. Section 4.2 describes how feasible decision policies can be learned from a data set of scenarios and decisions. Section 4.3 builds on the idea of exploiting several scenario-tree approximations for learning several policies. Section 4.4 implements the proposed approach on a family of test problems, introduces specific tree generation algorithms, tunes them in the context of the test problems, and discusses the overall complexity of the approach. Section 4.6 concludes by discussing the potential of some possible extensions of the proposed algorithms, at the light of the results and insights obtained in this chapter.

4.1 Motivation

Our approach is motivated by two complementary and intimately related considerations induced by our analysis of current approximation methods for multistage stochastic programming, and their confrontation to the problems addressed in the field of machine learning in the last years.

The first motivation is derived from the need for intensive testing of decision-making policies for multistage programs (Section 2.4.3). This need is primarily a consequence of the lack of tight theoretical results that would provide broadly-usable prior guarantees on scenario tree based methods. Intensive testing is needed, because for obtaining performance estimators that are statistically significant, it is important to test a decision policy on a sufficiently large number of independent scenarios. Testing decisions a posteriori by the shrinking-horizon approach (Section 2.4.4) is not a viable option, given the internal use of additional scenario trees by this approach, and given the overall computational complexity of the procedure. With respect to this motivation, machine learning offers a multitude of ways of extracting policies that are easy to test in an automatic way on a large number of independent samples.

The second motivation has to do with the intrinsic nature of the finite scenario-tree approximation for multistage stochastic programming. The variance in the quality of the optimal decisions that may be inferred from finite approximations suggests that those problems are essentially *ill-posed* in the same sense as the inverse problems addressed in machine learning are also ill-posed: small perturbations in the values of a finite data set — finite number of scenarios in stochastic programming; finite number of input-output pairs in machine learning — lead to perturbations of empirical expectations, and ultimately lead to large variations (instability) of the quantities of interest — first-stage decisions in stochastic programming; parameters of classifiers or regressors in machine learning — that are being optimized on the basis of empirical estimates.

This analogy suggests that regularization techniques and principles from statistical learning theory (Vapnik, 1998), such as the structural risk minimization principle, could help to extract solutions from scenario-tree approximations in a sound way from the theoretical point of view, and in an efficient way from the practical point of view.

The main ideas developed in the following subsections can be summarized as follows: we propose an approach that (i) allows to test small scenario trees quickly and reliably, (ii) is likely to offer better ways of exploiting individual scenario-tree approximations, and (iii) in the end, allows to revisit the initial question (Section 2.4.2) of generating, solving, ranking and exploiting tractable scenario trees for solving complex multistage decision making problems.

4.2 Learning and Evaluation of Scenario Tree Based Policies

We start from the following observation: estimators of the quality of a scenario-tree approximation that are computationally cheap to evaluate can be constructed by resorting to supervised learning techniques.

The basic principle consists in inferring a suboptimal decision policy by first learning a sequence of decision predictors $\hat{\pi}_1, \ldots \hat{\pi}_T$ from a data set of examples of information state/decision pairs. The examples of information states are extracted from the nodes of the scenario tree; they correspond to the partial scenario histories $(\xi_1^k, \ldots, \xi_{t-1}^k)$ in the tree. Later in the chapter (Section 4.4.3), we will see that the information states can also be represented differently, for instance by features or by states in the sense of dynamic programming. The examples of decisions are also extracted from the nodes of the tree: they correspond to the decisions u_t^k optimized on the tree.

When a decision predictor $\hat{\pi}_t$ is applied on a new scenario ξ (or more exactly, on the observed part $(\xi_1, \ldots, \xi_{t-1})$ of the scenario ξ), it outputs a predicted decision that cannot be assumed to satisfy the exact feasibility constraints $u_t \in \mathcal{U}_t(\xi)$ relative to the new scenario, if we want to define a framework that allows the use of existing standard supervised learning algorithms for building the decision predictors. Therefore, to obtain feasible decisions, we assume that the predicted decision can then be corrected in an ad-hoc fashion using a computationally cheap feasibility-restoration procedure, that we call repair procedure in the sequel and denote by M_t. The idea of using repair procedures is also suggested in Küchler and Vigerske (2010) as a means of restoring the feasibility of decisions extracted from a tree and applied to test scenarios.

We now formalize these ideas to describe how a learned decision policy can be used to assess (validate), in a certain sense, a given scenario-tree approximation.

4.2.1 Description of the Validation Method

We consider a multistage stochastic program in abstract form (see Section 2.1.5):

$$\begin{aligned}\mathcal{P}: \quad \text{minimize} \quad & \mathbb{E}\{f(\xi,\pi(\xi))\} \quad \text{subject to} \quad \pi_t(\xi) \in \mathcal{U}_t(\xi) \ ; \\ & \pi(\xi) \text{ non-anticipative,}\end{aligned}$$

where $\xi = (\xi_1, \ldots, \xi_T)$ is a random process, and where the optimization is over the decision policy $\pi = (\pi_1, \ldots, \pi_T)$. We assume that ξ_t has its outcomes in some space Ξ_t, say $\mathbb{R}^d$ for simplicity, and that π_t is valued in some space U_t (of which $\mathcal{U}_t(\xi)$ is a subset), say $\mathbb{R}^m$. We recall that π is non-anticipative if π_1 is a constant-valued function, π_2 is a function of ξ_1, and more generally π_t is a function of $\xi_1, \ldots, \xi_{t-1}$. Therefore, one can define π_t either as a mapping from $\Xi_1 \times \cdots \times \Xi_T = \mathbb{R}^{Td}$ to $\mathbb{R}^m$ restricted to the class of non-anticipative mappings, or as a mapping from $\Xi_1 \times \cdots \times \Xi_{t-1} = \mathbb{R}^{(t-1)d}$ to $\mathbb{R}^m$.

Given an approximation of $\mathcal{P}$ on a scenario tree having n scenarios ξ^k of probability p^k,

$$\begin{aligned}\mathcal{P}': \quad \text{minimize} \quad & \textstyle\sum_{k=1}^n p^k \ f(\xi^k, u^k) \quad \text{subject to} \quad u_t^k \in \mathcal{U}_t(\xi^k) \quad \forall\ k \ ; \\ & u_1^k = u_1^j \quad \forall\ k,j \ , \\ & u_t^k = u_t^j \quad \text{whenever} \quad (\xi_1^k, \ldots, \xi_{t-1}^k) \equiv (\xi_1^j, \ldots, \xi_{t-1}^j) \ ,\end{aligned}$$

let $\{\bar{u}^k\}_{1\le k\le n}$ denote an optimal solution to $\mathcal{P}'$, where each $\bar{u}^k = (\bar{u}_1^k, \ldots, \bar{u}_T^k)$ corresponds to the sequence of decisions associated to ξ^k. We define a decision predictor $\hat{\pi}_t$ as a mapping from inputs $X_t \overset{\text{def}}{=} (\xi_1, \ldots, \xi_{t-1}) \in \mathbb{R}^{(t-1)d}$ to outputs $Y_t \overset{\text{def}}{=} u_t \in \mathbb{R}^m$, learned from a data set $D_t \overset{\text{def}}{=} \{(X_t^k, Y_t^k)\}_{1\le k\le n}$ of input-output pairs, obtained by collecting from the scenario tree the observed parts of the scenarios and their associated optimized decisions:

$$\begin{aligned} X_t^k &\overset{\text{def}}{=} (\xi_1^k, \ldots, \xi_{t-1}^k) \ , \\ Y_t^k &\overset{\text{def}}{=} \bar{u}_t^k \ .\end{aligned}$$

Note that the duplicate samples $(X_t^k, Y_t^k) \equiv (X_t^j, Y_t^j)$ induced by the non-anticipativity conditions (the branching structure of the scenario tree) may be removed from the learning set D_t. In particular, D_1 is reduced to a single learning sample $Y_1 = \bar{u}_1$, leading to a trivial learning problem and to the decision predictor $\hat{\pi}_1 \equiv \bar{u}_1$.

By construction of $\mathcal{P}'$, and by the fact that $\mathcal{U}_1$ is constant-set-valued, the first-stage decision is feasible: $\hat{\pi}_1(\xi) = \bar{u}_1 \in \mathcal{U}_1(\xi)$. For the subsequent decisions, the supervised learning procedure cannot in general guarantee that $\hat{\pi}_t(\xi) \in \mathcal{U}_t(\xi)$ for all scenarios in the learning set and for all new scenarios ξ. Therefore, we repair the predictions to restore the feasibility of the decisions. The nature of the repair procedure varies with the feasibility constraints that have to be enforced. The realizations of the random quantities on which $\mathcal{U}_t(\xi)$ depend are passed in arguments of the repair procedure, and the procedure is then applied online on each new scenario and predicted decisions.

An example of repair procedure is the projection of a predicted decision on the feasibility set. Later in the thesis, we resort to simple problem-dependent heuristics for restoring feasibility (Section 5.3.4). Formally, we define as an admissible repair procedure for $\mathcal{U}_t$ any mapping

$$\begin{aligned} M_t : (\Xi_1 \times \cdots \times \Xi_{t-1}) \times (U_1 \times \cdots \times U_{t-1}) \times U_t \to U_t \\ \text{with values } M_t(\xi_1, \ldots, \xi_{t-1}; u_1, \ldots, u_{t-1}; \hat{\pi}_t(\xi_1, \ldots, \xi_{t-1})) \end{aligned}$$

such that the range of M_t is always contained in the feasible set $\mathcal{U}_t(\xi)$, assuming that $u_1, \ldots, u_{t-1}$ are in the corresponding feasibility sets $\mathcal{U}_1(\xi), \ldots, \mathcal{U}_{t-1}(\xi)$, and that $\mathcal{U}_t(\xi)$ is nonempty.

A learned (feasible) policy is made of the association of the decision predictors and the repair procedures.

We can exploit a learned policy for computing an estimate of the quality of a scenario tree, or a bound on the exact value of the original multistage program $\mathcal{P}$. The procedure can be described as follows.

i. Generate a scenario tree using a tree building algorithm $\mathcal{A}$. Solve the resulting program $\mathcal{P}'$, extract from its solution the first-stage decision $\bar{u}_1$, and the data sets D_t of scenario/decisions pairs.

ii. Learn the decision predictors $\hat{\pi}_t$ from the data set D_t for $t = 2, \ldots, T$.

iii. Generate a test sample of n' mutually independent scenarios $\{\xi^j\}_{1 \leq j \leq n'}$ by sampling realizations of the random process ξ.

iv. For each scenario ξ^j of the test sample, set $u_1^j = \bar{u}_1$ and compute sequentially the recourse decisions $u_2^j, \ldots, u_T^j$. Each decision u_t^j is obtained by first evaluating $\hat{\pi}_t(\xi_1^j, \ldots, \xi_{t-1}^j)$ and then restoring feasibility by the repair procedure M_t.

v. Estimate the performance of the learned decision policy on the test sample by forming the empirical average $V_{\text{TS}}(\mathcal{A}) = (1/n') \sum_{j=1}^{n'} f(\xi^j, u^j)$, where the sum runs over the indices relative to the scenarios in the test sample and their associated decision sequences $u^j = (u_1^j, \ldots, u_T^j)$.

The estimator $V_{\text{TS}}(\mathcal{A})$ computed in this way reflects the joint quality of the scenario tree, the learned predictors and the repair procedures.

The estimator $V_{\text{TS}}(\mathcal{A})$ is obtained by simulating an explicit policy that generates feasible decisions, and thus always provides a pessimistic bound (upper bound for minimization, lower bound for maximization) on the performance of the best policy that could be inferred from the considered scenario tree, up to the standard error of the test sample estimator. The pessimistic bound is also a reliable bound on the achievable performance of a decision policy for the true problem, up to the standard error of the test sample estimator.

Note that in theory, a learned policy is not necessarily worse than a shrinking-horizon policy using the same first-stage decision $\bar{u}_1$, since the supervised learning step could actually improve the quality of the recourse decisions $u_2^j, \dots, u_T^j$.

The pessimistic bound can be made tighter by testing various policies obtained from the same scenario tree, but with different learning algorithms and/or repair procedures. The best combination of algorithms and learning parameters could then be retained. Note, however, that due to the optimistic bias induced by the selection of the best bound on the test sample of size n', the value of the best policy should be evaluated again by simulation on a new independent test sample of size n''.

It is also possible to exploit estimators relative to policies learned from different scenario trees but computed on the same test sample of size n'. We may even expect that scenario tree variants can be ranked reliably based on the value of these estimators, despite the variance of the estimator due to the randomness in the generation of the test sample, and despite a new source of bias due to the use of suboptimal recourse decisions obtained from the learned policies. These ideas will be further developed in Section 4.3.

Note also that the input space of a learned policy is a simple matter of convenience. As long as the policy remains non-anticipative, the input space can be described differently, typically by letting appear explicitly past decisions, state variables, and additional features derived from the information state, that might facilitate the generalization of the decisions in the data sets, or later on, the online evaluation of the learned decision predictors. These ideas are illustrated in Section 4.4.3.

To simplify the exposition in the sequel, we will assume that all the considered algorithms for learning policies use the same repair procedures M_t, and differ only by the choice of the hypothesis space $\mathcal{H}_t$ for $\hat{\pi}_t$ (space of functions considered by the supervised learning algorithm). It is convenient to denote the possible hypothesis spaces by $\mathcal{H}_t^\lambda$, where λ belongs to some index set Λ. For instance, λ could represent the weight of a regularization term used in the supervised learning algorithm. For simplicity, we assume that Λ has a finite cardinality $|\Lambda|$.

4.2.2 Complexity Analysis

In this section, we consider the complexity of computing an upper bound (a performance guarantee) on the value of an exact multistage program $\mathcal{P}$ by simulating a series of policies learned from a single scenario-tree approximation $\mathcal{P}'$.

Recall that we have assumed for simplicity that $u_t \in \mathbb{R}^m$ and $\xi_t \in \mathbb{R}^d$, $1 \leq t \leq T$. We denote by $\bar{u}_1 \in \mathbb{R}^m$ the constant first-stage decision. For $t = 2, \dots, T$, the mappings $\hat{\pi}_t : \mathbb{R}^{(t-1)d} \to \mathbb{R}^m$, with values $\hat{\pi}_t(\xi_1, \dots, \xi_{t-1})$, represent the learned decision predictors, and the mappings $M_t : \mathbb{R}^{(t-1)d} \times \mathbb{R}^{(t-1)m} \times \mathbb{R}^m \to \mathbb{R}^m$, with values $M_t(\xi_1, \dots, \xi_{t-1}, u_1, \dots, u_{t-1}, u_t)$, represent the repair procedures.

Then the mappings $\bar{\pi}_t : \mathbb{R}^{dT} \to \mathbb{R}^m$, with values $\bar{\pi}_t(\xi)$, defined iteratively for $t = 1, \dots, T$ by

$$\begin{aligned} \bar{\pi}_1(\xi) &= \bar{u}_1 \ , \\ \bar{\pi}_t(\xi) &= M_t(\xi_1, \dots, \xi_{t-1}; u_1, \dots, u_{t-1}; \hat{\pi}_t(\xi_1, \dots, \xi_{t-1})) = u_t \ , \end{aligned}$$

correspond to a non-anticipative feasible decision policy $\bar{\pi} = (\bar{\pi}_1, \dots, \bar{\pi}_T)$ for the original

Algorithm 4.1 Selection and evaluation of a decision policy

```
Input:  A first-stage decision $\bar{u}_1$, learning sets $D_t$ of pairs $(X_t^k, Y_t^k)$, hy-
        pothesis spaces $\mathcal{H}_t^\lambda$ indexed by $\lambda \in \Lambda$, repair procedures $M_t$, and
        a test sample of new scenarios (size $n'$).
Output: A feasible policy $\bar{\pi}$.

1. For each $\lambda \in \Lambda$,
   learn the decision predictors $\hat{\pi}_t^\lambda$ given the data sets $D_t$,
   using the hypothesis spaces $\mathcal{H}_t^\lambda$,   $t = 1, \ldots, T$.

2. For each $\lambda \in \Lambda$,
   evaluate the performance of the policy $\bar{\pi}^\lambda$ obtained by combining $\hat{\pi}_t^\lambda$ with $M_t$.
   Let $v^\lambda$ be that performance evaluated on the common test sample of size $n'$.

3. Select $\nu \in \arg\min_{\lambda \in \Lambda} v^\lambda$ and return $\bar{\pi}^\nu$.

4. Optional: return the value of $v^\nu$ reevaluated on
             an independent test sample of size $n''$.
```

program $\mathcal{P}$.

The computational complexity of exploiting $\bar{\pi}_t$ on new scenarios depends on the complexity of evaluating $\hat{\pi}_t$ and M_t for all t.

The mappings $\hat{\pi}_t$ should ideally be the best mappings from the best hypothesis spaces one could consider, but in practice they correspond to the mappings identified by a given supervised learning algorithm on the basis of the data sets D_t. We find it useful to consider a series of policies in this section, because there is some leeway in the choice of the supervised learning algorithm and/or its parameters, that can be exploited in the search for ideal mappings.

In the usual supervised learning framework, one generally selects a model by evaluating its performance on a fraction of the data set kept apart for testing purpose. In the present setup, it is preferable to evaluate models by directly simulating the learned policy $\bar{\pi}$ on a common test sample of new scenarios (Algorithm 4.1).

If Algorithm 4.1 is merely run to select a best learned policy, a single test sample of size n' on which the policies are compared suffices. If in addition an unbiased upper bound on the exact value of $\mathcal{P}$ is sought, an additional independent test sample of size n'' is required on which the best policy should be simulated again.

In practice, the selection bias may be very small if n' is large enough with respect to the considered hypothesis spaces. Therefore, in some numerical experiments, we sometimes allow ourselves to report directly the estimates obtained on the first test sample of size n'.

To discuss the complexity of Algorithm 4.1, let us introduce the following quantities.

- $c_{\mathcal{A}}$: expected time for forming the approximation $\mathcal{P}'$ to $\mathcal{P}$ using a scenario tree building algorithm $\mathcal{A}$,

- c_S: expected time for obtaining a solution to $\mathcal{P}'$,

- $c_L(t)$: expected running time of the learning algorithm on data D_t,
- $c_E(t)$: expected running time of the combined computation of $\hat{\pi}_t$ and M_t on a new scenario.

We assume that $c_L(1) = c_E(1) = 0$ since the first decision $\bar{u}_1$ is fixed and simply extracted from a solution to $\mathcal{P}'$. For $t \geq 2$, note that $c_L(t)$ and $c_E(t)$ usually grow with the dimension of the random variables ξ_t, the dimension of the decisions u_t, and the cardinality of the data sets D_t. The ratio between $c_L(t)$ and $c_E(t)$ depends largely on the type of supervised learning algorithm and the type of repair procedure for M_t. We neglect the time for computing $f(\xi, u)$ given ξ and u.

The following proposition is a straightforward consequence of the definition of Algorithm 4.1:

4.1 Proposition. *Algorithm 4.1 runs in expected time*

$$|\Lambda| \cdot \left[\sum_{t=2}^{T} c_L(t) + n' \sum_{t=2}^{T} c_E(t) \right] = \sum_{t=2}^{T} |\Lambda| \cdot [c_L(t) + n'\, c_E(t)] \quad ,$$

starting from data sets obtained in expected time $c_{\mathcal{A}} + c_S$. The optional step of Algorithm 4.1 adds to the expected time a term $n'' \sum_{t=2}^{T} c_E(t)$.

If Algorithm 4.1 is run on N parallel processes, one can essentially replace $|\Lambda|$ in Proposition 4.1 by $|\Lambda|/N$, and n'' by n''/N.

The complexity of Algorithm 4.1 can be compared to the complexity of the usual shrinking-horizon validation approach (Section 2.4.4). To this end, we extend our notations as follows.

- $\mathcal{P}(t)$ denotes the program for the minimization of the objective over the remaining stages $t, t+1, \dots, T$. The program $\mathcal{P}(1)$ is the original program $\mathcal{P}$. Given realizations for $\xi_1, \dots, \xi_{t-1}$ and the corresponding implemented decisions $\bar{u}_1, \dots, \bar{u}_{t-1}$, one can obtain $\mathcal{P}(t)$ by replacing in $\mathcal{P}$ the random variables $\xi_1, \dots, \xi_{t-1}$ by their outcomes, conditioning the distribution of $\xi_t, \dots, \xi_T$ accordingly, and introducing the constraints $\pi_1(\xi) = \bar{u}_1, \dots, \pi_{t-1}(\xi) = \bar{u}_{t-1}$.
- $\mathcal{P}'(t)$ denotes a scenario-tree approximation to $\mathcal{P}(t)$, built by some algorithm $\mathcal{A}(t)$.
- $c_{\mathcal{A}}(t)$ denotes the expected time for forming the approximation $\mathcal{P}'(t)$ to $\mathcal{P}(t)$ using algorithm $\mathcal{A}(t)$ for building a scenario tree over the shrunk horizon.
- $c_S(t)$ denotes the expected time for obtaining a solution to $\mathcal{P}'(t)$.

If the tree building algorithm $\mathcal{A}(t)$ is based on a pure Monte Carlo sampling, $c_{\mathcal{A}}(t)$ should be relatively small, and approximately proportional to the size of the scenario tree. If $\mathcal{A}(t)$ is based on a deterministic method and the dimension of the random process is not say 1 or 2, $c_A(t)$ may actually be quite large, even for t near the horizon T. The time $c_S(t)$ can also be quite large, except perhaps for $t = T$ or t close to T.

In Section 2.4.4, we had denoted all the algorithms $\mathcal{A}(t)$ simply by $\mathcal{A}$, and written $V_{\mathrm{TS}}(\mathcal{A})$ for the estimate produced by the shrinking-horizon validation approach. In the following proposition, we assume that the shrinking-horizon approach is run on the independent test sample of size n'' used for reevaluating the best policy selected by Algorithm 4.1.

4.2 Proposition. *The shrinking-horizon approach runs in expected time*

$$\sum_{t=2}^{T} n'' \cdot [c_{\mathcal{A}}(t) + c_S(t)] \ ,$$

using a first-stage decision obtained in expected time $c_{\mathcal{A}} + c_S$.

(The use of N parallel processes allows to replace n'' by n''/N).

Note that when the scenario tree building algorithms $\mathcal{A}(t)$ are not deterministic, each algorithm $\mathcal{A}(t)$ constitutes a new source of variance for the shrinking-horizon estimate. These new sources of variance can greatly affect the sample size n'' that would be needed to obtain a meaningful estimate.

The comparison of the complexity estimates stated in Propositions 4.1 and 4.2 suggests that Algorithm 4.1 should be far more tractable than the shrinking-horizon validation strategy, provided that $|\Lambda|$ is kept under control, and $c_E(t)$ is small enough.

4.2.3 Other Validation Strategies

In this section, we mention variants of the validation approaches discussed above, motivated by the complexity estimates of Propositions 4.1 and 4.2. These variants are interesting to consider, but their implementation has been left as future work.

We begin by observing that it is possible to combine the two preceding validation approaches (supervised learning of policies and shrinking-horizon optimization) by combining learned policies at stages $2, \dots, t_0$ to a shrinking-horizon decision making procedure for $t = t_0 + 1, \dots, T$. This hybrid approach would run in expected time

$$\textstyle\sum_{t=2}^{t_0} |\Lambda| \cdot [c_L(t) + n'\, c_E(t)] + \sum_{t=t_0+1}^{T} n' \cdot [c_{\mathcal{A}}(t) + c_S(t)] \ ,$$

starting from data obtained in expected time $c_{\mathcal{A}} + c_S$. We would also add to the expected time the term $n'' \cdot [\sum_{t=2}^{t_0} c_E(t) + \sum_{t=t_0+1}^{T} [c_{\mathcal{A}}(t) + c_S(t)]]$, relative to the reevaluation of the selected hybrid policy on the test sample of size n''.

The number of stage t_0 could be chosen to minimize the expected running time, namely (neglecting the reevaluation term)

$$t_0 = \sup\{t \le T : \ |\Lambda| \cdot [c_L(t) + n'\, c_E(t)] \le n' \cdot [c_{\mathcal{A}}(t) + c_S(t)]\} \ ,$$

but a complication with the optimal choice of t_0 is the possible dependence of the standard error of the estimates on nondeterministic algorithms $\mathcal{A}(t)$.

Another possible variant is to carry out the selection of the models for $\hat{\pi}_1, \dots, \hat{\pi}_T$ sequentially, that is, stage by stage. To describe this variant, we extend our notations as follows.

- The index $\lambda \in \Lambda$ is replaced by indices $(\lambda_1, \dots, \lambda_T) \in \Lambda_1 \times \cdots \times \Lambda_T$. This allows to denote by $\mathcal{H}_t^{\lambda_t}$ a hypothesis space for the predictor $\hat{\pi}_t$, with the choice of $\lambda_t \in \Lambda_t$ decoupled from the previous choices of $\lambda_1, \dots, \lambda_{t-1}$.

Algorithm 4.2 Stage by stage selection and evaluation of a decision policy

Input: A first-stage decision $\bar{u}_1$, a data set D_t of pairs (X_t, Y_t) for $t = 2$ only, hypothesis spaces $\mathcal{H}_t^{\lambda_t}$ indexed by $\lambda_t \in \Lambda_t$, repair procedures M_t, and a test sample of new scenarios (size n').
Output: A feasible policy $\bar{\pi}$.

```
1. Set $\bar{\pi}_1(\xi) = \bar{u}_1$, and then set $t = 2$.

2. For each $\lambda_t \in \Lambda_t$,
   learn the predictor $\hat{\pi}_t^{\lambda_t}$ given the data set $D_t$, using the hypothesis space $\mathcal{H}_t^{\lambda_t}$;
   build the policy $\pi^{\lambda_t} = (\bar{\pi}_1, \dots, \bar{\pi}_{t-1}, \pi_t^{\lambda_t})$, where $\pi_t^{\lambda_t}$ combines $\hat{\pi}_t^{\lambda_t}$ and $M_t$.
   If $t = T$, go to Step 4.

3. For each $\lambda_t \in \Lambda_t$,
   form and solve the problem $\mathcal{P}'_+(t; \pi^{\lambda_t})$; let $v^{\lambda_t}$ denote its optimal value.
   Set $\nu_t \in \mathrm{argmin}_{\lambda_t \in \Lambda_t} v^{\lambda_t}$ and set $\bar{\pi}_t = \pi_t^{\nu_t}$.
   Form the data set $D_{t+1}$ relative to $u_{t+1}$ from the solution to $\mathcal{P}'_+(t; \pi^{\nu_t})$.
   Set $t$ to $t + 1$ and go to Step 2.

4. For each $\lambda_T \in \Lambda_T$,
   evaluate the performance of $\pi^{\lambda_T}$ on the common test sample of size $n'$;
   let $v^{\lambda_T}$ be that performance.

5. Set $\nu_T \in \mathrm{argmin}_{\lambda_T \in \Lambda_T} v^{\lambda_T}$ and set $\bar{\pi}_T = \pi_T^{\nu_T}$. Return $\bar{\pi} = (\bar{\pi}_1, \dots, \bar{\pi}_T)$.

6. Optional: return the value of $v^{\nu_T}$ reevaluated on
             an independent test sample of size $n''$.
```

- Given $t < T$ and a policy $\bar{\pi}^\dagger = (\bar{\pi}_1, \dots, \bar{\pi}_t)$ specified only from stage 1 to stage t, the notation $\mathcal{P}_+(t; \bar{\pi}^\dagger)$ refers to the original program $\mathcal{P}$ over a policy π, subject to the additional constraints $\pi_1(\xi) = \bar{\pi}_1(\xi)$, $\dots$, $\pi_t(\xi) = \bar{\pi}_t(\xi)$. Thus, the program $\mathcal{P}_+(t; \bar{\pi}^\dagger)$ is the original problem $\mathcal{P}$, except that $\bar{\pi}_1, \dots, \bar{\pi}_t$ are already specified.

- $\mathcal{P}'_+(t; \bar{\pi}^\dagger)$ denotes the scenario-tree approximation to $\mathcal{P}_+(t; \bar{\pi}^\dagger)$ built by some algorithm $\mathcal{A}(t)$. The algorithm $\mathcal{A}(t)$ must always return the same tree, while the trees relative to $\mathcal{A}(1), \dots, \mathcal{A}(T-1)$ must all be different. Thus, $\mathcal{P}'_+(t; \bar{\pi}^\dagger)$ is the approximate program $\mathcal{P}'$ posed over a new scenario tree proper to t, and subject to the additional constraints $u_1^k = \bar{\pi}_1(\xi^k)$, $\dots$, $u_t^k = \bar{\pi}_t(\xi^k)$ for all k.

Algorithm 4.2 describes how decision predictors are learned from data sets that incorporate the effect of the decision rules already selected for the previous stages, and left unspecified for the subsequent stages. At each stage t, there is also a selection step among possible decision predictors indexed by $\lambda_t \in \Lambda_t$.

Indeed, the advantage of Algorithm 4.2 over Algorithm 4.1 is that the learning problem for a decision predictor for stage $t + 1$ takes into account the learned decisions rules $\bar{\pi}_1, \dots, \bar{\pi}_t$. As the learned decision rules introduce a loss of optimality and modify the information states that can be reached at stage $t + 1$, other recourse decisions at stages $t + 1$, $\dots$, T are preferable, and in fact, the ideal recourse decisions are those that would be obtained by solving the problem $\mathcal{P}_+(t; \bar{\pi}^\dagger)$ with $\bar{\pi}^\dagger$ suitably defined (see Step 2 of Algorithm 4.2, where π^{λ_t} plays the role of $\bar{\pi}^\dagger$). We cannot solve $\mathcal{P}_+(t; \bar{\pi}^\dagger)$, but we can exploit a scenario-tree approximation $\mathcal{P}'_+(t; \bar{\pi}^\dagger)$ from which a data set D_{t+1} for learning $\hat{\pi}_{t+1}$ can constructed (Step 3 of Algorithm 4.2).

From the statistical point of view, the main drawback of Algorithm 4.2 is that it cannot evaluate on the test sample of size n' a policy that is not specified on the full horizon. Instead, Step 3 in Algorithm 4.2 performs a weak form of model selection by scoring the incomplete policies of Step 2 with the optimal value of the programs $\mathcal{P}'_+(t; \bar{\pi}^\dagger)$. The programs $\mathcal{P}'_+(t; \bar{\pi}^\dagger)$ use a common scenario tree independent of the trees relative to $\bar{\pi}_1, \dots, \bar{\pi}_{t-1}$, so as to reduce the selection bias. The selection is weak in the sense that the score of an incomplete specified policy $\bar{\pi}^\dagger$ is not a reliable estimate of the optimal value of the exact program $\mathcal{P}_+(t; \bar{\pi}^\dagger)$. An unbiased upper bound on the exact value of $\mathcal{P}$ can be obtained by the optional Step 6 of Algorithm 4.2.

From the complexity point of view, the main drawback of Algorithm 4.2 is that the programs $\mathcal{P}'_+(t; \bar{\pi}^\dagger)$ must be solved for each $\lambda_t \in \Lambda_t$, $t = 2, \dots, T$. Another concern is the new source of variance of the test sample estimates coming from use of several scenario trees, that could force us to use larger test samples.

4.3 Monte Carlo Selection of Scenario Trees

We now sketch a workable and generic scheme for obtaining approximate solutions to a multistage stochastic program with performance guarantees, and for selecting good scenario-tree approximations to the multistage stochastic program. The scheme builds on the validation procedure described in Section 4.2.1 (Algorithm 4.1), which infers a decision policy from examples of scenarios and decisions collected from a scenario-tree approximation, and also computes an accurate estimate of the value of the learned policy by Monte Carlo simulation.

A first idea simply consists in perturbing the data sets D_t of scenario/decisions pairs used by the supervised learning procedure, by obtaining these data sets from different scenario-tree approximations. This source of variation creates new opportunities for finding better policies by supervised learning.

A second idea consists in identifying good scenario trees, on the basis of the performance of the policies that can be learned from the data sets D_t collected from those trees. This approach allows to study empirically algorithms that construct the scenario-tree approximations, and to tune or modify these algorithms so as to improve the solution procedure in terms of solution accuracy or in terms of computational complexity.

4.3.1 Description of the Selection Scheme

In this section, we describe the scheme that allows to identify good scenario trees. The scheme consists in generating a possibly large set of randomized scenario-tree approximations $\mathcal{P}'$ for a given problem $\mathcal{P}$, ranking them according to the estimated value of the best policy learned from them, and identifying in this way a presumably best scenario tree among the considered sample of trees. The best policy of the best scenario tree is then viewed as the best solution for $\mathcal{P}$ found by the method, and its value can be assessed

Algorithm 4.3 Monte Carlo selection of scenario trees

Input: Algorithms $\mathcal{A}_S$ for building random tree structures and $\mathcal{A}_V$ for building the trees for the process ξ given a branching structure, and 3 independent test samples of size n', n'', n''', made of realizations of ξ sampled independently.
Output: A scenario tree, a feasible policy $\bar{\pi}$, a performance guarantee.

```
1. Generate a set $\mathcal{T}$ of scenario trees, using algorithms $\mathcal{A}_S$ and $\mathcal{A}_V$.

2. For each tree in the set $\mathcal{T}$ indexed by $1 \leq \nu \leq M$,
   select a best policy $\pi^\nu$ learned from the data set extracted from the tree,
   using Algorithm 4.1 on the first test sample of size $n'$.

3. For each policy $\pi^\nu$,
   reassess the performance of $\pi^\nu$ on the second test sample of size $n''$.
   Let $v^\nu$ denote that performance.

4. Set $\mu \in \operatorname{argmin}_{1 \leq \nu \leq M} v^\nu$.
   Return the scenario tree indexed by $\mu$ and the policy $\bar{\pi} = \pi^\mu$.

5. Reevaluate $\pi^\mu$ on the third test sample of size $n'''$.
   Let $V^\mu$ denote that performance.
   Return the bound $\min_\pi \mathbb{E}\{f(\xi, \pi(\xi))\} \leq \mathbb{E}\{f(\xi, \bar{\pi}(\xi))\} \simeq V^\mu$ (see text for details).
```

by Monte Carlo simulation on an independent test sample. Algorithm 4.3 describes each step of the procedure.

Having enough diversity in the considered scenario-tree approximations multiplies our chance of obtaining good data sets, from which good policies can be learned. Therefore, it is interesting to assume that the generated scenario trees have a random branching structure — a novelty with respect to the usual practice of multistage stochastic programming. In our presentation of Algorithm 4.3, we formally decompose a tree generation algorithm $\mathcal{A}$ into 2 components: $\mathcal{A}_S$ for generating a random branching structure, and $\mathcal{A}_V$ for sampling realizations ξ^k of the random process ξ according to the fixed branching structure and for assigning probabilities to the nodes of the tree. Existing tree generation methods from the stochastic programming literature, briefly discussed in Section 2.4.2, correspond to a particular choice of $\mathcal{A}_V$.

Developing algorithms $\mathcal{A}_S$ able to generate rich but tractable branching structures, for low-dimensional processes or for high-dimensional processes, valid for short horizons and long horizons, is an interesting open problem. We have investigated several variants for $\mathcal{A}_S$ in the context of a concrete family of problems (Section 4.4.2), without however providing general-purpose algorithms for generating random branching structures adapted to high-dimensional random processes.

Algorithm 4.3 uses in theory 3 independent test samples of size n', n'', n''': one for selecting a best hypothesis space for the best learned policy from the data sets relative to a given tree; one for selecting the best tree; and one for estimating the performance of the overall best policy. In practice, we do not always reevaluate our estimates on distinct independent test samples.

The tree and the policy that Algorithm 4.3 returns can be exploited in several ways. We have already mentioned the possibility of tuning the scenario tree generation algorithms to the problem at hand. Another possibility is to implement only the first-stage decision $\bar{\pi}_1$ of the policy returned by Algorithm 4.3, and therefore to see the whole procedure as a single step of a general shrinking-horizon or receding-horizon decision making scheme.

It is also possible to use the policy $\bar{\pi}$ on the full horizon, with a performance guarantee, since $\bar{\pi}$ should achieve, for the original problem $\mathcal{P}$, an objective value close to the performance guarantee estimated on the independent test sample of size n'''. More precisely, the variance of the empirical estimate $V^{\mu} = \frac{1}{n'''} \sum_{j=1}^{n'''} f(\xi^j, \pi(\xi^j))$ in Step 5 of Algorithm 4.3 should be approximately equal to $\hat{\sigma}^2 = \frac{1}{n'''(n'''-1)} \sum_{j=1}^{n'''} [f(\xi^j, \pi(\xi^j)) - V^{\mu}]^2$, so that $V^{\mu} + z_{\alpha}\hat{\sigma}$ would yield a conservative estimate of the performance of $\bar{\pi}$ with confidence $1-\alpha$, where z_{α} is the α-critical value of the standard normal distribution (Shapiro et al., 2009, Section 5.6).

We could also mention that when the only information on the random process ξ is a finite set of realizations ξ^j, the tree selection method could be extended as follows. One would split the set of realizations into a test set and a learning set from which a generative model for simulating realizations of the random process would be inferred. The random scenario trees would then be built by querying new samples from the generative model.

4.3.2 Discussion

The generic procedure presented in this section is based on various open ingredients that may be exploited for the design of a wide class of algorithms in a flexible way. Namely, the main ingredients are (i) the scenario tree sampling scheme, (ii) the (possibly regularized) optimization technique used to obtain data sets from a scenario tree, (iii) the supervised learning algorithm used to obtain the decision strategies from the data sets, (iv) the repair procedure used to restore the feasibility of the decisions on new scenarios.

The main ideas of the proposed scheme are evaluated in the case study section on a family of problems proposed by other authors. We illustrate how one may adjust the scenario tree generation algorithm and the policy learning algorithm to one's needs, and by doing so we also illustrate the flexibility of the proposed approach and the potential of the combination of scenario-tree based decision making with supervised learning. In particular, the efficiency of supervised learning strategies makes it possible to rank large numbers of policies inferred from large numbers of randomly generated scenario trees.

Although we do not illustrate this in the present work, we would like also to stress that the scenario tree sampling scheme may be coupled in various other ways with the inference of policies by machine learning. For example, one could seek to use sequential Monte Carlo techniques inspired from the importance sampling literature, in order to progressively guide the scenario tree sampling and machine learning methods towards regions of high interest, given the quality of the policies inferred from scenarios trees at previous iterations. Also, instead of using the data set obtained from each scenario tree to extract a policy, one could use data sets collecting data from several scenario-tree approximations to extract a single policy, in the spirit of the wide range of model perturbation and combination schemes reviewed in chapter 3.

4.4 *Case Study*

We will show the interest of the approximate solution techniques presented in the chapter by applying them to a family of multistage stochastic programs. Implementation choices difficult to discuss in general terms, such as choices concerning the supervised learning of a policy for the recourse decisions, and the choices for the random generation of the trees, will be illustrated on a concrete case.

The section starts by the formulation of a multistage stochastic program that various researchers have presented as difficult for scenario tree methods (Hilli and Pennanen, 2008; Koivu and Pennanen, 2010; Küchler and Vigerske, 2010). Several instances of the problem will be addressed, including instances on horizons considered as almost unmanageable by scenario tree methods.

4.4.1 Description of the Problem

We consider a multistage problem adapted from Hilli and Pennanen (2008), interpreted in that paper as the valuation of an electricity swing option. In this chapter, we interpret the problem rather as the search for risk-aware strategies for distributing the sales of a commodity over T stages in a flexible way adapted to market prices. A risk-aware objective is very interesting for our purposes, but it is difficult to justify it in a context of option valuation. The formulation of the problem is as follows:

$$\begin{array}{ll} \text{minimize} & \rho^{-1} \log \mathbb{E}\{\exp\{-\rho \sum_{t=1}^{T} \xi_{t-1} \cdot \pi_t(\xi)\}\} \\ \text{subject to} & 0 \leq \pi_t(\xi) \leq 1 \text{ and } \sum_{t=1}^{T} \pi_t(\xi) \leq Q \ , \\ & \pi \ \text{ non-anticipative.} \end{array} \tag{4.1}$$

The objective uses the exponential utility function, with risk aversion coefficient ρ. Such objectives are discussed at the end of the chapter.

In our formulation of the problem, there is no constant first-stage decision to optimize. We begin directly by the observation of ξ_0, followed by a recourse decision $u_1 = \pi_1(\xi_0)$. Observations and decisions are intertwined so that in general $u_t = \pi_t(\xi_0, \ldots, \xi_{t-1})$. The random variable ξ_{t-1} is the unitary profit ($\xi_{t-1} > 0$) or loss ($\xi_{t-1} < 0$) that can result from the sale of the commodity at time t. Potential profits and losses fluctuate in time, depending on market conditions (we later select a random process model for market prices to complete the problem specification). The commodity is sold in quantity $u_t = \pi_t(\xi_0, \ldots, \xi_{t-1})$ at time t, meaning that the quantity u_t can depend on past and current prices. The decision is made under the knowledge of the potential profit or loss at time t, given by $\xi_{t-1} \cdot u_t$, but under uncertainty of future prices. This is by the way why scenario tree techniques must be used with great care on this problem when the planning horizon is long: as soon as the scenarios cease to have branchings, there is no more residual uncertainty on future prices, and the optimization process wrongly identifies opportunities anticipatively. Those spurious future opportunities may significantly degrade the quality of previous decisions.

We seek strategies where the sales per stage are bounded (constraint $0 \leq \pi_t(\xi) \leq 1$). The constraint can model a bottleneck in the production process. Notice also that bounded sales are consistent with the model assumption of an exogenous random process: very large sales are more likely to influence the market prices on long planning horizons. The scalar Q bounds the total sales (we assume $Q \geq 1$). It represents the initial stock of commodity, the sale of which must be distributed optimally over the horizon T.

When the risk aversion coefficient ρ tends to 0, the problem reduces to the search of a risk-neutral strategy. This case has been studied by Küchler and Vigerske (2010). It admits a linear programming formulation:

$$\begin{array}{ll} \text{minimize} & -\mathbb{E}\{\sum_{t=1}^{T} \xi_{t-1} \cdot \pi_t(\xi)\} \\ \text{subject to} & 0 \leq \pi_t(\xi) \leq 1 \text{ and } \sum_{t=1}^{T} \pi_t(\xi) \leq Q \ , \\ & \pi \ \text{non-anticipative,} \end{array} \tag{4.2}$$

and an exact analytical solution (which thus serves as a reference)

$$\pi_t^{\mathrm{ref}}(\xi) = \begin{cases} 0 & \text{if } t \leq T - Q \text{ or } \xi_{t-1} \leq 0 \ , \\ 1 & \text{if } t > T - Q \text{ and } \xi_{t-1} > 0 \ . \end{cases} \tag{4.3}$$

- In a first series of experiments, we will take the numerical parameters and the process ξ selected in Hilli and Pennanen (2008) (to ease the comparisons): $\rho = 1$, $T = 4$, $Q = 2$; $\xi_t = (\exp\{b_t\} - K)$ where $K = 1$ is the fixed cost (or the strike price, when the problem is interpreted as the valuation of an option) and b_t is a random walk: $b_0 = \sigma\, \epsilon_0$, $b_t = b_{t-1} + \sigma\, \epsilon_t$, with $\sigma = \sqrt{0.2}$ and ϵ_t following a standard normal distribution $\mathcal{N}(0,1)$.

  Noting that a priori $b_t = \sigma \sum_{t'=0}^{t} \epsilon_{t'}$ is normally distributed with mean 0 and variance $(t+1)\sigma^2$, we record, for future reference, that the first process ξ is such that

  $$\frac{1}{\sigma\sqrt{t}} \log(\xi_{t-1} + K) \quad \text{is a priori distributed as } \mathcal{N}(0,1) \ , \tag{4.4}$$
  $$\text{where } \sigma = \sqrt{0.2} \text{ and } K = 1 \ .$$

- In a second series of experiments over various values of the parameters (ρ, T, Q) with T up to 52, we will take for ξ the process selected in Küchler and Vigerske (2010) (because otherwise on long horizons the price levels of the first process blow out in an unrealistic way, making the problem rather trivial): $\xi_t = \xi'_t - K$ with $\xi'_t = \xi'_{t-1} \exp\{\sigma\epsilon_t - \sigma^2/2\}$ where $\sigma = 0.07$, $K = 1$, and ϵ_t following a standard normal distribution. Equivalently $\xi_t = (\exp\{b_t - (t+1)\ \sigma^2/2\} - K)$ with b_t a random walk such that $b_0 = \sigma\, \epsilon_0$ and $b_t = b_{t-1} + \sigma\, \epsilon_t$.

  We record, for future reference, that the second process ξ is such that

  $$\frac{1}{\sigma\sqrt{t}} \log(\xi_{t-1} + K) + \frac{\sigma\sqrt{t}}{2} \quad \text{is a priori distributed as } \mathcal{N}(0,1) \ , \tag{4.5}$$
  $$\text{where } \sigma = \sqrt{0.07} \text{ and } K = 1 \ .$$

4.4.2 Algorithms for Generating Small Scenario Trees

At the heart of tree selection procedure relies our ability to generate scenario trees reduced to a very small number of scenarios, with interesting branching structures. As the trees are small, they can be solved quickly and then scored using the supervised learning policy inference procedure. Fast testing procedures make it possible to rank large numbers of random trees.

The generation of random branching structures has not been explored in the classical stochastic programming literature; we thus have to propose a first family of algorithms in this section. The algorithms are developed with our needs in view, with the feedback provided by the final numerical results of the tests, until results on the whole set of considered numerical instances suggest that a particular algorithm suffices for the application at hand. We believe that the main ideas behind the algorithms will be reused in subsequent work for addressing the representation of stochastic processes of higher dimensions. Therefore, in the following explanations we put more emphasis on the methodology we followed than on the final resulting algorithms.

Method of Investigation.

The branching structure is generated by simulating the evolution of a branching process. We will soon describe the branching process that we have used, but observe first that the probability space behind the random generation of the tree structure is not at all related to the probability space of the random process that the tree approximates. It is the values and probabilities of the nodes that are later chosen in accordance to the target probability distribution, either deterministically or randomly, using any new or existing method.

For selecting the node values, we have tested different deterministic quantizations of the one-dimensional continuous distributions of random variables ξ_t, and alternatively different quantizations of the gaussian innovations ϵ_t that serve to define $\xi_t = \xi_t(\epsilon_t)$, as described by the relations given in the previous section. Namely, we have tested the minimization of the quadratic distortion (Pages and Printems, 2003) and the minimization of the Wasserstein distance (Hochreiter and Pflug, 2007). On the considered problems we did not notice significant differences in performance attributable to a particular deterministic variant.

What happened was that with deterministic methods, performances began to degrade as the planning horizon was increased, perhaps because trying to preserve statistical properties of the marginal distributions ξ_t distorts other statistics of the joint distribution of $(\xi_0, \ldots, \xi_{T-1})$, especially in higher dimensions. Therefore, for treating instances on longer planning horizons, we switched to a crude Monte Carlo sampling for generating node values.

By examining trees with the best scores in the context of the present family of problems, we observed that the empirical estimates of several statistics of the random process, that were computed from the values and probabilities of the nodes of these scenario trees, could be very far from values consistent with the exact model of the random process. For instance, even the empirical first moments $\sum_{k=1}^{N} p^k \xi_t^k$ could be very far from their theoretical values $\mathbb{E}\{\xi_t\}$. This observation might suggest that it is very difficult to predict, without any information on the optimal solutions, which properties should be preserved in small scenario trees, and thus which objective should be optimized when attempting to build a small scenario tree. If we had discovered a correlation between some features of the trees and the scores, we could have filtered out bad trees without actually solving the programs associated to these trees, simply by computing the identified features.

Description of the Branching Processes.

We now describe the branching process used in the first series of experiments, made with deterministic node values. Let $r \in [0, 1]$ denote a fixed probability of creating a branching. We start by creating the root node of the tree (depth 0), to which we assign the conditional probability 1. With probability r, we create 2 successor nodes to which we assign the values ± 0.6745 and the conditional probabilities 0.5 (see Remark 4.1 below). With probability $(1 - r)$ we create instead a single successor node to which we assign the value 0 and the conditional probability 1; this node is a degenerate approximation of the distribution of ϵ_t. Then we take each node of depth 1 as a new root and repeat the process of creating 1 or 2 successor nodes to these new roots randomly. The process is further repeated on the nodes of depth $2, \ldots, T - 1$, yielding a tree of depth T for representing the original process $\epsilon_0, \ldots, \epsilon_{T-1}$. The scenario tree for ξ is derived from the scenario tree for ϵ.

Remark 4.1 *(Wasserstein distance).* The discrete distribution that assigns probabilities 0.5 on the values ± 0.6745 is the discrete distribution with support of cardinality 2 that has the smallest Wasserstein distance l_1 to the normal distribution $\mathcal{N}(0, 1)$ followed by ϵ_t. The Wasserstein distance l_1 may be defined as follows. Let X, Y be random variables following marginal distributions G and H respectively. Assume that G and H are such that X and Y have finite first moments. Let $\mathcal{P}$ denote the collection of coupling measures between X and Y, that is, the collection of probability measures $\mathbb{P}$ such that X follows G, and Y follows H. Then the Wasserstein distance l_1 between G and H is defined as $l_1(G, H) = \inf_{\mathbb{P} \in \mathcal{P}} \{\mathbb{E}\{|X - Y|\}\}$. It admits a dual representation $l_1(G||H) = \sup_{f \in \mathcal{F}_1} \{\mathbb{E}\{f(X)\} - \mathbb{E}\{f(Y)\}\}$, where $\mathcal{F}_1 = \{f : \mathbb{R} \to \mathbb{R} : \quad |f(x) - f(y)| \leq |x - y|\}$ denotes the class of functions with Lipschitz constant 1. It can be shown that the distribution with values y^k and probabilities p^k, $1 \leq k \leq N$, closest in the l_1 sense to a density g (with respect to the Lebesgue measure) can be computed as follows: Set $y^0 = -\infty$, $y^{N+1} = +\infty$ and minimize over $y^1 < y^2 < \cdots < y^N$ the sum

$$\sum_{k=1}^{N} \int_{(y^{k-1}+y^k)/2}^{(y^k+y^{k+1})/2} |x - y^k| g(x) \mathrm{d}x \ , \qquad \text{and then set } p^k = \int_{(y^{k-1}+y^k)/2}^{(y^k+y^{k+1})/2} g(x) \mathrm{d}x \ .$$

For the case $N = 2$ and g the density of $\mathcal{N}(0, 1)$, one can use a symmetry argument and then evaluate $\mathrm{argmin}_y \int_0^\infty |x - y| (2\pi)^{-1/2} \exp\{-x^2/2\} \mathrm{d}x \simeq 0.6745$. □

For problems on larger horizons, it is difficult to keep the size of the tree under control with a single fixed branching parameter r — the number of scenarios would have a large variance. Therefore, in the second series of experiments (made with random node values), we used a slightly more complicated branching process, by letting the branching probability r depend on the number of scenarios currently developed (Algorithm 4.4). Specifically, let N be a target number of scenarios and T a target depth for the scenario tree with the realizations of ξ_t relative to depth $t + 1$. Let n_t be the number of parent

Algorithm 4.4 Branching structure generation for the second series of experiments

Input: A targeted number of scenarios $N \geq 1$, and a tree depth $T \geq 1$.

Output: A random branching structure for a scenario tree having $n \simeq N$ scenarios.

```
1. Create a root node (depth 0). Set $t = 0$.
2. Set $n_t$ to the number of nodes at depth $t$. Set $r_t = \frac{N-1}{T}(1/n_t)$.
3. For each node $j$ of depth $t$:
   Draw $Z_j$ uniformly in the interval $[0,1]$.
   If $Z_j \leq r_t$, append 2 children nodes to node $j$ (binary branching).
   If $Z_j > r_t$, append 1 child node to node $j$ (no branching).
4. If $t < T-1$, increment $t$ and go to Step 2.
   Otherwise, return the branching structure.
```

nodes at depth t. Note that n_t is a random variable, except at the root where $n_0 = 1$. During the construction of the tree, parent nodes at depth $t < T$ are developed and split in 2 children nodes with a probability $r_t = n_t^{-1}(N-1)/T$. Parent nodes have a single child node with a probability $1 - r_t$. If $r_t > 1$, we set $r_t = 1$ and all nodes are split in 2 children nodes. Thus in general $r_t = \min\{1,\ n_t^{-1}(N-1)/T\}$. Note that the truncation of r_t to 1 has no effect on Algorithm 4.4 and has thus been omitted.

Algorithm 4.4 produces branching structures having approximately N scenarios in the following sense. Assume that the number n_{T-1} of existing nodes at depth $T-1$ is large. By the independence of the random decision of creating 1 or 2 successor nodes, and by a concentration of measure argument, the number of nodes created at depth T is approximately equal to

$$\begin{aligned} n_T &= n_{T-1}(2 \cdot r_{t-1} + 1 \cdot (1 - r_{t-1})) = n_{T-1}(1 + r_{t-1}) \\ &= n_{T-1}(1 + (1/n_{T-1})(N-1)/T) = n_{T-1} + (N-1)/T. \end{aligned}$$

Iterating this recursion yields $n_T = n_0 + T(N-1)/T = N$. To establish this latter result, we have neglected the fact that when n_{t-1} is small, the random value of n_t conditionally to n_{t-1} should not be approximated by the conditional mean of n_t, as done in the recursive formula. Thus, we have only $n_T \sim N$. The error affects mostly the first levels of the tree under development, and seems to have a relatively small effect in practice.

4.4.3 Algorithm for Learning Policies

Solving a program on a scenario tree yields a data set of scenario/decision sequence pairs (ξ, u). To infer a decision policy that generalizes the decisions of the tree to test scenarios, we have to learn mappings from $(\xi_0, \ldots, \xi_{t-1})$ to u_t and ensure the compliance of the decisions with the constraints. To some extent the procedure is thus problem-specific. Here again we insist on the methodology.

Dimensionality Reduction.

We first try to represent the information state $(\xi_0, \dots, \xi_{t-1}$ by a smaller number of variables, because the representation $(\xi_0, \dots, \xi_{t-1}$ risks to become very cumbersome as t grows. In particular, we can try to get back to a state-action space representation of the policy (and postprocess data sets accordingly to recover the states). Note that in general, the states we need are those that would be used by a hypothetical reformulation of the optimization problem using dynamic programming. Here the objective is based on the exponential utility function. By the property that

$$\begin{aligned}&\mathbb{E}\{\exp\{-\textstyle\sum_{t'=1}^{T} \xi_{t'-1} \cdot u_{t'}\} \mid \xi_0, \dots, \xi_{t-1}\} \\ &= \exp\{-\textstyle\sum_{t'=1}^{t-1} \xi_{t'-1} \cdot u_{t'}\}\ \mathbb{E}\{\exp\{-\textstyle\sum_{t'=t}^{T} \xi_{t'-1} \cdot u_{t'}\} \mid \xi_0, \dots, \xi_{t-1}\}\ ,\end{aligned}$$

we can see that decisions at $t' = 1, \dots, t-1$ scale by a same factor the contribution to the return brought by the decisions at $t' = t, \dots, T$. Therefore, if the feasibility set at time t can be expressed from state variables, the decisions at $t' = t, \dots, T$ can be optimized independently of the decisions at $t' = 1, \dots, t-1$. This suggests to express u_t as a function of the state ξ_{t-1} of the process ξ, and of an additional state variable ζ_t defined by

$$\zeta_0 = Q\ , \qquad\qquad \zeta_t = Q - \textstyle\sum_{t'=1}^{t-1} u_{t'}\ ,$$

that allows to reformulate, at time t, the constraint $\sum_{t'=1}^{T} u_{t'} \leq Q$ in (4.1) as

$$\textstyle\sum_{t'=t}^{T} u_{t'} \leq \zeta_t\ . \tag{4.6}$$

Feasibility Guarantees Sought Before Repair Procedures.

We try to map the output space in such a way that the predictions learned under the new geometry and then transformed back using the inverse mapping comply with the feasibility constraints. Here, we scale the output u_t so as to have to learn the fraction $y_t = y_t(\xi_{t-1}, \zeta_t) \in [0,1]$ of the maximal allowed output $\min(1, \zeta_t)$. Indeed, note that $0 \leq u_t \leq \min(1, \zeta_t)$ summarizes the constraints of the problem at time t, namely the constraint $0 \leq \pi_t(\xi) \leq 1$ in (4.1) and the constraint (4.6). Since $\zeta_0 = Q$ is fixed (with Q greater than 1 by assumption), we distinguish the cases $u_1 = y_1(\xi_0) \cdot 1$ and $u_t = y_t(\xi_{t-1}, \zeta_t) \cdot \min(1, \zeta_t)$. It will be easy to ensure that fractions y_t predicted by the learned models are valued in $[0,1]$ (thus we actually do not need to define an a posteriori repair procedure).

