This is a repository copy of *The use of vicinal-risk minimization for training decision trees*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/90870/

Version: Accepted Version

Article:

Cao, Y. and Rockett, P.I. (2015) The use of vicinal-risk minimization for training decision trees. Applied Soft Computing, 31. 185 - 195. ISSN 1568-4946

https://doi.org/10.1016/j.asoc.2015.02.043

Reuse

Unless indicated otherwise, fulltext items are protected by copyright with all rights reserved. The copyright exception in section 29 of the Copyright, Designs and Patents Act 1988 allows the making of a single copy solely for the purpose of non-commercial research or private study within the limits of fair dealing. The publisher or other rights-holder may allow further reproduction and re-use of this version - refer to the White Rose Research Online record for this item. Where records identify the publisher as the copyright holder, users can verify any specific terms of use on the publisher's website.

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

The Use of Vicinal-Risk Minimization for Training Decision Trees

Yilong Cao[a], Peter I. Rockett[a]

[a]*Department of Electronic and Electrical Engineering, University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK*

Abstract

We propose the use of Vapnik's vicinal risk minimization (VRM) for training decision trees to approximately maximize decision margins. We implement VRM by propagating uncertainties in the input attributes into the labeling decisions. In this way, we perform a global regularization over the decision tree structure. During a training phase, a decision tree is constructed to minimize the total probability of misclassifying the labeled training examples, a process which approximately maximizes the margins of the resulting classifier. We perform the necessary minimization using an appropriate meta-heuristic (genetic programming) and present results over a range of synthetic and benchmark real datasets. We demonstrate the statistical superiority of VRM training over conventional empirical risk minimization (ERM) and the well-known C4.5 algorithm, for a range of synthetic and real datasets. We also conclude that there is no statistical difference between trees trained by ERM and using C4.5. Training with VRM is shown to be more stable and repeatable than by ERM.

Keywords:
Decision trees, Vicinal-risk minimization, Decision trees, Classification

1. Introduction

Decision trees are a popular and widely-used classification paradigm, largely due the ease with which the trained classifiers can be interpreted.

Email addresses: naodai.mmx@gmail.com (Yilong Cao), p.rockett@sheffield.ac.uk (Peter I. Rockett)

Unfortunately, it has been shown that constructing an optimal decision tree (DT) is an NP-complete problem [1] and so a number of greedy heuristics have been proposed over the years, probably the foremost being the C4.5 algorithm [2], which seek sequentially to maximize information gain at each node in the tree. Typically with C4.5, a DT is trained to the point of over-fitting and then pruned with a second heuristic to improve generalization.

As an alternative training method for DTs, a wide range of meta-heuristics have been explored; see [3] for a recent survey. In this work we have used genetic programming (GP) since, as a population-based stochastic search method, GP has been shown to be well-suited to finding approximate solutions to NP-hard problems, such as DT training. Koza [4] seems to have been the first to propose GP for this purpose although see [5] for a comprehensive review.

Previous work on the evolutionary training of DTs has clearly established that credible decision trees can be induced by minimizing *empirical risk* [6] (i.e., misclassification error, or the fraction of patterns incorrectly classified) over some training set. Notwithstanding, there is a dearth of work which *quantitatively* compares GP results with conventional tree induction methods, such as C4.5. A number of authors—for example, [7]—have noted that GP-induced trees can give smaller misclassification errors compared to C4.5 but smaller errors do not necessarily denote any statistical significance. Since conventional tree induction methods are greedy algorithms, one would expect sub-optimal performance. The theoretical advantage of meta-heuristic methods is that they have been demonstrated to provide good—although not necessarily optimal—solutions to NP-hard problems with acceptable computing times. Evolutionary methods could therefore be expected to out-perform greedy methods, in general.

The principal and novel contribution of this paper is to introduce a new risk functional, vicinal risk, for training DTs, which addresses the challenging issue of how to regularize tree structures. To support this, we present statistically-founded comparison between trees induced using the new risk and conventional methods. Since, we believe, minimizing new risk functional can only be carried-out using an appropriate meta-heuristic, this paper reports the application of soft computing to an important problem in machine learning.

In general, training in machine learning is ill-posed [6] and empirical risk minimization (ERM) does not necessarily produce best generalization over an unseen test set, a problem which is exacerbated by small datasets; it is this ‘small data’ scenario we explicitly address in this paper. The deficiencies of ERM are illustrated in Fig. 1 for the trivial case of classifying linearly separable patterns with a plane. Reduction of the ER to zero can be achieved by any of the infinite number of hyperplanes passing between the two groups of patterns. In particular, hyperplane ‘B’—although minimizing the risk to zero—lacks the robustness to cope with even small noise perturbations of the pattern attributes. It is obvious that hyperplane ‘A’ will deliver the greatest ‘margin’ against noise. Since empirical risk minimization (ERM) does not necessarily produce acceptable margins, this motivates us to investigate a superior risk functional and apply it to decision trees.

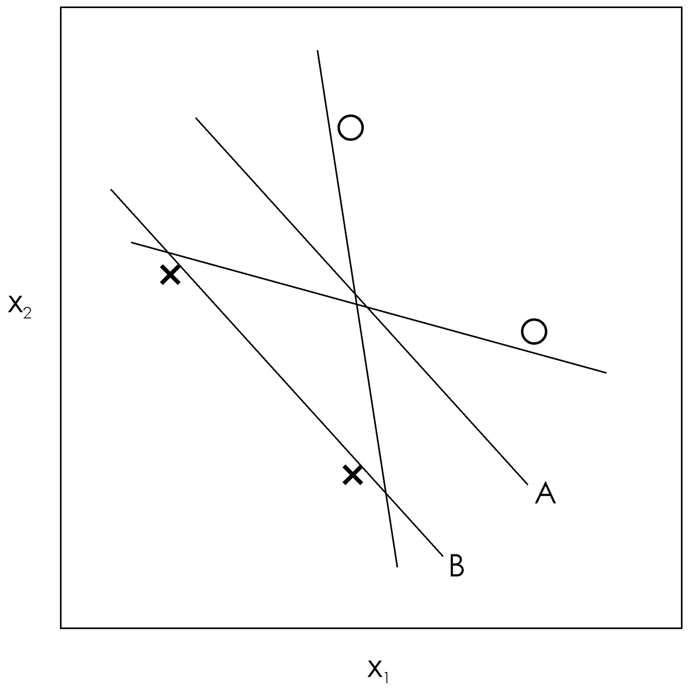


Figure 1: Illustration, for the simple case of classifying linearly separable patterns, of the deficiency of minimizing empirical risk. The crosses and circles represent patterns of differing classes.

The principal contribution of this paper is to report the induction of decision trees using *vicinal risk minimization* (VRM) [8] which displays significantly greater stability than ERM. Since this new risk is a continuous function, it is able to discriminate between the competing decision surfaces in Fig. 1 which (discrete) ERM cannot distinguish. We make statistical comparisons with decision trees induced with VRM and conventional ERM using the same GP method. As an underpinning baseline, we compare the above results with trees induced using the popular, deterministic C4.5 algorithm. We demonstrate that VRM is able to produce decision trees with superior generalization performance compared to C4.5, and GP-trained trees which minimize ER.

In Section 2 we describe the adaptation of VRM to decision trees. In Section 3 we discuss related work on decision trees and their training using genetic programming (GP); we review genetic programming and its single and multiple objective variants. We outline our experimental methodology in Section 4, and present experimental results in Section 5. We offer further insights into the application of VRM to decision trees in Section 6; Section 7 concludes the paper.

2. Vicinal Risk

Starting with the development of vicinal risk for a conventional scoring classifier given by Vapnik [8], for some set of ℓ training data, $\mathcal{D} = \{\mathbf{x}_1 \to y_1, \mathbf{x}_2 \to y_2, \ldots \mathbf{x}_\ell \to y_\ell\}$ drawn from a data distribution $P(\mathbf{x}, y)$, where $\mathbf{x}_i \in \mathbb{R}^N$ and $y \in \{-1, +1\}$, the task of training a scoring classifier is to select some discriminant function $f(\mathbf{x}) \to y$. We desire to select the $f(\mathbf{x})$ which minimizes the expected risk $R(f)$, which will ensure optimum generalization over future unseen examples drawn from $P(\mathbf{x}, y)$ where:

$$R(f) = \int L[f(\mathbf{x}), y] \, dP(\mathbf{x}, y) \tag{1}$$

and L is some loss function. Unfortunately, $P(\mathbf{x}, y)$ is not known in practice and so the conventional approach has been to approximate $P(\mathbf{x}, y)$ using the set of samples $\{\mathbf{x}_i, y_i\}$ $i \in [1 \ldots \ell]$:

$$P(\mathbf{x}, y) \approx \frac{1}{\ell} \sum_{i=1}^{\ell} \delta(\mathbf{x} - \mathbf{x}_i) \tag{2}$$

where δ is the Dirac delta function, and to minimize the *empirical risk*, R_{emp} (i.e., the expected 0/1 loss) over the training set. We can conveniently take the loss function to be [8]:

$$L[f(\mathbf{x}), y] = H[-y f(\mathbf{x})] \tag{3}$$

where H is the Heaviside step function. Thus for $\mathbf{x}$-values which would give rise to a misclassification, (3) is unity; conversely, for $\mathbf{x}$-values which yield correct classification, the loss is zero. Thus, empirical risk, R_{emp} is defined as:

$$R_{emp}(f) = \frac{1}{\ell} \sum_{i=1}^{\ell} H[-y_i f(\mathbf{x_i})] \tag{4}$$

As is clear from Section 1, the fundamental shortcoming of the 0/1 loss is due to its discrete nature, in particular, that a pattern is either classified correctly, in which case it contributes zero to the cumulative loss, or the pattern is misclassified and so contributes unity to the loss. Crucially, no account is taken of the *margin* by which a pattern is misclassified (or indeed, correctly classified). A misclassified pattern which is just the wrong side of a decision surface is weighted equally with a pattern that is a very large distance from the decision surface; intuitively, the latter case should be treated as more serious than the former. As a logical consequence, a pattern's distance from the decision surface should weight its contribution to the loss.

