Chapter 8

Dual perturb and combine or soft decision trees

Dual perturb and combine is a new algorithm which, rather than generating several perturbed models at the learning stage, consists in producing the perturbed predictions at the prediction stage using only one single model derived from the learning sample. In this chapter, we develop and validate a closed form implementation of this algorithm in the context of decision trees, which yields soft decision trees. Some connections are drawn between dual P&C and other P&C algorithm like bagging. Then, the combination of these soft trees with bagging is assessed. We conclude this chapter by discussing related work on other approaches to build soft tree models.

8.1 Introduction

The generic dual perturb and combine algorithm presented in chapter 4 consists in generating perturbed predictions by perturbing the attribute vector corresponding to a test point and using a single model to produce a prediction for each of these perturbed vectors. It was introduced in Chapter 4 as an efficient alternative to bagging which needs only one model. In the present chapter, we analyze this method in the context of decision trees. We will see that with respect to other P&C techniques, dual P&C has the further advantage of preserving the interpretability of decision tree by making a prediction with only one model.

In Section 8.2.2, we show that the general dual P&C algorithm recalled in Section 8.2.1 is, in the case of decision trees, equivalent to a method that softens the decisions at test nodes. Instead of taking the "hard" (or crisp) decision of propagating an object to the right or to the left successor of a test node, this object is propagated with dual P&C in both directions with some probability. This vision allows to compute efficiently the asymptotic prediction corresponding to an infinite number of perturbations of an attribute vector. Depending on the value chosen for the noise level (the meta-parameter of the dual P&C algorithm), the sharpness of these soft decisions may be adjusted. The automatic choice of the noise level is thus discussed in Section 8.2.3 and the resulting method is validated in Section 8.3. The soft tree vision also allows to draw connections between dual P&C and traditional perturb an combine techniques. However, despite these connections, our experimental results show that dual P&C and bagging have also some complementary behavior and hence their combination may be a further interesting step; this idea is validated in Section 8.4.1. The last section of this chapter provides a discussion of other soft decision tree methods found in the literature.

8.2 Proposed algorithm

8.2.1 Generic dual P&C algorithm

The idea behind dual P&C is very simple: a certain number of perturbed versions of the attribute vector $\underline{a}$ of a test instance are produced. The model (induced in the standard way from the original learning sample) is applied to these vectors resulting in a set of predictions which are aggregated to obtain the final prediction. Considering numerical attributes only, a natural choice to perturb the attribute vector is to add a zero-mean Gaussian noise to it. Thus, from a model f_{ls}, the prediction of dual perturb and combine at a point $\underline{a}$ is given by:

$$f_{DPC}(\underline{a}) = \mathrm{aggr}_{i=1}^{T} f_{ls}(\underline{a} + \underline{\epsilon}^i), \tag{8.1}$$

where aggr is one of the aggregation operators also used in the context of other P&C methods (averaging or majority vote) and $\underline{\epsilon}^i$ are realizations of a random vector $\underline{\varepsilon} = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m)$, where ε_i are independent random variables distributed according to a Gaussian law $\varepsilon_i \sim N(0, \lambda_i.\sigma_i)$, with σ_i the standard deviation of the attribute A_i in the learning set and λ_i the noise level on this attribute. In our experiments, we have simplified this by using the same level of noise, $\lambda_i = \lambda$, for all the attributes, which has the advantage of minimizing the number of meta-parameters of the algorithm. Although many other perturbation schemes could be imagined, this is the only one considered in this dissertation.

In regression, this procedure with finite T actually computes an approximation of:

$$f_{DPC}(\underline{a}) = E_{\underline{\varepsilon}}\{f_{ls}(\underline{a} + \underline{\varepsilon})\} \tag{8.2}$$

In classification, the majority class for T growing to infinity is given by:

$$f_{DPC}(\underline{a}) = \arg\max_{c} E_{\underline{\varepsilon}}\{1(f_{ls}(\underline{a} + \underline{\varepsilon}) = c)\}, \tag{8.3}$$

and, when the aggregation operator consists in averaging conditional class probability estimates $\hat{P}_{ls}(c|\underline{a})$, the asymptotic value of 8.1 is:

$$f_{DPC}(\underline{a}) = \arg\max_{c} E_{\underline{\varepsilon}}\{\hat{P}_{ls}(c|\underline{a} + \underline{\varepsilon})\} \tag{8.4}$$

In the next section, we will present a closed-form implementation of this algorithm in the context of decision tree induction which computes exact expectations according to $\underline{\varepsilon}$ under some simplifying hypothesis.

8.2.2 Closed-form dual P&C of decision trees

The general form of dual P&C can be particularized when models f_{ls} are decision trees. Denoting by $\mathcal{L}_j$ $(j = 1, \ldots, L)$ the leaves of a tree and by g_j, $(j = 1, \ldots, L)$ some numerical predictions associated to these leaves, the prediction given by this tree at a point $\underline{a}$ can be written as:

$$g(\underline{a}) = \sum_j 1(\underline{a} \to \mathcal{L}_j) \cdot g_j, \tag{8.5}$$

where $1(\underline{a} \to \mathcal{L}_j)$ is the characteristic function of the region of the attribute space covered by $\mathcal{L}_j$. On the other hand, the asymptotic prediction given by dual perturb and combine is:

$$\begin{aligned} g_{DPC}(\underline{a}) &= E_{\underline{\varepsilon}}\{g(\underline{a} + \underline{\varepsilon})\} && (8.6) \\ &= \sum_j E_{\underline{\varepsilon}}\{1(\underline{a} + \underline{\varepsilon} \to \mathcal{L}_j)\} \cdot g_j && (8.7) \\ &= \sum_j P_{\underline{\varepsilon}}(\underline{a} + \underline{\varepsilon} \to \mathcal{L}_j) \cdot g_j, && (8.8) \end{aligned}$$

