Decision and regression tree ensemble methods and their applications in automatic learning

Louis Wehenkel

Department of Electrical Engineering and Computer Science
Centre of Biomedical Integrative Genoproteomics

University of Liège

IAP Study Day - Colonster - May 19, 2005

Find slides: http://montefiore.ulg.ac.be/∼lwh/

UNIVERSITÉ de Liège

UNIVERSITÉ de Liège

Part I

Some applications

Steal-mill control

Wide area control of power systems

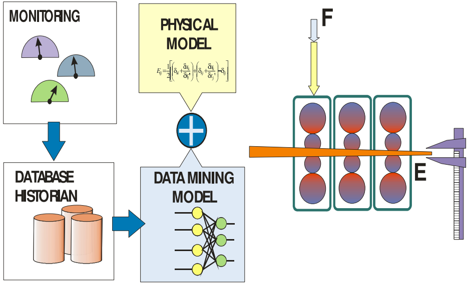
Computer vision based quality control

Proteomics biomarker identification

Part II: Methods

Steal-mill control

(ULg, PEPITe, ARCELOR)



Problem

- Pre-setting of steel-mill controller
- Improve friction force model

Approach

- Collect data from process measurements
- Determine error of physical model
- Automatically learn blackbox model of prediction error
- Combine physical and blackbox model to predict friction forces
- Adaptive pre-setting reduces waste

Wide area control of power systems (ULg, PEPITe, Hydro-Québec)

Problem

- Improve emergency control scheme
  - Churchill-Falls power plant
- Reduce probability of blackout

Approach

- 10,000 real-time snapshots sampled (several years)
- Massive time-domain simulations
- Automatically learn decision rules to determine optimal amount of generation and load to trip
- Implement rules in real-time
- New rules enhance security

Vision based quality control

(EC Project FINDER)



Problem

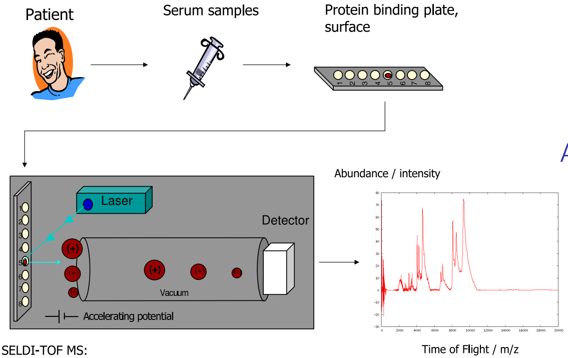
- Car light reflector manufacturing
- Quality control of aesthetic defects

Approach

- Robotics (handling of reflectors)
- Computer vision (defect detection)
- Extraction of images of defects (10000×300)
- Expert classification into 15 classes
- Build classifiers by automatic learning
- Integration into automatic QC system

Medical diagnosis

(CBIG/GIGA collaboration)



SELDI-TOF MS:

Surface Enhanced Laser Desorption/ Ionisation Time of Flight Mass Spectrometry

Problem

- Diagnosis of Rhumatoid Arthritis and other inflammatory diseases

Approach [GFd+04]

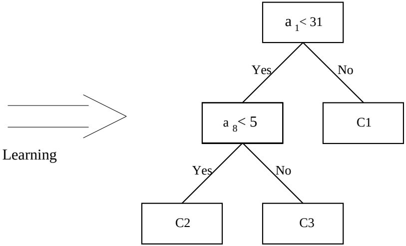
- Proteomic analysis of serum samples
- Automatic learning to
  - identify biomarkers (protein fragments) specific of disease
  - derive classifier for medical diagnosis

Part II

Supervised learning algorithm (Batch Mode)

- Inputs: learning sample ls of (x, y) observations ($ls \in (X \times Y)^*$)
- Output: a model $f_A^{ls} \in \mathcal{F}_A \subset Y^X$ (decision tree, MLP, ...)

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	Y
60	19	18	17	0	1	1	1	C1
60	3	22	23	1	29	11	23	C1
75	9	2	1	3	77	46	3	C1
2	10	10	2	234	0	0	0	C2
3	7	9	18	5	0	0	0	C2
2	14	5	10	8	10	8	10	C3
65	3	20	21	2	0	1	1	?



NB. $x = (a_1, \ldots, a_n)$

- Objectives:
  - maximise accuracy on independent observations
  - interpretability, scalability

Induction of single decision/regression trees (Reminder)

- Algorithm development (1960-1995)
  - Top-down growing of trees by recursive partitioning
    - local optimisation of split score (square-error, entropy)
  - Bottom-up pruning to prevent over-fitting
    - global optimisation of complexity vs accuracy (B/V tradeoff)
- Characterization
  - Highly scalable algorithm
  - Interpretable models (rules)
  - Robustness: irrelevant variables, scaling, outliers
  - Expected accuracy often low (because of high variance)
- Many variants and extensions
  - ID3, CART, C4.5, C5 ...
  - oblique, fuzzy, hybrid ...