Vapnik [8] has motivated *vicinal risk* by assuming that the (unknown) data distribution is locally 'smooth' in which case $P(\mathbf{x}, y)$ can be approximated by placing a *vicinity function* on each training datum—this process can be thought of as either resampling or, equivalently, interpolating $\mathcal{D}$. Since the shortcomings of 0/1 loss are due to its discrete nature, smoothing the training set will have the effect of stabilizing the training process. Vapnik [8] described two possible types of vicinity functions, *hard* and *soft*. Hard vicinity functions have an abrupt cutoff at some distance from a training datum—under a 2-norm, this would be a ball or hypersphere centered on each datum. Whereas a hard vicinity function has a constant, non-zero value up to the cutoff distance and zero beyond, a soft vicinity function, such as a Gaussian kernel, typically has a peak value at the training datum and a monotonically-reducing value with increasing distance from the datum. Entirely equivalently, placing a kernel over each training datum can be viewed as approximating $P(\mathbf{x}, y)$ using a Parzen windows density estimator [9, 10] for which a Gaussian kernel is a natural choice. Here we develop the soft vicinity function approach because: i) it is more tolerant of the setting of scale of the kernel and ii) there is a technical requirement with hard vicinity functions that they do not overlap in pattern space [8].

Taking the loss function given in (3), analogous to minimizing (1), we wish to select the f which minimizes the vicinal risk, R_{VR} which is the expectation of (3) over the data distribution. Writing this functional in modified form from that given by Vapnik [8]:

$$R_{\text{VR}}(f) = \int L[f(\mathbf{x}), y]\, dP(\mathbf{x}, y) \tag{5}$$

$$\approx \frac{1}{\ell} \sum_{i=1}^{\ell} \int H[-y_i f(\mathbf{x})]\, G(\mathbf{x}|\mathbf{x}_i, \sigma_i^2) d\mathbf{x} \tag{6}$$

where $G()$ is the Gaussian kernel of variance σ_i^2 placed on the i-th datum. Here $P(\mathbf{x}, y)$ is approximated by the Parzen windows estimate of a sum of Gaussians. The integral within (6) has a straightforward interpretation as the hypervolume, in the N-dimensional pattern space, of the portion of the i-th kernel which falls on the 'wrong' side of the decision surface and would hence give rise to misclassification. A number of properties of vicinal risk minimization (VRM) is apparent:

- Under VRM, we seek to minimize a continuous function (6), thereby removing the problem with 0/1 loss of being discrete. Patterns contribute to the loss depending on their distance from the decision surface, or more strictly, the hypervolume of the kernel function falling on the 'wrong' side of the decision surface. It is clear that correctly-classified patterns a long way from the decision surface will make a very small contribution to the loss and will hence have a minimal influence on the placement of the decision surface—this is highly desirable since only data in the vicinity of the decision surface run the risk of misclassification and should 'negotiate' the location of the decision surface.

- At distances greater than $\simeq 3\sigma$ from the decision surface, the contribution to the loss of an incorrectly-labeled datum saturates at unity, conferring robustness to outliers.

- As $\sigma^2 \to 0$, the Gaussian kernel in (6) tends to a δ-function and so the vicinal risk tends to the empirical (i.e., 0/1) risk in (4). Thus empirical risk can be understood as a special case of vicinal risk.

- σ^2 defines a characteristic 'scale' for the learning problem which will vary by dataset.

Chapelle et al. [9] have directly minimized VRM for linear classifiers, assigning each training datum kernel its own value of σ_i^2 proportional to a measure of local density although the constant of proportionality had to be determined by cross-validation.

Key to the computational tractability of VRM is the evaluation of the integral under the summation in (6), the *vicinal loss* for the i-th pattern, which, in general, needs to be performed in N-dimensional pattern space over the arbitrary-shaped decision surface prescribed by f. In a monothetic decision tree (i.e., one that splits at each node on a single pattern attribute), this N-D integral can be reduced to a sequence of trivial, analytical 1D integrations, one per internal node decision, which we can consider as the probability of misclassifying the i-th training pattern at the node in question. By propagating the probabilities of errors at each internal node through to the leaf nodes, we evaluate the total probability of misclassification of the i-th training pattern. It is the expectation of these probabilities over the whole training set that we seek to minimize.

Ultimately, we desire a risk functional which is more predictive of (i.e., better correlated with) test error than empirical risk, which is a one-to-many mapping. Namely, minimizing a given risk functional will produce a classifier with superior generalization performance. We return to this theme in Section 6 where we present results that demonstrate that vicinal risk does indeed have this key property of superior correlation with test error. In Section 5 we show that VRM yields statistically lower generalization errors over a range of datasets.

3. Related Work

In this section we briefly review the relevant concepts in decision trees and genetic programming before describing our implementation of vicinal risk for decision trees in Section 3.3.

3.1. Decision Trees

Decision trees [10, 11] have remained an active area in pattern recognition for many decades, largely because they frame classification as a human-comprehensible sequence of questions. (In addition, they can handle mixed pattern attributes straightforwardly although in this paper we focus on real attributes.) Further, we consider only binary, axis-parallel DTs, that is, each (internal) tree node produces a two-way ‘split’ on a single pattern attribute since such trees have the advantage of ready human interpretation which more sophisticated tree architectures lack [11]. A typical decision tree is shown in Fig. 2 where the root and internal nodes represent some decision of the form $if(x_i < t)$, where $x_i \in \mathbb{R}$ is the i-th pattern attribute and $t \in \mathbb{R}$ is some threshold on this attribute for a specific node. If this predicate returns true, the left exit path from the node is taken. Otherwise the right path is taken. The leaf nodes of the tree represent some predicted class label. Under classification, a pattern vector is presented to the root node of the tree, and the process follows a single traversal path until it reaches a leaf node which indicates the predicted class of the pattern.

Key to the innovation in this paper is the replacement of the 'hard' $(x_i < t)$ decision within an internal node with a probabilistic split leading to multiple evaluation paths. It is thus relevant to review other DT architectures which employ 'soft' splits.

The notion of a soft split is far from new. Friedman [12] suggested handling a missing attribute in a DT by following both left and right successor nodes, thereby producing multiple paths of evaluation, and taking as the classifier prediction, the majority class of the terminal nodes reached.

Hierarchical mixtures of experts (HMEs) are tree-like structures in which a set of experts specialize on different regions of the input space, and a hierarchical set of gates form a weighted prediction over the set of experts [11, 13]. HMEs are fundamentally different from the work in this paper since, although they employ internal nodes with 'soft' outputs, they use this information to weight the predictions of an independent set of classifiers rather than as part of the classification process. Moreover, due to their complex architecture, HMEs do not lend themselves to human interpretation as a set of rules.

İrsoy et al. [14] have implemented 'soft' binary nodes in DTs by combining the predictions of the left and right sub-trees with a logistic gating function; this requires the recursive, parallel traversal of the tree to the leaf nodes and then the upwards propagation and combination of predictions at the internal nodes. The tree construction process is a greedy algorithm which grows the tree by adding nodes based on local improvement of the error metric. Yildiz and Alpaydin [15] have subsequently extended this work to include either L_1 or L_2 regularization at the individual node level.

Yildiz [16] has considered the optimal node thresholds for a univariate decision tree classifier which maximizes the margins although this is a local (i.e. per node) margin maximization; the method used for constructing the whole DT was not specified but, by implication, was a greedy method and therefore probably globally sub-optimal.

A number of authors [17, 18] have considered the problem of maximizing the margins at each internal node in *oblique* decision trees, typically using greedy construction algorithms. Again, oblique trees do not lend themselves to ready human interpretation.

Unfortunately, optimal decision tree induction is known to be NP-complete [1] and conventional algorithms employ ‘greedy’ methods which can be expected to be sub-optimal. In recent years, heuristic approaches have gained popularity for finding good, although not necessarily truly optimal, solutions to NP-hard problems using acceptable computer resources. GP is a natural approach to tree induction since it is typically formulated as a tree structure. The principal contribution of this work is to propose, within an evolutionary framework, a regularization method for the *whole* decision tree structure as opposed to within individual nodes [15, 17, 18].

3.2. Genetic Programming

Genetic programming is a population-based evolutionary meta-heuristic [19] loosely inspired by Darwinian ‘survival of the fittest’. GP starts with a randomly-generated population of individuals; here we have used the ramped half-and-half method of initialization [19]. Like all evolutionary algorithms, GP performs repeated iterations of: stochastically selecting two parents for ‘breeding’ biased in their fitness on the task in hand, followed by some genetic operation(s) to produce two offspring. Fitter individuals tend to be more likely to be chosen for breeding leading to a tendency for the offspring to be fitter.

