Enlarging the Margins in Perceptron Decision Trees

Kristin P. Bennett
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
110 8th St., Troy, NY 12180, USA
(518)276-6899
bennek@rpi.edu

Nello Cristianini
Dept of Engineering Mathematics
University of Bristol
Bristol BS8 1TR, UK
nello.cristianini@bristol.ac.uk

John Shawe-Taylor
Dept of Computer Science
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
jst@dcs.rhbnc.ac.uk

Donghui Wu
Department of Mathematical Sciences
Dept of Mathematical Sciences
Rensselaer Polytechnic Institute
110 8th St., Troy, NY 12180, USA
Troy, NY 12180, USA
wud2@rpi.edu

Abstract Capacity control in perceptron decision trees is typically performed by controlling their size. We prove that other quantities can be as relevant to reduce their flexibility and combat overfitting. In particular, we provide an upper bound on the generalization error which depends both on the size of the tree and on the margin of the decision nodes. So enlarging the margin in perceptron decision trees will reduce the upper bound on generalization error. Based on this analysis, we introduce three new algorithms, which can induce large margin perceptron decision trees. To assess the effect of the large-margin bias, OC1 [18] of Murthy, Kasif and Salzberg, a well-known system inducing perceptron decision tree, is used as the baseline algorithms. An extensive experimental study on real world data showed that all three new algorithms perform better or at least not significantly worse than OC1 on almost every dataset with only one exception. OC1 did worse than the best margin-based method on every dataset.

Key Words: Capacity Control, Decision Trees, Perceptron, Learning Theory, Learning Algorithm.
Running Head: Margins in Perceptron Decision Trees

1 Introduction

Perceptron Decision Trees (PDT) have been introduced by a number of authors under different names [17, 6, 7, 8, 10, 11, 27, 18]. They are decision trees in which each internal node is associated with a hyperplane in general position in the input space. They have been used in many real world pattern classification tasks, with good results [7, 18, 9]. Given their high flexibility, a feature that they share with more standard decision trees such as the ones produced by C4.5 [20], they tend to overfit the data if their complexity is not somehow kept under control. The standard approach to controlling their complexity is to limit their size, with early stopping or pruning procedures.

In this paper we introduce a novel approach to complexity control in PDTs, based on the concept of the margin (namely, the distance between the decision boundaries and the training points). We prove that this quantity can be as important as the tree-size as a capacity control parameter.

The theoretical motivations behind this approach lie in the Data-Dependent Structural Risk Minimization [25]: the scale of the cover used in VC theory to provide a bound on the generalization error depends on the margin and hence the hierarchy of classes is chosen in response to the data. Of course the two complexity control criteria can be used together, combining a pruning phase with the bias towards large margins, to obtain a better performance.

These results motivate a new class of PDT learning algorithms, aimed at producing large margin trees. We propose three such algorithms: FAT, MOC1 and MOC2, and compare their performance with that of OC1, one of the best known PDT learning systems. All three large-margin systems outperform OC1 on most of the real world data-sets we have used, indicating that overfitting in PDTs can be efficiently combatted by enlarging the margin of the decision boundaries on the training data.

It is worth noting that the same learning bias is present in Support Vector Machines [12], Adaboost [24] and Bayesian Classifiers [13].

2 Perceptron Decision Trees

The most common decision trees, in which each node checks the value of a single attribute, could be defined as *axis parallel*, because the tests associated with each node are equivalent to axis-parallel hyperplanes in the input space. Many variations of this simple model have been proposed, since the introduction of such systems in the early '80s. Some of them involve more complex tests at the decision nodes, usually testing more than one attribute.

Decision Trees whose nodes test a linear combination of the attributes have been proposed by different researchers under different names: Linear Combination Trees, multivariate DT [11], oblique DTs [18], Perceptron Decision Trees [27], etc. The first of such systems was proposed by Breiman, who incorporated it into the package CART[10]. The tests associated at each node are equivalent to hyperplanes *in general position*, and they partition the input space into polyhedra as illustrated in Figure 1. They obviously include as a special case the more common decision trees output by systems like C4.5.

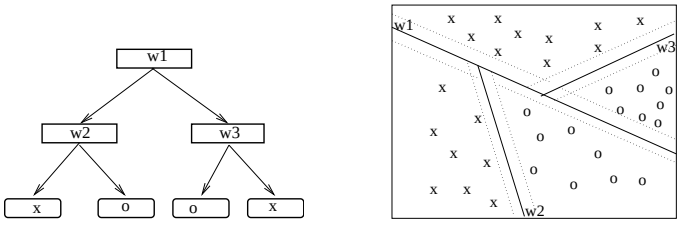

Figure 1: A Perceptron Decision Tree and the way it splits the input space

The extreme flexibility of such systems makes them particularly exposed to the risk of overfitting. This is why efficient methods for controlling their expressive power (typically pruning techniques) have always to be used in combination with the standard TopDown growth algorithms.

The class of functions computed by PDTs is formally defined as follows.

Definition 2.1 Generalized Decision Trees (GDT). *Given a space X and a set of boolean functions $\mathcal{F} = \{f : X \to \{0,1\}\}$, the class* GDT($\mathcal{F}$) *of Generalized Decision Trees over*

$\mathcal{F}$ are functions which can be implemented using a binary tree where each internal node is labeled with an element of $\mathcal{F}$, and each leaf is labeled with either 1 or 0.

To evaluate a particular tree T on input $x \in X$, All the boolean functions associated to the nodes are assigned the same argument $x \in X$, which is the argument of $T(x)$. The values assumed by them determine a unique path from the root to a leaf: at each internal node the left (respectively right) edge to a child is taken if the output of the function associated to that internal node is 0 (respectively 1). This path is known as the *evaluation path*. The value of the function $T(x)$ is the value associated to the leaf reached. We say that input x reaches a node of the tree, if that node is on the evaluation path for x.

In the following, the *nodes* are the internal nodes of the binary tree, and the *leaves* are its external ones.

Examples.

- Given $X = \{0,1\}^n$, a *Boolean Decision Tree (BDT)* is a GDT over

$$\mathcal{F}_{\text{BDT}} = \{f_i : f_i(\mathbf{x}) = \mathbf{x}_i, i = 1, \ldots, n\}$$

- Given $X = R^n$, a *C4.5-like Decision Tree (CDT)* is a GDT over

$$\mathcal{F}_{\text{CDT}} = \{f_{i,\theta} : f_{i,\theta}(\mathbf{x}) = 1 \Leftrightarrow x_i > \theta, \theta \in R, i = 1, \ldots, n\}$$

  This kind of decision tree defined on a continuous space are the output of common algorithms like C4.5 and CART, and we will refer to them as CDTs.

