Dual Perturb and Combine Algorithm

Pierre Geurts[*]
Electrical and Computer Engineering Dept. - University of Liège
Institut Montefiore - Sart-Tilman B28 - B4000 Liège - Belgium
geurts@montefiore.ulg.ac.be

Abstract

In this paper, a dual perturb and combine algorithm is proposed which consists in producing the perturbed predictions at the prediction stage using only one model. To this end, the attribute vector of a test case is perturbed several times by an additive random noise, the model is applied to each of these perturbed vectors and the resulting predictions are aggregated. An analytical version of this algorithm is described in the context of decision tree induction. From experiments on several datasets, it appears that this simple algorithm yields significant improvements on several problems, sometimes comparable to those obtained with bagging. When combined with decision tree bagging, this algorithm also improves accuracy in many problems.

1 INTRODUCTION

The bias/variance tradeoff is a well-known problem in machine learning. Bias relates to the systematic error component, whereas variance relates to the variability resulting from the randomness of the learning sample. Both contribute to prediction errors. There exist different means to reduce the variance of a machine learning method and hence increase the accuracy of the model. Besides regularization and complexity control (e.g. pruning) which also affect bias, a more drastic method is model averaging. Model averaging consists in aggregating the predictions made by several models. Most of the time, these models are obtained by perturbing the learning set in several ways (bagging (Breiman, 1994), boosting (Freund & Schapire, 1995), by randomization of outputs of learning cases (Breiman, 2000)...), but also by perturbing the model (option trees (Buntine, 1992), randomization (Dietterich, 2000)...). Breiman (1996) has proposed the label P&C (for Perturb and Combine) to designate this group of methods.

Bagging (Breiman, 1994) for example consists in drawing a certain number of bootstrap samples from the original learning set, build a model from each of them and then aggregate the predictions provided by these models on a new test case to yield the final prediction. Bagging has been shown to reduce very strongly variance while in most cases leaving unchanged the bias of the method and so is mostly effective in conjunction with unstable predictors like decision trees which present high variance. Actually, focusing on classification, bagging works by smoothing the classification frontier. A set of perturbed predictions on a particular point of the input space is obtained by perturbing the learning set. Averaging these predictions gives a smoothed prediction which is more stable (less variable, and hence less often wrong) than isolated predictions. Although very effective, this process assumes to construct and to save a set of models and thus is very time and memory consuming.

In this paper, we propose a new technique to stabilize the predictions made by a model. Unlike bagging, our algorithm makes use of only one model and delays at the prediction stage the generation of multiple predictions which are averaged. Thus, one of the main advantages of our method is that it preserves the interpretability and the computational efficiency of the initial learning method. This technique shows to consistently improve predictive accuracy of decision trees and, in several problems, yields competitive results with respect to those obtained by bagging. Furthermore, its generality makes it possible to combine it with bagging where it appears to give also significant improvement. The general algorithm and an analytical version of it in the context of decision trees are presented in Section 2. In the same section, we propose an interpretation of the algorithm and a discus-

[*]Research Fellow, FNRS

sion about the determination of its single parameter. Experiments are summarized in Section 3. Section 4 concludes and discusses future work.

2 DUAL P&C

The dual idea of perturb and combine is to delay the generation of perturbed predictions at the prediction stage (e.g. when testing) by using different modified versions of a test case with the same model. Based on this idea, we propose in this paper a new way to use a model. A certain number of perturbed versions of the attribute vector of the tested object are produced. The model (induced from the original learning set) is applied to these vectors resulting in a set of predictions which are aggregated to obtain the final prediction. Like model averaging, dual P&C aims at reducing the prediction variance by averaging different predictions. However, unlike model averaging, this method requires only one model and hence is much more efficient from the computational point of view.

Obviously, dual P&C is a wrapper technique which, like bagging, boosting, meta-cost ... , may be applied to any machine learning technique.