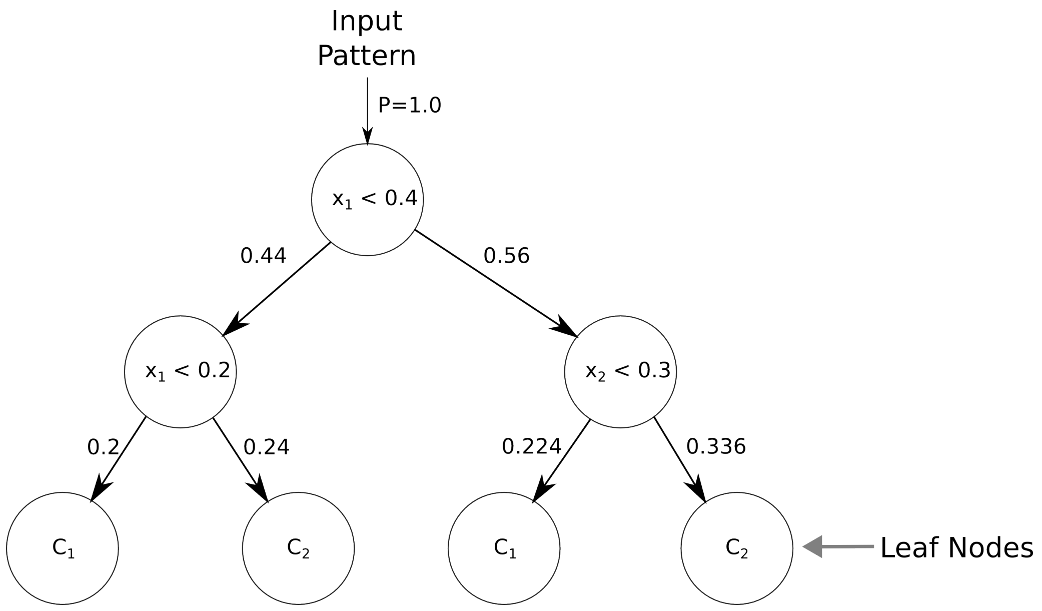


Figure 2: Illustration of the propagation of decision probabilities down a decision tree.

Typical genetic operations (and those used in this work) are: crossover, sub-tree mutation, and node mutation [19]. In crossover, a 'crossover' point is stochastically and independently chosen in each of the two parent trees and offspring are formed by exchanging the genetic fragments. In sub-tree mutation, a point is (again stochastically) chosen in each offspring and the sub-tree below this point replaced with a randomly-generated sub-tree. Finally, in node mutation a randomly-selected node in a tree is modified either by flipping the class in the case of a terminal node, or changing the decision variable or threshold in the case of an internal node. Node mutation can facilitate 'fine-tuning' of the tree.

The key driving force in evolutionary methods is the fitness objective, the measure of an individual's performance on the task in hand. Previous work on GP-evolution of DTs has mostly used the single objective of empirical risk calculated over the training set. A naively-used single objective, however, invariably results in 'bloat', the tendency of the size of the GP-trees to grow without limit but without any accompanying increase in performance. The issue of bloat has received a great deal of attention in the GP literature since it is highly undesirable—see [20] for a recent survey. In the single-objective work reported in Section 5 we have employed the dynamic depth control method [21] to control bloat.

A highly successful alternative to explicitly limiting tree depth for controlling bloat is the multi-objective parsimony method [19, 22]. Here, the fitness of an individual is a 2-vector comprising the risk, and a measure of tree complexity, here the number of nodes in the tree. The resulting population can be ranked for selection using Pareto dominance [23]. Given two vectors, $\mathbf{x}$ and $\mathbf{y}$ where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$, $\mathbf{x}$ is said to *dominate* $\mathbf{y}$ *iff*:

$$\begin{aligned} \mathbf{x} \prec \mathbf{y} \quad \textit{iff} \quad & \forall i : 1 \dots N \quad x_i \leqslant y_i \\ & \wedge \exists j : 1 \dots N \; x_j < y_j \end{aligned}$$

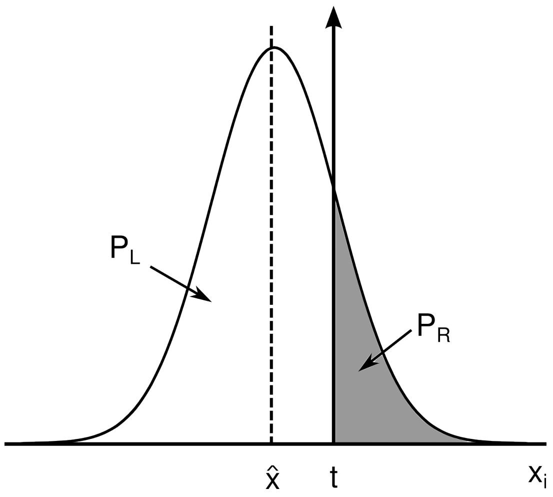
As well as effectively controlling bloat by exerting a selective pressure to minimize the tree size for a given value of risk, MOGP explores the range of possible models across the bias-variance trade-off [11]. It thus implicitly addresses the model selection problem in designing a classifier. In this work, we have used a GP adaptation of the steady-state Pareto Converging Genetic Algorithm (PCGA) [24] in which crossover, sub-tree mutation and node mutation are always applied – the effective crossover and mutation probabilities are both, therefore, 100%. The experimental parameters used in this work for both the single and multiple objective GP approaches are given in Section 4.

Typically, GP-induced decision trees have been trained to minimize only the single objective of empirical risk (or *misclassification error*) over a training set, but minimization of multiple objectives has been explored previously for DTs [25–28]. Usually in multi-objective optimization (MOO), the objectives conflict and so what tends to emerge is a set of trade-offs between the objectives. Haruyama and Qiangfu [29] have suggested that Pareto-based methods give the best compromise between error rate and tree size for DTs, and it is this approach we use here.

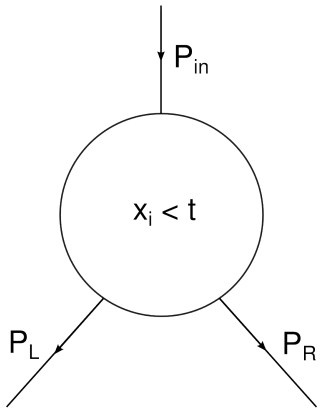
3.3. Vicinal Risk Minimization in Decision Trees

In a conventional binary, axis-parallel decision tree (DT) [11], each internal node implements a simple threshold, t on a single pattern attribute x_i where $i \in [1 \dots n]$. If $x_i < t$ then the recursive evaluation procedure follows the left sub-tree, otherwise it follows the right sub-tree. Eventually, this single evaluation path leads to a unique leaf node which indicates the predicted class of pattern $\mathbf{x}$. Training the decision tree by minimizing empirical risk over a training set using, say, GP, inevitably leads to the variability in the test error performance depicted in Fig. 1; note that this variability is a fundamental property of the discrete empirical risk measure, not of the training method.

For VRM, in terms of a single decision-tree node and considering an individual pattern attribute x_i, we can treat x_i as our best (i.e., mean) estimate $\hat{x}_i$ of the attribute value but superimpose on it a Gaussian to smooth the empirical distribution in (2). This is illustrated in Fig. 3a. Whereas training by conventional ERM would be framed in terms of a binary predicate of the form $x_i < t$ returning either true or false, VRM is framed in terms of calculating the *probability*, P_R that $x_i \geq t$ and the complementary quantity P_L which is the probability that $x_i < t$. P_R is shown as the shaded portion under the curve for $x_i \geq t$ in Fig. 3a. Note that if a probability value P_{in} is passed into a node, $P_{in} = P_L + P_R$. Rather than following a single evaluation path to either the left or the right child nodes as would be done with empirical risk, training with VR allows us to propagate continuous probability measures $P_L = 1 - P_R$, and P_R down to *both* the left and the right sub-trees, respectively, as illustrated in Fig. 3b. The probability values, $P_{L,R}$ are simply the areas under the probability density function (PDF) for the regions $x_i < t$ and $x_i \geq t$, respectively—see Fig. 3a.



(a)



(b)

Figure 3: (a) Probability of $x_i \geq t$ for a single decision node assuming , and (b) the propagation of probabilities to child nodes.

Calculation of the empirical risk in training proceeds by passing a pattern vector $\mathbf{x}$ to the root node of the tree and then following a single path of execution—to either the left or the right—at each internal node until a leaf node is reached. The label associated with this terminal node is the tree's class prediction for $\mathbf{x}$ and is either correct—in which case the loss is unchanged—or is incorrect in which case the running count of the number of errors is incremented by one. Passing the whole training set through the tree, and dividing the total number of errors by the cardinality of the training set yields the empirical risk.

For training with vicinal risk, however, each $\mathbf{x}$ is passed to the tree root and an initial probability of one is divided by the root node between left and right sub-trees. This process of splitting the probability passed into each node is continued recursively until the probabilities are propagated down to the leaf nodes. Note that under VRM we have a *multiple* paths of execution (only) during training since, in general, some probability is passed to the left and some to the right at each internal node. The vicinal risk for a pattern can be calculated by summing the probabilities that propagate to the leaf nodes that predict the *wrong* class. Averaging over the training set produces a continuous measure of the mean probability of misclassifying a training set pattern. Minimizing this vicinal risk (i.e., average *probability* of misclassification) using GP places the decision surface such that the final classifier has (approximately) the greatest margins under classification.

Our proposed *training* procedure thus involves minimizing the expected VR over a training set, which (approximately) maximizes the tree's margins. Thereafter, classification of query patterns proceeds as for conventional DTs with no further consideration of vicinal risk. That is, VRM only influences the training phase and the structure of the tree. (A further option—which we do not explore in this paper—is to calculate the vicinal risk while classifying an unknown pattern. The VR will thus be a measure of confidence in assigning a class prediction).

When the variances associated with all pattern attributes tend to zero, the tree traversal will reduce to a single evaluation path. Therefore, conventional ERM can be considered as a limiting case of VRM for no errors on the pattern attributes [9].