Input Normalization.

It is convenient for the sequel to normalize the inputs. From the definition of ξ_{t-1} we can recover the state of the random walk b_{t-1}, and use as first input $x_{t1} \overset{\text{def}}{=} (\sigma^2 t)^{-1/2} b_{t-1}$, which follows a standard normal distribution. Thus for the first version of the process ξ, recalling (4.4), instead of ξ_{t-1} we use $x_{t1} = \sigma^{-1}\, t^{-1/2} \log(\xi_{t-1} + K)$. For the second version of the process ξ, recalling (4.5), instead of ξ_{t-1} we use $x_{t1} = \sigma^{-1}\, t^{-1/2} \log(\xi_{t-1} + K) + \sigma t^{1/2}/2$. Instead of the second input ζ_t (for $t > 1$) we use $x_{t2} \overset{\text{def}}{=} \zeta_t / Q$, which is valued in $[0,1]$. We will also rewrite the fraction $y_t = y_t(\xi_{t-1}, \zeta_t)$ as $y_t = g_t(x_{t1}, x_{t2})$ to stress the change of input variables.

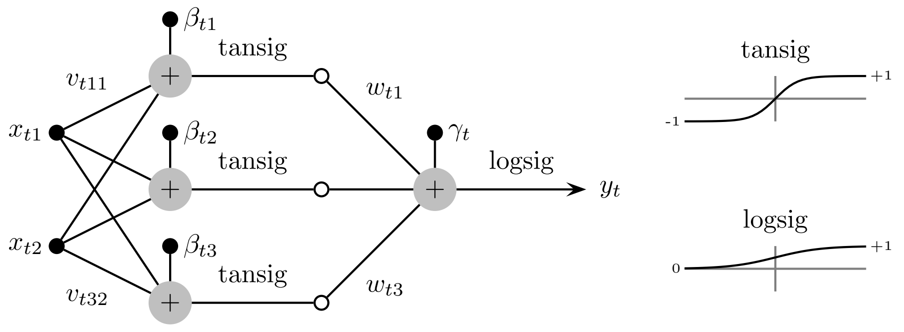

Fig. 4.1: Neural Network model with $L = 3$ hidden layers for the component g_t of the policy (4.7) to be learned from data. The figure is a graphical representation of (4.8). Training the neural networks consists in finding, for each t, values for the parameters v_{tjk}, β_{tj}, w_{tj}, γ_t that best explain examples of pairs (x_t, y_t).

To summarize, the decisions $u_t = \pi_t(\xi)$ will be obtained as follows:

$$\begin{aligned}
x_{t1} &= \begin{cases} \sigma^{-1}\, t^{-1/2} \log(\xi_{t-1} + K) & \text{for the process } \xi \text{ of (4.4)} \\ \sigma^{-1}\, t^{-1/2} \log(\xi_{t-1} + K) + \sigma t^{1/2}/2 & \text{for the process } \xi \text{ of (4.5)} \end{cases} \\
x_{t2} &= \zeta_t / Q = 1 - Q^{-1} \textstyle\sum_{t'=1}^{t-1} u_{t'} \\
y_t &= g_t(x_{t1}, x_{t2}) \\
u_t &= y_t \cdot \min\{1,\ \zeta_t\} = y_t \cdot \min\{1,\ Q - \textstyle\sum_{t'=1}^{t-1} u_{t'}\}\ ,
\end{aligned} \tag{4.7}$$

with π non-anticipative and feasible if and only if g_t is always valued in $[0, 1]$.

Hypothesis Space.

We have to choose the hypothesis space for the functions g_t in (4.7). In the present situation, we find it convenient to choose the class of feed-forward neural networks with one hidden layer of L neurons (Figure 4.1):

$$g_t(x_{t1}, x_{t2}) = \text{logsig}\left(\gamma_t + \textstyle\sum_{j=1}^{L} w_{tj} \cdot \text{tansig}\left(\beta_{tj} + \textstyle\sum_{k=1}^{2} v_{tjk}\, x_{tk}\right)\right), \tag{4.8}$$

with weights v_{tjk} and w_{tj}, biases β_{tj} and γ_t, and activation functions

$$\begin{aligned}
\text{tansig}(x) &= 2 \cdot (1 + e^{-2\,x})^{-1} - 1 && \text{valued in } [-1, +1]\ , \\
\text{logsig}(x) &= (1 + e^{-x})^{-1} && \text{valued in } [0, 1]\ ,
\end{aligned}$$

a usual choice for imposing the output ranges $[-1, +1]$ and $[0, 1]$ respectively.

Since the training sets are extremely small, we take $L = 2$ for g_1 (which has only one input x_{11}) and $L = 3$ for g_t ($t > 1$).

We recall that artificial neural networks have been found to be well-adapted to nonlinear regression. Standard implementations of neural networks (data structure construction and training algorithms) are widely available (Demuth and Beale, 1993). We report here the parameters chosen in our experiments for the sake of completeness; the method is largely off-the-shelf.

Details on the Implementation.

The weights and biases are determined by training the neural networks. We used the Neural Network toolbox of Matlab with the default methods for training the networks by backpropagation — the Nguyen-Widrow method for initializing the weights and biases of the networks randomly, the mean square error loss function, and the Levenberg-Marquardt optimization algorithm. We used $[-3, 3]$ for the estimated range of x_{t1}, corresponding to 3 standard deviations, and $[0, 1]$ for the estimated range of x_{t2}.

Trained neural networks are dependent on the initial weights and biases before training, because the loss minimization problem is nonconvex. Therefore, we repeat the training 5 times from different random initializations. We obtain several candidate policies (to be ranked on the test sample). In our experiments on the problem with $T = 4$, we randomize the initial weights and biases of each network independently. In our experiments on problems with $T > 4$, we randomize the initial weights and biases of $g_1(x_{11})$ and $g_2(x_{21}, x_{22})$, but then we use the optimized weights and biases of g_{t-1} as the initial weights and biases for the training of g_t. Such a warm-start strategy accelerates the learning tasks. Our intuition was that for optimal control problems, the decision rules π_t would change rather slowly with t, at least for stages far from the terminal horizon.

We do not claim that using neural networks is the only or the best way of building models g_t that generalize well and are fast in exploitation mode. The choice of the Matlab implementation for the neural networks could also be criticized. It just turns out that these choices are satisfactory in terms of implementation efforts, reliability of the codes, solution quality, and overall running time.

4.4.4 Solving Programs Approximately by Linear Programming

An option of the proposed testing framework that we have not discussed, as it is linked to technical aspects of numerical optimization, is that we can form the data sets of scenario/decisions pairs using inexact solutions to the optimization programs associated to the trees. Indeed, simulating a policy based on any data set will still give a pessimistic bound on the optimal solution of the targeted problem. The tree selection procedure will implicitly take this new source of approximation into account. In fact, every approximation one can think of for solving the programs could be tested on the problem at hand and thus ultimately accepted or rejected, on the basis of the performance of the policy on the test sample, and the time taken by the solver to generate the decisions of the data set. In the present setting, we judged that solving multiple instances of large-scale nonlinear programs would be too slow with cvx, and preferred to use a large-scale linear programming approximation of the initial objective.

Principle of the Approximation.

Here, we present an approximation used for the problems with $\rho > 0$ on horizons larger than $T = 4$, that turned out to perform satisfactorily on that family of problems. We approximated the function $\exp\{z\}$ in the objective by a convex piecewise linear approximation, $\exp_{\mathrm{L}}\{z\} \overset{\mathrm{def}}{=} \max_{j\in\{0,1,\dots,J-1\}}\{c_j \cdot z + d_j\}$, with $c_{J-1} = d_{J-1} = 0$, and with $c_j, d_j \in \mathbb{R}$ chosen such that $\exp_{\mathrm{L}}\{z_i\} = \exp\{z_i\}$ on a sequence of anchor points

$z_0 > z_1 > \cdots > z_{J-1}$:

$$c_j = \frac{\exp\{z_{j+1}\} - \exp\{z_j\}}{z_{j+1} - z_j} \ , \qquad d_j = \frac{z_{j+1}\exp\{z_j\} - z_j \exp\{z_{j+1}\}}{z_{j+1} - z_j} \ .$$

This allows to approximate the nonlinear formulation of the targeted problem by a linear formulation, at a very light cost in terms of additional optimization variables (representing a new function $v = v(\xi)$ valued in $\mathbb{R}$) and at a controllable cost in terms of additional constraints (J new constraints per scenario ξ). Precisely, everything happens as if the targeted multistage program were formulated as

$$\begin{array}{ll}
\text{minimize} & \mathbb{E}\{v(\xi)\} \\
\text{subject to} & v(\xi) \geq c_j \cdot [-\rho \sum_{t=1}^{T} \xi_{t-1} \cdot \pi_t(\xi)] + d_j \\
& \qquad \text{for } j = 0, \dots, J-2 \ , \\
& v(\xi) \geq 0 \text{ (case } j = J-1), \\
& 0 \leq \pi_t(\xi) \leq 1 \ \text{ and } \sum_{t=1}^{T} \pi_t(\xi) \leq Q \ , \\
& \pi \ \text{non-anticipative.}
\end{array}$$

Details on the Implementation.

The anchor points z_j may be chosen as follows. It is easy to see that at optimality we should always have $\pi_t(\xi) = 0$ if $\xi_{t-1} < 0$. This means that the arguments $z = -\rho \sum_{t=1}^{T} \xi_{t-1} \cdot \pi_t(\xi)$ of the exponential function will always be nonpositive at optimality. Thus we may set $z_0 = 0$: the exponential function will be approximated by the linear function $c_0(z)+d_0$ for $z > 0$ during the optimization process, without loss of precision. On the other hand, in a finite-dimensional approximation, the support of the approximation to the distribution of ξ_t has a maximal value, say ξ_{M}. The minimal value of the argument of the exponential is thus greater or equal to $\bar{z} = -\rho \cdot \xi_{\mathrm{M}} \cdot Q$. Thus if $z_{J-1} \leq \bar{z}$ the exponential function will approximated by $\max\{0, c_{J-2} \cdot z + d_{J-2}\}$ for $z < z_{J-1}$ during the optimization process, without loss of precision. We can then select J and $z_{J-1} < z_{J-2} < \cdots < z_0 = 0$, with $z_{J-1} \leq \bar{z}$, such that the approximation of $\exp\{z\}$ by $\exp_{\mathrm{L}}\{z\}$ is tight enough on the domain $[\bar{z}, 0]$. For all $z \in [\bar{z}, 0]$, we have $\exp(z)_{\mathrm{L}} \geq \exp(z)$ and $\max(\exp_{\mathrm{L}}\{z\} - \exp\{z\}) < \max_j\{|\exp\{z_{j+1}\} - \exp\{z_j\}|\}$.

For solving the linear programs we still use the interior-point solver associated to cvx. One could also switch to simplex methods — arguments in favor of simplex methods may be found in Bixby (2002).

4.4.5 Numerical Results

We now describe the numerical experiments we have carried out and comment on the results.

Experiment on the short-horizon problem instance.

First, we consider the process ξ and parameters (ρ, Q, T) taken from Hilli and Pennanen (2008). We generate a sample of $n' = 10^4$ scenarios drawn independently, on which each learned policy will be tested. We generate 200 random tree structures as described previously (using $r = 0.5$ and rejecting structures with less than 2 or more than 10 scenarios).

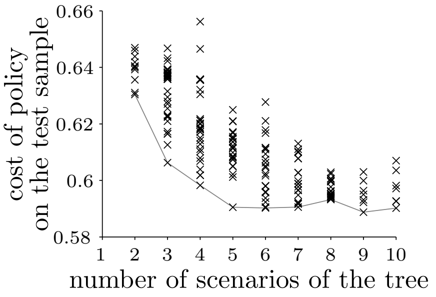

Fig. 4.2: First experiment: scores on the test sample associated to the random scenario trees (lower is better). The linear segments join the best scores of policies inferred from trees of equivalent complexity.

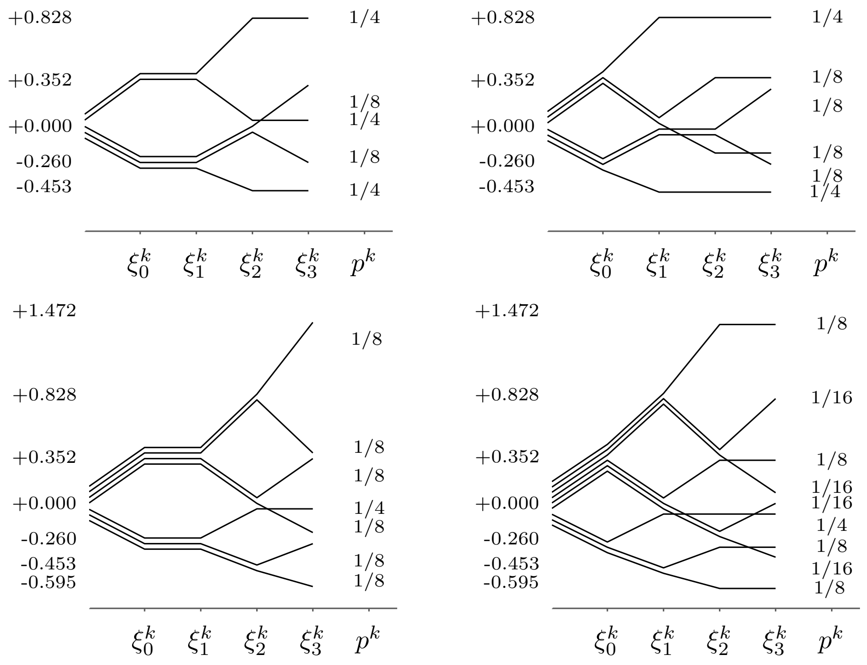

Fig. 4.3: Small trees (5,6,7,9 scenarios) from which good data sets could be obtained. The scenarios $\xi^k = (\xi_0^k, \xi_1^k, \xi_2^k)$ are shifted vertically to distinguish them when they pass through common values, written on the left. Scenario probabilities p^k are indicated on the right.

Node values are set by the deterministic method, thus the variance in performance that we will observe among trees of similar complexity will come mainly from the branching structure. We form and solve the programs on the trees using cvx, and extract the data sets. We generate 5 policies per tree, by repeatedly training the neural networks from random initial weights and biases. Each policy is simulated on the test sample and the best of the 5 policies is retained for each tree.

The result of the experiment is shown on Figure 4.2. Each point is relative to a particular scenario tree. Points from left to right are relative to trees of increasing size.

We report the value of $(1/n')\sum_{j=1}^{n'}\exp\{-\sum_{t=1}^{T}\xi_{t-1}^{j}\cdot\hat{\pi}_t(\xi^j)\}$ for each learned policy $\hat{\pi}$, in accordance with the objective minimized in Hilli and Pennanen (2008). Lower is better. Notice the large variance of the test sample scores among trees with the same number of scenarios but different branching structures.

The tree selection method requires a single lucky outlier to output a good valid upper bound on the targeted objective — quite an advantage with respect to approaches based on worst-case reasonings for building a single scenario tree. With a particular tree of 6 scenarios (best result: 0.59) we already reach the guarantee that the optimal value of our targeted problem is less or equal to $\log(0.59) \simeq -0.5276$. On Figure 4.3, we have represented graphically some of the lucky small scenario trees associated to the best performances. Of course, tree structures that perform well here may not be optimal for other problem instances.

The full experiment, that allows to draw Figures 4.2 and 4.3, takes 10 minutes to run on a pc with a single 1.55 GHz processor and 512 Mb RAM. By comparing our bounds to the results reported in Hilli and Pennanen (2008) (who have undertaken validation experiments taking up to 30 hours on a pc with a single 3.8 GHz processor, 8 Gb RAM, using a test sample of 10000 scenarios, and whose Figure 1 seems to indicate that the best possible value for the bounds should be slightly greater than 0.58), we deduce that we reached essentially the quality of the optimal solution.

Experiment on long-horizon problem instances.

Second, we consider the process ξ taken from Küchler and Vigerske (2010) (see Equation (4.5)) and a series of 15 sets of parameters for ρ, Q, T (see the first columns of Table 4.1). We repeat the following experiment on each (ρ, Q, T) with 3 different values for the parameter N that controls the size of the random trees obtained with Algorithm 4.4: Generate 25 random trees (we recall that this time the node values are also randomized), solve the resulting 25 programs, learn 5 policies per tree (depending on the random initialization of the neural networks), and report as the best score (best upper bound) the lowest of the resulting 125 values computed on a common test sample of $n' = 10000$ scenarios. The test sample is proper to the problem instance (in fact, proper to the time horizon T).

Table 4.1 reports values corresponding to the average performance

$$\rho^{-1}\log\{(1/n')\sum_{j=1}^{n'}\exp\{-\rho\sum_{t=1}^{T}\xi_{t-1}^{j}\cdot\hat{\pi}_t(\xi^j)\}\}$$

obtained for the considered series of problem instances, for the 3 considered nominal tree sizes N (so as to illustrate the effect of the size of the trees on the performance of the learned policies). One column is dedicated to the performance of the analytical reference policy π^{ref} on the test sample.

Note that the case that Küchler and Vigerske (2010) have considered is the case corresponding to $(\rho, Q, T) = (0, 20, 52)$ in our table. The plots from their Figure 3 seem to confirm that the optimal value for this case is around -3.6.

For the cases with $\rho = 0$, the reference value provided by the analytical optimal policy suggests that the best policies found by our approach are close to optimality. For the

Tab. 4.1: Second experiment: Best upper bounds for a family of problem instances.

Problem			Upper bounds[1] on the value of problems (4.1) with the process (4.5)			
ρ	Q	T	Reference[2]	Value of the best policy[3], for 3 tree sizes N		
				$N = 1 \cdot T$	$N = 5 \cdot T$	$N = 25 \cdot T$
0	2	12	**-0.19**	-0.18	-0.17	-0.18
	2	52	**-0.40**	-0.34	-0.32	-0.39
	6	12	**-0.51**	-0.50	-0.49	-0.49
	6	52	**-1.19**	-1.07	-1.03	-1.18
	20	52	**-3.64**	-3.59	-3.50	-3.50
0.25	2	12	**-0.18**	-0.17	-0.17	-0.17
	2	52	**-0.34**	-0.32	-0.31	-0.33
	6	12	**-0.44**	**-0.44**	**-0.44**	**-0.44**
	6	52	-0.75	-0.78	-0.78	**-0.80**
	20	52	-1.46	-1.89	**-1.93**	-1.91
1	2	12	**-0.15**	**-0.15**	**-0.15**	**-0.15**
	2	52	-0.22	**-0.25**	-0.22	-0.24
	6	12	-0.31	**-0.34**	**-0.34**	**-0.34**
	6	52	-0.37	-0.53	-0.53	**-0.54**
	20	52	-0.57	-0.96	**-0.98**	-0.96

[1] Estimated on a test sample of $n' = 10000$ scenarios.
In a same row, lower is better. The best upper bound is in bold.
[2] Defined by $\pi_t^{\text{ref}}(\xi)$ (Equation (4.3)) and optimal for the risk-neutral case $\rho = 0$.
[3] Out of 125 policies learned from 25 random scenario trees (considered separately) of about N scenarios, built with Algorithm 4.4.

cases with $\rho = 0.25$, the reference policy is now suboptimal. It still slightly dominates the learned policies when $Q = 2$, but not anymore when $Q = 6$ or $Q = 20$. For the cases with $\rho = 1$, the reference policy is dominated by the learned policies, except perhaps for the cases with $Q = 2$. We also observe that results obtained with smaller trees (cases $N = 1 \cdot T$) are sometimes better than results obtained with larger trees (cases $N = 25 \cdot T$, that is, $N = 300$ if $T = 12$ and $N = 1300$ if $T = 52$). There is indeed a random component in our tree generation approach, and it may happen that one small tree ultimately gives a better data set than the data sets of the large trees, especially given the relatively small number of trials in this experiment (25 trees per size N) compared to the number of stages.

Overall, the approach seems promising in terms of the usage of computational resources. Table 4.2 reports the times taken for computing the bounds reported in Table 4.1, using a Matlab/cvx implementation on a pc with a single 1.55 GHz processor, 512 Mb RAM. We recall that obtaining one bound involves generating 25 trees, forming and solving the 25 corresponding mathematical programs, learning 125 policies, and testing the 125 policies on 10000 scenarios. For instance, obtaining one of the 15 bounds of the column $N = 1 \cdot T$ of Table 4.1 takes between 2 minutes (for the case $\rho = 0$, $Q = 2$, $T = 12$) and 9 minutes (for the case $\rho = 1$, $Q = 20$, $T = 52$). Obtaining one of the 15

Tab. 4.2: Cpu Times for computing the bounds in Table 4.1

Problem			Total[1] cpu time (in seconds)		
ρ	Q	T	$N = 1 \cdot T$	$N = 5 \cdot T$	$N = 25 \cdot T$
0	2	12	122	156	220
	2	52	415	551	1282
	6	12	123	150	223
	6	52	435	590	1690
	20	52	465	666	1783
0.25	2	12	136	169	250
	2	52	460	780	2955
	6	12	133	161	263
	6	52	504	1002	4702
	20	52	524	1084	5144
1	2	12	136	168	268
	2	52	485	986	4425
	6	12	139	187	313
	6	52	524	1095	5312
	20	52	543	1234	6613

[1] Time for generating 25 trees of about N scenarios, forming and solving the corresponding 25 programs, learning 125 policies, testing each policy on 10^4 scenarios.

bounds of the column $N = 25 \cdot T$ of Table 4.1 takes from less than 4 minutes (for the case $\rho = 0$, $Q = 2$, $T = 12$, $N = 300$) to 110 minutes (for the case $\rho = 1$, $Q = 20$, $T = 52$, $N = 1300$).

The experiment shows that even if the proposed scenario tree selection method requires generating and solving several trees, rather than one single tree, it can work very well. In fact, the experiment illustrates that with a random tree generation process that can generate an "interesting" set of small trees, there is a good likelihood (on the studied family of problems) that at least one of those trees will lead to excellent performances.

4.5 Time Inconsistency and Bounded Rationality Limitations

This section briefly discusses the notion of dynamically consistent decision process, which is relevant to sequential decision making with risk-sensitivity — by opposition to the optimization of the expectation of a total return over the planning horizon, which can be described as risk-indifferent, or risk-neutral.

4.5.1 Time-Consistent Decision Processes

We will say that an objective induces a dynamically consistent policy, or time-consistent policy, if the decisions selected by a policy optimal for that objective coincide with the decisions selected by a policy recomputed at any subsequent time step t and optimal for the same objective with decisions and observations prior to t set to their realized value (and decisions prior to t chosen according to the initial optimal policy).

Time-consistent policies are not necessarily time-invariant: we simply require that the optimal mappings π_t from information states i_t to decisions u_t at time t, evaluated from some initial information state at $t = 0$, do not change if we take some decisions following these mappings, and then decide to recompute them from the current information state. We recall that in the Markov Decision Process framework, the information state i_t is the current state x_t, and in the multistage stochastic programming framework, i_t is the current history $(\xi_1, \dots, \xi_{t-1})$ of the random process, with t indexing decision stages. We say that a decision process is time-consistent if it is generated by a time-consistent policy.

A close notion of time-consistency can also be defined by saying that the preferences of the decision maker among possible distributions for the total return over the planning horizon can never be affected by future information states that the agent recognizes, at some point in the decision process, as impossible to reach (Shapiro, 2009; Defourny et al., 2008).

In the absence of time-consistency, the following situation may arise (the discussion is made in the multistage stochastic programming framework). At time $t = 1$, an agent determines that for each possible outcome of a random variable ξ_2 at time $t = 2$, the decision $u_2 = a$ at time $t = 2$ is optimal (with respect to the stated objective and constraints of the problem, given the distribution of $\xi_2, \xi_3, \dots$, and taking account of optimized recourse decisions $u_3, u_4, \dots$ over the planning horizon). Then at time $t = 2$, having observed the outcome of the random variable ξ_1 and conditioned the probability distributions of $\xi_2, \xi_3, \dots$ over this observation, and in particular, having ruled out all scenarios where ξ_1 differs from the observed outcome, the agent finds that for some possible realizations of ξ_2, $u_2 = a$ is not optimal.

The notion of time-consistency already appears in Samuelson (1937), who states: "as the individual moves along in time there is a sort of perspective phenomenon in that his view of the future in relation to his instantaneous time position remains invariant, rather than his evaluation of any particular year" (page 160). Several economists have rediscovered and refined the notion (Strotz, 1955; Kydland and Prescott, 1977), especially when trying to apply expected utility theory (von Neumann and Morgenstern, 1947), valid for comparisons of return distributions viewed from a single initial information state, to sequential decision making settings, where the information state evolves.

In fact, if an objective function subject to constraints can be optimized by dynamic programming, in the sense that a recursive formulation of the optimization is possible using value functions (on an augmented state space if necessary, and irrespectively of complexity issues), then an optimal policy will satisfy the time-consistency property. This connection between Bellman's principle (1957) and time-consistency is well-established (Epstein and Schneider, 2003; Riedel, 2004; Ruszczyński and Shapiro, 2006; Boda and Filar, 2006; Artzner et al., 2007). By definition and by recursion, a value function is not affected by states that have a zero probability to be reached in the future; when the value function is exploited, a decision u_t depends only on the current information state i_t. Objectives that can be optimized recursively include the expected sum of rewards, and the expected exponential utility of a sum of rewards (Howard and Matheson, 1972), with discount permitted, although the recursion gets more involved (Chung and Sobel, 1987). A typical example of objective that cannot be rewritten recursively in general is the variance of the total return over several decision steps. This holds true even if the state fully describes the distribution of total returns conditionally to the current state. Note, however, that a nice way of handling a mean-variance objective on the total return is to relate it to the expected exponential utility: if R denotes a random total return, $\Phi_\rho\{R\} = \mathbb{E}\{R\} - (\rho/2)\text{var}\{R\} \simeq -\rho^{-1} \log \mathbb{E}\{\exp(-\rho R)\}$. The approximation holds for small $\rho > 0$. It is exact for all $\rho > 0$ if R follows a Gaussian distribution.

4.5.2 Limitations of Validations Based on Learned Policies

In our presentation of multistage stochastic programming, we did not discuss several extensions that can be used to incorporate risk awareness in the decision making process. In particular, a whole branch of stochastic programming is concerned with the incorporation of *chance constraints* in models (Prékopa, 1970; Prékopa, 1995), that is, constraints to be satisfied with a probability less than 1. Another line of research involves the incorporation of modern risk measures such as the *conditional value-at-risk* at level α (expectation of the returns relative to the worst α-quantile of the distribution of returns) (Rockafellar and Uryasev, 2000). An issue raised by many of these extensions, when applied to sequential decision making, is that they may induce time-inconsistent decision making processes (Boda and Filar, 2006).

The validation techniques based on supervised learning that we have proposed are not adapted to time-inconsistent processes. Indeed, these techniques rely on the assumption that the optimal solution of a multistage stochastic program is a sequence of optimal mappings π_t from reachable information states $(\xi_1, \dots, \xi_{t-1})$ to feasible decisions u_t, uniquely determined by some initial information state at which the optimization of the mappings takes place. We believe, however, that the inability to address the full range of possible multistage programming models should have minor practical consequences. On the one hand, we hardly see the point of formulating a sophisticated multistage model with optimal recourse decisions unrelated to those that would be implemented if the corresponding information states are actually reached. On the other hand, it is always possible to simulate any learned policy, whatever the multistage model generating the learning data might be, and score an empirical return distribution obtained with the simulated policy according to any risk measure important for the application. Computing a policy and sticking to it, even if preferences are changing over time, is a form of precommitment (Hammond, 1976).

Finally, let us observe that a shrinking-horizon policy can be time-inconsistent for two reasons: (i) the policy is based on an objective that cannot induce a time-consistent decision process; (ii) the policy is based on an objective that could be reformulated using value functions, but anyway the implicit evaluation of these value functions changes over time, due to numerical approximations local to the current information state. Similarly, if an agent uses a supervised-learning based policy to take decisions at some stage and is then allowed to reemploy the learning procedure at later stages, the overall decision sequence may appear as dynamically inconsistent. The source (ii) of inconsistency appears rather unavoidable in a context of bounded computational resources; more generally, it seems that bounded rationality (Simon, 1956) would necessarily entail dynamical inconsistency.

4.6 Conclusions

This chapter has presented a generic procedure for estimating the value of approximate solutions to multistage stochastic programs. A direct application of this procedure is the evaluation of the quality of the discretization of the original program. The proposed selection of a best scenario tree among an ensemble of trees generated randomly, with the branching structure also randomized, contributes to bring partial answers to the general problem of building good scenario trees efficiently.

Our simple description of the proposed tree selection scheme (Algorithm 4.3), based on an ensemble of random scenario trees generated independently, is less naive than it might appear at first view, with in mind more advanced Monte Carlo sampling techniques for generating the trees sequentially. Indeed, there is a terrible dimensionality challenge in the search for a proper approximate representation of a random process $\xi = (\xi_1, \ldots, \xi_T)$ by a scenario tree, already on short horizons, say T equal to 4 or 5, and especially if the dimension of the random vectors ξ_t is larger than say 1 or 2. In that context, it is not clear whether more advanced importance sampling schemes would be tractable for problems of practical interest.

On the other hand, given a scenario tree and optimal decisions associated to its nodes, there is still much liberty in the way a policy can be learned, and in the way the feasibility of the output of a learned decision predictor can be efficiently restored. The next chapter will explore some of these possibilities.

We leave as future work the investigations concerning policies learned from the data obtained from several scenario trees. Based on the numerical results collected in this chapter, our first intuition is that the trees would have first to be sorted out. Indeed, many trees, as we currently generate them, give very poor decisions. Adding the decisions of such trees to a common data set is likely to hurt policies learned from the common data set. The issue, however, is that we can sort out trees only if we can score them. Currently, we score the trees by testing a policy learned from them. Our conclusion is that learning a policy from several scenario trees would imply a computationally intensive, boosting-like approach: the best policies learned from say the largest trees one could solve would serve to identify the scenario/decisions pairs to be collected in a large data set, that would then be used by a next generation of policies. Such ideas are difficult to test and refine on the problems we have considered in this chapter, because the best policies learned from single trees already yield near-optimal results.

Chapter 5

Inferring Decisions from Predictive Densities

In this chapter, we investigate alternative methods for learning feasible policies given a data set of scenario/decisions pairs. We seek to infer conditional probability models (predictive densities) for the decisions u_t given the information state $(\xi_1, \ldots, \xi_{t-1})$, and then to obtain feasible decisions on new scenarios ξ by maximizing online the probability of the decision u_t subject to the current feasibility constraints $u_t \in \mathcal{U}_t(\xi)$.

The chapter is organized in a backward fashion: Section 5.1 assumes that a predictive density is available and seeks to exploit it so as to select a feasible decision; Section 5.2 concentrates on the inference of conditional predictive densities, given the current information state. In Section 5.3, a certain number of the proposed ideas are illustrated, evaluated, and sometimes modified, in the context of a particular problem.

Notations.

In this chapter, we use the following notations.

- $A^T \in \mathbb{R}^{n\times m}$ is the transpose of $A \in \mathbb{R}^{m\times n}$.
- $\langle a, b\rangle = a^T b$ is the inner product between 2 vectors a, b of the same dimension.
- $||a|| = \langle a, a\rangle^{1/2}$ is the Euclidian norm of the vector a.
- For $x = [x_1 \, \ldots \, x_n]^T$ and $y = [y_1 \, \ldots \, y_n]^T \in \mathbb{R}^n$, $x \preceq y$ means $x_i \leq y_i$, $1 \leq i \leq n$, and $x \prec y$ means $x_i < y_i$, $1 \leq i \leq n$.
- Given column vectors $z_1, \ldots, z_n$, we freely write $z = (z_1, \ldots, z_n)$ to define a column vector $z = [z_1^T \ldots z_n^T]^T$, especially when the vectors z_i are replaced by vectors with superscripts.

5.1 Constrained MAP Repair Procedure

We consider the following setup: Given a predictive density $\hat{p}_t$ for the decision $u_t \in \mathbb{R}^n$, infer (select) a decision $\bar{u}_t$ such that $\bar{u}_t$ satisfies the feasibility constraints $\bar{u}_t \in \mathcal{U}_t(\xi)$. The given density $\hat{p}_t$ is in fact an estimated density, obtained for instance as described in Section 5.2. For the selection of a decision from the density, we maximize (the logarithm of) the predictive density subject to constraints, which leads to the following estimate:

$$\bar{u}_t \in \operatorname{argmax}_{u_t \in \mathcal{U}_t(\xi)} \log \hat{p}_t(u_t) \ . \tag{5.1}$$

If $\hat{p}_t$ is unimodal with its mode in $\mathcal{U}_t(\xi)$, then $\bar{u}_t$ is given by the mode of $\hat{p}_t$. The nontrivial case is when the mode is not in $\mathcal{U}_t(\xi)$.

To make the approach computationally viable, we have to introduce restrictions on the density $\hat{p}_t$ and the feasible sets $\mathcal{U}_t$. Moreover, one may want to ensure that the solution set in (5.1) is a singleton. Indeed, we are interested in the situation where $\bar{u}_t$ is viewed as the decision of a deterministic policy $\pi_t(\xi_1, \dots, \xi_{t-1})$; by selecting $\bar{u}_t$ arbitrarily from the solution set, an undesirable source of randomness would be added to the decision process.

5.1.1 Assumptions

An interesting restriction on the models for $\hat{p}_t$ is to assume that $\hat{p}_t$ is taken from an exponential family of distributions.

The following description of exponential families will suffice for our purposes. Given an index set $\mathcal{I}$ of finite cardinality $|\mathcal{I}| = d$, and a finite collection $\{\phi_\ell\}_{\ell\in\mathcal{I}}$ of functions $\phi_\ell : \mathbb{R}^n \to \mathbb{R}$, let $\phi(u_t) \in \mathbb{R}^d$ denote the d-dimensional column vector with elements $\phi_\ell(u_t)$, $\ell \in \mathcal{I}$, and define the (natural) exponential family associated to the collection $\{\phi_\ell\}_{\ell\in\mathcal{I}}$ as

$$p(u_t;\theta) = \exp\{\langle\theta, \phi(u_t)\rangle - A(\theta)\} \ , \tag{5.2}$$

where θ is allowed to take values from a set $\Theta \subset \mathbb{R}^d$ described below, and where $A(\theta)$ is the so-called cumulant generating function (log-partition function) defined by

$$A(\theta) = \log \int_{\mathbb{R}^n} \exp\{\langle\theta, \phi(u_t)\rangle\} \mathrm{d}u_t \ . \tag{5.3}$$

Choosing a value for θ amounts to select a distribution among the members of the exponential family.

The domain of the parameter θ is the set $\Theta = \{\theta \in \mathbb{R}^d : A(\theta) < \infty\}$. In the terminology of Appendix A, the set Θ is the effective domain of the cumulant generating function $A(\theta)$ viewed as an extended-real-valued function. The (natural) exponential family is said to be regular if Θ is open. In the sequel, we assume that the family is regular. It is well-known (Brown, 1986; Robert, 2007; Wainwright and Jordan, 2008) that $A(\theta)$ is a convex function of θ (and thus in particular that Θ is convex). Moreover, $A(\theta)$ is strictly convex for the so-called minimal exponential families. Minimal exponential families are (natural) exponential families such that the functions ϕ_ℓ, $\ell \in \mathcal{I}$, and the constant-valued function $\phi_0(x) = 1$, form a set of linearly independent functions — that is, for any $\theta \neq 0$, $\langle\theta, \phi(u_t)\rangle$ is not a constant-valued function of u_t.

For minimal exponential families, there is a one-to-one correspondence between a value $\theta \in \Theta$ and a distribution from the family.

Using $p(u_t;\theta)$ from (5.2) for $\hat{p}_t(u_t)$, the problem (5.1) becomes

$$\bar{u}_t \in \operatorname{argmax}_{u_t\in\mathcal{U}_t(\xi)} \langle\theta, \phi(u_t)\rangle \ , \tag{5.4}$$

which is independent of the constant term $A(\theta)$, and corresponds formally to a maximum a posteriori (MAP) estimation problem subject to additional constraints.

To ensure that (5.4) has a solution, we assume that the set $\mathcal{U}_t(\xi)$ is nonempty, closed, and convex. Moreover, we assume that the support of $p(u_t;\theta)$ meets the interior of $\mathcal{U}_t(\xi)$, in order to guarantee that (5.4) has a nonempty solution set and does not lead to a pathological optimization problem. It is well-known that the support of exponential families does not depend on the value of their parameter θ. Therefore, given a subset C of $\mathbb{R}^n$ such that $\mathcal{U}_t(\xi)$ is always in C for all ξ (it is possible to choose $C = \mathbb{R}^n$), one can choose the exponential family so that its support covers C.

5.1.2 Particularizations

In multistage stochastic programming models, a frequent form for a set $\mathcal{U}_t(\xi)$ is

$$\mathcal{U}_t(\xi) = \{u_t \in \mathbb{R}^n : A_t u_{t-1} + B_t u_t = h_t,\ \ u_t \succeq 0\} \ , \tag{5.5}$$

where u_{t-1} is the decision relative to the previous stage (with u_{t-1} actually depending only on $\xi_1, \dots, \xi_{t-2}$), B_t is very often a fixed matrix (recourse matrix), and A_t, h_t are a matrix (technology matrix) and a vector that may both depend on $\xi_1, \dots, \xi_{t-1}$ (often only affinely). The form (5.5) is in part justified by results for two-stage stochastic programming problems (Appendix D). When one uses (5.4) for computing $\bar{u}_t$ online on a new scenario, the realizations of u_{t-1}, A_t and h_t are known, and (5.4) becomes the problem of solving over $u_t \in \mathbb{R}^n$ the program

$$\text{maximize} \quad \langle \theta_t, \phi(u_t) \rangle \quad \text{subject to} \quad B_t u_t = h_t - A_t u_{t-1},\ \ u_t \succeq 0 \ . \tag{5.6}$$

In the sequel, we seek to identify some exponential families that lead to a concave objective in (5.6).

Multivariate normal distributions.

We consider for $\hat{p}_t$ in (5.1) the multivariate normal distribution $\mathcal{N}(\lambda, \Lambda)$ with mean $\lambda \in \mathbb{R}^n$ and covariance matrix $\Lambda \in \mathbb{R}^{n \times n}$ (we do not stress in the notation λ, Λ a possible dependence of these parameters on $\xi_1, \dots, \xi_{t-1}$ and on t). We assume that Λ is positive definite, so that the normal distribution has a density, namely,

$$\begin{aligned} \hat{p}_t(u_t) &= ((2\pi)^n \det \Lambda)^{-1/2} \exp\{-\tfrac{1}{2}(u_t - \lambda)^T \Lambda^{-1} (u_t - \lambda)\} \\ &= \exp\{-\tfrac{1}{2}(\mathrm{tr}\{\Lambda^{-1} u_t u_t^T\} - 2\langle \Lambda^{-1}\lambda, u_t\rangle + \lambda^T \Lambda^{-1} \lambda - \log\{(2\pi)^n \det \Lambda^{-1}\})\} \ . \end{aligned}$$

In that case, using the precision matrix $S = \Lambda^{-1}$, the program (5.6) becomes the strictly convex quadratic program

$$\begin{aligned} \text{minimize} \quad & (u_t - \lambda)^T S (u_t - \lambda) \\ \text{subject to} \quad & B_t u_t = h_t - A_t u_{t-1},\ \ u_t \succeq 0 \ . \end{aligned} \tag{5.7}$$

The program (5.7) has a simple geometrical interpretation (Figure 5.1) in terms of the Mahalanobis distance $d_M(u_t, v_t) = ||S^{1/2}(u_t - v_t)||_2$ between two vectors $u_t, v_t \in \mathbb{R}^n$ (Mahalanobis, 1936). For conditions ensuring that the feasibility set is nonempty, see Definitions D.3, D.4, D.5 in Appendix D.

A zero-valued element S_{ij} of the precision matrix has the interpretation that the components i, j of u_t are conditionally independent given the other components (Dempster, 1972).

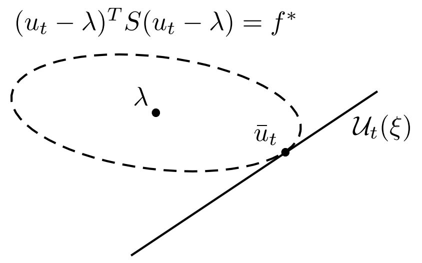

Fig. 5.1: Geometrical interpretation for (5.7). The matrix S defines a Mahalanobis distance in $\mathbb{R}^n$. The program (5.7) consists in computing the projection of λ on the set $\mathcal{U}_t(\xi)$ according to this metric, by minimizing the distance between λ and $u_t \in \mathcal{U}_t(\xi)$. On this figure, $u_t \in \mathbb{R}^2$, and $\lambda \notin \mathcal{U}_t(\xi)$. The level set corresponding to the optimal objective value f^* has been drawn (dashed line).

For stochastic programming problems where the components of the decisions u_t can be put in correspondence with spatial locations, for instance problems defined on networks, it could make sense to use a Gaussian Markov random field model (Speed and Kiiveri, 1986) for the density $\hat{p}_t(u_t)$.

Product of log-concave univariate densities.

We consider exponential families obtained as the product of log-concave univariate densities $\hat{p}_t^i$ (densities such that $\log \hat{p}_t^i(\cdot)$ is a concave function) relative to the i-th component of $u_t = [u_{t\,1} \ \ldots \ u_{t\,n}]^T$. Here, $\hat{p}_t^i$ is taken from an exponential family relative to a collection of functions $\{\phi_\ell^i\}_{\ell \in \mathcal{I}^i}$, through the choice of a parameter vector $\theta^i \in \Theta^i$. We write $\phi^i(u_{t\,i})$ for the vector collecting the elements $\phi_\ell^i(u_{t\,i})$, $\ell \in \mathcal{I}^i$. With these choices, the program (5.6) becomes

$$\text{maximize} \quad \sum_{i=1}^{n} \langle \theta^i, \phi^i(u_{t\,i}) \rangle \quad \text{subject to} \quad B_t u_t = h_t - A_t u_{t-1}, \;\; u_t \succeq 0 \ . \tag{5.8}$$

The form (5.8) is well suited to situations where probabilistic models for the scalar components $u_{t\,i}$ have been learned separately, so as to obtain more tractable learning problems. There is probably some structure among the components $u_{t\,i}$ once u_t is optimized, and we may hope that by enforcing the condition $u_t \in \mathcal{U}_t(\xi)$, we recover, to a certain extent, a part of that structure — while the part of the structure that is induced by the objective function of the original multistage decision making problem is unlikely to be restored by this myopic feasibility restoration procedure.

As an example of log-concave density, we can cite the univariate normal distribution $\mathcal{N}(\mu, \sigma^2)$ with $\sigma^2 > 0$. Another potentially useful example is the gamma distribution $\Gamma(\alpha, \beta)$ with $\alpha \geq 1$ (condition ensuring the log-concavity) and $\beta > 0$, supported on $(0, \infty)$. If we choose for $\hat{p}_t^i$ the gamma distribution $\Gamma(\alpha_i, \beta_i)$, then the density of $\hat{p}_t^i$ is given by

$$\begin{aligned} \hat{p}_t^i(u_{t\,i}) &= (\beta_i)^{\alpha_i} [\Gamma(\alpha_i)]^{-1} (u_{t\,i})^{\alpha_i - 1} \exp\{-\beta_i u_{t\,i}\} \\ &= \exp\{-\beta_i u_{t\,i} + (\alpha_i - 1) \log u_{t\,i} - \log\{[\Gamma(\alpha_i)]/(\beta_i)^{\alpha_i}\}\} \ , \end{aligned}$$

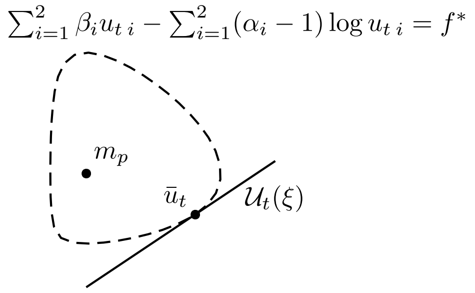
$\sum_{i=1}^{2} \beta_i u_{t\,i} - \sum_{i=1}^{2} (\alpha_i - 1) \log u_{t\,i} = f^*$
m_p
$\bar{u}_t$
$\mathcal{U}_t(\xi)$

Fig. 5.2: Geometrical interpretation for (5.10). The parameters α_i of the marginal distributions $\Gamma(\alpha_i, \beta_i)$ for $1 \le i \le n$ define a weighted Itakura-Saito distance in $\mathbb{R}^n$ (see Remark 5.1). The program (5.10) consists in computing the projection of the mode m_p of the distribution of u_t on the set $\mathcal{U}_t(\xi)$ according to this (pseudo) metric, by minimizing the distance between m_p and $u_t \in \mathcal{U}_t(\xi)$, where $m_p = [m_{p\,1} \ldots m_{p\,n}]^T$, $m_{p\,i} = (\alpha_i - 1)/\beta_i$. On the present figure, $u_t = [u_{t\,1}\; u_{t\,2}]^T \in \mathbb{R}^2$, and $m_p \notin \mathcal{U}_t(\xi)$. The level set corresponding to the optimal objective value f^* has been drawn (dashed line).

where $\Gamma(\alpha_i) = \int_0^\infty t^{\alpha_i - 1} \exp\{-t\} \mathrm{d}t$ is the gamma function evaluated at α_i. One then obtains the objective component

$$\langle \theta^i, \phi^i(u_{t\,i}) \rangle = -\beta_i u_{t\,i} + (\alpha_i - 1) \log u_{t\,i} \ , \tag{5.9}$$

which is strictly concave if $\alpha_i > 1$. Note that its unconstrained maximization would then yield the mode of the distribution $\Gamma(\alpha_i, \beta_i)$, namely $m_{p\,i} \overset{\mathrm{def}}{=} (\alpha_i - 1)/\beta_i$.

Now, if for instance each component $u_{t\,i}$ follows a distribution $\Gamma(\alpha_i, \beta_i)$ with $\alpha_i > 1$ and $\beta_i > 0$, the program (5.8) becomes the strictly convex program

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i=1}^{n} \beta_i u_{t\,i} - \sum_{i=1}^{n} (\alpha_i - 1) \log u_{t\,i} \\ \text{subject to} \quad & B_t u_t = h_t - A_t u_{t-1}, \ \ u_t \succ 0 \end{aligned} \tag{5.10}$$

over the decision vector $u_t = [u_{t\,1}, \ldots, u_{t\,n}]^T$.

A geometrical interpretation of (5.10) is presented on Figure 5.2. In the context of stochastic programming, it could make sense to choose a Gamma density for a decision $u_{t\,i}$ (or some invertible transform of $u_{t\,i}$) that is naturally valued on the positive reals.

Remark 5.1 *(Justification of the geometrical interpretation for* (5.10)*).* The strictly convex function $F(u_t) = -\sum_{i=1}^{n} \langle \theta^i, \phi^i(u_{t\,i}) \rangle$, obtained by summing the components (5.9) and changing the sign, induces a Bregman divergence (Bregman, 1967; Banerjee et al., 2005) between $u_t, v_t \in \mathbb{R}^n$ given by

$$\begin{aligned} B(u_t || v_t) &= F(u_t) - F(v_t) - \nabla F(v_t)^T (u_t - v_t) \\ &= \sum_{i=1}^{n} \left[\beta_i (u_{t\,i} - v_{t\,i}) - (\alpha_i - 1) \log \frac{u_{t\,i}}{v_{t\,i}} \right] - \sum_{i=1}^{n} \left(\beta_i - \frac{\alpha_i - 1}{v_{t\,i}} \right) (u_{t\,i} - v_{t\,i}) \\ &= \sum_{i=1}^{n} (\alpha_i - 1) \left(\frac{u_{t\,i}}{v_{t\,i}} - \log \frac{u_{t\,i}}{v_{t\,i}} + 1 \right) \ . \end{aligned} \tag{5.11}$$

The divergence (5.11) is a weighted version of the Itakura-Saito distance (Itakura and Saito, 1968) , defined by $d_{IS}(u_t \| v_t) = \sum_{i=1}^n [(u_{t\,i}/v_{t\,i}) - \log(u_{t\,i}/v_{t\,i}) + 1]$, which can be obtained as the Bregman divergence induced by $F_{IS}(u_t) = -\sum_{i=1}^n \log(u_{t\,i})$. (Originally, the Itakura-Saito distance is defined between power spectrum functions, so that the sum over components i is in fact an integral over phases between $-\pi$ and π.)

Now, setting v_t to the mode m_p of the joint distribution of u_t defined componentwise by $m_{p\,i} = (\alpha_i - 1)/\beta_i$, the divergence (5.11) becomes

$$B(u_t \| m_p) = \sum_{i=1}^n \beta_i u_{t\,i} - \sum_{i=1}^n (\alpha_i - 1) \log u_{t\,i} + \sum_{i=1}^n (\alpha_i - 1) \left(\log \frac{\alpha_i - 1}{\beta_i} + 1 \right) \ ,$$

that is, the objective of (5.10) up to a constant term. The omission of the constant term shifts the value of the objective but does not alter the geometry of the level sets. □

5.2 Gaussian Predictive Densities

In this section, we consider joint probability models over $(\xi_1, \dots, \xi_T, u_1, \dots, u_T)$, from which conditional densities $\hat{p}_t$ for u_t can be obtained by conditioning over the observation of $\xi_1, \dots, \xi_{t-1}$. An interesting case is when the conditional densities for u_t are Gaussian, since this allows us to use (5.7) and find a decision $u_t \in \mathcal{U}_t(\xi)$.

Note that the Gaussian case is in fact rather general inasmuch as one can also approximate a density by a multivariate normal density (Laplace's approximation): Given a density $\hat{p}_t(x)$ with mode $m_p \in \mathbb{R}^n$, twice differentiable in a neighborhood of m_p, compute the Hessian matrix $H \in \mathbb{R}^{n \times n}$ of $\hat{p}_t$ at m_p (elements $H_{ij} = \partial^2 \hat{p}_t(m_p)/\partial x_i \partial x_j$), and then replace $\hat{p}_t$ by the density of a normal $\mathcal{N}(\lambda, \Lambda)$ with $\lambda = m_p$ and $\Lambda^{-1} = -H$.

5.2.1 Joint Gaussian Model

We consider the following joint Gaussian model as a base case for learning probabilistic models.

Description.

It is well known that if a random vector $z = (x, y)$ follows a multivariate normal distribution $\mathcal{N}(\bar{z}, \Sigma)$ with

$$\bar{z} = \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} \ , \qquad \Sigma = \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{xy}^T & \Sigma_y \end{bmatrix} \ ,$$

with Σ positive definite, then y conditionally to x follows a multivariate normal distribution $\mathcal{N}(\lambda(x), \Lambda)$, where

$$\lambda(x) = \bar{y} + \Sigma_{xy}^T \Sigma_x^{-1} (x - \bar{x}) \ , \qquad \Lambda = \Sigma_y - \Sigma_{xy}^T \Sigma_x^{-1} \Sigma_{xy} \ . \tag{5.12}$$

A simple model of the predictive density for u_t given $\xi_1, \dots, \xi_{t-1}$ can be obtained by setting $x = (\xi_1, \dots, \xi_{t-1})$, $y = u_t$, and then using the conditioning formulae (5.12) on a multivariate normal model $\mathcal{N}(\bar{z}, \Sigma)$ for $z = (x, y)$, with $\bar{z}$ and Σ learned (estimated) from a data set of scenario/decisions pairs (see below).

The evaluation of (5.12) for $u_2, \dots, u_T$ requires $T-1$ matrix inversions, but as Σ_x^{-1} is independent of the observations x, the inversions and matrix products need not be recomputed online on new scenarios.

By (5.12), the conditional mean of u_t is an affine function of the observed history $(\xi_1, \dots, \xi_{t-1})$. In fact, $\lambda(x)$ would be called a linear decision rule in the context of stochastic programming (Garstka and Wets, 1974).

Estimation.

There is a large literature on the estimation of the mean and the covariance matrix (or its inverse) of a Gaussian random vector (Stein, 1956; Haff, 1980; Banerjee et al., 2008). In the present context, given a data set of samples $\{z^k\}_{1 \le k \le N}$, where $z^k = (x^k, y^k)$, $x^k = (\xi_1^k, \dots, \xi_{t-1}^k)$, $y^k = u_t^k$, we can estimate the mean $\bar{z}$ by $\hat{z} = (1/N) \sum_{k=1}^{N} z^k$, and estimate the covariance matrix Σ by a simple shrinkage estimator of the form

$$\hat{\Sigma} = (1-\epsilon)\Sigma_{\mathrm{ML}} + \epsilon\, I \ , \quad \text{with } \Sigma_{\mathrm{ML}} = (1/N) \sum_{i=1}^{N} (z^k - \hat{z})(z^k - \hat{z})^T \ . \tag{5.13}$$

The identity matrix I is added with weight $\epsilon \in (0,1)$ in order to ensure that the estimated covariance is positive definite and well-conditioned.

If $\hat{\Sigma}$ in (5.13) is scaled by some positive factor, the conditional covariance Λ in (5.12) is scaled by the same factor, whereas the conditional mean $\lambda(x)$ is left unchanged. As the minimizer of (5.7) is invariant with respect to a rescaling of the objective, one can thus rescale (5.13) by a factor $(1-\epsilon)^{-1}$, set $\epsilon' = \epsilon/(1-\epsilon) > 0$ and simply use

$$\hat{\Sigma} = \Sigma_{\mathrm{ML}} + \epsilon'\, I \quad . \tag{5.14}$$

By the same token, there is no potential advantage in replacing the maximum likelihood estimator Σ_{ML} by an unbiased empirical estimator

$$\Sigma_{\mathrm{emp}} = (N-1)^{-1} \sum_{i=1}^{N} (z^k - \hat{z})(z^k - \hat{z})^T \ .$$

Discussion and Extension.

The program (5.7) that restores the feasibility of u_t uses larger corrections for components $u_{t\,i}$ of u_t with larger conditional variances Λ_{ii}. Under the joint Gaussian model, the components $u_{t\,i}$ of the decision vector u_t that have a larger estimated variance (relatively to the other components) are those that are not well explained by the linear model (compared to the other components).

Due to the corrections made by (5.7), the actual decision $\bar{u}_t$ will not in general depend affinely on $(\xi_1, \dots, \xi_{t-1})$. Therefore, it might be beneficial to consider the actual decisions $(u_2, \dots, u_{t-1})$ as new observations, and extend the conditional model for u_t by computing (5.12) with $y = u_t$ and $x = (\xi_1, \dots, \xi_{t-1}, u_2, \dots, u_{t-1})$.

5.2.2 Gaussian Process Model

Gaussian processes allow to define nonparametric models (by opposition to models with an a priori fixed number of parameters for summarizing the data, whatever the size of the data). Following O'Hagan (1978), it is often said that Gaussian processes allow to define prior distributions over spaces of functions, that are then updated to posterior distributions, given a data set of observations of the relation between inputs and outputs. Note that since Gaussian process models often incorporate the effect of a noise process on observations, the relation between inputs and noisy observed outputs is actually not of a purely functional nature (Neal, 1997, page 4).

Description.

Let $\mathcal{J}$ denote an index set (typically infinite), and let $\{X^\alpha\}_{\alpha\in\mathcal{J}}$ denote a collection of vectors $X^\alpha \in \mathbb{R}^n$ such that $X^\alpha \neq X^\beta$ if $\alpha \neq \beta$. The vectors X^α are interpreted as query points uniquely identified by labels $\alpha \in \mathcal{J}$ (the labels can indicate an ordering between distinct query points). For each $\alpha \in \mathcal{J}$, let Y^α be a real-valued random variable with finite variance. For any finite subset S of indices from $\mathcal{J}$, let $|S|$ denote the cardinality of S, and let $Y(S)$ denote the $|S|$-dimensional random vector with elements Y^α, $\alpha \in S$. We assume that for any such subset S, the random vector $Y(S)$ follows a multivariate normal distribution $\mathcal{N}(\mu(S), K(S))$, with its mean vector $\mu(S) \in \mathbb{R}^{|S|}$ and covariance matrix $K(S) \in \mathbb{R}^{|S|\times|S|}$ defined below. This defines a so-called Gaussian process $\{Y^\alpha\}_{\alpha\in\mathcal{J}}$.

