Journal of Classification 34:85-107 (2017)
DOI: 10.1007/s00357-017-9223-0

Fusing Vantage Point Trees and Linear Discriminants for Fast Feature Classification

Hugo Proença

Instituto de Telecomunicações (IT), University of Beira Interior, Portugal

João C. Neves

University of Beira Interior, Portugal

Abstract: This paper describes a classification strategy that can be regarded as a more general form of nearest-neighbor classification. It fuses the concepts of *nearest neighbor*, *linear discriminant* and *Vantage-Point* trees, yielding an efficient indexing data structure and classification algorithm. In the learning phase, we define a set of disjoint subspaces of reduced complexity that can be separated by linear discriminants, ending up with an ensemble of simple (weak) classifiers that work locally. In classification, the closest centroids to the query determine the set of classifiers considered, which responses are weighted. The algorithm was experimentally validated in datasets widely used in the field, attaining error rates that are favorably comparable to the state-of-the-art classification techniques. Lastly, the proposed solution has a set of interesting properties for a broad range of applications: 1) it is deterministic; 2) it classifies in time approximately logarithmic with respect to the size of the learning set, being far more efficient than nearest neighbor classification in terms of computational cost; and 3) it keeps the generalization ability of simple models.

Keywords: Vantage-point tree; Linear discriminants; Nearest neighbor classification.

Corresponding Author's Address: H. Proença, Instituto de Telecomunicações (IT), University of Beira Interior, Tel.: +351 275 242 081, Fax: +351 275 319 899, email: hugomcp@di.ubi.pt.

Published online: 20 March 2017

1. Introduction

Supervised feature classification (or simply classification) is the assignment of a category (class) to an input value, on the basis of a learning set. For n pairs $(\vec{x}_i, y_i)$, $\vec{x}_i$ is represented in a d-dimensional space Ω and y_i its label (class), the goal is to find a function $f : \Omega^d \rightarrow \mathbb{N}$ that maps feature points into labels. In this context, an *ensemble* $f_e : \Omega^{d \times m} \rightarrow \mathbb{N}$ combines the output of multiple classifiers and seek to improve performance, when compared to individual elements. Ensemble classifiers are widely seen in the literature (e.g., Kuncheva, 2004 and several performance evaluation initiatives were conducted (e.g., Bauer and Kohavi, 1999; Banfield et al., 2007; Dietterich, 2000; Alpaydin, 1999; Demsar, 2006).

Under a computational perspective, the burden of classification is a primary concern for various domains (e.g., computer vision), due to extremely large amounts of data or to very demanding temporal constraints. Here, the turnaround time of classification is usually more concerning than the one of learning, as the latter is carried out off-line and a reduced number of times. As an example of demanding temporal requirements for classification, the defect detection in industrial environments can be referred, where a high number of frames per second must be processed (Kumar, 2008).

This paper proposes an efficient ensemble algorithm based on the concepts of *Vantage-Point* trees (Yianilos, 1993) and linear discriminant analysis, and is from now on designated as *Vantage-Point Classification* (VPC). The idea accords the philosophy of boosting and combines a set of base (weak) classifiers learned from feature subspaces and positioned in leaves of tree. The insight is that, regardless the complexity of the feature space, a linear discriminant is able to separate classes at a sufficiently deep level in the tree. Classification results from weighted voting of the base classifiers, selected according to the distance between the unlabelled sample and the centroid of each subspace. We come out with a solution that has two interesting properties: it attains classification accuracy similar to the state-of-the-art techniques in different problems, and it is efficient in terms of the computational cost of classification.

Hence, the major findings reported in this paper are: 1) for a broad range of the problems considered, VPC obtains better performance than state-of-the-art individual and ensemble models; 2) improvements in classification accuracy were observed along with a decrease in the computational cost of classification, when compared to related models (e.g., neural networks and K-nearest neighbors). Also, it should be stressed that experiments were carried out in datasets widely used in the classification domain (*UCI Machine Learning Repository, Univ. California*), that vary in terms of different criteria: binary/n-ary classification, discrete/continuous features, balanced/unbalanced prior probabilities and densely/sparsely populated feature spaces.

The remainder of this paper is organized as follows: Section 2 summarizes the most relevant ensemble classification methods. Section 3 provides a description of the VPC algorithm. Section 4 presents and discusses the results. Finally, the conclusions are given in Section 5.

2. Related Work

Sharing several properties with the method proposed in this paper, Ting et al. (2011) proposed the concept of *Feating*, a generic ensemble approach that is claimed to improve the predictive accuracy of both stable and unstable classification models. As in our case, their original concept is that *"a local model formed from instances similar to one we wish to classify will often be more accurate than a global model formed from all instances"* (Frank, Hall, and Pfahringer, 2003). The idea is to divide the feature space into a set of disjoint subspaces, according to user-specified features that control the level of localization. Their trees use, at a given level, the same attribute for feature subspace division; which does not happen in our case, and we claim to be a much more intuitive variant, i.e., the feature that best divides a subspace is not guaranteed to optimally divide another subspace, even if both spaces are represented at the same level of the tree. Another major difference is that in Feating, all models in the ensemble are used for every query (as in Bagging), while in VPC only the models that regard the closets subspaces to the query instances are used, in a weighted way. Comparing the results observed for our strategy and the results reported in Ting et al. (2011), a major advantage is the ability of our ensemble model to get error rates comparable to state-of-the-art classification algorithms, while keeping a relatively short ensemble size, i.e., without building classification trees that are impracticable for most situations.

According to Canuto et al. (2007), there are two major ensemble categories: 1) hybrid ensembles, that combine different types of algorithms; and 2) non-hybrid ensembles, where a single type of algorithm is replicated multiple times. The most popular schemes are non-hybrid and apply a base algorithm to permutated training sets. Among these, *Bagging* and *Boosting* are the most prominent strategies. Originally proposed by Breiman (1996), Bagging (Bootstrap aggregating) builds multiple models for a problem, each one based on a subset of the learning data. Then, voting combines the outputs, improving stability and accuracy when compared to base models. Kuncheva and Rodríguez (2007) replaced each classifier by a *mini-ensemble* of two classifiers and a random linear function (*oracle*). In classification, the oracle decides which classifier to use. The classifiers and oracle are trained together, so that each classifier is pushed into a different region of the feature space. Also, Hothorn and Lausen (2005) construct a set of classifiers for different bootstrap samples. Then, the predictions of such classifiers in each element of the bootstrap are used for a classification tree. This tree implicitly selects the most effective predictors, which authors consider to *bundle* the predictions for the bootstrap sample. Classification is done by averaging the predictions from a set of trees. In a related topic of tree-based regression models, Ceci, Appice and Malerba (2003) used two basic operations (pruning and grafting) to obtain simpler structures for the regression tree. The core of these simplification operations is to put aside some data for independent pruning, which was observed in practice to improve classification performance.

The idea of Boosting resulted from the stochastic discrimination theory (Kleinberg, 1990), a branch that studies the ways to divide the feature space for class discrimination. This algorithm combines several weak classifiers, each one with high bias and low variance. Experiments point out that is possible to meet high accuracy far before using all the weak classifiers. A relevant boosting method was due to Shapire (1990), but the most well-known is the Adaboost variant (Freund and Schipire, 1995), that increases adaptability by tweaking subsequent classifiers in favor of instances misclassified by previous ones.

Ho (1995) suggested the notion of *Random Forest*, from where a generalization was proposed (*Random Subspace:* Ho, 1998), later known as *Attribute Bagging* (Bryll, Gutierrez-Osuna and Quek, 2003). The idea is to build an ensemble of classifiers, each one using a subset of the available features. In classification, voting produces the final answer. In this context, Zaman and Hirose (2013) enlarged the feature space of the base tree classifiers in a random forest, by adding features extracted from additional predictive models, having empirically concluded that such *hybrid* random forests can be a more efficient tool than the traditional forests for several classification tasks.