Considering the VRM training method in detail, Fig. 2 shows an example tree. Both internal and terminal nodes of the tree need to be slightly modified for the training procedure. An internal node receives some probability mass from its parent node and apportions this between left and right sub-trees according to the method described above. A terminal node indicates a class label but also accumulates the incoming probability masses for training exemplars for which the label predicted by the node is incorrect. In Fig. 2, for each training pattern, a probability of 1.0 is initially propagated into the root node of the tree which applies the test $x_1 < 0.4$. Depending on the value of pattern attribute x_1, t and the width of the distribution of x_1, some fraction of the probability will propagate to the left sub-tree, say 0.44, and the remainder, 0.56 will pass to the right sub-tree[1]. Fig. 3 depicts this process in detail. Following (arbitrarily) the left sub-tree in Fig. 2, the next node will apply the test $x_1 < 0.2$. Since a probability mass of 0.44 passes into this node, this will divide with, say, 0.2 passing to the left sub-tree and 0.24 passing to the right. Note that the sum of probabilities leaving an internal decision node is always equal to the probability which flows into that node. The process repeats along multiple paths until the total probability mass is distributed across all the leaf nodes. In the case of the example tree in Fig. 2, the class-1 (C_1) leaf nodes accumulate a probability of $0.2 + 0.224 = 0.424$ and the class-2 (C_2) nodes accumulate a probability of $0.24 + 0.336 = 0.576$. If this particular pattern is really of class C_1, its value of vicinal loss (i.e., probability of misclassification) is 0.576. Such a large value of vicinal risk (> 0.5) would imply the pattern is quite close to, but nonetheless on the 'wrong' side of, the decision surface since this DT would classify it as belonging to class C_2. This process of propagating unit probability down the tree for each training pattern is repeated for the whole training set and the vicinal losses summed. It is the average vicinal loss over the training set, the vicinal risk, that we wish to minimize to obtain the best compromise location of the decision surface.

Although the previous paragraph presents a general outline of training by VRM, the tree in Fig. 4 illustrates a special case where refinement is necessary. No patterns for which $x_1 \geq 0.5$ can propagate down to the right sub-tree of the '$x_1 < 0.5$' node since they will have all been removed by the $x_1 < 0.4$ test implemented by the root node. Therefore, the division of probability at a node has to take into account previous decisions on the same variable, in this case x_1. Consequently, the training procedure has to be refined to propagate not only the probability mass from the node above but also the upper (t_u) and lower (t_l) limits on the thresholds already applied

[1]These arbitrary values of probability have been used purely for illustration.

to each attribute. (Starting at the root node, the upper and lower limits are initially $\pm\infty$, respectively.) Thus the probabilities propagated to left and right sub-trees can be generalized to:

$$P_L = \int_{t_l}^{t} p(x_i)dx_i \quad \text{and} \quad P_R = \int_{t}^{t_u} p(x_i)dx_i$$

where t is the current node threshold and $p(x_i)$ the PDF for x_i, and the integrals involve the calculation of the cumulative probability. Since $P_{L,R}$ must be strictly non-negative, if $t \leq t_l$ then $P_L = 0$, and if $t \geq t_u$ then $P_R = 0$. The thresholds t_u and t_l, of course, need to be updated and propagated onward during training on each pattern.

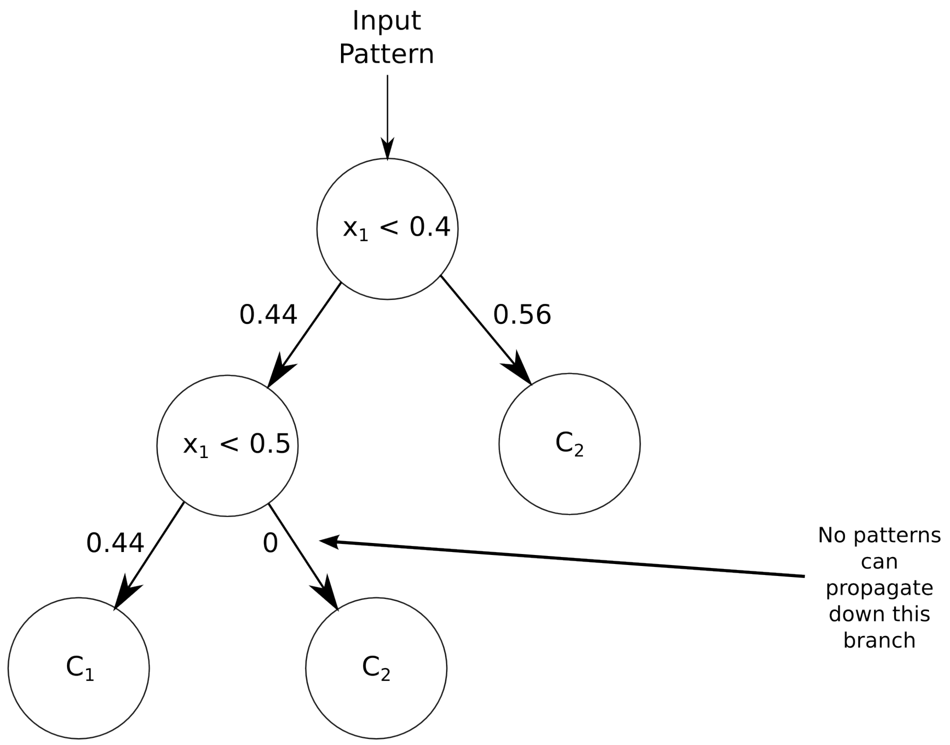


Figure 4: Example for which the basic propagation of probabilities has to be refined.

4. Methodology

We compare results of training decision trees by ERM and VRM over a number of datasets, both real and synthetic, to explore the properties of vicinal risk. In addition, we have also employed both single objective genetic programming (SOGP) and multiple objective genetic programming (MOGP). Throughout, we use the conventional C4.5 method[2] as a benchmark; although pruning methods are frequently used with C4.5, we have found that in this work that they have little effect.

4.1. Datasets

The two-class synthetic datasets used are:

- Linearly-separable 2D Data: A linearly-separable dataset comprising 20 training examples per class with an (optimal) decision surface inclined at 45 degrees to the attribute axes. The test set comprises 1000 examples per class.

- 2D Gaussian Dataset: Bivariate Gaussians with mean vectors $(0, 0)^T$ and $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^T$ respectively, and identical unit covariance matrices. The training set comprised 90 data and the test set, 10 000 data. The two classes are balanced. The Bayes' error on the test set is estimated to be about 31%.

- 10D Gaussian Dataset: 10D Gaussians with mean vectors $(0 \ldots 0)^T$ and $(\frac{1}{\sqrt{10}} \ldots \frac{1}{\sqrt{10}})^T$ respectively, and identical unit covariance matrices. The training set comprised 1000 data per class and the test set 10 000 data. The two classes are balanced. The Bayes' error on the test set is also about 31%.

The datasets above were chosen as they have Bayes-optimal decision surfaces inclined at 45 degrees to the attribute axes [10]. Such datasets are challenging to an axis-parallel DT as the decision surface has to be approximated with a 'staircase' of axis-parallel splits. The sizes of the test sets were chosen to give accurate estimates of the test errors.

Eighteen, two-class real datasets were taken from the UCI Machine Learning Repository and the Statlog project, and are summarized in Table 1. Any instances with missing attributes were removed. The Adult dataset was randomly sub-sampled to reduce its size to 20% of the original. Ripley's version

[2]We have used the Weka version 3.6 implementation of C4.5, known as 'J48' in Weka. See `http://http://www.cs.waikato.ac.nz/ml/weka/`. We have used the default J48 settings in Weka.

of the Pima dataset[3] has been modified to remove implausible records. The Glass dataset was converted to a two-class problem (float glass vs. non-float glass). Any categorical attributes were recoded to integers $(0, 1, \ldots)$ in the sequences listed in the UCI/Statlog documentation; binary attributes were recoded as true/yes = 1 and false/no = 0. Finally, we have pre-processed the data such that all attributes in the training set had unit variance, and then scaled each individual test set accordingly.

It is worth noting that our VRM training procedure assumes real attributes. Some of the attributes in the datasets in Table 1 are categorical, not real. In part, we have examined these datasets to explore robustness to any assumptions, and partly because only very few of the standard benchmark UCI datasets contain purely real attributes. Regardless, the use of categorical attributes in regression is well-established [30].

4.2. Statistical Testing

As suggested by Demšar [31], we have adopted two non-parametric statistical tests depending on the experimental situation: the Wilcoxon signed-ranks test where the comparison is between two algorithms on a single data set, and the Friedman test for comparing between multiple algorithms over multiple data sets. For the Friedman test we used the improved Iman-Davenport statistic [32] followed by the step-down post hoc correction to the critical value [33]. See [31] for further details. The UCI/Statlog datasets were repeatedly split into two equal-sized training and test folds, and the expected test error estimated by averaging over ten replications of this splitting process.

Due to the fact that GP is a non-deterministic approach where the final solutions may vary depending on the starting population and evolutionary path, for each fold, we carried out 30 independently-initialized runs and took the median test error as the central tendency for statistical testing.

4.3. GP Parameters

The genetic programming parameters used in this work are summarized in Table 2. The numbers of iterations used were adjusted empirically to account for the varying difficulties of the test problems; for a given dataset, training with both vicinal and empirical risks used identical numbers of iterations to ensure a fair test.

[3]Downloadable from `http://www.stats.ox.ac.uk/pub/PRNN/`.

Table 1: Summary of the 18 UCI/Statlog datasets used, with the number of examples and the number/type of attributes.