Bias/variance decomposition (of average error)

Accuracy of models produced by an algorithm in a given context

- Assume problem (inputs X, outputs Y, relation $P(X,Y)$)
  and sampling scheme (e.g. fixed size $LS \sim P^N(X,Y)$).
- Take model error function (e.g. $Err_{f,Y} \equiv E_{X,Y}\{(f(X)-Y)^2\}$)
  and evaluate *expected* error of algo A (i.e. $\overline{Err}_{A,Y} \equiv E_{LS}\{Err_{f_A^{LS},Y}\}$)

$$\boxed{\text{We have } \overline{Err}_{A,Y} - Err_{B,Y} = Bias_A^2 + Var_A}$$

where

- B is the best possible model (here, $B(x) \equiv E_{Y|x}\{Y\}$)
- $Bias_A^2 = Err_{\overline{f}_A,B}$ ($\overline{f}_A(x) \equiv E_{LS}\{f_A^{LS}(x)\}$)
- $Var_A = \overline{Err}_{A,\overline{f}_A}$ (dependence of model on sample)

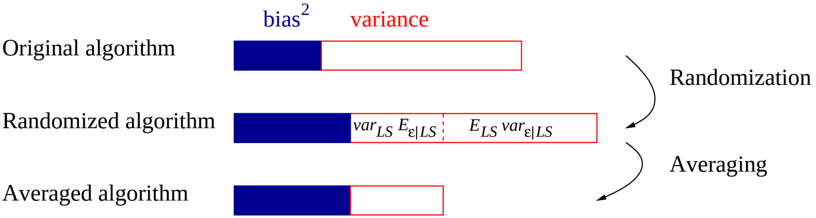
Ensembles of trees (How?/Why?)

- Perturb and Combine paradigm (1990-2005)
  - Build several (M) trees (e.g. $M = 100$, by randomization)
  - Combine trees by voting, averaging... (i.e. aggregation)
- Characterization
  - Can preserve scalability (+ trivially parallel)
  - Does not preserve interpretability
  - Can preserve robustness (irrelevant variables, scaling, outliers)
  - Can improve accuracy significantly
- Many generic variants (Bagging, Stacking, Boosting, ...)
- Non-generic variants (Random Forests, Random Subspace, ...)

Variance reduction by randomization and averaging

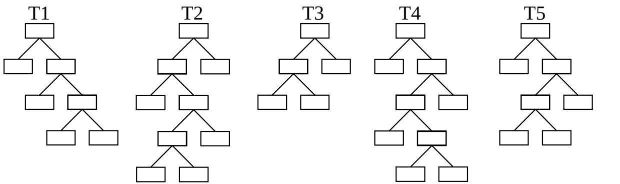
Denote by $f_A^{ls,\varepsilon}$ randomized version of A (where $\varepsilon \sim U[0,1)$)

M averaged models: $f_{A,T}^{ls,\epsilon} = M^{-1} \sum_{i=1}^{M} f_A^{ls,\epsilon_i}$ (in the limit $f_{A,\infty}^{ls}$)



Can reduce *Variance* strongly, without increasing too much *Bias*.

Extra-Trees: overall learning algorithm



- Ensemble of trees $T_1, T_2, \ldots, T_M$ (generated independently)
- Random splitting (choice of attribute and cut-point)
- Trees are fully developed (perfect fit on *ls*)
- Ultra-fast ($\sqrt{n}N \log N$)

(Presentation based on [Geu02, GEW04])

Extra-Trees: node splitting algorithm (for numerical attributes)

Given a node of a tree and a sample S corresponding to it

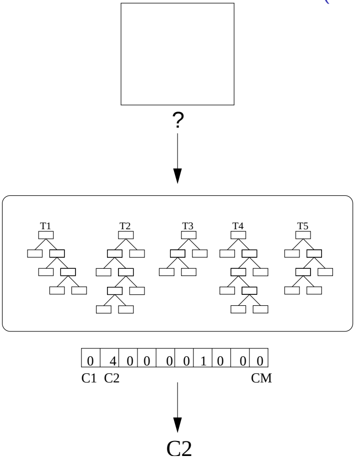
- Select K attributes (i.e. input vars) $\{a_1, \ldots, a_K\}$ at random;
- For each a_i (draw a split at random)
  - Let $a^S_{i,\min}$ and $a^S_{i,\max}$ be the min and max values of a_i in S;
  - Draw a cut-point $a_{i,c}$ uniformly in $]a^S_{i,\min}, a^S_{i,\max}]$;
  - Let $t_i = [a_i < a_{i,c}]$.
- Return a split $t_i = \arg\max_{t_i} \text{Score}(t_i, S)$.