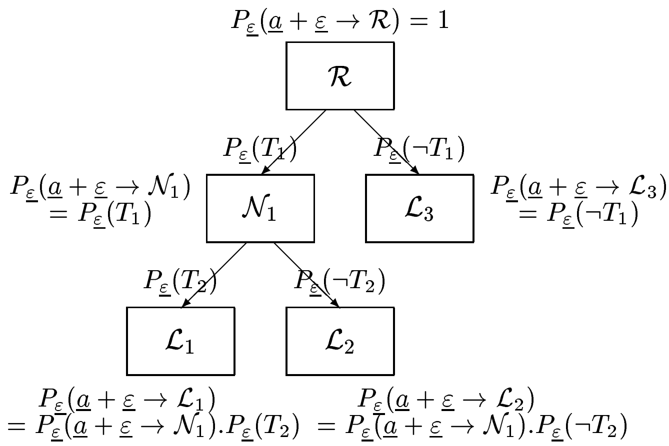


Figure 8.1: Example leaves distribution computation

where $P_{\underline{\varepsilon}}(\underline{a}+\underline{\varepsilon}\to\mathcal{L}_j)$ is thus the probability that a perturbation of $\underline{a}$ reaches the leaf $\mathcal{L}_j$. Denoting by T_1,T_2,...,T_{N_j} the tests along the path connecting the root node to the leaf $\mathcal{L}_j$, the probability that a perturbed case $\underline{a}$ reaches this leaf is the probability that all tests are verified:

$$P_{\underline{\varepsilon}}(\underline{a}+\underline{\varepsilon}\to\mathcal{L}_j) = P_{\underline{\varepsilon}}(T_1(\underline{a}+\underline{\varepsilon}) \wedge T_2(\underline{a}+\underline{\varepsilon}) \wedge \ldots \wedge T_{N_j}(\underline{a}+\underline{\varepsilon})). \tag{8.9}$$

But tests are based on only one attribute and attribute values are perturbed independently (ε_i are independent random variables). So, assuming there is no more than one test using the same attribute along each branch of the tree, equation (8.9) can be factorized into:

$$P_{\underline{\varepsilon}}(\underline{a}+\underline{\varepsilon}\to\mathcal{L}_j) = P_{\underline{\varepsilon}}(T_1(\underline{a}+\underline{\varepsilon}))\cdot P_{\underline{\varepsilon}}(T_2(\underline{a}+\underline{\varepsilon}))\cdots P_{\underline{\varepsilon}}(T_{N_j}(\underline{a}+\underline{\varepsilon})). \tag{8.10}$$

Furthermore, each test T is of the form $[A_i < (\geq) a_{th}]$ where A_i is some attribute and hence the probability of the test being true for a perturbation of the vector $\underline{a}$ is computed by:

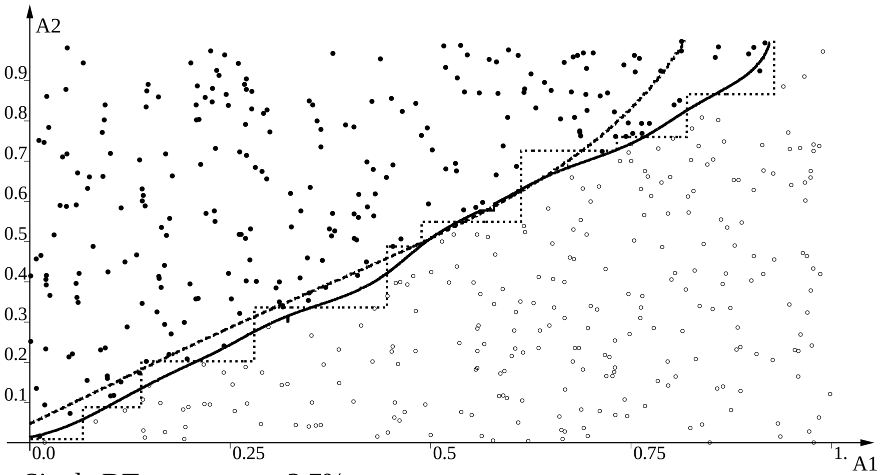
$$P_{\underline{\varepsilon}}(T(\underline{a}+\underline{\varepsilon})) = P_{\underline{\varepsilon}}(a_i+\varepsilon_i < (\geq) a_{th}) = P(Z < (\geq)\frac{a_{th}-a_i}{\lambda\sigma_i}), \tag{8.11}$$

where Z is a $N(0,1)$ random variable.

For each test node, the probability expressed in (8.11) can be obtained by elementary table look-up. On the other hand, the computation of $E_{\underline{\varepsilon}}\{g(\underline{a}+\underline{\varepsilon})\}$ can be done efficiently using a forward-backward propagation scheme (see Figure 8.1) : in the forward pass information about the attributes of an object is sent from the root node to the leaves in the form of probabilities of belonging to the traversed nodes, starting with a probability of 1.0 at the root and multiplying this probability by the probabilities associated to the arcs which are traversed; in the backward pass information about the predicted output variable is sent back towards the root with test-nodes aggregating information received from their successors. The complexity of this algorithm (which recursively factorizes equation (8.8)) is thus proportional to the tree complexity.