Dataset	#Examples	#Real	#Integer	#Ordinal	#Binary	#Nominal
Adult	904	0	6	0	0	8
Australian Credit	690	3	3	0	4	4
Bands	365	13	6	0	0	0
BUPA	345	0	6	0	0	0
Credit Approval	653	3	3	0	4	5
German Credit	1000	0	7	0	1	12
Glass	163	9	0	0	0	0
Haberman	306	0	3	0	0	0
Heart	270	6	0	1	3	3
Hepatitis	80	2	4	0	13	0
Ionosphere	351	32	0	0	2	0
Mammographic	830	0	5	0	0	0
Pima (Ripley)	532	2	5	0	0	0
Pima (Original)	768	2	6	0	0	0
Spect Heart	267	0	44	0	0	0
Sonar	208	60	0	0	0	0
Wisconsin (Original)	683	0	9	0	0	0
Wisconsin (Diagnostic)	569	30	0	0	0	0

Table 2: GP parameters used in this work

Population size	100
Population initialization	Ramped half-and-half [19]
Crossover	Point crossover [19]
Mutation	Point mutation [19] & Node mutation [34]
Crossover probability	100% – see text
Mutation probability	100% – see text
Static depth limit	10
Dynamic depth limit	See [21]

Number of iterations	Separable = 20K
	2D Gaussian = 20K
	10D Gaussian = 1 million
	UCI = 200K

5. Experimental Results

We report the results of training by both minimizing the empirical and vicinal risk measures. In this section we report results for the median test errors corresponding to the trees which had the lowest training errors taken over 30 independent runs. We have used zero-mean, normally-distributed attribute kernels with $\sigma^2 = 0.1$ for all datasets; we discuss this choice in Section 5.4. The repeatability of the results is considered in Section 5.5 and the variability of training considered in Section 5.6.

Although we have explored single-objective GP (SOGP), on the three synthetic datasets we find that over 100 independent training sets, there is no evidence of statistical difference between the test errors for SOGP and MOGP on a given risk functional. For example, for the 2D Gaussian dataset, comparing ERM-trained trees for SOGP and MOGP yields a p-value of 0.5026 for the Wilcoxon signed-rank test. Similarly, comparing VRM for the same dataset gives a p-value of 0.7014. That is, training with single or multiple-objectives for a given risk functional yields essentially the same test error. Where there is a difference, however, is in the sizes of the evolved trees, with the use of multiple objectives producing trees around 40% to 200% smaller (depending on dataset) implying that MOGP leads to faster training since the runtime is dominated by the fitness evaluations. Consequently, we therefore present only test errors for MOGP.

5.1. Linearly Separable Data

The results of training on the synthetic separable data set are summarized in Table 3. The clear ranking of test errors is: training by VRM, training with empirical risk, and C4.5.

Table 3: Median test error results—synthetic datasets.

Dataset	**Method**	**Test Error**	**#Nodes**
Separable	C4.5(J48)	0.1064	5
	ERM MOGP	0.0689	23
	VRM MOGP	0.0245	55
2D Gaussian	C4.5(J48)	0.3683	5
	ERM MOGP	0.3596	31
	VRM MOGP	0.3429	53
10D Gaussian	C4.5(J48)	0.3941	157
	ERM MOGP	0.3910	193
	VRM MOGP	0.3821	225

The typical decision surfaces generated by these trees are shown in Fig. 5. The C4.5 decision surface is very rudimentary explaining this method's poor error performance. The decision surfaces induced by SO- and MOGP make better attempts at approximating the true decision surface although they still deviate well outside the margins (dotted lines). We can also infer that both GP trees contain many more nodes than necessary to implement the decision surfaces shown in Fig. 5. For example, the ERM-SOGP decision surface in Fig. 5 would require a minimum of 11 nodes whereas the actual evolved DT contains 43 nodes. Redundant (non-functional) code is a common feature of GP trees [20] and it appears MOGP is better able than SOGP to reduce, although not eliminate, this.

The decision surfaces for training by VRM are shown in Fig. 6. It is clear that the inclined surfaces are much more complicated than for ERM (shown

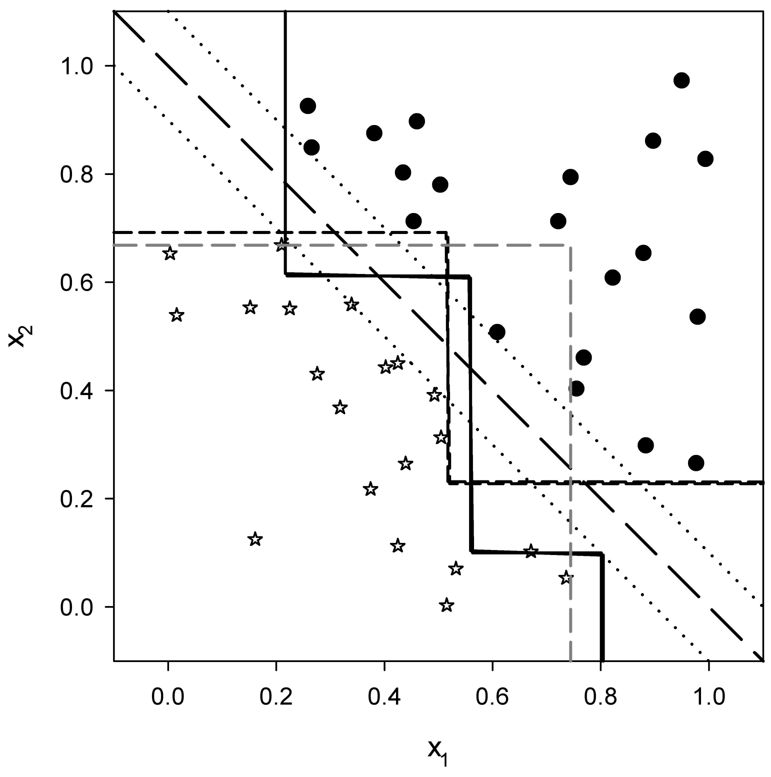


Figure 5: Decision surfaces obtained by ERM for the linearly-separable data (stars and circles). The diagonal dashed lines is the optimal decision surface and the light dotted lines above and below this are the margins. The gray dashed line is for C4.5, the solid line for SOGP, and the medium-dashed line for MOGP.

in Fig. 5) but come much closer to the Bayes-optimal surface. In addition, the VRM surfaces come far closer to staying within the two decision margins.

5.2. 2D Gaussian Data

The test errors for the 2D Gaussian data are also shown in Table 3. Typical decision surfaces induced by ERM and C4.5 are shown in Fig. 7 and 8. Again, the C4.5 decision surface is overly simple for this data set. The ERM decision surfaces for SO- and MOGP are rather complex although both basically resemble the C4.5 surface but with seemingly 'opportunistic' disjoint boxes which correctly classify one or two extra training points. For example, in Fig. 7, the two boxes, top-left and bottom-right, serve to correctly classify the two class '1' training instances (open circles) contained within those

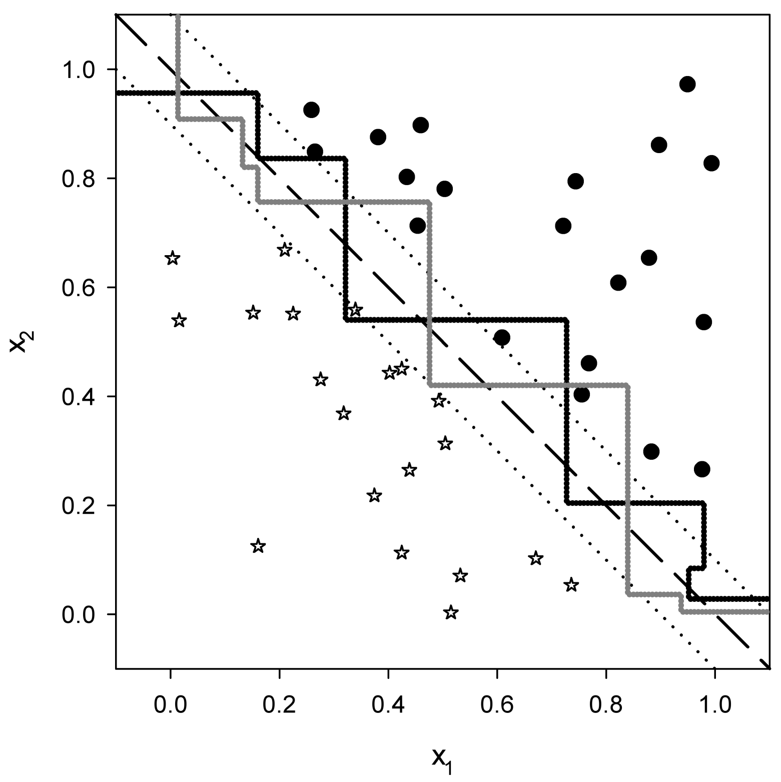


Figure 6: Decision surfaces obtained training on VRM for the linearly-separable data (stars and circles). The diagonal dashed lines is the optimal decision surface and the light dotted lines above and below this are the margins. The solid line is for SOGP, and the grey line for MOGP.

boxes. The additional features generated by GP training produces a slightly better test error compared to C4.5 although it appears to be something of an over-specialization on the particular training examples and therefore, arguably, over-fitting.

The two decision surfaces for VRM are shown in Fig. 9 from which it is clear that VRM has produced decision surfaces which well approximate the Bayes-optimal surface, at least in the top-left quadrant of the pattern space. Closer inspection of the decision surfaces in the bottom-right quadrant reveals an absence of Class 1 training data (open circles) in the region of the Bayes-optimal surface. Given this training set, the induced decision trees are reasonable.