- Given $X = R^n$, a *Perceptron Decision Tree (PDT)* is a GDT over

$$\mathcal{F}_{\text{PDT}} = \{f_w : f_w(\mathbf{x}) = 1 \Leftrightarrow w^T\mathbf{x} > 0, w \in R^{n+1}\},$$

where we have assumed that the inputs have been augmented with a coordinate of constant value, hence implementing a thresholded perceptron.

PDTs are generally induced by means of a TopDown growth procedure, which starts from the root node and greedily chooses a perceptron which maximizes some cost function, usually a measure of the "impurity" of the subsamples implicitly defined by the split. This maximization is usually hard to perform, and sometimes replaced by randomized (sub)optimization. The subsamples are then mapped to the two children nodes. The procedure is then recursively applied to the nodes, and the tree is grown until some stopping criterion is met. Such a tree is then used as a starting point for a "BottomUp" search, performing a pruning of the tree. This implies eliminating the nodes which are redundant, or which are unable to "pay for themselves" in terms of the cost function. Generally pruning an overfitting tree produces better classifiers than those obtained with early stopping, since this makes it possible to check if promising directions were in fact worth exploring, and if locally good solutions were on the contrary a dead-end. So, while the standard TopDown algorithm is an extremely greedy procedure, with the introduction of pruning it can be possible to look-ahead: this allows for discovery of more hidden structure.

The capacity control in PDTs is hence completely achieved by controlling the size of the tree, that is the complexity of the overall classifier. We will propose an alternative method, which on the contrary focuses on reducing the complexity of the node classifiers, independently of the tree size. This will be possible thanks to a theoretical analysis of generalization performance of the function class defined by PDTs, in the framework of VC theory.

3 Theoretical Analysis of Generalization

The generalization performance of a learning machine can be studied by means of uniform convergence bounds, with a technique introduced by Vapnik and Chervonenkis [30]. The

central concept in such an analysis is the "effective capacity" of the class of hypotheses accessible by the machine: the richer such a class, the higher the risk of overfitting. This feature of a learning machine is often referred to as its flexibility or *capacity*. The issue of preventing overfitting by allowing just the right amount of flexibility is therefore known as *capacity control*.

The notion of effective cardinality of a function class is captured by its "growth function" for Boolean classes or "covering numbers" for real valued functions. The size of the covering numbers depends on the accuracy of the covering as well as the function class itself. The larger the margin the less accuracy is required in the covering.

In the following we will be concerned with estimating the capacity of the class of PDTs. We will see that the margin does affect the flexibility of such a hypothesis class, as does the tree-size. This will motivate some alternative techniques for controlling overfitting which – albeit conceptually similar to pruning – act on the complexity of the node classifiers rather than on the complexity of the overall tree.

We begin with the definition of the fat-shattering dimension, which was first introduced in [15], and has been used for several problems in learning since [1, 4, 2, 3].

Definition 3.1 *Let $\mathcal{F}$ be a set of real valued functions. We say that a set of points X is* γ-shattered by $\mathcal{F}$ relative to $r = (r_x)_{x \in X}$ *if there are real numbers r_x indexed by $x \in X$ such that for all binary vectors b indexed by X, there is a function $f_b \in \mathcal{F}$ satisfying*

$$f_b(x) \begin{cases} \geq r_x + \gamma & \text{if } b_x = 1 \\ \leq r_x - \gamma & \text{otherwise.} \end{cases}$$

The fat shattering dimension $\mathrm{fat}_{\mathcal{F}}$ *of the set $\mathcal{F}$ is a function from the positive real numbers to the integers which maps a value γ to the size of the largest γ-shattered set, if this is finite, or infinity otherwise.*

As an example which will be relevant to the subsequent analysis consider the class:

$$\mathcal{F}_{\text{lin}} = \{x \to \langle w, x\rangle + \theta : \|w\| = 1\}.$$

We quote the following result from [5].

Theorem 3.2 *[5] Let $\mathcal{F}_{\text{lin}}$ be restricted to points in a ball of n dimensions of radius R about the origin. Then*

$$\text{fat}_{\mathcal{F}_{\text{lin}}}(\gamma) \leq \min\{R^2/\gamma^2, n+1\}.$$

The following theorem bounds the generalization of a classifier in terms of the fat shattering dimension rather than the usual Vapnik-Chervonenkis or Pseudo dimension.

Let T_θ denote the threshold function at θ: T_θ: $R \to \{0,1\}$, $T_\theta(\alpha) = 1$ iff $\alpha > \theta$. For a class of functions $\mathcal{F}$, $T_\theta(\mathcal{F}) = \{T_\theta(f)\text{: } f \in \mathcal{F}\}$.

Theorem 3.3 *[25] Consider a real valued function class $\mathcal{F}$ having fat shattering function bounded above by the function* afat : $R \to N$ *which is continuous from the right. Fix $\theta \in R$. If a learner correctly classifies m independently generated examples $\mathbf{z}$ with $h = T_\theta(f) \in T_\theta(\mathcal{F})$ such that the test error is zero and $\gamma = \min|f(x_i) - \theta|$, then with confidence $1-\delta$ the expected error of h is bounded from above by*

$$\epsilon(m,k,\delta) = \frac{2}{m}\left(k\log\left(\frac{8em}{k}\right)\log(32m) + \log\left(\frac{8m}{\delta}\right)\right),$$

where $k = \text{afat}(\gamma/8)$.

The importance of this theorem is that it can be used to explain how a classifier can give better generalization than would be predicted by a classical analysis of its VC dimension. Essentially expanding the margin performs an automatic capacity control for function classes with small fat shattering dimensions. The theorem shows that when a large margin is achieved it is as if we were working in a lower VC class.

We should stress that in general the bounds obtained should be better for cases where a large margin is observed, but that a priori there is no guarantee that such a margin will occur. Therefore a priori only the classical VC bound can be used. In view of corresponding lower bounds on the generalization error in terms of the VC dimension, the a posteriori bounds depend on a favorable probability distribution making the actual learning task easier. Hence, the result will only be useful if the distribution is favorable or at least not adversarial. In this sense the result is a distribution dependent result, despite not being distribution dependent in the traditional sense that assumptions about the distribution have had to be made in its derivation. The benign behavior of the distribution is automatically estimated in the learning process.