The mean vector $\mu(S) = \mathbb{E}\{Y(S)\}$ collects (stacks into a column vector) the elements

$$\mu^\alpha = g(X^\alpha) \ , \quad \alpha \in S \ ,$$

defined using some fixed real-valued function $g : \mathbb{R}^n \to \mathbb{R}$ called the mean function.

The covariance matrix $K(S) = \mathbb{E}\{[Y(S) - \mu(S)][Y(S) - \mu(S)]^T\}$ collects (stacks into a symmetric $|S| \times |S|$ matrix) the elements

$$K^{\alpha\beta} = k(X^\alpha, X^\beta) \ , \quad \alpha, \beta \in S \ ,$$

defined using some fixed positive definite kernel $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ (see Definition C.11 in Appendix C — the name "positive definite kernel" is standard whereas the corresponding matrix $K(S)$ is only positive *semi*-definite). The kernel k (also called covariance function) is parametrized by a vector η of hyperparameters that has not been written explicitly to lighten the notation. For example, a kernel $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ with values

$$k(X^\alpha, X^\beta) = v_0 \exp\{-\tfrac{1}{2}\textstyle\sum_{i=1}^n (X_i^\alpha - X_i^\beta)^2/\sigma_i^2\}$$

(radial basis kernel) is parametrized by $\eta = (v_0, \sigma_1^{-2}, \ldots, \sigma_n^{-2})$, with $v_0 > 0$ and where each $\sigma_i > 0$ is a bandwidth parameter associated to the i-th coordinate of the inputs X^α and X^β.

Now, let (S_1, S_2) denote a partition of S, that is, $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \varnothing$. Let $K(S_1, S_2) = \mathbb{E}\{[Y(S_1) - \mu(S_1)][Y(S_2) - \mu(S_2)]^T\}$ be the matrix with elements $K^{\alpha\beta}$ for $\alpha \in S_1$, $\beta \in S_2$, and let

$$\mu = \begin{bmatrix} \mu(S_1) \\ \mu(S_2) \end{bmatrix} \ , \qquad K = \begin{bmatrix} K(S_1) & K(S_1, S_2) \\ K(S_1, S_2)^T & K(S_2) \end{bmatrix} \ .$$

Let $Z(S_1)$ denote a $|S_1|$-dimensional random vector representing a noisy observation of $Y(S_1)$, collecting elements defined by

$$Z^\alpha = Y^\alpha + \sigma W^\alpha \ , \quad \alpha \in S_1 \ ,$$

where $\sigma^2 > 0$ represents the variance of the observation noise, assumed to be i.i.d. Gaussian, and where each W^α is assumed to be drawn independently from the standard normal distribution $\mathcal{N}(0,1)$. Then, the random vector $(Z(S_1), Y(S_2))$ follows a multivariate normal $\mathcal{N}(\mu', K')$ with

$$\mu' = \mu \ , \qquad K' = \begin{bmatrix} K(S_1) + \sigma^2 I & K(S_1,S_2) \\ K(S_1,S_2)^T & K(S_2) \end{bmatrix} \ ,$$

where I stands for the $|S_1| \times |S_1|$ identity matrix. In particular, the random vector $Y(S_2)$ conditionally to Z follows a multivariate normal $\mathcal{N}(\lambda_Y(Z(S_1)), \Lambda_Y)$ with the conditional mean and conditional covariance matrix given respectively by

$$\lambda_Y(Z(S_1)) = \mu(S_2) + K(S_1,S_2)^T (K(S_1) + \sigma^2 I)^{-1} (Z(S_1) - \mu(S_1)) \ , \tag{5.15}$$
$$\Lambda_Y = K(S_2) - K(S_1,S_2)^T (K(S_1) + \sigma^2 I)^{-1} K(S_1,S_2) \ . \tag{5.16}$$

When one actually observes a realization $z(S_1) \in \mathbb{R}^{|S_1|}$ of the random vector $Z(S_1)$, the conditional mean of $Y(S_2)$ given $z(S_1)$ is a real vector $\hat{\lambda} = \lambda_Y(z(S_1)) \in \mathbb{R}^{|S_2|}$ that represents the best prediction for the realization of $Y(S_2)$ in the mean-square error sense, while the covariance matrix of the prediction error $\hat{\lambda} - Y(S_2)$ is given by $\hat{\Lambda} = \Lambda_Y$.

The contribution $\sigma^2 I$ from the noise vector W can be viewed as a jitter term that stabilizes the matrix inversion without perturbing too much the model (Neal, 1997). It also allows to consider in Equations (5.15), (5.16), several independent noisy observations at a same query point X^α, by reinterpreting S_1 as a multiset (collection) of indices of $\mathcal{J}$.

We now apply the described Gaussian Process model to the inference of the distribution of a decision vector u_t conditionally to a new scenario ξ (of which we can only observe $\xi_1, \dots, \xi_{t-1}$), given a data set of scenario/decisions pairs (ξ^k, u^k) extracted from a scenario tree. We describe the calculations for the i-th component of u_t, written $u_{t\,i}$.

We define S_1 as an index set relative to the distinct values of $(\xi_1^k, \dots, \xi_{t-1}^k)$ found in the data set, and we set

$$X^\alpha = (\xi_1^\alpha, \dots, \xi_{t-1}^\alpha) \ , \qquad z^\alpha = u_{t\,i}^\alpha \ , \qquad \alpha \in S_1 \ . \tag{5.17}$$

This allows to compute the term $(K(S_1) + \sigma^2 I)^{-1}(z(S_1) - \mu(S_1))$ in (5.15) as soon as we obtain the data set (the realization $z(S_1)$ of $Z(S_1)$). Then, we view S_2 as a singleton relative to a new scenario ξ^*, and we set

$$X^\beta = (\xi_1^*, \dots, \xi_{t-1}^*) \ , \qquad Y^\beta = u_{t\,i} \ , \qquad \beta \in S_2 \ . \tag{5.18}$$

This allows to compute $K(S_1, S_2)$ as soon as we actually observe $(\xi_1^*, \dots, \xi_{t-1}^*)$, with $K(S_1,S_2)$ interpreted as a vector of weights describing the similarity of the new scenario ξ^* with respect to each example ξ^k stored in the data set.

At this stage, we can infer that the real-valued random variable $u_{t\,i}$ follows a univariate normal distribution $\mathcal{N}(\lambda_i, \Lambda_{ii})$ with $\lambda_i = \lambda_Y(z(S_1))$ given by (5.15), and $\Lambda_{ii} = \Lambda_Y$ given by (5.16). As for the predictive density for the full decision u_t, we assume that each component is independent conditionally to $(\xi_1, \dots, \xi_{t-1})$, so that u_t follows a multivariate normal $\mathcal{N}(\lambda, \Lambda)$ with λ formed from the components λ_i, and Λ defined as a diagonal matrix with diagonal entries Λ_{ii}.

Remark 5.2. It is also conceivable to infer a noisy predictive density (Rasmussen and Williams, 2006, page 18), that is, build a model for $Z(S_2)$ given $Z(S_1)$. In that case, $K(S_2)$ is replaced by $K(S_2) + \sigma^2$ in the expression of K' (assuming that S_2 is a singleton), so that $u_{t\,i}$ follows a normal distribution $\mathcal{N}(\lambda_Y(z(S_1)), \Lambda_Y + \sigma^2)$. Note also that if we use the same kernel k (with the same hyperparameter values) for each component of u_t and model these components as being conditionally independent given $(\xi_1, \dots, \xi_{t-1})$, the joint Gaussian model for the components of u_t has a covariance matrix equal to a multiple of the identity matrix. Under that choice, whatever the chosen noise variance σ^2 and the value of Λ_Y common to all components i, the program (5.7) is always equivalent to the minimization of $||u_t - \lambda||^2$ subject to the feasibility constraints, with the conditional mean λ of u_t unaffected by the jitter term σ^2 added to $K(S_2)$. $\square$

It can be seen from (5.15) that the mean and thus the mode of the predicted Gaussian density for $u_{t\,i}$ combine the decisions $u^{\alpha}_{t\,i}$ of the data set, in a way that depends on the similarity (determined by the kernel k) between the observed part $(\xi^*_1, \dots, \xi^*_{t-1})$ of the new scenario ξ^*, and the scenarios ξ^{α}, $\alpha \in S_1$, stored in the data set.

The factor $(K(S_1)+\sigma^2 I)^{-1}(Z(S_1)-\mu(S_1))$ in (5.15) has to be evaluated once, whereas $\mu(S_2)$ and the vector $K(S_1, S_2) \in \mathbb{R}^{|S_1|}$ must be evaluated online for each new scenario. Therefore, training requires a time cubic in the cardinality $|S_1|$ of the training set due to the matrix inversion, whereas the computation of the conditional mean λ can be done in linear time. The online computation of the variance Λ_Y would require a time quadratic in the cardinality of the training set, but following Remark 5.2, it is possible to bypass the estimation of the variance by keeping the same kernel for each component of u_t. The storage of the Gram matrix $K(S_1)$ takes a space quadratic in the cardinality of the training set.

Estimation.

In Gaussian Process regression, the mean function g with values $g(X^{\alpha})$ is often set to the constant zero-valued function, so that the terms $\mu(S_1)$, $\mu(S_2)$ do not appear in (5.15). Sometimes, the mean function is set to a linear function of the inputs X^{α}. In the present context, the values of g could also be set to constant reference decisions, for instance, to the decisions from a nominal plan (Section 2.1.1).

Selecting a kernel type automatically is not easy. In support vector machines, the problem is partially addressed by working over a set of kernels (Lanckriet et al., 2004; Micchelli and Pontil, 2005; Sonnenburg et al., 2006). Once the kernel type is chosen, the selection of the hyperparameters η can be formulated as the maximization over η of the loglikelihood of the observed data $z(S_1)$ (Mardia and Marshall, 1984), that is,

$$\begin{aligned}\ell(\eta\,;\, z(S_1)) = &-(N_1/2)\log\{2\pi\} - \tfrac{1}{2}\log\{\det(K(S_1)+\sigma^2 I)\} \\ &- \tfrac{1}{2}(z(S_1)-\mu(S_1))^T(K(S_1)+\sigma^2 I)^{-1}(z(S_1)-\mu(S_1))\end{aligned}$$

where the value of $K^{\alpha\beta} = k(X^\alpha, X^\beta)$ for $\alpha, \beta \in S_1$ depends on η. A local maximum can be found by gradient-ascent based optimization techniques. Another, Bayesian, approach consists in putting prior distributions over the hyperparameters, and make predictions by integrating out the hyperparameters.

For more details, we refer to the recent review by Nickisch and Rasmussen (2008). We do not deem it essential to discuss these techniques further inasmuch as our ultimate goal is not to find the best explanation to a training set of scenario/decisions pairs: the decisions of the training set are not the optimal decisions for the original multistage stochastic programming problem in the first place. In this chapter, our procedure for selecting a model is limited to the simulation of repaired predicted decisions using candidate kernels with fixed hyperparameter values.

5.3 *Case Study*

In this section, we consider a particular multistage stochastic program (described in detail in Section 5.3.1) of the form

$$\begin{aligned}
\text{minimize} \quad & \mathbb{E}\{\textstyle\sum_{t=1}^T \langle c_t, u_t\rangle\} \\
\text{subject to} \quad & B_1 u_1 = h_1 \ , \quad u_1 \succeq 0 \ , \\
& A_t u_{t-1} + B_t u_t = h_t \ , \quad u_t \succeq 0 \qquad \text{for } t = 2, \dots, T \ ,
\end{aligned}$$

where A_t, B_t for $t \geq 1$ denote fixed matrices of proper dimension, and where the cost coefficients c_t, the constraint right-hand sides h_t, and decision vectors u_t may depend, for $t \geq 2$, on the realization $(\xi_1, \dots, \xi_{t-1})$ of a random process $\xi = (\xi_1, \dots, \xi_T)$. As usual, the expectation is taken over ξ and can be decomposed in nested conditional expectations.

Recall that a scenario tree for ξ is a set of realizations $\{\xi^k\}_{1 \leq k \leq N}$ of ξ, along with probabilities $p^k > 0$ assigned to scenarios ξ^k and summing to 1. Recall that the branching structure of the tree causes histories $(\xi_1^k, \dots, \xi_{t-1}^k)$ to be identical among some scenarios k. Let us denote by c_t^k and h_t^k the values of c_t and h_t associated to ξ^k, noting in particular that c_1 and h_1 are necessarily constant-valued.

Now, observe that if a deterministic scenario-tree generation algorithm is chosen, and if the parameters for building a scenario tree for ξ, a scenario tree for ξ given ξ_1 (meaning that $\xi_1^k = \xi_1$ for each scenario k), a scenario tree for ξ given (ξ_1, ξ_2), ..., a scenario tree for ξ given $(\xi_1, \dots, \xi_{T-1})$, are also fixed, then these choices uniquely determine a shrinking-horizon policy $\pi^{\text{SH}} = (\pi_1^{\text{SH}}, \dots, \pi_T^{\text{SH}})$.

A shrinking-horizon policy $\pi^{\text{SH}} = (\pi_1^{\text{SH}}, \dots, \pi_T^{\text{SH}})$ is defined as follows. The mapping π_1^{SH} is a constant-valued function with value $\bar{u}_1$, where $\bar{u}_1$ corresponds to an optimal solution for u_1, relative to the following program over u_1 and u_t^k for $1 \leq k \leq N$ and $1 \leq t \leq T$,

$$\begin{aligned}
\text{minimize} \quad & N^{-1} \textstyle\sum_{k=1}^N \sum_{t=1}^T p^k \langle c_t^k, u_t^k\rangle \\
\text{subject to} \quad & B_1 u_1 = h_1 \ , \quad u_1 \succeq 0 \ , \\
& u_1^k = u_1 \qquad \text{for each } k \ , \\
& A_t u_{t-1}^k + B_t u_t^k = h_t^k \ , \quad u_t^k \succeq 0 \quad \text{for each } k \text{ and for } t \geq 2 \ , \\
& u_t^j = u_t^k \text{ for each } t \geq 2 \text{ and } j, k \text{ such that } (\xi_1^k, \dots \xi_{t-1}^k) \equiv (\xi_1^j, \dots \xi_{t-1}^j) \ .
\end{aligned}$$

The mapping π_t^{SH} for $t \geq 2$ is a function of $(\xi_1, \dots, \xi_{t-1})$, with value $\bar{u}_t$, where $\bar{u}_t$ corresponds to an optimal solution for u_t^k (any k), relative to the following program over $u_{t'}^k$ for $1 \leq k \leq N$ and $t \leq t' \leq T$,

$$\begin{array}{rl} \text{minimize} & N^{-1} \sum_{k=1}^{N} \sum_{t'=t}^{T} p^k \langle c_{t'}^k, u_{t'}^k \rangle \\ \text{subject to} & A_{t'} u_{t'-1}^k + B_{t'} u_{t'}^k = h_{t'}^k \ , \quad u_{t'}^k \succeq 0 \quad \text{for each } k \text{ and for } t' \geq t \quad , \\ & \qquad \text{where we set, for } t' = t,\ u_{t-1}^k \overset{\text{def}}{=} \bar{u}_{t-1} \quad \text{for each } k \ , \\ & u_{t'}^j = u_{t'}^k \text{ for each } t' \geq t \text{ and } j,k \text{ such that } (\xi_1^k, \dots \xi_{t'-1}^k) \equiv (\xi_1^j, \dots \xi_{t'-1}^j) \\ & \qquad \text{(thus in particular for } t' = t \text{, we have } u_t^j = u_t^k \text{ for each } j,k) \ , \end{array}$$

where N and all scenario-dependent quantities $p^k, \xi_t^k, c_t^k, h_t^k$ should here be understood as relative to the scenario tree for ξ given $(\xi_1, \dots, \xi_{t-1})$, which is built once the realization of $(\xi_1, \dots, \xi_{t-1})$ becomes available, and which instantiates, along with $\bar{u}_{t-1}$, the parameters of the program.

Our intention in this section is to take shrinking-horizon policies as the golden standard for sequential decision making, and compare them to other policies built with the techniques proposed in the chapter on a common test sample of $M = 10^4$ scenarios.

For the simplicity of the parametrization of the scenario tree building algorithm, we consider scenario trees with a uniform branching factor, and use the same branching factor for rebuilding scenario trees on the shrinking horizon. Therefore, once the discretization method for ξ_t is fixed (choices are explained in length in Section 5.3.2), a shrinking-horizon policy is uniquely determined by the branching factor. Moreover, using the same branching factor at each stage results in the following property: If the realization of $(\xi_1, \dots, \xi_{t-1})$ is identical to $(\xi_1^k, \dots, \xi_{t-1}^k)$ for some scenario k in the initial scenario tree for ξ, then the subtree rooted at the node relative to $(\xi_1^k, \dots, \xi_{t-1}^k)$ is exactly the subtree built at stage t for the scenario tree of ξ given $(\xi_1, \dots, \xi_{t-1})$. Hence, if one simulates the shrinking horizon policy with uniform branching factor on the scenario ξ^k, one will recover the decisions $u^k = (u_1^k, \dots, u_T^k)$ that were found to be optimal on the initial scenario tree for computing $\bar{u}_1$.

To a single shrinking-horizon policy π^{SH} will correspond several learned policies, obtained by different learning algorithm applied to the same training data $\{(\xi^k, u^k)\}_{1 \leq k \leq N}$, relative to the scenario tree used to optimize $\pi_1^{\mathrm{SH}} = \bar{u}_1$. Obviously, all these learned policies start with the same first-stage decision $\bar{u}_1$.

5.3.1 *Description of the Test Problem*

The test problem is a multi-product assembly problem under demand uncertainty. The multistage structure of the problem is summarized in Table 5.1: the decisions to take at each stage is put in correspondence with the available information at those stage, represented by the realization of certain random variables. The mathematical formulation of the problem is presented in Table 5.2 in nested form. The nested form is a generalization to several stages of the formulation for two-stage programs presented in Appendix D; it enables a reader to distinguish easily the constraints specific a decision stage t, that is, the actual definition of the sets $\mathcal{U}_t(\xi)$. We have put at the end of the chapter (page 104) a table that specifies the numerical value of all the parameters for the test problem (Table 5.10).

Tab. 5.1: Multi-product Assembly Problem: Multistage structure.

Stage	Available information		Decision to take	
	Description	Variables	Description	Variables
1	*No information*		Components to buy	$q_1 \in \mathbb{R}^{12}$
2	Factor 1	$\epsilon_1 \in \mathbb{R}$	Subparts to make	$v_2 \in \mathbb{R}^{12\times 8}$, $q_2 \in \mathbb{R}^8$
3	Factors 1,2	$\epsilon_1, \epsilon_2 \in \mathbb{R}$	Products to assemble	$v_3 \in \mathbb{R}^{8\times 5}$, $q_3 \in \mathbb{R}^5$
4	Factors 1,2,3	$\epsilon_1, \epsilon_2, \epsilon_3 \in \mathbb{R}$	Sales	$q_4 \in \mathbb{R}^5$

Tab. 5.2: Multi-product Assembly Problem: Nested formulation.

$$\begin{aligned}
\text{minimize} \quad & \langle c_1, q_1\rangle + \mathbb{E}_{\epsilon_1}\{Q_1(q_1,\epsilon_1)\} \\
\text{subject to} \quad & q_1 \succeq 0 \ , \\
Q_1(q_1,\epsilon_1) = \ & \min \ \langle c_2, q_2\rangle + \mathbb{E}_{\epsilon_2}\{Q_2(q_2,\epsilon_1,\epsilon_2)\} \\
\text{subject to} \quad & w_{2ij}(q_2)_j \le (v_2)_{ij} \ , \quad \textstyle\sum_j (v_2)_{ij} \le (q_1)_i \ , \\
& q_2, v_2 \succeq 0 \ , \qquad {\scriptstyle (1 \le i \le 12,\ 1 \le j \le 8)} \\
Q_2(q_2,\epsilon_1,\epsilon_2) = \ & \min \ \langle c_3, q_3\rangle + \mathbb{E}_{\epsilon_3}\{Q_3(q_3,\epsilon_1,\epsilon_2,\epsilon_3)\} \\
\text{subject to} \quad & w_{3jk}(q_3)_k \le (v_3)_{jk} \ , \quad \textstyle\sum_k (v_3)_{jk} \le (q_2)_j \ , \\
& q_3, v_3 \succeq 0 \ , \qquad {\scriptstyle (1 \le j \le 8,\ 1 \le k \le 5)} \\
Q_3(q_3,\epsilon_1,\epsilon_2,\epsilon_3) = \ & \min \ \langle c_4, q_4\rangle \\
\text{subject to} \quad & q_4 \preceq [b_0 + b_1\epsilon_1 + b_2\epsilon_2 + b_3\epsilon_3]_+ \overset{\text{def}}{=} d \ , \\
& 0 \preceq q_4 \preceq q_3 \ .
\end{aligned}$$

The test problem can be described as follows. A manufacturer can assemble 5 products P_i, for which the demand $d_i \in \mathbb{R}$ is unknown, but influenced by three random factors $\epsilon_t \in \mathbb{R}$, $t = 1,2,3$, observed at distinct decision stages (see Table 5.1). We let $d \in \mathbb{R}^5$ be the random vector representing the demand. The products are made of subparts, some of which are common among several products. There is a total of 8 distinct subparts. The subpart are themselves made of components, some of which are common among several subparts. There is a total of 12 distinct components that the manufacturer can buy.

The random demand d is assumed to be distributed according to the following model:

$$d = [b_0 + b_1\epsilon_1 + b_2\epsilon_2 + b_3\epsilon_3]_+ \tag{5.19}$$

$$\epsilon_1 \sim \mathcal{N}(0,1) \ , \quad \epsilon_2 \sim \mathcal{N}(0,1) \ , \quad \epsilon_3 \sim \mathcal{N}(0,1) \ , \tag{5.20}$$

where $b_0, b_1, b_2, b_3 \in \mathbb{R}^5$ are fixed parameters, the random variables ϵ_i are mutually independent, and $[\cdot]_+$ denotes the componentwise positive part. To relate this model with the usual notation ξ for the gradually observed random process, one has to set $\xi = (\xi_1, \xi_2, \xi_3)$ with $\xi_1 = \epsilon_1$, $\xi_2 = \epsilon_2$, $\xi_3 = (\epsilon_3, d)$.

The decision vector u is decomposed into two groups of variables. A first group of so-called strategic decisions $q_1 \in \mathbb{R}^{12}$, $q_2 \in \mathbb{R}^8$, $q_3 \in \mathbb{R}^5$ corresponds to the quantities of components, subparts and products that are bought or assembled. A second group of so-called ancillary decisions $v_2 \in \mathbb{R}^{12\times 8}, v_3 \in \mathbb{R}^{8\times 5}$ determines each quantity of component/subpart allocated to a given subpart/product in the next stage of the production process. A decision $q_4 \in \mathbb{R}^5$, corresponding to the quantity of product actually sold, and defined by $q_4 = \min\{q_3, d\}$ (elementwise minimum), is added to the group of ancillary decisions, for the convenience of the problem formulation (convexity). To summarize,

$$u = (q_1,\, v_2,\, q_2,\, v_3,\, q_3,\, q_4) \in \mathbb{R}^{166} \ , \tag{5.21}$$

with a slight abuse of notation (since v_2 and v_3 were defined as matrices).

The time horizon $T = 4$ and the total dimension of a scenario $(\epsilon_1, \epsilon_2, \epsilon_3, d) \in \mathbb{R}^8$, where d is actually a function of $(\epsilon_1, \epsilon_2, \epsilon_3) \in \mathbb{R}^3$, are small enough to let us test shrinking-horizon policies of various complexity on a test sample of significant size.

The objective to be minimized is the expected cost $\mathbb{E}\{\langle c, u\rangle\}$, where the vector

$$c = (c_1,\, 0,\, c_2,\, 0,\, c_3,\, c_4) \in \mathbb{R}^{166} \tag{5.22}$$

collects the unit cost of each decision in u, in the order determined by the decomposition (5.21). The subvectors c_1, c_2, c_3 associated to q_1, q_2, q_3 correspond to fixed production costs and are nonnegative. Zero costs are associated to the decisions v_2, v_3. The subvector c_4 associated to q_4 has negative entries that correspond to the fixed prices of the 5 products with a sign change.

The decision vector u is structured by various constraints. Besides a nonnegativity constraint $u \succeq 0$, these constraints are of two types:

$$w_{tjk}(q_t)_k \leq (v_t)_{jk} \ , \tag{5.23}$$

$$\textstyle\sum_k (v_t)_{jk} \leq (q_{t-1})_j \ . \tag{5.24}$$

Constraints (5.23) express that w_{tjk} units of j are necessary for obtaining one unit of k, where $\alpha_{tjk} \geq 0$ is a fixed parameter, j refers to a component (if $t = 1$) or a subpart (if $t = 2$), and k refers to a subpart (if $t = 1$) or a product (if $t = 2$). Note that if j does not enter in the composition of k, one has $w_{tjk} = 0$, so that (5.23) reduces to a redundant nonnegativity constraint that can be removed. Constraints (5.24) express that the total quantity of j employed in the various k cannot exceed the available quantity of j.

The relation $q_4 = \min\{q_3, d\}$ can be expressed by the constraints

$$q_4 \preceq q_3 \ , \tag{5.25}$$

$$q_4 \preceq d \tag{5.26}$$

since the components of c associated to q_4 in the objective are negative.

The constraints at each stage can thus be expressed in the format (5.5), converting the inequality constraints to equality constraints by introducing nonnegative slack variables. It is easy to see that the right-hand side h_t of (5.5) is fixed at $t = 1, 2, 3$ and depends affinely on d at $t = 4$, due to (5.26). The dependence of the feasibility sets on the demand factors ϵ_t is implicit. For instance q_3 depends on q_2 through (5.23) and (5.24), while q_2 itself depends on ϵ_1. Such dependences give a rich structure to the feasibility sets.

5.3.2 Discretization of the Random Process

This section details how the scenario trees with uniform branching factors are built. We focus on the problem of approximating $\mathcal{N}(0,1)$ by a discrete distribution on S points, specified by a support $(\hat{\epsilon}^1,\dots,\hat{\epsilon}^S)$ and associated positive probability masses $(\hat{p}^1_\epsilon,\dots,\hat{p}^S_\epsilon)$. Indeed, once the support $(\hat{\epsilon}^1,\dots,\hat{\epsilon}^S)$ and the probabilities $(\hat{p}^1_\epsilon,\dots,\hat{p}^S_\epsilon)$ of the discrete distribution are determined, the scenario tree is made of the S^3 distinct realizations of $\xi=(\epsilon_1,\epsilon_2,\epsilon_3,d)$ of the form

$$\xi^k = (\hat{\epsilon}^{i_1},\hat{\epsilon}^{i_2},\hat{\epsilon}^{i_3},[b_0+b_1\hat{\epsilon}^{i_1}+b_2\hat{\epsilon}^{i_2}+b_3\hat{\epsilon}^{i_3}]_+)$$

where the indices i_1,i_2,i_3 are valued in $\{1,\dots,S\}$, and the probability of the scenario ξ^k is given by $p^k=\hat{p}^{i_1}_\epsilon\hat{p}^{i_2}_\epsilon\hat{p}^{i_3}_\epsilon$.

Let $\hat{\epsilon}=(\hat{\epsilon}^1,\dots,\hat{\epsilon}^S)$ denote the support of the discrete distribution, treated as an optimization variable in $\mathbb{R}^S$. We will use the quadratic distortion D^2 between the discrete distribution and the target distribution $\mathcal{N}(0,1)$, defined for any $\hat{\epsilon}\in\mathbb{R}^S$ as

$$D^2(\hat{\epsilon}) = \mathbb{E}\left\{\min\nolimits_{1\le i\le S}||\hat{\epsilon}^i-\epsilon||^2\right\}\ , \tag{5.27}$$

where ϵ is a random variable following the target distribution $\mathcal{N}(0,1)$. By defining the cells $C^i(\hat{\epsilon})=\{\epsilon\in\mathbb{R}: ||\hat{\epsilon}^i-\epsilon||\le||\hat{\epsilon}^j-\epsilon||, 1\le j\le S\}$, whose boundaries have a null measure under the target probability measure, (5.27) can be written as $D^2(\hat{\epsilon})=\sum_{i=1}^S\int_{C^i(\hat{\epsilon})}||\hat{\epsilon}^i-\epsilon||^2\phi(\epsilon)d\epsilon$, with ϕ the probability density function of $\mathcal{N}(0,1)$.

If $\nabla D^2(\hat{\epsilon})=0$, that is,

$$\textstyle\int_{C^i(\hat{\epsilon})}(\hat{\epsilon}^i-\epsilon)\phi(\epsilon)d\epsilon=0\ ,\quad 1\le i\le S\ , \tag{5.28}$$

then $\hat{\epsilon}$ is called a stationary quantizer. When the distortion is minimized over $\hat{\epsilon}$ without constraint, as here where the support of the target distribution is unbounded, a local minimum of the distortion is a stationary quantizer.

On the real line, the attention can be restricted to the points $\hat{\epsilon}$ such that $-\infty<\hat{\epsilon}^1<\cdots<\hat{\epsilon}^S<\infty$, since the distortion decreases when a new point distinct from others is added to the support of the discrete distribution. Under the convention that $\hat{\epsilon}^0=-\infty$ and $\hat{\epsilon}^{S+1}=\infty$, the cell $C^i(\hat{\epsilon})$ is the closure of the interval $([\hat{\epsilon}^{i-1}+\hat{\epsilon}^i]/2,[\hat{\epsilon}^i+\hat{\epsilon}^{i+1}]/2)$.

With the univariate normal distribution, which has a strictly log-concave density, a local minimum of D^2 can be found by Newton's method (Pages and Printems, 2003), and this minimum is also a global minimum (this does not hold in the multivariate case). Optimal solutions $\hat{\epsilon}$ for values of S used in the sequel are represented on Figure 5.3. The probabilities reported on the figure are obtained by integrating the normal density over the cells C_i:

$$\hat{p}^i_\epsilon=\textstyle\int_{C^i(\hat{\epsilon})}\phi(\epsilon)d\epsilon=\Phi(\hat{\epsilon}^{i+1}/2+\hat{\epsilon}^i/2)-\Phi(\hat{\epsilon}^i/2+\hat{\epsilon}^{i-1}/2)\ ,$$

where Φ is the cumulative distribution function (cdf) of $\mathcal{N}(0,1)$. The probabilities have a closed-form expression thanks to the simple domain of integration.

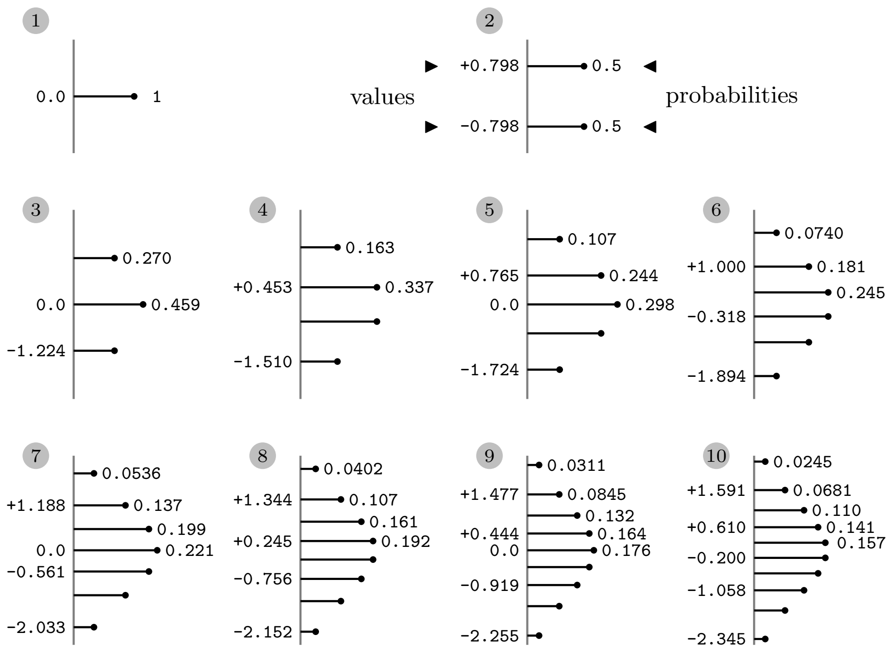

Fig. 5.3: Discretizations of $\mathcal{N}(0,1)$ for branching factors from 1 to 10, obtained by minimizing the quadratic distortion. Values that can be guessed by symmetry are not indicated.

Remark 5.3. A property of stationary quadratic quantizers is noteworthy in the context of stochastic programming. It is well known (Pages and Printems, 2003) that for any function f_0 convex in ϵ, one has, by (5.28),

$$\sum_{i=1}^{S} \hat{p}_\epsilon^i f_0(\hat{\epsilon}^i) = \sum_{i=1}^{S} \hat{p}_\epsilon^i f_0\left(\frac{\int_{C^i(\hat{\epsilon})} \epsilon\phi(\epsilon)d\epsilon}{\int_{C^i(\hat{\epsilon})} \phi(\epsilon)d\epsilon}\right) \leq \sum_{i=1}^{S} \hat{p}_\epsilon^i \int_{C^i(\hat{\epsilon})} f_0(\epsilon)\frac{\phi(\epsilon)}{\hat{p}_\epsilon^i}d\epsilon = \mathbb{E}\{f_0(\epsilon)\} \ ,$$

where the inequality holds by Jensen's inequality with the conditional density $\phi(\epsilon)/\hat{p}_\epsilon^i$. This implies that for a function f with values $f(\epsilon, x)$ convex in ϵ, one has, for any fixed x, $\sum_{i=1}^{S} \hat{p}_\epsilon^i f(\hat{\epsilon}^i, x) \leq \mathbb{E}\{f(\epsilon, x)\}$. Let $\bar{x} \in \operatorname{argmin}_x \mathbb{E}\{f(x,\epsilon)\}$. Then it holds that

$$\min_x \textstyle\sum_{i=1}^{S} \hat{p}_\epsilon^i f(\hat{\epsilon}^i, x) \leq \sum_{i=1}^{S} \hat{p}_\epsilon^i f(\hat{\epsilon}^i, \bar{x}) \leq \mathbb{E}\{f(\epsilon, \bar{x})\} = \min_x \mathbb{E}\{f(\epsilon, x)\} \ . \tag{5.29}$$

Now, as a particular function convex in ϵ, consider

$$f(\epsilon, x) = \langle c, x\rangle + \min_{\{y:\ Ax+By=C\epsilon,\ y\succeq 0\}} g(y) \tag{5.30}$$

where g is convex in y. The function f is convex in ϵ as the sum of a fixed term $\langle c, x\rangle$ and a function obtained as the composition of the affine transform $\delta = C\epsilon - Ax$ with the function $\tilde{f}(\delta) = \min_{\{y:\ By=\delta,\ y\succeq 0\}} g(y)$, which can be shown to be convex in δ (Rockafellar, 1970, Theorem 5.7). Then (5.29) becomes, restricting the minimization over x to some set X,

$$\begin{aligned} &\min_{x\in X}\{\langle c, x\rangle + \textstyle\sum_{i=1}^{S} \hat{p}_\epsilon^i \min_{\{y^i:\ Ax+By^i=C\hat{\epsilon}^i,\ y^i\succeq 0\}} g(y^i)\} \\ &\leq \min_{x\in X}\{\langle c, x\rangle + \mathbb{E}\{\min_{\{y(\epsilon):\ Ax+By(\epsilon)=C\epsilon,\ y(\epsilon)\succeq 0\}} g(y(\epsilon))\}\} \ . \end{aligned} \tag{5.31}$$

The argument can be extended by taking $g(y) = \mathbb{E}\{f_2(\epsilon_2, y)\}$, with ϵ_2 a new random variable independent of ϵ, and $f_2(\epsilon_2, y)$ defined by

$$f_2(\epsilon_2, y) = \langle c_2, y\rangle + \min_{\{z:\ A_2y+B_2z=C_2\epsilon_2,\ z\succeq 0\}} g_2(z)$$

where g_2 is convex in z. Given a stationary quantizer for ϵ_2, with values $\hat{\epsilon}_2^j$ and probabilities $\hat{p}_2^j$, $j = 1, \dots, S_2$, it holds by (5.29) and the convexity of f_2 in ϵ_2 that

$$\min_{y^i \in Y^i(x)} \sum_{j=1}^{S_2} \hat{p}_2^j f_2(\hat{\epsilon}_2^j, y^i) \le \min_{y^i\in Y^i(x)} \mathbb{E}\{f_2(\epsilon_2, y^i)\}\ , \qquad (5.32)$$

where $Y^i(x) = \{y^i :\ Ax + By^i = C\hat{\epsilon}^i,\ y^i \succeq 0\}$. Let $\bar{x}$ be an optimal solution to the minimization over $x \in X$ of the left-hand side of (5.31). One then obtains the chain of inequalities

$$\begin{aligned}
&\min_{x\in X}\{\langle c, x\rangle + \textstyle\sum_{i=1}^S \hat{p}_\epsilon^i \min_{\{y^i:\ Ax+By^i=C\hat{\epsilon}^i,\ y^i\succeq 0\}}\{\langle c_2, y^i\rangle \\
&\qquad\qquad + \textstyle\sum_{j=1}^{S_2} \hat{p}_2^j \min_{\{z^{ij}:\ \{A_2y^i+B_2z^{ij}=C_2\hat{\epsilon}_2^j,\ z^{ij}\succeq 0\}} g_2(z^{ij})\}\} \\
\le\ &\qquad \langle c, \bar{x}\rangle + \textstyle\sum_{i=1}^S \hat{p}_\epsilon^i \min_{\{y^i:\ A\bar{x}+By^i=C\hat{\epsilon}^i,\ y^i\succeq 0\}}\{\langle c_2, y^i\rangle \\
&\qquad\qquad + \textstyle\sum_{j=1}^{S_2} \hat{p}_2^j \min_{\{z^{ij}:\ \{A_2y^i+B_2z^{ij}=C_2\hat{\epsilon}_2^j,\ z^{ij}\succeq 0\}} g_2(z^{ij})\} \\
=\ &\qquad \langle c, \bar{x}\rangle + \textstyle\sum_{i=1}^S \hat{p}_\epsilon^i \min_{y^i\in Y^i(\bar{x})} \textstyle\sum_{j=1}^{S_2} \hat{p}_2^j f_2(\hat{\epsilon}_2^j, y^i) \\
\le\ &\qquad \langle c, \bar{x}\rangle + \textstyle\sum_{i=1}^S \hat{p}_\epsilon^i \min_{y^i\in Y^i(\bar{x})} \mathbb{E}\{f_2(\epsilon_2, y^i)\} \\
&= \min_{x\in X}\{\langle c, x\rangle + \textstyle\sum_{i=1}^S \hat{p}_\epsilon^i \min_{y^i\in Y^i(x)} g(y^i)\} \\
&\le \min_{x\in X}\{\langle c, x\rangle + \mathbb{E}_\epsilon\{\min_{\{y(\epsilon):\ Ax+By(\epsilon)=C\hat{\epsilon},\ y(\epsilon)\succeq 0\}} \mathbb{E}_{\epsilon_2}\{f_2(\epsilon_2, y(\epsilon))\}\}\} \\
&= \min_{x\in X}\{\langle c, x\rangle + \mathbb{E}_\epsilon\{\min_{\{y(\epsilon):\ Ax+By(\epsilon)=C\hat{\epsilon},\ y(\epsilon)\succeq 0\}}\{\langle c_2, y(\epsilon)\rangle \\
&\qquad\qquad + \mathbb{E}_{\epsilon_2}\{\min_{\{z(\epsilon,\epsilon_2):\ A_2y(\epsilon)+B_2z(\epsilon,\epsilon_2)=C_2\epsilon_2,\ z(\epsilon,\epsilon_2)\succeq 0\}} g_2(z)\}\}\}\ ,
\end{aligned}$$

where the last inequality follows from (5.31).

By induction, the result can further be extended to several decision stages. □

In Remark 5.3, a class of multistage programs has been identified, for which a *single* scenario-tree approximation based on quadratic quantization yields a lower bound on the exact optimal value of the program.

For this result to hold, the stagewise independence assumption between the random variables ϵ, ϵ_2, ..., is essential. The function $f(x, \epsilon)$ in (5.30) has to be convex in ϵ, preventing us to consider, instead of $g(y)$, a general function $g(y, \epsilon)$, as would be the case if the expectation in the definition of g were conditioned on ϵ. The only dependence of g on ϵ is through the value of its argument y, which depends on the realization of ϵ.

Now, there exists a formulation trick that allows to pass the value of ϵ to functions at subsequent stages. It suffices to extend the decision vector y to the vector $y^+ = (y, y^\epsilon)$, where y^ϵ is a dummy decision variable subject to the constraint $y^\epsilon = \epsilon$. The value of ϵ can then be passed to the function g through y^+ itself, and by the same mechanism to any subsequent function inside the nested expectations.

In fact, the multi-product assembly problem described in Section 5.3.1 could be put under that form if (5.20) were replaced by $d = b_0 + b_1\epsilon_1 + b_2\epsilon_2 + b_3\epsilon_3$. Indeed, in the reasoning of Remark 5.3, the transform $\delta = C\epsilon - Ax$ can be extended to $\delta = C\epsilon + D - Ax$, which is also an affine transform of ϵ but allows fixed right-hand sides when $C = 0$. With

Tab. 5.3: Minimum of the approximate programs.

Branching factor	Optimal value	Cpu time (seconds)
1	-805.73	1.8
2	-450.89	1.8
3	-397.80	5.0
4	-388.57	8.0
5	-383.36	17.2
6	-379.85	40.7
7	-378.09	79.8
8	-377.23	177.5
9	-376.91	353.5
10	-376.56	670.6

the extension trick, it is possible to pass the value of ϵ_1, ϵ_2, ϵ_3 to the last stage, and to express d through the linear equality constraint $d = b_0 + b_1\epsilon_1 + b_2\epsilon_2 + b_3\epsilon_3$.

Unfortunately, the lower bound certificate cannot be extended to the case where d is defined by (5.20): the value of the last stage is convex in d but not in ϵ_3. We expect, however, that when the conditional probability of having all components of d not truncated is large enough (we refer to the probability $\mathbb{P}\{d \succ 0\} = \mathbb{P}\{\cap_{j=1}^{5} (b_3)_j \epsilon_3 > -\lambda_j\}$ when $\lambda = (b_0 + b_1\epsilon_1 + b_2\epsilon_2) \succeq 0$), one is close to the case where d is affine in ϵ_1, ϵ_2, ϵ_3, and thus close to being able to certify that the quadratic quantization yields a lower bound. When one or several components of λ_j are close to 0 or below, then it is likely that the optimal choice of q_3 will attempt to redirect the assembly to products with the largest expected profit $\mathbb{E}\{|(c_4)_j|(q_4)_j - (c_3)_j(q_3)_j\}$, and thus to favor products with a larger conditional expected demand, which happens to be the products that follow the affine demand model more closely — potentially diminishing the impact of a discretization bias in the wrong direction. By bias in the wrong direction, we mean this: If we were able to dynamically adjust a quantizer for the distribution of the components d_j to make it stationary given the values of ϵ_1 and ϵ_2, so as to take the expectation over d rather than ϵ_3, then the values of the adjusted quantizer would be greater than the values of the fixed quantizer induced by the fixed quantization of $\epsilon_1, \epsilon_2, \epsilon_3$, that neglects the truncation of d at 0.

Empirically, on our problem data, the optimal value of the scenario-tree approximations with uniform branching factor $S = 1, 2, \ldots$ increases with S and stabilizes at a certain level for higher values of S. This strongly suggests that on our problem data, the quadratic quantization approach consistently provides lower bounds on the value of the exact multistage program (Table 5.3). The time taken by the numerical optimization algorithm for solving the successive approximations has also been indicated on Table 5.3, so as to provide an indication of the increasing difficulty of solving programs posed on larger scenario trees.

5.3.3 Shrinking-Horizon Policies on the Test Sample

As already mentioned, the present problem is simple enough to let us simulate shrinking-horizon policies on mutually independent test scenarios. We considered one cpu day as the time limit beyond which the simulation time of one policy on 10^4 scenarios is not acceptable. Simulation results for 4 shrinking-horizon policies on a fixed test sample of 10^4 scenarios are reported on Table 5.4 (page 100). The average cpu time for the evaluation of the sequence of decisions on one new scenario is also indicated on Table 5.4, clearly illustrating the growing complexity of simulating shrinking-horizon policies. The policy with branching factor 7 takes 6.5 seconds per scenario, that is, $6.5{\cdot}10^4/(3600{\cdot}24) \simeq 0.75$ days to be evaluated on the test sample.

The reported empirical averages on the test sample are our estimate for the expected cost of the policies. The standard error, defined as the standard deviation of the costs on the test sample divided by the square root of the test sample size, indicates the order of magnitude of our uncertainty about the true value of the policies as solutions to the multistage program.

The apparent plateau of performance beyond a branching factor of 5 suggests that the shrinking-horizon policy with branching factor 5 already attains performances that are almost optimal, and this is confirmed by comparing the empirical average on the test sample to the lower bounds of Table 5.3, in particular the best bound obtained on the single program with the largest scenario tree (branching factor 10).

Remark 5.4. As the same test sample is used for each policy, the difference of costs between pairs of policies should be significant enough to allow us to rank the various policies reliably. On Table 5.9 (page 101), the reported standard error is the standard deviation, on the test sample, of the difference of costs between each pair of policies considered in the section, divided by the square root of the test sample size. Thus, for instance, a confidence interval for the difference of average cost between shrinking-horizon policies with branching factors 3 and 5 could be built by considering that the estimator for the difference is approximately normally distributed with a standard deviation of 0.70. For some pairs of policies, the standard errors reported in Table 5.9 are larger, but then they correspond to policies with a larger difference in their empirical performance.

In general, the uncertainty about the true value of the difference of expected costs among policies appears to be considerably smaller than the uncertainty about the level of the expected cost itself, and actually small enough to justify with hindsight the choice of the test sample size for ranking the policies. With a test sample 4 times larger, we would be able to improve our statistical estimates by a factor of 2, but then 4 cpus would be needed to simulate the shrinking-horizon policies on the test sample in less than one day. □

Remark 5.5. The shrinking-horizon policy with branching factor 1 (that uses a single scenario to represent the future, corresponding to the mean scenario conditionally to the information state, and thus implements a Model Predictive Control approach) is already far better than a two-stage approximation strategy, that would consist in

- relaxing the multistage program to a two-stage program, with first-stage decision $(q_1, v_2, q_2, v_3, q_3)$ and second-stage decision q_4 adjusted to the observation of $(\epsilon_1, \epsilon_2, \epsilon_3)$, and then
- implementing the resulting optimal first-stage decision $(q_1, v_2, q_2, v_3, q_3)$ in open-loop (that is, neglecting the observations of ϵ_1 and ϵ_2), followed by the optimal second-stage decision $q_4 = \min\{d, q_3\}$ given the observation of d.

When simulating such a policy on the test sample, using a first-stage decision computed on the scenario tree with branching factor 10 (and simply imposing that the decisions q_1, v_2, q_2, v_3, q_3 are common to every scenario), we obtain an empirical cost equal to -261.39 (standard deviation of the estimate: 6.15), far worse than the value -305.48 of the simplest shrinking-horizon policy. Such a test confirms the interest of taking into account the available information on the demand and adjust the production process online. It also allows to compute quickly a lower bound on the value of multistage stochastic programming (VMS): the VMS can be estimated as at least the difference of performance between the simplest shrinking-horizon policy and the policy based on the two-stage approximation. □

5.3.4 Performances of Learned Policies

In the following experiments, we test policies that are built with the data extracted from a given single scenario-tree approximation solved to optimality (the optimal value of which being already reported in Table 5.3). We consider 3 such data sets, namely, the ones obtained with branching factors 3, 5, and 7 respectively. Larger data sets are advantageous from the statistical learning point of view, and at the same time they provide better recourse decision examples, due to the finer discretization of the random process used in the approximate stochastic programs.

The first-stage decision of the learned policies are exactly that of the corresponding shrinking-horizon policy. The policy for the last stage decision is always set to the optimal policy with decisions $q_4 = \min\{q_3, d\} \in \mathbb{R}^5$. It remains to learn a mapping π_1 from $\epsilon_1 \in \mathbb{R}$ to $q_2 \in \mathbb{R}^8$, and a mapping π_2 from $(\epsilon_1, \epsilon_2) \in \mathbb{R}^2$ to $q_3 \in \mathbb{R}^5$. Indeed, once q_{t-1} and q_t are determined, the value of v_t can be deduced by solving a simple optimization program, that had to be solved anyway to ensure that a predicted decision $\hat{q}_t$ is feasible, and to repair it if necessary.

Policies based on the Joint Gaussian Model.

First, we test the simple approach described in Section 5.2. We estimate the mean and the covariance matrix of a joint Gaussian model for $(\epsilon_1, \epsilon_2, \epsilon_3, q_1, q_2, q_3)$ from the considered data set. The value of the parameter ϵ' in (5.14) is set to 0.01 in all the experiments. The predicted conditional densities of q_2 given (ϵ_1, q_1), and of q_3 given $(\epsilon_1, \epsilon_2, q_2)$, are computed with the conditioning formulae (5.12). The decisions are then inferred by solving programs of the form (5.7), as described in Section 5.1.2. The optimized variables are q_t and v_t, structured by the constraints (5.23), (5.24).

The performances of those policies are reported on Table 5.5. The branching factor identifies the data set from which policies are learned.

The performance of the learned policies are worse than that of the corresponding shrinking-horizon policies reported in Table 5.4, but already much better than the score of the policy with a fixed optimized production plan described in Remark 5.5.

Policies based on the Gaussian Process Model.

Next, we test the nonparametric approach described in Section 5.2.2. Experiments were limited to the case of a radial basis kernel with a common bandwidth parameter $r > 0$ set beforehand for each component of the decision vectors. For the components of the predictive conditional mean of q_2, we used the kernel with values

$$k(\epsilon_1^i, \epsilon_1^j) = \exp\{-(\epsilon_1^i - \epsilon_1^j)^2/(2r^2)\} \ ,$$

and for the components of the predictive conditional mean of q_3, we used the kernel with values

$$k'(\epsilon_1^i, \epsilon_2^i, \epsilon_1^j, \epsilon_2^j) = \exp\{-\textstyle\sum_{t'=1}^2 (\epsilon_{t'}^i - \epsilon_{t'}^j)^2/2r^2\} = k(\epsilon_1^i, \epsilon_1^i) \cdot k(\epsilon_2^i, \epsilon_2^j) \ .$$

We did not try to determine the best value of the bandwidth parameter r from the data set, but rather tested the resulting policies on the test sample. The jitter parameter σ^2 that enters the expression of the predictive conditional means (5.15) was always set to 0.01.

The performance of the policies with the best found value of r — which depends on the size of the data set from which the policy is learned — are reported in Table 5.6. If we compare the results of Table 5.6 to the results of Table 5.5, we observe that on a same training set (identified by the branching factor), the selected policy based on the Gaussian Process model is better, in the case of branching factors 3 and 7, than the corresponding policy based on the joint Gaussian model, and in fact a lot better with the branching factor 3, corresponding to the smallest studied training set. On the training set with the branching factor 5, however, the policy based on the joint Gaussian model is better. In fact, that latter policy seems to dominate the 3 policies of Table 5.6, suggesting that the simple approach based on the joint Gaussian model was worth investigating.

Finally, we tested the idea of emulating input-dependent bandwidth choices by using kernels with values

$$\begin{aligned} k(\epsilon_1^i, \epsilon_1^j) &= \exp\{-[\Phi(\epsilon_1^i) - \Phi(\epsilon_1^j)]^2/(2r^2)\} \ , \\ k'(\epsilon_1^i, \epsilon_2^i, \epsilon_1^j, \epsilon_2^j) &= k(\epsilon_1^i, \epsilon_1^i) \cdot k(\epsilon_2^i, \epsilon_2^j) \ , \end{aligned}$$

where Φ is the cumulative distribution function of $\mathcal{N}(0,1)$. In fact, since each ϵ_t follows $\mathcal{N}(0,1)$, it holds that $\Phi(\epsilon_t)$ is uniformly distributed on the interval $[0,1]$. It seems then wise to use a constant bandwidth r on this transformed input space, rather than on the original input space.

The performance of the policies with the best found value of r — which happened to be independent of the size of the data set from which the policy is learned — are reported in Table 5.7. If we compare the results of Table 5.7 to the results of Table 5.6, we observe

Tab. 5.4: Simulation results for shrinking-horizon policies.

Branching	Empirical results on the test sample		Cpu time (sec.)
factor	Average	Standard error	per scenario
1	-305.48	4.88	0.9
3	-369.52	5.91	1.7
5	-374.34	6.37	3.1
7	-374.56	6.17	6.5

Tab. 5.5: Results for policies based on the joint Gaussian model.

Branching	Empirical results on the test sample		Cpu time (sec.)
factor	Average	Standard error	per scenario
3	-307.57	5.27	1.3
5	-360.81	6.13	1.3
7	-356.07	5.88	1.3

Tab. 5.6: Results for policies based on the Gaussian Process model.

Branching	Empirical results on the test sample		Cpu time (sec.)
factor	Average	Standard error	per scenario
3	-347.49	5.58	1.1
5	-348.94	6.12	1.1
7	-357.63	6.02	1.2

Tab. 5.7: Gaussian Process model with a transformed input space.

Branching	Empirical results on the test sample		Cpu time (sec.)
factor	Average	Standard error	per scenario
3	-359.50	5.73	1.2
5	-368.76	6.31	1.2
7	-363.26	6.05	1.2

Tab. 5.8: Gaussian Process with a transformed input space and a fast repair procedure.

Branching	Empirical results on the test sample		Cpu time (sec.)
factor	Average	Standard error	per scenario
3	-359.87	5.74	0.001
5	-371.10	6.33	0.001
7	-370.28	6.12	0.001

Tab. 5.9: Standard error of pairwise differences on the test sample.

		Tab. 5.4			Tab. 5.5			Tab. 5.6			Tab. 5.7			Tab. 5.8		
		3	5	7	3	5	7	3	5	7	3	5	7	3	5	7
Tab. 5.4	1	1.94	2.39	2.17	2.03	2.22	2.02	1.74	2.13	2.06	1.88	2.33	2.06	1.88	2.35	2.13
	3		0.70	0.43	1.81	0.97	0.96	0.72	0.81	0.59	0.42	0.66	0.43	0.42	0.68	0.43
	5			0.33	2.21	0.93	1.13	1.23	0.80	0.76	0.97	0.27	0.56	0.96	0.26	0.46
	7				2.00	0.88	1.00	0.98	0.69	0.57	0.71	0.32	0.32	0.71	0.35	0.24
Tab. 5.5	3					1.98	1.72	1.33	1.72	1.83	1.68	2.15	1.95	1.68	2.16	1.95
	5						0.37	1.12	0.72	1.07	1.12	0.88	0.92	1.12	0.88	0.88
	7							0.94	0.77	1.09	1.05	1.07	0.98	1.05	1.08	0.97
Tab. 5.6	3								0.88	0.79	0.63	1.17	0.93	0.63	1.19	0.95
	5									0.68	0.99	0.76	0.75	0.99	0.76	0.69
	7										0.73	0.68	0.51	0.73	0.70	0.53
Tab. 5.7	3											0.87	0.62	0.01	0.89	0.63
	5												0.42	0.87	0.07	0.30
	7													0.62	0.46	0.19
Tab. 5.8	3														0.89	0.63
	5															0.33
	7															

that on a same training set, the policies using the kernel on the transformed space are significantly better than the policies using the kernel on the original input space.

Therefore, these experiments illustrate that the performances of the policies based on the Gaussian Process model are sensitive to the choice of the kernel. Depending on the efforts that one is ready to make to test different choices of kernels, one can thus expect to obtain good policies with the Gaussian Process model, perhaps even with small data sets, as it was the case here with the training set relative to branching factor 3.