Substituting g_i for the appropriate predictions in (8.6), this algorithm allows to compute efficiently the values of (8.2), (8.3), or (8.4). For example, in the next section, we will make predictions according to the average of conditional class probability estimates. So the probability of class c for a case $\underline{a}$ will be estimated by:

$$\hat{P}_{DPC}(c|\underline{a}) = E_{\underline{\varepsilon}}\{\hat{P}_{ls}(c|\underline{a}+\underline{\varepsilon})\} = \sum_j P_{\underline{\varepsilon}}(\underline{a}+\underline{\varepsilon}\to\mathcal{L}_j).\hat{P}(c|\mathcal{L}_j), \tag{8.12}$$



..... Single DT - error rate : 3.7%

— Dual P&C, noise level=0.28 - error rate 1.45%

--- Dual P&C, noise level=1.5 - error rate 4.6%

Figure 8.2: Decision tree boundaries with different noise levels

where $\hat{P}(c|\mathcal{L}_j)$ is the class c probability estimate at the leaf $\mathcal{L}_j$.

This closed-form implementation has several advantages over the general dual perturb and combine algorithm which consists in averaging a finite number T of predictions obtained from a set of perturbed attribute vectors. First, it makes dual perturb and combine deterministic while the general form is dependent on the random choice of perturbation vectors $\underline{\epsilon}^i$. Since the general dual P&C algorithm computes a sample estimate of the expectation which is computed by the closed-form algorithm, the prediction given by the general algorithm for finite T will suffer from a larger variance than the closed-form algorithm and hence its average accuracy should be lower, especially if T is small. In terms of computational efficiency, the computation of (8.12) (which is proportional to the tree complexity) requires more time than the computation of one prediction with a classical tree (which is proportional to the tree depth). For small T, the general approach could be more efficient while the second one is better when large values of T are considered. Although it would be interesting to study this tradeoff between computational efficiency and accuracy with respect to T, we will nevertheless focus in our experiments on the closed-form implementation which is more appealing in terms of accuracy.

8.2.3 Impact of the noise level on bias and variance

Before discussing the way to choose an appropriate noise level, let us see on a simple example what is the effect of dual P&C with different values of the noise level. Suppose we want to separate empty circles from full circles on Figure 8.2. This figure shows the partitioning done by a single tree and what this partitioning becomes when using dual P&C of the same tree with an optimal noise level of $\lambda = 0.28$ and with a higher noise level of $\lambda = 1.5$. The smoothing ability of dual P&C is pretty clear from this figure and also the over-smoothing when increasing too much λ.

The effect of the noise level may also be approached by a study of bias and variance. Figure 8.3 shows the effect of the noise level λ on the two-norm problem, left on Tibshirani's bias and variance terms and right on regression bias and variance of probability estimates. These graphs are representative of other problems as well. In terms of regression, the bias is monotonically increasing with the value of λ while the variance is monotonically

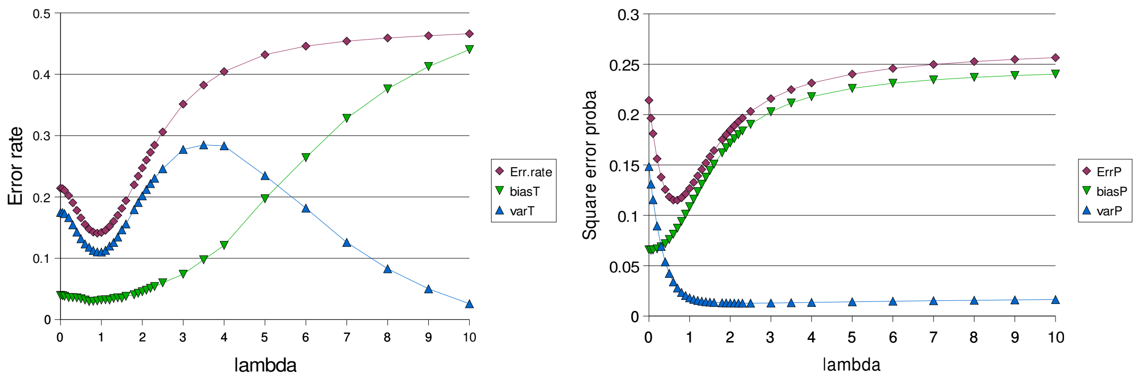


Figure 8.3: Evolution of error, bias and variance with the noise level on the two-norm problem. Left, with error rate, right with square error.

decreasing and this happens whatever the datasets. Indeed, the more we extend the region over which the predictions are averaged, the less flexible is the regression function which can be represented and hence bias increases and variance decreases (at some point, it starts slightly increasing). This compromise is very similar to the compromise which appears in k-NN with the number k of neighbors.

The evolution of Tibshirani's bias and variance terms is more difficult to explain. First, small values of λ reduce both variance and bias, thus resulting in an overall decrease of error rates. When λ further increases, both bias and variance reach a minimum, and start increasing again. Then, while bias is monotonically increasing, the variance goes through a maximum and then starts decreasing. This apparently strange behavior can nevertheless be explained. As trees are unpruned, class probability estimates, $\hat{P}_{DPC}(C|\underline{a})$, given by dual P&C with no noise are either 0 or 1 (tree leaves are mostly pure). When the noise increases, probability estimates move away from these extreme values. For points of the input space which are well classified by the original tree, the regression bias inevitably increases while the classification bias remains unchanged as long as the estimates do not go on the other side of the decision boundaries (corresponding to 0.5 in the two class case). On the other hand, some biased points become unbiased and both their regression and classification bias decrease. As bias is rather small and thus the number of biased points is much smaller than the number of unbiased points, the overall effect is an increase of the regression bias. On the other hand, at the beginning, the classification bias decreases until the number of newly biased points exactly balances the number of newly unbiased points. When the probability estimates get near 0.5, even small regression variance of the probability estimates can make the classification change from one tree to another and hence the classification variance increases. When we further increase λ, probabilities $P_{\underline{\varepsilon}}(\underline{a}+\underline{\varepsilon} \rightarrow \mathcal{L}_j)$ and hence $\hat{P}_{DPC}(C|\underline{a})$ becomes more and more independent of the object. Indeed, asymptotically, we have $P_{\underline{\varepsilon}}(\underline{a}+\underline{\varepsilon} \rightarrow \mathcal{L}_j) = (0.5)^{N_j}$, where N_j is the depth of the leaf $\mathcal{L}_j$. Hence, $\hat{P}_{DPC}(C|\underline{a})$ becomes an average of the classification over all the leaves and the value of the majority class is highly dependent on the exact tree but is constant allover the input space. Thus, there is a high number of biased points and hence bias is dominating the error. On the other hand, as the "variance" of biased points is in fact negative, the overall effect is a decrease of variance with large value of λ.