In order to perform a similar analysis for perceptron decision trees we will consider the set of margins obtained at each of the nodes, bounding the generalization as a function of these values.

It turns out that bounding the fat shattering dimension of PDT's viewed as real function classifiers is difficult. We will therefore do a direct generalization analysis mimicking the proof of Theorem 3.3 but taking into account the margins at each of the decision nodes in the tree.

Definition 3.4 *Let (X, d) be a (pseudo-) metric space, let A be a subset of X and $\epsilon > 0$. A set $B \subseteq X$ is an* ϵ-cover *for A if, for every $a \in A$, there exists $b \in B$ such that $d(a, b) < \epsilon$. The* ϵ-covering number *of A, $\mathcal{N}_d(\epsilon, A)$, is the minimal cardinality of an ϵ-cover for A (if there is no such finite cover then it is defined to be ∞).*

We write $\mathcal{N}(\epsilon, \mathcal{F}, \mathbf{x})$ for the ϵ-covering number of $\mathcal{F}$ with respect to the ℓ_∞ pseudo-metric measuring the maximum discrepancy on the sample $\mathbf{x}$. These numbers are bounded in the following Lemma.

Lemma 3.5 (Alon *et al.* [1]) *Let $\mathcal{F}$ be a class of functions $X \to [0, 1]$ and P a distribution*

over X. Choose $0 < \epsilon < 1$ and let $d = \mathrm{fat}_{\mathcal{F}}(\epsilon/4) \leq em$. Then

$$E\left(\mathcal{N}(\epsilon, \mathcal{F}, \mathbf{x})\right) \leq 2\left(\frac{4m}{\epsilon^2}\right)^{d\log(2em/(d\epsilon))},$$

where the expectation E is taken w.r.t. a sample $\mathbf{x} \in X^m$ drawn according to P^m.

Corollary 3.6 *[25] Let $\mathcal{F}$ be a class of functions $X \to [a, b]$ and P a distribution over X. Choose $0 < \epsilon < 1$ and let $d = \mathrm{fat}_{\mathcal{F}}(\epsilon/4) \leq em$. Then*

$$E\left(\mathcal{N}(\epsilon, \mathcal{F}, \mathbf{x})\right) \leq 2\left(\frac{4m(b-a)^2}{\epsilon^2}\right)^{d\log(2em(b-a)/(d\epsilon))},$$

where the expectation E is over samples $\mathbf{x} \in X^m$ drawn according to P^m.

We are now in a position to tackle the main lemma which bounds the probability over a double sample that the first half has zero error and the second error greater than an appropriate ϵ. Here, error is interpreted as being differently classified at the output of the tree. In order to simplify the notation in the following lemma we assume that the decision tree has K nodes and we denote $\mathrm{fat}_{\mathcal{F}_{\mathrm{lin}}}(\gamma)$ by $\mathrm{fat}(\gamma)$.

Lemma 3.7 *Let T be a perceptron decision tree with K decision nodes with margins $\gamma^1, \gamma^2, \ldots, \gamma^K$ at the decision nodes satisfying $k_i = \mathrm{fat}(\gamma_i/8)$. If it has correctly classified m labeled examples generated independently according to the unknown (but fixed) distribution P with support in a ball of radius R, then we can bound the following probability to be less than δ,*

$$P^{2m}\left\{\mathbf{xy}\colon \exists \text{ a tree } T : T \text{ correctly classifies } \mathbf{x}, \right.$$
$$\left. \text{fraction of } \mathbf{y} \text{ misclassified } > \epsilon(m, K, \delta)\right\} < \delta,$$

where $\epsilon(m, K, \delta) = \frac{1}{m}(D\log(8m) + \log\frac{2^K}{\delta})$. where $D = \sum_{i=1}^{K} k_i \log(4em/k_i)$.

Proof: Using the standard permutation argument (as in [30]), we may fix a sequence $\mathbf{xy}$ and bound the probability under the uniform distribution on swapping permutations that the sequence satisfies the condition stated. We consider generating minimal $\gamma_k/2$-covers $B^k_{\mathbf{xy}}$ for each value of k, where $\gamma_k = \min\{\gamma' : \text{fat}(\gamma'/8) \leq k\}$. Suppose that for node i of the tree the margin γ^i of the hyperplane w_i satisfies $\text{fat}(\gamma^i/8) = k_i$. We can therefore find $f_i \in B^{k_i}_{\mathbf{xy}}$ whose output values are within $\gamma^i/2$ of w_i. We now consider the tree T' obtained by replacing the node perceptrons w_i of T with the corresponding f_i. This tree performs the same classification function on the first half of the sample, and the margin at node i remains larger than $\gamma^i - \gamma_{k_i}/2 > \gamma_{k_i}/2$. If a point in the second half of the sample is incorrectly classified by T it will either still be incorrectly classified by the adapted tree T' or will at one of the decision nodes i in T' be closer to the decision boundary than $\gamma_{k_i}/2$. The point is thus distinguishable from left hand side points which are both correctly classified and have margin greater than $\gamma_{k_i}/2$ at node i. Hence, that point must be kept on the right hand side in order for the condition to be satisfied. Hence, the fraction of permutations that can be allowed for one choice of the functions from the covers is $2^{-\epsilon m}$. We must take the union bound over all choices of the functions from the covers. Using the techniques of [25] the numbers of these choices is bounded by Corollary 3.6 as follows

$$\Pi_{i=1}^{K} 2(8m)^{k_i \log(4em/k_i)} = 2^K (8m)^D,$$

where $D = \sum_{i=1}^{K} k_i \log(4em/k_i)$. The value of ϵ in the lemma statement therefore ensures that the union bound is less than δ. $\square$

Lemma 3.7 applies to a particular tree with a specified number of nodes, architecture and fat shattering dimensions for each node. In practice we will observe these quantities after running the learning algorithm which generates the tree. Hence, to obtain a bound that can be applied in practice we must bound the probabilities uniformly over all of the possible architectures and dimensions that can arise. Before giving the theorem that will give this

bound we require two results. The first is due to Vapnik [28, page 168] and is the key to bounding error probabilities in terms of the probabilities of discrepancies on a double sample.