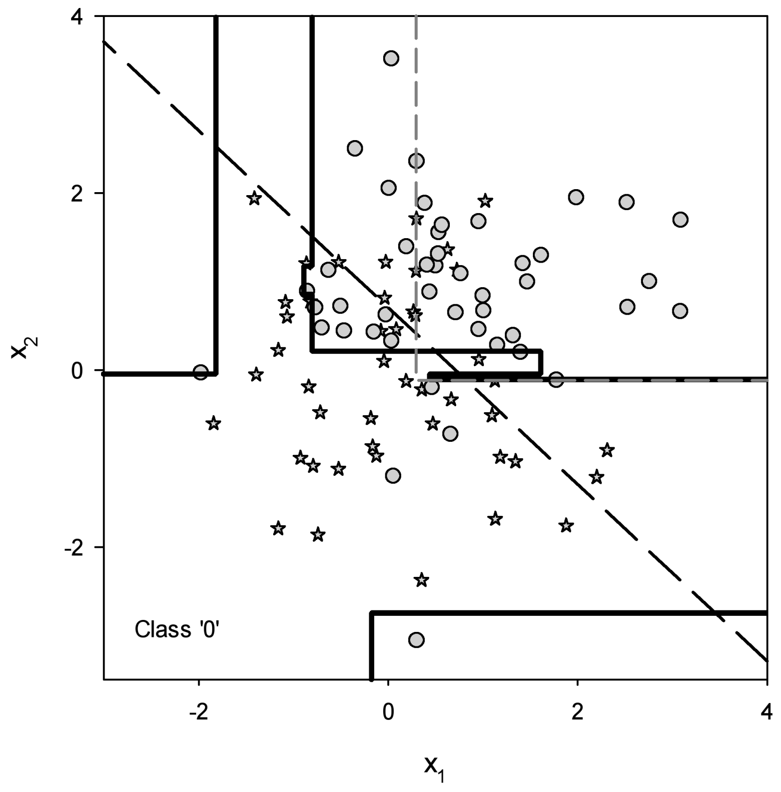


Figure 7: Decision surfaces obtained training by ERM for the 2D Gaussian data (stars and circles) by SOGP. The diagonal dashed lines is the Bayes-optimal decision surface. The gray dashed line is for C4.5 (J48). The decision regions for class '0' are the interiors of the regions labeled Class '0'.

5.3. 10D Gaussian Data

The test errors for the 10D Gaussian data are shown in Table 3 and a similar pattern can be seen as for the 2D Gaussian dataset. Again, MOGP results in smaller trees than SOGP, this time by slightly more than a factor of two.

5.4. UCI and Statlog Data

We have repeated the experiments of the preceding sections on the eighteen benchmark UCI and Statlog datasets described in Section 4.1, and applied the statistical testing procedure of Section 4.2. Since experience with the synthetic data indicates that there is little between SOGP and MOGP

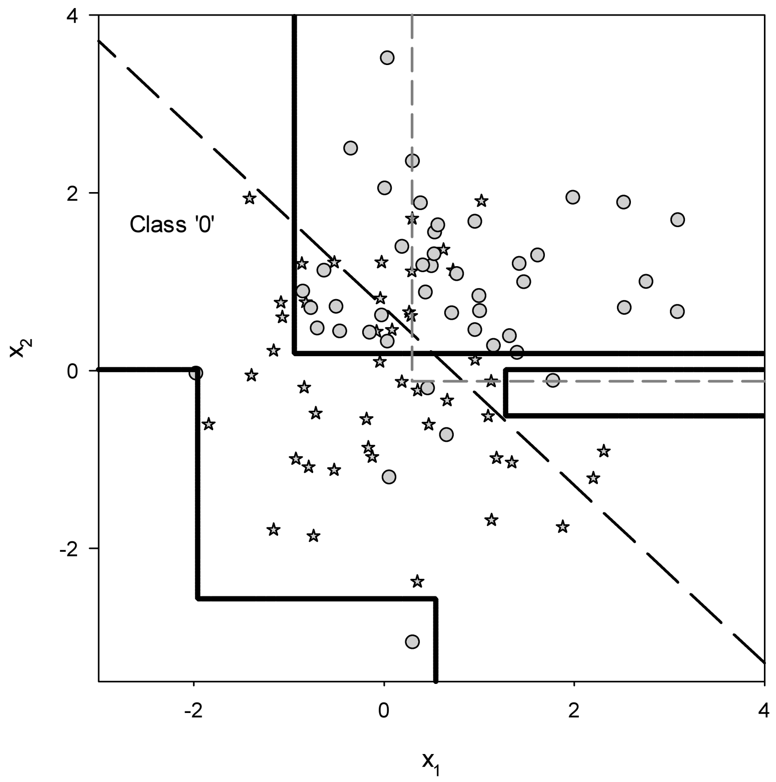


Figure 8: Decision surfaces obtained training by ERM for the 2D Gaussian data (stars and circles) by MOGP. The diagonal dashed lines is the Bayes-optimal decision surface. The gray dashed line is for C4.5 (J48). The decision regions for class '0' are the interiors of the regions labeled Class '0'.

training other than MOGP producing systematically smaller trees, we have considered only MOGP-trained trees.

The mean test errors over 10 splits of the datasets are shown in Table 4 together with the ranks, shown in parentheses, computed in the Friedman test. In most cases, the VRM trees have the smallest errors and are therefore of first rank, the only exceptions being the BUPA, Glass and Mammographic datasets; Adult is a draw with C4.5. The mean rank for VRM is notably lower than for the other two algorithms. Pairwise comparison between C4.5 and ERM shows mixed rank results.

From the data in Table 4, we have computed the Iman-Davenport F-statistic which, for 2 and 34 degrees of freedom, is 8.77 corresponding to a p-value of 2.4×10^{-3} providing strong evidence for rejecting the null hypothesis

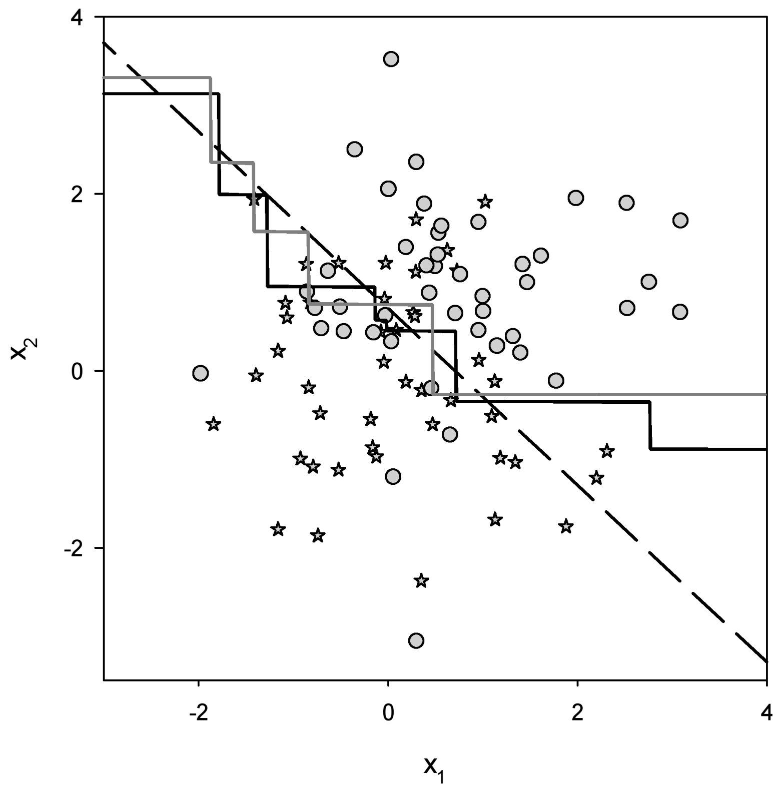


Figure 9: Decision surfaces obtained training by VRM for the 2D Gaussian data (stars and circles). The diagonal dashed lines is the Bayes-optimal decision surface. The solid line is for SOGP, and the gray line for MOGP.

that the algorithms are equivalent.

We have then considered VRM as a control method and followed Holm's treatment to compare VRM against both C4.5 and ERM leading to p-values of 1.15×10^{-3} and 5.96×10^{-3}, respectively. Again this is strong evidence to reject the null hypotheses and to infer that VRM is superior to both C4.5 and ERM.

To compare C4.5 and ERM, we have repeated the above procedure but making C4.5 the control method. The p-value is 0.617 which implies that there is very little evidence to reject the null hypothesis that C4.5 and ERM perform identically.

Garcia et al. [35] have suggested that the Friedman aligned-ranks test may be preferable when the number of algorithms being compared is low. The Friedman aligned-ranks produces even stronger evidence to support the

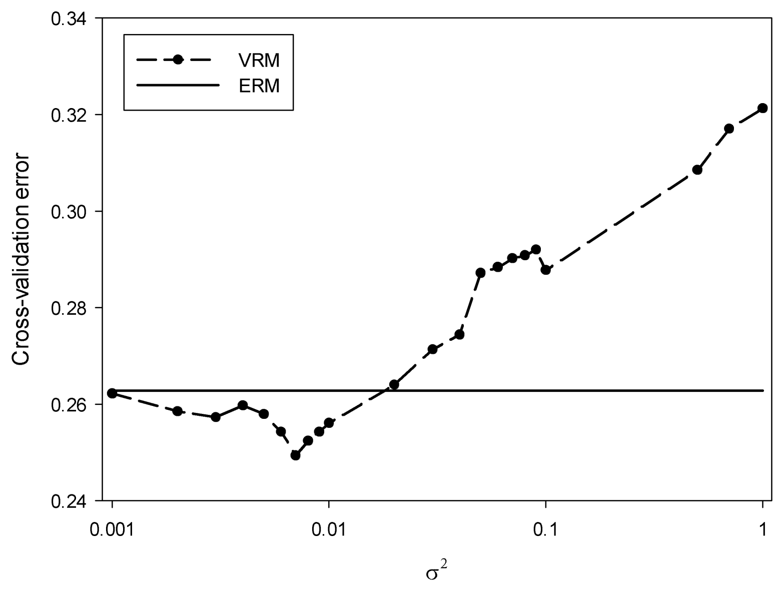


Figure 10: Mean test error over 10 replications for the Glass dataset as a function of σ^2. The solid line shows the (constant) value for ERM as a reference.

superiority of VRM over both C4.5 and ERM ($p = 5.96 \times 10^{-10}$ and 1.81×10^{-7}, respectively). Comparing C4.5 and ERM yields a p-value of 0.33, still not sufficient evidence to reject the null hypothesis. The conclusions using this test remain unchanged over the regular Friedman test.