All in all, this study shows that there exists an optimal value of the noise level. Many experiments on several datasets (not reported here for the sake of brevity) show that the optimal value of λ is highly dependent of the induced model and learning task and also not related to the learning set size. So, one possible way to determine its value is to

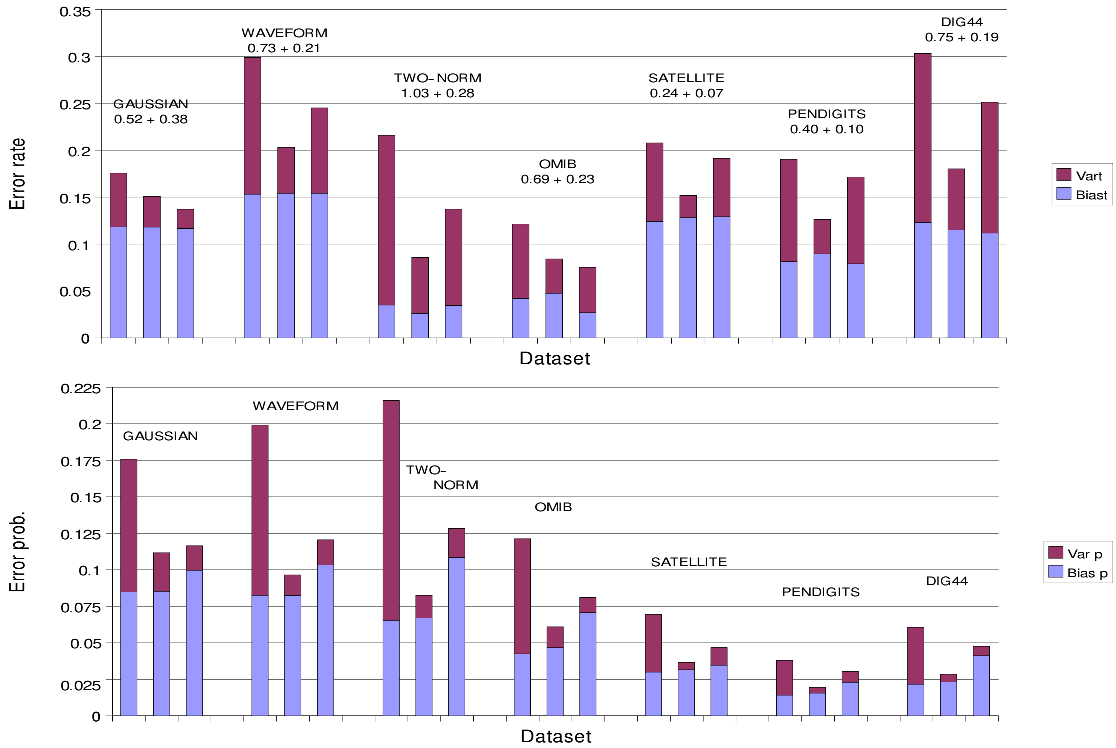


Figure 8.4: Comparison of single trees (left), bagging (center), and, dual P&C (right) on all datasets. Top, in terms of error rates, bottom in terms of the square error of probability estimates.

Table 8.1: Average effect of dual P&C

	Mean error	compl	bias_T	var_T	var_{KW}	$Err_{\hat{P}}$	$\text{bias}^2_{\hat{P}}$	$\text{var}_{\hat{P}}$
Single tree	0.2161	90	0.0966	0.1195	0.1326	0.1257	0.0486	0.0771
Dual P&C	0.1726	90	0.0931	0.0796	0.0989	0.0816	0.0686	0.0130
Bagging (25 trees)	0.1402	1631	0.0969	0.0434	0.0689	0.0624	0.0502	0.0121

devote a validation set to this task. As the curve of error rate is typically convex with only one minimum (like in Figure 8.3), a simple minimization method will be appropriate to determine the optimal value of λ. In our experiments, we use a simple optimization by dichotomy.

8.3 Empirical study

The next step is of course to carry out experiments with dual P&C on our seven classification problems. To this end, we use the closed-form implementation of dual P&C described in section 8.2.2 and average of conditional class probabilities (equation 8.4). In these experiments, the trees are not pruned, so as to minimize bias, and the optimal value of λ is determined individually for each tree from a separate validation set which size is given in Table B.1. Again, even if the assumption of availability of a validation set is not realistic in practice, the goal of these experiments is mainly to study the effect of dual P&C on decision trees in an optimal setting. Figure 8.4 compares dual P&C with bagging and single trees on all datasets. Table 8.1 shows average results.