Discussion.

In terms of optimality, the results obtained here suggest that the learned policies are able to attain performances that are quite decent with respect to the shrinking-horizon policies. With trees of branching factor 3, for instance, the policy based on Gaussian Process regression (with a good choice for the kernel) attains an average cost of about

-360 on the test sample, while the corresponding shrinking-horizon policy attains -370.

In terms of simulation times, with our Matlab implementation that calls cvx for formulating and solving all programs, the learned policies are penalized by the need to repair the predictions by solving a quadratic program, and the simulation times are thus similar to the time taken by simulating the shrinking-horizon policy with branching factor 1.

These results led us to try to replace the generic MAP repair procedure of Section 5.1 by a problem-specific, faster heuristic. In the present context, a possible heuristic consists in fixing an ordering of the components of q_t a priori, and then using the stocks q_{t-1} as needed to reach the nominal level $(\hat{q}_t)_j$ predicted by the learned policy, or to a lower level if one needed component of q_{t-1} gets depleted. The priority order is a hyper-parameter of the repair procedure, that can be tested; our prior belief is that products with higher profit per unit should be given a higher priority to the available stocks of components.

On the test sample, this new repair procedure combined with the Gaussian model turns out to degrade the performance of the policy considerably. But combined with the Gaussian process model, the performance is maintained (with the best found ordering for the repair procedure), suggesting that the predictions of the Gaussian Process model are precise enough to mitigate the potential inaccuracies of the repair procedure (Table 5.8).

Remark 5.6. It is a recurrent observation on our tables that the policies learned from the data set with branching factor 5 slightly dominate those learned with branching factor 7. One possible explanation is that despite its smaller cardinality, the first data set contains better examples of decisions. In particular, the first-stage decision may be better, or at least more robust to inaccuracies in the subsequent recourse decisions. In fact, we have often observed that in two-stage programs, the exact value of the first-stage decision optimal with respect to an approximate program built with a deterministic method can actually be degraded by using more discretization points, by a simple effect of luck in the selection of the values. □

We can now claim that the best learned policy for our problem is the middle policy of Table 5.8. Thanks to the high efficiency in the evaluation of this learned policy with the fast repair procedure, we are able to test the policy on a new, independent test sample of 10^6 scenarios.

The empirical average of the cost of the policy on this new test sample is -371.87, estimated with standard error 0.63. The simulation of the policy on the new independent test sample takes about 15 minutes in cpu time. With a confidence of approximately 95 %, the exact value of the selected policy lies in the 2-standard error interval $[-370.61, -373.14]$.

5.4 Conclusions

In this chapter, alternative methods for learning policies from data sets of scenario-decisions pairs were explored, especially methods based on Gaussian Process regression. The framework of Gaussian Processes was found attractive for several reasons: the predictions are relatively easy to compute (with small data sets, or in fact with kernels that induce sparse Gram matrices), and are not based on probabilistic assumptions concerning the way the scenarios of a data set were generated, in particular independence assumptions. This last observation is important, because the scenarios of a data set usually come from a scenario tree built by conditional sampling or by deterministic methods, and as such, are not independent. It is also true that the sequence of decisions associated to a scenario actually depends, through the optimization of the decisions, on the other scenario/decisions pairs present in the tree, so that we may be far from a situation where each scenario/decisions pair in the data set could be viewed as generated independently from some unknown probability distribution. Our case study suggests that Gaussian processes can be combined gracefully with scenario-tree generation methods, with choices guided by the knowledge on the way inputs were distributed or generated.

The MAP repair procedure expounded in the beginning of the chapter is a repair procedure which is generic, but complicates the online evaluation of a learned policy. In the next chapter, we review in detail the theory on Euclidian projections, and investigate to which extent it is possible to accelerate the algorithm that computes that kind of projection mapping by exploiting a data set of examples of projections already computed.

Nevertheless, our experiment in the present chapter seems to show that when the feasibility sets are described by many constraints, it is better, from the point of view of the computational complexity, to try to tailor a simple heuristic to restore the feasibility of the decisions and obtain policies that are simple to evaluate, than to resort to a generic procedure based on online optimization.

Tab. 5.10: Multi-product assembly problem: Values of the parameters in Table 5.2.

$$c_1 = \begin{bmatrix} 0.25 & 1.363 & 0.8093 & 0.7284 & 0.25 & 0.535 & 0.25 & 0.25 & 0.25 & 0.4484 & 0.25 & 0.25 \end{bmatrix}^T$$

$$c_2 = \begin{bmatrix} 2.5 & 2.5 & 2.5 & 2.5 & 13.22 & 2.5 & 3.904 & 2.5 \end{bmatrix}^T$$

$$[w_2] = \begin{bmatrix} 0.4572 & 0 & 4.048 & 0 & 0 & 0 & 0.8243 & 11.37 \\ 0 & 0 & 0.7674 & 0.5473 & 0.3776 & 0 & 0 & 0 \\ 0.4794 & 0 & 0.4861 & 1.223 & 0 & 1.475 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5114 & 0.3139 & 0 & 0 \\ 0 & 12.29 & 1.378 & 0 & 0.3748 & 0.4554 & 0 & 0 \\ 0.7878 & 0 & 0.293 & 1.721 & 0 & 0 & 0 & 0 \\ 1.504 & 0.4696 & 0.248 & 0 & 0.1852 & 0 & 0.3486 & 0 \\ 0 & 1.204 & 0 & 0.7598 & 0.452 & 0 & 0 & 0 \\ 0 & 0 & 0.2515 & 0.3753 & 0.6249 & 0 & 1.248 & 0 \\ 1.545 & 0 & 0 & 0 & 0 & 0 & 0.2732 & 0 \\ 0 & 0 & 0 & 0.6597 & 0 & 2.525 & 0 & 0 \\ 0 & 0 & 1.595 & 0 & 0 & 1.51 & 1.041 & 0.9847 \end{bmatrix}$$

$$c_3 = \begin{bmatrix} 3.255 & 2.5 & 2.5 & 8.418 & 2.5 \end{bmatrix}^T$$

$$[w_3] = \begin{bmatrix} 0 & 1.223 & 0.6367 & 0 & 0 \\ 0 & 0 & 0 & 1.111 & 0 \\ 0 & 0 & 0.4579 & 0 & 0 \\ 0 & 0.1693 & 0.6589 & 0 & 0 \\ 0.5085 & 2.643 & 0 & 0 & 0 \\ 0.4017 & 0 & 0 & 0 & 0 \\ 0 & 0.7852 & 85.48 & 0 & 0 \\ 0 & 0 & 0 & 0.806 & 0.5825 \end{bmatrix}$$

$$c_4 = \begin{bmatrix} -21.87 & -98.16 & -31.99 & -10 & -10 \end{bmatrix}^T$$

$$b_0 = \begin{bmatrix} 13.9 & 12.86 & 18.21 & 10.14 & 17.21 \end{bmatrix}^T$$

$$b_1 = \begin{bmatrix} 9.708 & 9.901 & 7.889 & 4.387 & 4.983 \end{bmatrix}^T$$

$$b_2 = \begin{bmatrix} 2.14 & 6.435 & 3.2 & 9.601 & 7.266 \end{bmatrix}^T$$

$$b_3 = \begin{bmatrix} 4.12 & 7.446 & 2.679 & 4.399 & 9.334 \end{bmatrix}^T$$

Chapter 6

Learning Projections on Random Polyhedra

Recent advances in numerical optimization algorithms (Nesterov, 2007; Nemirovski et al., 2009) seem to suggest that two very different categories of convex feasibility sets can be distinguished: the sets on which the Euclidian projection (or its generalization via Bregman divergences) can be computed in closed-form, and the sets for which evaluating projections requires the use of standard iterative methods.

In many applications, the feasibility set of interest is a convex polyhedron, that is, a set described by a finite number of linear equality and linear inequality constraints, for which Euclidian projectors in closed-form are typically not available. In this chapter, we consider the fundamental operation of evaluating the Euclidian projection of the origin (zero vector) on a random convex polyhedral set. We study a subclass of that problem in depth, namely, a subclass related to the MAP repair procedure evaluated in the case study of chapter 5. The analysis suggests an algorithm able to predict exact projections by generalizing information from a data set of examples of projections. We say that the algorithm is able to learn projections, even if strictly speaking, the algorithm knows exactly to which extent it can generalize the examples already encountered, so that when it is unable to return an exact result for the projection, it can simply call a standard optimization procedure.

The overall goal of the chapter is less to build an efficient implementation of the studied approach, than to identify its limitations, inasmuch as this latter perspective may also shed light on limitations of learning applied to data sets of minimizers.

The chapter is organized as follows. Section 6.1 motivates the studied problem. Section 6.2 presents geometrical insights, and Section 6.3 builds on these insights to study the properties of the projections. Section 6.4 presents algorithms derived from those results. Section 6.5 evaluates empirically on a series of random problems the circumstances for the success of the approach, and Section 6.6 concludes with references to related work.

We have written proofs for a series of propositions collected in Sections 6.1, 6.2, 6.3, as this is by this mechanism that we have come to the ideas of Sections 6.4 and 6.5. The proofs have been established independently of the existing literature. With hindsight, we believe that the results of Sections 6.1, 6.2, 6.3 are natural and rather standard (see, for instance, Facchinei and Pang (2003); Dontchev and Rockafellar (2009)), while being sometimes rediscovered in some communities (see Section 6.6).

Notations.

In the sequel, we use the following notations.

- $A^T \in \mathbb{R}^{n\times m}$ is the transpose of $A \in \mathbb{R}^{m\times n}$.
- $||z|| = (z^T z)^{1/2} = \langle z, z\rangle^{1/2}$ is the Euclidian norm of $z \in \mathbb{R}^n$.
- $\mathbb{B} = \{z : ||z|| \leq 1\}$ is the closed unit ball in $\mathbb{R}^n$ with n understood from the context.
- For a scalar ρ and a set B, ρB stands for the set $\{\rho v : v \in B\}$.
- For $v_1 \in \mathbb{R}^n$ and a set $B_2 \subset \mathbb{R}^n$, $v_1 + B_2$ stands for the set $\{v_1 + v_2 : v_2 \in B_2\}$.
- For sets $B_1, B_2 \subset \mathbb{R}^n$, $B_1 + B_2$ stands for the set $\{v_1 + v_2 : v_1 \in B_1, v_2 \in B_2\}$. If B_1 is a singleton $B_1 = \{v_1\}$, we write $v_1 + B_2$ rather than $\{v_1\} + B_2$.
- For $x = [x_1 \ \ldots \ x_n]^T$ and $y = [y_1 \ \ldots \ y_n]^T \in \mathbb{R}^n$, $x \preceq y$ means $x_i \leq y_i$, $1 \leq i \leq n$, and $x \prec y$ means $x_i < y_i$, $1 \leq i \leq n$.
- Given $x = [x_1 \ldots x_n]^T \in \mathbb{R}^n$, x_+ (or $[x]_+$) denotes the vector in $\mathbb{R}^n$ with components $\max\{0, x_i\}$, $1 \leq i \leq n$.
- Given $x = [x_1 \ \ldots \ x_n]^T \in \mathbb{R}^n$ and a subset I of $\{1, \ldots, n\}$ of cardinality $|I|$, the vector $x_I \in \mathbb{R}^{|I|}$ is the subvector of x that stacks the components x_i such that $i \in I$. For a matrix $A \in \mathbb{R}^{n\times m}$ with rows $a_1^T, \ldots, a_n^T$, the matrix $A_I \in \mathbb{R}^{|I|\times m}$ is the submatrix of A that stacks the rows a_i^T of A such that $i \in I$.

6.1 Problem Statement

We consider the following parametric optimization program over $y \in \mathbb{R}^m$,

$$\mathcal{P}(x(\omega)) : \quad \text{minimize} \quad f(y) = \tfrac{1}{2}||y||^2 \quad \text{subject to} \quad Ay \preceq x(\omega) \ , \tag{6.1}$$

assuming that the parameter is the realization $x(\omega) \in \mathbb{R}^s$ of a random variable x drawn from some unknown but fixed probability distribution, and that $A \in \mathbb{R}^{s\times m}$ is a fixed matrix.

We are interested in the prediction of the optimal solution $y^*(\omega)$ to $\mathcal{P}(x(\omega))$, given $x(\omega)$, assuming that we know $\mathcal{P}$ (we do not have to estimate A, for instance). This problem could be addressed from a machine learning point of view by trying to learn a hypothesis h in some hypothesis space $\mathcal{H}$ that approximates well the optimal solution $y^*(\omega)$, in the sense that the distance between $h(x(\omega))$ and the feasibility set

$$\mathcal{C}(x(\omega)) \overset{\text{def}}{=} \{y \in \mathbb{R}^m : Ay \preceq x(\omega)\} \tag{6.2}$$

is small, and the regret $||h(x(\omega))||^2/2 - ||y^*(\omega)||^2/2$ is small.

However, the interest of a prediction $y^*(\omega)$ which is only "nearly feasible" remains hard to define in the absence of a precise interpretation of the constraints in the context of an application. Without renouncing totally to that possible avenue (results that could be useful in that perspective are also given in this chapter), we find it more adequate to look first for approaches that could accelerate the repeated evaluation of the optimal solution of $\mathcal{P}(x(\omega))$ for a sequence of realizations of x, that is, in some sense, build a self-improving algorithm (Ailon et al., 2006).

In the sequel, we assume that $x(\omega)$ is valued in the set

$$\operatorname{dom}\mathcal{C} \overset{\text{def}}{=} \{x \in \mathbb{R}^s : \mathcal{C}(x) \neq \varnothing\} \ , \tag{6.3}$$

called the *domain of* $\mathcal{C}$, with $\mathcal{C}$ interpreted as a set-valued mapping $\mathcal{C} : \mathbb{R}^s \rightrightarrows \mathbb{R}^m$ with values $\mathcal{C}(x)$ (Dontchev and Rockafellar, 2009) (see Appendix B, Definition B.8). In probabilistic terms, we assume that the support of the distribution of x is in $\operatorname{dom}\mathcal{C}$. We do not assume that the support of the distribution of x is bounded, although some specific results can be established in that case.

The setting covers a large class of parametric, strictly convex quadratic programs, as shown by the following proposition and its corollary.

6.1 Proposition. *Let $S \in \mathbb{R}^{m\times m}$ be a positive definite matrix, let $F \in \mathbb{R}^{s\times m}$ be a matrix, and let $u \in \mathbb{R}^m$, $v \in \mathbb{R}^s$ be vectors. The quadratic program over $z \in \mathbb{R}^m$,*

$$\textit{minimize} \quad \tfrac{1}{2}z^T S z + u^T z \quad \textit{subject to} \quad Fz \preceq v \ , \tag{6.4}$$

becomes, with a suitable change of variables, the problem of projecting (with respect to the Euclidian metric) the origin 0 on some polyhedral set.

Proof. Let $S = R^T R$ be the Cholesky factorization of S (where R is upper triangular). Let $z = R^{-1}y - S^{-1}u$. By substitution, we obtain a program over $y \in \mathbb{R}^m$,

$$\text{minimize} \quad \tfrac{1}{2}y^T y - \tfrac{1}{2}u^T S^{-1} u \quad \text{subject to} \quad FR^{-1}y \preceq v + FS^{-1}u \ ,$$

where the constant term $-u^T S^{-1} u/2$ can be dropped. Hence the program on z is equivalent to the evaluation of the Euclidian projection of $0 \in \mathbb{R}^m$ on the set $\mathcal{C}(x) = \{y \in \mathbb{R}^m : Ay \preceq x\}$ with $A = FR^{-1}$ and $x = v + FS^{-1}u$. Assuming that (6.4) is feasible and thus $\mathcal{C}(x)$ is nonempty, the optimal solution z^* to (6.4) is recovered from the optimal solution y^* using $z^* = R^{-1}y^* - S^{-1}u$. □

6.2 Corollary. *The parametric optimization program over $y \in \mathbb{R}^m$ with parameters $u(\omega) \in \mathbb{R}^m$, $v(\omega) \in \mathbb{R}^s$,*

$$\mathcal{Q}(u(\omega), v(\omega)) : \quad \textit{minimize} \quad \tfrac{1}{2}z^T S z + u(\omega)^T z \quad \textit{subject to} \quad Fz \preceq v(\omega) \ , \tag{6.5}$$

can be recast as the parametric program $\mathcal{P}(x(\omega))$ by setting $x(\omega) = v(\omega) + FS^{-1}u(\omega)$ and $A = FR^{-1}$ in (6.1), where $S = R^T R$ is the Cholesky factorization of S.

6.2 *Geometry of Euclidian Projections*

Let us start by recalling some useful geometrical facts about Euclidian projections on convex polyhedral sets (Rockafellar and Wets, 1998, Example 6.16, Theorems 6.9 and 6.46, Proposition 6.17). Figure 6.1 provides a visual support to the following definitions.

6.3 Definition. *Let $C \subset \mathbb{R}^m$ be a closed set. The Euclidian projection mapping on C is the set-valued mapping $P_C : \mathbb{R}^m \rightrightarrows \mathbb{R}^m$ with values*

$$P_C(y) = \{\bar{y} \in C : ||\bar{y} - y|| \leq ||y' - y|| \ \textit{for every} \ y' \in C\} \ .$$

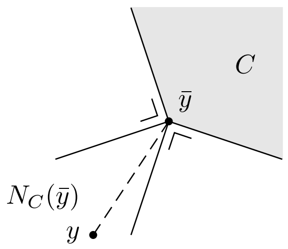

Fig. 6.1: $\bar{y}$ is the projection of y on C if $y - \bar{y}$ is in the normal cone to C at $\bar{y}$.

When C is a nonempty closed convex set, P_C is single-valued, in the sense that $P_C(y)$ is a singleton.

6.4 Definition. *Let $C \subset \mathbb{R}^m$ be a closed set. The* **proximal normals** *to C at $\bar{y} \in \mathbb{R}^m$ are the vectors $d \in \mathbb{R}^m$ such that $\bar{y} \in P_C(\bar{y} + \tau d)$ for some $\tau > 0$.*

6.5 Definition. *Let $C \subset \mathbb{R}^m$ be a closed convex set and $\bar{y} \in C$. A vector d is* **normal** *to C at $\bar{y}$ if $\langle d, y' - \bar{y} \rangle \leq 0$ for every $y' \in C$. The* **normal cone** *to C at $\bar{y}$ is the set $N_C(\bar{y}) = \{d \in \mathbb{R}^m : \langle d, y' - \bar{y} \rangle \leq 0 \text{ for every } y' \in C\}$ if $\bar{y} \in C$, or $N_C(\bar{y}) = \varnothing$ if $\bar{y} \notin C$.*

The normal cone to a closed convex set C at $\bar{y} \in C$ always contains 0. If $\bar{y}$ is in the interior of C, the normal cone is reduced to $\{0\}$. A more general definition for the normal cone, valid for an abstract set C, is also available, but it is not needed in the sequel.

The normal cone to a convex polyhedral set has a particular expression, given by the following proposition.

6.6 Proposition. *Let $C = \{y \in \mathbb{R}^m : Ay \preceq b\}$, where A is a matrix with rows a_i^T. For $\bar{y} \in C$, let $I(\bar{y}) = \{i : a_i^T \bar{y} = b_i\}$ denote the set of active constraints at $\bar{y}$. Then the normal cone to C at $\bar{y}$ is given by*

$$N_C(\bar{y}) = \{d = A^T \lambda : \lambda_i \geq 0 \text{ for } i \in I(\bar{y}), \lambda_i = 0 \text{ for } i \notin I(\bar{y})\} \ .$$

The relation between the normal cone and the Euclidian projection mapping is given in the following proposition, only valid for closed convex sets.

6.7 Proposition. *For a closed convex set $C \subset \mathbb{R}^m$, every normal vector is a proximal normal vector: $d \in N_C(\bar{y})$ iff $\bar{y} \in P_C(\bar{y} + d)$, where in fact $\bar{y} = P_C(\bar{y} + d)$.*

From Proposition 6.7, one deduces that every point $\bar{y}$ of $C = \{y \in \mathbb{R}^m : Ay \preceq b\}$ defines an equivalence class of points

$$\begin{aligned} [\bar{y}] &= \{y \in \mathbb{R}^m : P_C(y) = \bar{y}\} \\ &= \{\bar{y} + A^T \lambda : \lambda_i \geq 0 \text{ for } i \in I(\bar{y}), \lambda_i = 0 \text{ for } i \notin I(\bar{y})\} \end{aligned}$$

with $[\bar{y}]$ reduced to the singleton $\{\bar{y}\}$ when $\bar{y}$ is in the relative interior of C — the relative interior of a nonempty convex set C corresponds to the interior of C when C is viewed as a subset of the smallest linear space containing C (the affine hull of C).

Remark 6.1. Let $C = \{y \in \mathbb{R}^m : Ay \preceq b\}$, let y be some point in $\mathbb{R}^m$, and let $\bar{y} = P_C(y)$ be the projection of y on C. Let $I(\bar{y}) = \{i : a_i^T \bar{y} = b_i\}$ denote the index set of active constraints at $\bar{y}$. If $I(\bar{y})$ were known in advance for any y, then one could compute $P_C(y)$ as the projection of y on the linear space C_I defined with the set of active constraints $I = I(P_C(y))$ by

$$C_I = \{y \in \mathbb{R}^m : a_i^T y = b_i, i \in I\} = \{y \in \mathbb{R}^m : A_I y = b_I\} \ .$$

In that hypothetical situation, a closed-form formula is available for the projection. For instance, assuming for simplicity that the rows of A_I are linearly independent (Dontchev and Rockafellar, 2009, Exercise 2D.10), one has

$$P_{C_I}(y) = y - A_I^T (A_I A_I^T)^{-1}(A_I y - b_I) \ . \tag{6.6}$$

For some particular sets C (for example, hyperrectangles), it holds that $I = I(P_C(y))$ is equal to the index set of active or violated constraints at y. But for an arbitrary polyhedral set C and point y, $I(P_C(y))$ is difficult to guess, and does not usually coincide with the index set of active or violated constraints at y. □

Hoffman's lemma (Hoffman, 1952), stated next, shows that the Euclidian distance $d(y, C) = ||y - P_C(y)||$ from any point y to a polyhedral set C can be related to a measure that does not depend on $P_C(y)$.

6.8 Lemma (Hoffman's Lemma). *Let $C = \{y \in \mathbb{R}^m : Ay \preceq b\}$ be nonempty with $A \in \mathbb{R}^{s \times m}$ a nonzero matrix. For any $y \in \mathbb{R}^m$, there exists a scalar $\kappa(A) > 0$ depending on A such that $d(y, C) \leq \kappa(A) \ || \ [Ay - b]_+ \ ||$.*

Estimating $\kappa(A)$ and its sensitivity with respect to perturbations of A is an important subject of study — useful references are collected in Facchinei and Pang (2003, Notes to Chapter 3, page 332).

A well-known corollary of Hoffman's lemma is stated in the next proposition (Proposition 6.10). Let us first define the Hausdorff "distance" between two sets (Dontchev and Rockafellar, 2009, page 138).

6.9 Definition. *The* ***excess*** *of $C_0 \subset \mathbb{R}^m$ beyond $C_1 \subset \mathbb{R}^m$ is the quantity*

$$e(C_0, C_1) = \sup\nolimits_{y \in C_0} d(y, C_1) \ ,$$

with $e(\varnothing, C_1) = 0$ if $C_1 \neq \varnothing$ and $e(\varnothing, \varnothing) = \infty$. Equivalently,

$$e(C_0, C_1) = \inf\{\rho \geq 0 : C_0 \subset C_1 + \rho\mathbb{B}\} \ .$$

The ***Pompeiu-Hausdorff "distance"*** *between C_0 and C_1 can then be defined as the quantity*

$$d_h(C_0, C_1) = \max\{e(C_0, C_1), e(C_1, C_0)\} \ .$$

6.10 Proposition. *Let $\mathcal{C}(b) = \{y \in \mathbb{R}^m : Ay \preceq b\}$. Let b_0 and b_1 be two vectors such that $\mathcal{C}(b_0)$ and $\mathcal{C}(b_1)$ are nonempty. Then $d_h(\mathcal{C}(b_0), \mathcal{C}(b_1)) \leq \kappa(A)||b_0 - b_1||$.*

Proof. For $y \in \mathcal{C}(b_0)$, we have $Ay \preceq b_0$ and thus $Ay - b_1 \preceq b_0 - b_1$; in particular $[Ay - b_1]_+ \preceq [b_0 - b_1]_+$. Hence $||[Ay - b_1]_+|| \leq ||[b_0 - b_1]_+|| \leq ||b_0 - b_1||$. By Hoffman's lemma applied to $\mathcal{C}(b_1)$, we have $d(y, \mathcal{C}(b_1)) \leq \kappa(A)||[Ay - b_1]_+|| \leq \kappa(A)||b_0 - b_1||$. As $y \in \mathcal{C}(b_0)$ is arbitrary, $\sup_{y \in \mathcal{C}(b_0)} d(y, \mathcal{C}(b_1)) \leq \kappa(A)||b_0 - b_1||$. Similarly, $\sup_{y \in \mathcal{C}(b_1)} d(y, \mathcal{C}(b_0)) \leq \kappa(A)||b_1 - b_0||$, and the result follows. □

Remark 6.2. Observe that having A constant is essential in Proposition 6.10. For instance, consider the set-valued mapping $\mathcal{C}' : \mathbb{R} \rightrightarrows \mathbb{R}^2$ with values

$$\mathcal{C}'(\epsilon) = \{(x,t) \in \mathbb{R}^2 : t \geq (1-\epsilon)|x|\} = \{y \in \mathbb{R}^2 : A(\epsilon)y \preceq 0\}$$

with $A(\epsilon) = \begin{bmatrix} (1-\epsilon) & -1 \\ -(1-\epsilon) & -1 \end{bmatrix}$. For each $\eta \geq 0$, the point $(\eta, \eta(1-\epsilon)) \in \mathcal{C}'(\epsilon)$ is at distance $\epsilon\eta/\sqrt{2}$ from the set $\mathcal{C}'(0)$, so that by definition of the Pompeiu-Hausdorff distance, $d_h(\mathcal{C}'(0), \mathcal{C}'(\epsilon)) = \infty$ for any $\epsilon > 0$, whereas the matrices $A(0)$ and $A(\epsilon)$ could be made arbitrary "close" by choosing $\epsilon > 0$ small enough. □

Now, coming back to the parametric program (6.1), we observe that Hoffman's lemma allows to prove that if x in (6.1) follows a distribution having a compact support, then the projection of the origin $0 \in \mathbb{R}^m$ on the random polyhedral set $\mathcal{C}(x)$ defined by (6.2) lies in a bounded set.

6.11 Proposition. *Let $\mathcal{C} : \mathbb{R}^s \rightrightarrows \mathbb{R}^m$ be the set-valued mapping with values $\mathcal{C}(x)$ defined by* (6.2). *If x follows a probability distribution with compact support, then there exists a finite $\bar{\kappa} > 0$ such that the projection $y^*(\omega)$ of the origin on the polyhedral set $\mathcal{C}(x(\omega))$ satisfies $||y^*(\omega)|| \leq \bar{\kappa}$ for all possible realizations $x(\omega)$ of x.*

Proof. We assume that $x(\omega) \in X \cap \operatorname{dom} \mathcal{C}$, where X is a bounded subset of $\mathbb{R}^s$. We must show that the minimizer $y^*(\omega)$ of $\mathcal{P}(x(\omega))$ lies in a bounded subset Y of $\mathbb{R}^m$. But actually, if $x(\omega) \in \rho\mathbb{B}$ for some constant $\rho > 0$, and if $x(\omega) \in \operatorname{dom}\mathcal{C}$, where $\mathcal{C}(x(\omega)) = \{y \in \mathbb{R}^m : Ay \preceq x(\omega)\}$, then by Hoffman's lemma it holds that $||y^*(\omega)|| = d(0, \mathcal{C}(x(\omega))) \leq \kappa(A)|| \, [-x(\omega)]_+ \, || \leq \kappa(A)||x(\omega)|| \leq \kappa(A)\rho$, where $\kappa(A)$ is a constant depending on A, so that $y^*(\omega)$ lies in the ball $Y = \kappa(A)\rho\mathbb{B}$. We set $\bar{\kappa} = \kappa(A)\rho$. □

Proposition 6.11 shows that if one wants to try to predict from $x(\omega)$ a "nearly feasible" optimal solution $y^*(\omega)$, with x drawn from a distribution with compact support, then one could legitimately select a hypothesis space $\mathcal{H}$ of bounded functions.

6.3 Properties of Optimal Solutions

In this section, we establish a list of properties of optimal solutions to the parametric program (6.1). The results that are not directly used in the subsequent sections are marked by a star ($\star$). The results converted to an algorithm in the sequel are Propositions 6.19 and 6.22.

We will first note the following simple characterization of the domain of the set-valued mapping $\mathcal{C}$ defined by (6.2):

6.12 Proposition. *The domain of the set-valued mapping $\mathcal{C} : \mathbb{R}^s \rightrightarrows \mathbb{R}^m$ with values $\mathcal{C}(x) = \{y \in \mathbb{R}^m : Ay \preceq x\}$ is the closed convex cone* $\operatorname{dom}\mathcal{C} = \operatorname{range}(A) + \mathbb{R}^s_+$.

Proof. The set $\operatorname{dom}\mathcal{C}$ is the projection of the set $\{(x,y) \in \mathbb{R}^s \times \mathbb{R}^m : Ay \preceq x\}$ on $\mathbb{R}^s$ (first s components) and is thus closed as the projection of a closed set (the inequality constraints defining the set in $\mathbb{R}^{s+m}$ are non-strict). The set $\operatorname{dom}\mathcal{C}$ is convex since $x_0, x_1 \in \operatorname{dom}\mathcal{C}$ means that $Ay_0 \preceq x_0$, $Ay_1 \preceq x_1$ for some $y_0, y_1 \in \mathbb{R}^m$, implying the existence of $y_t = (1-t)y_0 + ty_1$ satisfying $Ay_t \preceq (1-t)x_0 + tx_1$ for $0 \le t \le 1$. Furthermore, $\operatorname{dom}\mathcal{C}$ is a cone since $Ay \preceq x$ entails $A(ty) \preceq tx$ for $t \ge 0$, so that $x \in \operatorname{dom}\mathcal{C}$ entails $tx \in \operatorname{dom}\mathcal{C}$ for $t \ge 0$. Now, the constraints defining the set $\mathcal{C}(x)$ are equivalent to $x = Ay + \xi$, $\xi \succeq 0$, where $\{v \in \mathbb{R}^s : v = Ay,\ y \in \mathbb{R}^m\}$ is by definition the range of $A \in \mathbb{R}^{s\times m}$. □

A possible way to draw random points $x(\omega) \in \operatorname{dom}\mathcal{C}$ is thus to draw a linear combination of vectors forming an orthonormal basis for A, and then add to the resulting vector a random vector with nonnegative components.

Although we do not directly invoke it in the sequel, for completeness we also recall the following structural property:

6.13 Proposition($\star$). *The function $g(x) = \inf_{y\in\mathcal{C}(x)} \frac{1}{2}||y||^2$ is convex in x.*

For the notion of extended-real-valued function used in the following proof, see Appendix A.1.

Proof. The program $\mathcal{P}(x)$ amounts to the minimization of the extended-real-valued function $\bar{f}$ defined by $\bar{f}(x,y) = ||y||^2/2$ if $Ay \preceq x$, and $\bar{f}(x,y) = \infty$ otherwise. We check that $\bar{f}(x,y)$ is jointly convex in x, y. Let us write $x_t = (1-t)x_0 + tx_1$ and $y_t = (1-t)y_0 + ty_1$ for $0 < t < 1$. If $\bar{f}(x_0,y_0)$ and $\bar{f}(x_1,y_1)$ are finite, implying $Ay_0 \preceq x_0$, $Ay_1 \preceq x_1$, then $\bar{f}(x_t,y_t)$ is also finite, since $Ay_t \preceq x_t$ and $\bar{f}(x_t,y_t) = ||y^t||^2/2 \le (1-t)||y_0||^2/2 + t||y_1||^2/2 = (1-t)\bar{f}(x_0,y_0) + t\bar{f}(x_1,y_1)$. If $\bar{f}(x_0,y_0) = \infty$ or $\bar{f}(x_1,y_1) = \infty$, the convexity inequality $\bar{f}(x_t,y_t) \le (1-t)\bar{f}(x_0,y_0) + t\bar{f}(x_1,y_1) = \infty$ for $0 < t < 1$ is trivially verified. Hence $\bar{f}(x,y)$ is convex in (x,y) (Rockafellar, 1970, Theorem 4.1). As a convex set, the epigraph of $\bar{f}$ defined by $\operatorname{epi}\bar{f} = \{(x,y,\alpha) \in (\mathbb{R}^s \times \mathbb{R}^m) \times \mathbb{R} : \alpha \ge \bar{f}(x,y)\}$ has its projection on its component $\mathbb{R}^s \times \mathbb{R}$ convex as well. The function $g(x) = \inf_y \bar{f}(x,y)$ whose epigraph is $\operatorname{epi} g = \{(x,\alpha) \in \mathbb{R}^s \times \mathbb{R} : (x,y,\alpha) \in \operatorname{epi}\bar{f}$ for some $y\}$ is thus convex. □

Now, for $x(\omega) \in \operatorname{dom}\mathcal{C}$, the program $\mathcal{P}(x(\omega))$ has a single minimizer $y^*(\omega)$ corresponding to the projection of $0 \in \mathbb{R}^m$ on the convex polyhedral set $\mathcal{C}(x(\omega))$. By Proposition 6.7, setting $C = \mathcal{C}(x(\omega))$, a point $y \in \mathbb{R}^m$ is thus optimal if the vector $0 - y = -y$ lies in the normal cone to C at y, that is, $-y \in N_C(y)$, or equivalently $y + N_C(y) \ni 0$.

Given the optimal solution $y^*(\omega)$ to $\mathcal{P}(x(\omega))$, it is easy to describe sets of nearly optimal solutions, called ϵ-optimal solutions (see Appendix A, Section A.4). To this end, let us recall the notion of tangent cone to an arbitrary set C (Rockafellar and Wets, 1998, Definition 6.1, Theorem 6.9).

6.14 Definition($\star$). *A vector d is tangent to C at $\bar{y} \in C$ if for some sequence $\{y^\nu\}_{\nu\in\mathbb{N}}$ of points $y^\nu \in C$ converging to $\bar{y}$, and some sequence $\{\tau^\nu\}_{\nu\in\mathbb{N}}$ of scalars τ^ν converging to 0 with $0 < \tau^{\nu+1} < \tau^\nu$, one has*

$$(y^\nu - \bar{y})/\tau^\nu \to d \ .$$

The set of all such vectors d is a closed cone, possibly reduced to the singleton $\{0\}$, called the **tangent cone** *to C at $\bar{y}$, and written $T_C(\bar{y})$. In the particular case where C is a convex subset of $\mathbb{R}^m$, the tangent cone to C at $\bar{y}$ is a convex set given by*

$$T_C(\bar{y}) = \mathrm{cl}\{d \in \mathbb{R}^m : \bar{y} + \lambda d \in C \textit{ for some } \lambda > 0\} \ .$$

For a polyhedral set $C = \{y \in \mathbb{R}^m : Ay \preceq x\}$, the tangent cone to C at $\bar{y}$ is given by

$$T_C(\bar{y}) = \{d \in \mathbb{R}^m : a_i^T d \leq 0 \text{ for all } i \in I(\bar{y})\}$$

(Rockafellar and Wets, 1998, Theorem 6.46).

The next proposition describes properties of the sets of ϵ-optimal solutions, denoted by $S_\epsilon(\omega)$ for a given ϵ and a given realization of ω.

6.15 Proposition($\star$). *The sets of ϵ-optimal solutions $S_\epsilon(\omega)$ to $\mathcal{P}(x(\omega))$ satisfy two properties, expressed with respect to the exact optimal solution $y^*(\omega)$ and the set $\mathcal{C}(x(\omega))$:*

i. $S_\epsilon(\omega) \cap ||y^*(\omega)||\mathbb{B} = \{y^*(\omega)\}$ *for all* $\epsilon > 0$*;*

ii. There exists an $\epsilon_0 > 0$ such that for every $\epsilon \in [0, \epsilon_0]$,

$$S_\epsilon(\omega) = \rho\mathbb{B} \cap [y^*(\omega) + T_{\mathcal{C}(x(\omega))}(y^*(\omega))] \ ,$$

where $\rho = \sqrt{||y^*(\omega)||^2 + 2\epsilon}$ and $T_{\mathcal{C}(x(\omega))}(y^*(\omega)) = \{d \in \mathbb{R}^m : a_i^T d \leq 0, i \in I(y^*(\omega))\}$.

The following proof relies on standard arguments — see for instance Dontchev and Rockafellar (2009, Theorem 2E.3).

Proof. To lighten the notation, we write S_ϵ for $S_\epsilon(\omega)$, C for $\mathcal{C}(x(\omega))$, and y^* for $y^*(\omega)$. The set $S_\epsilon = \epsilon\text{-}\operatorname{argmin}_{y \in C} f(y)$ is given by

$$\begin{aligned} S_\epsilon &= \{y \in C : f(y) \leq f(y^*) + \epsilon\} = \{y \in C : ||y||^2 \leq ||y^*||^2 + 2\epsilon\} \\ &= C \cap (\sqrt{||y^*||^2 + 2\epsilon})\mathbb{B}. \end{aligned}$$

There is no feasible vector y with $||y|| < ||y^*||$, whereas $||y|| = ||y^*||$ entails $y = y^*$ by the strict convexity of $f(y)$, hence the first part of the proposition. On the other hand, the feasibility set C is described by a finite number of constraints, so that in a sufficiently small neighborhood of y^*, say R_0, there is no new constraint that becomes active: $I(y) \subset I(y^*)$ for $y \in R_0 \cap C$. As the constraints are linear, C can be approximated locally by the set $C_{y^*} = \{y \in \mathbb{R}^m : a_i^T y \leq b_i, i \in I(y^*)\}$. Since $a_i^T y_i^* = b_i$ for $i \in I(y^*)$, we have $C_{y^*} = \{y^* + d : a_i^T d \leq 0, i \in I(y^*)\} = y^* + T_C(y^*)$. □

Remark 6.3. Having the set C polyhedral is important in the proof of Proposition 6.15. If the set C were not polyhedral (it can still be convex), there would not necessarily exist a neighborhood R_0 of y^* in which a proper inclusion of $C \cap R_0$ in $[y^* + T_C(y^*)] \cap R_0$ can be precluded. The local approximation at y^* of the set C by the set $y^* + T_C(y^*)$ could thus include infeasible points. For example, for $C = \{(x,t) \in \mathbb{R}^2 : t \geq |x| + x^2\}$, one has $T_C(0) = \{(x,t) \in \mathbb{R}^2 : t \geq |x|\}$, and consequently $(C \setminus T_C(0)) \cap \epsilon\mathbb{B} \neq \varnothing$ for any $\epsilon > 0$. □

The following proposition relies on duality theory (Rockafellar and Wets, 1998, Chapter 11).

6.16 Proposition($\star$). *The dual of $\mathcal{P}(x(\omega))$ corresponds, after a sign change, to the program*

$$\begin{aligned}\mathcal{D}(x(\omega)): \quad & \textit{minimize} \quad && -g(\lambda) = \tfrac{1}{2}\lambda^T(AA^T)\lambda + x(\omega)^T\lambda \\ & \textit{subject to} && \lambda \succeq 0 \ .\end{aligned}$$

Proof. The Lagrangian for $\mathcal{P}(x(\omega))$ is $L(y,\lambda) = \frac{1}{2}||y||^2 + \lambda^T(Ay - x(\omega))$, with $\lambda \succeq 0$. The infimum of $L(y,\lambda)$ over y is attained at $\bar{y} = -A^T\lambda$. Hence the dual function $g(\cdot) = \inf_y L(y,\cdot)$ has values $g(\lambda) = -\frac{1}{2}\lambda^T AA^T\lambda - x(\omega)^T\lambda$. The dual formulation is obtained by maximizing $g(\lambda)$ subject to $\lambda \succeq 0$. $\square$

Given $y^*(\omega)$, it is often possible to obtain a solution to the dual problem, as shown by the following proposition. Note that from now on, when ω or $x(\omega)$ is clear from the context, we freely write C for $\mathcal{C}(x(\omega))$, and y^* for the optimal solution $y^*(\omega)$ to $\mathcal{P}(x(\omega))$. We will also freely write x for its realization $x(\omega)$.

6.17 Proposition($\star$). *If y^* is optimal for $\mathcal{P}(x)$, any optimal solution $\lambda^* \in \mathbb{R}^s$ for the dual $\mathcal{D}(x)$ is determined by a subvector $\lambda_I \in \mathbb{R}^p$ of possibly nonzero elements λ_i^*, $i \in I(y^*)$, $p = |I(y^*)|$, such that λ_I is a nonnegative solution to $A_I^T\lambda_I = -y^*$.*

Proof. Having $y^* \in C$ optimal means $-y^* \in N_C(y^*)$, that is, there exists at least one vector $\lambda \in \mathbb{R}^s$ such that

$$y^* + \textstyle\sum_{i=1}^s \lambda_i a_i = 0, \quad \lambda \succeq 0, \quad \lambda_i = 0 \text{ if } i \notin I(y^*) \ ,$$

where $I(y^*) = \{i : a_i^T y^* = x_i\}$ is the index set of active constraints at $y^* \in C$. These conditions are nothing else but the usual Karush-Kuhn-Tucker optimality conditions

$$\nabla f(y^*) + A^T\lambda = 0, \quad Ay^* \preceq x, \quad \lambda \succeq 0, \quad \lambda_i(a_i^T y^* - x_i) \geq 0$$

with multipliers λ_i optimal for the dual problem. Let $A_I \in \mathbb{R}^{p\times m}$ be the submatrix of A with rows a_i^T, $i \in I(y^*)$, $p = |I(y^*)|$, so that the subvector $\lambda_I \in \mathbb{R}^p$ of λ stacking the possibly nonzero elements λ_i, $i \in I(y^*)$, has to satisfy $y^* + A_I^T\lambda_I = 0$. If the rows of A_I are linearly independent (a constraint qualification which always holds for $p = 1$ and never holds for $p > m$), then

$$\lambda_I = -(A_I A_I^T)^{-1}A_I y^* = -(A_I A_I^T)^{-1}x_I$$

where $x_I \in \mathbb{R}^p$ is the subvector of x stacking the elements x_i, $i \in I(y^*)$. We can assume that the solution λ_I is nonnegative inasmuch as y^* is optimal. Now if $p > m$ and the columns of A_I are linearly independent, the equation $y^* + A_I^T\lambda_I = 0$ is underdetermined and admits the particular solution $v_0 = -A_I(A_I^T A_I)^{-1}y^*$ (least-norm solution). If $\ker(A_I^T) = \{v : A_I^T v = 0\}$ denotes the null space of A_I^T, then

$$\lambda_I \in \{v_0 + v : v \in \ker(A_I^T)\} \cap \mathbb{R}^p_+ \ .$$

Recall that the null space of A_I^T can be described as the span of the eigenvectors associated to the zero eigenvalues of $(A_I^T A_I)$. $\square$

Uniquely determined multipliers do not always exist: this is consistent with the observation that the dual problem can have a continuum of optimal solutions if the matrix (AA^T) in Proposition 6.16 is only positive semi-definite.

Remark 6.4. A property of the objective function f that facilitated the developments in Proposition 6.17 is the expression of its gradient $\nabla f(y) = y$. The solution to the inversion of the generalized equation $u \in \partial f(y)$, where $\partial f(y)$ is the subgradient of f evaluated at y, is then simply $y = u$. We recall that in general, when f is a proper lower-semicontinuous convex function, $u \in \partial f(y)$ if and only if $y \in \partial f^*(u)$ with $f^*(u) = \sup_y \{u^T y - f(y)\}$ (Rockafellar and Wets, 1998, Proposition 11.3). $\square$

It is possible to extract information from the index set of active constraints at an optimal solution y^*, as shown by the following proposition.

6.18 Proposition($\star$). *Let $S(x) = \{y^* \in \mathbb{R}^m : Ay^* \preceq x,\ ||y^*|| \leq ||y||$ whenever $Ay \preceq x\}$ denote the set of optimal solutions for $\mathcal{P}(x)$ (the set is a singleton, assuming $x \in \operatorname{dom} \mathcal{C}$). Let $S^{-1}(y^*) = \{x \in \mathbb{R}^s : y^* \in S(x)\}$ denote the set of parameter vectors x such that y^* is optimal for $\mathcal{P}(x)$. Then, it holds that*

$$\begin{aligned} S^{-1}(y^*) \supset & \{x \in \mathbb{R}^s : x_i = a_i^T y^* \text{ for } i \in I(y^*), x_i \in [a_i^T y^*, \infty) \text{ for } i \notin I(y^*)\} \\ &= Ay^* + N_1(I(y^*)) \times \cdots \times N_s(I(y^*)) \end{aligned}$$

where $I(y^*) = \{i : a_i^T y^* = x_i\}$ is the set of active constraints at y^*, and where we define $N_i(I) = \{0\}$ if $i \in I$ and $N_i(I) = [0, \infty)$ if $i \notin I$. The inclusion can be refined by considering, instead of I, the index set $I^+ = \{i : \lambda_i > 0\} \subset I(y^*)$ of the positive KKT multipliers associated to the active constraints at y^*.

Proof. Let $b_i = a_i^T y^*$, $1 \leq i \leq s$. By definition of $I(y^*)$, we have $b_i = x_i$ if $i \in I(y^*)$ and $b_i < x_i$ if i is in the complement of $I(y^*)$, that is, $i \in J(y^*) = \{i : a_i^T y^* < x_i\} = J$. A constraint indexed by $i \in J$ remains inactive if x_i is in the open interval (b_i, ∞), and becomes active but does not alter the optimal solution y^* if $x_i = b_i$, whence the first part of the proposition. Now, relaxing the constraints to which are associated zero-valued KKT multipliers does not alter the optimal solution y^*, so that in fact

$$S^{-1}(y^*) = \{x \in \mathbb{R}^s : x_i = a_i^T y^* \text{ if } i \in I^+,\ x_i \in [a_i^T y^*, \infty) \text{ if } i \notin I^+\}$$

where $I^+ = \{i : \lambda_i > 0\}$ is the index set of active constraints with positive multipliers. $\square$

Example 6.1. Propositions 6.17 and 6.18 can be illustrated on a numerical example (Figure 6.2). Let A have 4 rows $a_1^T = [\ 0 \quad -1\ ]$, $a_2^T = [\ -1 \quad 1\ ]$, $a_3^T = [\ -1 \quad 0\ ]$, $a_4^T = [\ -1 \quad -2\ ]$. Let x have the value $x(\omega_1) = [\ -4 \quad 2 \quad -2 \quad 0\ ]^T$. The optimal solution to $\mathcal{P}(x(\omega_1))$ is $y^*(\omega_1) = [\ 2 \quad 4\ ]^T$. The set of active constraints is $I(y^*(\omega_1) = \{1, 2, 3\}$, meaning that 3 hyperplanes meet at $y^*(\omega_1)$. The matrix A_I has the 3 rows a_1^T, a_2^T, a_3^T. The optimality condition is $y^*(\omega_1) = -A_I^T \lambda_I$. The set

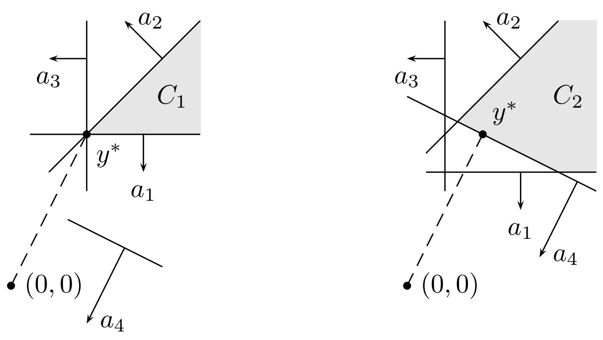

Fig. 6.2: Left: Pathological case $x(\omega_1)$ for which the dual $\mathcal{D}(x(\omega_1))$ has several optimal solutions described in the example (see text). Right: Case $x(\omega_2)$ where the dual problem $\mathcal{D}(x(\omega_2))$ has a single optimal solution. The primal problems $\mathcal{P}(x(\omega_1))$, $\mathcal{P}(x(\omega_2))$ have the same unique optimal solution $y^* = (2,4) \in \mathbb{R}^2$. The dashed line indicates the minimal distance between the origin and the set $C_i = \{y \in \mathbb{R}^2 : Ay \preceq x(\omega_i)\}$.

of solutions for λ_I is

$$\begin{aligned}\Lambda_I =& (-A_I(A_I^T A_I)^{-1} y^*(\omega_1) + \ker\{A_I^T\}) \cap \mathbb{R}^3_+ \\ &= ([\ \tfrac{10}{3}\ \ \tfrac{-2}{3}\ \ \tfrac{8}{3}\ ]^T + \mu[\ 1\ \ 1\ \ -1\ ]^T : \mu \in \mathbb{R}) \cap \mathbb{R}^3_+ \\ &= \{[\ 4\ \ 0\ \ 2\ ]^T + \mu[\ 1\ \ 1\ \ -1\ ]^T : \mu \in [0,2]\}\ .\end{aligned}$$

Note that a numerical solution algorithm applied to the dual problem could return any particular solution $\lambda \in \Lambda_I \times \{0\}$. The solutions corresponding to $\mu = 0$ and $\mu = 2$ are $\lambda = [\ 4\ \ 0\ \ 2\ \ 0\ ]^T$ and $\lambda = [\ 6\ \ 2\ \ 0\ \ 0\ ]^T$ respectively. The zero elements of the solutions indicate that $y^*(\omega_1)$ is still optimal when

$$x \in Ay^*(\omega_1) + \{0\} \times [0,\infty) \times \{0\} \times [0,\infty) \cup \{0\} \times \{0\} \times [0,\infty) \times [0,\infty)$$

with $Ay^*(\omega_1) = [\ -4\ \ 2\ \ -2\ \ -10\ ]^T$.

Now if x has the value $x(\omega_2) = x(\omega_1) + [\ 1\ \ 1\ \ 1\ \ -10\ ]^T$, the optimal solution to $\mathcal{P}(x(\omega_2))$ is $y^*(\omega_2) = [\ 2\ \ 4\ ]^T = y^*(\omega_1)$, showing that the inclusion concerning $S^{-1}(y^*)$ in Proposition 6.18 may be proper.

Given that $I(y^*(\omega_2)) = \{4\}$, and thus $A_I = a_4^T$, the solution to the optimality condition $y^*(\omega_2) + A_I^T \lambda_I = 0$ is uniquely determined by $\lambda_I = -(A_I A_I^T)^{-1} A_I y^* = 2 = \lambda_4$. Therefore, the dual $\mathcal{D}(x(\omega_2))$ admits the unique solution $\lambda = [\ 0\ \ 0\ \ 0\ \ 2\ ]^T$. The zero elements of the solution indicate that $y^*(\omega_2) = y^*(\omega_1)$ is still optimal when $x \in Ay^*(\omega_1) + [0,\infty) \times [0,\infty) \times [0,\infty) \times \{0\}$. □

Remark 6.5. Proposition 6.18 has formalized an invariance property with respect to a subset of translations of the input x, where the subset depends on the output y^*. In the perspective of using supervised learning to predict nearly feasible optimal solutions, invariance properties could be used as a means to obtain virtual samples (x^ν, y^ν) with $x^\nu \in S^{-1}(y^\nu)$, or can be embedded in learning algorithms to improve generalization abilities from prior knowledge (Decoste and Schölkopf, 2002). □

The next proposition shows that from a single pair $(\bar{x}, \bar{y}^*)$ with $\bar{y}^*$ optimal for $\mathcal{P}(\bar{x})$, it is sometimes possible to predict the optimal solution $y^*(\omega)$ for parameters $x(\omega)$ in a neighborhood of $\bar{x}$. The size of the neighborhood is estimated in the proof, and is related to the smallest singular value of the matrix A_I defined in Proposition 6.19.

6.19 Proposition. *Let $\bar{y}$ be the optimal solution to the program $\mathcal{P}(\bar{x})$. Let $A_I \in \mathbb{R}^{p\times m}$, $p = |I(\bar{y})|$, be the submatrix of A stacking the rows a_i^T of active constraints $i \in I(\bar{y})$, and for a vector $x \in \mathbb{R}^s$, let $x_I \in \mathbb{R}^p$ be the subvector of x stacking the elements x_i, $i \in I(\bar{y})$. If the rows of A_I are linearly independent and if $(A_I A_I^T)^{-1}\bar{x}_I \prec 0$, then there exists a neighborhood Q of $\bar{x}$ such that for all $x(\omega) \in Q \cap \operatorname{dom}\mathcal{C}$, the optimal solution to $\mathcal{P}(x(\omega))$ is given by $y^*(\omega) = A_I^T(A_I A_I^T)^{-1}x_I(\omega)$.*

Proof. First, we show that there exist a neighborhood Q_0 of $\bar{x}$ and a neighborhood R_0 of $\bar{y}$ such that $I(y) \subset I(\bar{y})$ whenever $x(\omega) \in Q_0 \cap \operatorname{dom}\mathcal{C}$ and $y \in R_0 \cap \mathcal{C}(x(\omega))$. Let J denote the set of inactive constraints at $\bar{y}$. For all $j \in J$, let d_j be the distance of $\bar{y}$ to the hyperplane $\{y : a_j^T y = \bar{x}_j\}$, namely, $d_j = ||a_j||^{-1}(\bar{x}_j - a_j^T\bar{y}) > 0$. Let $d_0 = \min\{d_j : j \in J\}$ and let us define

$$\eta_0 = \min\{||a_j||(d_j - d_0/2) : j \in J\} > 0 \ .$$

We choose $Q_0 = \{\bar{x} + \eta_0 u : ||u|| < 1\}$ and $R_0 = \{\bar{y} + (d_0/2)\, v : ||v|| < 1\}$. Then, the distance of $\bar{y}$ to any hyperplane $\{y : a_j^T y = x_j(\omega)\}$, $j \in J$, is greater than $d_0/2$ whenever $x(\omega) \in Q_0 \cap \operatorname{dom}\mathcal{C}$, and $y \in R_0 \cap \mathcal{C}(x(\omega))$ is separated from the hyperplanes $\{y : a_j^T y = x_j(\omega)\}$ for $j \in J$. Hence $j \notin I(y)$ and thus $I(y) \subset I(\bar{y})$ (no new active constraints).