NB: the node becomes a LEAF

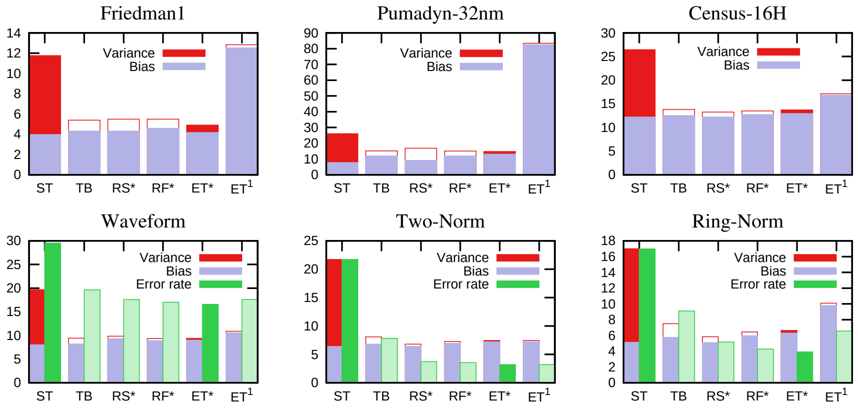
- if $|S| < n_{\min}$;
- if all attributes are constant in S;
- if the output is constant in S;

Extra-Trees: prediction algorithm

- Aggregation (majority vote or averaging)

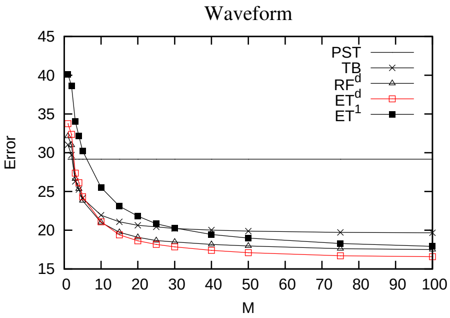


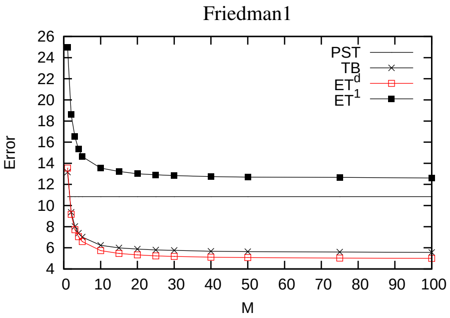
Bias/variance tradeoff (of Extra-Trees models with $M = 100$)



Parameters (of the Extra-Trees learning algorithm)

Averaging strength M





Kernel interpretation of trees

(assuming fully developed trees)

- Kernel defined by a single tree T:

  $K_T(x, x') = 1$ (or 0) if x and x' belong (or not) to same leaf

- Model defined by a single tree T: $(ls = ((x^1, y^1), \ldots, (x^N, y^N)))$

$$f_T(x) = \sum_{i=1}^{N} y^i K_T(x^i, x)$$

- Kernel defined by a tree ensemble $\mathcal{T} = \{T_1, T_2, \ldots, T_M\}$:

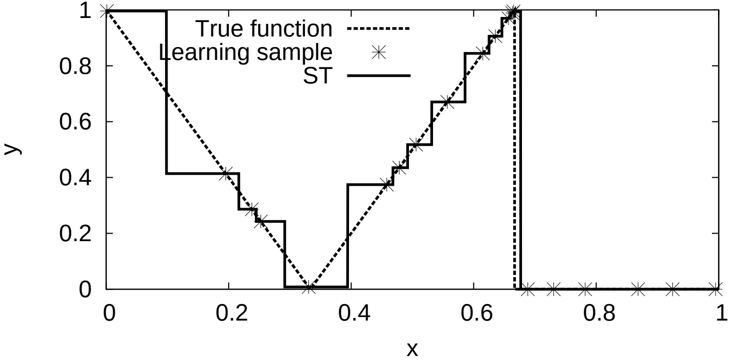
$$K_{\mathcal{T}}(x, x') = M^{-1} \sum_{j=1}^{M} K_{T_j}(x, x')$$

- Model defined by a tree ensemble $\mathcal{T}$:

$$f_{\mathcal{T}}(x) = M^{-1} \sum_{j=1}^{M} f_{T_j}(x) = \sum_{i=1}^{N} y^i K_{\mathcal{T}}(x^i, x)$$

Geometric properties

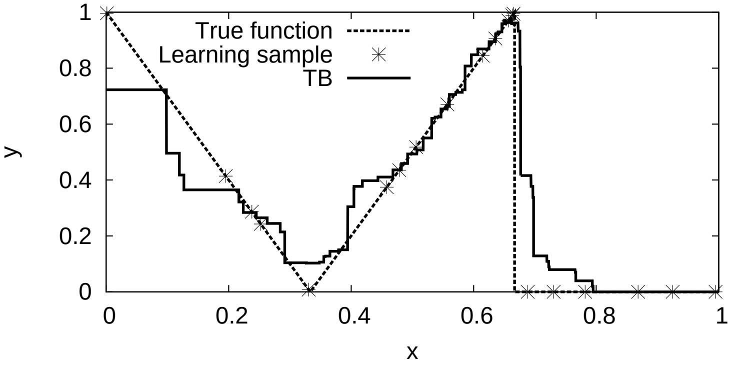
(of Single Trees)



A single fully developed CART tree.