Dual perturb and combine decreases significantly mean error rates. This decrease of

Table 8.2: Computation times of several variants on the two-norm problem, LS size 5000 and TS size 5000

	compl	Growing (s)	Testing (s)	Error(%)
Classical (full) tree	671	19.4	0.4	15.84
Bagging (25 trees)	11809	290.4	5.5	4.3
Dual P&C	671	49.4	13.3	6.72

error rates comes from a decrease of variance (by about 33%) and a very slight decrease of the classification bias. In average, bagging does better than dual perturb and combine in reducing the variance. Dual P&C decreases the variance of probability estimates almost as well as bagging but at the same time, it increases very strongly the regression bias and thus yields also less accurate estimates of conditional class probabilities than bagging. On two problems however (Gaussian and Omib), dual P&C gives lower error rates than bagging mainly because of a stronger reduction of variance. In the next section, we will try to explain in which situations dual perturb and combine may be superior to bagging.

The average optimal value of λ determined from the validation set and its standard deviation are given beside the name of the dataset in Figure 8.4. Mean values are very different from one dataset to another and on some problems, the standard deviation is quite large, showing that the optimal value is also very dependent on the particular decision trees. Actually, the optimal λ, just like decision tree, suffers from a high variance in some cases.

In terms of computational efficiency, there is an overhead during learning to determine the optimal value of λ. The determination of this value requires several tests of the tree (with different values of λ) using the forward-backward algorithm of Section 8.2.2. The computation of a prediction according to this latter algorithm takes also more time than the traditional propagation of an object in a decision tree (corresponding to $\lambda = 0$). Indeed, the former is generally proportional to the tree complexity while the latter is proportional to the average tree depth. To give an indication of the computation times, Table 8.2 compares dual P&C with a single tree and a set of 25 bagged trees in terms of growing and testing times. Like in the previous chapter, the model is built on the two-norm problem from a learning sample of 5000 cases and tested on a test sample of the same size. In the case of dual P&C, the time for growing includes the construction of the decision tree (which, according to the first line, takes 19.4s) and the determination of the noise level on an independent sample of size 1000 (which, in this particular case, requires 10 tests of the tree). This Table shows that testing a tree with dual P&C is much slower than testing a classical tree. On this problem, the determination of the value of λ is actually the dominant part of the induction stage. Nevertheless, the resulting method is still much faster than bagging and in this particular case, it decreases error rates substantially with respect to a single tree. Furthermore, even if the determination of λ is slow, it is not very demanding in terms of computer resources since objects can be considered sequentially.

8.4 Relationship with other methods

In this section, we discuss the relationships which exist on the one hand between dual P&C and other P&C algorithms and, on the other hand, between dual P&C and other soft tree models. The first connection gives some explanation of the good behavior of the dual perturb and combine algorithm (which works like a model averaging technique) and the second one gives a possible explanation of the good behavior of soft tree models (which, like dual P&C, have lower variance than decision trees).

8.4.1 Connection with model averaging techniques

As tree tests on numerical attributes are of the form $[A < a_{th}]$, adding Gaussian noise to the attribute vector before propagating a test case in the tree is more or less equivalent to adding Gaussian noise to the discretization thresholds and leaving the test case unchanged. Actually, this is strictly equivalent if there is no more than one test on the same attribute along each branch of the tree (otherwise, thresholds on the same attribute should be perturbed in the same way for the two approaches to be equivalent). Hence for decision trees, dual P&C is very similar to a traditional perturb and combine algorithm restricted to randomizing discretization thresholds.

On the other hand, focusing on discretization thresholds, bagging amounts to generate different decision trees, using the different samples generated by the bootstrap technique, which differ mainly in the precise value of the thresholds. Thus, the bootstrap sampling distribution induces a discretization threshold sampling distribution at each test-node. Consequently, when averaging the predictions of the set of trees created in this way, we essentially average predictions according to the joint sampling distribution of the thresholds at the different test-nodes of the tree.

To illustrate this idea, let us consider the case of a decision stump in a system with only one input attribute A. Neglecting the variance of probability estimates in this example amounts to assume that all the predictions attached to the left nodes (resp. right nodes) are independent of the particular instance of the bootstrap sample (and hence identical to those obtained with the full sample LS). Denoting by g_l and g_r, these (assumed) common values, the aggregated prediction of the bagged stump is given by:

$$g_{BAG}(\underline{a}) = \frac{1}{T}\sum_{i=1}^{T}[1(a \leq a_{th}(ls_i^b)).g_l + 1(a > a_{th}(ls_i^b)).g_r], \tag{8.13}$$

where ls_i^b denotes the i^{th} bootstrap sample from ls and $a_{th}(ls_i^b)$ is the threshold induced from this sample. So, as T grows to infinity, this expression converges to:

$$g_{BAG}(\underline{a}) = P_{LS^b}(a \leq a_{th}(LS^b)).g_l + P_{LS^b}(a > a_{th}(LS^b).g_r, \tag{8.14}$$

where P_{LS^b} denotes the probability according to a bootstrap sample. This equation has to be compared to the prediction of dual perturb and combine particularized to this simple case (from equation (8.8) and (8.11)):

$$g_{DPC}(\underline{a}) = P_{\underline{\varepsilon}}(a \leq a_{th} + \underline{\varepsilon}).g_l + P_{\underline{\varepsilon}}(a > a_{th} + \underline{\varepsilon}).g_r. \tag{8.15}$$

So at a first (rough) approximation, both bagging and dual perturb and combine appear to replace a crisp decision tree by a soft one. In the case of bagging, the thresholds are softened according to their sampling distribution estimated by bootstrap. In the case of dual perturb and combine, thresholds are softened according to a Gaussian distribution. For illustration, Figure 8.5 draws three distributions on the omib problem: $P_{LS^b}(a \leq a_{th}(LS^b))$ given by bagging, $P_{\underline{\varepsilon}}(a \leq a_{th} + \underline{\varepsilon})$ given by dual P&C, and also $P_{LS}(a \leq a_{th}(LS))$, i.e. the true threshold distribution (estimated from a high number of learning sets drawn from the pool). Note that both the distribution obtained with bagging and the one obtained by dual P&C are quite far from the true distribution. While the closed-form of dual P&C gives a perfectly continuous distribution, the one induced by bagging remains discontinuous even for a large number T of bootstrap samples since the number of candidate thresholds is limited by the size of the learning set.