Domingos (1996) described an algorithm based on rule induction and instance-based learning, considering individual instances as maximally specific rules, and then devising an algorithm to gradually fuse instances into more general rules. The proposed algorithm was considered an inductive learning approach that produces specialized rules that span the entire feature space, by searching for the best mixture of instances and increasingly augment abstraction of rules, yielding a more general-form of nearest neighbor classification.

Particularly interested in problems with a reduced amount of learning data, Lu and Tan (2011) sought for a subspace that minimizes the within-class to between-class distances ratio. To enlarge the amount of learning data, they used a linear model to interpolate pairs of prototypes, simulating variants of the available samples. Bock, Coussement and Poel (2010) used generalized additive models as base classifiers for ensemble learning, proposing variants based on bagging and random subspaces. Classification yields from average aggregation.

In a related topic, instance selection aims at obtain a subset of the learning data, so that models on each subset have similar performance to the attained in the complete set. García-Pedrajas (2009) used instance selection in boosting processes, optimizing the training error by the weighted distribution of instances erroneously classified by previous models. Yu et al. (2012) divided the feature space into disjoint subspaces. Then, defined a neighborhood graph in each subspace and trained a linear classifier on this graph, used as base classifier of the ensemble. In classification, the majority-voting rule is used. Starting from models learned by random space and bootstrap data samples, Yan and Tešić (2007) estimated the decision boundaries for each class, concluding that a few shared subspace models are able to support the entire feature space. This scheme is claimed to reduce redundancy while enjoying the advantages of being built from simple base models.

Considering that—typically—the performance of classifier ensembles is maximized in case of Uniform distributions of observations, Jirina and Jirina Jr. (2013) suggested a transformation on the data space that approximates the distribution of observations in the feature space into a uniform distribution, at least in the neighborhood of a query observation. Their transformation is based on a scaling exponent that relates distances between pairs of points in a multivariate space.

As described in the next section, the VPC model shares some of the above referred foundations: similar to the concept behind boosting, we divide the feature space into subspaces, pushing each base classier into disjoint regions of the feature space. Similar to Lu and Tan (2011), the within-to-between class distances proportion is used to determine the number of divisions of the feature space. Then, by preserving neighborhoods between subspaces, for a given query we are able to select a subset of the base-classifiers in a computationally effective way.

3. Vantage Point Classification

Figure 1 illustrates the key idea behind the VPC scheme: the feature space is divided into subspaces Ω_i, according to the distance of elements to pivots p_i. Each subspace (leaf) is simple so that a linear discriminant Φ_i separates classes with a reduced expected error. In classification, for a query $\vec{x}$, only the closest subset of the leaves vote, according to the distances between $\vec{x}$ and p_i. This schema creates a set of spaces Ω_i where classification is done locally.

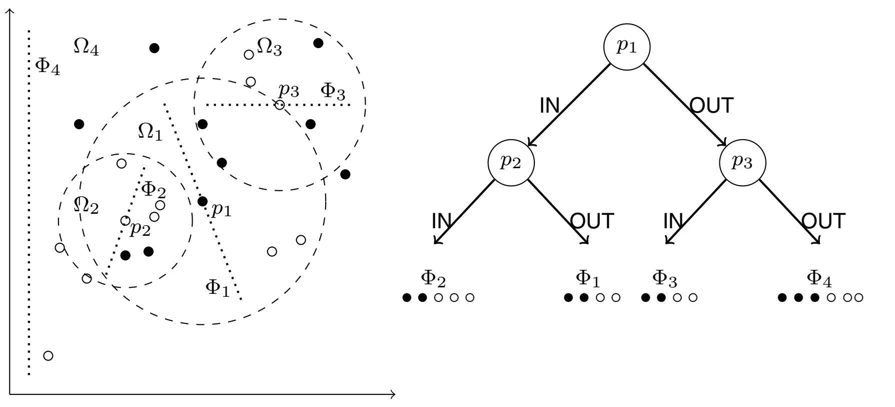

Figure 1. Illustration of the idea behind VPC. At left, the dotted line segments denote the projections Φ_i found for compact spaces Ω_i and the dashed circles denote the median distance between pivots p_i and elements on that space. This value is used to separate elements at each side of the vantage point tree. The right figure gives the VPC data structure corresponding to this feature space.

3.1 Learning

The learning process starts by evaluating if a linear projection Φ separates the feature space with a misclassification rate lower than γ. Let Ω be a d-dimensional feature space containing n instances $\vec{x}_i$ with labels y_i. According to Johnson and Wichern (1988), multiple discriminant analysis is a natural extension of the Fisher linear discriminant, having the within-class matrix given by:

$$\hat{\Sigma_w} = \sum_{c=1}^{k} \sum_{\vec{x}_i|y_c} (\vec{x}_i - \bar{\vec{x}}_{y_c})(\vec{x}_i - \bar{\vec{x}}_{y_c})^T, \tag{1}$$

where k is the number of classes, $\vec{x}_i|y_c$ denotes the elements in the cth class and $\bar{\vec{x}}_{y_c}$ is the centroid of these elements. The scatter matrix is given by:

$$\hat{\Sigma_s} = \sum_{c=1}^{k} n_{y_c} (\bar{\vec{x}} - \bar{\vec{x}}_{y_c})(\bar{\vec{x}} - \bar{\vec{x}}_{y_c})^T, \tag{2}$$

being n_{y_c} the number of training samples in class y_c. $\bar{\vec{x}}$ is the dataset mean vector. A linear transformation, Φ, is obtained by solving the generalized eigenvalue system:

$$\hat{\Sigma_s}\Phi = \lambda \hat{\Sigma_w}\Phi, \tag{3}$$

where λ is a scalar that is usually called the generalized eigenvalue of Σ_s and

Σ_w. Classification is done in the transformed space according to a distance function $\xi(.,.)$. Each query element $\vec{x}$ is classified by:

$$\hat{k_i} = \arg\min_k \xi(\vec{x}\Phi, \bar{x}_{y_k}\Phi). \tag{4}$$

Our stopping criterion for the division of Ω considers the error rate in the learning set, given by:

$$e(\Omega) = \frac{\sum_i \mathbb{I}_{\{k_i \neq y_c\}}}{n}, \tag{5}$$

where $\mathbb{I}$ is an indictor function that evaluates if the predicted and true class values are the same. When $e(\Omega) \leq \gamma$, we consider that Φ appropriately discriminates the feature space and the node is considered a leaf, with support $s(\Omega) = n$. Otherwise, Ω is divided into two halves, according to a pivot.

Note that the term *appropriately*, in terms of discrimination, is used to indicate subspaces where a linear discriminant attain classification error lower than γ. For $\gamma = 0$, the term is equivalent to linearly separable. In practice, for most cases the optimal performance is attained when $\gamma > 0$, i.e., stopping the division of subspaces when the number of elements per class is still much higher than the dimension of the feature space (otherwise, the algorithm would simply return the pseudo-inverse of the Fisher linear Discriminant at a leaf). $\gamma > 0$ values are regarded as a soft margins, i.e., we allow a few mistakes (some points - outliers or noisy examples might be on the wrong side of the linear discriminant), but most times obtain a solution that better separates the bulk of data.