2.1 GENERAL ALGORITHM

In this paper, we will restrict ourself to numerical attributes and to classification problems (although the idea could also be applied to regression and/or to symbolic attributes). To perturb an object, we naturally propose to add to each numerical attribute value a zero-mean Gaussian noise. Let us denote by $f_{LS}(.)$ a model extracted from a learning set LS and by $\mathbf{x} = (x_1, x_2, \dots, x_m)$ the attribute vector of an object. The prediction given by our method for this object will be:

$$aggr_{i=1}^{n} f_{LS}(\mathbf{x} + \epsilon^i), \tag{1}$$

where $aggr$ is an aggregation operator and ϵ^i are realizations of a random vector $\epsilon = (\epsilon_1, \epsilon_2, \dots, \epsilon_m)$, where ϵ_i are independent random variables distributed according to a Gaussian law $\epsilon_i \sim N(0, \lambda_i.\sigma_i)$, with σ_i the standard deviation of the attribute x_i in the learning set and λ_i the noise level on this attribute. In this paper, we use the same level of noise, λ, for all the attributes. Possible aggregation operators are similar to those proposed for model averaging in the context of classification, i.e. majority vote or maximum average conditional class probability if the model provides probability estimates.

2.2 ANALYTICAL DUAL P&C IN THE CONTEXT OF DECISION TREES

Tests on numerical attributes at decision tree nodes usually are of the following form:

$$X_i < x_{th}, \tag{2}$$

where X_i is an attribute and x_{th} is a discretization threshold for this attribute. So, in the context of decision tree induction with numerical attributes, adding Gaussian noise to the attribute vector is more or less equivalent to adding Gaussian noise to the discretization thresholds (strictly equivalent if there is no more than one test on the same attribute along each branch of the tree). Hence for decision trees, dual P&C is very similar to traditional P&C restricted to randomizing discretization thresholds.

This observation allows us also to compute the asymptotic value of the prediction corresponding to an infinite number of random propagations ($n \to \infty$ in eqn. (1)). Indeed, assuming a Gaussian noise on the threshold, we can compute the probability of a perturbed new case going to each successor of a test node. For instance, if x_i is the value of the attribute X_i for this new case, the probability of the test (2) to be true is computed by:

$$P(x_i + \epsilon_i < x_{th}) = P(Z < \frac{x_{th} - x_i}{\lambda_i \sigma_i}), \tag{3}$$

where Z is a $N(0,1)$ random variable. Further, we can also compute the probability that a new case reaches a particular leaf of the tree. Indeed, denoting by $\mathcal{L}_j$ a leaf of the tree and by T_1,T_2,...,T_{N_j} the tests along the path connecting the root node to this leaf, the probability that a case $\mathbf{x}$ reaches this leaf is

$$P(\mathbf{x} \to \mathcal{L}_j) = P(T_1 \wedge T_2 \wedge \dots \wedge T_{N_j} | \mathbf{x}). \tag{4}$$

Since the noise variables ϵ_i are independent, this yields the product:

$$P(\mathbf{x} \to \mathcal{L}_j) = P(T_1|\mathbf{x}) P(T_2|\mathbf{x}) \cdots P(T_{N_j}|\mathbf{x}), \tag{5}$$

where the $P(T_i|\mathbf{x})$ are computed according to (3). This yields a probability distribution on the leaves of the tree and hence the final prediction by aggregating the predictions at each leaf taking into account this distribution. In what follows, we will make predictions according to the average of class conditional probability estimates. So the probability of class C for a case $\mathbf{x}$ will be estimated by:

$$\hat{P}(C|\mathbf{x}) = \sum_j P(\mathbf{x} \to \mathcal{L}_j).\hat{P}(C|\mathcal{L}_j), \tag{6}$$

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

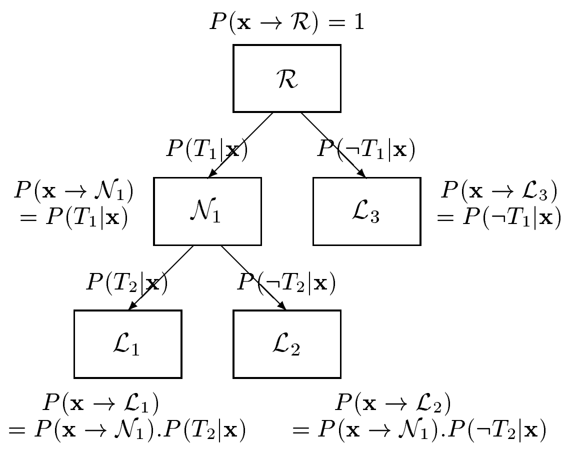
$P(\mathbf{x} \to \mathcal{R}) = 1$
$\mathcal{R}$
$P(T_1|\mathbf{x})$
$P(\neg T_1|\mathbf{x})$
$P(\mathbf{x} \to \mathcal{N}_1) = P(T_1|\mathbf{x})$
$\mathcal{N}_1$
$\mathcal{L}_3$
$P(\mathbf{x} \to \mathcal{L}_3) = P(\neg T_1|\mathbf{x})$
$P(T_2|\mathbf{x})$
$P(\neg T_2|\mathbf{x})$
$\mathcal{L}_1$
$\mathcal{L}_2$
$P(\mathbf{x} \to \mathcal{L}_1) = P(\mathbf{x} \to \mathcal{N}_1).P(T_2|\mathbf{x})$
$P(\mathbf{x} \to \mathcal{L}_2) = P(\mathbf{x} \to \mathcal{N}_1).P(\neg T_2|\mathbf{x})$