Lemma 3.8 *Let X be a set and S a system of sets on X, and P a probability measure on X. For $\mathbf{x} \in X^m$ and $A \in S$, define $\nu_{\mathbf{x}}(A) := |\mathbf{x} \cap A|/m$. If $m > 2/\epsilon$, then*

$$P^m \left\{ \mathbf{x}: \sup_{A \in S} |\nu_{\mathbf{x}}(A) - P(A)| > \epsilon \right\} \leq 2P^{2m} \left\{ \mathbf{xy}: \sup_{A \in S} |\nu_{\mathbf{x}}(A) - \nu_{\mathbf{y}}(A)| > \epsilon/2 \right\}.$$

The second result gives a bound on the number of different tree architectures that have a given number of computational nodes.

Theorem 3.9 *[21] The number S_k of k node Decision Tree skeletons is*

$$S_k = \frac{1}{k+1} \binom{2k}{k}.$$

Combining these two results with Lemma 3.7 we obtain the following theorem.

Theorem 3.10 *Suppose we are able to classify an m sample of labeled examples using a perceptron decision tree and suppose that the tree obtained contained K decision nodes with margins γ_i at node i, then we can bound the generalization error with probability greater than $1 - \delta$ to be less than*

$$\frac{130R^2}{m} \left(D' \log(4em) \log(4m) + \log \frac{(4m)^{K+1} \binom{2K}{K}}{(K+1)\delta} \right)$$

where $D' = \sum_{i=1}^{K} \frac{1}{\gamma_i^2}$.

Proof: We must bound the probabilities over different architectures of trees and different margins. We first use Lemma 3.8 to bound the probability of error in terms of the probability of the discrepancy between the performance on two halves of a double sample. In order to apply Lemma 3.7 we must consider all possible architectures that can occur and for each

architecture the different patterns of k_i's over the decision nodes. For a fixed value of K Theorem 3.9 gives the number of decision tree skeletons. The largest allowed value of k_i is m and so for fixed K we can bound the number of possibilities by

$$\frac{1}{K+1}\binom{2K}{K}2^{K+1}m^K,$$

where 2^{K+1} counts the possible labeling of the $K+1$ leaf nodes. Hence, there are this number of applications of Lemma 3.7 for a fixed K. Since the largest value that K can take is m, we can let $\delta_K = \delta/m$, so that the sum $\sum_{K=1}^{m}\delta_k = \delta$. Choosing

$$\epsilon\left(m, K, \frac{(K+1)\delta_K}{\binom{2K}{K}2^{K+2}m^K}\right) = \frac{65R^2}{m}\left(D'\log(4em)\log(4m) + \log\frac{4(4m)^K\binom{2K}{K}}{(K+1)\delta_K}\right)$$

in the applications of Lemma 3.7, ensures that the probability of any of the statements failing to hold is less than $\delta/2$. Note that we have replaced the constant $8^2 = 64$ by 65 in order to ensure the continuity from the right required for the application of Theorem 3.3 and have upperbounded $\log(4em/k_i)$ by $\log(4em)$. Hence, applying Lemma 3.8 in each case the probability that the statement of the theorem fails to hold is less than δ. □

4 Experimental Results

From the theory presented in the previous section, it follows that large-margin PDTs are more likely to generalize well. A bias toward large-margin trees can be implemented in a number of different ways, either as a post-processing phase of existing trees or as a brand new impurity measure to determine splitting/stopping criteria in TopDown growth algorithms.

To facilitate comparisons, we have implemented three such algorithms as modifications of one of the best-known PDT learning systems OC1 [18] of Murthy, Kasif and Salzberg, which is freely available over the Internet. The effect of the large-margin bias can hence be directly

assessed, by running the margin-arbitrary version of the same algorithm on the same data. The first such algorithm, FAT, accepts in input a PDT constructed using OC1 and outputs a large margin version of the same tree. The other two, MOC1 and MOC2, have different impurity measures which take into consideration the margins. All three algorithms work for multi-class data.

The three systems have been compared with OC1 on 10 benchmarking data sets. The results confirm the predictions of the theoretical model, clearly indicating that the generalization is improved by enlarging the margin.

The data sets we have used for the study are 6 data sets used in the original OC1 paper[18], and 4 other data sets, which are publicly available in the UCI data repository [31]. The data sets studied in [18] are *Dim*, *Bright*, *Wisconsin Breast Cancer*, *Pima Indians Diabetes*, *Boston Housing* and *Iris*. The four additional data sets are *Bupa*, *Sonar*, *Heart* and *Wisconsin Breast Cancer Prognosis*. The data sets differ greatly from subjects, sizes and number of attributes, the subjects of data sets range from medical to astronomical, sizes from 150 to 4192, number of attributes from 4 to 60. For details of these data sets see [18, 31]. For each data set, a single run of 10-fold cross-validation is carried out. The relevant quantity, in this experiment, is the difference in the test accuracy between PDTs with arbitrary margins constructed by OC1 and the PDTs with large margins on the same data.

Comparing learning algorithms has drawn extensive attention recently [16, 14, 23, 19]. A single run of 10-fold cross-validation on a reasonable number of data sets is still a preferred practical approach. It is prone to detect the difference of two algorithms. We basically followed the approach recommended in [23].

In the rest of this section, first we will briefly review the OC1 system, then present our three large margin algorithms, and compare their performances with OC1.

4.1 Review of OC1

OC1 [18] is a randomized algorithm, which performs a randomized hill-climbing search for learning the perceptrons, and builds the tree TopDown. Starting from the root node, the system chooses the hyperplane which minimizes a predefined "impurity" measure (e.g. information gain [20], or Gini index [10], or the Twoing Rule [10, 18], etc.). The system is greedy because at each stage it chooses the best split available, and randomized because such a best split is not obtained by means of exhaustive search but with a randomized hill-climbing process.

Throughout this study we use the twoing rule as the impurity measure, for OC1, FAT, and MOC1. MOC2 uses a modified twoing rule as impurity measure. Other impurity measures can also be applied in FAT and MOC1 without change, while MOC2 would need minor changes.