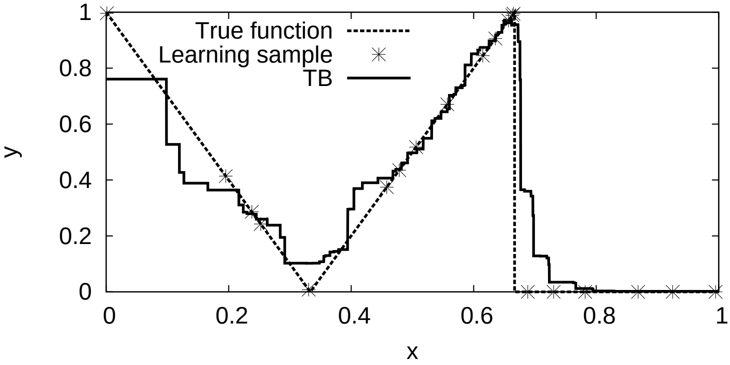
Geometric properties

(of Tree Bagging models)



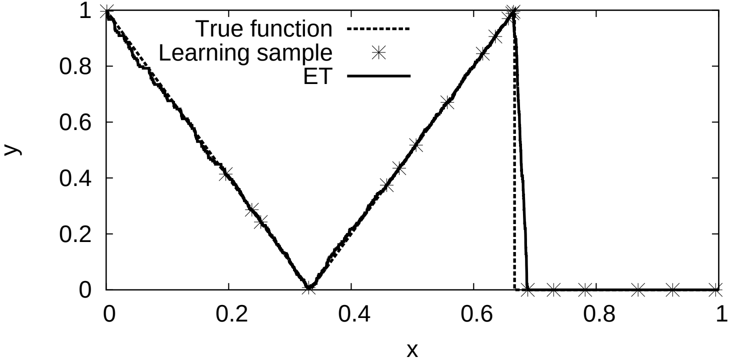
With $M = 100$ trees in the ensemble.

Geometric properties (of Tree Bagging models)



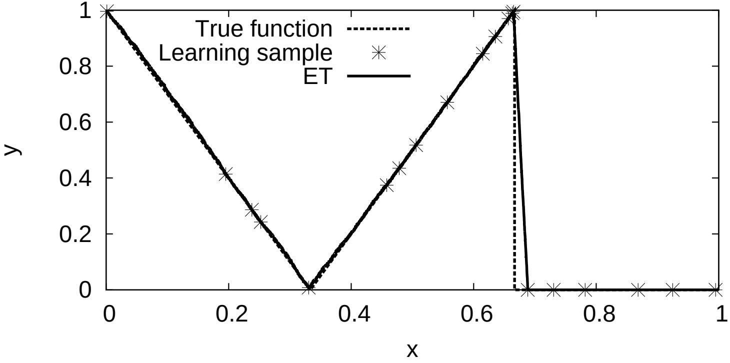
With $M = 1000$ trees in the ensemble.

Geometric properties (of Extra-Trees models)



With $M = 100$ trees in the ensemble.

Geometric properties (of Extra-Trees models)



With $M = 1000$ trees in the ensemble.

Totally randomized trees

(variant of Extra-Trees with $K = 1$)

- Select splits (attribute and cut-point) totally at random

$\Rightarrow$ Tree structures independent of sample output values $\{y^i\}$

$\Rightarrow$ Kernel tuned only on sample distribution in the input space

$\Rightarrow$ Can use the same ensemble of trees for different y-variables

$\Rightarrow$ Ultra-fast "non-supervised" learning algorithm

NB. If $K > 1$: kernel depends more strongly on $\{y^i\}$ ($CPU \propto K$)

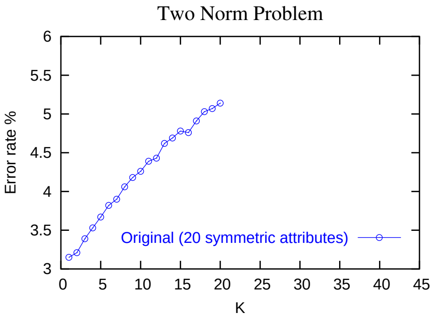
NB. Extra-Trees fit "weakly" the *ls* (Strength $\propto K$)

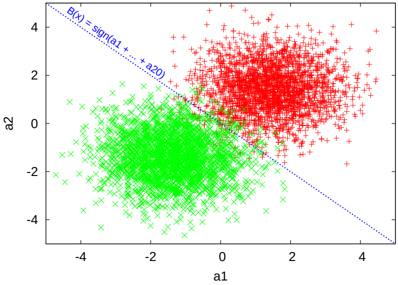
Parameters

(of the Extra-Trees learning algorithm)

Attribute selection strength K

(w.r.t. symmetries, irrelevant attributes)