Bagging and dual P&C thus work the same way if the tree structure is fixed and probability estimates are not changed. Of course, this is a rough approximation and, in

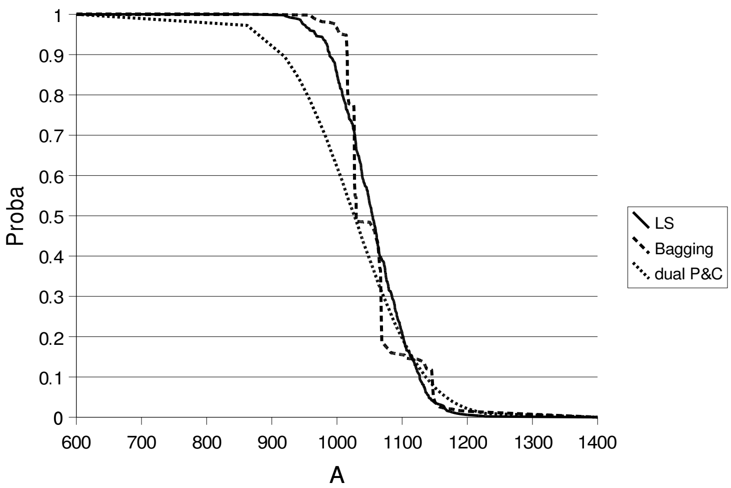


Figure 8.5: Discretization threshold distributions on the omib problem (attribute PU)

Table 8.3: Comparison of structure fixed bagging and dual P&C

	Mean error	compl	bias_T	var_T	var_{KW}	$Err_{\hat{P}}$	$\text{bias}^2_{\hat{P}}$	$\text{var}_{\hat{P}}$
Dual P&C	0.1726	90	0.0931	0.0796	0.0989	0.0816	0.0686	0.0130
Bagging (25 trees)	0.1402	1631	0.0969	0.0434	0.0689	0.0624	0.0502	0.0121
Str. fixed Bag.	0.1861	2168	0.0970	0.0883	0.1084	0.0766	0.0496	0.0270

the case of bagging predictions at leaf nodes may change from one bootstrap sample to another and the structure of the trees is also highly variable. To confirm this statement, we carried out another experiment with a variant of bagging which leaves the structure fixed. A maximal tree is first built from the learning set ls and its structure is used to guide the induction from 25 bootstrap samples. The average results obtained with this method on all datasets are compared in Table 8.3 with bagging and dual P&C (under the name "structure fixed bagging"). The mean error of this variant is not as good as bagging but it is close to the mean error of dual P&C. Their classification bias and variance are also quite close. Nevertheless, dual perturb and combine works better than structure fixed bagging. Indeed, in dual P&C, the noise level on discretization is adapted to the learning set while bagging introduces this noise by bootstrap sampling and hence is less flexible. The adaptation of the noise level allows to exchange some regression bias for a reduction of the regression variance which leads to a more important reduction of the classification variance. The counterpart is that structure fixed bagging gives slightly more reliable probability estimates essentially because it only slightly increases the regression bias.

So, contrary to bagging dual P&C only fights variance coming from the discretization threshold. This analysis suggests that dual perturb and combine will give similar results to bagging if the number of relevant attributes is small and the attribute choice variance is small. In our experiments, the two problems where dual perturb and combine is better than bagging, Gaussian and omib, precisely correspond to the two problems with the smallest number of attributes. On the Gaussian problem, there are only two candidate attributes and the classification frontier is symmetric in the input space. On the omib problem, two attributes among the six ones are especially useful for classification. Another consequence is that it is also likely that the error of dual perturb and combine should come closer to the error of bagging if the learning set size is increased since larger learning set

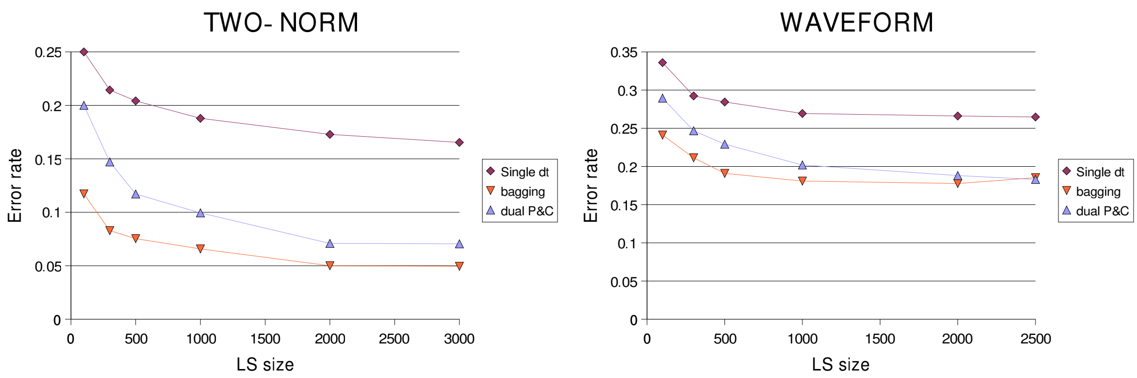


Figure 8.6: Dual P&C versus bagging with respect the the LS size. Left, on two-norm, right, on waveform