The Twoing Rule

$$TwoingValue = \frac{|T_L|}{n} * \frac{|T_R|}{n} * \left(\sum_{i=1}^{k} \left| \frac{|Li|}{|T_L|} - \frac{|R_i|}{|T_R|} \right| \right)^2 \quad (1)$$

where

$n = |T_L| + |T_R|$ - total number of instances at current node

k - number of classes, for two class problems

$|T_L|$ - number of instances on the left of the split, i.e. $w^T x + b >= 0$

$|T_R|$ - number of instances on the right of the split i.e. $w^T x + b < 0$

$|L_i|$ - number of instances in category i on the the left of the split

$|R_i|$ - number of instances in category i on the the right of the split

This is a goodness measure rather than an impurity one, and OC1 attempts to maximize it

at each split during the tree growth via minimizing $1/TwoingValue$. Further details about the randomization, the pruning, and the splitting criteria can be found in [18].

4.2 Results of FAT

Description of algorithm FAT

The algorithm FAT uses the tree produced by OC1 as a starting point, and maximizes its margins. This involves finding - for each node - the hyperplane which performs *the same* split as performed by the OC1 tree but with the maximal margin. This can be done by considering the subsample reaching each node as perfectly divided into two parts, and feeding the data accordingly relabeled to an algorithm which finds the optimal separating hyperplane - separating hyperplane with maximal margin in this now linearly separable data. The optimal separating hyperplanes are then placed in the corresponding decision nodes and the new tree is tested on the same test data. Note that, the PDT produced by FAT will have the same tree structure and training accuracy as the original PDT constructed by OC1. They will only differ on test accuracy. We use the Support Vector Machine (SVM) algorithm [29] to find the optimal separating hyperplane. To conform with the definition of a PDT, no kernel is used in the SVM, the optimal separating hyperplane is constructed in the input space.

Algorithm for FAT

1. Construct a decision tree using OC1, call it OC1-PDT.

2. Starting from root of OC1-PDT, traverses through all the non-leaf nodes. At each node,

   - Relabel the points at this node with $\omega^T x + b \geq 0$ as class *right*, the other points at this node as class *left*.

- Find the perceptron (optimal separating hyperplane) $f(x) = \omega^{*T} x + b^*$, which separates class *right* and class *left* perfectly with maximal margin.
- replace the original perceptron with the new one.

3. Output the FAT-PDT.

Optimal Separating Hyperplane - SVM algorithm for the linearly separable case

The following problems are solved at each node, to find the optimal separating hyperplane for linearly separable data [29].

$$\begin{aligned} \min_{w,b} \quad & \tfrac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i[(w^T x_i) - b] \geq 1, \quad y_i \in \{1, -1\},\ i = 1, \ldots, \ell \end{aligned} \tag{2}$$

where $y_i = 1$ corresponds to class *right* and $y_i = -1$ corresponds to class left and ℓ is the number of points reaching the decision node.

For computational reason we usually solve the dual problem of (2):

$$\begin{aligned} \min_{\alpha} W(\alpha) = \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i, y_j (x_i \cdot x_j) - \sum_{i=1}^{\ell} \alpha_i \\ \text{subject to} \quad \textstyle\sum_{i=1}^{\ell} \alpha_i y_i = 0 \\ \alpha_i \geq 0, \qquad i = 1, \ldots, \ell \end{aligned} \tag{3}$$

FAT-PDT has a generalization error bounded by theorem 3.10. We observed that FAT completely relied on and was restricted by the perceptron decision tree induced by OC1. In many cases, the margins in the splits found by OC1 are very small, so FAT has little scope for optimization. In general, if there is a margin in the top split at the root node, FAT will generalize much better. It implies that the greedy algorithm OC1 is not a good tree inducer for FAT, in the sense of the margin. We need to find a better non-greedy tree inducer for

FAT. On the other hand, FAT provides a new approach to applying the Support Vector Machine for multi-class classification tasks.

Comparison of FAT and OC1

For each dataset, 10-fold cross-validation is used to measure the learning ability of the algorithm FAT and OC1. A paired t-test is used to test the difference of the means of FAT and OC1.

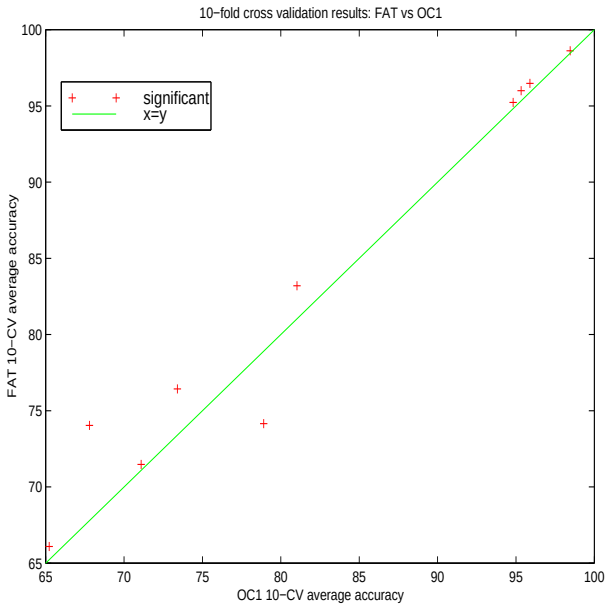

Figure 2: Comparison the 10-fold CV results of FAT versus OC1. If the point is above the line, it indicates the 10-fold CV mean of FAT is higher than that of OC1, and vice versa. The figure shows that FAT outperforms OC1 on 9 out of 10 data sets and is outperformed only on 1 data set.

From Figure 2, we can see that, FAT outperforms OC1 on 9 out of the 10 data sets, and outperforms OC1 on all the 6 data sets studied in [18]. The 10-fold cross-validation mean differences of FAT and OC1 on those 9 data sets are all significant when a paired t-test is applied. On one data set *Prognosis*, OC1 outperforms FAT and the difference is significant. We also observed that, except in one case (*Prognosis*), FAT performs as good as or better than OC1 in *every* fold of 10-fold cross-validation. So when FAT has a higher mean than

OC1, it is significant at a small α level for the paired t-test even though the difference is small. α is not so small when a two sample t-test is applied. This is a strong indication that Perceptron Decision Trees with large margins generalize better. The 10-fold cross-validation means and p values are summarized in Table 2.

4.3 Results of MOC1

Description of MOC1

MOC1 (Margin_OC1) is a variation of OC1, which modifies the objective function of OC1 to consider the size of the margin. The underlying philosophy is to find a separating plane with a tradeoff between training accuracy and the size of margin at each node. This idea is motivated by the Support Vector Machine for the linearly non-separable case, which minimizes the classification error and maximizes the margin at the same time. SVM with soft margin minimizes the sum of misclassification errors and a constant C multiplying the reciprocal of the soft margin. SVM tries to find a split with high classification accuracy and large soft margin. Analagously, SVM minimizes the sum of the impurity measure and a constant times the reciprocal of the hard margin.