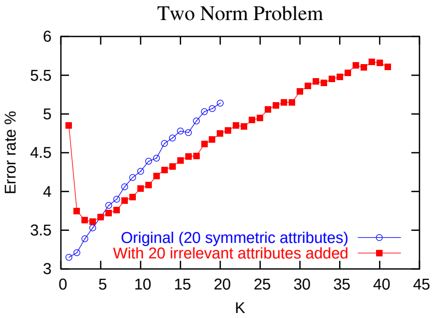


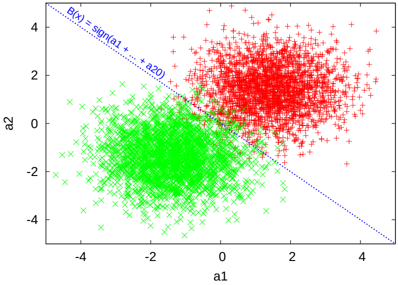
Parameters

(of the Extra-Trees learning algorithm)

Attribute selection strength K

(w.r.t. symmetries, irrelevant attributes)





Some theoretical properties of Extra-Tree models

Interpolation: $\forall(x^i, y^i) \in ls : f_{\mathcal{T}}(x^i) = y^i$ (if $n_{\min} = 2$)

Boundedness: $\forall x \in X : f_{\mathcal{T}}(x) \leqslant \max_{ls} y^i$ (convexity w.r.t. y^i)

Smoothness: continuous & pw smooth (in the limit $M \to \infty$)

Convergence: (w.r.t. $M \to \infty$)

$\forall x \in X : f_{T_i}(x)$: sequence of discrete finite iid rv.

$\forall x \in X : M^{-1} \sum_{i=1}^{M} f_{T_i}(x) \xrightarrow{a.s.} f_{\infty}(x)$. (SLLN)

Consistency conjecture: (distribution free; i.i.d. sampling; $N, M \to \infty$)

$$\boxed{\text{If } n_{\min} \text{ and } M \propto \sqrt{N}, \text{ then } f_{\mathcal{T}}^N(\cdot) \xrightarrow{i.s.s.} B(\cdot).}$$

NB. $n_{\min} \to \infty \Rightarrow$ regularisation of i/o map
$M \to \infty \Rightarrow$ cancelling of randomization variance

Generic pixel-based image classification

Challenge:

Create a robust image classification algorithm by the sole use of supervised learning on the low-level pixel-based representation of the images.

Question:

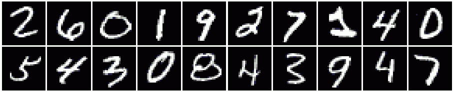
How to inject invariance (translation, scale, orientation) in a generic way into a supervised learning algorithm ?

NB: work used mainly on Extra-Trees, but other supervised learners could also be used (e.g. SVMs, K*NN*...).

(Presentation based on [MGPW04, MGPW05])

Examples

- Hand written digit recognition *(0, 1, 2, ..., 9)*



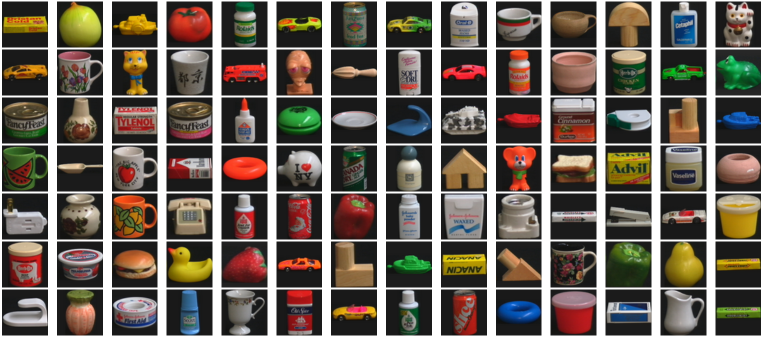
- Face classification *(Jim, Jane, John, ...)*

Examples

- Texture classification *(Metal, Bricks, Flowers, Seeds, ...)*

Examples

- Object recognition (*Cup X, Bottle Y, Fruit Z, ...*)



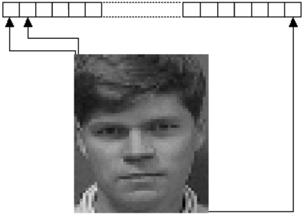
Naive solution

(global learning and prediction)

- Learning sample of N pre-classified images,

$$ls = \{(\mathbf{a}^i, c^i), i = 1, \ldots, N\}$$

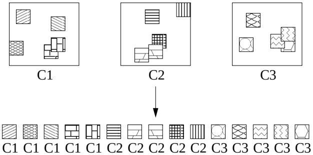
$\mathbf{a}^i$: vector of pixel values of the entire image
c^i: image class



- Prediction: same approach

Segment & Combine

(training to classify sub-windows)



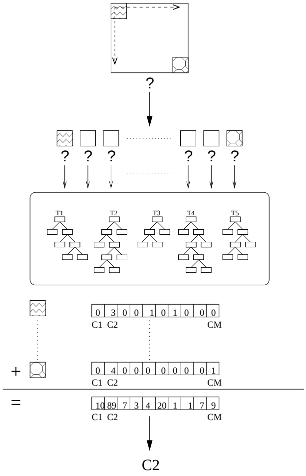
Learning sample of N_w sub-windows (size $w \times w$, pre-classified),

$$ls = \{(\mathbf{a}^i, c^i), i = 1, \ldots, N_w\}$$

$\mathbf{a}^i$: vector of pixel-values of the sub-window
c^i: class of mother image (from which the window was extracted)