Next, we claim that if the rows a_i^T for $i \in I(\bar{y})$ are linearly independent, and if $(A_I A_I^T)^{-1}\bar{x}_I \prec 0$, then there exists a neighborhood $Q \subset Q_0$ of $\bar{x}$ such that $I(y^*(\omega)) = I(\bar{y})$ whenever $x(\omega) \in Q \cap \operatorname{dom}\mathcal{C}$, where $y^*(\omega)$ denotes the optimal solution to $\mathcal{P}(x(\omega))$. It is sufficient to show that whenever $x(\omega) \in Q \cap \operatorname{dom}\mathcal{C}$, $y^*(\omega)$ lies in R_0, and any optimal $\lambda_i^*(\omega)$, $i \in I(\bar{y})$, associated to $y^*(\omega)$ is positive, as $\lambda_i^* > 0$ entails $i \in I(y^*(\omega))$. Since the rows of A_I are linearly independent, the vector

$$\bar{\lambda}_I = -(A_I A_I^T)^{-1}A_I\bar{y} = -(A_I A_I^T)^{-1}\bar{x}_I \succ 0 \quad (I = I(\bar{y}))$$

is the only vector of possibly nonzero multipliers associated to $\bar{y}$ (the reference optimal solution). Let us replace the dual problem $\mathcal{D}(x(\omega))$ by a problem on the reduced set of variables $\delta_I \in \mathbb{R}^p$ with $\lambda_I(\omega) = \bar{\lambda}_I + \delta_I$, $I = I(\bar{y})$, namely,

$$\begin{aligned} \text{minimize} \quad & -g_I(\delta_I) = \tfrac{1}{2}(\bar{\lambda}_I + \delta_I)^T(A_I A_I^T)(\bar{\lambda}_I + \delta_I) + x_I(\omega)^T(\bar{\lambda}_I + \delta_I) \\ \text{subject to} \quad & \bar{\lambda}_I + \delta_I \succeq 0 \ . \end{aligned}$$

If we relax the constraint $\delta_I \succeq -\bar{\lambda}_I$, and if we set $x(\omega) = \bar{x}(\omega) + \Delta x(\omega)$, the optimality condition for the resulting problem is $\nabla g_I(\delta_I^*) = 0$, and its optimal solution is

$$\delta_I^* = -(A_I A_I^T)^{-1}(A_I A_I^T\bar{\lambda}_I + x_I(\omega)) = -(A_I A_I^T)^{-1}\Delta x_I(\omega) \ ,$$

where we have used the fact that $\bar{\lambda}_I = -(A_I A_I^T)^{-1}\bar{x}_I$. Let us define

$$\epsilon = \min\{\bar{\lambda}_i : i \in I\} > 0 \ .$$

Since $\delta_i^* > -\bar{\lambda}_i$ for each $i \in I$ whenever $||\delta_I^*|| < \epsilon$, we can guarantee, using the inequality

$$||\delta_I^*|| \leq ||(A_I A_I^T)^{-1}|| \cdot ||\Delta x_I(\omega)|| \leq ||(A_I A_I^T)^{-1}|| \cdot ||\Delta x(\omega)||$$

that whenever $||x(\omega) - \bar{x}|| = ||\Delta x(\omega)|| < \eta_1$ with $\eta_1 = \min\{\eta_0,\ ||(A_I A_I^T)^{-1}||^{-1}\epsilon\}$, the solution δ_I^* satisfies the constraint of the initial reduced problem, and $x(\omega) \in Q_0$. Thus δ_I^* is also optimal for the reduced problem. We note that $||(A_I A_I^T)^{-1}||^{-1} = (\sigma_p(A_I))^2$, where $\sigma_p(A_I) > 0$ is the smallest singular value of A_I (A_I has rank $p = |I(\bar{y})|$). Reverting now to the full dual problem over $\lambda \in \mathbb{R}^m$, we see that the vector λ^* with $\lambda_i^* = \bar{\lambda}_i + \delta_i^* > 0$ if $i \in I(\bar{y})$, $\lambda_i^* = 0$ if $i \notin I(\bar{y})$, induces a vector

$$y = -\textstyle\sum_{i\in I} \lambda_i^* a_i = \bar{y} - \sum_{i \in I} \delta_i^* a_i = \bar{y} + A_I^T (A_I A_I^T)^{-1} \Delta x_I(\omega).$$

Using $||y - \bar{y}|| \leq ||A_I^T(A_I A_I^T)^{-1}|| \cdot ||\Delta x(\omega)||$, we have $||y - \bar{y}|| < d_0/2$ if $||\Delta x(\omega)|| < ||A_I^T(A_I A_I^T)^{-1}||^{-1} d_0/2$. In fact $||A_I^T(A_I A_I^T)^{-1}||^{-1} = \sigma_p(A_I)$. By setting

$$\eta = \min\{\eta_0,\ \sigma_p(A_I)\, d_0/2,\ (\sigma_p(A_I))^2\, \epsilon\}$$

and choosing for Q the open ball of radius η centered at $\bar{y}$, we can ensure that $y \in R_0$, so that $I(\bar{y})$ is the set of constraints active at y. This means that the vector y is optimal for the primal problem, and that λ^* is optimal for the dual problem.

Now, given the existence of a neighborhood Q of $\bar{x}$ for which $I(y^*(\omega)) = I(\bar{y})$ when $x(\omega) \in Q \cap \operatorname{dom} \mathcal{C}$, $y^*(\omega)$ can be obtained as the projection of the origin on the linear subspace $\{y \in \mathbb{R}^m : a_i^T y = x_i(\omega), i \in I(\bar{y})\}$ whenever $x(\omega) \in Q \cap \operatorname{dom} \mathcal{C}$. With the rows of A_I linearly independent, the projection is given by $y^*(\omega) = A_I^T(A_I A_I^T)^{-1} x_I(\omega)$. □

In the context of the supervised learning of nearly feasible optimal solutions, where one looks for a hypothesis h in a hypothesis space $\mathcal{H}$ of mappings from $x(\omega)$ to $y^*(\omega)$, the knowledge of a local model for $y^*(\omega)$ in a neighborhood of $\bar{x}$, for instance a first-order approximation $y^*(\omega) \simeq \bar{y} + D(x(\omega) - \bar{x})$, means that one could learn h not only by penalizing the discrepancies between the sampled targets $y(\omega)$ and the predictions $h(x(\omega))$, but also by penalizing the discrepancy between the gradient of h at $\bar{x}$ and the gradient D of the local model known a priori. Such ideas have been developed by Simard et al. (1998). We also note that it is technically possible to incorporate derivative information in Gaussian Process regression (Solak et al., 2003).

Another possibility would be to learn classifiers for the events $i \in I(\bar{y})$, $1 \leq i \leq s$, since we know that the information on active constraints can be generalized locally around $\bar{x}$, and followed by a straightforward computation of $y^*(\omega)$.

Remark 6.6. A typical situation where the assumptions of Proposition 6.19 fail is the case where two inequality constraints form an equality constraint: $a_1^T y \leq x_1$, $a_2^T y \leq x_2$ with $a_2 = -a_1$ and $x_2 = -x_1$. In that case, a solution $\bar{y}$ has to satisfy $a_1^T \bar{y} = x_1$, and A_I is always rank-deficient. In the event where the two parallel hyperplanes are separated, it is not easy to predict which side of the so-induced slab region the optimal solution will follow. If q pairs of hyperplanes are merged, there might exist 2^q distinct configurations of active constraints in the neighborhood of $\bar{x}$, provided that the assumptions of Proposition 6.19 hold with one element of each pair of equality-forming hyperplanes removed from the index set I of active constraints at $\bar{y}$. □

Now, an important question is whether a local model shared by a finite collection of points can be generalized to the convex hull of the points.

6.20 Lemma. *Given $x(0), x(1) \in \operatorname{dom}\mathcal{C}$, let $x(t) = (1-t)x(0) + tx(1)$ for $0 \le t \le 1$. Let $y^*(t)$ denote the optimal solution to $\mathcal{P}(x(t))$. If $I = I(y^*(0)) = I(y^*(1))$, then $y^*(t) = (1-t)y^*(0) + ty^*(1)$. If in addition the rows a_i^T, $i \in I$, are linearly independent, then $y^*(t) = A_I^T(A_I A_I^T)^{-1}x_I(t)$.*

Proof. We consider the points $y(t) = (1-t)y^*(0) + ty^*(1)$, $0 \le t \le 1$, in correspondence with $x(t) = (1-t)x(0) + tx(1)$. Let $j(t)$ represent the constraint $a_j^T y \le x(t)$, $1 \le j \le s$, $0 \le t \le 1$. We have $a_j^T y(0) - x_j(0) < 0$ for each $j(0) \notin I$, and $a_j^T y(1) - x_j(1) < 0$ for each $j(1) \notin I$, by definition of I for $y(0) = y^*(0)$ and $y(1) = y^*(1)$. Hence $(1-t)(a_j^T y(0) - x_j(0)) + t(a_j^T y(1) - x_j(1)) = a_j^T y(t) - x_j(t) < 0$ whenever $j \notin I$, meaning that $y(t)$ is feasible with respect to $j(t)$ with $j(t) \notin I(y(t))$. Similarly, for each $i \in I$, it holds that $a_i^T y(0) = x_i(0)$ and $a_i^T y(1) = x_i(1)$. Hence, $a_i^T y(t) = x_i(t)$, meaning that $y(t)$ is feasible with respect to $i(t)$ with $i(t) \in I(y(t))$. We have thus shown that $y(t)$ is feasible and that $I(y(t)) = I(y_0^*) = I$. Now, let $\lambda(t) = (1-t)\lambda(0) + t\lambda(1)$, where $\lambda_j(0) = \lambda_j(1) = 0$ for $j \notin I$, and where $\lambda_I(0) \succeq 0$ is a solution to $y^*(0) + A_I^T \lambda_I(0) = 0$, and $\lambda_I(1) \succeq 0$ is a solution to $y^*(1) + A_I^T \lambda(1) = 0$. The equality $(1-t)(y^*(0) + A_I^T\lambda(0)) + t(y^*(0) + A_I^T\lambda(1)) = y(t) + A_I^T \lambda_t = 0$ with $\lambda_t \succeq 0$ shows that $y(t)$ satisfies the optimality conditions for $\mathcal{P}(x(t))$. Therefore, $y(t)$ is the projection of 0 on the active constraints, and if the rows of A_I are linearly independent, $y(t) = A_I^T(A_I A_I^T)^{-1}x(t)$ for $0 \le t \le 1$. □

As the convex hull of a collection of points $\{x^\nu\}$, written $\operatorname{conv}(\{x^\nu\})$, contains the line segments between any two of its points, we have:

6.21 Proposition (Inner generalization). *Let $\{x^\nu\}$ be a collection of points in $\operatorname{dom}\mathcal{C}$ with a common set I of active constraints at the optimal solution to $\mathcal{P}(x^\nu)$. If the rows of A_I are linearly independent, then $y^*(\omega) = A_I^T(A_I A_I^T)^{-1}x_I(\omega)$ whenever $x(\omega) \in \operatorname{conv}(\{x^\nu\})$.*

Another interesting question is whether we can, from a single point $(\bar{x}, \bar{y})$ equipped with a local model, infer the domain of validity of the model.

6.22 Proposition (Outer generalization). *Let $\bar{y}$ be the optimal solution to the program $\mathcal{P}(\bar{x})$. Let $I(\bar{y})$, written I for short, be the index set of active constraints at $\bar{y}$, and let J be its complement. Let A_I be the submatrix of active rows of A. If the rows of A_I are linearly independent, then the subset of $\operatorname{dom}\mathcal{C}$ (values for the parameter x) where the index set of active constraints at the optimal solution y^* of the program $\mathcal{P}(x)$ coincides with $I = I(\bar{y})$ can be described as the polyhedral cone*

$$X(I) = \{x \in \mathbb{R}^s : B_I x_I \preceq 0,\ D_I x_I - x_J \preceq 0\}$$

where $B_I = (A_I A_I^T)^{-1} \in \mathbb{R}^{p\times p}$ and $D_I = A_J A_I^T B_I \in \mathbb{R}^{(s-p)\times p}$.

Proof. Having $\bar{y}$ as an optimal solution shows that there exists some $\bar{\lambda}_I \in \mathbb{R}^p$, $p = |I(\bar{y})|$, such that $\bar{y} + A_I^T \bar{\lambda}_I = 0$, $\bar{\lambda}_I \succeq 0$, $A_I \bar{y} = \bar{x}_I$, and $A_J \bar{y} \prec \bar{x}_J$. If the rows of A_I are linearly independent, $\bar{\lambda}_I = -(A_I A_I^T)^{-1}\bar{x}_I$, which implies $\bar{y} = A_I^T(A_I A_I^T)^{-1}\bar{x}_I$. Now, we can replace $\bar{x}_I$ by any x_I and obtain a corresponding optimal solution y determined by

$$y = A_I^T(A_I A_I^T)^{-1}x_I \ , \tag{6.7}$$

as long as we keep

$$\lambda_I = -(A_I A_I^T)^{-1} x_I \succeq 0 \ , \tag{6.8}$$

$$A_I y = x_I \ , \tag{6.9}$$

$$A_J y \prec x_J \ . \tag{6.10}$$

To satisfy (6.8) we must enforce $(A_I A_I^T)^{-1} x_I \preceq 0$. Equation (6.9) is a consequence of (6.7) multiplied by A_I. To satisfy (6.10), we must enforce $A_J A_I^T (A_I A_I^T)^{-1} x_I - x_J \prec 0$. Actually, y will still be optimal if (6.10) is replaced by $A_J y \preceq x_J$ (non-strict inequality). In that case, we use the convention that if some new constraints enter the set of active constraints at y, the index set I is still understood as the set of active constraints at $\bar{y}$.

To easily see that the resulting set $X(I)$ as defined in the proposition with B_I and D_I is a cone, assume without loss of generality that $x = \begin{bmatrix} x_I \\ x_J \end{bmatrix}$, allowing us to rewrite

$$X(I) = \{x \in \mathbb{R}^s : G_I x \preceq 0\} \quad \text{with} \quad G_I = \begin{bmatrix} B_I & \mathbf{0} \\ D_I & -\mathbb{I} \end{bmatrix} \in \mathbb{R}^{s \times s} \ ,$$

where $\mathbf{0}$ is the zero matrix of dimension $|I|$ and $\mathbb{I}$ the identity matrix of dimension $|J|$. □

Remark 6.7. The subset of $\operatorname{dom} \mathcal{C}$ for which there is no active constraint at the optimal solution is $X(\varnothing) = \mathbb{R}^s_+$: it is easy to check that $0 \in \operatorname{argmin} \mathcal{P}(x)$ if and only if $x \succeq 0$. That the point $x = 0$ is included in every set $X(I)$ corresponds to the existence of pathological cases (recall Figure 6.2) where several hyperplanes meet at zero. □

We close the section by a particularization of the results.

6.23 Proposition($\star$). *Consider the parametric program over $z \in \mathbb{R}^m$,*

$$\mathcal{Q}(\mu(\omega), v(\omega)) : \quad \textit{minimize} \quad (z - \mu(\omega))^T \Sigma^{-1} (z - \mu(\omega)) \quad \textit{subject to} \quad Fz \preceq v(\omega) \ .$$

Let $\mathcal{F}$ be the set-valued mapping with values $\mathcal{F}(v) = \{z \in \mathbb{R}^m : Fz \preceq v\}$, and let $\operatorname{dom} \mathcal{F} = \{v : \mathcal{F}(v) \neq \varnothing\}$. For some fixed $\bar{\mu}$ and $\bar{v} \in \operatorname{dom} \mathcal{F}$, let $\bar{z}$ be the optimal solution to $\mathcal{Q}(\bar{\mu}, \bar{v})$. With f_i^T denoting the i-th row of F, let $I = \{i : f_i^T \bar{z} = \bar{v}_i\}$ be the index set of active constraints at $\bar{z}$. Then, there exist a neighborhood Q_μ of $\bar{\mu}$ and a neighborhood Q_v of $\bar{v}$ such that for all $\mu(\omega) \in Q_\mu$ and $v(\omega) \in Q_v \cap \operatorname{dom} \mathcal{F}$, the optimal solution of $\mathcal{Q}(\mu(\omega), v(\omega))$ is

$$z^*(\omega) = \mu(\omega) + \Sigma F_I^T (F_I \Sigma F_I^T)^{-1} (v_I(\omega) - F_I \mu(\omega)) \ , \tag{6.11}$$

if the rows of F_I are linearly independent and $(F_I \Sigma F_I^T)^{-1}(v_I - F_I \mu(\omega)) \prec 0$. In fact, the expression (6.11) is valid if one has $v(\omega) \in \operatorname{dom} \mathcal{F}$, the rows of F_I linearly independent, and $\mu(\omega)$, $v(\omega)$ satisfying

$$(F_I \Sigma F_I^T)^{-1} (v_I(\omega) - F_I \mu(\omega)) \preceq 0 \ , \tag{6.12}$$

$$(v_J(\omega) - F_J \mu(\omega)) - F_J \Sigma F_I^T (F_I \Sigma F_I^T)^{-1} (v_I(\omega) - F_I \mu(\omega)) \succeq 0 \ , \tag{6.13}$$

where $J = \{j : f_j^T \bar{z} < \bar{v}_j\}$ is the complement of I.

Remark 6.8. There is a nice interpretation of $z^*(\omega)$ in (6.11) as the conditional mean of a random variable Z with realizations $Z(\eta)$, such that Z follows a priori a normal $\mathcal{N}(\mu(\omega), \Sigma)$, and then is conditioned on the observation $F_I Z(\eta) = v_I(\omega)$. $\square$

Proof of Proposition 6.23. All the developments in the section have been done for the parametric program (6.1), but can be applied easily to the parametric program (6.5),

$$\mathcal{Q}(u(\omega), v(\omega)) : \quad \text{minimize} \quad \tfrac{1}{2} z^T S z + u(\omega)^T z \quad \text{subject to} \quad Fz \preceq v(\omega) \enspace ,$$

with S positive definite. To adapt Proposition 6.19, for instance, let $\bar{z}$ be the optimal solution to $\mathcal{Q}(\bar{u}, \bar{v})$, and let $I = \{i : f_i^T \bar{z} = \bar{v}_i\}$. Let $S = R^T R$ be the Cholesky factorization of S. The change of variables $z = R^{-1}y - S^{-1}u(\omega)$ applied to the system of active constraints $F_I z = v_I(\omega)$ yields $F_I R^{-1} y = v_I(\omega) + F_I S^{-1} u(\omega)$, that is, $A_I y = x_I(\omega)$ if we set $A = FR^{-1}$ and $x(\omega) = v(\omega) + FS^{-1}u(\omega)$. Applying Proposition 6.19 and substituting back, we deduce that there exist some neighborhoods Q_u of $\bar{u}$ and Q_v of $\bar{v}$ such that for all $u(\omega) \in Q_u$ and $v(\omega) \in Q_v \cap \operatorname{dom} \mathcal{F}$, the optimal solution $z^*(\omega)$ to (6.5) is given by

$$z^*(\omega) = S^{-1} \left[F_I^T (F_I S^{-1} F_I^T)^{-1} (v_I(\omega) + F_I S^{-1} u(\omega)) - u(\omega) \right] \enspace ,$$

if the rows of F_I are linearly independent and $(F_I S^{-1} F_I^T)^{-1}(v_I(\omega) + F_I S^{-1} u(\omega)) \prec 0$. It remains to set $S = \Sigma^{-1}$ and $u(\omega) = -\Sigma^{-1}\mu(\omega)$ to get (6.11). The rest of the proposition follows similarly from Proposition 6.22. $\square$

6.4 Classifiers for Sets of Active Constraints

Our study of the optimal solution $y^*(\omega)$ to the program $\mathcal{P}(x(\omega))$ defined by (6.1) suggests that the exact prediction problem, mapping an input $x(\omega)$ to the output $y^*(\omega)$, can be reduced to the prediction of the index set of active constraints, mapping $x(\omega)$ to $I(y^*(\omega))$. The index sets of active constraints I partition the input space into subregions $X(I)$, found to be polyhedral cones. (The subregions are also called cells in the sequel.) Once $x(\omega)$ is known to belong to some cell $X(I)$, it is straightforward to find $y^*(\omega)$.

A cell $X(I)$ requires s linear inequalities to be described as a polyhedron, where s is the number of constraints of the parametric program $\mathcal{P}$. In a problem with s constraints, there could be an astronomically large number of index sets of active constraints I to consider. Enumerating them individually without prior knowledge would be a daunting task. But at least, Proposition 6.22 allows us to build instantly the cell $X(I)$ associated to a sample $(\bar{x}, \bar{y})$, where $\bar{y}$ is the optimal solution to $\mathcal{P}(\bar{x})$, and create a classifier associated to $X(I)$ for indicating whether a new input x is in $X(I)$. A classifier is simply a 0-1 indicator function of the set $X(I)$, that could be represented by a decision tree (Algorithm 6.1). By creating and exploiting existing classifiers, it would then be possible to "learn" minimizers in an online fashion (Algorithm 6.2).

6.4.1 Description of the Algorithms.

The proposed learning strategy is essentially memory-based: it consists in building a growing collection of local linear models. It does not attempt to generalize results beyond

Algorithm 6.1 Building a decision-tree classifier associated to a set of active constraints

Input: A sample point $(\bar{x}, \bar{y})$ such that $\bar{x} \in \operatorname{dom} \mathcal{C}$, $\bar{y} \in \operatorname{argmin} \mathcal{P}(\bar{x})$, and the rows of A_I, $I = I(\bar{y})$, are linearly independent.

Output: A classifier $\delta_I : \mathbb{R}^s \to \{0, 1\}$ defined on $\operatorname{dom} \mathcal{C}$ with values $\delta_I(x) = 1$ if $I(\operatorname{argmin} \mathcal{P}(x)) = I(\bar{y})$, and 0 otherwise.

```
1. Let $J = \{j : a_j^T \bar{y} < \bar{x}_i\}$ be the index set of inactive constraints at $\bar{y}$.
   Set $B = (A_I A_I^T)^{-1}$, and let $b_j^T$ denote the $j$-th row of $B$.
   Set $D = A_J A_I^T B$, and let $d_k^T$ denote the $k$-th row of $D$.
   Define $\phi_j(x) = b_j^T x_I$.
   Define $\psi_k(x) = d_k^T x_I - x_{J(k)}$, where $J(k)$ is the $k$-th index of $J$.

2. Create a root node and call it the current node.

3. Repeat for $j = 1, \dots, p = |I|$ :
   Split the current node using test $\phi_j(x) \leq 0$ (true for the left child),
   attach label $\{0\}$ to the right child, and call the left child the current node.

4. Repeat for $k = 1, \dots, s - p$ :
   Split the current node using test $\psi_k(x) \leq 0$ (true for the left child),
   attach label $\{0\}$ to the right child, and call the left child the current node.

5. Attach label $\{+1\}$ to the current node,
   meaning that the local model $y^* = A_I^T (A_I A_I^T)^{-1} x_I$, $I = I(\bar{y})$, is valid.
```

Algorithm 6.2 Learning minimizers (online version)

Input: A set of M classifiers $\{\delta_{I^\mu}\}_{1 \leq \mu \leq M}$ and a new sample $x \in \operatorname{dom} \mathcal{C}$.

Output: $y \in \operatorname{argmin} \mathcal{P}(x)$ and an updated set of classifiers.

```
1. If $x \succeq 0$, return $y = 0$, leaving the set of classifiers intact.

2. Evaluate at $x$ the classifiers $\delta_{I^\mu}$, $1 \leq \mu \leq M$.

3. As soon as $\delta_{I^\mu}(x) = +1$ for some $\mu \leq M$,
   set $I = I^\mu$ and return $y = A_I^T (A_I A_I^T)^{-1} x_I$,
   leaving the set of classifiers intact.

4. Otherwise, call a solver. Set $y \in \operatorname{argmin} \mathcal{P}(x)$, and set $I = I(y)$.

5. If the rows of $A_I$ are linearly independent,
   build a classifier $\delta_{I^{M+1}}$ associated to $I^{M+1} = I$,
   and append it to the set of existing classifiers.
```

the domain of validity of the local models; in particular, it does not attempt to build a single classifier per constraint, that would tell us whether a single constraint is active at the optimal solution. When the input data cannot be processed by existing local models, a standard quadratic programming solver is called, and its result is used to build a new local model.

Algorithm 6.2 can be viewed as a quadratic programming solver that adapts itself to the input data it receives, so as to "minimize" its response time. One could for example fix a maximal number of local models, and then allow local models that are infrequently called to be disposed of after some time, since the membership tests induce an overhead at most linear with the number of existing local models (we refine the linear complexity estimate in Lemma 6.28 below, and implement a strategy for managing the local models in Section 6.5).

Algorithm 6.2 can also be viewed as the builder of a decision policy h, that assigns to each input $x \in \mathbb{R}^s$ an optimal decision $y \in \mathbb{R}^m$. Initially, the decision policy always calls a quadratic programming solver, except when the solution is trivially 0. Denoting by π the mapping from the input x to the optimal solution y implemented by the solver, the policy h can be expressed as

$$h(x) = \begin{cases} 0 & \text{if } x \succeq 0 \ , \\ \pi(x) & \text{otherwise.} \end{cases}$$

After after some training during which inputs x^μ are received, outputs $y^\mu = h(x^\mu)$ are self-generated, and classifiers δ_{I^μ} are built with $I^\mu = I(y^\mu)$ the set of active constraints at y^μ, the decision policy $h : \mathbb{R}^s \to \mathbb{R}^m$ can exploit, in the region $\mathbb{R}^s_+ \cup \left(\bigcup_\mu X(I^\mu)\right)$ of the input space, an explicit representation of the optimal solution mapping from x to y. Namely, if $\mathcal{M}$ denotes the collection of index sets I^μ already seen, the policy h can be formally expressed by 3 pieces, assuming for notational simplicity that the probability of x falling on the boundaries of the cells $X(I)$ is 0:

$$h(x) = \begin{cases} 0 & \text{if } x \succeq 0 \ , \\ \sum_{I \in \mathcal{M}} \delta_I(x) A_I (A_I A_I^T)^{-1} x_I & \text{if } x \in \bigcup_{I \in \mathcal{M}} X(I) \\ \pi(x) & \text{otherwise.} \end{cases} \tag{6.14}$$

6.4.2 Complexity.

With a finite number s of constraints, the number of possible cells $X(I)$, say N, is finite but very large:

6.24 Lemma. *For the parametric program $\mathcal{P}(x)$ over $y \in \mathbb{R}^m$ with s constraints, the number of cells $X(I)$, written N, is at most*

$$\sum_{p=1}^{\min\{s,\, m\}} s!/(p!(s-p)!) \leq 2^s - 1 \ ,$$

if we do not count $X(\varnothing) = \{x \in \mathbb{R}^s : x \succeq 0\}$.

Proof. For the feasibility set $C = \{y \in \mathbb{R}^m : Ay \preceq x\}$, $A \in \mathbb{R}^{m \times s}$, and an index set I of active constraints of cardinality p, the cell $X(I)$ is well defined when the p rows of A_I $(1 \leq p \leq s)$ are linearly independent. Note that if A has rank s (possible only if $s \leq m$), then A_I has rank p and its rows linearly independent. Having the p rows of A_I linearly independent is impossible if $p > m$. Thus, the index sets to consider are those obtained by picking p constraints out of s, with $p \leq m$. For programs with $s \leq m$, there exist $2^s - 1$ theoretical combinations of active constraints. $\square$

Clearly, there is no hope of covering efficiently all the cells $X(I)$. The proposed approach is expected to work when the support of the distribution of x is concentrated on a relatively modest number of cells. Instead of building in a systematic way the explicit part of the mapping h, we let the construction process be driven by *i.i.d.* samples of x, always allowing calls $\pi(x)$ to the solver for samples that fall out of the domain where h is known explicitly.

The complexity of building a classifier can be estimated as follows.

6.25 Lemma. *Building a classifier δ_I requires at most $\mathcal{O}(ms^2)$ operations.*

Proof. We follow the notations of Proposition 6.22. If $A \in \mathbb{R}^{m\times s}$ is not assumed to be sparse, building a new classifier δ_I with $|I| = p$ requires $\mathcal{O}(mp^2)$ operations to form $A_I A_I^T$, $\mathcal{O}(p^3)$ operations to invert $A_I A_I^T$ and obtain $B_I = (A_I A_I^T)^{-1}$, $\mathcal{O}(mp^2)$ operations to form $A_I^T (A_I A_I^T)^{-1}$ from A_I^T and B_I, and $\mathcal{O}(m(s-p)p)$ operations to form $D_I = A_J A_I^T B_I$. Note that $m(s-p)p \leq ms^2/4$. $\square$

The complexity of exploiting a classifier δ_I (steps 1 and 2 of Algorithm 6.2) can be estimated as follows.

6.26 Lemma. *A test $x \in X(I)$ with $|I| = p$ requires at most $\mathcal{O}(sp)$ operations, meaning that the complexity is at most $\mathcal{O}(s^2)$ for all I.*

Proof. We follow the notations of Proposition 6.22. Checking that $B_I x_I \preceq 0$ requires $\mathcal{O}(p^2)$ operations, and checking that $D_I x_I - x_J \preceq 0$ requires $\mathcal{O}((s-p)p)$ operations. Note that $(s-p)p \leq s^2/4$. $\square$

(Lemma 6.26 does not take into consideration the fact that a test should be aborted as soon as one of the s inequalities to check is false.)

6.27 Lemma. *A prediction for y given $x \in X(I)$ with $|I| = p$ requires at most $\mathcal{O}(mp)$ operations, meaning that the complexity is at most $\mathcal{O}(ms)$ for all I if $s \leq m$, or at most $\mathcal{O}(m^2)$ if $s > m$.*

Proof. The prediction is $y = A_I^T (A_I A_I^T)^{-1} x_I$. The matrix product has already been evaluated in the construction of the classifier δ_I, so that the complexity of evaluating the prediction is reduced to the complexity of the matrix-vector multiplication. $\square$

If we assume that a stored classifier δ_I is never replaced by another in the course of the training phase, appending a new classifier to the collection $\mathcal{M}$ of stored classifiers as described in Algorithm 6.2 can only make smaller the probability

$$p_0 = \mathbb{P}\{x \not\preceq 0 \text{ and } x \notin X(I) \text{ for all } I \in \mathcal{M}\} \tag{6.15}$$

of a new sample x falling in an unknown subregion of the input space, but potentially delays the call $\pi(x)$ to the solver.

It would be interesting to be able to check whether x is in the union of the cells in $\mathcal{M}$, so as to avoid unnecessary tests, but this approach seems rather difficult to concretize — at least considering the number of tests induced by the expression of the convex hull of the union of k polyhedra (Balas, 1998). In the following lemma, we assume that all tests $x \in X(I)$, $I \in \mathcal{M}$, have to be done before being able to conclude that $x \notin \bigcup_{I \in \mathcal{M}} X(I)$.

6.28 Lemma. *Let N be the total number of cells $X(I)$ induced by the parametric program $\mathcal{P}(x)$. Let $M = \alpha N$, $\alpha \in (0, 1]$, be the number classifiers appended sequentially to an initially empty collection $\mathcal{M}$ of classifiers, new cells $X(I)$ being potentially discovered as new i.i.d. samples of x are received. Let h be the policy* (6.14) *that maps x to* $\operatorname{argmin} \mathcal{P}(x)$. *Then, the expected number of tests of the form $x \in X(I)$ in the evaluation of $h(x)$ is upper bounded by $M(1 - \alpha/2) + \alpha/2$.*

Proof. Let $X_1, \ldots, X_N$ be the polyhedral cells $X(I)$ induced by the parametric program $\mathcal{P}$. Let q_i be the probability that $x \in X_i$. Let $\delta_1, \ldots, \delta_M$ denote the distinct classifiers of the collection $\mathcal{M}$, indexed in the order of their creation. Each classifier δ_j is associated to a different cell X_i, and the probability of the possible matchings is a function of the probabilities q_i. Given the sequence $\{\delta_j\}_{1 \le j \le M}$, let $t(x)$ be the number of tests (implemented by the classifiers) needed to detect any event $x \in X_i$, or the event that x falls in a cell not covered by any δ_j. We have $t(x) = 0$ if $x \succeq 0$, $t(x) = j$ if there exists some j such that $\delta_j(x) = 1$, and $t(x) = M$ otherwise.

The choice of q_i that maximizes $\mathbb{E}\{t(x)\}$ (where the expectation is taken over x and over the possible sequences of classifiers) is obtained with $q_i = 1/N$ for each i, since having any $q_i > 1/N$ would make X_i more likely to appear sooner in the sequence of classifiers, and every new sample x more likely to hit X_i. Note that $q_i = 1/N$ also means that the probability of $x \succeq 0$ is chosen to be zero.

Then, thanks to the property that all the orderings of the classifiers are now equiprobable, it holds that $t(x) = 0$ with probability 0, $t(x) = j$ with probability $1/N$ if $j < M$, and $t(x) = M$ with probability $1/N + (N - M)/N$. Writing $M = \alpha N$ for some $\alpha \in (0, 1]$, the expectation of t with the worst-case distribution is

$$\begin{aligned}\mathbb{E}\{t(x)\} &= \sum_{j=1}^{M} \frac{j}{N} + M\frac{N-M}{N} = \frac{M(1+M)}{2N} + M\frac{N-M}{N} \\ &= N(\alpha - \alpha^2/2) + \alpha/2 \ = M(1 - \alpha/2) + \alpha/2 \ .\end{aligned}$$

□

The expected time complexity of evaluating h as specified by (6.14) with $M = \alpha N$ stored classifiers on a new sample x could be estimated as follows. With T_π denoting the expected time for evaluating $\pi(x)$, T_X the expected time for evaluating a cell membership test (Lemma 6.26), T_y the expected time for evaluating a prediction given the positive classifier (Lemma 6.27), and p_0 the probability (6.15), h would be evaluated in expected time

$$T_h \le p_0 \cdot T_\pi + (1 - p_0) \cdot T_y + (M(1 - \alpha/2) + \alpha/2) \cdot T_X \ ,$$

using the bound of Lemma 6.28 based on a worst-case distribution, which is also insensitive to the possible reordering of the tests $x \in X(I)$.

6.5 Numerical Experiments

The potential merits of the proposed algorithms are evaluated on various random problems. We recall that problems with random constraint matrices are not necessarily representative of practical problems (Edelman, 1992) — for the simplex method, they do provide insights, but they are unable to explain the behavior of the algorithm in a relevant neighborhood of some fixed input data (Spielman and Teng, 2004). However, random problems are easy to specify, and by a statistical concentration phenomenon, large problems tend to be very similar; taken together, these two features facilitate the replication of experiments and observations.

6.5.1 Description of the Test Problems

We create the test problems as follows. We select sets of parameters for the input dimension and the number of constraints, namely:

$$(m, s) = (5, 10), (10, 5), (10, 20), (20, 10), (20, 40), (40, 20).$$

For each set, we generate one random matrix $U \in \mathbb{R}^{s\times m}$ with i.i.d. elements U_{ij} drawn from the uniform distribution in $[0, 1]$. We define $V^k = U - 0.1k\mathbf{11}^T$ for $k = 1, 2, \ldots, 5$, where $\mathbf{11}^T$ is a matrix of ones in $\mathbb{R}^{s\times m}$. We form the matrix A^k from V^k by stacking the s rows $a_i^k = 2\alpha_i^k v_i^k / ||v_i^k||$, where α_i^k is drawn from the uniform distribution in $[0, 1]$, and v_i^k is the i-th row of V^k. We call $P(m, s; k)$ the problem with $A = A^k \in \mathbb{R}^{s\times m}$. It will turn out that problems get harder with k higher.

The parameter k controls the diversity of the directions normal to the halfspaces defined by the random rows of A. With k small, the first singular value of A^k tends to dominate the others.

In every problem $P(m, s; k)$, we draw samples for x as follows. Starting from an orthonormal basis $\tilde{A} \in \mathbb{R}^{s\times m_0}$ for the range of A, where $m_0 = \min\{s, m\}$ (the basis is obtained from the svd decomposition of A), we set $x = \tilde{A}\xi_0 + 0.5|\xi_1|$, where $\xi_0 \in \mathbb{R}^{m_0}$ and $\xi_1 \in \mathbb{R}^s$ are drawn from standard multivariate normal distributions, and $|\cdot|$ denotes the elementwise absolute value. By Proposition 6.12, $x \in \operatorname{dom} \mathcal{C}$. We have kept the magnitude of the term in $|\xi_1|$ relatively small, so as to avoid the case $x \succeq 0$, to which is associated the optimal solution $y^* = 0$. Notice that the support of the distribution of x is unbounded.

6.5.2 Description of the Experiments

We conduct a test on a problem $P(m, s; k)$ as follows. We run Algorithm 6.2 on 5000 i.i.d. samples for x, storing classifiers if the rows of A_I are linearly independent. The rank deficiency of A_I is checked from the svd decomposition of A_I, which is also used to compute $(A_I A_I^T)^{-1}$. Then, we test the stored classifiers on 5000 new i.i.d. samples to estimate to which extent these classifiers cover the part of the input space relevant for the distribution of the input data.

The results are presented on Table 6.1. We report:

- M : the number of classifiers δ_I built during training (online learning). Note that

M also gives the number of calls to the quadratic programming solver made during the training phase.

- n_0 : the number of learning samples with $y^* = 0$ (identified by $X(\varnothing)$: test $x \succeq 0$). These samples are not compensated by new samples, so they simply diminish the effective size of the training set.
- n_h : the number of samples processed by an already existing classifier during training. These samples are not compensated by new samples. They are lost for the detection of a useful cell $X(I)$, but can be used to estimate the relative importance of the classifiers, so as to reorder the classifiers at some point.
- X_h : the fraction of test samples that can be processed by the learned classifiers, corresponding to the fraction of samples from the test set that hit some cell $X(I)$ seen during the training phase.

In these experiments, we did not encounter rank-deficient matrices A_I, so for each row of the table, we have $M + n_0 + n_h = 5000$.

We have also compared the cpu time taken by calling the quadratic programming solver $\pi(x)$ on the 5000 i.i.d. samples (reference method), to the cpu time taken by running the online learning/prediction algorithm on the same 5000 samples, using in addition the following rules for building and updating the collection of stored classifiers:

1. Never create more than 1000 classifiers over the course of the online learning process.
2. Never store more than 250 classifiers simultaneously.
3. Every 250 samples, reorder the classifiers by decreasing frequency of use, and remove the stored classifiers that were never recalled after their creation.

Note that a same classifier could be rebuilt several times if it is removed too soon (especially for the last classifiers to be added within the window of 250 samples). However, the rules imply together that a same classifier will be rebuilt at most 4 times. The purpose of the first rule is to be able to decide online whether the learning approach should be pursued: if one keeps building or rebuilding classifiers all the time, the problem is not well adapted to the online learning approach, and one should stop building new classifiers.

The cpu times are reported on Table 6.1. In those tests, the solver is the Matlab function `quadprog`.

6.5.3 Discussion of the Results.

The results of Table 6.1 suggest that for several problems, especially those where $s < m$ or k is small, the proposed approach is promising. A relatively small number of classifiers suffices to cover almost all the input space relevant to the (unbounded) input distribution, as shown by the fractions X_h close to 1. For those problems, we measured speed-up factors between 2 and 15 over the systematic strategy that calls a quadratic programming solver for each sample.

For other problems, especially those with many constraints and higher values for the parameter k, the merits of the approach are less clear. The local models are valid on a relatively small volume of the input space, leading to a multiplication of the classifiers to build. As the multiplication of the tests begin to hurt computing times, the management rules for maintaining a useful collection of classifiers start to be important, if the proposed approach is to remain competitive with the usual approach consisting in solving every problem instance by calling the solver. Notice that the problem $P(10, 5; 5)$ has $M = 2^s - 1 = 31$ classifiers, meaning that all possible combinations of active constraints are needed. Therefore, we believe that the problem $P(20, 40; 5)$, on which the worst fraction of the input space covered by 5000 classifiers is recorded, could in fact require, by Lemma 6.24, as many as $0.6 \cdot 10^{12}$ classifiers.

The last series of problems with $m = 40$, $s = 20$, illustrates clearly that the practical speed-up performance of the proposed approach depends on the problem data A. The whole table illustrates that the overhead cost of learning and maintaining classifiers can be controlled, so that there is in fact a very strong incentive to use the proposed approach.

To close this section, we give on Figure 6.3 an example of prediction for the first test problem P(5,10;1): two points x_0, $x_1 \in \mathbb{R}^{10}$ have been drawn randomly, and the 5 components of the optimal solution along the line segment $x(t) = (1 - t)x_0 + tx_1$, $t \in [0, 1]$, have been plotted against t.

Tab. 6.1: Results on test problems: Covering by classifiers and Speed-up performance.

Parameters			Training			Testing	Cpu time (sec)	
m	s	k	M	n_0	n_h	X_h	(Ref)	
5	10	1	101	423	4476	0.9956	11.81	1.51
5	10	2	131	418	4451	0.9946	12.54	2.08
5	10	3	176	400	4424	0.9930	14.56	3.20
5	10	4	228	340	4432	0.9868	15.19	4.08
5	10	5	251	416	4333	0.9866	15.02	4.64
10	5	1	25	609	4366	1.0000	11.38	0.59
10	5	2	25	584	4391	0.9998	12.02	0.69
10	5	3	30	588	4382	1.0000	13.51	0.85
10	5	4	30	535	4435	0.9998	14.20	0.89
10	5	5	31	567	4402	1.0000	15.50	0.94
10	20	1	465	36	4499	0.9544	15.45	6.31
10	20	2	742	29	4229	0.9276	17.23	11.17
10	20	3	1398	30	3572	0.8394	22.00	21.06
10	20	4	3092	22	1886	0.5216	28.87	31.83
10	20	5	3361	28	1611	0.4594	29.14	31.98
20	10	1	96	74	4830	0.9926	15.20	1.66
20	10	2	162	66	4772	0.9932	17.56	2.92
20	10	3	311	62	4627	0.9844	23.84	7.49
20	10	4	399	65	4536	0.9792	28.95	9.98
20	10	5	575	68	4357	0.9756	37.83	18.38
20	40	1	1513	0	3487	0.8032	24.63	21.08
20	40	2	2322	1	2677	0.6574	29.05	29.68
20	40	3	3957	0	1043	0.3134	38.38	40.26
20	40	4	4966	0	34	0.0180	57.51	58.24
20	40	5	5000	0	0	0.0002	86.14	86.86
40	20	1	276	3	4721	0.9766	22.99	5.59
40	20	2	674	0	4326	0.9236	33.92	15.36
40	20	3	1645	0	3355	0.7782	44.92	36.11
40	20	4	3181	1	1818	0.4850	88.13	79.72
40	20	5	4516	0	484	0.1442	130.01	127.88

m, s : number of variables and constraints of the problem (dimensions of A)
k : method of construction of the constraint matrix A (see text)
M : number of classifiers after training (without storage management rules)
n_0 : samples with a zero optimal solution
n_h : samples processed by an already existing classifier during training
X_h : empirical fraction of the relevant input space covered by the classifiers

Cpu time: of online learning on 5000 samples, using classifier storage management rules (see text).

Ref: cpu time if one calls a solver on each of the 5000 samples.

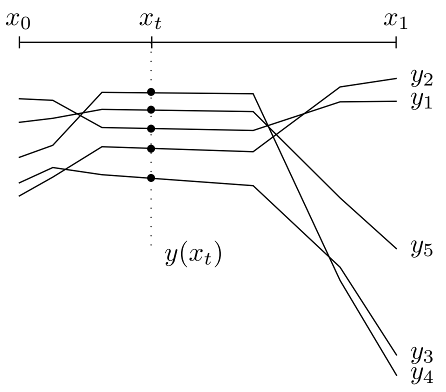

Fig. 6.3: The components y_k of the optimal solutions $y \in \mathbb{R}^5$ for the test problem $P(5, 10; 1)$, along a random line segment defined by $x_t = (1-t)x_0 + tx_1 \in \mathbb{R}^{10}$, $0 \leq t \leq 1$. Breakpoints in the y_k-curves indicate where the segment cuts a boundary between the domains of 2 classifiers.

6.6 Conclusions

This chapter has discussed a family of parametrized optimization programs and a hypothesis class for predicting, after some training on the task of solving random instances, optimal solutions to new instances of the program. Based mainly on geometrical insights, the analysis of the properties of optimal solutions has also emphasized the role of constraint qualifications, and the possible occurrence of pathological cases for some distributions of the input data. A natural choice for the hypothesis class was a piecewise linear model describing how optimal solutions vary locally with the input data. Fitting the model was possible using a strategy based on the exploitation of first-order optimality conditions.

The technical assumption that the rows of A_I (rows of active constraints at the optimal solution) are linearly independent corresponds to a linear independence constraint qualification (LICQ). It implies that the set of optimal dual solutions is a singleton (Facchinei and Pang, 2003, Proposition 3.2.1). Note that in nonlinear programming, a necessary and sufficient condition ensuring that the set of optimal primal-dual solutions is a singleton is the strict Mangasarian-Fromowitz constraint qualification (SMFCQ) (Kyparisis, 1985) — see again Facchinei and Pang (2003, Proposition 3.2.1).

It is well known that a quadratic program subject to constraints with parametrized righthand side admits a piecewise linear optimal solution (Garstka and Wets, 1974, Proposition 3.5). Early results involving perturbations of the constraint matrix are also available (Daniel, 1973), but they do not really allow to circumvent the difficulties that arise from inequality constraints forming new equality constraints (the merging effect detailed in Remark 6.6). An example revealing the combinatorial nature of the difficulties, inspired from a linear program given by Martin (1975), is provided by the following program over $y = (y_1, y_2) \in \mathbb{R}^2$, where the constraint matrix depends affinely on $t \in \mathbb{R}$:

$$\text{minimize} \quad \tfrac{1}{2}||y||^2 \qquad \text{subject to} \quad y_1 + y_2 \geq 1 \ , \quad y_1 + ty_2 \leq 1 \ .$$

The optimal solution, $y^* = (1/2, 1/2)$ if $t \leq 1$, $y^* = (1, 0)$ if $t > 1$, is a discontinuous function of the parameter t. The optimal value as a function of t is also discontinuous at $t = 1$.

That the input space (set of values for the parameter x of the program $\mathcal{P}(x)$) can be partitioned into polyhedral subregions associated to index sets of active constraints is a well-established result in variational analysis. One already finds it in spirit in a basis decomposition theorem for linear programming proved in Walkup and Wets (1969a) and restated in Wets (1974, Theorem 7.2). Interestingly, local Lipschitz continuity properties useful in the analysis of the stability of two-stage stochastic linear programs (see Appendix D, Lemma D.13) are proved in Rachev and Römisch (2002, Proposition 3.2) by appealing to such a decomposition.

The decomposition of the input space into polyhedral cells is rediscovered in Bemporad et al. (2002) in the context of online quadratic programming for Model Predictive Control (MPC). Further work along that line (including work on better implementations) has been pursued since then (Tøndel et al., 2003; Spjøtvold et al., 2007; Baotić et al., 2008). We also arrive in Section 6.3 to the polyhedral decomposition in the context of our learning setting, but then we let the construction of the cells be guided by the empirical distribution of the input data. In the practical implementation of the approach, we use a hybrid method able to find a tradeoff between cell-based, closed-form calculations and standard optimization, recognizing that all problems are not addressable efficiently by the subregion-based approach.

Indeed, the worst-case complexity of parametric linear programming (Murty, 1980) suggests that it is always possible to come up with examples contrived in such a way that a subregion-based approach performs badly — in our case, the problems $P(m, s; 5)$ with the rows of the constraint matrix uniformly distributed in every direction.

The identification of active constraints has been studied by Facchinei et al. (1998) in the context of nonlinear programming and variational inequalities through so-called *identification functions*. This work starts from the observation that under the Mangasarian-Fromowitz constraint qualification, a primal-dual solution has a neighborhood where the set of active constraints is not modified. The technical conditions that identification functions have to satisfy are defined, and identification functions are built for specific problem classes by exploiting error bounds valid for those classes — see also Facchinei and Pang (2003, Chapter 6). The determination of the subregion where the identification of active constraints is correct is left as an open problem. The methodology that we have adopted in this chapter is (i) to find conditions ensuring the existence of a neighborhood where active constraints are not modified (Proposition 6.19); (ii) to check whether the subregion associated to a set of active constraints is a connected set (Proposition 6.21); (iii) to see whether the extent of the subregion could be found (Proposition 6.22).

The notion of local models used throughout the chapter can be seen as a basic form of *single-valued localization* for general solution mappings, as developed by Dontchev and Rockafellar (2009). Investigating some aspects of the framework expounded in Dontchev and Rockafellar (2009) was also one of the goals of this chapter.

Finally, we mention that some additional work would be needed to adapt the results established in this chapter to the concrete MAP repair procedure used in Section 5.3 for restoring the feasibility of decisions predicted by learned policies. In particular, the results relative to the parametric program (6.1) would have to be generalized to the parametric program

$$\mathcal{P}'(x(\omega), w(\omega)) : \quad \text{minimize } \tfrac{1}{2}||y||^2 \quad \text{subject to} \quad Ay \preceq x(\omega)\ ,\ \ B_1 y + B_2 z \preceq w(\omega)\ \ ,$$

where the minimization is over $y \in \mathbb{R}^{m_1}$ and $z \in \mathbb{R}^{m_2}$, and the parameters are $x(\omega) \in \mathbb{R}^{s_1}$ and $w(\omega) \in \mathbb{R}^{s_2}$. This form is more general than (6.1), except when B_2 is such that $\{B_2 z : z \in \mathbb{R}^{m_2}\} = \mathbb{R}^{s_2}$ (which allows to ignore the constraint $B_1 y + B_2 z \preceq w(\omega)$, trivially satisfied for any y with some proper choice of z). The mapping of interest is the mapping from $(x(\omega), w(\omega))$ to the uniquely determined part $y^*(\omega)$ of an optimal solution.

Chapter 7

Conclusion

This thesis presents novel strategies for the search of approximate solutions to multistage stochastic programs. The framework is based on the association of statistical learning techniques to scenario-tree approximation techniques from the multistage stochastic programming literature. At first, the framework serves two purposes:

- Making it possible to estimate the true value of an approximate solution in a generic way;
- Making it possible to build discretizations (scenario trees) that allow to obtain satisfying solutions to the true problem.

We propose several practical implementations of the framework based on various supervised learning techniques, ranging from neural networks to Gaussian processes, and apply the general methodology to a variety of multistage problems (a family of risk-averse production management problems under price uncertainty, and a multi-product assembly problem under demand uncertainty).

From a higher level perspective, the association of multistage stochastic programming and supervised learning can be viewed as a specific method for sequential decision making under uncertainty, well adapted to large continuous action spaces. The quality of the decision policies found on test problems with this approach suggests that it is possible to capture a part of the value of multistage stochastic programming models in a variety of contexts, at an acceptable computational cost.

7.1 Summary of Contributions

A multistage stochastic programming problem is an optimization problem of the form $\min_\pi \mathbb{E}\{f(\xi, \pi(\xi))\}$, where $\xi = \xi_1, \dots, \xi_T$ is a random process, π is a mapping from ξ to a sequence of decisions $u = u_1 \dots, u_T$ adapted to the gradual observation of ξ, and $f(\xi, \pi(\xi))$ is the cost of applying $\pi(\xi)$ with ξ. The cost is formally set to $+\infty$ if the sequence of decisions $\pi(\xi)$ is infeasible on ξ. Chapter 2 has provided an introduction to this mathematical formalization, while more technical material has been collected in the Appendices.

In the case where ξ has a continuous distribution, the expectation operator represents a multidimensional integral, so even before considering the problem of searching for a best mapping π, the mere evaluation of the cost function f given a fixed mapping $\bar{\pi}$ is difficult. For general distributions, the question "Is $\mathbb{E}\{f(\xi, \bar{\pi}(\xi))\} \leq \theta$?" can only be answered up to a certain probabilistic confidence level $\alpha < 1$.

A second computational difficulty stems from the fact that approximate stochastic programming solution techniques furnish "solutions" that are fully specified only for the first decision stage. To evaluate on a new realization of ξ the mapping π (the decision policy) induced by these approximation techniques, one has to solve a sequence of approximate versions of the original problem posed over gradually shrunk time horizons (see Chapter 2).

Our diagnosis is that combined together, these two computational difficulties render it impossible to assess in practice the third level in the hierarchy of the stochastic programming methodology, namely, the adjustment of the sampling or discretization method that replaces expectations by finite sums (so as to yield a program on a finite number of optimization variables). Yet, we view this third level as a key ingredient for the success of the whole approach (specific constructions back up this view in Chapter 4): the fact that the random process is gradually observed, translated to a tree-structured representation of the samples (scenarios), leaves many degrees of freedom for adjusting the location of branchings in the tree, a possibility that should be exploited in the context of problems posed over long time horizons (or more generally in the context of multistage problems where the dimension of the random process is high).

In Chapter 2, we have presented multistage stochastic programming in the context of several competing frameworks and methods for sequential decision making uncertainty, such as Markov Decision Processes (MDP) and Model Predictive Control (MPC). We have mentioned several solution heuristics for multistage stochastic programming that have been explored in the optimization and operations research literature, such as two-stage approximations, aggregation and averaging strategies, and consensus strategies (Section 2.3).

In principle, the value of a multistage stochastic programming model over other or simpler models cannot be estimated without building and developing a solution method for all those models on the real data. The examples and case studies presented in the thesis have been selected (Section 4.4) or created (Example 2.1, Section 5.3) after careful experimentations on the model and problem data, in such a way that the multistage model had a high value with respect to other models — in particular two-stage approximations — given the numerical data. This was an important stage for a sound evaluation of the solution methods proposed in the thesis, but also time-consuming, which explains in part why we chose not to multiply the number of examples or assess the methods on problem instances with random or arbitrary data.

In Chapter 3, a series of statistical estimation methods has been considered, from maximum likelihood and maximum a posteriori estimation to bootstrap aggregating methods (bagging). The particular mix of perturbation, averaging and selection steps that differentiates those methods suggests that the estimation and optimization aspects in stochastic programming problems could in fact be given a unified treatment, based on Monte Carlo methods and importance sampling methods. The Cross-Entropy method for the simulation of rare events, and its application to combinatorial optimization, was identified as a promising candidate for reducing the conceptual gap between the two aspects (Section 3.2.3). At the same time, the idea of solving an ensemble of random approximations to a multistage stochastic program, and then aggregating the results, was presented as an estimation technique, related to bagging, that can be applied to a set of perturbed (noisy) first-stage decisions.

To serve as a test bed for the development of these ideas, a multistage stochastic programming formulation of a simple benchmark problem from reinforcement learning with a finite discrete action space has been considered (Section 3.3). The optimization of scenario-tree approximations is made via the Cross-Entropy method. Standard aggregation procedures based on averages or majority votes are not suitable for the processing of discrete decisions, so we have proposed a generalization of the aggregation operation based on the introduction of positive-definite kernels. This approach has been evaluated by a series of experiments on the test problem, for which it was shown that the proposed aggregation technique allows indeed to select a good first-stage decision out of the set of candidate optimized first-stage decisions.

In Chapter 4, we have proposed a practical procedure for deriving candidate mappings π (decision policies) from a data set containing optimized sequences of decision contingent to scenarios, originating from an optimal solution to a given scenario-tree based approximate program. We have proposed the use of supervised learning techniques for inferring a mapping π, decomposed into its successive decision rules for $u_1, \dots, u_T$. The learning task raises however several issues: (i) Each individual learning problem is over a growing input space, and over an output space corresponding to a decision u_t of potentially large dimension; (ii) The decision policy π has to comply with the original program's feasibility constraints, that structure the sequence of decisions and the components of the individual decisions u_t, so that the learning problems cannot be completely decoupled; (iii) The data sets of decisions are noisy and biased (as they are obtained from approximate programs), whereas the learned policy should have good performances on the true multistage stochastic program; (iv) The sequences of decisions $\pi(\xi)$ must be computed quickly, a requirement that restricts the policy representations usable in practice (for example, by introducing limitations on the size of the models built with nonparametric methods).

A first set of strategies to circumvent these difficulties has then been proposed and evaluated in the context of a practical problem (Section 4.4). Namely,

- Preprocessing the data set of optimal solutions and scenarios, so as to obtain a compact representation of the information state. The introduction of new state variables, that may depend on past decisions and past state variables, allows a dynamical decoupling of the learning problems.
- Posing the learning problems over a transformed output space, where the feasibility constraints can be more easily enforced.
- Combining learned models to repair procedures, so as to adjust the predicted decisions to the actual feasibility constraints.
- Basing the model selection of a feasible policy π on an estimate of its expected cost on the true multistage problem, rather than on the loss function of the supervised learning problems.

Two model selection strategies have been proposed. The first one consists in the simultaneous search of the hyperparameters of π viewed as a whole entity. The second one consists in searching the hyperparameters of π gradually on the sequence of decisions, using optimizations over independent scenario trees of shrinking depth as a proxy as long as the policy π is not fully specified (see Section 4.2.3).

In Chapter 4, we have taken advantage of these novel solution valuation capabilities to open up new avenues for the generation of scenario trees adapted to multistage problems on long time horizons. Specifically,

- We have proposed to consider several scenario trees for a same multistage problem, each of them leading to distinct approximations of the true problem. The trees are to be ranked according to the value of the best policy that can be learned using the data set of decisions optimized on those trees.
- We have suggested to retain the best policy of the best tree, say $\pi^\star$, as a suboptimal but feasible solution to the true multistage problem. The empirical estimate $\hat{\theta}$ of $\mathbb{E}\{f(\xi, \pi^\star)\}$, obtained by simulating the policy $\pi^\star$ on a new independent test sample, furnishes a certificate of performance on the true problem. The estimate $\hat{\theta}$ can be adjusted if one wants to consider confidence levels. In other words, $\pi^\star$ is a witness to the claim

$$\min_{\pi} \mathbb{E}\{f(\xi, \pi(\xi))\} \leq \hat{\theta}$$

  at a certain level of confidence. As the full distribution of the cost of using $\pi^\star$ can also be estimated from Monte Carlo simulation (using histograms), arbitrary measures of risk could also be reliably estimated, using a very large test sample.
- Practically, we have proposed to randomize the branching structure of the scenario trees, so as to obtain a rich diversity in the considered scenario trees. We have proposed simple strategies that allow to generate the trees in a top-down fashion (and thus do not require to build and store large trees before pruning them in a bottom-up fashion).
- We have demonstrated on a family of test problems that the full approach is implementable in practice, at a very moderate computational cost, and yields, for those test problems, near-optimal policies.
- Thanks to the moderate computational cost of this novel tree selection method, we were able to study empirically the effect of meta parameters on the quality of the solution, such as the number of scenarios in the trees to consider, or the type of sampling processes for the values at the nodes of the trees.