If the ith element in Ω is used as pivot, the remaining elements with indexes $j \in \{1, \ldots, n\}, j \neq i$, span through its left or right branch, depending of the distance $\xi(\vec{x}_i, \vec{x}_j)$. Let Y_{ic}^0 and Y_{ic}^1 be the sets of labels of class c in the left and right branches, when using the ith element as pivot. The *suitability* of that pivot $s(i)$ is equal the support of the corresponding discriminant:

$$s(i) = -\sum_{j=0}^{1}\sum_{c=1}^{k} \frac{|Y_{ic}^{(j)}|}{n-1}\log_2\Big(\frac{|Y_{ic}^{(j)}|}{n-1}\Big), \tag{6}$$

where $|.|$ denotes set cardinality. The best pivot minimizes (6), i.e., is the one that puts all elements of each class in different branches of the tree:

$$\hat{i} = \arg\min_i s(i). \tag{7}$$

Let $d_{\hat{i}j} = \xi(\vec{x}_j, \vec{x}_{\hat{i}}), i = 1, \ldots, n, i \neq \hat{i}$ be the $n{-}1$ distances between the pivot and the remaining elements and let $d_{\hat{i}}^*$ be the median value of $\{d_{\hat{i}j}\}$. The jth element of the training set is included in the left Ω^L or right Ω^R

subsets, according to:

$$\begin{cases} \{\vec{x}_j, y_j\} \in \Omega^L & \text{if } d_{\hat{i}j} \geq d^*_{\hat{i}} \\ \{\vec{x}_j, y_j\} \in \Omega^R & \text{if } d_{\hat{i}j} \leq d^*_{\hat{i}}. \end{cases} \tag{8}$$

The process is repeated for Ω^L and Ω^R in a way similar to Ω, until the stopping criterion is verified for all the subspaces. In practice, the optimal performance of the VPC model is attained when $\gamma > 0$, i.e., stopping the learning process before having all elements of a single class, which contributes to avoid overfitting.

3.2 Classification

Classification is done by traversing the VPC tree and accumulating the support values of the class predicted at each leaf. Let $\vec{x}$ be an unlabelled element and γ^* the radius of the query. The classification of $\vec{x}$ is given by (9), where $l(\Omega)$ is an indicator function that discriminates between leaves ($l(\Omega) = 1$) and non-leaves ($l(\Omega) = 0$) nodes:

$$c(\vec{x}, \Omega) = \begin{cases} \vec{0} & \text{, if } l(\Omega) = 1 \ \wedge \xi(\vec{x}, \vec{x}_p) > \gamma^* \\ s(\Omega).\vec{v}_i & \text{, if } l(\Omega) = 1 \ \wedge \xi(\vec{x}, \vec{x}_p) \leq \gamma^* \\ \{s(\Omega^L) + s(\Omega^R)\} & \text{, if } l(\Omega) = 0 \ \wedge (d^* - \gamma^*) \leq \xi(\vec{x}, \vec{x}_p) \\ & \quad \leq (d^* + \gamma^*) \\ s(\Omega^L) & \text{, if } l(\Omega) = 0 \ \wedge \xi(\vec{x}, \vec{x}_p) \leq (d^* - \gamma^*) \\ s(\Omega^R) & \text{, if } l(\Omega) = 0 \ \wedge \xi(\vec{x}, \vec{x}_p) \geq (d^* + \gamma^*) \end{cases}, \tag{9}$$

being $\vec{x}_p$ the pivot of a node and $\vec{v_i}$ a unit vector with a single non-zero component at the ith position (corresponding to the ith predicted class). This way, $c(\vec{x}, \Omega)$ returns a vector with k elements, each one containing the accumulated support for the predicted class, i.e., $\vec{s} = \{s_1, \ldots, s_k\}$. The response given by the ensemble corresponds to the position where $c(\vec{x}, \Omega)$ is maximum:

$$\hat{i} = \arg\max_i \{s_i\}. \tag{10}$$

For comprehensibility, Algorithm 1 details the VPC classification scheme in terms of the computational steps. As input, the method receives the Vantage Point tree with a linear discriminant in each leaf. For a query element $\vec{x}$, the binary tree is traversed down to leaves. In each leaf, a linear discriminant predicts the class and the accumulation of the support values gives the final response.

Algorithm 1 VPC Classification Scheme

```
Require: Vantage Point Tree $\boldsymbol{V}$, unlabelled instance $\vec{x} \in \mathbb{R}^d$, radius query $\gamma^*$.
 1: Get current node: $\boldsymbol{c} \leftarrow \boldsymbol{V}$.root;
 2: Support values for each class: $\vec{s} \leftarrow [0, \ldots, 0]$
 3: if leaf($\boldsymbol{c}$) then
 4:     Accumulate support: $\vec{s} \leftarrow LDA(\boldsymbol{c}, \boldsymbol{x})$
 5:     return $\vec{s}$
 6: end if
 7: Get pivot: $\boldsymbol{p} \leftarrow \boldsymbol{c}$.pivot
 8: Get median distance: $m \leftarrow \boldsymbol{c}$.median
 9: if distance($\vec{x}, \boldsymbol{p}$) $\leq \boldsymbol{m} - \gamma$ then
10:     Accumulate support: $\vec{s} \leftarrow \vec{s}$ + VPC($\boldsymbol{cn}$.left, $\vec{x}$, $\gamma^*$)
11: end if
12: if distance($\vec{x}, \boldsymbol{p}$) $\geq \boldsymbol{m} + \gamma$ then
13:     Accumulate support: $\vec{s} \leftarrow \vec{s}$ + VPC($\boldsymbol{cn}$.right, $\vec{x}$, $\gamma^*$)
14: end if
15: return $\vec{s}$
```

3.3 Usability And Completeness

According to the theory of classification ensembles, it is particularly important that the ensemble is fully *usable* and *complete*. Let Ω be the d-dimensional feature space with Ω_i compact subspaces. The usability of the projection Φ_i was approximated by:

$$P\big(\vec{x} \in \Omega_i\big) \approx \frac{\sum_j \mathbb{I}_{\{(x_j^{(1)}, \ldots, x_j^{(d)}) \in \Omega_i\}}}{n}, \tag{11}$$

where $\mathbb{I}_{\{.\}}$ is the indicator function and n is the number of learning instances. Based on the stopping criterion used for learning, $P\big(\vec{x} \in \Omega_i\big) > 0,\ \forall i \in \{1, \ldots, k\}$, guaranteeing that every Φ_i is usable and that the ensemble is fully usable.

As described in Section 3.1, it can be stated that:

$$\Omega^d = \bigcup_i \Omega_i^d, \tag{12}$$

assuring that the ensemble completely covers the feature space (provided that $n \geq d$). Similarly, as:

$$\Omega_j^d \cap \Omega_i^d = \emptyset, \forall i, j \mid i \neq j, \tag{13}$$

the domains of each discriminant Φ_i are disjoint and the full diversity of the ensemble is also assured, i.e., every discriminant Φ_i is obtained from completely disjoint data with respect to the remaining discriminants, which reduces the probability of obtaining correlated models. The fact of using disjoint data for building each discriminant might augment the probability of overfitting. As a counterbalance, we note that several nodes usually vote for a query, which reduces the probability of overfitting.

3.4 Computational Complexity

Here we analyze the time complexity, which can refer to the *learning* or *classification* phases. But, as the latter phase is done on-line and requires repeated execution, our efforts were concentrated in keeping low the complexity of the classification phase.

The learning phase has two major steps: 1) create the VPC tree; and 2) obtain a linear discriminant for each leaf node. Let n be the number of learning instances. Determining one pivot per node takes $O(n^2)$. The insertion always takes place at the deepest level, with complexity $O(h)$, being h the height of the tree (if the tree is balanced, $h = lg(n)$). Next, learning a linear discriminant for each leaf involves three major steps: 1) singular value decomposition to obtain the within and scatter matrices; 2) compute eigenvectors; and 3) solve the final linear system, which has $O(n\, d \min(n,d) + \min(n,d)^3)$ (Cai, He and Han, 2008) temporal complexity, being d the dimension of the feature space.