Segment & Combine

(classify image by voting on sub-windows)



Datasets and protocols

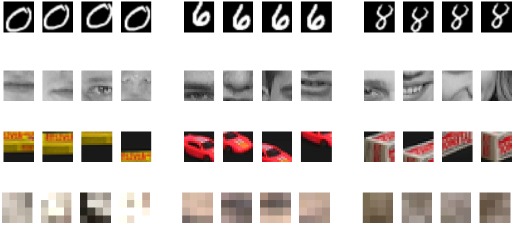
Datasets	# images	# base attributes	# classes	N_w	w
 MNIST	70000	784 $(28 * 28 * 1)$	10	360,000	24
 ORL	400	10304 $(92 * 112 * 1)$	40	120,000	20
 COIL-100	7200	3072 $(32 * 32 * 3)$	100	120,000	16
 OUTEX	864	49152 $(128 * 128 * 3)$	54	120,000	4

- MNIST: LS = 60000 images ; TS = 10000 images
- ORL: Stratified cross-validation: 10 random splits $LS = 360$; $TS = 40$
- COIL-100: LS = 1800 images ; TS = 5400 images (54 images per object)
- OUTEX: LS = 432 images (8 images per texture) ; TS = 432 images (8 images per texture)

A few results: accuracy

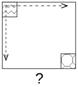
DBs	Extra-Trees Naive	Extra-Trees Segment & Combine	State-of-the-art
MNIST	3.26%	2.63%	0.5% [DKN04]
ORL	4.56% ± 1.43	1.66% ± 1.08	2.0% [Rav04]
COIL-100	1.96%	0.37%	0.1% [OM02]
OUTEX	65.05%	2.78%	0.2% [MPV02]

24×24:
20×20:
16×16:
4×4 :

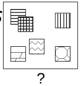


A few results: CPU times

- Learning stage: depends on parameters
  MNIST: 6h, ORL: 37s, COIL-100: 1h, OUTEX: 11m
- Prediction: depends on parameters and sub-window sampling
  - *Exhaustive (all sub-windows)*

    MNIST: 2msec, ORL: 354msec
    COIL-100: 14msec, OUTEX: 800msec
  - *Random subset of sub-windows*

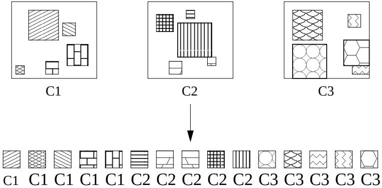
    

    MNIST: 1msec, ORL: 10msec
    COIL-100: 5msec, OUTEX: 33msec

Sub-windows of randomized size

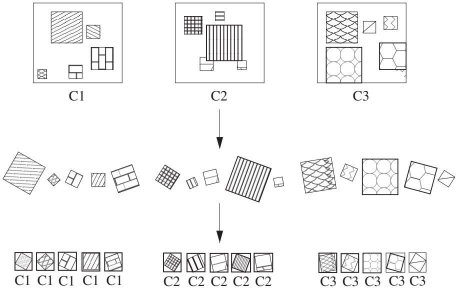
(robustness w.r.t. scale)

- Extraction of sub-windows of random size
- Rescaling to standard size



. . . and randomized orientation (more robustness)

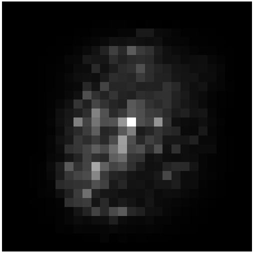
- Extraction of sub-windows of random size
- + Random rotation
- Rescaling to standard size



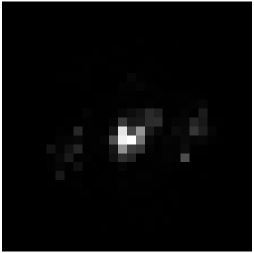
Attribute importance measures (global approach)

Compute (Shannon) information quantity brought by each pixel in each tree, and average over the ensemble of trees.

ORL (faces)



MNIST (all digits)



MNIST (0 vs 8)

Optimal control problem

(stochastic, discrete-time, infinite horizon)

$x_{t+1} = f(x_t, u_t, w_t)$ (stochastic dynamics, $w_t \sim P_w(w_t|x_t, u_t)$)

$r_t = r(x_t, u_t, w_t)$ (real valued reward signal bounded over $X \times U \times W$)

γ (discount factor $\in [0, 1)$)

$\mu(\cdot) : X \to U$ (closed-loop, stationary control policy)

$J_h^\mu(x) = E\left\{\sum_{t=0}^{h-1} \gamma^t r(x_t, \mu(x_t), w_t)|x_0 = x\right\}$ (finite horizon return)

$J_\infty^\mu(x) = \lim_{h\to\infty} J_h^\mu(x)$ (infinite horizon return)

Optimal *infinite* horizon control policy

$\mu_\infty^*(\cdot)$ that maximises $J_\infty^\mu(x)$ for all x.