size allows deeper trees and thus allows to take into account more attributes. To check this, we carried out experiments on the two-norm and waveform datasets for increasing learning set sizes. Figure 8.6 plots average error rates of single tree, dual P&C and bagging for increasing learning set size. For each size, the plotted values are the mean error (estimated from the test sample) over 10 learning sets drawn from the pool sample (the same samples are used by each method). On both problems, the error rates of dual P&C get near to the error rates of bagging as the learning set size increases. On the waveform problem, dual P&C even yields better results than bagging for larger samples. These results are also confirmed by the results presented in [Geu01a] and [GOW01] where we use larger learning sample sizes and indeed dual P&C compares more favorably to bagging.

8.4.2 Combination of dual P&C and bagging

In the previous section, dual P&C and bagging both are described as a way to soften decision trees. However, it is shown by a comparison between structure fixed bagging and dual P&C that the latter technique is better in this goal essentially because it builds really smooth models and also because of its parametric form. So, it could be a good idea to consider a combination of the two methods.

There are several possible combinations of dual P&C with bagging. For example, we could build several trees from bootstrap samples and optimize a different value of λ for each one of these trees before aggregating their predictions, i.e. apply bagging on the top of dual P&C. In this section, we rather choose to apply dual P&C on the top of bagging because it requires to optimize only one parameter λ.

When the output of the model is numerical (regression variable or probability estimates), the prediction given by applying dual P&C on the top of bagging is the following:

$$E_{\underline{\varepsilon}}\{\frac{1}{T}\sum_{i=1}^{T} f_{ls_i}(\underline{a}+\underline{\varepsilon})\} = \frac{1}{T}\sum_{i=1}^{T} E_{\underline{\varepsilon}}\{f_{ls_i}(\underline{a}+\underline{\varepsilon})\} \tag{8.16}$$

and thus is equivalent to applying bagging on the top of dual P&C trees which use the same value of the perturbation level. However, in classification with the majority vote aggregation operator, there is no such equivalence since generally:

$$\arg\max_c E_{\underline{\varepsilon}}\{1([\arg\max_c \sum_{i=1}^{T} 1(f_{ls_i}(\underline{a}+\underline{\varepsilon})=c)]=c)\}$$

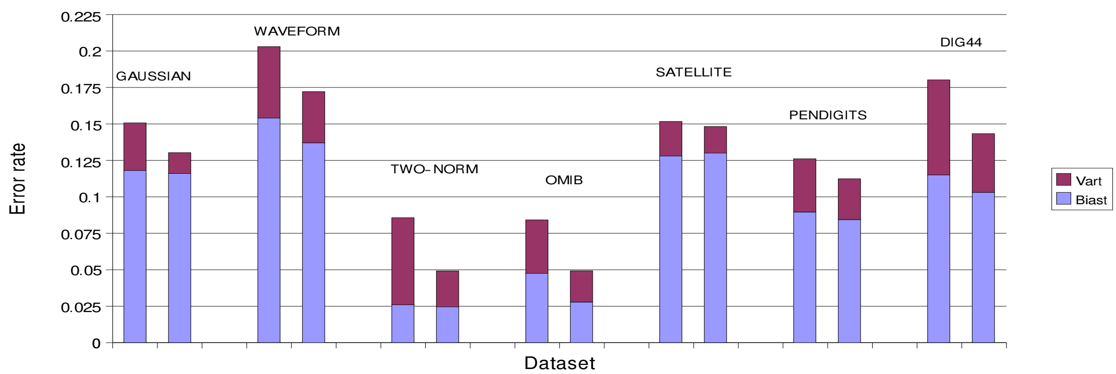


Figure 8.7: Comparison of bagging (left), and, bagging with dual P&C (right) on all datasets in terms of error rates.

Table 8.4: bagging with dual P&C

	Mean error	compl	bias_T	var_T	var_{KW}	$Err_{\hat{P}}$	$\text{bias}^2_{\hat{P}}$	$\text{var}_{\hat{P}}$
Bagging (25 trees)	0.1402	1631	0.0969	0.0434	0.0689	0.0624	0.0502	0.0121
Bagging + dual P&C	0.1150	1569	0.0889	0.0260	0.0468	0.0770	0.0727	0.0043

$$\neq \arg\max_c \sum_{i=1}^{T} 1([\arg\max_c E_{\underline{\varepsilon}}\{1(f_{ls_i}(\underline{a}+\underline{\varepsilon})=c)\}]=c)$$

So, in our experiment, we will continue to aggregate predictions by averaging conditional class probability estimates, since this will allow us to use the closed-form implementation of dual P&C in the context of decision trees. Like for single trees, the optimal value of λ is determined by cross-validation from the validation set whose size is described in Table 5.2.

Figure 8.7 compares bagging with bagging in combination with dual P&C on all datasets. Table 8.4 summarizes average results. On all datasets, the combination improves results with respect to bagging (less significantly on the satellite dataset). The improvement comes from a decrease of both classification bias and variance. The decrease in classification variance is by 35% with respect to bagging alone and it is more than 75% with respect to single trees. Unsurprisingly, the combination gives worse probability estimates than bagging because of an increase of the bias. Nevertheless, the regression variance of these estimates is now almost null with respect to their bias. So it is the ability of dual P&C to increase the regression bias for the benefit of the regression variance that makes the two approaches actually complementary.