In classification, the complexity depends of the radius γ^*, varying between $O(n)$ (when all leaves vote) and $O(lg(n))$ (when a single leaf votes). Our experiments confirm that optimal performance is attained when a reduced number of nodes vote for the ensemble, yielding a temporal complexity around $O(\alpha\ lg(n))$, being α the number of nodes voting ($1 \leq \alpha \leq n$). At each leaf, the temporal complexity of classification is $O(d(k-1)X)$, being k the number of classes. Hence, the time complexity of classification is $O(\alpha d(k-1)) + O(\alpha lg(n))$. Keeping moderate values for k and d, the predominant term is clearly $O(\alpha lg(n))$. Keeping in mind that usually each node of the tree represents more than one training instance (due to the parameter $\gamma > 0$), it follows that $\alpha \ll n$, reducing the complexity to approximately logarithmic.

3.5 Bias-Variance Tradeoff

In VPC, the tradeoff between bias and variance depends of the number of classifiers in the ensemble and of the number of votes per query. The former is determined by γ and the latter by γ^*. Both γ_I and γ_q are in direct proportion to bias, and inversely correlated to variance. As illustrated in Figure 2, high values for γ and γ^* reduce the number of leaves, from where a large proportion is used in classification. At the other extreme, as the values of γ and γ^* decrease, more leaves are created, and smaller proportions of

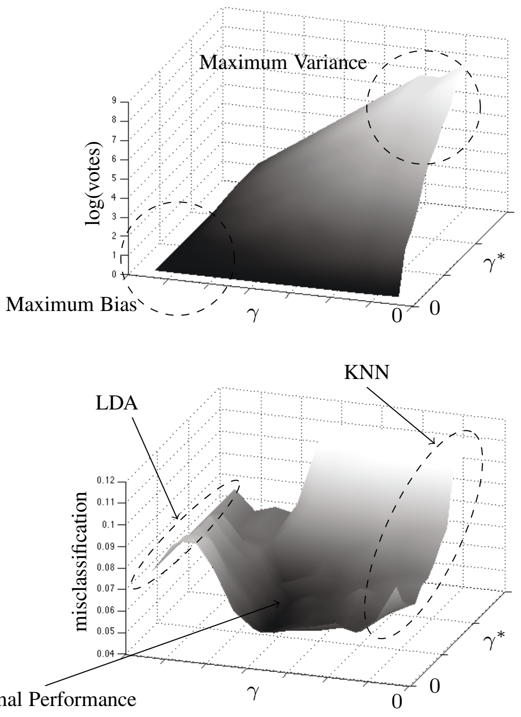

Figure 2. Top plot: effect of the γ and γ^* parameters in the number of classifiers that vote in each query. Bottom plot: corresponding misclassification rates, with the optimal configuration highlighted. The configurations that are equivalent to linear discriminant analysis and k-nearest neighbors appear inside dashed ellipses.

these are used in each query. This corresponds to getting purer—and less biased—estimates. However, a tree with more leaves reduces the sample size per classifier and increases the potential estimation error, increasing the variance of the model.

It is interesting to note that, under specific parameterizations, VPC is equivalent to linear discriminant analysis (LDA) or to k-nearest neighbors (KNN) (Figure 2). It is equivalent to LDA in cases where the tree is composed by a single node (large γ values). Oppositely, for $\gamma = 0$, each leaf of the tree offers perfect separability between classes, which in practice reduces to the nearest neighbor rule. However, as this example illustrates ("Optimal Performance" arrow), one of the key findings reported in this paper is that optimal performance is most times attained for intermediate values of γ and γ^*.

4. Experiments and Discussion

4.1 Comparison Terms

Eight well known classification techniques were used as comparison terms. Four individual models (k-nearest neighbors: Cover and Hart, 1967; linear discriminant analysis: Duda, Hart and Stork, 2000; neural networks: Moller, 1993; and support vector machines: Cortes and Vapnik, 1995) and four ensembles: Bagging with classification trees (CART), quadratic and pseudo-linear weak classifiers, Boosting and Random Spaces with quadratic discriminants and decision tree weak classifiers and the Random Forest (Breiman, 2001) with decision trees as weak classifiers.

The selected algorithms were considered to represent the state-of-the-art in terms of classification. Even though several variants exist, the ones used are the most frequently reported in the literature and in previous performance evaluation initiatives (e.g., Bauer and Kohavi, 1999; Dietterich, 2000; and Demsar, 2006). This way, our idea is that by transitivity, it is possible to compare the performance of VPC against any other classification model that shared any comparison term used in this paper.

Table 1 describes the parameters tested in the optimization of each algorithm. A set of *parameterizations* was tested, by an exhaustive grid combination of parameters in the given range. Additionally, for non-deterministic methods, each parameterization was tested 10 times and the best performance taken. Regarding the VPC method, the Euclidean distance ($\ell 2$-norm) was used in all cases as $\xi(.,.)$ function. All the results correspond to implementations in the MATLAB environment.

4.2 Synthetic Datasets

Performance started to be analyzed on synthetic bi-dimensional datasets (Figure 3). All regard binary classification problems, ranging from linearly separable (Problem A) to complex decision environments: with continuous/discontinuous boundaries (problems B and C) and balanced/unbalanced prior probabilities (problems D and E). VPC denotes the proposed method, LDA the linear discriminant analysis, NN stands for neural networks and KNN for k-nearest neighbor classification. For each problem/algorithm, the decision boundaries appear in black. An immediate conclusion is the suitability of VPC to handle all classes of problems tested, both linearly separable (problem A), with complex decision boundaries (problems C-E) and different levels of prior probabilities per class (problem E). In these experiments, the smoothness of the decision boundaries of VPC was lower than for NN and KNN, which was explained by the parameters used (low values of γ and γ^* were used in this example).

Table 1. Variants of each classification algorithm evaluated and corresponding parameters/intervals evaluated in the optimization process.

Algorithm	**Parameters Optimization**
LDA	-
Neural Networks (NN)	Learning Algorithm: Levenberg-Marquardt backpropagation, scaled-conjugate gradient, gradient descend with adaptive learning rate/momentum; Topology: neurons hidden layer; Learning stopping criteria: validation checks, performance, and epochs.
KNN	Number neighbors: [1, d]
SVM	Kernel type: linear, polynomial and sigmoid; kernel degree: [1, 4]; γ kernel functions: [0.5/d, 2*d]
Bagging (BAG)	Number ensemble classifiers: [2, 2*d], Weak learners: classification trees (CART), quadratic/pseudo-linear
Boosting (BOS)	Number ensemble classifiers: [2, 2*d]; Weak learners: quadratic discriminants and decision trees; Type learning algorithm: multi-class AdaBoost, RUSBoost (Seiffert et al., 2008)
Random Subspaces (RSP)	Number ensemble classifiers: [2, 2*d]; Weak learners: quadratic discriminants and decision trees, Gini's diversity index/maximum deviance split criteria.
Random Forests (RFO)	Number weak classifiers: [2, 2*d]; Number of variables selected per split: [1, $\sqrt{d}$].

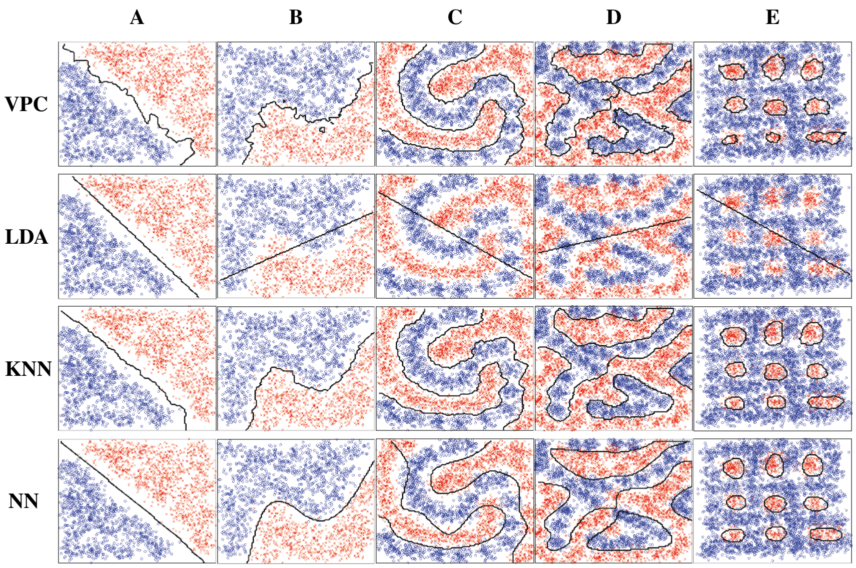

Figure 3. Results attained by the VPC for bi-dimensional synthetic datasets, when compared to linear discriminant analysis (LDA), neural networks (NN) and k-nearest neighbors (KNN) classification methods.