Figure 1: Example Leaves Distribution Computation

For each test node, the probability expressed in (3) can be estimated from elementary table look-up. On the other hand, the computation of $\hat{P}(C|\mathbf{x})$ can be done efficiently using a forward-backward propagation scheme (see Figure 1) : in the forward pass information about a case is sent from the root node to the leaves 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 is sent back towards the root with test-nodes aggregating information received from their successors. The complexity of this algorithm (which recursively factorizes eqn. (6)) is thus proportional to the tree complexity. Because pruned tree sizes are generally rather small, this is very efficient in practice.

2.3 INTERPRETATION

A decision tree cuts the input space into regions where, ideally, the output variable would be constant. When tests based on numerical attributes consist in comparing attribute values to thresholds, each such region corresponds to a hyper-rectangle. Splitting thus aims at refining an initially rough partition into a finer one by splitting its component regions. Recursive partitioning has several drawbacks (see Friedman (1996)) : there are more errors near the region boundaries because of the discontinuity of the approximation, regions depend only on a few directions of the input space because of the fast decrease of learning set samples when going down in the tree, and the parameters of the cuttings are subject to high variance. Dual P&C is a simple way to partially overcome some of these disadvantages.

Like classical P&C methods, dual P&C transforms a discontinuous classification frontier into a more continuous one by averaging. Actually, the classification even becomes continuous when using analytical dual

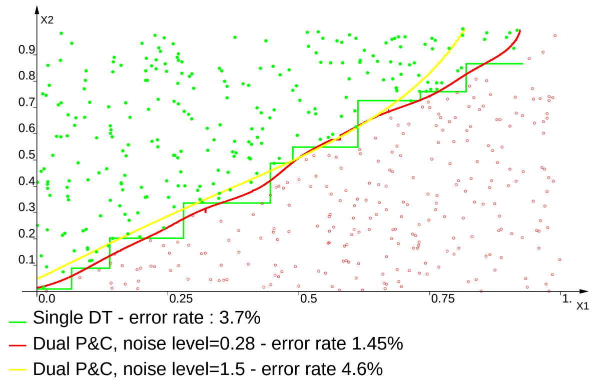

Figure 2: DT Boundaries with Different Noise Levels

P&C with decision trees (a small perturbation of the attribute vector leads to a small perturbation of the conditional class probability estimates). The classification at a particular point of the input space now depends on the training subsets of all non-zero probability leafs corresponding to this point. The strength with which each subset influences this classification depends on the distance of the point to the region frontier. So dual P&C with decision trees should reduce error rates near the classification boundaries.

Recursive partitioning depends on the choice of attribute and discretization thresholds at each node of the tree. These choices have been shown to be highly unstable, which puts decision tree induction among the machine learning methods which present the highest variance. Especially, variance on discretization thresholds for numerical attributes has been shown to be a very important source of DT variance in Geurts and Wehenkel (2000). Dual P&C may be considered as a way to take into account the uncertainty on discretization thresholds (or equivalently the uncertainty on attribute values) and thus will only be able to reduce DT variance coming from this latter source. So it should work well especially on problems where this is the main source of variance. However, unlike dual P&C, classical P&C methods can take into account all kind of variance sources, and, in particular, the variance on the attribute selection at DT nodes. So, with dual P&C, we do not expect as good results as those obtained with model averaging since our algorithm can not take into account all variance sources.