Our experimentations indicate that considering a large number of small trees can lead to an excellent tradeoff between solution quality and computational time.

In Chapter 5, we have considered a second set of strategies for learning policies, in the context of a four-stage multi-product assembly problem under demand uncertainty for which the value of the multistage formulation is high (Section 5.3).

- The general principle under the proposed learning approach is that the decisions of a policy could initially be represented as probability densities conditioned on a growing number of observations.

- The selection of a single decision is formulated as a maximum a posteriori (MAP) estimation problem, subject to feasibility constraints (Section 5.1). Under suitable assumptions on the densities to consider, the procedure is closely related to the projection of a single predicted decision on the current feasibility set.

- The framework has been found to lead naturally to Gaussian Process regression techniques.

- The choice of the covariance matrices (kernels) of the Gaussian Process models is sometimes facilitated by the knowledge of the algorithm that generates the scenario tree.

- Experiments on the test problem have demonstrated that with a suitable choice of the kernels and of the repair procedure (the projection of the candidate decisions on the current feasibility set), a near-optimal policy could be selected.

In Chapter 6, we have addressed the more fundamental issue of the usefulness, in terms of computational complexity, of building a model for predicting exact optimal solutions rather than computing them with a standard optimization procedure, albeit the issue has been addressed in the context of a specific setting (projections on random polyhedra).

- We have indeed formulated the question in a setting where fruitful results and insights can be derived. The setting is a well-known class of parametric strictly convex quadratic programs, and is related to the feasibility restoration task as formulated in Chapter 5, although some additional work would be needed to generalize the results to general convex quadratic programming problems.

- We have improved our understanding of the structure of the solutions to these parametric programs by establishing, from basic geometrical principles, properties of the optimal solutions.

- In particular, we have identified a relation between the potential size of an exact predictive model for the optimal solutions, and the smallest positive singular values of some matrices. We have illustrated that relation by generating parametric test problems in a particular way, and then estimating the size of the predictive models.

- We have developed a self-improving algorithm for solving a set of instances from the considered class of parametric optimization problems, able to find a tradeoff between an online learning approach and a pure optimization approach, mostly by controlling the size of the learned model.

7.2 Perspectives

Learning a policy from a data set of optimized decisions is a technical compromise. One would certainly prefer to optimize the parameters of a parametric policy directly on a scenario tree, or by simulation combined to stochastic gradient descent techniques.

The idea of searching for policies directly seems to be as old as stochastic programming itself (Garstka and Wets, 1974). Unfortunately, at the exception of particular settings where simple parametrizations can be used (decisions u affine in ξ or in features of ξ) (Garstka and Wets, 1974; Ben-Tal et al., 2004), such formulations lead to nonconvex optimization problems. If one can still in theory argue for the selection of a local minimum, one has also to observe that a parametric policy can seldom accommodate hard constraints, so that for most practical multistage problems, fixing the parametric form and optimizing the parameters directly would simply lead to an infeasible problem. Circumventing such difficulties by relaxing hard constraints or reformulating the optimization problem in terms of state variables would then simply bring us back to frameworks based on Markov Decision Processes (Section 2.2.2).

By taking the intermediate step of exploiting the usual multistage stochastic programming framework to obtain sequences of decisions tractably, one furnishes an initial data set of decisions that plays the role of an initial condition for the simulation-based optimization of a policy. As much flexibility is gained in the representation of the policy, the policy can now be nonlinear (as in neural networks), non-parametric (as in Gaussian processes), and/or incorporate repair procedures adapted to the hard constraints. From that point of view, the use of several scenario trees can be interpreted as the use of several initial conditions from which a policy can be locally optimized, while algorithms for generating good scenario trees would implicitly aim at generating good initial conditions.

Clearly, by introducing an intermediate step in the direct policy optimization task, we isolate (decouple) the two subtasks of optimizing decisions and optimizing policy parameters, and thus block the circulation of information between the two subtasks. A full range of iterative methods, alternating between the two subtasks, could be developed with the aim of restoring the information flow — the iterative model selection procedure (Algorithm 4.2) is a step in that direction. However, one would still have to demonstrate that such an approach has some practical advantages over standard decision making strategies based on successive shrinking-horizon optimizations.

In this thesis, we have presented methods, algorithms, and test problems on which significant computational gains over alternative approaches could be shown. One can still think of several potential improvements or extensions to the proposed techniques, that will have to be sorted out by applying the proposed solution methodology based on multistage stochastic programming and supervised learning to a wider variety of applications.

- We have considered an application with many decision stages and a low-dimensional random process (Section 4.4), but we have not considered an application with few decisions stages and a high-dimensional random process. An adaptation of the choices concerning the supervised learning component, and the model selection procedure may be needed in that context.

- We have not considered multistage problems based on mixed-integer linear programming formulations, that are currently under active investigation (Escudero, 2009). Our impression is that the advantage of a multistage stochastic programming formulation over a Markov Decision Process formulation is less clear when convex optimization tools cannot be used. Yet, an adaptation of the supervised learning approach to discrete decision spaces is possible, in the spirit of the proposals made in Chapter 3, and in that case, it should be possible to identify an application on which the advantages of the solution methodology developed in

Chapter 4 would appear.

- For specific applications, specific decisions rules might be proposed and tested. For instance, it is often the case in planning and sequential decision making under uncertainty that one is offered the choice to act now (the implementation details being adjusted greedily), or to postpone the decision. Such situations have been analyzed by Van Hentenryck and Bent (2006, Chapter 8) in the context of the online dispatching of a taxi fleet, but could also be found in electricity generation planning (optimal response to contingencies). Then, a fundamental component of the decision policy is the mapping from the information state, possibly represented by features, to the delay before taking irreversible or expensive decisions. The mapping could be learned according to the data collected from optimized multistage stochastic programs, and then further adjusted by simulations. If the decision of acting now is selected, what we refer to as the repair procedure could be anything from a greedy, one-step online optimization, to a call to another policy dedicated to immediate actions.

- The proper way to associate, in a same data set, scenario/decisions examples collected from several scenario-tree approximations solved to optimality, so as to infer from this data set, or a post-processed version of it, a policy with better performances on the exact problem than the best of the policies learned from the data relative to a single scenario tree, is still to be found and shown to be computationally efficient. Our intuition is that this approach could be especially interesting for multistage problems with high-dimensional random processes, but would require much work to ensure that the inconsistencies among the data sets of decisions are innocuous in the context of the learning algorithm, or can be corrected by some problem-dependent processing step.

- In Section 4.4.2, it was observed that a near-optimal policy had been obtained from a scenario tree with statistical properties (including first moments) very far from those of the targeted random process. This suggests that the paradigm according to which finding ways to build a unique scenario tree as "close" as possible (in any sense) to the original random process is the more rational objective one could aim at, is in fact too limitative in the context of challenging multistage problems.

  Besides the proposed multi-tree framework based on branching structure randomization, it might be conceivable to perturb the parameters of the targeted random process itself (as long as the learned policies are ultimately tested on the exact random process).

  The objective of the approximate multistage programs could also be perturbed or modified, for instance by adding regularization terms (as long as the learned policies are ultimately tested with the exact cost function).

In Chapter 6, a simple setting was identified in which the complexity of an explicit representation of an optimal solution mapping could be studied, and put in relation with problem data (rather than problem structure). At the time of writing, we are still discovering recent work on explicit representations of solution mappings in slightly more general settings (Patrinos and Sarimveis, 2010), and we may expect that such research directions will continue to be investigated in the future. Nevertheless, it is clear from our experiments that the construction of a purely explicit representation of an exact solution mapping is doomed to fail in some circumstances.

If the hybrid approach that we have proposed in Chapter 6 is shown to be able to extract the computational benefit of explicit solution mappings when it exists, while avoiding the risk of catastrophic performances in terms of computational times in other circumstances, a question not yet addressed in Chapter 6 is the determination of the extent to which an inexact approach, based on supervised learning, and followed by a simple feasibility restoration, allows to overcome the potentially overwhelming complexity of an exact model.

In the pure supervised learning setting, this question has been answered (at least for classification) by the notions of hypothesis space, structural risk minimization, and VC-dimension (Vapnik, 1998). In this thesis, we have not constructed a general theory for predicting the sample complexity (number of scenarios, location of branchings in the trees, number of generated trees) for learning a good enough policy in the context of a chosen hypothesis class (space of policies), but rather have relied on standard model selection principles, and on numerical testing. It could be interesting to see whether supervised learning notions of combinatorial complexity (such as VC-dimension and Rademacher complexity) (Bartlett and Mendelson, 2002) could be adapted more directly to the setting of optimal solution mapping approximation, for instance, referring to Chapter 6 again, by using quantities related to the the smallest positive singular values of matrices of active constraints.

7.3 Influences on Research in Machine Learning

From a general point of view, technical or conceptual advances in stochastic programming techniques can influence specific fields of machine learning. We list below some possibilities that come to mind.

- The investigations on general risk functionals made in the context of multistage stochastic programming can have an impact on the research in reinforcement learning focussed on risk-aware strategies, as confirmed by various recent work (Morimura et al., 2010).
- Decomposition algorithms initially motivated by stochastic programming applications (Rockafellar and Wets, 1991) can be used in the context of supervised learning (Defourny and Wehenkel, 2009). Algorithmic advances in robust stochastic approximation methods (Nemirovski et al., 2009) could also have an impact on supervised learning approaches relying on large-scale optimization techniques.
- The work on scenario tree generation methods is likely to have an impact on optimal experiment design, active learning, and on the direct selection of concise data sets (Rachelson et al., 2010) for reducing, at the source, the complexity of non-parametric models, or for facilitating the processing of data by complex algorithms.

From a point of view more specific to the present work, Chapter 6 suggests possible research directions in supervised learning and artificial intelligence, based on simple settings in which concepts such as “learning to learn” or “learning faster” are given a simple mathematical formalization. It would certainly be worth exploring and developing further such approaches, that could allow to better integrate existing technologies, and focus on context detection, rather than on the learning task itself. Related work in this general orientation includes Ailon et al. (2006) and Hartland et al. (2006).

Appendix A

Elements of Variational Analysis

This appendix presents material from variational analysis (Rockafellar and Wets, 1998) useful in the study of minimization problems subject to constraints, and approximations of those minimization problems.

This material is a part of the fundamental theoretical background supporting many works in stochastic programming, and more generally many work in optimization. It provides a convenient formalism that we use in the thesis for discussing optimization programs abstractly, although we do not insist in the main body of the thesis on some of the technical subtleties highlighted in the present appendix, as these subtleties are not absolutely required to communicate on the kind of work made in the context of the thesis.

The appendix is organized as follows. Section A.1 defines minimization problems through extended-real-valued functions. Section A.2 introduces notations for dealing with sequences, subsequences and neighborhoods. Section A.3 defines the notion of semi-continuity. Section A.4 gives sufficient conditions for the existence of optimal solutions. Section A.5 defines the notion of epigraph. Section A.6 defines the notion of epi-convergence of functions. Section A.7 connects epi-convergence to the property that optimal solutions converge to true optimal solutions. Section A.8 relates epi-convergence to other modes of convergence of functions. Section A.9 consider the generalization of results to parametric optimization. Section A.10 consider the particularization of results to convex optimization. Section A.11 defines the notion of local Lipschitz continuity.

A.1 Minimization

Let $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ denote the set of extended real numbers. Minimization problems and constrained minimization problems can be defined through the notion of extended-real-valued functions.

A.1 Definition. *An* **extended-real-valued function** $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ *assigns to each element* $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ *a value in* $\overline{\mathbb{R}}$*, written* $f(x)$*.*

The **infimum** of f, written $\inf f$, is the greatest lower bound of f, that is, the greatest value $v \in \overline{\mathbb{R}}$ such that $v \le f(x)$ for all $x \in \mathbb{R}^n$. The infimum of f on a (possibly empty) set $C \subset \mathbb{R}^n$, written $\inf_C f$, is the greatest lower bound of the extended-real-valued function that assigns to $x \in C$ the value $f(x)$, and to $x \in \mathbb{R}^n \setminus C$ the value ∞. When $C = \mathbb{R}^n$, $\inf_C f = \inf f$. To emphasize the argument of f, we may write $\inf_x f(x)$ instead of $\inf f$, and $\inf_{x \in C} f(x)$ instead of $\inf_C f$.

Similarly, the **supremum** of f, written $\sup f$, is the least upper bound of f, and the supremum of f on $C \subset \mathbb{R}^n$ is the least upper bound of the function that assigns to $x \in C$ the value $f(x)$, and to $x \in \mathbb{R}^n \setminus C$ the value $-\infty$. We always have $\sup f = -\inf -f$. We have $\inf_C f \leq \sup_C f$ if and only if $C \neq \varnothing$.

If $\inf_C f < \infty$ and there exists some $x \in C$ such that $f(x) = \inf_C f(x)$, we say that the minimum of f on C is attained, we denote $\inf_C f$ by $\min_C f$, and we define the set of minimizers of f over C, written $\operatorname{argmin}_C f$, as the subset of elements of C such that $f(x) = \inf f$. If $\inf_C f < \infty$ but no $x \in C$ satisfies $f(x) = \inf_C f(x)$, we say that the minimum of f on C is not attained, and we set $\operatorname{argmin}_C f$ to the empty set $\varnothing$. If $\inf_C f = \infty$, we set $\operatorname{argmin}_C f = \varnothing$.

Minimizing f on C refers to the task of evaluating $\min_C f$ and finding a point $x \in \operatorname{argmin}_C f$. Very often in applications, f and x have an interpretation, and requirements on x are expressed through inequality constraints $f_j(x) \leq 0$ and equality constraints $h_j(x) = 0$ using functions $f_j : \mathbb{R}^n \to \mathbb{R}$, $1 \leq j \leq p$, and $h_j : \mathbb{R}^n \to \mathbb{R}$, $1 \leq j \leq q$. In that specific case,

$$C = \{x \in \mathbb{R}^n : f_j(x) \leq 0 \text{ for } 1 \leq j \leq p,\ h_j(x) = 0 \text{ for } 1 \leq j \leq q\} \ .$$

As minimizing f on C is equivalent to minimizing the function that coincides with f on C and is set to ∞ on $\mathbb{R}^n \setminus C$, in the sequel we will simply focus on the minimization of f [on $\mathbb{R}^n$], assuming that C is already embedded in f.

Given a nonempty set $X \in \overline{\mathbb{R}}$, we also use the notation $\inf X$ for the greatest $v \in \overline{\mathbb{R}}$ satisfying $v \leq x$ for all $x \in X$. When $v = \inf X$ is in X, we write $v = \min X$. Similarly, $\sup X$ is the least $v \in \overline{\mathbb{R}}$ satisfying $v \geq x$ for all $x \in X$, written $\max X$ when $v \in X$. The case $X = \varnothing$ is handled by setting $\inf \varnothing = \infty$ and $\sup \varnothing = -\infty$.

A.2 Sequences

Let $\mathcal{N}(\bar{x})$ denote the collection of all neighborhoods of $\bar{x} \in \mathbb{R}^n$. We take the notions of open set and neighborhood for granted (Mendelson, 1990). We are about to deal with properties that must hold for all $V \in \mathcal{N}(\bar{x})$. In the metric space $(\mathbb{R}^n, d)$ where $d(x, y) = ||x - y|| = [\sum_{i=1}^{n} (x_i - y_i)^2]^{1/2}$, the properties that we will consider hold for all $V \in \mathcal{N}(\bar{x})$ iff they hold for all open Euclidian balls of rational radius centered at $\bar{x}$, that is, for all V of the form $\{x \in \mathbb{R}^n : ||x - \bar{x}|| < \delta\}$ with $0 < \delta \in \mathbb{Q}$.

Let the topological closure and interior of a set $C \subset \mathbb{R}^n$ be defined by

$$\begin{aligned} \operatorname{cl} C &= \{x \in \mathbb{R}^n : V \cap C \neq \varnothing \text{ for all } V \in \mathcal{N}(x)\} \ , \\ \operatorname{int} C &= \{x \in \mathbb{R}^n : V \subset C \text{ for some } V \in \mathcal{N}(x)\} \end{aligned}$$

(Rockafellar and Wets, 1998, page 14). The topological boundary of a set C is defined by $\operatorname{bdry} C = \operatorname{cl} C \setminus \operatorname{int} C$.

Let $\{x^\nu\}_{\nu \in \mathbb{N}}$ denote a sequence $x_1, x_2, \ldots$ with $x^\nu \in \mathbb{R}^n$ and $\nu \in \mathbb{N}$ (the set of natural numbers, taken as the index set of the elements of the sequence). The set of points x^ν in a sequence $\{x^\nu\}_{\nu \in \mathbb{N}}$ is called the *range* of the sequence (Rudin, 1976, page 48). A sequence is said to be bounded if its range is bounded. In $(\mathbb{R}^n, d)$, a sequence $\{x^\nu\}_{\nu \in \mathbb{N}}$ is said to converge to $\bar{x}$ (or to have $\bar{x}$ as its limit point), written $x^\nu \to \bar{x}$ or $\lim_{\nu\to\infty} d(x^\nu, \bar{x}) = 0$, if for any $\epsilon > 0$, there is some $\nu_0 \in \mathbb{N}$ such that $\nu \geq \nu_0$ entails $d(x^\nu, \bar{x}) < \epsilon$. For instance, the constant sequence with $x^\nu = \bar{x}$ converges to $\bar{x}$. We can also have $x^\nu \to \bar{x}$ despite $x^\nu \neq \bar{x}$ for all ν.

Given a sequence $\{x^\nu\}_{\nu\in\mathbb{N}}$, the sequence $x^{\nu_1}, x^{\nu_2}, \dots$, where $\nu_1, \nu_2, \dots$ is a sequence of positive integers such that $\nu_1 < \nu_2 < \dots$, is called a subsequence of $\{x^\nu\}_{\nu\in\mathbb{N}}$. To facilitate statements involving subsequences, let $\mathcal{N}_\infty$ denote the collection of subsets of $\mathbb{N}$ of the form $\{\nu_0, \nu_0 + 1, \dots\}$, which contain all integers k greater or equal to some positive integer ν_0. Let $\mathcal{N}_\infty^\#$ denote the collection of all subsets of $\mathbb{N}$ of infinite cardinality. Note that $\mathcal{N}_\infty \subset \mathcal{N}_\infty^\#$. Given $N \in \mathcal{N}_\infty$ or $N \in \mathcal{N}_\infty^\#$, we shall write $x^\nu \overset{N}{\to} x$ to indicate that the subsequence $\{x^k\}_{k\in N}$ of the sequence $\{x^\nu\}_{\nu\in\mathbb{N}}$ converges to x. The limit point x of a subsequence $\{x^k\}_{k\in N}$ with $N \in \mathcal{N}_\infty^\#$ is called a cluster point of the sequence $\{x^\nu\}_{\nu\in\mathbb{N}}$. It is also called an accumulation point of the sequence $\{x^\nu\}_{\nu\in\mathbb{N}}$ if $x^\nu \overset{N}{\to} x$ with $x^\nu \neq x$ for all $\nu \in N$. For instance, the sequence $\{(-1)^\nu\}_{\nu\in\mathbb{N}}$ has no limit point, but it has two cluster points -1, $+1$ that are not accumulation points.

In a metric space, it is often illuminating to view a neighborhood $V \in \mathcal{N}(\bar{x})$ as the [interior of the closure of the] union of the ranges from all sequences in V that converge to $\bar{x}$. Such a viewpoint leads to definitions based on sequences — consider, for instance,

$$\begin{aligned}
\operatorname{cl} C &= \{x \in \mathbb{R}^n : \text{there is some sequence } \{x^\nu\}_{\nu\in\mathbb{N}} \\
&\qquad\qquad \text{converging to } x \text{ with } x^\nu \in C \text{ for all } \nu \in \mathbb{N}\} \\
&= \{x \in \mathbb{R}^n : \exists x^\nu \to x \text{ with } x^\nu \in C\} \qquad \text{(brief statement)}, \\
\operatorname{int} C &= \{x \in \mathbb{R}^n : \text{for all } x^\nu \to x,\ \text{there is some } \nu_0 \in \mathbb{N} \\
&\qquad\qquad \text{such that } x^k \in C \text{ for all } k \geq \nu_0\} \\
&= \{x \in \mathbb{R}^n : \forall\, x^\nu \to x,\ \exists\, N \in \mathcal{N}_\infty \text{ such that } x^\nu \in C \text{ for all } \nu \in N\} \ .
\end{aligned}$$

In the sequel, following Rockafellar and Wets (1998), statements are made preferably in terms of sequences.

A.3 Semicontinuity

The following definition of lower and upper limits uses a min/max characterization proved in Rockafellar and Wets (1998, Lemma 1.7).

A.2 Definition. *The lower and upper limits of an extended-real-valued function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ at $\bar{x}$ are values in $\overline{\mathbb{R}}$ defined respectively as*

$$\begin{aligned}
\liminf_{x\to\bar{x}} f(x) &= \sup_{V\in\mathcal{N}(\bar{x})} \inf_{x\in V} f(x) \\
&= \min\{\alpha \in \overline{\mathbb{R}} : \exists\, x^\nu \to \bar{x} \text{ with } f(x^\nu) \to \alpha\} \ , \\
\limsup_{x\to\bar{x}} f(x) &= \inf_{V\in\mathcal{N}(\bar{x})} \sup_{x\in V} f(x) \\
&= \max\{\alpha \in \overline{\mathbb{R}} : \exists\, x^\nu \to \bar{x} \text{ with } f(x^\nu) \to \alpha\} \ .
\end{aligned}$$

The convergence to $\alpha = +\infty$ is interpreted as follows: $f(x^\nu) \to \infty$ $(-\infty)$ if for any $\rho > 0$, there is some $N \in \mathcal{N}_\infty$ such that $\nu \in N$ entails $x^\nu \geq \rho$.

By considering the constant sequence $x^\nu = \bar{x}$ one sees that $\liminf_{x\to\bar{x}} f(x) \leq f(\bar{x})$ and $\limsup_{x\to\bar{x}} f(x) \geq f(\bar{x})$.

A.3 Definition. *A function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is* **lower semicontinuous** *(l.s.c) at a point $\bar{x}$ if $\liminf_{x\to\bar{x}} f(x) \geq f(\bar{x})$, or equivalently $\liminf_{x\to\bar{x}} f(x) = f(\bar{x})$. It is* **upper semicontinuous** *(u.s.c.) at a point $\bar{x}$ if $\limsup_{x\to\bar{x}} f(x) \leq f(\bar{x})$, or equivalently $\limsup_{x\to\bar{x}} f(x) = f(\bar{x})$. The function f is l.s.c (respectively u.s.c.) if it is l.s.c. (respectively u.s.c.) at every point $\bar{x} \in \mathbb{R}^n$.*

Sometimes we speak of functions with properties (such as semicontinuity) *relative to a set* X with X a subset of $\mathbb{R}^n$. This means that we only consider sequences x^ν converging to $\bar{x} \in X$, with $x^\nu \in X$ for all ν. In that case, we write $x^\nu \overset{X}{\to} \bar{x}$, and redefine the liminf and limsup operators as follows.

A.4 Definition. *For properties invoked as relative to X, the limits are taken over sequences in X. In particular, the lower and upper limits of $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ at $\bar{x}$ relative to X become*

$$\liminf_{x \overset{X}{\to} \bar{x}} f(x) = \sup_{V\in\mathcal{N}(\bar{x})} \inf_{x\in V\cap X} f(x) \quad \text{and} \quad \limsup_{x \overset{X}{\to} \bar{x}} f(x) = \inf_{V\in\mathcal{N}(\bar{x})} \sup_{x\in V\cap X} f(x) \ .$$

Among the numerous characterization of continuity, we can thus find the following ones.

A.5 Proposition. *A function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is continuous iff it is both l.s.c. and u.s.c.*

A.6 Proposition. *A function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is continuous relative to X iff $x^\nu \overset{X}{\to} \bar{x}$ entails $f(x^\nu) \to f(\bar{x})$. In particular with $X = \mathbb{R}^n$, f is continuous iff $x^\nu \to \bar{x}$ entails $f(x^\nu) \to f(\bar{x})$.*

A.4 Attainment of a Minimum

Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be an extended-real-valued function. We consider the minimization of f, and in that context, we define the **effective domain** of f as the set

$$\operatorname{dom} f = \{x \in \mathbb{R}^n : f(x) < \infty\} \ .$$

A.7 Definition. *The function f is* **proper** *if f has a nonempty effective domain and is finite-valued there, that is, $f(x) > -\infty$ for all $x \in \operatorname{dom} f$.*

The extended-real-valued functions that coincide with some real-valued function f_0 on a nonempty set C and are equal to ∞ outside C are proper, whereas all other extended-real-valued functions are improper. When f is proper, the corresponding real-valued function f_0 is referred to as the essential objective function.

A.8 Definition. *The function f is* **lower level-bounded** *if for all $\alpha \in \mathbb{R}$, the set $\operatorname{lev}_{\leq\alpha} f = \{x \in \mathbb{R}^n : f(x) \leq \alpha\}$ is bounded.*

For example, the function $x \mapsto x^2$ is lower level-bounded whereas $x \mapsto \exp(x)$ is not. Both functions are continuous on their domain $\mathbb{R}$.

The following theorem (Rockafellar and Wets, 1998, Theorem 1.9) provides sufficient conditions under which an extended-real-valued function attains its minimum.

A.9 Theorem. *If $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is lower semicontinuous, lower level-bounded, and proper, the infimum $\inf f$ is finite and attained on a nonempty compact subset of $\mathbb{R}^n$.*

To handle situations where the evaluation of $\inf f$ and $\operatorname{argmin} f$ has a finite precision, it is useful to consider, when $\inf f$ is finite, the set of ϵ-optimal solutions

$$\epsilon\text{-}\operatorname{argmin} f = \{x \in \mathbb{R}^n : f(x) \leq \inf f + \epsilon\}.$$

As the elements of $\epsilon\text{-}\operatorname{argmin} f$ are themselves evaluated with a finite precision, it is useful to clarify in which sense elements $\tilde{x}$ close to ϵ-optimal solutions are close to being optimal (Rockafellar and Wets, 1998, Theorem 1.43):

A.10 Theorem. *If $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is l.s.c. with $\inf f$ finite, the closed Euclidian ball of radius $\rho > 0$ centered at an ϵ-optimal solution $\bar{x}$ ($\epsilon > 0$) contains a point $\tilde{x}$ which is the unique solution to the minimization of the perturbed function $f(x) + \epsilon\rho^{-1}||x - \tilde{x}||$ and satisfies $f(\tilde{x}) \leq f(\bar{x})$.*

A.5 Epigraph

For a real-valued function $f_0 : \mathbb{R}^n \to \mathbb{R}$, recall that the graph of f_0 is the set $\operatorname{gph} f_0 = \{(x, \alpha) \in \mathbb{R}^n \times \mathbb{R} : \alpha = f_0(x)\}$. For extended-real-valued functions to be minimized, we consider epigraphs.

A.11 Definition. *The* **epigraph** *of an extended-real-valued function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is the subset of $\mathbb{R}^{n+1}$ defined by $\operatorname{epi} f = \{(x, \alpha) \in \mathbb{R}^n \times \mathbb{R} : \alpha \geq f(x)\}$.*

There are correspondences between the properties of an extended-real-valued function and properties of its epigraph.

A.12 Proposition. *For an extended-real-valued function f and its epigraph $\operatorname{epi} f$:*

i. $\operatorname{dom} f = \{x \in \mathbb{R}^n : (x, \alpha) \in \operatorname{epi} f$ *for some* $\alpha \in \mathbb{R}\}$.

ii. *f is proper iff $\operatorname{epi} f \neq \varnothing$ and $\{(x, \alpha) : \alpha \in \mathbb{R}\} \not\subseteq \operatorname{epi} f$ for all fixed $x \in \mathbb{R}^n$.*

iii. $\operatorname{cl}(\operatorname{epi} f) = \{(x, \alpha) \in \mathbb{R}^{n+1} : \alpha \geq \liminf_{x' \to x} f(x')\}$ *(Rockafellar and Wets, 1998, Exercise 1.13(a)).*

iv. *f is lower semicontinuous iff $\operatorname{epi} f$ is closed (Rockafellar and Wets, 1998, Theorem 1.6(b)).*

v. $\operatorname{int}(\operatorname{epi} f) = \{(x, \alpha) \in \mathbb{R}^{n+1} : \alpha > \limsup_{x' \to \bar{x}} f(x')\}$ *(Rockafellar and Wets, 1998, Exercise 1.13(b)).*

A.13 Definition. *The* **lower closure** *of a function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$, denoted $\operatorname{cl} f$, is the function whose epigraph is $\operatorname{cl}(\operatorname{epi} f)$. A function f is said to be* **closed** *when $f = \operatorname{cl} f$. Specifically, $\operatorname{cl} f(x) = \liminf_{x' \to x} f(x')$ (Rockafellar and Wets, 1998, Equation 1(7)).*

We note that in earlier references (Rockafellar, 1970, 1974), a slightly altered definition of the closure of a function was used: the function $\liminf_{x' \to \bar{x}} f(x)$ of definition A.13 was denoted $\operatorname{lsc} f$, and $\operatorname{cl} f$ was set to the constant function $-\infty$ whenever $\liminf_{x' \to x} f(x') = -\infty$ for some x; $\operatorname{cl} f$ was set to $\operatorname{lsc} f$ otherwise.

A.6 Epi-convergence

First we consider notions of limits for sequences of subsets of $\mathbb{R}^n$.

A.14 Definition (Painlevé-Kuratowski convergence of sets). *The* **outer limit** *of a sequence $\{C^\nu\}_{\nu\in\mathbb{N}}$ of subsets $C^\nu \subset \mathbb{R}^n$ is the set of limit points (if any) of all sequences $\{x^\nu\}_{\nu\in N}$ such that $N \in \mathcal{N}_\infty^\#$ and $\varnothing \neq C^\nu \ni x^\nu$ for each $\nu \in N$:*

$$\begin{aligned}\limsup_{\nu\to\infty} C^\nu &= \{x \in \mathbb{R}^n : \exists\, N \in \mathcal{N}_\infty^\#, \exists\, x^\nu \in C^\nu \textit{ for each } \nu \in N,\ \textit{such that } x^\nu \xrightarrow{N} x\} \\ &= \bigcap_{N\in\mathcal{N}_\infty} \operatorname{cl} \bigcup_{\nu\in N} C^\nu \ .\end{aligned}$$

The **inner limit** *of a sequence $\{C^\nu\}_{\nu\in\mathbb{N}}$ of subsets $C^\nu \subset \mathbb{R}^n$ is the set of limit points (if any) of all sequences $\{x^\nu\}_{\nu\in N}$ such that $N \in \mathcal{N}_\infty$ and $\varnothing \neq C^\nu \ni x^\nu$ for each $\nu \in \mathbb{N}$:*

$$\begin{aligned}\liminf_{\nu\to\infty} C^\nu &= \{x \in \mathbb{R}^n : \exists\, N \in \mathcal{N}_\infty, \exists\, x^\nu \in C^\nu \textit{ for each } \nu \in N,\ \textit{such that } x^\nu \xrightarrow{N} x\} \\ &= \bigcap_{N\in\mathcal{N}_\infty^\#} \operatorname{cl} \bigcup_{\nu\in N} C^\nu \ .\end{aligned}$$

From the definition, the inner and outer limits are (possibly empty) closed sets. In particular, for an arbitrary subset V of $\mathbb{R}^n$, the constant sequence with $C^\nu = V$ has $\liminf_{\nu\to\infty} C^\nu = \limsup_{\nu\to\infty} C^\nu = \operatorname{cl}(V)$. We always have the inclusion $\liminf_{\nu\to\infty} C^\nu \subset \limsup_{\nu\to\infty} C^\nu$ since $\mathcal{N}_\infty \subset \mathcal{N}_\infty^\#$. For instance, given two closed subsets $A, B \subset \mathbb{R}^n$, the sequence $\{C^\nu\}_{\nu\in\mathbb{N}}$ with $C^\nu = A$ for ν odd and $C^\nu = B$ for ν even has $\limsup_{\nu\to\infty} C^\nu = A \cup B$ and $\liminf_{\nu\to\infty} C^\nu = A \cap B$.

A.15 Definition. *If $\liminf_{\nu\to\infty} C^\nu = \limsup_{\nu\to\infty} C^\nu = C$, the limit $\lim_\nu C^\nu$ exists and is equal to C. One writes $C^\nu \to C$ to indicate that the sequence $\{C^\nu\}_{\nu\in\mathbb{N}}$ converges to C (in the sense of the Painlevé-Kuratowski convergence of sets).*

Next we consider a sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ of extended-real-valued functions $f^\nu : \mathbb{R}^n \to \overline{\mathbb{R}}$.

A.16 Definition. *The* **lower epi-limit** *of the sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$, denoted* e-lim $\inf_\nu f^\nu$, *is the function defined by identifying its epigraph to the outer limit of the sequence of sets* epi f^ν*:*

$$\begin{aligned}&\operatorname{epi}(\text{e-lim }\inf\nolimits_\nu f^\nu) = \limsup_{\nu\to\infty}(\operatorname{epi} f^\nu) \\ &= \{(x,\alpha) \in \mathbb{R}^{n+1} : \exists\,(x^\nu,\alpha^\nu) \xrightarrow{N} (x,\alpha) \textit{ with } \alpha^\nu \geq f^\nu(x^\nu) \textit{ for some } N \in \mathcal{N}_\infty^\#\} \ .\end{aligned}$$

The **upper epi-limit** *of the sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$, denoted* e-lim $\sup_\nu f^\nu$, *is the function defined by identifying its epigraph to the inner limit of the sequence of sets* epi f^ν*:*

$$\begin{aligned}&\operatorname{epi}(\text{e-lim }\sup\nolimits_\nu f^\nu) = \liminf_{\nu\to\infty}(\operatorname{epi} f^\nu) \\ &= \{(x,\alpha) \in \mathbb{R}^{n+1} : \exists\,(x^\nu,\alpha^\nu) \xrightarrow{N} (x,\alpha) \textit{ with } \alpha^\nu \geq f^\nu(x^\nu) \textit{ for some } N \in \mathcal{N}_\infty\} \ .\end{aligned}$$

The value of the epi-limits at x has the following characterization proved in Rockafellar and Wets (1998, Proposition 7.2). Note that for a sequence $\{y^\nu\}_{\nu\in\mathbb{N}}$ in $\overline{\mathbb{R}}$, the least and greatest cluster points of the sequence are respectively $\liminf_\nu y^\nu = \lim_{\nu\to\infty}[\inf_{\kappa\geq\nu} y^\kappa]$ and $\limsup_\nu y^\nu = \lim_{\nu\to\infty}[\sup_{\kappa\geq\nu} y^\kappa]$.

A.17 Proposition. *For a sequence of functions $f^\nu : \mathbb{R}^n \to \overline{\mathbb{R}}$ and a point $x \in \mathbb{R}$,*

$$(\text{e-lim inf}_\nu\, f^\nu)(x) = \min\{\alpha \in \overline{\mathbb{R}} : \exists\, x^\nu \to x \textit{ with } \liminf_\nu f^\nu(x^\nu) = \alpha\} \ ,$$
$$(\text{e-lim sup}_\nu\, f^\nu)(x) = \min\{\alpha \in \overline{\mathbb{R}} : \exists\, x^\nu \to x \textit{ with } \limsup_\nu f^\nu(x^\nu) = \alpha\} \ .$$

A.18 Definition. *The sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ is said to* **epi-converge** *to f when the lower and upper epi-limits are identical and equal to f. This is written $f^\nu \xrightarrow{e} f$.*

Since by definition the lower and upper limits are closed, the epi-limit f, when it exists, is lower semicontinuous (Rockafellar and Wets, 1998, Proposition 7.4(a)).

A.19 Proposition. *The sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ epi-converges to f iff at each point x, the two following conditions hold (Rockafellar and Wets, 1998, Equation 7(3)):*

i. $\liminf_\nu f^\nu(x^\nu) \geq f(x)$ *for every sequence $x^\nu \to x$,*

ii. $\limsup_\nu f^\nu(x^\nu) \leq f(x)$ *for some sequence $x^\nu \to x$.*

We note the following monotone convergence property (Rockafellar and Wets, 1998, Proposition 7.4(c-d)): if $f^\nu \geq f^{\nu+1}$ for all ν (in the sense that $f^\nu(x) \geq f^{\nu+1}(x)$ for every x and ν), then $f^\nu \xrightarrow{e} \mathrm{cl}[\inf_\nu f^\nu]$. If $f^\nu \leq f^{\nu+1}$ for all ν, then $f^\nu \xrightarrow{e} \sup_\nu[\mathrm{cl}\, f^\nu]$.

A.7 Convergence in Minimization

Consider a sequence of extended-real-valued functions $f^\nu : \mathbb{R}^n \to \overline{\mathbb{R}}$ representing a sequence of minimization problems. Among the several possible notions of convergence according to which we could say that f^ν converges to f, epi-convergence plays a key role for ensuring that the optimal value of f^ν also converges to the optimal value of f as $\nu \to \infty$, and that sequences of minimizers for the functions f^ν have cluster points that are also optimal for f.

A.20 Theorem. *Assume that the sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ epi-converges to a proper function f, with the functions f^ν satisfying the assumptions of Theorem A.9 (f^ν is proper, l.s.c., and level-bounded). Then*

i. The sets $\operatorname{argmin} f^\nu$ *are nonempty and compact (by A.9);*

ii. $\inf f$ *is finite with* $\operatorname{argmin} f$ *nonempty and compact;*

iii. $\inf f^\nu \to \inf f$*;*

iv. The cluster points of sequences $\{x^\nu\}_{\nu\in\mathbb{N}}$ with $x^\nu \in \operatorname{argmin} f^\nu$ are optimal for f:

$$\varnothing \neq \limsup_\nu(\operatorname{argmin} f^\nu) \subset \operatorname{argmin} f \ ;$$

v. If $\operatorname{argmin} f = \{\bar{x}\}$*, any sequence $\{x^\nu\}_{\nu\in\mathbb{N}}$ with $x^\nu \in \operatorname{argmin} f^\nu$ converges to $\bar{x}$.*

In Rockafellar and Wets (1998, Theorem 7.33), the assumptions on the functions f^ν are weakened: the sets $\mathrm{lev}_{\leq\alpha} f^\nu$ have to be bounded for all $\alpha \in \mathbb{R}$ only for ν in some $N \in \mathcal{N}_\infty$, as having $\operatorname{argmin} f^\nu$ nonempty and compact in the tail of the sequence suffices. Also, the sets from which are extracted the solutions x^ν are the sets

$$\epsilon^\nu\text{-}\operatorname{argmin} f^\nu = \{x \in \mathbb{R}^n : f^\nu(x) \leq \inf f^\nu + \epsilon^\nu\}$$

with $\{\epsilon^\nu\}_{\nu\in\mathbb{N}}$ a sequence with $\epsilon^\nu > 0$ decreasing monotonically to 0. In Theorem A.20 it is explicitly assumed that f is lower semicontinuous but actually as an epi-limit, f is necessarily lower semicontinuous (Rockafellar and Wets, 1998, Proposition 7.4(a)).

A.8 Pointwise, Continuous and Uniform Convergence

Pointwise convergence of a sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ concerns the convergence of $f^\nu(x)$ with x fixed, in contrast to epi-convergence concerned with $f^\nu(x^\nu)$.

A.21 Definition. *Let $\{f^\nu\}_{\nu\in\mathbb{N}}$ be a sequence of functions $f^\nu : \mathbb{R}^n \to \overline{\mathbb{R}}$. The sequence is said to converge pointwise on X when $f^\nu(\bar{x}) \to f(\bar{x})$ for every $\bar{x} \in X$.*

The pointwise convergence f^ν to a function f on $\mathbb{R}^n$ means that $\liminf_\nu f^\nu(x) = \limsup_\nu f^\nu(x) = f(x)$ for every $x \in \mathbb{R}^n$.

In contrast to epi-convergence, pointwise convergence does not entail that $\inf f^\nu$ converges to $\inf f$ (Rockafellar and Wets, 1998, Figure 7-1).

A.22 Proposition. *Assume that $\{f^\nu\}_{\nu\in\mathbb{N}}$ converges pointwise to f. Even if f^ν and f satisfy the assumptions of Theorem A.9, we can have $\lim_{\nu\to\infty}(\inf_x f^\nu(x)) \neq \inf_x f(x)$ and $\limsup_\nu(\operatorname{argmin} f^\nu) \cap \operatorname{argmin} f = \varnothing$.*

With the following definition, taken from Rockafellar and Wets (1998, Exercise 7.9), one gets a condition under which pointwise convergence entails epi-convergence (Rockafellar and Wets, 1998, Theorem 7.10). Here $\min\{a, b\}$ and $\max\{a, b\}$ refer to the lowest and highest value between a and b.

A.23 Definition. *A sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ is* **equi-lower semicontinuous** *at $\bar{x}$ relative to $X \subset \mathbb{R}^n$ iff for every $\rho > 0$ and $\epsilon > 0$,*

$$\liminf_{x \xrightarrow{X} \bar{x}} f^\nu(x) \geq \min\{f^\nu(\bar{x}) - \epsilon,\ \rho\} \qquad \textit{for all } \nu \in \mathbb{N}\ .$$

The sequence is **equi-upper semicontinuous** *at $\bar{x}$ relative to X iff for every $\rho > 0$ and $\epsilon > 0$,*

$$\limsup_{x \xrightarrow{X} \bar{x}} f^\nu(x) \leq \max\{f^\nu(\bar{x}) + \epsilon,\ -\rho\} \qquad \textit{for all } \nu \in \mathbb{N}\ .$$

A sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ is **equicontinuous** *at $\bar{x}$ relative to X if it is both equi-l.s.c. and equi-u.s.c. at $\bar{x}$ relative to X. It is equi-l.s.c./u.s.c./continuous relative to X if it has the corresponding property at every $\bar{x} \in X$. It is said to be* **asymptotically** *equi-l.s.c./u.s.c./continuous relative to X if the stated conditions hold for all $\nu \in N$ for some $N \in \mathcal{N}_\infty$.*

A.24 Theorem. *Let $\{f^\nu\}_{\nu\in\mathbb{N}}$ be a sequence of l.s.c. functions $f^\nu : \mathbb{R} \to \overline{\mathbb{R}}$. Assume that the sequence is asymptotically equi-lower semicontinuous. Then f^ν epi-converges to a function f iff f^ν converges pointwise to f.*

At the same time, observe from Proposition A.17 that when f^ν epi-converges to f, there is at least one sequence $x^\nu \to \bar{x}$ such that $f^\nu(x^\nu) \to f(\bar{x})$, whereas for an arbitrary sequence $x^\nu \to \bar{x}$ epi-convergence does not ensure that $f^\nu(x^\nu) \to f(\bar{x})$.

A.25 Definition. *The sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ is said to* **converge continuously** *to f at $\bar{x}$ relative to X iff $x^\nu \xrightarrow{X} \bar{x}$ entails $f^\nu(x^\nu) \to f(\bar{x})$.*

The following theorem is taken from Rockafellar and Wets (1998, Theorem 7.10):

A.26 Theorem. *The sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ converges continuously to f at $\bar{x}$ relative to X iff f^ν epi-converges to f at $\bar{x}$ relative to X and the sequence is asymptotically equicontinuous at $\bar{x}$ relative to X.*

Finally, we define the notion of uniform convergence for a sequence of extended-real-valued functions (Rockafellar and Wets, 1998, Definition 7.12) and its relation with epi-convergence (Rockafellar and Wets, 1998, Proposition 7.15(a)).

A.27 Definition. *The* ***ρ-truncation*** *of an extended-real-valued function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is the real-valued function $f_\rho : \mathbb{R}^n \to \mathbb{R}$ defined by $f_\rho(x) = f(x)$ on $\{x : |f(x)| \leq \rho\}$, $f_\rho(x) = \rho$ on $\{x : f(x) > \rho\}$, $f_\rho(x) = -\rho$ on $\{x : f(x) < -\rho\}$.*

A.28 Definition. *A sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ of real-valued functions $f^\nu : \mathbb{R}^n \to \mathbb{R}$ is said to* **converge uniformly** *to f on $X \subset \mathbb{R}^n$ if for each $\epsilon > 0$, there is some $N \in \mathcal{N}_\infty$ such that $|f^\nu(x) - f(x)| \leq \epsilon$ for every $x \in X$ when $\nu \in N$. A sequence $\{f^\nu\}_{nu\in\mathbb{N}}$ of extended-real-valued functions $f^\nu : \mathbb{R}^n \to \mathbb{R}$ is said to* **converge uniformly** *to f on $X \subset \mathbb{R}^n$ if for each $\rho > 0$, the real-valued ρ-truncations f^ν_ρ converge uniformly to the real-valued ρ-truncation f_ρ.*

A.29 Theorem. *Let $\{f^\nu\}_{\nu\in\mathbb{N}}$ be a sequence of lower semicontinuous functions $f^\nu : \mathbb{R}^n \to \overline{\mathbb{R}}$. If the functions f^ν converge uniformly to a function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ on a set $X \subset \mathbb{R}^n$, then the functions f^ν epi-converge to f relative to X.*

A.9 Parametric Optimization

A minimization problem with n optimization variables and m parameters can be viewed as a single extended-real-valued function $f : \mathbb{R}^n \times \mathbb{R}^m \to \overline{\mathbb{R}}$. Such a function with values $f(x, u)$ induces a function-valued mapping $u \mapsto f(\cdot, u)$.

Parametric optimization is concerned with the variation of $p(u) = \inf_x f(x, u)$ and of $P(u) = \operatorname{argmin}_x f(x, u)$ with u, that is, the characterization of the extended-real-valued function $u \mapsto \inf_x f(x, u)$ and of the set-valued mapping $u \mapsto \operatorname{argmin}_x f(x, u)$.

Sufficient conditions for the attainment of the infimum $\inf_x f(x, u)$ will use the following generalization of Definition A.8 (Rockafellar and Wets, 1998, Definition 1.16).

A.30 Definition. *A function $f : \mathbb{R}^n \times \mathbb{R}^m \to \overline{\mathbb{R}}$ with values $f(x, u)$ is said to be* **(lower) level-bounded in x locally uniformly in u** *if for each $\bar{u} \in \mathbb{R}^m$, the sets*

$$lev_{\leq\alpha}(u) = \{x : f(x, u) \leq \alpha\}$$

are bounded for all $\alpha \in \mathbb{R}$ and for every u in some neighborhood $V \in \mathcal{N}(\bar{u})$ of $\bar{u}$.

The following definition is useful inasmuch as one cannot usually assert that $p(u)$ is continuous in u.

A.31 Definition. *Let p be an arbitrary function from $\mathbb{R}^m$ to $\overline{\mathbb{R}}$. A sequence of points $u^\nu \in \mathbb{R}^m$ is said to* **converge in the p-attentive sense to $\bar{u}$** *if $u^\nu \to \bar{u}$ and $p(u^\nu) \to p(\bar{u})$.*

A.32 Proposition. *For having u^ν converging to $\bar{u}$ in the p-attentive sense, it suffices to have $u^\nu \to \bar{u}$ with p continuous at $\bar{u}$ relative to a set U containing u^ν and $\bar{u}$.*

As with Theorem A.20, there is a useful notion of epi-convergence, now adapted to function-valued mappings. The following definition results from combining Proposition 7.2 and Exercise 7.40 in Rockafellar and Wets (1998).

A.33 Definition. *Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \overline{\mathbb{R}}$ with values $f(x, u)$. The function-valued mapping $u \to f(\cdot, u)$ is said to be* **epi-continuous at $\bar{u}$** *iff for every sequence $u^\nu \to \bar{u}$, the sequence of functions $f^\nu = f(\cdot, u^\nu)$ epi-converges to the function $f(\cdot, \bar{u})$.*

A.34 Theorem. *Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \overline{\mathbb{R}}$ be proper and l.s.c. (on $\mathbb{R}^n \times \mathbb{R}^m$) with $f(x, u)$ level-bounded in x locally uniformly in u. Consider $p(u) = \inf_x f(x, u)$ and $P(u) = \operatorname{argmin}_x f(x, u)$.*

i. The function p is proper and l.s.c. (on $\mathbb{R}^m$).

ii. (Generalization of Theorem A.9.) The set-valued mapping P assigns to each $u \in \operatorname{dom} p$ a nonempty compact set, and is empty-valued for each $u \notin \operatorname{dom} p$.

iii. If a sequence of points $u^\nu \in \operatorname{dom} p$ converges to $\bar{u} \in \operatorname{dom} p$ in the p-attentive sense, then any sequence $\{x^\nu\}_{\nu \in \mathbb{N}}$ with $x^\nu \in P(u^\nu)$ is bounded and has its cluster points in $P(\bar{u})$, that is (given that $P(\bar{u})$ is bounded),

$$\varnothing \neq \limsup_{\substack{u \xrightarrow{\operatorname{dom} p} \bar{u} \\ p(u) \to p(\bar{u})}} P(u) \subset P(\bar{u}).$$

iv. If p is continuous at $\bar{u}$ relative to a set U with $\bar{u} \in U$ and $P(\bar{u}) \neq \varnothing$, then any sequence $\{x^\nu\}_{\nu \in \mathbb{N}}$ with $x^\nu \in P(u^\nu)$ is bounded and has its cluster points in $P(\bar{u})$, that is, $\limsup_{u \xrightarrow{U} \bar{u}} P(u) \subset P(\bar{u})$ with $P(\bar{u}) \subset B$ for some bounded set B.

v. For p to be continuous at $\bar{u}$ relative to U containing $\bar{u}$, a sufficient condition is the existence of some $\bar{x} \in P(\bar{u})$ such that $f(\bar{x}, u)$ is continuous in u at $\bar{u}$ relative to U (Rockafellar and Wets, 1998, Theorem 1.17(c)).

vi. For p to be continuous at $\bar{u}$ relative to U containing $\bar{u}$, another sufficient condition is to have the function-valued mapping $u \mapsto f(\cdot, u)$ epi-continuous at $\bar{u}$ relative to U (Rockafellar and Wets, 1998, Theorem 7.31(b)).

A.10 Convexity

We start with the notion of convexity for subsets of $\mathbb{R}^n$.

A.35 Definition. *A subset C of $\mathbb{R}^n$ is* **convex** *if for all points $x, y \in C$ and for $0 < \lambda < 1$, the points $(1 - \lambda)x + \lambda y$ are in C.*

Convex extended-real-valued functions are defined as follows.

A.36 Definition. *A function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is* **convex** *iff its epigraph* epi *f is convex in $\mathbb{R}^n \times \mathbb{R}$.*

We recall for comparison the definition of convexity for real-valued functions(Rockafellar, 1970, page 10): A function f from a convex set $C \subset \mathbb{R}^n$ to $\mathbb{R}$ is **convex** iff for all points $x, y \in C$ and for $0 < \lambda < 1$, it holds that $f((1 - \lambda)x + \lambda y) \leq (1 - \lambda)f(x) + \lambda f(y)$. Note also that f is said to be **strictly convex** iff for all $x, y \in C$, $x \neq y$, and for $0 < \lambda < 1$, it holds that $f((1 - \lambda)x + \lambda y) < (1 - \lambda)f(x) + \lambda f(y)$.

Convex functions are continuous at least on the interior of their effective domain (Rockafellar and Wets, 1998, Theorem 2.35):

A.37 Proposition. *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be convex. Then f is continuous on* $\operatorname{int}(\operatorname{dom} f)$. *If f is also l.s.c., f is continuous relative to the convex hull of any finite subset of* $\operatorname{dom} f$.

The last implication means that for every integer m and sets $X(m) = \{x \in \mathbb{R}^n : x = \sum_{k=1}^{m} \lambda_k\, x_k$ with $\lambda_k > 0$, $\sum_{k=1}^{m} \lambda_k = 1$, $x_k \in \operatorname{dom} f\}$, it holds that $x^\nu \overset{X(m)}{\to} \bar{x}$ implies $f(x^\nu) \to f(\bar{x})$.

Proposition A.37 also covers the result that lower semicontinuous convex functions whose effective domain is a polytope are continuous on their domain (Gale et al., 1968).

Convex functions *can* have discontinuities on the boundary of their effective domain, *even* if they are also l.s.c.:

A.38 Proposition. *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be convex, proper, and lower semicontinuous.*

i. f may fail to be continuous relative to a compact subset of $\operatorname{dom} f$ *(Rockafellar and Wets, 1998, Example 2.38).*

ii. However, if $n = 1$, $f : \mathbb{R} \to \overline{\mathbb{R}}$ is continuous relative to the closure of its domain (Rockafellar and Wets, 1998, Corollary 2.37)).

A sequence of convex functions can epi-converge under favorable circumstances. The following theorem is taken from Rockafellar and Wets (1998, Theorem 7.17(c)):

A.39 Theorem. *Let $\{f^\nu\}_{\nu \in \mathcal{N}}$ be a sequence of convex functions $f^\nu : \mathbb{R}^n \to \overline{\mathbb{R}}$. If the functions f^ν converge uniformly to f on every compact set that does not meet the boundary of* $\operatorname{dom} f$, *with f convex, l.s.c., and having a nonempty interior, then f^ν epi-converges to f.*

A.11 Lipschitz Continuity

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be Lipschitz continuous if there exists a finite constant $\kappa \geq 0$ such that

$$||f(x) - f(x')|| \leq \kappa ||x' - x|| \quad \text{for all } x, x' \in \mathbb{R}^n \ .$$

For extended-real valued functions, it makes sense to focus on Lipschitz continuity properties locally (Rockafellar and Wets, 1998, page 350).

A.40 Definition. *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be an extended-value function and let X be an open subset of $\mathbb{R}^n$ containing a point $\bar{x}$. Then, the Lipschitz modulus of f at $\bar{x}$ relative to X is defined as the value*

$$lip_X f(\bar{x}) = \limsup_{\substack{x,\, x' \overset{X}{\to} \bar{x} \\ x \neq x'}} \frac{|f(x) - f(x')|}{||x - x'||} \ ,$$

where by convention $|f(x) - f(x')| = \infty$ if $f(x)$ or $f(x')$ (or both) is infinite. The function f is said to be **strictly continuous** *(or* **locally Lipschitz continuous***) at $\bar{x}$ relative to X if $lip_X f(\bar{x})$ is finite, and f is said to be strictly continuous relative to X if it has that property at each $\bar{x} \in X$. The mention to X is omitted when $X = \operatorname{int}\operatorname{dom} f$.*

Appendix B

Basic Probability Theory

This appendix presents standard material from measure and probability theory (Billingsley, 1995; Pollard, 1990).

The definitions and results collected in the present appendix are relevant to this thesis inasmuch as stochastic programming fundamentally deals with randomness. As random objects more general than random vectors are required in the context of stochastic programming, we define them here, using a formalism based on set-valued mappings (see Definition B.8), following Rockafellar and Wets (1998).

The appendix is organized as follows. Section B.1 defines the notions of sigma-algebra and probability space. Section B.2 defines random variables and random vectors. Section B.3 defines random sets. Section B.4 defines random functions. Section B.5 defines the expectation, including the treatment of extended-real-valued functions (Rockafellar and Wets, 1998, Chapter 14). Section B.6 defines distributions and cumulative distribution functions.

B.1 The Probability Space

The probability space is made of three elements: the sample space, the sigma-algebra, and the probability measure.

B.1 Definition. *The* **sample space** Ω *is an arbitrary nonempty set.*

An element of Ω is denoted by ω. Often the sample space is interpreted as an arbitrary space or set of points consisting of all the possible results or outcomes of an experiment (Billingsley, 1995, page 17).

Then, a collection of subsets of Ω is identified for the purpose of performing set operations involving limits along sequences of sets. In essence, admissible collections are those that are closed under countable set operations.