The MOC1 algorithm minimizes the following objective function:

$$(1-\lambda) * OC1\_Objective + C * \frac{1}{current\_margin} \tag{4}$$

where

- $OC1\_Objective$ is the impurity measure of OC1, in this study, the default twoing rule is used as impurity measure.

- $current\_margin$ is the distance between two nearest points on the different side of the current separating hyperplane.

- λ is a scalar weight, $\lambda \in [0, 1]$

- $C = \lambda * log10(\text{no\_of\_points\_at\_current\_node})$

λ determines how much the large margin is weighted in selecting the split. Tuning λ could improve the performance. When determining the weight of the margin, we also take the number of points at the current node into consideration. The idea is that a constant weight of margin for all nodes is not good. The weight should be able to adapt to the position of current node and size of training examples at the current node. Since we are not particularly interested in finding the tree with highest possible accuracy, but rather demonstrating that large margins can improve the generalization, we did not tune the λ for each data set to achieve the highest possible accuracy. We set $\lambda = .05$ in all data sets. In other words, the results of MOC1 presented below are not the best results possible.

Comparison of MOC1 and OC1

As in the previous section, we use 10-fold cross-validation to measure the learning ability of the algorithm MOC1 and OC1. To test the difference between the means of MOC1 and OC1, here again a paired t-test is used.

From figure 3, we can see that MOC1 has higher 10-fold cross-validation mean than that of OC1 on 8 of the 10 data sets, and 5 of them are significantly higher; OC1 has higher means on the other two data sets (*Cancer, Prognosis*), the differences are tiny and both are not significant. Overall, MOC1 outperforms OC1 on 6 of the 10 data sets and as good as OC1 on the other four. Of the six data sets studied in [18], MOC1 outperforms OC1 on five of them and performs as well as OC1 on the final one (*Cancer*). See table 2 for respective means and p values.

4.4 Results of MOC2

Description of MOC2

MOC2 uses a modified twoing rule, which directly incorporates the idea of large margin to the

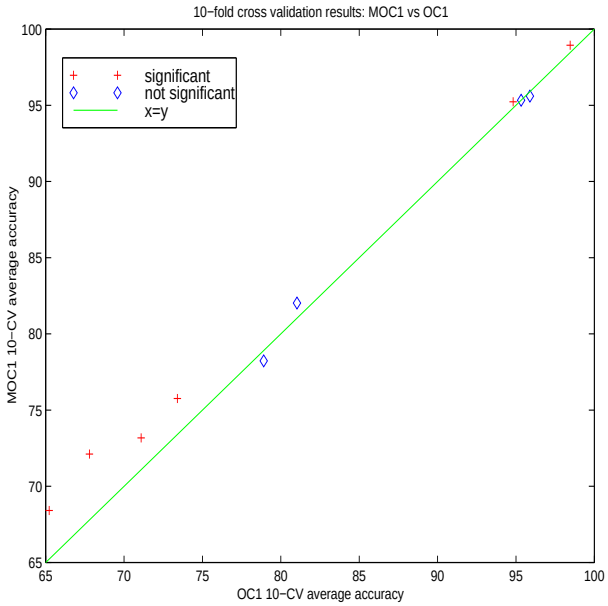

Figure 3: Comparison the 10-fold CV results of MOC1 versus OC1. If the point is above the line, it indicates the 10-fold CV average of FAT is higher than that of OC1, and vice versa. The figure shows that MOC1 outperforms OC1 on 6 out of 10 data sets, and performs as good as OC1 on the rest four data sets.

impurity measure. Unlike MOC1, MOC2 uses a soft margin. It treats points falling within the margin and outside of the margin differently. Only the impurity measure is altered. The rest is same as in the standard OC1 algorithm.

The modified twoing rule

$$TwoingValue = \frac{|MT_L|}{n} * \frac{|MT_R|}{n} * \sum_{i=1}^{k} \left| \frac{|L_i|}{|T_L|} - \frac{|R_i|}{|T_R|} \right| * \sum_{i=1}^{k} \left| \frac{|ML_i|}{|MT_L|} - \frac{|MR_i|}{|MT_R|} \right|$$

where

$n = |T_L| + |T_R|$ - total number of instances at current node

k - number of classes, for two class problems

$|T_L|$ - number of instances on the left of the split, i.e. $w^T x + b >= 0$

$|T_R|$ - number of instances on the right of the split i.e. $w^T x + b < 0$

$|L_i|$ - number of instances in category i on the the left of the split

$|R_i|$ - number of instances in category i on the the right of the split

$|MT_L|$ - number of instances on the left of the split, $w^T x + b >= 1$

$|MT_R|$ - number of instances on the right of the split $w^T x + b <= -1$

$|ML_i|$ - number of instances in category i with $w^T x + b >= 1$

$|MR_i|$ - number of instances in category i with $w^T x + b <= -1$

In the modified twoing rule, our goal is, at each node, to find a split with fewer points falling within the margin (in between $-1 < w^T x + b < 1$), with high accuracy outside the margin and good overall accuracy. Here again, we try to achieve a balance of classification accuracy and size of margin. By doing this, we want to push apart the two classes from the the separating hyperplane as far as possible while maintaining a reasonable good classification accuracy, hence, improve the generalization of the induced decision tree. The advantage of MOC2 is that there are no free parameters to tune.

Comparison of MOC2 and OC1

As in previous section, 10-fold cross-validation is used to measure the learning ability of the algorithms MOC2 and OC1. Paired t-tests are used to test the difference of the means of MOC2 and OC1.