Table 2. Datasets of the *UCI Machine Learning Repository* (Univ. California) used in the performance evaluation of VPC.

ID	**Data Set**	**Instances (Training/Test)**	**Features**	**Classes**	**Prior Probs (%).**
BC	**Breast Cancer Wiscosin (Original)**	683 (10 × 615/68)	9	2	65.00, 35.00
HS	**Haberman's Survival**	306 (10 × 276/30)	3	2	74.00, 26.00
IS	**Image Segment**	2310 (10 × 2079/231)	19	7	(balanced)
IR	**Iris**	150 (10 × 135/15)	4	3	(balanced)
IT	**Isolet**	7797 (10 × 7 018/779)	617	26	(balanced)
LR	**Letter Recognition**	20 000 (10 × 18 000/2 000)	16	26	3.94,3.83,3.68,4.03,3.84,3.87,3.86, 3.67,3.77,3.64,3.69,3.81,3.96,3.91, 3.77,4.01,3.91,3.79,3.74,3.98,4.06, 3.82,3.76,3.94,3.93,3.67
MF	**Multifeature Digit**	2 000 (10 × 1 800/200)	649	10	(balanced)
MU	**Musk (Version 2)**	6 598 (10 × 5 939/659)	168	2	84.59,15.41
PB	**Page Blocks**	5 473 (10 × 4 926/547)	10	5	89.77,6.01,0.51,1.61,2.10
SK	**Skin Segmentation**	245 057 (10 × 220 552/24 505)	3	2	20.75,79.25
SP	**Spambase**	4 601 (10 × 4 141/460)	57	2	60.60,39.40
ST	**Statlog (Shuttle)**	58 000 (10 × 52 200/5 800)	9	7	78.60,0.08,0.29,15.35,5.63,0.01, 0.02

4.3 UCI - Machine Learning Repository

Performance was also compared in the *University of California, Irvine: Machine Learning Repository*[1] datasets, which are freely available and widely known in the field of classification. The used sets are summarized in Table 2, and were selected according to four criteria: 1) containing multivariate or univariate features; 2) suitable for classification tasks; 3) exclusively with numeric attributes; and 4) without missing values. We give the number of instances, features, classes and prior probabilities per class. Values inside parenthesis denote the amounts of data selected for learning/testing purposes. For all sets, 10-fold cross validation was adopted and features were rescaled to [0,1] interval according to the *min-max* rule.

1. http://archive.ics.uci.edu/ml/

Table 3. Results obtained for the datasets of the *UCI Machine Learning Repository*. The mean errors are given, together with the standard deviations observed in the 10-fold cross validation procedure. Cells in bold highlight the best algorithm per data set.

Dataset	**VPC**	**LDA**	**NN**	**SVM**	**KNN**	**BAG**	**BOS**	**RSP**	**RFO**
BC	**2.48** $\pm$ **1.83**	3.68 $\pm$ 2.71	3.38 $\pm$ 2.78	2.94 $\pm$ 2.19	2.50 $\pm$ 1.71	3.68 $\pm$ 2.79	2.97 $\pm$ 2.06	6.62 $\pm$ 2.88	2.91 $\pm$ 1.95
HS	20.33 $\pm$ 5.54	28.00 $\pm$ 6.52	24.00 $\pm$ 4.66	25.33 $\pm$ 7.57	23.33 $\pm$ 4.16	23.67 $\pm$ 4.57	**16.70** $\pm$ **4.98**	25.67 $\pm$ 6.49	23.06 $\pm$ 6.06
IS	3.58 $\pm$ 1.07	11.26 $\pm$ 1.56	5.37 $\pm$ 5.29	7.01 $\pm$ 1.73	**3.12** $\pm$ **1.29**	8.18 $\pm$ 1.63	5.17 $\pm$ 2.03	21.17 $\pm$ 2.13	3.58 $\pm$ 1.52
IR	2.00 $\pm$ 1.22	4.00 $\pm$ 4.66	4.67 $\pm$ 6.32	4.00 $\pm$ 5.62	3.33 $\pm$ 4.71	**1.33** $\pm$ **2.81**	2.67 $\pm$ 3.44	4.00 $\pm$ 4.66	3.69 $\pm$ 1.02
IT	5.73 $\pm$ 0.72	5.73 $\pm$ 0.72	6.78 $\pm$ 6.89	**3.07** $\pm$ **0.84**	8.97 $\pm$ 1.27	5.35 $\pm$ 0.83	12.91 $\pm$ 2.28	30.90 $\pm$ 1.79	11.03 $\pm$ 2.02
LR	9.01 $\pm$ 0.63	37.49 $\pm$ 1.09	26.20 $\pm$ 5.17	17.65 $\pm$ 0.66	**3.89** $\pm$ **0.29**	11.21 $\pm$ 0.64	11.30 $\pm$ 0.69	49.03 $\pm$ 0.72	12.47 $\pm$ 1.28
MF	1.75 $\pm$ 0.67	1.75 $\pm$ 0.68	2.35 $\pm$ 2.96	1.55 $\pm$ 0.83	1.65 $\pm$ 0.58	**1.05** $\pm$ **0.69**	7.91 $\pm$ 0.97	9.25 $\pm$ 2.12	2.28 $\pm$ 0.80
MU	2.40 $\pm$ 0.38	6.43 $\pm$ 0.97	**0.93** $\pm$ **0.45**	5.13 $\pm$ 0.76	3.11 $\pm$ 0.37	3.25 $\pm$ 0.68	3.35 $\pm$ 0.60	11.90 $\pm$ 1.55	2.57 $\pm$ 0.94
PB	**3.32** $\pm$ **0.70**	7.35 $\pm$ 1.48	3.33 $\pm$ 0.77	7.11 $\pm$ 1.20	4.04 $\pm$ 0.97	5.60 $\pm$ 1.10	4.52 $\pm$ 1.45	7.77 $\pm$ 0.73	3.48 $\pm$ 0.53
SK	0.06 $\pm$ 0.01	6.60 $\pm$ 0.14	0.31 $\pm$ 0.53	1.02 $\pm$ 0.06	**0.04** $\pm$ **0.01**	1.64 $\pm$ 0.08	1.64 $\pm$ 0.08	17.45 $\pm$ 0.27	1.70 $\pm$ 0.22
SP	6.34 $\pm$ 1.31	9.48 $\pm$ 2.02	6.17 $\pm$ 1.74	9.57 $\pm$ 1.82	8.85 $\pm$ 1.03	10.80 $\pm$ 1.91	6.13 $\pm$ 1.02	34.28 $\pm$ 3.71	**4.36** $\pm$ **0.80**
ST	0.05 $\pm$ 0.02	17.48 $\pm$ 0.82	0.42 $\pm$ 0.02	3.08 $\pm$ 0.22	0.06 $\pm$ 0.03	5.52 $\pm$ 0.29	1.59 $\pm$ 0.32	14.66 $\pm$ 1.21	**0.03** $\pm$ **0.00**

Note that the used sets are heterogenous from different perspectives, ranging from *easy* problems (such as SK and ST), to extremely *hard* (such as the HS), due to low feature-to-instance ratio and classes overlapping. Also, for some problems large amounts of data are available (e.g., SK), while others have a reduced number of instances available (e.g., IR).