Although VRM has been formulated here to deal with real attributes, it is interesting to observe that for the only purely-real UCI problem, Glass, VRM is the lowest-ranked algorithm. We have made no attempt in the above experiments to tune the value of σ^2, rather taking a fixed value of 0.1 which appeared a reasonable compromise value for all datasets. Fig. 10 shows the mean test errors over 10 replications for Glass but varying σ^2. It is clear that $\sigma^2 = 0.1$ is sub-optimal for this dataset and that a value of $\sigma^2 \approx 0.007$ gives a minimum test error of 0.2493, which would easily elevate VRM to rank 1 in Table 4; a similar observation holds for the BUPA dataset. In general, tuning σ^2 could, of course, be carried-out for a given dataset by cross validation. We reiterate that the statistical results above were obtained with a single, fixed value of $\sigma^2 = 0.1$.

To summarize the key findings on the real datasets: i) VRM significantly outperforms both C4.5 and ERM, and ii) the performance of C4.5 and ERM are not statistically different.

Table 4: The Friedman ranks of three algorithms (C4.5, ERM and VRM). The numbers in parentheses are the ranks computed in the Friedman test; where algorithms return the same error the rank is averaged.

Dataset	C4.5		ERM		VRM	
Adult	0.1540	(1.5)	0.1594	(3)	0.1540	(1.5)
Australian Credit	0.1609	(3)	0.1549	(2)	0.1490	(1)
Bands	0.3699	(3)	0.3650	(2)	0.3489	(1)
BUPA	0.3780	(1)	0.3902	(2.5)	0.3902	(2.5)
Credit Approval	0.1557	(3)	0.1485	(2)	0.1454	(1)
German Credit	0.2884	(2)	0.2944	(3)	0.2866	(1)
Glass	0.2793	(2)	0.2628	(1)	0.2876	(3)
Haberman	0.2784	(2)	0.2961	(3)	0.2725	(1)
Heart	0.2556	(3)	0.2452	(2)	0.2430	(1)
Hepatitis	0.1850	(3)	0.1763	(2)	0.1650	(1)
Ionosphere	0.1216	(2)	0.1247	(3)	0.1068	(1)
Mammographic	0.1723	(1)	0.1775	(2)	0.1828	(3)
Pima (Ripley)	0.2613	(3)	0.2556	(2)	0.2462	(1)
Pima (Original)	0.2784	(3)	0.2770	(2)	0.2671	(1)
Spect Heart	0.2470	(3)	0.2425	(2)	0.2298	(1)
Sonar	0.2644	(2)	0.2798	(3)	0.2442	(1)
Wisconsin (Original)	0.0477	(3)	0.0423	(2)	0.0385	(1)
Wisconsin (Diagnostic)	0.0709	(3)	0.0693	(2)	0.0621	(1)
Mean Ranks		2.4167		2.2500		1.3333

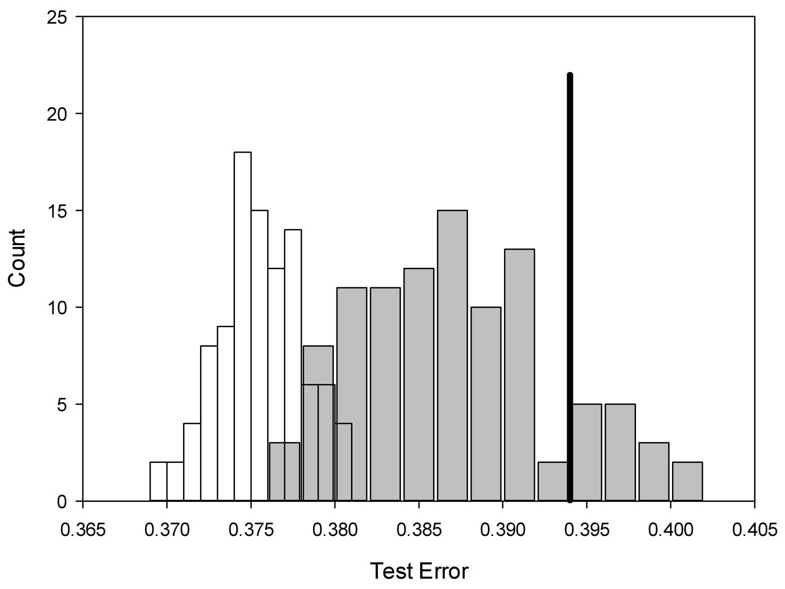


Figure 11: Histograms of tests errors for training on the 10D Gaussian dataset (MOGP). Open bars = VRM. Filled bars = ERM. Solid black line = (deterministic) C4.5 result.

5.5. Repeatability of Training

It has been argued [31, 36] that other properties of algorithms should be considered rather than simply testing on benchmark datasets. Although VRM has been demonstrated to yield lower test errors than ERM, the *consistency* with which such a result can be produced is an important criterion. We examined the histograms of test errors for the competing training methods for a fixed training set but 100 independent initializations to provide a measure of the repeatability of the stochastic GP search process. For brevity, only the typical results for the 10D dataset and MOGP are shown in Fig. 11. VRM is much more likely to produce more consistent test error values with more compact distributions compared to ERM. The modes from VRM training are also at lower values than for ERM, and the two histograms overlap little, implying that VRM produces consistently superior training to ERM. The modes of the ERM distributions are lower than the (deterministic) C4.5 results although by only a small amount. The superiority of VRM training relative to ERM and C4.5 is manifest.

Further, the VRM formulation embeds the variance on the pattern attributes as a parameter. It is appropriate to ask: how sensitive is the training outcome to the value of attribute variance σ^2? We have explored this issue by varying the σ^2 up and down by a factor of 10 and examining the resulting test errors. The test error is observed to be fairly insensitive to the exact value of σ^2 with a slight increase as σ^2 is reduced. For example, for the 2D Gaussian dataset, decreasing σ^2 by $\times 10$ increases the test error from 0.3201 to 0.3300; increasing σ^2 by $\times 10$ changes to test error to 0.3217. This effect can be understood in terms of a physical analogy since reducing σ^2 reduces the distance in pattern space over which an individual datum can affect a reduction on its own vicinal loss. The compromise placement of the decision surface over the whole training set is thus slightly degraded. The overall 'tuning' effect is, however, small.

5.6. Variability of Training

Complementary to the question of the repeatability of training is the question: how variable is the test error when the training set is varied? We have investigated this issue by examining the spread of test errors using 100 independent training sets for the 2D and 10D Gaussian problems. Again, VRM produces more compact distributions with modes at smaller values than ERM. C4.5 produces the poorest results with the largest variability which is interesting since it is a deterministic method, unlike the stochastic search employed for ERM and VRM; again see [37] for more details. Wilcoxon signed-rank tests on these data imply that VRM is superior to both ERM and C4.5 with p-values of effectively zero (Z-values of -6.75 to -8.64).

6. Discussion

An enduring problem in empirical modeling of data—either regression or classification—has been how to ensure best generalization over the set of, as yet unseen, examples the model would encounter in operation. Due to the ill-posedness of the learning problem, minimization of some risk, such as empirical risk in the case of classification, has typically been supplemented with an a priori 'smoothness' or complexity measure, most commonly framed as *regularization* [6, 8]. Minimizing the ERM over a training set does not necessarily imply a minimum risk over an independent test set. In Figure 12 we plot the correlations between training risk (ERM or VRM) and test error for the 10D synthetic Gaussian datasets for examples of trees ranging from randomly-created through partially trained to fully trained by GP. From Figure 12(a) for conventional ERM, it is clear that there is a many-to-1 correspondence between ERM and test error. The test error which results from a particular instance of a training procedure is presumably random. Figure 12(b) for VRM presents as a 'cloud' of correspondences but, most crucially, this cloud reduces to a *cusp* near the minimum of vicinal risk. In fact, the minimum VR does not coincide exactly with the minimum of observed test error, but minimizing VR gives a single, more consistent and repeatable optimization of test error.

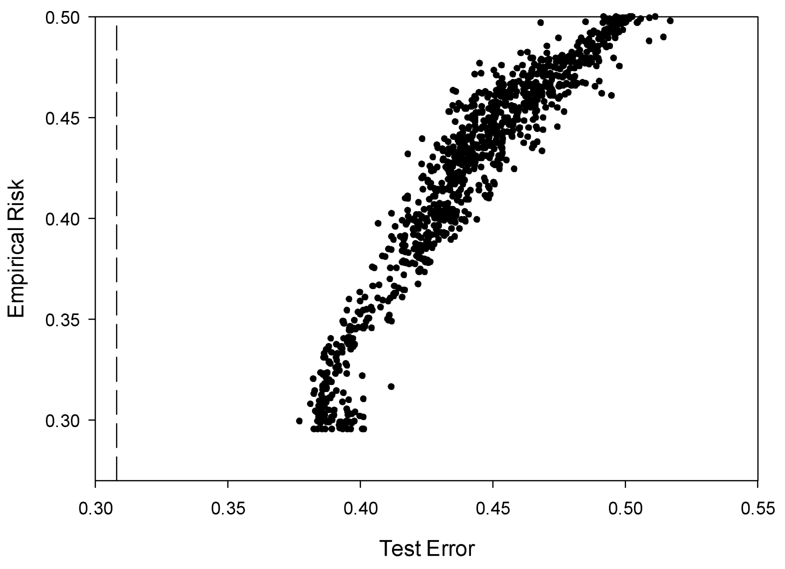
An intuitive understanding of VRM can be gained by noting that a given training pattern can minimize its own vicinal loss (i.e., its probability of misclassification) by egocentrically forcing the decision surface as far away from itself as possible. Every pattern in the training set, however, will try to use a similar strategy. What results will be a compromise placement of the decision surface which provides (approximately) the lowest expected probability of misclassification over the whole training set. In the sense that the decision surface will be forced as far away from the training data as possible, VRM approximately maximizes the classifier margins.