(Presentation based on [EGW03, EGW05])

Batch mode reinforcement learning problem

Suppose that instead of system model $(f(\cdot,\cdot,\cdot), r(\cdot,\cdot,\cdot), P_w(\cdot|\cdot,\cdot))$, the only information we have is a (finite) sample F of four-tuples:

$$F = \{(x_{t^i}, u_{t^i}, r_{t^i}, x_{t^i+1}), i = 1, \ldots, N\}.$$

Each four-tuple corresponds to a system transition.

The objective of batch mode RL is to determine an approximation $\hat{\mu}(\cdot)$ of $\mu^*_\infty(\cdot)$ from the sole knowledge of F.

(Many one-step episodes: x_{t^i} distributed independently)

(One single episode with many steps: $x_{t^{i+1}} = x_{t^i+1}$)

(In general: several multi-step episodes)

Q-function iteration to solve Bellman equation

Idea: $\mu^*_\infty(\cdot) \equiv$ can be obtained as the limit of a sequence of optimal finite horizon (time-varying) policies.

Define sequence of value-functions Q_h and policies $\mu^*_h(t,x)$ by:

$Q_0(x,u) \equiv 0$

$Q_h(x,u) = E_{w|x,u}\{r(x,u,w) + \gamma \max_{u'} Q_{h-1}(f(x,u,w),u')\}$ $(\forall h \in \mathbb{N})$

$\mu^*_h(t,x) = \arg\max_u Q_{h-t}(x,u)$ $(\forall h \in \mathbb{N}, \forall t = 0,\ldots,h-1)$

NB: these sequences converge $(Q_h \xrightarrow{\sup} Q_\infty$ and $\mu^*_h(t,x) \xrightarrow{J^\mu_\infty} \mu^*_\infty(x))$

Alternative view: (Bellman equation)

$Q_\infty(x,u) = E_{w|x,u}\{r(x,u,w) + \gamma \max_{u'} Q_\infty(f(x,u,w),u')\}$

$\mu^*_\infty(x) = \arg\max_u Q_\infty(x,u)$

Fitted Q iteration algorithm

Idea1: replace expectation operator $E_{w|x,u}$ by average over sample
Idea2: represent Q_h by model to interpolate from samples
Supervised learning (regression): does the two in a single step

---

- Inputs:
  - a sample F of four-tuples $((x_{t^i}, u_{t^i}, r_{t^i}, x_{t^i+1}), i = 1, \ldots, N)$
  - a regression algorithm A $(A : ls \rightarrow f_A^{ls})$
- Initialisation: $\hat{Q}_0(x, u) \equiv 0$
- Iteration: (for $h = 1, 2, \ldots$)
  - Training set construction: $(\forall i = 1, \ldots N)$
    $x^i = (x_{t^i}, u_{t^i})$;
    $y^i = r_{t^i} + \gamma \max_u \hat{Q}_{h-1}(x_{t^i+1}, u)$,
  - Q-function fitting:
    $\hat{Q}_h = A(ls)$ where $ls = ((x^1, y^1), \ldots, (x^N, y^N))$

Coupling with tree-based models

Use tree-based regression as supervised learning algorithm

- Tree-based methods: boundedness $\Rightarrow$ 'non-divergence to ∞'
- Kernel independent of h: '$\Rightarrow$ convergence' (when $h \rightarrow \infty$)
- Tree structures frozen for $h > h_0 \Rightarrow$ 'convergence'

Solves at the same time

- System identification (implicitly)
- State-space discretization (and curse-of-dimensionality)
- Bellman equation (iteratively and approximately)

Generality of the framework

- No strong hypothesis on f, r (discrete, continuous, high-dimensional)
- Minimum-time problems (define $r(x,u,w) = 1_{Goal}(f(x,u,w))$)
- Stabilization problems (define $r(x,u,w) = ||f(x,u,w) - x_{ref}||$)

Academic illustration - Electric power system stabilization

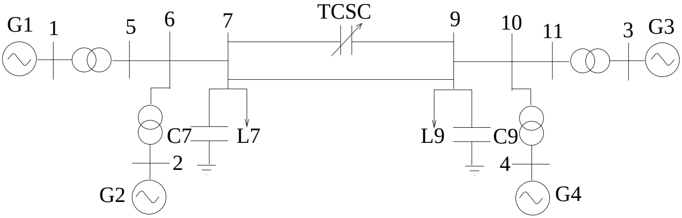


Figure: Four-machine test system (nonlinear, 60 state variables)

- Use of simulator + 1000 random episodes (60s, Δt =50ms)
- 5-dimensional $X \times U$ space; $\mathcal{F}$ contains 1100,000 four-tuples.
- "Reward": power oscillations and loss of stability ($\gamma = 0.95$)
- Policy learning by fitted Q-function iteration ($h = 100$) with Extra-Trees ($M = 50$; $K = 5$; $n_{\text{min}} = 2$)