Results are given in Table 3 and a first evidence is that VPC only got the *best* performance among all algorithms in two different problems (BC and PB). However, the most important observation is that, for all the remaining cases, VPC was among the best half of the algorithms. Also—as expected—KNN got the best results in problems with low feature-to-instance ratio, that correspond to densely populated feature spaces (e.g., SK).

Not only VPC, but also NN, SVM, KNN, Bagging and Random Forest got the 1^{st} rank in some problem. Among the ensemble algorithms, Bagging, Boosting and Random Forest outperformed all the remaining algorithm in some problem. In opposition, random subspaces got particularly hazardous results in low dimensionality datasets, where the projection into feature subspaces does not keep enough discriminating information.

To perceive the classes of problems where each algorithm got the best results, their relative effectiveness was tested. Differences in performance were validated in terms of statistical significance using Student t-tests (at the 95% level), assuming that errors are normally distributed. Having performance scores of two algorithms (vectors $\vec{v}_1$ and $\vec{v}_2$, length 10), a t-test t_e was carried out, stating as null hypothesis H_0 that "$\vec{v}_1$ *and* $\vec{v}_2$ *are independent random samples from normal distributions with equal means and unknown variances*". The alternative hypothesis was that means are different:

$$t_e = \frac{\text{abs}(\mu_{\vec{v}_1} - \mu_{\vec{v}_2})}{\sqrt{\frac{\sigma^2_{\vec{v}_1} + \sigma^2_{\vec{v}_2}}{10}}}, \tag{14}$$

Table 4. Summary of the *algorithm-to-algorithm* relative performance, for datasets of the *UCI Machine Learning Repository* (represented in each cell in the same order as in Table 2). The symbol "○" denotes that the algorithm in the column is *better* than the algorithm in the row, with statistical significance at 95% level. "●" denotes *worse* performance and "·" corresponds to differences in results without statistical significance. Each cell in the bottom row summarizes the total of "○", "·" and "●" cases for an algorithm.

Alg.	VPC	LDA	NN	SVM	KNN	BAG	BOS	RSP	RFO
VPC	-	· · ● · · ● · ● ● ● · ●	· · · · · ● · ○ · · · ●	· · ● · ○ ● · ● ● ● · ●	· · · · ● ○ · ● · · ● ·	· · ● · · ● · ● ● ● ● ●	· · · · ● ● ● ● · ● · ●	· · ● · ● ● ● ● ● ● ● ●	· · · ● ● ● · · · ● ○ ○
LDA		-	· · · · · ○ · ○ ○ ○ · ○	· · ○ · ○ ○ · · · ○ · ○	· · ○ · ● ○ · ○ ○ ○ · ○	· · · · · ○ · ○ · ○ · ○	· ○ ○ · ● ○ ● ○ · ○ ○ ○	· · ● · ● ● ● ● · ● ● ○	· · ○ · ○ ○ · ○ ○ ○ ○ ○
NN	· · · · · ○ · ● · · · ○	· · · · · ● · ● ● ● · ●	-	· · · · · ○ · ● ● · · ●	· · · · · ○ · ● · · · ○	· · · · · ○ · ● ● ● ● ●	· ○ ● ● · ○ ● ● · ● · ●	· · ● · ● ● ● ● ● ● ● ●	· · · · ● ○ · ● · ● ● ○
SVM	· · ○ · ● ○ · ○ ○ ○ · ○	· · ● · ● ● · · · ● · ●	· · · · · ● · ○ ○ · · ○	-	· · ○ · ● ○ · ○ ○ ○ · ○	· · · · ● ○ · ○ · ● · ●	· · · · ● ○ ● ○ · ● · ·	· · ● · ● ● ● ● · ● ● ●	· · ○ · ● ○ · ○ ○ ○ ○ ○
KNN	· · · · ○ ● · ○ · · ○ ·	· · ● · ○ ● · ● ● ● · ●	· · · · · ● · ○ · · · ●	· · ● · ○ ● · ● ● ● · ●	-	· · ● · ○ ● · · · ● · ●	· · · · ○ ○ ● · · ● · ●	· · ● · ● ● ● ● ● ● ● ●	· · · · · ● · · · ● ○ ○
BAG	· · ○ · · ○ · ○ ○ ○ ○ ○	· · · · · ● · ● · ● · ●	· · · · · ● · ○ ○ ○ ○ ○	· · · · ○ ● · ● · ○ · ○	· · ○ · ● ○ · · · ○ · ○	-	· · · · ● · ● · · · ○ ○	· · ● · ● ● ● ● ● ● ● ●	· · ○ · ● · · · ○ · ○ ○
BOO	· · · · ○ ○ ○ ○ · ○ · ○	· · ○ · ○ ● ○ ● · ● · ○	· · ○ · ○ ● ○ ○ · ○ ○ ○	· · ○ · ○ ● ○ ● · ○ · ○	· · ○ · ○ ○ ○ · · ○ · ○	· · ○ · ○ · ○ · · · · ○	-	· · · · ○ ● ○ ● · ● ● ○	· ● · · · · ○ · · · · ○
RSP	· · ○ · ○ ○ ○ ○ ○ ○ ○ ○	· · ○ · ○ ○ ○ ○ · ○ ○ ●	· · ○ · ○ ○ ○ ○ ○ ○ ○ ○	· · ○ · ○ ○ ○ ○ · ○ ○ ○	· · ○ · ○ ○ ○ ○ ○ ○ ○ ○	· · ○ · ○ ○ ○ ○ ○ ○ ○ ○	· · ○ · ○ ○ · ○ ○ ○ ○ ●	-	○ · ○ · ○ ○ ○ ○ ○ ○ ○ ○
RFO	· · · ○ ○ ○ · · · ○ ● ●	· · ● · ● ● · ● ● ● ● ●	· · · · ○ ● · ○ · ○ ○ ●	· · ● · ○ ● · ● ● ● ● ●	· · · · · ○ · · · ○ ● ●	· · ● · ○ · · · ● · ● ●	· ○ · · · · ● · · · · ●	● · ● · ● ● ● ● ● ● ● ●	-
Total	43, 48, 5	12, 46, 38	35, 52, 9	25, 45, 26	36, 51, 9	22, 51, 23	23, 48, 25	4, 27, 65	37, 46, 13

where abs(.) denotes the absolute value, $\mu_{\vec{v}_1}$ and $\mu_{\vec{v}_2}$ are the means and $\sigma_{\vec{v}_1}$ and $\sigma_{\vec{v}_2}$ the standard deviations. Every time $t_e > 2.10$, H_0 was rejected and assumed that both algorithms actually have different performance.

Table 4 summarizes the *algorithm-to-algorithm* comparison in the datasets evaluated (in the same order in each cell as in the rows of Table 2). Symbol "○" denotes cases where the algorithm in the column got better results (with statistical significance) than the one in the row. Oppositely, symbol "●" denotes worse performance and "·" denotes results without statistical significance. As main conclusion, the best performance of VPC among all is evident: only for three problems VPC got worse results than any other algorithm: *Musk* (worse than neural networks), *Isolet* (worse than support vector machines) and *Letter Recognition* (worse than k-nearest neighbors). The bottom row of the table gives the summary statistics, showing the number of "○", "·" and "●" cases. VPC, NN, SVM, KNN and Random Forests got overall positive balance, meaning that they were *better* more times than *worse*. VPC attained maximal balance ("○" - "●") value (38), followed by KNN (27) and NN (26) algorithms. Interestingly, the performance attained by two of the ensemble strategies (Boosting and Random Subspaces) was poorer than the observed for individual algorithms. When compared to LDA, in no case did VPC get worse performance, and in the IT dataset results were exactly equal, corresponding to a case where the VPC learning process stopped at the tree root (yielding the LDA).