B.2 Definition. *A* **sigma-algebra** $\mathcal{B}$ *of subsets of a space* Ω *is a class of subsets of* Ω *(a collection of sets* $B \subset \Omega$*) such that*

i. The set Ω *belongs to* $\mathcal{B}$*;*

ii. If a set B *is in* $\mathcal{B}$*, then its complement* $B^c = \Omega \setminus B$ *is in* $\mathcal{B}$*;*

iii. For a countable collection $\{B^\nu\}_{\nu\in\mathbb{N}}$ *of sets* B^ν *in* $\mathcal{B}$*, the union* $\bigcup_{\nu=1}^\infty B^\nu$ *is in* $\mathcal{B}$*.*

The empty set $\Omega^c = \varnothing$ and the sample space Ω are always in $\mathcal{B}$ by definition. Countable intersections of sets in $\mathcal{B}$ are also in $\mathcal{B}$, since $(\cup_\nu B_\nu)^c = \cap_\nu (B^\nu)^c$ and $(\cap_\nu B^\nu)^c = \cup_\nu (B^\nu)^c$ (De Morgan's laws of set theory).

Sigma-algebras can be constructed from an initial class $\mathcal{B}_0$ of subsets of interest.

B.3 Definition. *The sigma-algebra* **generated** *by $\mathcal{B}_0$ is the intersection of all the sigma-algebras that contain the class $\mathcal{B}_0$.*

Note that there are often several ways of generating a same sigma-algebra.

Examples of useful sigma-algebras are given below.

- The *trivial sigma-algebra* is the smallest possible sigma-algebra, made of the two sets $\varnothing$ and Ω.
- The *Borel sigma-algebra of the unit interval* $\mathcal{B}((0,1])$ is the sigma-algebra generated by the class of subintervals of $(0,1]$ of the form $I = (a,b]$ with $0 < a < b \le 1$. In fact, the sigma-algebra generated by a countable number of subintervals $(a,b]$ with a, b restricted to rational numbers and $0 < a < b \le 1$ can also be shown to coincide with $\mathcal{B}((0,1])$.
- The *Borel sigma-algebra* $\mathcal{B}(\mathbb{R})$ is the sigma-algebra generated by the class of intervals $I = (a,b]$ of $\mathbb{R}$. It can also be generated by the class of intervals $(-\infty,t]$, $t \in \mathbb{R}$. When we define functions on sigma-algebra that may be valued on the extended real line $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$, we consider the Borel sigma-algebra $\mathcal{B}(\overline{\mathbb{R}})$ generated by the subsets of $\mathcal{B}(\mathbb{R})$ and the two sets $\{-\infty\}$ and $\{+\infty\}$, or alternatively by intervals of the form $(t,+\infty]$, $[-\infty,t)$, $t \in \mathbb{R}$.
- The k-dimensional *Borel sigma-algebra* $\mathcal{B}(\mathbb{R}^k)$ is the sigma-algebra generated by the class of bounded rectangles $\{x = (x_1,\ldots,x_k) \in \mathbb{R}^k : a_i < x_i \le b_i, i = 1,\ldots,k\}$.

In general, Borel sigma-algebras can be generated from all the open subsets of a topological space, or alternatively, from all the closed subsets of the topological space.

The elements of $\mathcal{B}(\mathbb{R}^k)$ are called Borel sets, without mention to $\mathbb{R}^k$ when k is clear from the context.

A set B of a sigma-algebra $\mathcal{B}$ is said to be $\mathcal{B}$-measurable. In the context of probability theory, a set $B \in \mathcal{B}$ is referred to as an *event*.

Probabilities can be assigned to events by the means of a probability measure.

B.4 Definition. *A* **measure** *on a sigma-algebra $\mathcal{B}$ is an extended-real-valued function μ defined on the class $\mathcal{B}$ of subsets of Ω such that*

i. (Nonnegativity.) $0 \le \mu\{B\} \le \infty$ for every $B \in \mathcal{B}$;

ii. $\mu\{\varnothing\} = 0$;

iii. (Countable additivity.) For any countable collection $\{B^\nu\}_{\nu\in\mathbb{N}}$ of sets $B^\nu \in \mathcal{B}$ with $B^i \cap B^j = \varnothing$ if $i \ne j$, $\mu\{\bigcup_{\nu=1}^\infty B^\nu\} = \sum_{\nu=1}^\infty \mu\{B^\nu\}$.

A **probability measure** *$\mathbb{P}$ on a sigma-algebra $\mathcal{B}$ is a measure $\mathbb{P}$ with $\mathbb{P}\{\Omega\} = 1$.*

A **measurable space** is a pair $(\Omega,\mathcal{B})$ with Ω a nonempty set and $\mathcal{B}$ a sigma-algebra on Ω. A **measure space** is a triple $(\Omega,\mathcal{B},\mu)$ with Ω a nonempty set, $\mathcal{B}$ a sigma-algebra on Ω, and μ a measure on $\mathcal{B}$. A **probability space** is a measure space $(\Omega, \mathcal{B}, \mathbb{P})$ with $\mathbb{P}$ a probability measure.

Some sets are special in the context of a given probability space $(\Omega, \mathcal{B}, \mathbb{P})$. A set B is said to be a **support** of $\mathbb{P}$ if B is in $\mathcal{B}$ with $\mathbb{P}\{B\} = 1$ (Billingsley, 1995, page 23). A set B is said to be **$\mathbb{P}$-negligible** if B is in $\mathcal{B}$ and $\mathbb{P}\{B\} = 0$. When a property holds for all ω on a set B with $\mathbb{P}\{B\} = 1$, the property is said to hold with probability 1, or almost surely.

B.5 Definition. *A probability space $(\Omega, \mathcal{B}, \mathbb{P})$ or a sigma-algebra $\mathcal{B}$ is said to be* **complete** *for the probability measure $\mathbb{P}$ if any subset of a $\mathbb{P}$-negligible set is also in $\mathcal{B}$ (and hence $\mathbb{P}$-negligible).*

A measurable space can be made complete relative to a measure $\mathbb{P}$ by enlarging its sigma-algebra (Pollard, 2001, Definition 2.27):

B.6 Definition. *The $\mathbb{P}$-completion of a sigma-algebra $\mathcal{B}$ is the class of sets B for which there exist sets $B_0, B_1 \in \mathcal{B}$ with $B_0 \subset B \subset B_1$ and $\mathbb{P}\{B_1 \setminus B_0\} = 0$.*

Measurable spaces can be combined together.

B.7 Definition. *Let $(\Omega_1, \mathcal{B}_1)$ and $(\Omega_2, \mathcal{B}_2)$ be measurable spaces.*

i. The **product space** *of Ω_1 and Ω_2, denoted $\Omega_1 \times \Omega_2$, is the set of all pairs (ω_1, ω_2) with $\omega_i \in \Omega_i$ $(i = 1, 2)$.*

ii. The **product sigma-algebra** *on $\Omega_1 \times \Omega_2$, denoted $\mathcal{B}_1 \otimes \mathcal{B}_2$, is the sigma-algebra generated by the collection of sets of the form $B_1 \times B_2 = \{(\omega_1, \omega_2) \in \Omega_1 \times \Omega_2 : \omega_i \in B_i \ (i = 1, 2)\}$ with $B_i \in \mathcal{B}_i$ $(i = 1, 2)$.*

B.2 Random Variables

Let us first describe notions relative to set-valued mappings.

B.8 Definition. *A* **set-valued mapping** *$F : X \rightrightarrows Y$ assigns to each element x of X one or more elements of Y, or possibly none. The set of elements $y \in Y$ assigned by F to x is denoted by $F(x)$ (Dontchev and Rockafellar, 2009, page 2, exact citation).*

i. The **domain** *of $F : X \rightrightarrows Y$ is the set $\operatorname{dom} F = \{x \in X : F(x) \neq \varnothing\}$.*

ii. The **range** *of F is the set $\operatorname{range} F = \{y \in Y : y \in F(x) \text{ for some } x \in X\}$.*

iii. The **inverse mapping** *F^{-1} is the set-valued mapping $F^{-1} : Y \rightrightarrows X$ defined by*

$$F^{-1}(y) = \{x \in X : y \in F(x)\} \ .$$

iv. The **image** *of a set $B \subset X$ by F is the set*

$$F(B) = \bigcup_{x \in B} F(x) = \{y \in Y : F^{-1}(y) \cap B \neq \varnothing\} \ .$$

v. The **inverse image** *of a set $C \subset Y$ (or* **pre-image** *of C) by F is the set*

$$F^{-1}(C) = \bigcup_{y \in C} F^{-1}(y) = \{x \in X : F(x) \cap C \neq \varnothing\} \ .$$

There is a one-to-one correspondence between set-valued mappings from X to Y and subsets of $X \times Y$, by identifying a set-valued mapping F to its graph $\operatorname{gph} F = \{(x, y) \in X \times Y : y \in F(x)\}$. Note that the projection $(x, y) \mapsto x$ maps $\operatorname{gph} F$ to $\operatorname{dom} F$, that $(x, y) \mapsto y$ maps $\operatorname{gph} F$ to $\operatorname{range} F$, and that $(x, y) \mapsto (y, x)$ maps $\operatorname{gph} F$ to $\operatorname{gph} F^{-1}$.

In the standard usage, the word **function** designates a relation f that assigns to each element x of a subset $\operatorname{dom} f$ of X, one element y of a subset $\operatorname{range} f$ of Y, written $y = f(x)$, and is undefined on $X \setminus \operatorname{dom} f$. The inverse of a function f is not defined on $Y \setminus \operatorname{range} f$. There is however a one-to-one correspondence between functions and set-valued mappings, in the sense that to each function f can be associated a set-valued mapping F, that is single-valued on $\operatorname{dom} f$ with values $F(x) = \{y\}$, and empty-valued on $X \setminus \operatorname{dom} f$.

In the sequel, we adopt a compromise that consists in calling indifferently **mapping** or **function** the relation $F : X \to Y$ that assigns to each element x of a subset $\operatorname{dom} F \subset X$, one element y of a subset $\operatorname{range} F$ of Y, written $y = F(x)$, and is empty-valued on $X \setminus \operatorname{dom} F$. This is because the functions $F : X \to Y$ considered in the sequel are defined on the full space X anyway. The inverse mapping of F, written F^{-1} is defined on the full space Y as a set-valued mapping.

Let us now consider the probability space $(\Omega, \mathcal{B}, \mathbb{P})$, and a measurable space $(\Omega', \mathcal{C})$. We view a random variable with values in Ω' as a mapping $F : \Omega \to \Omega'$. Mappings of interest are those for which the pre-image of every element of $\mathcal{C}$ is in $\mathcal{B}$.

B.9 Definition. *Let $(\Omega, \mathcal{B})$ and $(\Omega', \mathcal{C})$ be measurable spaces. A mapping $F : \Omega \to \Omega'$ is said to be $\mathcal{B}/\mathcal{C}$-**measurable** if for each $C \in \mathcal{C}$, the pre-image $F^{-1}(C)$ is $\mathcal{B}$-measurable.*

In practice, it is not necessary to check that the pre-image of every element of the sigma-algebra $\mathcal{C}$ is in $\mathcal{B}$: checking the condition for a class of subsets generating the sigma-algebra $\mathcal{C}$ is sufficient (Billingsley, 1995, Theorem 13.1(i)):

B.10 Theorem. *Let $F : \Omega \to \Omega'$ be a mapping with Ω equipped with a sigma-algebra $\mathcal{B}$ and Ω' equipped with a sigma-algebra $\mathcal{C}$. If a class $\mathcal{C}_0$ generates $\mathcal{C}$, and if for every $C_0 \in \mathcal{C}_0$, the inverse image $F^{-1}(C_0)$ is in $\mathcal{B}$, then F is $\mathcal{B}/\mathcal{C}$-measurable.*

When $\Omega' = \mathbb{R}$ with $\mathcal{C}$ the Borel sigma-algebra $\mathcal{B}(\mathbb{R})$, the $\mathcal{B}/\mathcal{C}$ measurable mapping is a real-valued mapping corresponding to a real-valued random variable.

B.11 Definition. *A real-valued random variable f on the probability space $(\Omega, \mathcal{B}, \mathbb{P})$ is a real-valued mapping from Ω to $\mathbb{R}$ that is $\mathcal{B}/\mathcal{B}(\mathbb{R})$-measurable.*

Similarly, extended-real-valued random variables on $(\Omega, \mathcal{B}, \mathcal{P})$ correspond to mappings $f : \Omega \to \overline{\mathbb{R}}$ that are $\mathcal{B}/\mathcal{B}(\overline{\mathbb{R}})$-measurable.

Random variables with values in $\mathbb{R}^k$ are called *random vectors* in Billingsley (1995).

B.12 Definition. *A k-dimensional* **random vector** *f on the probability space $(\Omega, \mathcal{B}, \mathbb{P})$ is a $\mathcal{B}/\mathcal{B}(\mathbb{R}^k)$-measurable mapping from Ω to $\mathbb{R}^k$.*

In fact, a random vector turns out to be simply a k-tuple $f = (f_1, \ldots, f_k)$ of one-dimensional random variables, since it can be shown that the random vector $f(\omega) = (f_1(\omega), \ldots, f_k(\omega))$ is $\mathcal{B}/\mathcal{B}(\mathbb{R}^k)$-measurable if and only if its components $f_i(\omega)$ are $\mathcal{B}/\mathcal{B}(\mathbb{R})$-measurable (Billingsley, 1995, page 183).

Random variables can also be defined as functions of other random variables, not necessarily defined on the same measurable space. For that purpose, the following result is useful (Billingsley, 1995, Theorem 13.1(ii)).

B.13 Theorem. *If $F : \Omega \to \Omega'$ is $\mathcal{B}/\mathcal{B}'$-measurable and $G : \Omega' \to \Omega''$ is $\mathcal{B}'/\mathcal{B}''$-measurable, then the composed mapping $G \circ F$ from Ω to Ω'' is $\mathcal{B}/\mathcal{B}''$-measurable.*

Sometimes one considers the random variables first, and then generates the sigma-algebras in such a way that the random variables of interest are measurable.

B.14 Definition. *The sigma-algebra generated by a collection of random variables is the intersection of all sigma-algebras with respect to which each random variable is measurable.*

That is, the sigma-algebra generated by $f_1,\dots,f_k$, with f_i a mapping from Ω to a space Ω'_i equipped with a sigma-algebra $\mathcal{C}_i$, is defined as the sigma-algebra generated by the class of sets $\{f_i^{-1}(C) : C \in \mathcal{C}_i, i = 1, \dots, k\}$.

Random variables measurable with respect to the sigma-algebra generated by a collection of random variables are equivalent to functions of those random variables (Billingsley, 1995, Theorem 20.1):

B.15 Theorem. *Let $f = (f_1, \dots, f_k)$ be a k-dimensional random vector.*

i. The sigma-algebra generated by $f_1, \dots, f_k$ consists of the sets $\{f \in H\}$ for $H \in \mathcal{B}(\mathbb{R}^k)$.

ii. A random variable h is measurable with respect to the sigma-algebra generated by $f_1, \dots, f_k$ iff there exists a measurable mapping g from $\mathbb{R}^k$ to $\mathbb{R}$ such that for all ω, $h(\omega) = g(f_1(\omega), \dots, f_k(\omega))$.

(For brevity, we also allow ourselves to say that h is measurable with respect to $f_1, \dots, f_k$ when h is measurable with respect to the sigma-algebra generated by $f_1, \dots, f_k$.)

Consider again the collection of random variables $f_1, \dots, f_k$ where f_i is a $\mathcal{B}/\mathcal{C}_i$-measurable mapping from Ω to C_i. Let $\mathcal{F}_0$ denote the trivial sigma-algebra, and let $\mathcal{F}_i$ denote the sigma-algebras generated by the subcollection $\{f_1, \dots, f_i\}$ of random variables. By definition, it holds that $\mathcal{F}_i \subseteq \mathcal{F}_j$ for $0 \leq i < j \leq k$.

B.16 Definition. *The* **natural filtration** *associated to a sequence $f_1, \dots, f_k$ of random variables is the family $\{\mathcal{F}_i : i = 0, \dots, k\}$ of sub-sigma-algebras of $\mathcal{B}$ generated by the growing subcollections $\{f_1, \dots, f_i\}$ of random variables.*

The natural filtration represents the growing information on ω obtained by considering growing collections of $\mathcal{B}$-measurable random variables. Note from Theorem B.15 that adding to a collection $f_1, \dots, f_i$ a random variable f_{i+1} which is merely a function of $f_1, \dots, f_i$ will not alter the sub-sigma-algebra generated by the collection, that is, will not refine the available information on ω.

B.3 Random Sets

Random sets are defined as measurable set-valued mappings. We will only consider mappings to closed subsets of $\mathbb{R}^n$.

B.17 Definition. *Let $(\Omega, \mathcal{B})$ be a measurable space. A* **set-valued mapping** $F : \Omega \rightrightarrows \mathbb{R}^n$ *is* **measurable** *if for every open set $C \subset \mathbb{R}^n$, the pre-image $F^{-1}(C)$ is in $\mathcal{B}$. If F is closed-valued (the sets $F(x)$ are closed), an equivalent measurability condition is $F^{-1}(C) \in \mathcal{B}$ for every closed set C (Rockafellar and Wets, 1998, Theorem 14.3(b)).*

The purpose of a measurable selection, defined below, is to reduce a measurable set-valued mapping to a measurable (single-valued) mapping. This is useful, for instance, for selecting a single optimal solution from a set of optimal solutions to a parametric optimization program. Other examples of selections that are frequently met in practice are the particular choices of matrix pseudo-inverses for returning a single solution to an underdetermined system of equations. For a particular example in the thesis, see Example 6.1.

We use the following non-standard definition of the measurable selection to avoid issues posed by set-valued mappings that are empty-valued in some regions of the sample space (see Remark B.1 below).

B.18 Definition. *Given a measurable set-valued mapping $F : \Omega \rightrightarrows \mathbb{R}^n$, a* **measurable selection** *for F is a measurable set-valued mapping $f : \Omega \rightrightarrows \mathbb{R}^n$ which is single-valued with values $f(\omega) \in F(\omega)$ whenever $F(\omega)$ is nonempty, and empty-valued whenever $F(\omega) = \varnothing$.*

If $F(\omega)$ is nonempty for all $\omega \in \Omega$, the measurable selection can be defined more simply as a measurable function $f : \Omega \to \mathbb{R}^n$ with values $f(\omega) \in F(\omega)$ for all $\omega \in \Omega$.

A measurable closed-valued mapping always admits a measurable selection (Rockafellar and Wets, 1998, Corollary 14.6).

Remark B.1. It is not clear to us whether the empty set is considered in Rockafellar and Wets as an admissible value for a closed-valued mapping, and how the selection defined as a function can handle that case. The definition of the selection has been changed in Dontchev and Rockafellar (2009, page 49) to allow for a local definition, but a local definition on the subset Ω_0 of Ω where F is not empty-valued is not desirable for a measurable selection, which should be defined on the full sample space Ω. Aubin and Frankowska (1990, Theorem 8.1.3) avoid the issue by dealing only with non-empty-closed-valued measurable mappings F, but this choice rules out the use of a measurable selection for selecting an optimal solution to a parametric optimization program which is infeasible in some region of the parameter space. □

There is a correspondence between the measurability of mappings and the measurability of their associated graph (Rockafellar and Wets, 1998, Theorem 14.8).

B.19 Theorem. *Let $(\Omega, \mathcal{B}, \mathbb{P})$ be a probability space. Let $F : \Omega \rightrightarrows \mathbb{R}^n$ be closed-valued. If the probability space is complete, the 3 following properties are equivalent.*

i. *The set-valued mapping F is measurable;*

ii. *For any set $C \in \mathcal{B}(\mathbb{R}^n)$, the pre-image $F^{-1}(C)$ is in $\mathcal{B}$;*

iii. $\operatorname{gph} F$ *is a $\mathcal{B} \otimes \mathcal{B}(\mathbb{R}^n)$-measurable subset of $\Omega \times \mathbb{R}^n$.*

B.4 Random Functions

Random functions can be interpreted as mappings from $\Omega \times \mathbb{R}^n$ to $\overline{\mathbb{R}}$: for every $\omega \in \Omega$, there is a function $f(\omega, \cdot)$ from $\mathbb{R}^n$ to $\overline{\mathbb{R}}$. They can also be defined using their epigraph representation: for every $\omega \in \Omega$, there is a subset $\operatorname{epi} f(\omega, \cdot)$ of $\mathbb{R}^{n+1}$. In the sequel, we will only consider l.s.c. random functions. Recall that the epigraph of a l.s.c. function is a closed set (Proposition A.12).

If x is a random variable with values $x(\omega)$ and f is a random l.s.c. function with "values" $f(\omega, \cdot)$, where $f(\omega, \cdot)$ is a l.s.c. function, then $f(\cdot, x(\cdot))$ defines a random variable with values $f(\omega, x(\omega))$ (a measurable mapping from Ω to $\overline{\mathbb{R}}$) if f satisfies suitable measurability conditions. These conditions are given below: f has to be a normal integrand (Rockafellar and Wets, 1998, Definition 14.27).

B.20 Definition. *Let $(\Omega, \mathcal{B}, \mathbb{P})$ be a probability space. Let $f : \Omega \times \mathbb{R}^n \to \overline{\mathbb{R}}$ be a random function with associated domain and epigraph*

$$\begin{aligned} D_f(\omega) &= \operatorname{dom} f(\omega, \cdot) = \{x \in \mathbb{R}^n : f(\omega, x) < \infty\} \\ S_f(\omega) &= \operatorname{epi} f(\omega, \cdot) = \{(x, \alpha) \in \mathbb{R}^n \times \mathbb{R} : f(\omega, x) \le \alpha\} \ . \end{aligned}$$

If $S_f : \Omega \rightrightarrows \mathbb{R}^n \times \mathbb{R}$ is closed-valued and measurable (as a set-valued mapping defined on $\Omega \times \mathbb{R}^n$), the function f is said to be a **normal integrand** *(Rockafellar and Wets, 1998, Definition 14.27).*

The following results are taken from Rockafellar and Wets (1998, Propositions 14.28, Theorem 14.37).

B.21 Proposition. *Let $(\Omega, \mathcal{B}, \mathbb{P})$ be a probability space and $f : \Omega \times \mathbb{R}^n \to \overline{\mathbb{R}}$ a random function with domain D_f and epigraph S_f. If f is a normal integrand, then $D_f : \Omega \rightrightarrows \mathbb{R}^n$ is measurable (as a set-valued mapping), f is l.s.c. in $x \in \mathbb{R}^n$ for each fixed $\omega \in \Omega$, and f is $\mathcal{B}$-measurable in $\omega \in \Omega$ for each fixed $x \in \mathbb{R}^n$. In addition, the random variable defined by $\omega \mapsto f(\omega, x(\omega))$, where x is $\mathcal{B}/\mathcal{B}(\mathbb{R}^n)$-measurable, is itself $\mathcal{B}/\mathcal{B}(\overline{\mathbb{R}})$-measurable.*

B.22 Theorem. *Let $f : \Omega \times \mathbb{R}^n \to \overline{\mathbb{R}}$ be a normal integrand. Then for $p(\omega) = \inf f(\omega, \cdot)$ and $P(\omega) = \operatorname{argmin} f(\omega, \cdot)$, it holds that the function $p : \Omega \to \overline{\mathbb{R}}$ is measurable, the mapping $P : \Omega \rightrightarrows \mathbb{R}^n$ is closed-valued and measurable, and in particular P admits a measurable selection.*

Examples of normal integrands are recorded below (Rockafellar and Wets, 1998, Examples 14.29, 14.31, 14.32; Proposition 14.39; Exercise 14.55), whereas Theorem B.19 gives the general measurability condition for set-valued mappings that characterizes normal integrands.

B.23 Proposition. *Let $f : \Omega \times \mathbb{R}^n \to \overline{\mathbb{R}}$ be a random function with domain $D_f(\omega)$. If any of the following conditions hold, f is a normal integrand.*

i. (Carathéodory integrands.) $f(\omega, x)$ is finite-valued, measurable in ω for each x, and continuous in x for each ω.

ii. There is a Carathéodory integrand $f_0 : \mathbb{R}^n \to \mathbb{R}$ and a closed-valued measurable mapping $C : \Omega \rightrightarrows \mathbb{R}^n$ such that $f(\omega, x)$ coincides with $f_0(\omega, x)$ if $x \in C(\omega)$ and $f(\omega, x) = \infty$ if $x \notin C(\omega)$.

iii. *(Jointly l.s.c. functions.) Ω is a Borel subset of $\mathbb{R}^d$ and f is l.s.c. (over $\Omega \times \mathbb{R}^n$).*

iv. *(Convex integrands.) $f(\omega, \cdot)$ is l.s.c. and convex (over $\mathbb{R}^n$) for each ω, the interior of $D_f(\omega)$ is nonempty whenever $D_f(\omega)$ is nonempty, and $f(\omega, x)$ is measurable in ω for each x.*

v. *(Simple integrands.) The range of D_f and the range of f are finite (this holds true in situations where the set of feasible x given ω is finite and Ω is finite).*

B.5 Expectation

Let $(\Omega, \mathcal{B}, \mu)$ be a measure space. Let $\mathcal{M}$ denote the class of all $\mathcal{B}/\mathcal{B}(\overline{\mathbb{R}})$-measurable mappings from Ω to $\overline{\mathbb{R}}$, and let $\mathcal{M}^+$ denote the class of all nonnegative mappings in $\mathcal{M}$.

The expectation (or expected value) of nonnegative random variables is defined through the integral (Billingsley, 1995, Equation 15.3).

B.24 Definition. *The* **integral** *of a function $f \in \mathcal{M}^+$ on a measure space $(\Omega, \mathcal{B}, \mu)$ is*

$$\int f \, d\mu = \sup \sum_{\nu} \left[\inf_{\omega \in B^\nu} f(\omega) \right] \mu(B^\nu)$$

where the supremum is over the partitions $\{B^\nu\}_{1 \le \nu \le N}$ of Ω with N finite and $B^\nu \in \mathcal{B}$. The **expectation** *of a random variable $f \in \mathcal{M}^+$ on a probability space $(\Omega, \mathcal{B}, \mathbb{P})$ is (setting μ to $\mathbb{P}$)*

$$\mathbb{E}\{f\} = \int f \, d\mathbb{P} \ .$$

The expectation of nonnegative random variables can also be defined through properties (Pollard, 2001, Theorem 2.12). For a set $B \in \mathcal{B}$, let $I_B \in \mathcal{M}^+$ denote the indicator function of B defined by $I_B(\omega) = 1$ if $\omega \in B$ and $I_B(\omega) = 0$ if $\omega \notin B$.

B.25 Definition. *For each probability measure $\mathbb{P}$ on the measurable space $(\Omega, \mathcal{B})$, there is a functional $\mathbb{E}$ from $\mathcal{M}^+$ to $[0, \infty]$ uniquely determined by the following properties.*

i. *$\mathbb{E}\{I_B\} = \mathbb{P}\{B\}$ for all $B \in \mathcal{B}$;*

ii. *$\mathbb{E}\{0\} = 0$ where the zero of the left-hand side denotes a zero-valued measurable mapping;*

iii. *For $\alpha, \beta \ge 0$ and $f, g \in \mathcal{M}^+$, $\mathbb{E}\{\alpha f + \beta g\} = \alpha\mathbb{E}\{f\} + \beta\mathbb{E}\{g\}$;*

iv. *If f, g are in $\mathcal{M}^+$ and $f(\omega) \le g(\omega)$ for almost all $\omega \in \Omega$, then $\mathbb{E}\{f\} \le \mathbb{E}\{g\}$;*

v. *(Monotone convergence.) For a sequence $\{f^\nu\}_{\nu \in \mathbb{N}}$ of functions $f^\nu \in \mathcal{M}^+$, if $f^\nu(\omega) \to f(\omega)$ with $f^\nu(\omega) \le f^{\nu+1}(\omega)$ for almost all $\omega \in \Omega$, then $\mathbb{E}\{f^\nu\} \to \mathbb{E}\{f\}$ with $\mathbb{E}\{f^\nu\} \le \mathbb{E}\{f^{\nu+1}\}$.*

The expectation of an extended-real-valued random variable f is obtained by decomposing f into its positive and negative parts $f^+, f^- \in \mathcal{M}^+$.

B.26 Definition. *Let $f : \Omega \to \overline{\mathbb{R}}$ be an extended-real-valued random variable. Define $f^+(\omega) = \max\{0, f(\omega)\}$ and $f^-(\omega) = \max\{0, -f(\omega)\}$. If $\mathbb{E}\{f^+\}$ and $\mathbb{E}\{f^-\}$ are not both infinite, the* **expectation** *of f is said to be well-defined, and*

$$\mathbb{E}\{f\} = \mathbb{E}\{f^+\} - \mathbb{E}\{f^-\} \ .$$

$\mathbb{E}\{\alpha f+\beta g\} = \alpha\mathbb{E}\{f\}+\beta\mathbb{E}\{g\}$ for α, β in $\mathbb{R}$ provided that the situation $\infty-\infty$ is avoided.

The expectation has additional properties.

B.27 Proposition. *Let $\{f^\nu\}_{\nu\in\mathbb{N}}$ be a sequence of functions in $\mathcal{M}$.*

vi. (Fatou's lemma.) $\mathbb{E}\{\liminf_\nu f^\nu\} \leq \liminf_\nu \mathbb{E}\{f^\nu\}$.

vii. (Dominated convergence.) If $f^\nu(\omega) \to f(\omega)$ for almost all ω and if there is some $g \in \mathcal{M}$ such that $|f^\nu(\omega)| \leq g(\omega)$ for all ν and almost all ω with $\mathbb{E}\{g\} < \infty$, then $\mathbb{E}\{f^\nu\} \to \mathbb{E}\{f\}$ with $\mathbb{E}\{f^\nu\}$ finite and $\mathbb{E}\{f\}$ finite.

viii. (Uniform integrability.) If $f^\nu(\omega) \to f(\omega)$ for almost all ω and if the sequence is uniformly integrable, in the sense that $\sup_\nu \mathbb{E}\{|f^\nu|I_{\{|f^\nu|>\alpha\}}\}$ tends to 0 as $\alpha \to \infty$, then $\mathbb{E}\{f^\nu\} \to \mathbb{E}\{f\}$.

Classes of random variables with finite expectations define particular spaces of measurable functions (Pollard, 2001, Section 2.7).

B.28 Definition. *Let $(\Omega, \mathcal{B}, \mathbb{P})$ be a probability space. For $1 \leq p < \infty$, consider the space $\mathcal{L}^p(\Omega, \mathcal{B}, \mathbb{P})$ of functions $f \in \mathcal{M}$ such that $\mathbb{E}\{|f|^p\}$ is finite. For $p = \infty$, consider the space $\mathcal{L}^\infty(\Omega, \mathcal{B}, \mathbb{P})$ of functions $f \in \mathcal{M}$ for which the essential supremum $\inf[\alpha \in \overline{\mathbb{R}} : \mathbb{P}\{\omega : |f(\omega)| > \alpha\} = 0\}$ is finite. Then, the* **Lebesgue space** *$L^p(\Omega, \mathcal{B}, \mathbb{P})$ $(1 \leq p \leq \infty)$ is defined as the space of equivalence classes of functions $[f] = \{g \in \mathcal{L}^p(\Omega, \mathcal{B}, \mathbb{P}) : g = f\mathbb{P}$-almost surely$\}$.*

To each element f of the space $\mathcal{L}^p(\Omega, \mathcal{B}, \mathbb{P})$ can be associated the real number $||f||_p = (\mathbb{E}\{|f|^p\})^{1/p}$. The reduction to equivalence classes of functions is made so that in $L^p(\Omega, \mathcal{B}, \mathbb{P})$, $||f-g||_p = 0$ entails $f = g$. ($||\cdot||_p$ is a semi-norm for $\mathcal{L}^p(\Omega, \mathcal{B}, \mathbb{P})$ and a norm for $L^p(\Omega, \mathcal{B}, \mathbb{P})$: see Definition C.3.)

B.29 Definition. *Let $\mathcal{L}^p(\Omega, \mathcal{B}, \mathbb{P}; \mathbb{R}^n)$ be the space of $\mathcal{B}$-measurable mappings $x : \Omega \to \mathbb{R}^n$ such that the Euclidian norm mapping $\omega \mapsto ||x(\omega)||$ is in $\mathcal{L}^p(\Omega, \mathcal{B}, \mathbb{P})$. Then, the* **Lebesgue space** *$L^p(\Omega, \mathcal{B}, \mathbb{P}; \mathbb{R}^n)$ $(1 \leq p \leq \infty)$ is defined as the space of equivalence classes of functions $[f] = \{g \in \mathcal{L}^p(\Omega, \mathcal{B}, \mathbb{P}; \mathbb{R}^n) : g = f$ $\mathbb{P}$-almost surely$\}$.*

Now we turn our attention to expectations over random functions. The expectation over a random function is well-defined for normal integrands (Rockafellar and Wets, 1998, Proposition 14.58):

B.30 Proposition. *Let $(\Omega, \mathcal{B}, \mathbb{P})$ be a probability space. Let $\mathcal{X}$ denote a space of $\mathcal{B}/\mathcal{B}(\mathbb{R}^n)$-measurable mappings, and let $x : \Omega \to \mathbb{R}^n$ be a mapping in $\mathcal{X}$. If the random function $f : \Omega \times \mathbb{R}^n \to \overline{\mathbb{R}}$ is a normal integrand, the functional E_f from $\mathcal{X}$ to $\overline{\mathbb{R}}$ given by*

$$E_f[x] = \mathbb{E}\{f(\omega, x(\omega))\}$$

is well-defined, under the additional convention (in a context of minimization) that $\mathbb{E}\{f(\omega, x(\omega))\} = \infty$ if $\mathbb{E}\{f^+(\omega, x(\omega))\} = \infty$. When $E_f[x]$ is finite, it holds that $x(\omega)$ lies in $\operatorname{dom} f(\omega, \cdot)$ almost surely.

An important theorem identifies conditions under which the infinite-dimensional minimization of $E_f[x]$ over $x \in \mathcal{X}$ reduces to a minimization of $f(\omega, \cdot)$ for each ω (Rockafellar and Wets, 1998, Theorem 14.60). The result makes use of a property possessed by certain spaces $\mathcal{X}$ of measurable functions (Rockafellar and Wets, 1998, Definition 14.59).

B.31 Definition. *A space $\mathcal{X}$ of measurable functions $x : \Omega \to \mathbb{R}^n$ is* **decomposable** *relative to a measure μ if for every function $x_0 \in \mathcal{X}$, every set $B \in \mathcal{B}$ with $\mu(B)$ finite and every bounded measurable function $x_1 : \Omega \to \mathbb{R}^n$, $\mathcal{X}$ contains the function x that coincides with x_0 on $\Omega \setminus B$ and coincides with x_1 on B.*

The Lebesgue spaces $L^p(\Omega, \mathcal{B}, \mathbb{P}; \mathbb{R}^n)$ are decomposable, whereas the space of constant-valued functions and the space of continuous mappings $f : \Omega \to \mathbb{R}^n$ are not decomposable relative to most measures $\mathbb{P}$ (Rockafellar and Wets, 1998, page 677).

B.32 Theorem. *Let $(\Omega, \mathcal{B}, \mathbb{P})$ be a probability space. Let $\mathcal{X}$ be a space of measurable functions $x : \Omega \to \mathbb{R}^n$ decomposable relative to $\mathbb{P}$. Let $f : \Omega \times \mathbb{R}^n \to \overline{\mathbb{R}}$ be a normal integrand. Then, as long as $E_f[x] \not\equiv \infty$,*

$$\inf_{x \in \mathcal{X}} \mathbb{E}\{f(\omega, x(\omega))\} = \mathbb{E}\{\inf_{x \in \mathbb{R}^n} f(\omega, x)\} \ ,$$

and as long as $\inf_{x \in \mathcal{X}} \mathbb{E}\{f(\omega, x(\omega))\} > -\infty$, it holds that $\bar{x} \in \mathcal{X}$ is in $\operatorname{argmin}_{x \in \mathcal{X}} E_f[x]$ iff $\bar{x}(\omega)$ is in $\operatorname{argmin}_{x \in \mathbb{R}^n} f(\omega, x)$ for $\mathbb{P}$-almost every $\omega \in \Omega$.

B.6 Distributions

Let $(\Omega, \mathcal{B}, \mathbb{P})$ be a probability space and let $x : \Omega \to \mathbb{R}^m$ be a $\mathcal{B}/\mathcal{B}(\mathbb{R}^m)$-measurable mapping.

B.33 Definition. *The* **distribution** *of a random vector $x : \Omega \to \mathbb{R}^m$ is the mapping μ from the Borel sets $B \in \mathcal{B}(\mathbb{R}^m)$ to the interval $[0, 1]$, with values*

$$\mu(B) = \mathbb{P}\{x(\omega) \in B\} \ .$$

The **support** *of the distribution of x, also referred to as the support of x, is defined as the smallest closed set B (with respect to the set-inclusion ordering) such that $\mu(B) = 1$.*

The **cumulative distribution function** *(cdf) of x is the mapping $F : \mathbb{R}^m \to \mathbb{R}$ with values*

$$F(t) = \mathbb{P}\{x_i(\omega) \le t_i \ , \ i = 1, \dots, m\} \ .$$

For a real-valued random variable $x : \Omega \to \mathbb{R}$, the corresponding distribution function F has an integral representation

$$F(t) = \textstyle\int_{-\infty}^{t} f(t) \mathrm{d}t$$

if and only if F is absolutely continuous (Billingsley, 1995, Theorem 31.8), in the following sense (Billingsley, 1995, Equation 31.28):

B.34 Definition. *A function $F : \mathbb{R} \to \mathbb{R}$ is absolutely continuous if for each $\epsilon > 0$, there is some $\delta > 0$ such that the following condition holds: for each collection of k intervals $[a_i, b_i]$ with a disjoint interior,*

$$\sum_{i=1}^{k} |F(b_i) - F(a_i)| < \epsilon \quad \text{if} \ \sum_{i=1}^{k} (b_i - a_i) < \delta \ .$$

Appendix C

Elements of Functional Analysis for Kernel Methods

This appendix presents results from functional analysis useful in the theory of kernel methods. We use kernels or kernel-based methods in several places in the thesis (Chapters 3, 5).

The appendix is organized as follows. Section C.1 defines Hilbert spaces. Section C.2 defines continuous linear mappings. Section C.3 defines reproducing kernels, positive definite kernels, and reproducing kernel Hilbert spaces. Section C.4 gives the interpretation of positive definite kernels as generalized inner products.

C.1 Hilbert Spaces

C.1 Definition. *Let F be a nonempty set. A* **metric** *for F is a function $d : F \times F \to \mathbb{R}$ with the following properties (where $f, g, h \in F$):*

i. $d(f, g) \geq 0$ *with* $d(f, g) = 0$ *iff* $f = g$

ii. $d(f, g) = d(g, f)$

iii. (Triangle inequality.) $d(f, h) \leq d(f, g) + d(g, h)$.

A **metric space** (F, d) *is defined as a nonempty set F equipped with a metric d.*

In a metric space (F, d), the distance from an element $f \in F$ to a set $C \subset F$ is given by $d(f, C) = \inf_{g \in C} d(f, g)$, with $d(f, \varnothing) = \infty$. We say that a metric space (F, d) is **separable** if F has a dense countable subset, in the sense that there exists a set $Q = \{q^\nu\}_{\nu \in \mathbb{N}}$ of elements of F such that for all $\epsilon > 0$ and $f \in F$, $d(f, Q) < \epsilon$.

A sequence $\{f^\nu\}_{\nu \in \mathbb{N}}$ in a metric space (F, d) is a **Cauchy sequence** if for each $\epsilon > 0$, there is some $N_\epsilon \in \mathcal{N}_\infty$ such that $d(f^\mu, f^\nu) < \epsilon$ when $\mu, \nu \in N_\epsilon$. We say that $\{f^\nu\}_{\nu \in \mathbb{N}}$ **converges** [**strongly**] to some limit point f if $\lim_{\nu \to \infty} d(f^\nu, f) = 0$. We denote the limit by $\text{s-lim}_{\nu \to \infty} f^\nu = f$ and write $f^\nu \to f$.

C.2 Definition. *A metric space (F, d) is* **complete** *if every Cauchy sequence in it converges to an element of F.*

If (F, d) is a complete metric space, then a set C is closed when $d(f, C) = 0$ entails $f \in C$.

Recall that a linear space F over a field $\mathbb{K}$ is a set F for which the addition $(+)$ of two elements $f, g \in F$ and the multiplication $(\cdot)$ of an element $f \in X$ by a scalar $\alpha \in \mathbb{K}$ obey the standard rules of algebra (commutativity, associativity, distributivity), with $f + g$ and $\alpha \cdot f$ being also elements of F (Yosida, 1980, Section 0.4). For the properties of fields we refer to Rudin (1976, Definition 1.12). A linear space is called a real linear space if $\mathbb{K} = \mathbb{R}$. A linear space is called a complex linear space if $\mathbb{K} = \mathbb{C}$.

C.3 Definition. *A linear space F over a field $\mathbb{K}$ is a* **normed linear space** *if to every element $f \in F$ is associated a real number $||f||$, called the norm of f, with the following properties (where $f, g \in F$ and $\alpha \in \mathbb{K}$):*

i. $||f|| \geq 0$ *with* $||f|| = 0$ *iff* $f = 0$

ii. (Subadditivity.) $||f + g|| \leq ||f|| + ||g||$

iii. $||\alpha f|| = |\alpha| \cdot ||f||$.

A normed linear space is a **pre-Hilbert space** *if its norm also satisfies*

iv. $||f + g||^2 + ||f - g||^2 = 2(||f||^2 + ||g||^2)$.

In a normed linear space F, the function $d(f, g) = ||f - g||$ is a metric for F.

In Definition C.3, if $||f||$ satisfies only the conditions *ii* and *iii* (which imply $||f|| \geq 0$), then $||f||$ is called a semi-norm. If $||f||$ satisfies conditions *i*, *ii*, and instead of condition *iii* the weaker set of conditions

iii'. $||-f|| = ||f||$,
$\alpha^\nu \to 0$ entails $||\alpha^\nu f|| \to 0$,
$||f^\nu|| \to 0$ entails $||\alpha f^\nu|| \to 0$,

then $||f||$ is called a quasi-norm, and F is called a quasi-normed linear space. When F is a quasi-normed or a normed linear space, $f^\nu \to f$ entails $||f^\nu|| \to ||f||$; furthermore, if $f^\nu \to f$, $g^\nu \to g$, and $\alpha^\nu \to \alpha$, it holds that $f^\nu + g^\nu \to f + g$ and $\alpha^\nu f^\nu \to \alpha f$ (Yosida, 1980, Proposition 2.2).

C.4 Proposition. *Let F be a real pre-Hilbert space. The* **inner product** *between $f, g \in F$, is defined by*

$$\langle f, g\rangle = \tfrac{1}{4}||f + g||^2 - \tfrac{1}{4}||f - g||^2 ,$$

and satisfies the following properties (where $f, g, h \in F$ and $\alpha \in \mathbb{R}$):
$\langle \alpha f, g\rangle = \alpha\langle f, g\rangle$; $\langle f + g, h\rangle = \langle f, h\rangle + \langle g, h\rangle$; $\langle f, g\rangle = \langle g, f\rangle$; $\langle f, f\rangle = ||f||^2$.
Moreover, we have $|\langle f, g\rangle| \leq ||f||\,||g||$ (Cauchy-Schwartz inequality).

If F is a complex pre-Hilbert space, the inner product is defined as $\langle f, g\rangle = (f, g) + j(f, j\,g)$ with $(f, g) = ||f + g||^2/4 - ||f - g||^2/4$ and $j = \sqrt{-1}$. The properties of Proposition C.4 hold with $\alpha \in \mathbb{C}$, except that now $\langle f, g\rangle = \overline{\langle g, f\rangle}$ (complex conjugate). In particular, $\langle f, \alpha g\rangle = \overline{\alpha}\langle f, g\rangle$.

C.5 Definition. *A normed linear space that is complete is called a* **Banach space**. *A pre-Hilbert space that is complete is called a* **Hilbert space**.

A set $B = \{f^\nu\}_{\nu \in I}$ of elements of a Hilbert space F is called an orthonormal set of F if $\langle f^\nu, f^\nu\rangle = 1$ and $\langle f^\mu, f^\nu\rangle = 0$ for $\mu \neq \nu$. If in addition B is not a proper subset of an orthonormal set of F, then B is an orthogonal basis of F.

C.6 Proposition. *A separable Hilbert space F has an orthogonal base $\{f^\nu\}_{\nu \in I}$ with at most a countable number of elements (Yosida, 1980, Corollary III.5). Any $f \in F$*

can be represented as $f = \sum_{\nu \in I} c^\nu f^\nu$ *with* $c^\nu = \langle f, f^\nu \rangle$ *(Fourier expansion), whereas* $||f||^2 = \sum_{\nu \in I} |c^\nu|^2$ *(Parseval's relation).*

C.2 Linear Mappings

C.7 Definition. *Let* X, Y *be Banach spaces over a field* $\mathbb{K}$. *A mapping* $T : X \to Y$ *is said to be a* **linear mapping** *if* $\operatorname{dom} T = X$ *and* $T(\alpha x_1 + \beta x_2) = \alpha T(x_1) + \beta T(x_2)$ *for every* $x_1, x_2 \in X$ *and scalar* $\alpha_1, \alpha_2 \in \mathbb{K}$.

Let us denote by $||\cdot||_X$ and $||\cdot||_Y$ the norms of the Banach spaces X and Y. Let $||T||$ denote the smallest constant $c > 0$ such that $||T(x)||_Y \leq c||x||_X$ for all $x \in \operatorname{dom} T$. We say that T is bounded if $||T||$ is finite, and call $||T||$ the operator norm of T. It holds that a linear mapping $T : X \to Y$ is continuous if and only if T is bounded (Yosida, 1980, Corollary I.6.2).

Let $\mathcal{L}(X, Y)$ denote the space of all continuous linear mappings $T : X \to Y$. The following statement of Riesz's representation theorem is taken from Yosida (1980, Section III.6).

C.8 Theorem. *Let* X *be a Hilbert space over the field* $\mathbb{K}$ *and let* f *be an element of* $\mathcal{L}(X, \mathbb{K})$. *Then there exists a unique element* $y_f \in X$ *such that* $f(x) = \langle x, y_f \rangle$ *for every* $x \in X$ *with* $||f|| = ||y_f||_X$. *Conversely, an element* $y \in X$ *defines a unique mapping* f_y *in* $\mathcal{L}(X, \mathbb{K})$ *by* $f_y(x) = \langle x, y \rangle$ *for every* $x \in X$ *with* $||f_y|| = ||y||_X$.

There is a one-to-one correspondence between elements $f \in \mathcal{L}(X, \mathbb{K})$ and elements $y_f \in X$. In particular, if X is a real Hilbert space, $\mathcal{L}(X, \mathbb{R})$ (the dual of X) can be identified to a real Hilbert space equipped with the inner product $\langle f, g \rangle = \langle y_f, y_g \rangle$. If X is a complex Hilbert space, $\mathcal{L}(X, \mathbb{C})$ can be identified to a complex Hilbert space equipped with the inner product $\langle f, g \rangle = \overline{\langle y_f, y_g \rangle}$.

C.3 Reproducing Kernel Hilbert Spaces

C.9 Definition. *Let* F *be a space of functions* $f : X \to \mathbb{K}$ *forming a Hilbert space. The inner product between* $f, g \in F$ *is written* $\langle f(\cdot), g(\cdot) \rangle$. *Then, the mapping* $k : X \times X \to \mathbb{R}$ *with values* $k(x, y)$ *is a* **reproducing kernel** *of* F *if*

i. For every $y \in X$, *the function* $f_y(\cdot) = k(\cdot, y)$ *is in* F;

ii. (Reproducing property.) For every $f \in F$ *and every* $y \in X$, $f(y) = \langle f(\cdot), k(\cdot, y) \rangle$.

Note that $k(\cdot, y)$ acts as a Dirac distribution centered at y by the reproducing property, whereas $k(\cdot, y)$ is actually a function defined on X.

With $f(\cdot) = k(\cdot, y)$, Property *ii.* yields $k(x, y) = \langle k(\cdot, y), k(\cdot, x) \rangle$. For real Hilbert spaces we can write $k(x, y) = \langle k(\cdot, x), k(\cdot, y) \rangle$, whereas for complex Hilbert spaces we have $k(x, y) = \overline{\langle k(\cdot, x), k(\cdot, y) \rangle}$.

If a reproducing kernel k exists, it is unique (Aronszajn, 1950). A Hilbert space for which a reproducing kernel exists is called a **reproducing kernel Hilbert space** (RKHS). A reproducing kernel of F exists if and only if for every $y \in X$, the mapping $f \mapsto f(y)$ (called the **evaluation functional**) is a continuous linear mapping with respect to $f \in F$, meaning that there exists a finite $c_y > 0$ such that $|f(y)| \leq c_y||f||$ for all $f \in F$. If k exists, the smallest c_y is $k(y,y)^{1/2}$ by Cauchy-Schwartz inequality (C.4) applied to $f(y) = \langle f(\cdot, k(\cdot, y))\rangle$, whereas if a continuous linear mapping $F_y(f) = f(y)$ exists for every y, we have $F_y(f) = \langle f(\cdot), g_y(\cdot)\rangle$ for some $g_y \in X$ (by Theorem C.8) so that $g_y(x) = k(x, y)$ is a reproducing kernel (Yosida, 1980, Proof of Theorem III.9.1).

From the relation $|f(y)| \leq k(y, y)^{1/2}||f||$, one deduces that if there exists a scalar $c > 0$ such that $k(y, y)^{1/2} \leq c$ for all $y \in X$, then $||f||_\infty = \sup_{y\in X} |f(y)| \leq c||f||$. For the particular case of normalized kernels $[k(y, y) = 1]$ we have $||f||_\infty \leq ||f||$.

For a sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ in a RKHS, $||f_n - f|| \to 0$ entails $f_n(y) \to f(y)$ for every $y \in X$, since we have, by definition of the strong convergence, $f_n \to f$, and then $f_n(y) \to f(y)$ by continuity of the evaluation functional $k(\cdot, y)$.

C.10 Proposition. *A reproducing kernel k for a class F of $\mathbb{K}$-valued functions has the property that $\sum_{i=1}^n \sum_{j=1}^n \overline{\alpha_i} k(y_i, y_j)\alpha_j = ||\sum_{i=1}^n k(\cdot, y_i)\alpha_i||^2 \geq 0$ for any finite collection of elements $y_i \in F$ and coefficients $\alpha_i \in \mathbb{K}$. That is, the* **Gram matrix** *$K \in \mathbb{K}^{n\times n}$ with elements $K_{ij} = k(y_i, y_j)$ is positive semi-definite.*

When $X \subset \mathbb{R}^d$, Proposition C.10 can also be stated as follows: the linear mappings $L : F \to \mathbb{K}$ defined by $L(f) = \int_X \int_X \overline{\alpha(x)} k(x, y)\alpha(y) \, \mathrm{d}x \, \mathrm{d}y$, with k a reproducing kernel and α any $\mathbb{K}$-valued continuous function with nonzero values on a compact subset of X, are such that $F(f) \geq 0$.

The converse of Proposition C.10 is also true (Aronszajn, 1950, Theorem 2.4 attributed to E.H. Moore). Before stating the theorem, we define the notion of positive definite kernel.

C.11 Definition. *A function $k : X \times X \to \mathbb{K}$ that is hermitian $[k(x, y) = \overline{k(y, x)}]$, and such that any matrix in $\mathbb{K}^{n\times n}$ $(n \in \mathbb{N})$ with elements $K_{ij} = k(y_i, y_j)$ $(y_i \in X, 1 \leq i \leq n)$ is positive semi-definite, is called a* **positive definite kernel***.*

C.12 Theorem. *To every positive definite kernel $k : X \times X \to \mathbb{K}$, there corresponds a unique class F of functions $f : X \to \mathbb{K}$ forming a Hilbert space with a uniquely determined inner product and with k as a reproducing kernel.*

Being a reproducing kernel, a positive definite kernel k has the property that $k(\cdot, y)$ is continuous for every $y \in X$. The property does not imply that k is continuous as a mapping from $X \times X$ to $\mathbb{K}$ (Lehto, 1952). A continuous positive definite kernel is called a **Mercer kernel**. Since a function is continuous at any isolated point of its domain (Rudin, 1976, Definition 4.5), the distinction between positive definite kernels and Mercer kernels is irrelevant when X is a discrete set.

To build the class F of Theorem C.12 corresponding to a positive definite kernel k, we follow Aronszajn (1950):

C.13 Proposition. *The class F is generated by functions of the form*

$$f(\cdot) = \textstyle\sum_{i=1}^m \alpha_i k(\cdot, y_i) \quad \text{for some } m \in \mathbb{N}, \quad \alpha_i \in \mathbb{K}, \quad y_i \in X$$

to which corresponds a norm $||f|| = [\sum_i \sum_j \overline{\alpha_i} k(y_i, y_j)\alpha_j]^{1/2}$ and then the class is completed by limit functions of Cauchy sequences in the metric of the norm.

The inner product between two functions $f(\cdot) = \sum_{i=1}^m \alpha_i k(\cdot, y_i)$ and $g(\cdot) = \sum_{j=1}^n \beta_j k(\cdot, y'_j)$

is then given by

$$\langle f, g\rangle = \textstyle\sum_{i=1}^m \sum_{j=1}^n \alpha_i \overline{\beta_j} k(y'_j, y_i) = \overline{\textstyle\sum_{i=1}^m \sum_{j=1}^n \overline{\alpha_i} \beta_j k(y_i, y'_j)} \ .$$

C.4 Positive Definite Kernels

Theorems C.10 and C.12 show that Reproducing Kernel Hilbert Spaces are uniquely determined by the choice of a positive definite kernel. In the sequel, we refer to a positive definite kernel simply as a kernel.

If X is a compact subspace of $\mathbb{R}^d$, a continuous kernel $k : X \times X \to \mathbb{R}$ admits an eigenfunction expansion

$$k(x, y) = \textstyle\sum_{\nu=1}^m \lambda^\nu \psi^\nu(x)\psi^\nu(y)$$

with $\lambda^\nu > 0$ and $m \leq \infty$ (Mercer, 1909). The vector $\phi(x) = \{\sqrt{\lambda^\nu}\psi^\nu(x)\}_{1\leq\nu\leq m}$ is interpreted as a *feature vector* for x, whereas $k(x, y)$ is viewed as a generalized inner product between the vectors $\phi(x), \phi(y)$ valued in some feature space $\mathcal{F}$. The mapping $\phi : X \to \mathcal{F}$ is called a feature map (Aizerman et al., 1964).

To elucidate the nature of $\mathcal{F}$, observe that $\phi(x)$ belongs to the space ℓ^2 of vectors $\{\xi^\nu\}_{\nu\in\mathbb{N}}$ such that $\sum_{\nu=1}^\infty |\xi^\nu|^2 < \infty$, since $k(x, x) = \sum_{\nu=1}^m \phi^\nu(x)^2$ is finite. In fact ℓ^2 is a linear normed space equipped with the norm $||\{\xi^\nu\}_{\nu\in\mathbb{N}}|| = [\sum_{\nu=1}^\infty (\xi^\nu)^2]^{1/2}$, which can be interpreted as a generalization of the Euclidian norm in $\mathbb{R}^n$ when n tends to ∞. The feature map is continuous: $x^\nu \to \bar{x}$ entails $\phi(x^\nu) \to \phi(\bar{x})$, since $||\phi(x^\nu) - \phi(\bar{x})|| = k(x^\nu, x^\nu) + k(\bar{x}, \bar{x}) - 2k(x^\nu, \bar{x}) \to 0$ by continuity of k (Cucker and Smale, 2001).

From $k(\cdot, y) = \sum_{\nu=1}^m \lambda^\nu \psi^\nu(\cdot)\psi^\nu(y)$ and $f(\cdot) = \sum_{i=1}^n \alpha_i k(\cdot, y_i)$ for some $n \leq \infty$, one can see that f has the form $f(\cdot) = \sum_{\nu=1}^m \alpha_f^\nu \psi^\nu(\cdot)$ with $\alpha_f^\nu = \sum_{i=1}^n \alpha_i \lambda^\nu \psi^\nu(y_i)$, and that $\langle f, g\rangle = \sum_{\nu=1}^m \alpha_f^\nu \alpha_g^\nu / \lambda^\nu$.