Electric power system stabilization

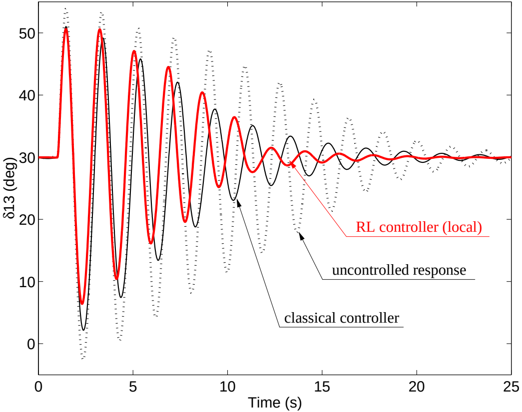


Figure: The system responses to 100 ms, self-clearing, short circuit

Electric power system stabilization

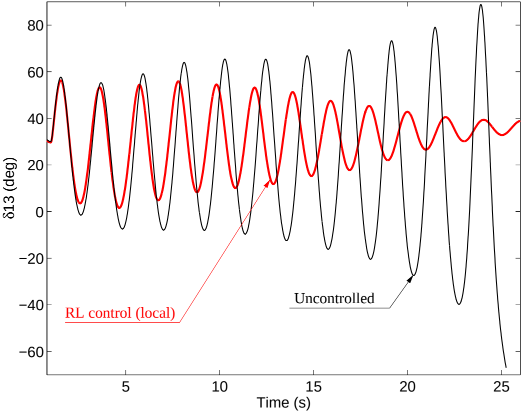


Figure: 100 ms short circuit cleared by opening line

Electric power system stabilization

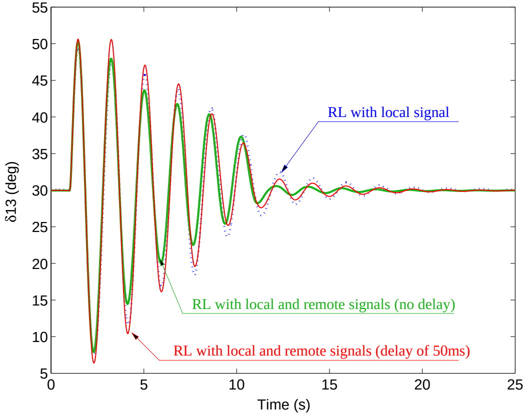


Figure: Local vs remote signals with/without communication delay

Closure - Research directions

Extra-Trees

- Theoretical analysis of randomized tree based algorithms
- Systematic handling of invariances, symmetries
- Incremental, non-supervised, semi-supervised learning

Segment and Combine

- Time-series and text classification
- Image and time-series segmentation
- Time-series forecasting

Reinforcement Learning

- Characterization w.r.t. model-based methods (e.g. MPC)
- Active learning, on-line learning and multi-agent systems
- Combination of RL & SC

Applications

- Modeling energy markets as adaptive multi-agent systems
- Exploitation of genomic and proteomic datasets
- Data mining for process control (e.g. learning from operators)

Bibliography

T. Deselaers, D. Keysers, and H. Ney.
Classification error rate for quantitative evaluation of content-based image retrieval systems.
In *Proceedings of the 7th International Conference on Pattern Recognition*, 2004.

D. Ernst, P. Geurts, and L. Wehenkel.
Iteratively extending time horizon reinforcement learning.
In *Proceedings of the 14th European Conference on Machine Learning*, 2003.

D. Ernst, P. Geurts, and L. Wehenkel.
Tree-based batch mode reinforcement learning.
Journal of Machine Learning Research, Volume 6, page 503-556 - April 2005.

P. Geurts.
Contributions to decision tree induction: bias/variance tradeoff and time series classification.
Phd. thesis, Department of Electrical Engineering and Computer Science, University of Liège, May 2002.

P. Geurts, D. Ernst, and L. Wehenkel.
Extremely randomized trees.
Submitted for publication, 2004.

P. Geurts, M. Fillet, D. de Seny, M.-A. Meuwis, M.-P. Merville, and L. Wehenkel.
Proteomic mass spectra classification using decision tree based ensemble methods.
Bioinformatics, advance access, May 2005.

R. Marée, P. Geurts, J. Piater, and L. Wehenkel.
A generic approach for image classsification based on decision tree ensembles and local sub-windows.
In *Proceedings of the 6th Asian Conference on Computer Vision*, 2004.

R. Marée, P. Geurts, J. Piater, and L. Wehenkel.
Random subwindows for robust image classification.
In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, June 2005.

T. Mäenpää, M. Pietikäinen, and J. Viertola.
Separating color and pattern information for color texture discrimination.
In *Proceedings of the 16th International Conference on Pattern Recognition*, 2002.

S. Obdržálek and J. Matas.
Object recognition using local affine frames on distinguished regions.
In *Electronic Proceedings of the 13th British Machine Vision Conference*, 2002.

S. Ravela.
Shaping receptive fields for affine invariance.
In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2004.