2.4 BIAS/VARIANCE AND DETERMINATION OF THE NOISE LEVEL

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

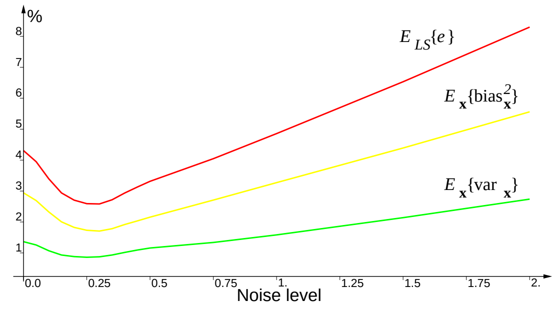

Figure 3: Bias/Variance Decomposition of $E_{LS}\{e\}$

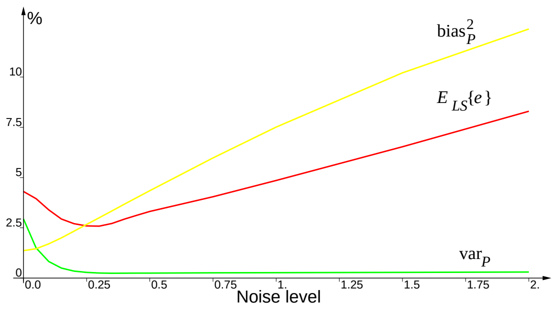

Figure 4: Bias and Variance of Probability Estimates

full circles on Figure 2. This figure shows the partitioning done by a simple 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 on this small example may also be approached by a study of bias and variance. To this end, we used the bias and variance decomposition of the mean error rate proposed by Kohavi and Wolpert (1996) as it is very close to the well-known decomposition in regression. The decomposition of the mean error is as follows:

$$E_{LS}\{e\} = E_{\mathbf{x}}\{\sigma^2_{\mathbf{x}} + \text{bias}^2_{\mathbf{x}} + \text{var}_{\mathbf{x}}\}, \qquad (7)$$

where

$$\sigma^2_{\mathbf{x}} = \frac{1}{2}(1 - \sum_{i=1}^{m} P(C_i|\mathbf{x})^2), \qquad (8)$$

$$\text{bias}^2_{\mathbf{x}} = \frac{1}{2}\sum_{i=1}^{m}[P(C_i|\mathbf{x}) - P_{LS}(C_i|\mathbf{x})]^2, \qquad (9)$$

$$\text{var}_{\mathbf{x}} = \frac{1}{2}(1 - \sum_{i=1}^{m} P_{LS}(C_i|\mathbf{x})^2). \qquad (10)$$

$P(C_i|\mathbf{x})$ is the true probability of the class C_i at the point $\mathbf{x}$ and $P_{LS}(C_i|\mathbf{x})$ is the probability that a model induced from a random learning set outputs the class C_i for a test case $\mathbf{x}$.

In our illustrative problem, there is no residual uncertainty ($\sigma^2_{\mathbf{x}} = 0$), so the mean error rate is exactly the sum of the bias and variance terms. Bias and variance are estimated from a database of 20000 cases using the following experimentation protocol: The database is split into two sets, a pool of 18000 cases and a test set of 2000 cases. 50 decision trees are built from 50 learning sets of size 500 randomly drawn from the pool. Then bias, variance and error rate are estimated by means of these 50 models for each test case using increasing values of λ. Figure 3 shows the result of this experiment. To complement this information, we also give the (à la) regression bias and variance of the class conditional probability estimates. Class probability variance is computed by:

$$\text{var}_{\hat{P}} = E_{\mathbf{x}}\{\frac{1}{m}\sum_{i=1}^{m} Var_{LS}(\hat{P}(C_i|\mathbf{x}))\}, \qquad (11)$$

while we define the class probability bias by:

$$\text{bias}^2_{\hat{P}} = E_{\mathbf{x}}\{\frac{1}{m}\sum_{i=1}^{m}(E_{LS}\{\hat{P}(C_i|\mathbf{x})\} - P(C_i|\mathbf{x}))^2\}. \qquad (12)$$

Figure 4 shows these values estimated according to the same protocol. Curves of Figures 3 and 4 are representative of what we have observed on other datasets.

As announced in section 2.3, small values of λ reduce variance (both classification variance and class probability variance). In addition, we observe also a decrease of the bias term (9), thus resulting in an overall decrease of error rates. When λ further increases, both (9) and (10) reach a minimum (at $\lambda = 0.25$), and start increasing again. On the other hand, variance of class probability estimates (11) is monotonically decreasing with increasing λ. This can be explained by looking at equation (6). When we increase λ, probabilities $P(x \to \mathcal{L}_j)$ and hence $\hat{P}(C|x)$ becomes more and more independent of the object and so the bias increases. Asymptotically, we have $P(x \to \mathcal{L}_j) = (0.5)^{N_j}$, where N_j is the depth of the leaf $\mathcal{L}_j$. Hence, $\hat{P}(C|\mathbf{x})$ becomes an average of the classification over all the leaves. Because of the exact class balance in this problem, it becomes very close to a uniform distribution, which leads to an increase of class probability bias (12). So, even very small variance of the estimates $\hat{P}(C|\mathbf{x})$ can lead to a high variability of the majority class from one tree to another. This explains the increase of classification variance with the noise level.