From Figure 4 we can see that MOC2 has higher mean on 9 out of the 10 data sets, and has slightly lower mean on only one data set (*Housing*). Of the 9 higher means, 5 are significantly higher. The one lower mean is not significant. Overall, MOC2 outperforms OC1 on 5 out of the 10 data sets and performs as well as OC1 on the other 5. Of the six data sets studied in [18], MOC2 outperformed OC1 on three of them, and perform as well as OC1 on the other three. The respective means and p values are summarized in Table 2

The modified twoing rule opens a new way of measuring the goodness of a split, which directly incorporates the generalization factor into the measure. In our experiments, it has

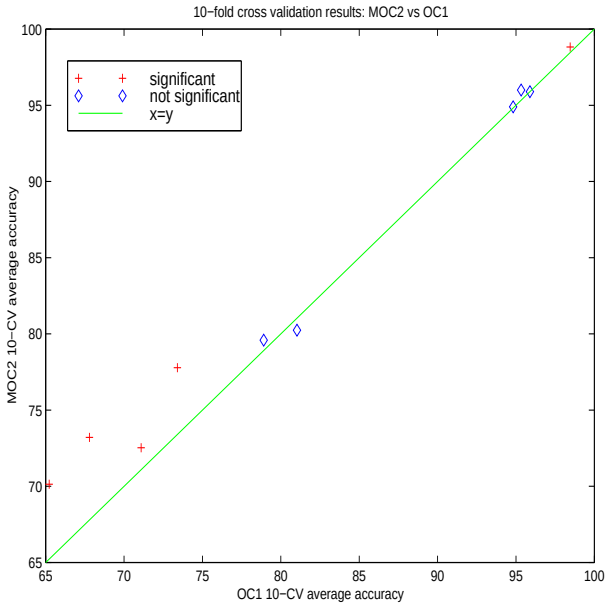

Figure 4: Comparison the 10-fold CV results of MOC2 versus OC1. If the point is above the line, it indicates the 10-fold CV mean of FAT is higher than that of OC1 on that data set, and vice versa. The figure shows that MOC2 outperforms OC1 on 5 out of 10 data sets, and performs as well as OC1 on the rest 5 data sets.

been proven to be a useful measure.

4.5 Tree Sizes

For FAT, the tree sizes are exactly the same as OC1, since FAT PDT has the same tree structure as OC1 PDT. FAT only replaces splits at nodes of the OC1 PDT with large-margin perceptrons which perform exactly the same splits. Of the ten data sets, MOC1 induced five smaller trees, one the same size tree, and four larger trees when compared with OC1. MOC2 induced five smaller trees and five bigger trees compared with OC1. We did not find consistent pattern of tree sizes. Table 1 list the tree sizes of OC1, FAT, MOC1 and MOC2.

Dataset	OC1 & FAT		MOC1		MOC2	
	Leaves	Depth	Leaves	Depth	Leaves	Depth
Bright	5.40	2.80	6.20	3.20	5.70	2.90
Bupa	5.00	2.80	2.10	1.10	7.40	3.60
Cancer	2.50	1.30	4.00	2.50	2.90	1.50
Dim	23.9	5.90	17.40	6.50	22.40	6.40
Heart	6.10	2.10	3.30	2.00	2.10	1.10
Housing	10.00	4.20	7.10	3.80	6.40	3.00
Iris	3.20	2.10	3.20	2.10	3.00	2.00
Pima	8.30	4.20	18.50	5.70	11.40	5.00
Prognosis	3.60	2.00	2.30	1.20	2.20	1.10
Sonar	4.30	2.60	6.10	3.30	5.90	2.90

Table 1: 10-fold CV average tree size of OC1, FAT, MOC1 and MOC2

Dataset	OC1 $\bar{x}$	FAT $\bar{x}$ (*p value*)	MOC1 $\bar{x}$ (*p value*)	MOC2 $\bar{x}$ (*p value*)	Best classifier
Bright	98.46	98.62 (*.05*)	**98.94** (*.10*)	98.82 (*.10*)	MOC1
Bupa	65.22	66.09 (*.10*)	68.41 (*.20*)	**70.14** (*.04*)	MOC2
Cancer	95.89	**96.48** (*.05*)	95.60	95.89	FAT
Dim	94.82	94.92 (*.20*)	95.23 (*.09*)	94.90	MOC1
Heart	73.40	76.43 (*.12*)	75.76 (*.21*)	**77.78** (*.10*)	MOC2
Housing	81.03	**83.20** (*.05*)	82.02	80.23	FAT
Iris	95.33	96.00 (*.17*)	95.33	96.00	FAT
Pima	71.09	71.48 (*.04*)	**73.18** (*.08*)	72.53 (*.23*)	MOC1
Prognosis	78.91	74.15	78.23	**79.59**	MOC2
Sonar	67.79	**74.04** (*.01*)	72.12 (*.19*)	73.21 (*.16*)	FAT

Table 2: 10-fold CV means of OC1, FAT, MOC1 and MOC2

4.6 Summary of experimental results

The theory states that maximizing the margins between the data points on each side of separating hyperplane in the perceptron decision tree will improve the error bounds, the perceptron decision tree will be more likely to generalize better. But the theory does not guarantee a specific classifier has a low error rate.

From the 10-fold cross-validation results of the 10 data sets, FAT has 9 higher means than OC1 and they are all significantly higher; MOC1 has 7 higher means, and 6 of them are significantly higher; MOC2 has 8 higher means and 5 of them are significantly higher. Equal or lower means only happened on 3 data sets, *Cancer*, MOC1 has a slightly smaller mean than OC1 on it, MOC2 has the same mean as OC1 on it; *Housing*, MOC2 has slightly smaller mean than OC1 on it; *Prognosis*, FAT has a significantly smaller mean on it, MOC1 also has a slightly smaller mean but the difference is not significant. Of the classifiers with highest mean, FAT produced four, MOC1 and MOC2 each produced three, and OC1 produced none.

From the experiments, we *believe* that PDTs with large margin are more likely to have smaller variance too. In our experiments, in most of the cases, FAT, MOC1 and MOC2 produce classifiers with smaller variances, and many of them are significantly smaller, though very occasionally they produce classifiers with significantly larger variance. However, we cannot draw any confident conclusion about the variances. We therefore did not present our study on variances here.

In short, the experimental results show that finding the separating hyperplane with large margin at each node of a perceptron decision tree can improve the error bound, hence the PDT is more likely to have a higher mean, i.e. generalizes better. Furthermore, we believe, by improving error bounds, the learning algorithm will perform more consistently, and be more likely to have smaller variance as well.

5 Conclusions

The experimental results presented in this paper clearly show that enlarging the margin does improve the generalization, and that this bias can be inserted into the growth algorithm itself, providing trees which are specifically built to minimize the theoretical bound on generalization error. Such trees do not loose any of their other desirable features, such as readability, ease of maintenance and updating, flexibility and speed.