In machine learning, it is common to extend the feature map interpretation to more general spaces X and say that a function $k : X \times X \to \mathbb{K}$ is a kernel if there exist a Hilbert space H and a mapping $\phi : X \to H$ such that $k(x, y) = \overline{\langle \phi(x), \phi(y)\rangle}$ for all $x, y \in X$ (Steinwart and Christman, 2008, Definition 4.1). The corresponding RKHS is the class F of functions of the form $f(\cdot) = \langle h, \phi(\cdot)\rangle_H$ for some $h \in H$, equipped with the norm $||f|| = \inf_{h\in H}\{||h||_H : f(\cdot) = \langle h, \phi(\cdot)\rangle_H\}$ (Steinwart and Christman, 2008, Theorem 4.21). Proposition C.13 still holds.

For the class of shift-invariant continuous kernels $k : X \times X \to \mathbb{R}$ with $X = \mathbb{R}^d$, where the shift-invariance property means that $k(x+\tau, y+\tau) = k(x, y)$ for any $\tau \in \mathbb{R}^d$, Bochner's theorem (Bochner, 1933) [see also Yosida (1980, Theorem XI.13.2)] characterizes the kernels k in the frequency domain. The following statement particularizes to real-valued normalized kernels ($k(x, x) = 1$ for all $x \in \mathbb{R}^d$) a form of Bochner's theorem given in Hofmann et al. (2008).

C.14 Theorem. *Let $h : \mathbb{R}^n \to \mathbb{R}$ by a continuous function with $h(0) = 1$ and $h(x) = h(-x)$. Then, the function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ with values $k(x, y) = h(x - y)$ is a kernel iff there exists a random vector $\xi \in \mathbb{R}^n$ on a probability space $(\Omega, \mathcal{B}, \mathbb{P})$ such that $h(x) = \mathbb{E}\{\exp\{j\ \langle x, \xi\rangle\}\}$.*

Thus we have $k(x, y) = \mathbb{E}\{\exp\{j\,\langle x, \xi\rangle\}\overline{\exp\{j\ \langle y, \xi\rangle\}}\}$, which is very similar to Mercer's eigenfunction expansion (the countable sum has been replaced by an integral as $X = \mathbb{R}^d$

is now unbounded).

From Bochner's theorem, Schoenberg (1938) obtains a characterization of shift-invariant kernels having a radial symmetry.

C.15 Definition. *A function $f : \mathbb{R} \to \mathbb{R}$ with values $f(t)$ is* **completely monotone** *(c.m.) for $t \geq 0$ if it is infinitely continuously differentiable on $(0, \infty)$ with $f(0) = f(0^+)$ and has for every k its derivative of order k satisfying*

$$(-1)^k f^{(k)}(t) \geq 0 \quad \text{for } 0 < t < \infty.$$

C.16 Theorem. *The function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ with values $k(x, y) = f(||x - y||^2)$ is a kernel iff $f : \mathbb{R} \to \mathbb{R}$ is a completely monotone (c.m.) function for $\mathbb{R}_+$.*

An example of c.m. function for $\mathbb{R}_+$ is $f(t) = \exp\{-at\}$ with $a \geq 0$; it shows that the function $k(x, y) = \exp\{-\frac{1}{2}||x - y||^2/\sigma^2\}$ is a kernel. Other simple examples of c.m. functions are $(a + b\,t)^{-q}$ with $q > 0$ and $a, b \geq 0$ (a, b not both 0), and $\log(a + bt^{-1})$ with $a \geq 1$ and $b > 0$. Simple composition rules are as follows: If f_1, f_2 are c.m. for $t \geq 0$, then $\alpha_1 f_1 + \alpha_2 f_2$ with $\alpha_1, \alpha_2 \geq 0$ is c.m. and $f_1(t)f_2(t)$ is c.m.; If f is c.m. and g is nonnegative with a c.m. derivative g', then $f(g(t))$ is c.m.

Kernels are closed under positive sums and pointwise products:

C.17 Proposition. *If $k_i : X \times X \to \mathbb{K}$ $(i = 1, 2)$ are kernels, then the following functions $k : X \times X \to \mathbb{K}$ are kernels.*

i. $k = \alpha_1 k_1 + \alpha_2 k_2$ $(\alpha_1, \alpha_2 \geq 0)$ *with values* $k(x, y) = \alpha_1 k_1(x, y) + \alpha_2 k_2(x, y)$.

ii. $k = k_1 \cdot k_2$ *with values* $k(x, y) = k_1(x, y)k_2(x, y)$.

If $k_i : X_i \times X_i \to \mathbb{K}$ $(i = 1, 2)$ are kernels, then the following functions $k : (X_1 \times X_2) \times (X_1 \times X_2) \to \mathbb{K}$ are kernels.

iii. $k = k_1 \oplus k_2$ *with values* $k(x_1, x_2, y_1, y_2) = k_1(x_1, y_1) + k_2(x_2, y_2)$.

iv. $k = k_1 \otimes k_2$ *with values* $k(x_1, x_2, y_1, y_2) = k_1(x_1, y_1)k_2(x_2, y_2)$.

If $k : (X \times X) \times (X \times X) \to \mathbb{K}$ with values $k(x_1, x_2, y_1, y_2)$ is a kernel, then $k^{\Delta} : X \times X \to \mathbb{K}$ with values $k^{\Delta}(x, y) = k(x, x, y, y)$ is a kernel (Haussler, 1999).

Kernels are also closed under pointwise limits:

C.18 Proposition. *If $\{k^\nu\}_{\nu \in \mathbb{N}}$ is a sequence of kernels $k^\nu : X \times X \to \mathbb{K}$ such that $k^\nu(x, y) \to k(x, y)$ for all $x, y \in X$, then k is a kernel.*

Using basic kernels and positivity-preserving operations, more complex kernels can be built. For example, one can define a kernel $k : X \times X \to \mathbb{R}$ with values

$$k(x, y) = \mathbb{E}\{\phi(\omega, x)\phi(\omega, y)\} \ ,$$

where $\phi : \Omega \times X \to \mathbb{R}$ is such that $\phi(\omega, x)$ is in $L^2(\Omega, \mathcal{B}, \mathbb{P})$ for each $x \in X$. If in addition $\mathbb{E}\{\phi(\omega, x)\} = 0$ for each x, we can interpret the Gram matrix for k evaluated at $x_1, \ldots, x_n$ as the covariance matrix of the random variables $\phi(\cdot, x_1), \ldots, \phi(\cdot, x_n)$.

Appendix D

Structural Results for Two-Stage Stochastic Programming

This appendix describes a classical formulation of the two-stage stochastic linear program with recourse, and gives details on the structure of optimal solutions. We have included this appendix in the thesis, because it clarifies the origin of certain assumptions that are found to be technically challenging to remove in stochastic programming models.

The material is mainly taken from Birge and Louveaux (1997), up to some adjustments based on Wets (1974); Römisch and Wets (2007); Shapiro et al. (2009).

The appendix is organized as follows. Section D.1 states the problem and gives a list of assumptions that ensure that the formulation is meaningful. Section D.2 gives useful properties that can be derived from the previous assumptions.

D.1 Problem Statement and Assumptions

Let $(\Omega, \mathcal{B}, \mathbb{P})$ be a probability space. A two-stage stochastic linear program with recourse is a program of the form

$$\begin{aligned}
\text{minimize} \quad & \langle c, x\rangle + \mathbb{E}\{Q(x,\omega)\} && \text{(D.1)}\\
\text{subject to} \quad & x \in K_1 \cap K_2 \ , && \text{(D.2)}\\
\text{where} \quad & Q(x,\omega) = \min_y \{ \ \langle q(\omega), y\rangle : \\
& \qquad\qquad T(\omega)x + W(\omega)y = h(\omega), \\
& \qquad\qquad 0 \preceq y \in \mathbb{R}^{m_2} \} && \text{(D.3)}\\
& K_1 = \{x \in \mathbb{R}^m : Ax = b, \ x \succeq 0\} && \text{(D.4)}\\
& K_2 = \{x \in \mathbb{R}^m : \mathbb{E}\{Q(x,\omega)\} < \infty\} && \text{(D.5)}
\end{aligned}$$

with $c \in \mathbb{R}^m$ (first-stage cost vector), $A \in \mathbb{R}^{s \times m}$, $b \in \mathbb{R}^s$, and $\mathcal{B}$-measurable mappings $q : \Omega \to \mathbb{R}^{m_2}$ (second-stage cost vector), $T : \Omega \to \mathbb{R}^{s_2 \times m}$ (technology matrix), $W : \Omega \to \mathbb{R}^{s_2 \times m_2}$ (recourse matrix), $h : \Omega \to \mathbb{R}^{s_2}$. We let a_i, t_i, w_i denote the i-th rows of A, T, W respectively.

Let $z : \Omega \to \mathbb{R}^{m_2} \times \mathbb{R}^{s_2 \times m} \times \mathbb{R}^{s_2 \times m_2} \times \mathbb{R}^{s_2}$ be the $\mathcal{B}$-measurable mapping with values

$$z(\omega) = (\ q(\omega), t_1(\omega), \dots, t_{s_2}(\omega), w_1(\omega), \dots, w_{s_2}(\omega), h(\omega)\ ) \qquad \text{(D.6)}$$

which collects all the (possibly non-random) elements of (q, T, W, h). Let $Z \subset \mathbb{R}^p$ with $p = m_2 + s_2(m + m_2 + 1)$ denote the support of z.

Various well-posed forms of the program can be distinguished. To this end, we describe standard assumptions. The joint role of those assumptions is detailed in Section D.2.

D.1 Definition (Measurability). *The support of z is a Borel set of $\mathbb{R}^p$, and the sigma-algebra $\mathcal{B}$ contains the collection of Borel sets of the support of z, that is,*

$$\mathcal{B} \supset \{B \cap Z : B \in \mathcal{B}(\mathbb{R}^p)\} \ .$$

The stated measurability assumption is consistent with Wets (1974, page 311). The measurability of $Q(x, \cdot)$ for each fixed x requires at least that the sigma-algebra generated by z be included in $\mathcal{B}$. It does not harm to allow sigma-algebras larger than the sigma-algebra generated by z since this would not alter the optimal value of the program. Note that using larger sigma-algebras makes it possible to select distinct vectors $y^*(\omega_1), y^*(\omega_2)$ for attaining the optimal value of Q when $z(\omega_1) = z(\omega_2)$, that is, to implement a stochastic policy for y. Most authors rule out this possibility, but in practice a numerical solution algorithm could indeed return distinct optimal values for y in face of duplicate realizations of z.

Now, recall that a mapping $F : \mathbb{R}^d \to \mathbb{R}^m$ is said to be affine iff it has values $F(\xi) = \bar{b} + B\xi$ for some fixed $\bar{b} \in \mathbb{R}^m$ and $B \in \mathbb{R}^{m\times d}$.

D.2 Definition (Affine dependence). *There exist a random variable $\xi : \Omega \to \mathbb{R}^d$ ($d \leq p$) and affine mappings $q_f : \mathbb{R}^d \to \mathbb{R}^{m_2}$, $T_f : \mathbb{R}^d \to \mathbb{R}^{s_2\times m}$, $W_f : \mathbb{R}^d \to \mathbb{R}^{s_2\times m_2}$, $h_f : \mathbb{R}^d \to \mathbb{R}^{s_2}$, possibly constant-valued, such that for all $\omega \in \Omega$,*

$$q(\omega) = q_f(\xi(\omega)) \ , \quad T(\omega) = T_f(\xi(\omega)) \ , \quad W(\omega) = W_f(\xi(\omega)) \ , \quad h(\omega) = h_f(\xi(\omega)).$$

The affine dependence assumption enforces the parametrization of (q, T, W, h) by ξ. It is made without loss of generality, as it is always possible to set $\xi = z$, and extract the appropriate coordinates of z through mappings q_f, T_f, W_f, h_f. One goal of the parametrization is to represent through ξ the randomness of the non-constant elements of z. Ideally, ξ is made of a small number of components, and the measure $\mathbb{P}$ is specified indirectly by the joint distribution of those components.

If we define

$$Q_f(x, \xi) = \min_y \{\langle q_f(\xi), y\rangle \ : \quad T_f(\xi)x + W_f(\xi)y = h_f(\xi), \quad 0 \preceq y \in \mathbb{R}^{m_2}\} \ , \qquad \text{(D.7)}$$

we have $Q(x, \omega) = Q_f(x, \xi(\omega))$.

D.3 Definition (Fixed Recourse). *For all ω, $W(\omega)$ is a fixed matrix $W \in \mathbb{R}^{s_2\times m_2}$.*

Fixed recourse is a simplifying assumption under which the value function $\mathbb{E}\{Q(\cdot, \omega)\}$ is easier to describe. The rows of the fixed matrix W are always assumed to be linearly independent to avoid trivial redundancies or conflicts among equality constraints (Wets, 1974, page 312).

D.4 Definition (Complete Recourse). *For all ω, $W(\omega)$ is a fixed matrix $W \in \mathbb{R}^{s_2\times m_2}$, and the positive hull of W coincides with $\mathbb{R}^{s_2}$:*

$$\operatorname{pos} W = \{Wy : y \succeq 0\} = W\mathbb{R}^{m_2}_{+} = \mathbb{R}^{s_2} \ .$$

The condition $W\mathbb{R}_+^{m_2} = \mathbb{R}^{s_2}$ implies that for any $x \in \mathbb{R}^m$ and all $\omega \in \Omega$, there exists some $y \succeq 0$ such that $T(\omega)x + Wy = h(\omega)$. This surjectivity condition is sufficient for having $Q(x, \omega) < \infty$ almost surely. From (D.8) below, one can show that complete recourse holds iff $\{\pi \in \mathbb{R}^{s_2} : W^T\pi \preceq 0\} = \{0\}$ (Shapiro et al., 2009, page 33). Recall from Rockafellar (1970, page 65) that the dimension of the largest subspace contained in a cone is called the *lineality* of the cone; methods to check that the lineality of pos W is s_2 are described in Wets and Witzgall (1967) and Wallace and Wets (1992).

D.5 Definition (Relatively Complete Recourse). *For all $x \in K_1$ and $\mathbb{P}$-almost all $\omega \in \Omega$, there exists some $y \succeq 0$ such that $T(\omega)x + W(\omega)y = h(\omega)$, that is,*

$$h(\omega) - T(\omega)x \in W(\omega)\mathbb{R}_+^{m_2} \ .$$

The relatively complete recourse assumption means that $Q(x, \omega) < \infty$ for almost all ω and $x \in K_1$. We could still have $\mathbb{E}\{Q(x, \omega)\} = \infty$ if the distribution of $Q(x, \cdot)$ is not integrable. In particular, the assumption alone does not guarantee that $K_1 \subset K_2$ — compare to Wets (1974, Definition 6.1) and Birge and Louveaux (1997, page 92). Note that no generic method is available for checking that a relatively complete recourse assumption holds. Relatively complete recourse is thus typically asserted at the modeling step, where penalties in the objective can be favored over hard constraints.

D.6 Definition (Dual Feasibility). *For $\mathbb{P}$-almost all $\omega \in \Omega$, the set*

$$\Pi(\omega) = \{\pi \in \mathbb{R}^s : W(\omega)^T\pi \preceq q(\omega)\}$$

is nonempty.

The dual feasibility assumption ensures that $Q(x, \omega) > -\infty$ for almost all ω and all x. Indeed, by weak duality,

$$\begin{aligned} Q(x,\omega) &= \inf_y\{\ \langle q(\omega),\ y\rangle : \qquad T(\omega)x + W(\omega)y = h(\omega), \quad y \succeq 0\ \} \\ &\geq \sup_\pi\{\ \langle \pi,\ h(\omega) - T(\omega)x\rangle : \qquad W(\omega)^T\pi \preceq q(\omega)\ \} \\ &> -\infty \quad \text{ if } \Pi(\omega) \neq \varnothing. \end{aligned} \tag{D.8}$$

D.7 Definition (Fixed Technology). *For all ω, $T(\omega)$ is a fixed matrix $T \in \mathbb{R}^{s_2 \times m}$.*

D.8 Definition (Finite Second Moments). *$z \in \mathcal{L}^2(\Omega, \mathcal{B}, \mathbb{P})$, that is, $\mathbb{E}\{||z||^2\} < \infty$.*

D.9 Definition (Finite Support). *The support of z is finite, that is, there exists a finite set $Z = \{z^1, z^2, \ldots, z^n\}$ such that $\mathbb{P}\{\omega : z(\omega) = z^\nu\} = p^\nu > 0$ with $\sum_{\nu=1}^n p^\nu = 1$.*

D.10 Definition (Polyhedral Support). *The support of z is a polyhedral set.*

A set is said to be polyhedral if it can be described as the intersection of a finite number of halfspaces. For instance, by definition, K_1 is polyhedral. A classic result from Weyl (1935) shows that polyhedral sets are the only sets that can also be described as the convex hull of a finite number of points and directions (Rockafellar, 1970, Theorem 19.1).

As the image of a polyhedral set in $\mathbb{R}^{n_1}$ by a linear transformation $\mathcal{F} : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ is a polyhedral set in $\mathbb{R}^{n_2}$ (Rockafellar, 1970, Theorem 19.3), the polyhedral support assumption holds if the affine dependence assumption holds with a polyhedral support for ξ. Note also that an affine mapping $F : \mathbb{R}^d \to \mathbb{R}^m$ with values $F(\xi) = \bar{b} + B\xi$ is Lipschitz continuous with modulus $||B|| = \max_{||u||=1} Bu$ (Rockafellar and Wets, 1998, Example 9.3).

D.2 Structural Properties

It is interesting to identify conditions under which $K_1 \cap K_2$ (constraint D.2) is polyhedral (Walkup and Wets, 1967). Clearly, the intersection of a finite number of polyhedral sets is a polyhedral set, so the question, if it is not circumvented by the relatively complete assumption, is reduced to investigating under which circumstances K_2 is polyhedral. Sufficient conditions are collected in the next proposition, mainly based on Wets (1974) and Birge and Louveaux (1997, Section 3.1). Essentially, these results concern cases where the recourse matrix W is fixed and cases where the random variables have a finite support.

D.11 Proposition (Representations of the effective domain of $\mathbb{E}\{Q(\cdot,\omega)\}$). *Under the following sufficient conditions, the set $K_2 = \{x \in \mathbb{R}^m : \mathbb{E}\{Q(x,\omega)\} < \infty\}$ in* (D.5) *admits the following representations.*

i. Under the finite second moments assumption,

$$K_2 = \{x \in \mathbb{R}^m : \mathbb{P}\{\omega : Q(x,\omega) < \infty\} = 1\} \enspace .$$

ii. Under the finite second moments and fixed recourse assumptions,

$$\begin{aligned} K_2 = \{x \in \mathbb{R}^m :\ & \textit{for } \mathbb{P}\textit{-almost all } \omega \in \Omega, \textit{ there is some } \quad y \succeq 0 \\ & \textit{such that} \quad Wy + T(\omega)x = h(\omega)\} \enspace , \end{aligned}$$

or equivalently, with Σ denoting the support of the distribution of (h, T),

$$K_2 = \bigcap_{(h,T)\in\Sigma} \{x \in \mathbb{R}^m : W\mathbb{R}_+^{m_2} \ni h - Tx\} \enspace ,$$

as shown in Wets (1974, Theorem 4.2) or Shapiro et al. (2009, Equation 2.33).

iii. Under the finite second moments and fixed recourse assumptions, if the support of T is polyhedral and if h, T are statistically independent, then K_2 is polyhedral (Wets, 1974, Corollary 4.13).

iv. Under the finite second moments, fixed recourse and fixed technology assumptions, K_2 is polyhedral; more precisely (Wets, 1974, Theorem 4.10) there exist a matrix $W^* \in \mathbb{R}^{p\times s_2}$ (p finite) and a vector $\alpha^* \in \overline{\mathbb{R}}^p$ such that

$$K_2 = \{x \in \mathbb{R}^m : W^*Tx \succeq \alpha^*\} \enspace .$$

v. Under the finite second moments and complete recourse assumptions, $K_2 = \mathbb{R}^m$.

vi. Under the finite support assumption, K_2 is polyhedral; more precisely,

$$\begin{aligned} K_2 &= \{x \in \mathbb{R}^m :\ \textit{for all } \omega \in \Omega, \textit{ there is some } \quad y(\omega) \succeq 0 \\ &\qquad \textit{such that} \quad W(\omega)y(\omega) = h(\omega) - T(\omega)x\} \\ &= \bigcap_{\nu=1}^{n} \{x \in \mathbb{R}^m : y^\nu \succeq 0, \quad W^\nu y^\nu = h^\nu - T^\nu x\} \end{aligned}$$

where the elements indexed by ν refer to the realizations ξ^ν.

A proper function is said to be polyhedral when its epigraph is a polyhedral set. The domain of a polyhedral function is necessarily a polyhedral set. A result established in Rockafellar and Wets (1998, Theorem 2.49) shows that the class of proper polyhedral functions is the class of proper convex piecewise linear functions.

It is interesting to identify conditions under which $\mathbb{E}\{Q(x,\omega)\}$ (second term of objective in D.1, often referred to as the value function) is polyhedral. The finite sum of proper polyhedral convex functions is polyhedral (Rockafellar, 1970, Theorem 19.4) and the multiplication of a proper polyhedral convex function by a nonnegative scalar is polyhedral (Rockafellar, 1970, Corollary 19.5.1), so when the support of ξ is finite, the question is reduced to investigating under which conditions the integrand of the value function, $Q(x,\omega)$, is proper and polyhedral.

The next lemma (Römisch and Wets, 2007, Lemma 3.1), which reformulates results in Walkup and Wets (1969a), is instrumental in describing the structure of $Q(x,\omega)$ without necessarily assuming fixed recourse. Under the affine dependence assumption, let $\Xi \subset \mathbb{R}^d$ denote the support of ξ, and let $\Phi : \mathbb{R}^d \times \mathbb{R}^{m_2} \times \mathbb{R}^{s_2} \to \overline{\mathbb{R}}$ be a mapping with values

$$\Phi(\xi, q, t) = \inf_y \{\langle q,\ y\rangle :\ W_f(\xi) y = t,\ y \succeq 0\} \ .$$

Observe that

$$Q(x,\omega) = Q_f(x, \xi(\omega)) = \Phi(\ \xi(\omega),\ q_f(\xi(\omega)),\ h_f(\xi(\omega)) - T_f(\xi(\omega))\, x\ ) \ .$$

By analogy to the relatively complete recourse assumption, let $H(\xi) = W_f(\xi)\mathbb{R}_+^{m_2}$, and by analogy to the dual feasibility assumption, let

$$\Pi_f(\xi) = \{\pi \in \mathbb{R}^{s_2} : W_f(\xi)^T \pi \preceq q_f(\xi)\} \ , \qquad D(\xi) = \{q \in \mathbb{R}^{m_2} : \Pi_f(\xi) \neq \varnothing\} \ .$$

D.12 Lemma. *Let the affine dependence and the polynomial support assumptions hold. Let $\xi \in \Xi$ be fixed. Then,*

i. The sets $D(\xi)$ and $H(\xi)$ are polyhedral;

ii. The function $\Phi(\xi, \cdot, \cdot)$ is finite and continuous on $D(\xi) \times H(\xi)$;

iii. The function $\Phi(\xi, q, \cdot)$ is piecewise linear convex on $H(\xi)$ for fixed $q \in D(\xi)$;

iv. The function $\Phi(\xi, \cdot, t)$ is piecewise linear concave on $D(\xi)$ for fixed $t \in H(\xi)$.

Inasmuch as $t = h_f(\xi(\omega)) - T_f(\xi(\omega))x$, it holds that t depends affinely on x when ω and thus h, T are fixed.

When the recourse is fixed, Lemma D.12 allows to establish the following local Lipschitz continuity properties for Q or equivalently Q_f (Rachev and Römisch, 2002, Proposition 3.2).

D.13 Lemma. *Let the affine dependence and the polynomial support assumptions hold. Under the finite second moments, the fixed recourse, the relatively complete recourse and the dual feasibility assumptions, there exist constants $L_1 > 0$, $L_2 > 0$, $K > 0$ such that for all $\xi, \xi' \in \Xi$, any $\rho > 0$, and for all $x, x' \in K_1 \cap K_2 \cap \rho\mathbb{B}$,*

i. $|Q_f(x,\xi) - Q_f(x,\xi')| \leq L_1 \rho \max\{1, ||\xi||, ||\xi'||\}\ ||\xi - \xi'||$;

ii. $|Q_f(x,\xi) - Q_f(x',\xi)| \leq L_2 \max\{1, ||\xi||^2\}\ ||x - x'||$;

iii. $|Q_f(x,\xi)| \leq K\rho \max\{1, ||\xi||^2\}$.

Finally, the following proposition, that addresses the differentiability of the value function, is based on Walkup and Wets (1969b), Wets (1974) and Shapiro et al. (2009, Propositions 2.7, 2.8, 2.9). Note that under the finite second moments and relatively complete recourse assumptions, we have $K_1 \subset K_2$, so that K_2 is nonempty if K_1 is nonempty.

D.14 Proposition. *Under the finite second moments, fixed recourse, relatively complete recourse and dual feasibility assumptions, and assuming that K_1 is nonempty,*

i. $\mathbb{E}\{Q(\cdot,\omega)\}$ *is proper;*

ii. $\mathbb{E}\{Q(\cdot,\omega)\}$ *is convex, lower semicontinuous and Lipschitz continuous on K_2;*

iii. If (q,T) is constant-valued, and the distribution of h is absolutely continuous, then $\mathbb{E}\{Q(\cdot,\omega)\}$ *is differentiable at $x_0 \in \operatorname{int} K_2$;*

iv. If for almost all (q,T), the distribution of h conditionally to (q,T) is absolutely continuous, and if for almost all $\omega \in \Omega$, the dual solution set at $x_0 \in \operatorname{int} K_2$

$$\arg\max_{\pi}\{\ \langle \pi, h(\omega) - T(\omega)x_0 \rangle : W^T\pi \preceq q(\omega)\ \}$$

is a singleton, then $\mathbb{E}\{Q(\cdot,\omega)\}$ *is differentiable at $x_0 \in \operatorname{int} K_2$.*

Bibliography

Ailon, N., B. Chazelle, K.L. Clarkson, D. Liu, W. Mulzer, C. Seshadhri. 2006. Self-improving algorithms. *Proceedings of the Seventeenth ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*. 261–270.

Aizerman, M., E. Braverman, L. Rozonoer. 1964. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control* **25** 821–837.

Ali, S.M., S. Koenig, M. Tambe. 2005. Preprocessing techniques for accelerating the DCOP algorithm ADOPT. *Proceedings of the Fourth International Joint Conference on Autonomous Agents & Multi Agent Systems*. 1041–1048.

Ali, S.M., S.D. Silvey. 1966. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society* **28** 131–142.

Antos, A., R. Munos, Cs. Szepesvári. 2008. Fitted Q-iteration in continuous action-space MDPs. *Advances in Neural Information Processing Systems 20 (NIPS-2007)*. 9–16.

Aronszajn, N. 1950. Theory of reproducing kernels. *Transactions of the American Mathematical Society* **68**(3) 337–404.

Arrow, K.J. 1958. Historical background. K.J. Arrow, S. Karlin, H. Scarf, eds., *Studies in the Mathematical Theory of Inventory and Production*. Stanford University Press, 3–15.

Artzner, P., F. Delbaen, J.-M. Eber, D. Heath, H. Ku. 2007. Coherent multiperiod risk adjusted values and Bellman's principle. *Annals of Operations Research* **152**(1) 5–22.

Aubin, J.-P., H. Frankowska. 1990. *Set-Valued Analysis*. Modern Birkhäuser Classics, Springer. 2009 Reprint of the 1990 Edition.

Audibert, J.-Y., R. Munos, C. Szepesvári. 2007. Tuning bandit algorithms in stochastic environments. *Proceedings of the Eighteenth International Conference on Algorithmic Learning Theory (ALT-2007)*. LNCS 4754, Springer, 150–165.

Auer, P., N. Cesa-Bianchi, P. Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* **47** 235–256.

Azuma, K. 1967. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal* **19** 357–367.

Balas, E. 1998. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics* **89** 3–44.

Balasubramanian, J., I.E. Grossmann. 2003. Approximation to multistage stochastic optimization in multiperiod batch plant scheduling under demand uncertainty. *Industrial & Engineering Chemistry Research* **43** 3695–3713.

Banerjee, A., S. Merugu, I.S. Dhillon, J. Ghosh. 2005. Clustering with Bregman divergences. *Journal of Machine Learning Research* **6** 1705–1749.

Banerjee, O., L. El Ghaoui, A. d'Aspremont. 2008. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research* **9** 485–516.

Baotić, M., F. Borrelli, A. Bemporad, M. Morari. 2008. Efficient on-line computation of constrained optimal control. *SIAM Journal on Control and Optimization* **47** 2470–2489.

Bartlett, P.L., S. Mendelson. 2002. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research* **3** 463–482.

Beale, E.M.L. 1955. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society* **17** 173–184.

Bellman, R.E. 1954. The theory of dynamic programming. *Bulletin of the American Mathematical Society* **60** 503–516.

Bemporad, A., M. Morari, V. Dua, E. Pistikopoulos. 2002. The explicit linear quadratic regulator for constrained systems. *Automatica* **38** 3–20. Corrigendum: Automatica 39 (2003) 1845-1846.

Ben-Tal, A., A. Goryashko, E. Guslitzer, A. Nemirovski. 2004. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming* **99**(2) 351–376.

Bertsekas, D.P. 2005a. *Dynamic Programming and Optimal Control*. 3rd ed. Athena Scientific, Belmont, MA.

Bertsekas, D.P. 2005b. Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control* **11** 310–334.

Bertsekas, D.P., J.N. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

Billingsley, P. 1995. *Probability and Measure*. 3rd ed. Wiley.

Birge, J.R. 1992. The value of the stochastic solution in stochastic linear programs with fixed recourses. *Mathematical Programming* **24** 314–325.

Birge, J.R., F. Louveaux. 1997. *Introduction to Stochastic Programming*. Springer-Verlag, New York.

Bixby, R. 2002. Solving real-world linear programs: A decade and more of progress. *Operations Research* **50** 3–15.

Bochner, S. 1933. Monotone funktionen, stieltjessche integrale und harmonische analyse. *Mathematische Annalen* **108**(1) 378–410.

Boda, K., J.A. Filar. 2006. Time consistent dynamic risk measures. *Mathematical Methods of Operations Research* **63** 169–186.

Bregman, L.M. 1967. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics* **7** 200–217.

Breiman, L. 1996. Bagging predictors. *Machine Learning* **24**(2) 123–140.

Breiman, L. 1998. Arcing classifiers (with discussion and a rejoinder by the author). *The Annals of Statistics* **26** 801–849.

Breiman, L. 2000. Randomizing outputs to increase prediction accuracy. *Machine Learning* **40** 229–242.

Breiman, L. 2001. Random forests. *Machine Learning* **45** 5–32.

Brown, L.D. 1986. *Fundamentals of statistical exponential families*, *IMS lecture notes – Monograph series*, vol. 9. Institute of Mathematical Statistics, Hayward, California.

Buja, A., W. Stuetzle. 2006. Observations on bagging. *Statistica Sinica* **16** 323–351.

Cesa-Bianchi, N., A. Conconi, C. Gentile. 2004. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory* **50** 2050–2057.

Cesa-Bianchi, N., Y. Freund, D.P. Helmbold, D. Haussler, R.E. Schapire, M.K. Warmuth. 1997. How to use expert advice. *Journal of the Association for Computing Machinery* **44** 427–485.

Cesa-Bianchi, N., G. Lugosi. 1999. On prediction of individual sequences. *The Annals of Statistics* **27** 1865–1895.

Cesa-Bianchi, N., G. Lugosi. 2006. *Prediction, Learning and Games*. Cambridge University Press, New York.

Chiralaksanakul, A. 2003. Monte Carlo methods for multi-stage stochastic programs. Ph.D. thesis, University of Texas at Austin.

Chung, K.-J., M. Sobel. 1987. Discounted MDP's: Distribution functions and exponential utility maximization. *SIAM J. Control and Optimization* **25**(1) 49–62.

Clarke, B.S., A.R. Barron. 1990. Information-theoretic asymptotics of Bayes methods. *IEEE Transactions of Information Theory* **36** 453–471.

Coquelin, P.-A., R. Munos. 2007. Bandit algorithms for tree search. *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI-2007)*. 67–74.

Csizár, I. 1967. Information-type measures of difference of probability distributions and indirect observation. *Studia Scientiarum Mathematicarum Hungarica* **2** 229–318.

Cucker, F., S. Smale. 2001. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society* **39**(1) 1–49.

Daniel, J.W. 1973. Stability of the solution of definite quadratic programs. *Mathematical Programming* **5** 41–53.

Dantzig, G.B. 1955. Linear programming under uncertainty. *Management Science* **1** 197–206.

Dawid, A.P. 1984. Present position and potential developments: some personal views. Statistical theory: The prequential approach (with discussion). *Journal of the Royal Statistical Society* **147** 278–292.

Decoste, D., B. Schölkopf. 2002. Training invariant support vector machines. *Machine Learning* **46** 161–190.

Defourny, B., D. Ernst, L. Wehenkel. 2008. Risk-aware decision making and dynamic programming. Y. Engel, M. Ghavamzadeh, S. Mannor, P. Poupart, eds., *NIPS-08 workshop on model uncertainty and risk in reinforcement learning*.

Defourny, B., L. Wehenkel. 2009. Large margin classification with the progressive hedging algorithm. S. Nowozin, S. Sra, S. Vishwanathan, S. Wright, eds., *Second NIPS workshop on optimization for machine learning*.

Dempster, A.M. 1972. Covariance selection. *Biometrics* 157–175.

Dempster, A.P., N.M. Laird, D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* **39** 1–38.

Dempster, M.A.H., G. Pflug, G. Mitra, eds. 2008. *Quantitative Fund Management*. Financial Mathematics Series, Chapman & Hall/CRC.

Demuth, H., M. Beale. 1993. Neural network toolbox for use with Matlab.

Dietterich, T. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting and Randomization. *Machine Learning* **40** 139–157.

Dontchev, A.L., R.T. Rockafellar. 2009. *Implicit Functions and Solution Mappings*. Springer.

Draper, D. 1995. Assessment and propagation of model uncertainty. *Journal of the Royal Statistical Society* **57** 45–97.

Dupacova, J., R. J.-B. Wets. 1988. Asymptotic behavior of statistical estimators and of optimal solutions of stochastic optimization problems. *The Annals of Statistics* **16** 1517–1549.

Dutech, A., T. Edmunds, J. Kok, M. Lagoudakis, M. Littman, M. Riedmiller, B. Russel, B. Scherrer, R. Sutton, S. Timmer, N. Vlassis, A. White, S. Whiteson. 2005. Reinforcement Learning benchmarks and bake-offs II: A workshop at the 2005 NIPS conference. Available at http://www.cs.rutgers.edu/∼mlittman/topics/nips05-mdp/bakeoffs05.pdf.

Edelman, A. 1992. Eigenvalue roulette and random test matrices. M.S. Moonen, G.H. Golub, B. De Moor, eds., *Linear Algebra for Large Scale and Real-Time Applications*, *NATO ASI*, vol. 232. Springer, 365–368.

Efron, B., R. Tibshirani. 1993. *An introduction to the bootstrap*. Chapman and Hall, London.

Epstein, L., M. Schneider. 2003. Recursive multiple-priors. *Journal of Economic Theory* **113** 1–13.

Ernst, D., P. Geurts, L. Wehenkel. 2005. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* **6** 503–556.

Escudero, L.F. 2009. On a mixture of the fix-and-relax coordination and Lagrangian substitution schemes for multistage stochastic mixed integer programming. *Top* **17** 5–29.

Evans, M., T. Swartz. 1995. Methods for approximating integrals in statistics with special emphasis on Bayesian integration problems. *Statistical Science* **10** 254–272.

Facchinei, F., A. Fischer, C. Kanzow. 1998. On the accurate identification of active constraints. *SIAM Journal on Optimization* **9** 14–32.

Facchinei, F., J.-S. Pang. 2003. *Finite-dimensional variational inequalities and complementary problems*. Springer. Published in two volumes, paginated continuously.

Fisher, R.A. 1925. Theory of statistical estimation. *Proceedings of the Cambridge Philosophical Society*, vol. 22. 700–725.

Frauendorfer, K. 1996. Barycentric scenario trees in convex multistage stochastic programming. *Mathematical Programming* **75** 277–294.

Freund, Y., R.E. Schapire. 1996. Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-1996)*. 148–156.

Gale, D., V. Klee, R.T. Rockafellar. 1968. Convex functions on convex polytopes. *Proceedings of the American Mathematical Society*, vol. 19. 867–873.

Garstka, S.J., R.J.-B. Wets. 1974. On decision rules in stochastic programming. *Mathematical Programming* **7**(1) 117–143.

Gassmann, H.I., A. Prékopa. 2005. On stages and consistency checks in stochastic programming. *Operations Research Letters* **33** 171–175.

Geurts, P., D. Ernst, L. Wehenkel. 2006. Extremely randomized trees. *Machine Learning* **63** 3–42.

Good, I.J., R.A. Gaskins. 1971. Non-parametric roughness penalties for probability densities. *Biometrika* **58** 255–277.

Haff, L.R. 1980. Empirical Bayes estimation of the multivariate normal covariance matrix. *The Annals of Statistics* **8** 586–597.

Hammond, P. J. 1976. Changing tastes and coherent dynamic choice. *The Review of Economic Studies* **43** 159–173.

Hartland, C., S. Gelly, N. Baskiotis, O. Teytaud, M. Sebag. 2006. Multi-armed bandit, dynamic environments and meta-bandits. P. Auer, N. Cesa-Bianchi, Z. Hussain, L. Newnham, J. Shawe-Taylor, eds., *NIPS-06 workshop on on-line trading of exploration and exploitation*.

Hasselblad, V. 1966. Estimation of parameters for a mixture of normal distributions. *Technometrics* **8** 431–444.

Haussler, D. 1999. Convolution kernels on discrete structures. Tech. rep., University of California at Santa Cruz.

Heitsch, H., W. Römisch. 2003. Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications* **24** 187–206.

Heitsch, H., W. Römisch. 2009. Scenario tree modeling for multistage stochastic programs. *Mathematical Programming* **118**(2) 371–406.

Higle, J.L., S. Sen. 1991. Stochastic decomposition: An algorithm for two stage stochastic linear programs with recourse. *Mathematics of Operations Research* **16** 650–669.

Hilli, P., T. Pennanen. 2008. Numerical study of discretizations of multistage stochastic programs. *Kybernetika* **44** 185–204.

Hochreiter, R., G.Ch. Pflug. 2007. Financial scenario generation for stochastic multi-stage decision processes as facility location problems. *Annals of Operations Research* **152** 257–272.

Hoeffding, W. 1963. Probability inequalities for sums of bounded random variables. *Journal of the Americal Statistical Association* **58** 13–30.

Hoffman, A.J. 1952. On approximate solutions of systems of linear inequalities. *Journal of Research of the National Bureau of Standards* **49** 263–265.

Hofmann, T., B. Schölkopf, A. Smola. 2008. Kernel methods in machine learning. *The Annals of Statistics* **36**(3) 1171–1220.

Howard, R.A. 1960. *Dynamic Programming and Markov Processes*. MIT Press.

Howard, R.A., J. Matheson. 1972. Risk-sensitive Markov Decision Processes. *Management Science* **18**(7) 356–369.

Høyland, K., M. Kaut, S.W. Wallace. 2003. A heuristic for moment-matching scenario generation. *Computational Optimization and Applications* **24** 1573–2894.

Høyland, K., S. Wallace. 2001. Generating scenario trees for multistage decision problems. *Management Science* **47**(2) 295–307.

Huang, K., S. Ahmed. 2009. The value of multistage stochastic programming in capacity planning under uncertainty. *Operations Research* **57** 893–904.

Huber, P.J. 1964. Robust estimation of a location parameter. *The Annals of Mathematical Statistics* **35** 73–101.

Infanger, G. 1992. Monte Carlo (importance) sampling within a Benders decomposition algorithm for stochastic linear programs. *Annals of Operations Research* **39** 69–95.

Itakura, F., S. Saito. 1968. Analysis synthesis telephony based on the maximum likelihood method. *Proceedings of the Sixth International Congress on Acoustics*. C17–20.

Jeffrey, H. 1939. *Theory of Probability*. Oxford University Press.

Kallrath, J., P.M. Pardalos, S. Rebennack, M. Scheidt, eds. 2009. *Optimization in the Energy Industry*. Springer.

Kearns, M.J., Y. Mansour, A.Y. Ng. 2002. A sparse sampling algorithm for near-optimal planning in large Markov Decision Processes. *Machine Learning* **49**(2-3) 193–208.

Kimeldorf, G.S., G. Wahba. 1970. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics* **41** 495–502.

Koivu, M., T. Pennanen. 2010. Galerkin methods in dynamic stochastic programming. *Optimization* **59** 339–354.

Koltchinskii, V., D. Panchenko. 2002. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics* **30** 1–50.

Kouwenberg, R. 2001. Scenario generation and stochastic programming models for asset liability management. *European Journal of Operational Research* **134** 279–292.

Küchler, C., S. Vigerske. 2010. Numerical evaluation of approximation methods in stochastic programming. *Optimization* **59** 401–415.

Kuhn, D. 2005. *Generalized Bounds for convex multistage Stochastic Programs*, *Lecture Notes in Economics and Mathematical Systems*, vol. 548. Springer.

Kydland, F.E., E.C. Prescott. 1977. Rules rather than discretion: The inconsistency of optimal plans. *The Journal of Political Economy* **85** 473–492.

Kyparisis, J. 1985. On uniqueness of Kuhn Tucker multipliers in nonlinear programming. *Mathematical Programming* **32** 242–246.

Lagoudakis, M.G., R. Parr. 2003. Reinforcement learning as classification: leveraging modern classifiers. *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*. 424–431.

Lanckriet, G., N. Cristianini, P. Bartlett, L.E. Ghaoui, M.I. Jordan. 2004. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research* **5** 27–72.

Langford, J., B. Zadrozny. 2005. Relating reinforcement learning performance to classification performance. *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML-2005)*. 473–480.

Lehto, O. 1952. Some remarks on the kernel functions in Hilbert spaces. *Annales Academiae Scientiarium, Fennicae Sereie A* **109** 3–6.

Li, J.Q., A.R. Barron. 2000. Mixture density estimation. *Advances in Neural Information Processing Systems 12 (NIPS-2000)*. 279–285.

Littlestone, N., M.K. Warmuth. 1989. The weighted majority algorithm. *Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science*. 256–261.

Littman, M.L., T.L. Dean, L.P. Kaelbling. 1995. On the complexity of solving Markov Decision Problems. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-1995)*. 394–402.

MacKay, D.J.C. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

MacKay, M.D., R.J. Beckman, W.J. Conover. 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21** 239–245.

Madigan, D., J. York. 1995. Bayesian graphical models for discrete data. *International Statistical Review* **63** 215–232.

Mahalanobis, P.C. 1936. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India*, vol. 12. 49–55.

Mak, Wai-Kei, D.P. Morton, R. Kevin Wood. 1999. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters* **24**(1-2) 47–56.

Mardia, K.V., R.J. Marshall. 1984. Maximum likelihood estimation of models for residual covariance in spatial regression. *Biometrika* **71** 135–146.

Martin, D.H. 1975. On the continuity of the maximum in parametric linear programming. *Journal of Optimization Theory and Applications* **17** 205–210.

McDiarmid, C. 1989. On the method of bounded differences. *Surveys in Combinatorics* **141** 148–188.

Mease, D., A. Wyner. 2008. Evidence contrary to the statistical view of boosting (with responses and rejoinder). *Journal of Machine Learning Research* 131–201.

Mendelson, B. 1990. *Introduction to Topology*. 3rd ed. Dover.

Mercer, J. 1909. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London, Series A* **209** 415–446.

Mercier, L., P. Van Hentenryck. 2007. Performance analysis of online anticipatory algorithms for large multistage stochastic integer programs. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*. 1979–1984.

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics* **21** 1087–1092.

Metropolis, N., S. Ulam. 1949. The Monte Carlo method. *J. Amer. Stat. Assoc.* **44**(247) 335–341.

Micchelli, C.A., M. Pontil. 2005. Learning the kernel function via regularization. *Journal of Machine Learning Research* **6** 1099–1125.

Morimura, T., M. Sugiyama, H. Kashima, H. Hachiya, T. Tanaka. 2010. Parametric return density estimation for reinforcement learning. *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI-2010).*

Mundhenk, M., J. Goldsmith, C. Lusena, E. Allender. 2000. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM* **47** 681–720.

Murty, K. 1980. Computational complexity of parametric linear programming. *Mathematical Programming* **19** 213–219.

Neal, R.M. 1993. Probabilistic inference using Markov Chain Monte Carlo methods. Tech. Rep. CRG-TR-93-1, University of Toronto.

Neal, R.M. 1997. Monte Carlo implementation of Gaussian Process models for Bayesian regression and classification. Tech. Rep. 9702, University of Toronto.

Neal, R.M. 2010. MCMC using Hamiltonian dynamics. S. Brooks, A. Gelman, G. Jones, X.-L. Meng, eds., *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC Press.

Nemirovski, A., A. Juditsky, G. Lan, A. Shapiro. 2009. Stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization* **19** 1574–1609.

Nesterov, Y. 2007. Gradient methods for minimizing composite objective function. CORE discussion paper 76, Catholic University of Louvain.

Nesterov, Y., J.-Ph. Vial. 2008. Confidence level solutions for stochastic programming. *Automatica* **44** 1559–1568.

Nickisch, H., C.E. Rasmussen. 2008. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research* **9** 2035–2078.

Norkin, V.I., Y.M. Ermoliev, A. Ruszczyński. 1998a. On optimal allocation of indivisibles under uncertainty. *Operations Research* **46** 381–395.

Norkin, V.I., G.Ch. Pflug, A. Ruszczyński. 1998b. A branch and bound method for stochastic global optimization. *Mathematical Programming* **83** 425–450.

O'Hagan, A. 1978. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society* **40** 1–42.

Pages, G., J. Printems. 2003. Optimal quadratic quantization for numerics: the Gaussian case. *Monte Carlo Methods and Applications* **9** 135–166.

Patrinos, P., H. Sarimveis. 2010. A new algorithm for solving convex parametric quadratic programs based on graphical derivatives of solution mappings. *Automatica* **46** 1405–1418.

Pennanen, T. 2009. Epi-convergent discretizations of multistage stochastic programs via integration quadratures. *Mathematical Programming* **116** 461–479.

Pflug, G.Ch., W. Römisch. 2007. *Modeling, Measuring and Managing Risk*. World Scientific Publishing Company.

Pollard, D. 1990. *Empirical Processes: Theory and Applications*, *NSF-CBMS Regional Conference Series in Probability and Statistics*, vol. 2. Institute of Mathematical Statistics.

Pollard, D. 2001. *A User's Guide to Measure Theoretic Probability*. Cambridge University Press.

Powell, W.B. 2007. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley.

Powell, W.B., H. Topaloglu. 2003. Stochastic programming in transportation and logistics. A. Ruszczyński, A. Shapiro, eds., *Stochastic Programming.* Handbooks in Operations Research and Management Science, vol. 10. Elsevier, 555–635.

Prékopa, A. 1970. On probabilistic constrained programming. H.W. Kuhn, ed., *Proceedings of the Princeton Symposium on Mathematical Programming*. Princeton University Press, 113–138.

Prékopa, A. 1995. *Stochastic Programming*. Kluwer, Dordrecht, Boston.

Puterman, M.L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

Rachelson, E., F. Schnitzler, L. Wehenkel, D. Ernst. 2010. Optimal sample selection for batch-mode reinforcement learning. Submitted.

Rachev, S.T., W. Römisch. 2002. Quantitative stability in stochastic programming: The method of probability metrics. *Mathematics of Operations Research* **27**(4) 792–818.

Rahimi, A., B. Recht. 2008. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems 20 (NIPS-2007)*. 1177–1184.

Rahimi, A., B. Recht. 2009. Random kitchen sinks: Replacing optimization with randomization in learning. *Advances in Neural Information Processing Systems 21 (NIPS-2008)*. 1313–1320.

Raiffa, H., R. Schlaifer. 1961. *Applied Statistical Decision Theory*. Harvard University.

Rasmussen, C.E., C.K.I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Riedel, F. 2004. Dynamic coherent risk measures. *Stochatic Processes and their Applications* **112** 185–200.

Rissanen, J. 1978. Modeling by shortest data description. *Automatica* **14** 465–471.

Robert, C.P. 2007. *The Bayesian Choice*. 2nd ed. Springer.

Rockafellar, R.T. 1970. *Convex Analysis*. Princeton University Press.

Rockafellar, R.T. 1974. *Conjugate Duality and Optimization*. CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM.

Rockafellar, R.T., S. Uryasev. 2000. Optimization of conditional value-at-risk. *Journal of Risk* **2**(3) 21–41.

Rockafellar, R.T., R.J.-B. Wets. 1991. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research* **16** 119–147.

Rockafellar, R.T., R.J.-B. Wets. 1998. *Variational Analysis*. 3rd ed. Springer.

Römisch, W., R.J.-B. Wets. 2007. Stability of ε-approximate solutions to convex stochastic programs. *SIAM Journal on Optimization* **18** 961–979.

Rubinstein, R.Y., D.P. Kroese. 2004. *The Cross-Entropy Method. A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Information Science and Statistics, Springer.

Rudin, W. 1976. *Principles of Mathematical Analysis*. 3rd ed. McGraw-Hill.

Rust, J. 1997. Using randomization to break the curse of dimensionality. *Econometrica* **65**(3) 487–516.

Ruszczyński, A., A. Shapiro. 2006. Conditional risk mappings. *Mathematics of Operations Research* **31** 544–561.

Samuelson, P.A. 1937. A note on measurement of utility. *The Review of Economic Studies* **4**(2) 155–161.

Schapire, R.E. 1990. The strength of weak learnability. *Machine Learning* **5**(2) 197–227.

Schapire, R.E., P. Stone, D. McAllester, M.L. Littman, J.A. Csirik. 2002. Modeling auction price uncertainty using boosting-based conditional density estimation. *Proceedings of the Nineteenth International Conference on Machine Learning (ICML-2002)*. 546–553.

Schoenberg, I.J. 1938. Metric spaces and completely monotone functions. *Annals of Mathematics* **39**(4) 811–841.

Sen, S., R.D. Doverspike, S. Cosares. 1994. Network planning with random demand. *Telecommunications Systems* **3** 11–30.

Shapiro, A. 2000. On the asymptotics of constrained local M-estimators. *The Annals of Statistics* **28** 948–960.

Shapiro, A. 2003a. Inference of statistical bounds for multistage stochastic programming problems. *Mathematical Methods of Operations Research* **58**(1) 57–68.

Shapiro, A. 2003b. Monte Carlo sampling methods. A. Ruszczyński, A. Shapiro, eds., *Stochastic Programming*. Handbooks in Operations Research and Management Science, vol. 10. Elsevier, 353–425.

Shapiro, A. 2006. On complexity of multistage stochastic programs. *Operations Research Letters* **34**(1) 1–8.

Shapiro, A. 2009. On a time-consistency concept in risk averse multistage stochastic programming. *Operations Research Letters* **37** 143–147.

Shapiro, A., D. Dentcheva, A. Ruszczyński. 2009. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM.

Shi, Q., J. Petterson, G. Dror, J. Langford, A. Smola, S.V.N. Vishwanathan. 2009. Hash kernels for structured data. *Journal of Machine Learning Research* **10** 2615–2637.

Shivaswamy, P., T. Jebara. 2010. Empirical Bernstein boosting. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2010)*. JMLR Workshop and Conference proceedings (vol. 9), 733–740.

Simard, P., Y. Le Cun, J. Denker, B. Victorri. 1998. Transformation invariance in pattern recognition – Tangent distance and tangent propagation. G.B. Orr, K.-R. Müller, eds., *Neural Networks: Tricks of the Trade*, vol. LNCS 1524. Springer, 239–274.

Simon, H. 1956. Rational choice and the structure of the environment. *Psychological Review* **63** 129–138.

Slyke, R. Van, R.J.-B. Wets. 1969. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics* **17** 638–663.

Solak, E., R. Murray-Smith, W.E. Leithead, D.J. Leith, C.E. Rasmussen. 2003. Derivative observations in Gaussian Process models of dynamic systems. *Advances in Neural Information Processing Systems 15 (NIPS-2002)*. 1033–1040.

Sonnenburg, S., G. Rätsch, C. Schäfer, B. Schölkopf. 2006. Large scale multiple kernel learning. *Journal of Machine Learning Research* **7** 1531–1565.

Speed, T.P., H.T. Kiiveri. 1986. Gaussian Markov distributions over finite graphs. *The Annals of Statistics* **14** 138–150.

Spielman, D., S.-H. Teng. 2004. Smoothed analysis: Why the simplex algorithm usually takes polynomial time. *Journal of the Association for Computing Machinery* **51** 385–463.

Spjøtvold, J., P. Tøndel, T.A. Johansen. 2007. Continuous selection and unique polyhedral representation of solutions to convex parametric quadratic programs. *Journal of Optimization Theory and Applications* **134** 177–189.

Stein, C. 1956. Inadmissibility of the usual estimator for the mean of a multivariate distribution. *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1. 197–206.

Steinwart, I., A. Christman. 2008. *Support Vector Machines*, chap. Kernels and Reproducing Kernel Hilbert Spaces. Springer, 111–164.

Stone, M. 1974. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society* **36** 111–147.

Strotz, R.H. 1955. Myopia and inconsistency in dynamic utility maximization. *The Review of Economic Studies* **23** 165–180.

Sutton, R.S., A.G. Barto. 1998. *Reinforcement Learning, an introduction*. MIT Press.

Tikhonov, A.N., V.Y. Arsenin. 1977. *Solutions of ill posed problems*. W.H. Winston and Sons (distributed by Wiley).

Tøndel, P., T. Johansen, A. Bemporad. 2003. Evaluation of piecewise affine control via binary search tree. *Automatica* **39** 945–950.

Van Hentenryck, P., R. Bent. 2006. *Online stochastic combinatorial optimization*. MIT Press.

Vapnik, V.N. 1998. *Statistical Learning Theory*. Wiley.

von Neumann, J., O. Morgenstern. 1947. *Theory of games and economic behavior*. 2nd ed. Princeton University Press.

Wainwright, M.J., M.I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning* **1** 1–305.

Walker, A.M. 1969. On the asymptotic behaviour of posterior distributions. *Journal of the Royal Statistical Society* **31** 80–88.

Walkup, D., R.J.-B. Wets. 1967. Stochastic programs with recourse. *SIAM Journal on Applied Mathematics* **15** 1299–1314.

Walkup, D., R.J.-B. Wets. 1969a. Lifting projections of convex polyhedra. *Pacific Journal of Mathematics* **28** 465–475.

Walkup, D., R.J.-B. Wets. 1969b. Stochastic programs with recourse II: On the continuity of the objective. *SIAM Journal on Applied Mathematics* **17** 98–103.

Wallace, S.W., R.J.-B. Wets. 1992. Preprocessing in stochastic programming: The case of linear programs. *ORSA Journal of Computing* **4** 45–49.

Wets, R.J.-B. 1974. Stochastic programs with fixed recourse: The equivalent deterministic program. *SIAM Review* **16** 309–339.

Wets, R.J.-B., C. Witzgall. 1967. Algorithms for frames and lineality spaces of cones. *Journal of Research of the National Bureau of Standards* **71** 1–7.

Weyl, H. 1935. Elementare Theorie der konvexen Polyeder. *Commentarii Mathematici Helvetici* **7** 290–306.

White, H. 1982. Maximum likelihood estimation of misspecified models. *Econometrica* **50** 1–25.

Yosida, K. 1980. *Functional Analysis*. Sixth ed. Springer.

Zhang, Z., M.I. Jordan, D.-Y. Yeung. 2009. Posterior consistency of the Silverman g-prior in Bayesian model choice. *Advances in Neural Information Processing Systems 21 (NIPS-2008)*. 1969–1976.

Index