8.4.3 Other soft tree models

The soft tree model which results here from the application of dual P&C to decision trees is not new in machine learning. Actually, several methods have been proposed that are somehow related to soft trees.

In the machine learning community, Carter and Catlett [CC87] noticed that a decision tree misclassifies more often examples which are close to the discretization thresholds and so propose to propagate such examples to both successors of a test node and weight the predictions of the corresponding subtrees according to the distance of these examples to the discretization threshold. Quinlan [Qui86] following them provides in its C4.5 algorithm a (quite rudimentary) way to soften discretization thresholds. Dietterich and Kong [DK95] observe that this variant of C4.5 indeed reduces the classification variance of decision trees while increasing their classification bias. In [Fri96], Friedman points out the main drawbacks of the recursive partitioning of decision tree growing: there are more errors at the classification boundaries, few attributes can be taken into account along a branch because of the fast decrease in learning sample size, and the parameters of the cutting are subject to high variance. To circumvent these problems, he proposes a method which recursively divides the learning sample into overlapping subsets and uses voting schemes to aggregate competing predictions (just like soft trees).

Markov tree models [Jor94] and hierarchical mixtures of experts (HME) [JJNH91] use a probabilistic framework to justify soft decisions. In such models, a test object goes to the left and right successors of a test node with probabilities that depends on the input attributes. In HMEs, tests are extended to take into account linear combinations of several attributes and linear models are further attached to terminal nodes. Because of the soft probabilistic decisions, the prediction (a probability distribution on the output variable) given by such a model is continuous with respect to variations of its parameters and hence optimization can be carried out by techniques that use gradient computation (in this probabilistic context, this optimization algorithm takes the form of the popular Expectation-Maximization or EM algorithm). These authors however do not provide an algorithm that learns the structure of the HME and often, a classical tree growing algorithm is used and the resulting tree is softened before optimization.

Soft trees have also been introduced in the context of fuzzy logic under the name fuzzy decision trees. In a fuzzy decision tree [BW99, MCSL01], the weight used to propagate an object to the left and right successors of a test node is interpreted as a fuzzy membership degree to a fuzzy set. Two approaches have been studied to build a fuzzy decision tree. Like for HME, it is possible to build a crisp tree, soften the threshold using fuzzy logic and then globally optimize the parameters of this model (e.g. [MCSL01]). There exist also algorithms which directly integrate fuzzy techniques during the growing stage (e.g. [OW02]) by dividing at each step the learning sample into overlapping subsets (like in [Fri96]).

Eventually, some connections exist also between soft tree models and neural networks. Indeed, the probability of our soft test (8.11) may be represented by a function:

$$P(Z < \frac{a_{th} - a_i}{\lambda\sigma_i}) = \Phi(\frac{1}{\lambda\sigma_i}.a - \frac{a_t h}{\lambda\sigma_i}), \tag{8.17}$$

which is the output of a perceptron with a integrated Gaussian activation function. It is thus possible to combine such simple units into several layers of a neural network to give a model which computes the same function as a soft tree. Some researchers have proposed to transform a decision tree to a neural networks and then use optimization techniques available in the context of neural networks [Set95]. This parallel between soft trees and neural networks has also stimulated the use of optimization techniques developed in the neural networks community (which are also related to the EM algorithm) to optimize fuzzy decision trees (see [MCSL01] or [OW02]).

With respect to all these works, the originality of our approach is to present soft trees as the particularization to decision tree of a general variance reduction technique. With respect to all these works, the next obvious step to improve the accuracy of our algorithm is to independently optimize the noise levels and discretization thresholds used to propagate objects. Nevertheless, good care should be taken in this generalization step. First, although the increase of flexibility provided by a full optimization of the parameters certainly reduces the bias, it is not sure that this will not increase the variance with respect to our algorithm which only makes use of one parameter and still improves very significantly the accuracy of decision trees. And, indeed, we know for example that complex neural networks also suffer from a high variance. Second, with further sophistications, soft trees begin to share several shortcomings of neural networks type of models: the parameters resulting from a global optimization are instable, models are not interpretable anymore, and global optimization is very computationally demanding. In this prospect, we believe that algorithms which still grow soft trees in a recursive and efficient way (for example [OW02]) are more appealing that global optimization techniques.

8.5 Conclusion

Dual P&C is an interesting alternative to bagging in the context of decision tree induction. Although not as good as bagging, it still gives substantial improvements of error rates. At the same time, it saves most of the efficiency of the method and the interpretability of decision trees. Furthermore, it can be combined with bagging where it gives the best results of all the methods considered in this thesis on our classification problems. Nevertheless, the algorithm requires the determination of (only) one parameter, the noise level. Since the optimal noise level seems to be very dependent on the particular model and hence learning sample, a separate validation set seems to be the only reasonable way to select its value.

There remain several open questions about this method. First, in combination with decision trees, we could study the impact of pruning on accuracy. The method could also be combined with extra-trees or boosted trees which are better than bagging. More interestingly, it would be useful to find a way to learn the noise level directly from the learning sample, or at least to give another interpretation of the noise level which would make it less dependent on the particular learning sample. More generally, we could consider different ways of perturbing the attribute vector, for example by using different noise levels for different attributes. Knowledge about the application problem and hence about the interpretation of the attributes may also be used to imagine different perturbation schemes. We will give an application of this idea in Chapter 11 in the context of time series classification with the segment and combine algorithm. Another interesting direction of research is the application of the general dual perturb and combine algorithm in combination with other learning algorithms. Preliminary studies show that this method can improve the accuracy when applied on the top of neural networks for example. In this context, it could be interesting to consider the derivation of a closed-form implementation similar to the one derived here for decision trees.