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

Table 2: Error Rates (%) ± Standard Deviations

Data set	DT	+ dual P&C	$\overline{\lambda}$	Bagging	+ dual P&C	$\overline{\lambda}$
Omib	9.34 ± 0.56	5.03 ± 0.99	0.61 ± 0.17	5.81 ± 0.55	3.22 ± 0.48	0.53 ± 0.15
Waveform	24.61 ± 1.64	17.54 ± 1.17	0.76 ± 0.17	18.21 ± 0.89	15.18 ± 0.84	1.09 ± 0.35
two-norm	19.08 ± 0.95	9.76 ± 1.59	1.08 ± 0.25	6.36 ± 0.84	3.26 ± 0.53	0.99 ± 0.24
VST	12.19 ± 1.49	11.28 ± 0.69	0.35 ± 0.20	9.74 ± 0.60	9.75 ± 0.55	0.50 ± 0.22
Dig44	18.49 ± 0.88	13.56 ± 1.18	0.65 ± 0.09	12.40 ± 0.38	9.47 ± 0.35	0.62 ± 0.11
Letter	25.32 ± 0.98	23.17 ± 0.80	0.34 ± 0.05	16.26 ± 1.27	14.49 ± 0.67	0.29 ± 0.06
Segment	4.30 ± 0.44	4.25 ± 0.47	0.04 ± 0.03	2.52 ± 0.24	2.84 ± 0.51	0.05 ± 0.04
Satellite	15.57 ± 0.66	14.10 ± 0.46	0.17 ± 0.05	11.46 ± 0.50	11.27 ± 0.28	0.13 ± 0.05
Pendigits	8.95 ± 0.75	7.13 ± 0.74	0.25 ± 0.08	5.42 ± 0.42	4.58 ± 0.39	0.26 ± 0.07

Table 1: Data Set Summaries

Data set	Atts	Class	GS size	PS size	TS size
Omib	6	2	2000	1000	1000
Waveform	21	3	3000	1000	1000
two-norm	20	2	1000	1000	2000
VST	136	2	2430	815	798
Dig44	16	10	3000	1000	2000
Letter	16	26	3000	1000	1000
Segment	19	7	1000	500	810
Satellite	36	6	3000	1435	2000
Pendigits	16	10	5000	2494	3498

to the learning set size. So, one possible way to determine its value is to devote a validation set to this task. As the curve of error rate is typically convex with only one minimum (like Figure 3), a simple minimization method (like the golden section search algorithm) will be appropriate to determine λ.

3 EXPERIMENTATION

Below, we experiment with the analytical version of dual P&C in the context of decision tree induction. Experiments are conducted on nine quite large datasets summarized in Table 1. All attributes are numerical. All of these datasets are available at the UCI repository (Blake & Merz, 1998) except for "omib" and "vst", two electrical power systems related datasets of our own, and two-norm, an artificial problem described in Breiman (1996).

Each dataset is first randomly split into two parts: a learning set LS and a test set TS. The test sets are kept fixed and used to determine the error rates of all models. Then each LS, is further partitioned into two sets: a growing set, GS, used to induce a decision tree and a pruning set, PS, used to prune it and to determine the value of λ. This latter step is repeated 10 times for each data set, and for each method ten models are determined using these 10 different GS/PS splits.

The second column of Table 2 gives average TS error rates of 10 DTs built (and pruned) from the 10 random splits of LS. The third and fourth columns show average error rates and values of λ when using dual P&C, using the same 10 DTs and determining the noise level for each DT from its PS. Dual P&C is also compared with 25-fold bagged trees induced from GS and pruned in a combined fashion using PS (see Geurts (2000) for a description of the pruning algorithm). The error (fifth column) is averaged over the same ten random splits of LS as for single DTs. As a last experiment, we have also tried dual P&C in conjunction with the bagged models. Average error rates and values of λ (determined from PS) are given in the last two columns of the same table.