Furthermore, the theoretical analysis of the algorithms shows that the dimension of the input space does not affect the generalization performance: it is hence possible to conceive of Perceptron Decision Trees in a high-dimensional feature space, which take advantage of kernels and margin-maximization such as Support Vector Machines. This would provide as a side effect a very natural approach to multi-class classification with Support Vector Machines.

Other theoretical results exist indicating that the tree size is not necessarily a good measure of capacity. Our analysis also shows how to take advantage of this theoretical observation, and design learning algorithms which control hypothesis complexity by acting on the complexity of the node-classifiers and hence that of the whole tree. All three of the proposed approaches, the post-processing method FAT, and the two with margin based splitting criteria MOC1 and MOC2 led to significant improvement over the baseline OC1 method. It is an open question which method is best, but maximizing margins should be a consideration of every PDT algorithm.

References

[1] Alon, N., Ben-David, S., Cesa-Bianchi, N., & Haussler, D. (1993). Scale-sensitive Dimensions, Uniform Convergence, and Learnability, in *Proceedings of the Conference on Foundations of Computer Science (FOCS)*. Also to appear in *Journal of the ACM*.

[2] Anthony, M., & Bartlett, P. (1994). Function learning from interpolation, Technical Report. (An extended abstract appeared in *Computational Learning Theory, Proceedings 2nd European Conference, EuroCOLT'95*, pages 211–221, ed. Paul Vitanyi, (Lecture Notes in Artificial Intelligence, 904) Springer-Verlag, Berlin, 1995).

[3] Peter L. Bartlett & Long P.M (1995), Prediction, Learning, Uniform Convergence, and Scale-Sensitive Dimensions, Preprint, Department of Systems Engineering, Australian National University

[4] Bartlett, P., Long, P., & Williamson, R. (1996). Fat-shattering and the learnability of Real-valued Functions, *Journal of Computer and System Sciences,* **52**(3), 434-452.

[5] Bartlett, P. & Shawe-Taylor, J. (1998). Generalization performance of Support Vector Machines and other pattern classifiers. *In Advances in Kernel Methods - Support Vector Learning*, Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola (eds.), MIT Press, Cambridge, USA

[6] Bennett K., & Mangasarian O. (1992). Robust Linear Programming discrimination of two linearly inseparable sets, Optimization Methods and Software, 1, 23-34.

[7] Bennett K., & Mangasarian O. (1994). Multicategory discrimination via linear programming, Optimization Methods and Software, 3, 29-39.

[8] Bennett K., & Mangasarian O. (1994). Serial and parallel multicategory discrimination, SIAM Journal on Optimization, 4 (4).

[9] Bennett, K., Wu, D., & Auslender, L. (1998). On Support Vector Decision Trees for Database Marketing, R.P.I Math Report No. 98-100, Rensselaer Polytechnic Institute, Troy, NY.

[10] Breiman L., Friedman J.H., Olshen R.A., & Stone C.J. (1984). *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA.

[11] Broadley C.E., Utgoff P.E. (1995). Multivariate Decision Trees, Machine Learning, vol. 19, pp. 45-77.

[12] Cortes, C. & Vapnik, V. (1995). Support-Vector Networks, Machine Learning, vol. 20, pp. 273–297, 1995.

[13] Cristianini, N., Shawe-Taylor, J. & Sykacek, P. (1998). Bayesian Classifiers are Large Margin Hyperplanes in a Hilbert Space, in Shavlik, J., ed., *Machine Learning: Proceedings of the Fifteenth International Conference*, Morgan Kaufmann Publishers, San Francisco, CA.

[14] Dietterich, T.G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation, 10(7).

[15] Kearns M. & Schapire, R. (1990). Efficient Distribution-free Learning of Probabilistic Concepts, pages 382–391 in *Proceedings of the 31st Symposium on the Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA.

[16] Kohavi, R.(1995). A study of Cross-Validation and Bootatraping for Accuracy Estimation and Model Selection, *International Joint Conference on Artifical Intelligence*.

[17] Mangasarian O., Setiono R., & Wolberg W. (1990) Pattern Recognition via Linear Programming: Theory and application to medical diagnosis, *Proceedings on Workshop on Large-Scale Numerical Optimization*, T. F. Coleman and Y. Li, editor, SIAM, Philadelphia, PA.

[18] Murthy S.K., Kasif S., & Salzberg S. (1994). A System for Induction of Oblique Decision Trees, Journal of Artificial Intelligence Research, 2 , pp. 1-32.

[19] Neal,R.N. (1998). Assessing Relevance Determination Methods Using DELVE Generalization in *Neural Networks and Machine Learning*, C. M. Bishop (editor), Springer-Verlag.

[20] Quinlan J.R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

[21] Quinlan,J.R, & Rivest, R. (1989). Learning Decision Trees Using the Minimum Description Lenght Principle, Information and Computation 80, 227-248.

[22] Sankar, A., & Mammone, R.J. (1993) Growing and Pruning Neural Tree Networks, IEEE Transactions on Computers, 42:291-299.

[23] Salzberg, S. (1997). On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach. Data Mining and Knowledge Discovery 1:3, 317-327.

[24] Schapire, R., Freund, Y., Bartlett, P. L., & Sun Lee, W. (1997). Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. In D.H. Fisher, Jr., editor, *Proceedings of International Conference on Machine Learning, ICML'97*, pages 322–330, Nashville, Tennessee. Morgan Kaufmann Publishers.

[25] Shawe-Taylor, J., Bartlett, P. L., Williamson, R.C., & Anthony M. (1996). Structural Risk Minimization over Data-Dependent Hierarchies, NeuroCOLT Technical Report NC-TR-96-053.
(`ftp://ftp.dcs.rhbnc.ac.uk/pub/neurocolt/tech_reports`), (1998) IEEE Transactions on Information Theory, 44 (5).

[26] Sirat, J.A. and Nadal, J.-P. (1990) "Neural trees: a new tool for classification", Network, 1, 423–438.

[27] Utgoff P.E. (1989). Perceptron Trees: a Case Study in Hybrid Concept Representations, Connection Science, 1, 377–391.

[28] Vapnik, V.(1982). *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, New York, 1982.

[29] Vapnik, V.(1995). *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.

[30] Vapnik, V. and Chervonenkis, A., (1971). On the Uniform Convergence of Relative Frequencies of Events to their Probabilities, Theory of Probability and Applications, 16, 264–280.

[31] University of California, Irvine - Machine Learning Repository, http://www.ics.uci.edu/ mlearn/MLRepository.html