Although we have only considered 2-class problems in this initial report, the vicinal risk approach presented here is readily extensible to multi-class problems. In Section 3.3, we describe how, for a 2-class problem, the probabilities of misclassification were propagated down to the leaf nodes of the decision tree. This same mechanism applies equally well to multi-class problems, where our objective is still to minimize the overall probability of misclassification over the training set. Any probability mass propagating down to a leaf node that predicts the wrong class (compared to the true class of the training exemplar) will contribute to the probability of error. The practical demonstration of the approach presented here for multi-class problems will be an area for future work.

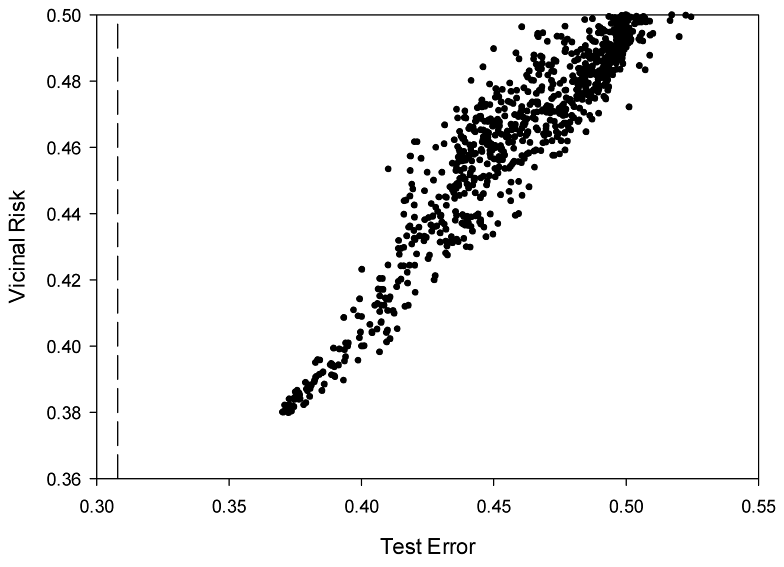
Finally, despite the fact that Vapnik's formulation of vicinal risk strictly only applies for real-valued attributes, the results shown in Section 5.4 for the UCI datasets indicates that the method is able to handle nominal attributes if these attributes are encoded as integers. A more detailed study of application to nominal/categorical attributes is another area for future work.

7. Conclusions

In this paper we have described the application of Vapnik's vicinal risk minimization for training axis-parallel decision trees. This risk functional is founded on propagating the uncertainty in the pattern's attributes through to uncertainties on the assigned class labels during training. By minimizing



(a)



(b)

Figure 12: Correlation between the (a) ERM and (b) vicinal risk, and test error. 10D Gaussian dataset. The vertical dashed line is the estimated Bayes' error. (Note the different, arbitrary ordinate scales.)

the total uncertainty over the training set, VRM approximately maximizes the margins of the trained classifier; in this way, we have performed a regularization over the whole decision tree rather than at the level of individual decision nodes. We have conveniently performed the necessary training optimization using genetic programming.

For synthetic datasets, VRM exhibits a consistently superior performance to both C4.5 and ERM. In addition, VRM produces intuitively pleasing decision surfaces, at least for the 2D datasets. On the benchmark UCI/Statlog datasets, we again observe highly-significant statistical differences in favor of VRM. Further, training with vicinal risk displays high degrees of repeatability and low variability.

[1] L. Hyafil, R. L. Rivest, Constructing optimal binary decision trees is NP-complete, Information Processing Letters 5 (1) (1976) 15–17.

[2] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

[3] R. Barros, M. Basgalupp, A. C. P. L. F. de Carvalho, A. Freitas, A survey of evolutionary algorithms for decision-tree induction, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42 (3) (2012) 291–312.

[4] J. Koza, Concept formation and decision tree induction using the genetic programming paradigm, in: 1^{st} Workshop on Parallel Problem Solving from Nature (PPSN1), Dortmund, Germany, 1990, pp. 124–128.

[5] P. G. Espejo, S. Ventura, F. Herrera, A survey on the application of genetic programming to classification, IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews 40 (2) (2010) 121–144.

[6] V. Cherkassky, F. M. Mulier, Learning from Data: Concepts, Theory and Methods, 2^{nd} Edition, Wiley-IEEE Press, 2007.

[7] G. Folino, C. Pizzuti, G. Spezzano, Genetic programming and simulated annealing: A hybrid method to evolve decision trees, in: European Conference on Genetic Programming (EuroGP2000), Edinburgh, Scotland, 2000, pp. 294–303.

[8] V. N. Vapnik, The Nature of Statistical Learning Theory, 2^{nd} Edition, Statistics for Engineering and Information Science, Springer, New York, 2000.

[9] O. Chapelle, J. Weston, L. Bottou, V. Vapnik, Vicinal risk minimization, in: T. K. Leen, T. G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems 13 (NIPS 2000), Denver, CO, 2000, pp. 416–422.

[10] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Recognition, 2nd Edition, John Wiley & Sons, New York, 2001.

[11] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2^{nd} Edition, Springer-Verlag, 2009.

[12] J. H. Friedman, A recursive partitioning decision rule for nonparametric classification, IEEE Transactions on Computers 26 (4) (1977) 404–408.

[13] S. Yuksel, J. Wilson, P. Gader, Twenty years of mixture of experts, IEEE Transactions on Neural Networks and Learning System 23 (8) (2012) 1177–1193.

[14] O. İrsoy, O. T. Yildiz, E. Alpaydin, Soft decision trees, in: 21^{st} International Conference on Pattern Recognition (ICPR 2012), Tsukuba, Japan, 2012, pp. 1819–1822.

[15] O. T. Yildiz, E. Alpaydin, Regularizing soft decision trees, in: 28^{th} International Symposium on Computer and Information Sciences (ISCIS 2013), Paris, France, 2013, pp. 15–21.

[16] O. T. Yildiz, Univariate decision tree induction using maximum margin classification, Computer Journal 55 (3) (2012) 293–298.

[17] D. Wu, K. P. Bennett, N. Cristianini, J. Shawe-Taylor, Large margin trees for induction and transduction, in: 16^{th} International Conference on Machine Learning (ICML '99), Bled, Slovenia, 1999, pp. 474–483.

[18] R. Tibshirani, T. Hastie, Margin trees for high-dimensional classification, Journal of Machine Learning Research 8 (2007) 637–652.

[19] R. Poli, W. B. Langdon, N. F. McPhee, A Field Guide to Genetic Programming, Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008.

[20] S. Silva, S. Dignum, L. Vanneschi, Operator equalisation for bloat free genetic programming and a survey of bloat control methods, Genetic Programming and Evolvable Machines 13 (2) (2012) 197–238.

[21] S. Silva, E. Costa, Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories, Genetic Programming and Evolvable Machines 10 (2) (2009) 141–179.

[22] Y. Zhang, P. I. Rockett, A generic optimising feature extraction method using multiobjective genetic programming, Applied Soft Computing 11 (1) (2011) 1087–1097.

[23] C. A. C. Coello, G. B. Lamont, Applications of Multi-Objective Evolutionary Algorithms, Vol. 1 of Advances in Natural Computation, World Scientific, Singapore, 2004.

[24] R. Kumar, P. I. Rockett, Improved sampling of the Pareto-front in multiobjective genetic optimizations by steady-state evolution: A Pareto converging genetic algorithm, Evolutionary Computation 10 (3) (2002) 283–314.

[25] H. Zhao, A multi-objective genetic programming approach to developing Pareto optimal decision trees, Decision Support Systems 43 (3) (2007) 809–826.

[26] T. M. Khoshgoftaar, L. Yi, A multi-objective software quality classification model using genetic programming, IEEE Transactions on Reliability 56 (2) (2007) 237–245.

[27] E. M. Mugambi, A. Hunter, Multi-objective genetic programming optimization of decision trees for classifying medical data, in: 7^{th} International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES2003), Oxford, UK, 2003, pp. 293–299.

[28] M. P. Basgalupp, R. C. Barros, A. C. de Carvalho, A. A. Freitas, Evolving decision trees with beam search-based initialization and lexicographic multi-objective evaluation, Information Sciences 258 (2014) 160–181.

[29] S. Haruyama, Z. Qiangfu, Designing smaller decision trees using multiple objective optimization based GPs, in: IEEE International Conference on Systems, Man and Cybernetics, Vol. 6, Yasmine Hammamet, Tunisia, 2002, p. 5.

[30] N. R. Draper, H. Smith, Applied Regression Analysis, 3rd Edition, John Wiley & Sons, New York, 1998.

[31] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[32] R. Iman, J. Davenport, Approximations of the critical region of the Friedman statistic, Communications in Statistics-Theory and Methods 9 (6) (1980) 571–595.

[33] S. Holm, A simple sequentially rejective multiple test procedure., Scandinavian Journal of Statistics 6 (2) (1979) 65–70.

[34] K. Chellapilla, Evolving computer programs without subtree crossover, IEEE Transactions on Evolutionary Computation 1 (3) (1997) 209–216.

[35] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (10) (2010) 2044–2064.

[36] S. Salzberg, On comparing classifiers: Pitfalls to avoid and a recommended approach, Data Mining and Knowledge Discovery 1 (3) (1997) 317–328.

[37] Y. Cao, P. Rockett, A Novel Loss Function for Training Decision Trees, Tech. Rep. 2012/CR/001, Dept. of Electronic and Electrical Engineering, University of Sheffield, Sheffield, UK (2012).