Single DT. From this table, it appears that dual P&C is able to reduce very significantly error rates on several problems. The dependency of the noise level to a particular problem is clear from this experiment. On two problems (omib and waveform), this simple algorithm gives better results than bagging of 25 trees. However, as expected, improvement is most of the time lower than the improvement of bagging. The only data set on which there is no significant increase is the "segment" data set where the method yields an optimal value of λ close to zero.

Bagging. More surprisingly, dual P&C combined with bagging is still able to improve accuracy significantly on several problems. Only on the "vst" and "segment" databases does it slightly deteriorate accuracy, and these are also the two databases where dual P&C of single trees was least effective. Generally, the effects of dual P&C on bagged and un-bagged trees are strongly correlated : on problems where dual P&C works well with a given value of λ for single trees, the same value of λ works also well with bagged trees.

This further improvement may be related to the ability of our algorithm to transform a discontinuous frontier into a really continuous one while decision frontiers of bagging, even if smoother, remain discontinuous. However, further investigation is required to find a more definite explanation of the complementary features of the two approaches.

Computational efficiency. With respect to classical decision trees, there is an overhead during the learning stage for the determination of the optimal value of λ. Practically, this overhead corresponds to several tests of the DT with different noise levels on the pruning set. In our experiment, the number of such tests was limited to a maximum of 15. However, since the propagation algorithm is fast, this stage is not so time-consuming. In the context of bagging, however, the number of propagation tests is multiplied by the number of trees and because bagged trees tend to be significantly more complex than single trees the overhead of tuning may become more limitative. But, since the optimal value of λ for a single tree is very close to the optimal of value of λ in the context of bagging, one possible way to circumvent this problem is to determine λ on a single (and simple) tree rather than on the ensemble of complex bagged trees. The exact impact of this simplification on accuracy has still to be evaluated.

4 CONCLUSION AND FUTURE WORK

This paper has proposed a new wrapper technique which consists in smoothing model output at the prediction stage by randomly perturbing attribute values and aggregating the corresponding randomized predictions, thus yielding softer decision boundaries. An analytical version of this algorithm has been developed in combination with (bagged and unbagged) decision tree models and has been evaluated on a variety of 9 datasets. On single DTs the dual P&C approach yields significant improvements on all but one of these problems, sometimes comparable to those obtained with bagging. When combined with bagging, it further improves accuracy in 6 of the 9 problems and leaves error rates essentially unchanged in 3 of them.

Future work will be in the following directions:

- determination of the optimal noise levels. It would be nice to find a way to learn the noise level from the learning set. Also, maybe we could get better results by using different noise levels for different attributes.
- deeper study of when and why dual P&C works.
- adaptation of the method to symbolic attributes. To randomize categorical inputs, we can use similar methods than the one used in Breiman (2000) to randomize classification outputs of learning cases.
- evaluation of the method in the context of regression. We may also expect significant improvements because of the output smoothing ability of our algorithm.
- experimentation of dual P&C with different models. Can dual P&C improve accuracy of other model types (neural nets for example) and if so, is it possible to derive an analytical algorithm in combination with these models?
- proper use of this method in combination with bagging and boosting. Preliminary results show that dual P&C is also able to improve boosting. Interesting questions in this context concern the number of ensemble terms which are really necessary when using dual P&C.

References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Breiman, L. (1994). *Bagging predictors* (Technical Report). University of California, Department of Statistics.

Breiman, L. (1996). *Arcing classifiers* (Technical Report). University of California, Department of Statistics.

Breiman, L. (2000). Randomizing outputs to increase prediction accuracy. *Machine Learning*, *40*, 229–242.

Buntine, W. (1992). Learning classification trees. *Statistics and Computing*, *2*, 63–73.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*, 139–157.

Freund, Y., & Schapire, R. (1995). A decision theoretic generalization of on-line learning and an application to boosting. *Proceedings of the 2nd European Conference on Computational Learning Theory* (pp. 23–27).

Friedman, J. (1996). *Local learning based on recursive covering* (Technical Report). Department of Statistics, Standford University.

Geurts, P. (2000). Some enhancements of decision tree bagging. *Proc. of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD-2000)* (pp. 136–147). Lyon, France.

Geurts, P., & Wehenkel, L. (2000). Investigation and reduction of discretization variance in decision tree induction. *Proc. of the 11th European Conference on Machine Learning (ECML-2000), Barcelona* (pp. 162–170).

Kohavi, R., & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. *Proc. of the 13th Int. Conf. on Machine Learning.*