Regarding the ensemble algorithms, Random Forests got the best results, followed by Bagging, in accordance to the results given by Banfield et al. (2007). Boosting got smaller errors than Bagging in half of the problems. The Random Forest algorithm outperformed all the remaining in two problems (*Spambase* and *Statlog*) that share the property of class imbalance. Random Subspaces got especially bad results in low dimensionality problems (e.g., Skin) and Boosting was among the best algorithms for problems with a reduced number of classes (e.g., *Musk*).

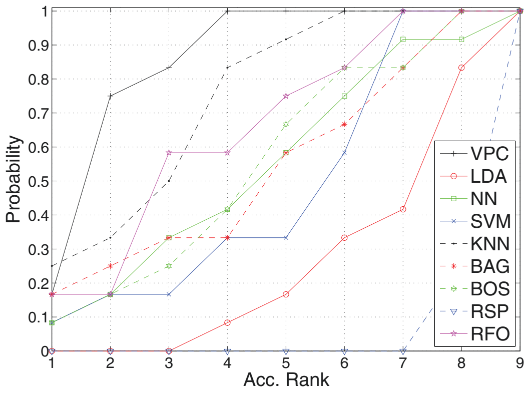

Figure 4. Accumulated probabilities of performance ranks observed for the eight algorithms in the datasets of the *UCI-Machine Learning Repository*

Demsar (2006) suggested that the fairest way to compare algorithms is to use their average ranking on multiple datasets and cross-validation accuracy. Hence, the order-rank of the algorithms in each problem was compared and the results illustrated in Figure 4, showing the accumulated probabilities in terms of ranks, i.e., the probabilities that an algorithm is among the top-k rank (*Acc. Rank* axis). Here, the *best* algorithm appears most close to the upper-left corner, which enables to intuitively visualise the relative effectiveness.

It is evident that VPC was the algorithm with ranks closest to the upper-left corner, followed by KNN and Random Forest. Next, a group of four algorithms (NN, SVM, Bagging and Boosting) got similar results. LDA appears next and the Random Subspaces algorithm got the worst results. Another particularly interesting property of VPC is that—for all problems—got performance among the top-half algorithms. This had only happened in about 85% for KNN, and around 60% of the problems for the Random Forest, which we consider a substantial difference. All the remaining algorithms were among the top-half in less than 50% of the problems.

Further attention should be given to the levels of correlation between the responses given by the best algorithms, in order to anticipate the advantages of using *meta-ensembles*, i.e., the improvements in performance that might result of fusing at the score level VPC, KNN, NN, SVM and Bagging classifiers.

4.4 Major Performance Covariates

According to the results given above, it is particularly important to perceive the factors that most evidently affect the performance of VPC with respect to the competitors. For such, we decided to compare the results of VPC to KNN and Random Forest classification strategies, with respect to

Figure 5. Examples of images used in the GTSR set released by the *Institute für Neuroinformatik*, in the scope of a competition held at the 2011 International Joint Conference on Neural Networks.

two different factors: 1) balance of classes prior probabilities; and 2) feature spaces dimension. The choice of the comparison terms used was motivated by the overall ranking of algorithms in the experiments above, summarized in Figure 4.

All results reported in this section were based on data from the German traffic sign recognition benchmark (Stallkamp et al., 2012) (Figure 5) was used. This data set is a multi-class image classification challenge held at the International Joint Conference on Neural Networks (IJCNN) 2011. There are 43 classes in the data set and more than 50 000 images [2], divided into disjoint training and test sets. Using the available meta-data that defines a region-of-interest for each sample, a set of 1,300 features per sample was extracted, scanning all 3×3 patches of the scale-normalized images (40×40 pixels) with the highly popular Local Binary Patterns descriptor (Ojala, Pietikainen, and Harwood, 1966). According to a bootstrapping-like strategy, random samples with 80% of the available learning and test data were drew, and the effectiveness of the measured, yielding the results given in the sub-sections below, where we plot the average performance plus the first and third quartile of the results, as a confidence interval.

4.4.1 Balance of Classes Prior Probabilities

To perceive the effect that the balance of classes priors has in VPC performance, we selected the most frequent classes in the GTSR data set (classes 1, 2, 12, 13 and 38). Next, we drew multiple samples of the learning and test sets, each one comprising two classes and varying the proportion of elements per class. Results are given in the left plot of Figure 6, for VPC (solid line), KNN (dashed line) and Random Forests (dashed-dotted

2. http://benchmark.ini.rub.de/

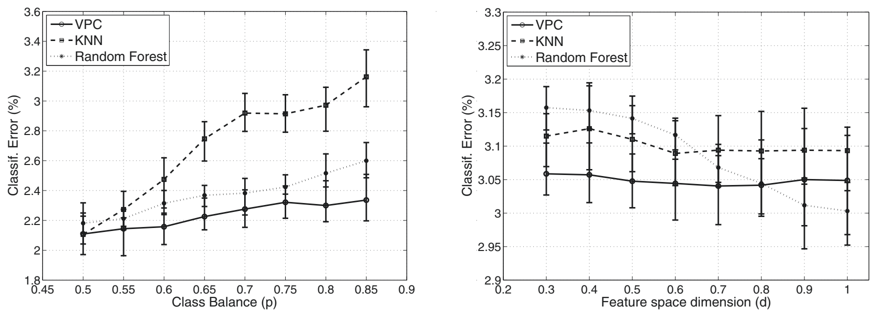

Figure 6. At left: Variations in performance with respect to the balance of classes prior probabilities. At right: Variations in performance with respect to the dimensionality of the feature spaces.

lines) algorithms, with the corresponding first and third quartile performance values observed (p is the proportion of elements the less frequent class) It is obvious that the gap between VPC and the competitors tends to increase directly in proportion to the levels of class unbalance, which accords our previous observations. Without surprise, KNN showed a larger deterioration in performance than the remaining algorithms with respect to this factor, whereas Random Forest showed a relatively small decrease in classification effectiveness.

4.5 Feature Spaces Dimension

Further, we compared the classification effectiveness of VPC, KNN and Random Forests with respect to the dimension of the feature space, which also correlates to the density of the learning feature space (as the number of used instances was kept constant). In this experiment, all classes of the GTSR set were considered, using feature subsets composed by 10 to 100% of all the available features, chosen randomly. The results are given in the right plot of Figure 6 (d represents the proportion of features considered), and appear to confirm that the performance of VPC with respect to competitors is maximized for problems of reduced and moderate dimensionality, i.e., corresponding to more densely populated feature spaces. For large dimensionality problems, the pivots chosen at each node of the classification tree were observed to decrease the representativity of the corresponding elements on that node. Noting that the confidence intervals associated with each algorithm largely overlap, it is still possible to perceive an inverse tendency between the performance of VPC/KNN and the Random Forest algorithm, that is the unique where the rule *"the more features the better"* appears to apply. Even due to different reasons, this does not holds for VPC and KNN, which is known to suffer from the *irrelevant features* issue that happens often in high dimensionality problems.

5. Conclusions

This paper proposes a classification strategy that accords the idea of Boosting and is based on a *Vantage-Point* tree that recursively divides the feature space into compact subspaces (leaves) that are separated by weak classifiers (linear discriminants). By preserving the neighborhood of subspaces, the binary data structure is traversed in a computationally efficient way and only a reduced number of leaves vote for the response of the ensemble, which yields the low computational cost of classification.

The resulting ensemble classifies in temporal cost of approximately $O(lg(n))$. Also, in terms of accuracy, it attains results similar to the state-of-the-art in most of the problems tested. The computational cost/accuracy balance is regarded in a particularly positive way due to the broad range of problems considered (binary/n-ary classification, discrete/continuous features, balanced /unbalanced priori probabilities, with densely/sparsely populated datasets).

In terms of the results observed, we highlight the following conclusions:

- Even though the proposed method (VPC) outperformed all the other algorithms in a relatively short proportion of the problems (2/12), the interesting property is that, for all problems VPC was among the best algorithms, which clearly did not happened for any of the remaining comparison terms.
- With respect to its competitors, the best results of VPC were observed for problems with unbalanced priori probabilities per class. This was explained by the fact that VPC classifies (locally) in subspaces, so that the prior probabilities in the complete feature space do not bias each local classifier.
- Also in terms of relative effectiveness, VPC is particularly suitable for problems of moderate dimensionality, where the vantage-point retrieval scheme works better. In very large dimensionally problems, as the $\ell 2$-norm was used to obtain the distance between feature points, pivots decrease the representativity of all the elements in the corresponding node.
- When compared to KNN, the major advantage of VPC is its smaller sensitivity to irrelevant features, which can be naturally disregarded by the linear discriminants at the tree leaves.
- It should be noted that in all the problems considered, the number of training instances n was always much higher than the dimensionally

d of the feature space. Hence, as further work, we plan to analyze the effectiveness of VPC in feature spaces with such high dimensionality and relatively reduced amount of learning data ($n \approx d$, or even $n < d$).

References

ALPAYDIN, E. (1999), "Combined 5 $\times$ 2 cv F-Test for Comparing Supervised Classification Learning Algorithms", *Neural Computation, 11(8)*, 1885–1892.

BANFIELD, R., HALL, L., BOWYER, K., and KEGELMEYER, W. (2007), "A Comparison of Decision Tree Ensemble Creation Techniques", *IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(1)*, 173–180.

BAUER, E., and KOHAVI, R. (1999), "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants", *Machine Learning, 36(1-2)*, 105–139.

BOCK, K., COUSSEMENT, K., and POEL, D. (2010), "Ensemble Classification Based on Generalized Additive Models", *Computational Statistics and Data Analysis, 54*, 1535–1546.

BREIMAN, L. (1996), "Bagging Predictors", *Machine Learning, 24(2)*, 123–140.

BREIMAN, L. (2001), "Random Forests", *Machine Learning, 45(1)*, 5–32.

BRYLL, R., GUTIERREZ-OSUNA, R., and QUEK, F. (2003), "Attribute Bagging: Improving Accuracy of Classifier Ensembles by Using Random Feature Subsets", *Pattern Recognition, 36(6)*, 291–1302,

CAI, D., HE, X., and HAN, J. (2008), "Training Linear Discriminant Analysis in Linear Time", in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pp. 209–217.

CANUTO, A., ABREU, M., OLIVEIRA, L., XAVIER JR., J., and SANTOS, A. (2007), "Investigating the Influence of the Choice of the Ensemble Members in Accuracy and Diversity of Selection-Based and Fusion-Based Methods for Ensembles", *Pattern Recognition Letters, 28(4)*, 472–486.

CECI, M., APPICE, A., and MALERBA, D. (2003), "Comparing Simplification Methods for Model Trees with Regression and Splitting Nodes", in *Proceedings of the Fourteenth International Symposium on Methodologies for Intelligent Systems,* Lecture Notes in Artificial Intelligence Vol. 2871, pp. 49–56.

CORTES, C., and VAPNIK, V. (1995), "Support Vector Networks", *Machine Learning, 20*, 1–25.

COVER, T., and HART, P. (1967), "Nearest Neighbor Pattern Classification", *IEEE Transactions on Information Theory, 13(1)*, 21–27.

DEMSAR, J. (2006), "Statistical Comparisons of Classifiers over Multiple Data Sets", *Journal of Machine Learning Research, 7*, 1–30.

DIETTERICH, T. (2000), "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization", *Machine Learning, 49(2)*, 139–157.

DOMINGOS, P. (1996), "Unifying Instance-Based and Rule-Based Induction", *Machine Learning, 24*, 141–168.

DUDA, R., HART, P., and STORK, D. (2000), *Pattern Classification* (2nd ed.), Wiley Interscience, ISBN 0-471-05669-3.

FRANK, E., HALL, M., and PFAHRINGER, B. (2003), "Locally Weighted Naive Bayes", in *Proceedings of the 19th conference on Uncertainty in Artificial Intelligence*, San Mateo, pp. 249–256.

FREUND, Y., and SCHAPIRE, R. (1995), "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting", in *Proceedings of the 2nd European Conference on Computational Learning Theory*, pp. 23–37.

HO, T.K. (1995), "Random Decision Forests", in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, pp. 278–282.

HO, T.K. (1998), "The Random Subspace Method for Constructing Decision Forests" *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8)*, 832–844.

HOTHORN, T., and LAUSEN, B. (2005), "Building Classifiers by Bagging Trees", *Computational Statistics and Data Analysis, 49*, 1068–1078.

JIRINA, M., and JIRINA JR., M. (2013), "Utilization of Singularity Exponent in Nearest Neighbor Based Classifier", *Journal of Classification, 30(1)*, 3–29.

JOHNSON, R., and WICHERN, D. (1988), *Applied Multivariate Statistic Analysis* (2nd. ed.), Englewood Cliffs NJ: Prentice Hall Inc.

KLEINBERG, E.M. (1990), "Stochastic Discrimination", *Annals of Mathematics and Artificial Intelligence, 1*, 207–239.

KUMAR, A. (2008), "Combining Pattern Classifiers: Methods and Algorithms", *IEEE Transactions on Industrial Electronics, 55(1)*, 348–363.

KUNCHEVA, L. (2004), *Combining Pattern Classifiers: Methods and Algorithms*, Hoboken NJ: John Wiley & Sons.

KUNCHEVA, L., and RODRÍGUEZ, J. (2007), "Classifier Ensembles with a Random Linear Oracle", *IEEE Transactions on Knowledge and Data Engineering, 19(4)*, 500–508.

LU, J., and TAN, Y-P. (2011), Nearest Feature Space Analysis for Classification", *IEEE Signal Processing Letters, 18(1)*, 55–58.

MOLLER, M. (1993), "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", *Neural Networks, 6*, 525–533.

OJALA, T., PIETIKAINEN, M., and HARWOOD, D. (1996), "A Comparative Study of Texture Measures with Classification Based on Feature Distributions", *Pattern Recognition, 29*, 51–59.

GARCÍA-PEDRAJAS, N. (2009), "Constructing Ensembles of Classifiers by Means of Weighted Instance Selection", *IEEE Transactions on Neural Networks, 20(2)*, 258–277.

SCHAPIRE, R. (1990), "The Strength of Weak Learnability", *Machine Learning, 5(2)*, 197–227.

SEIFFERT, C., KHOSHGOFTAAR, T., HULSE, J., and NAPOLITANO, A. (2008), "RUSBoost: Improving Classification Performance When Training Data Is Skewed", in *Proceedings of the 19th International Conference on Pattern Recognition*, pp. 1-4.

STALLKAMP, J., SCHLIPSING, M., SALMEN, J., and IGEL, C. (2012), "Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition", *Neural Networks, 32*, 323–332.

TING, K.M., WELLS, J.R., TAN, S.C., TENG, S.W., and WEBB, G.I. (2011), "Feature-Subspace Aggregating: Ensembles for Stable and Unstable Learners", *Machine Learning, 82*, 375–397.

VIOLA, P.A., and JONES, M.J. (2004), "Robust Real-Time Face Detection", *International Journal of Computer Vision, 57(2)*, 137–154.

YAN, R., and TEŠIĆ, J. (2007), "Model-Shared Subspace Boosting for Multi-Label Classification", in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 834–843.

YIANILOS, P. (1993), "Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces", in *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics,* pp. 311–321.

YU, G., ZHANG, G., YU, Z., DOMENICONI, C., YOUC, J., and HANA, G. (2012), "Semi-Supervised Ensemble Classification in Subspaces", *Applied Soft Computing, 12*, 1511–1522.

ZAMAN, M., and HIROSE, H. (2013), "DF-SVM: A Decision Forest Constructed on Artificially Enlarged Feature Space by Support Vector Machine", *Artificial Intelligence Review, 40*, 467–494.