Supervised Pattern Classification based on Optimum-Path Forest

João P. Papa, Alexandre X. Falcão and Celso T. N. Suzuki
Institute of Computing, University of Campinas
Email: {jpaulo,afalcao}@ic.unicamp.br, celso.suzuki@gmail.com

Corrected version (for Figs. 3a and 3b) of the paper published in Intl. Journal of Imaging Systems and Technology, Wiley, Vol. 19, Issue 2, pp. 120–131, Jun 2009, ISSN: 0899-9457, doi.wiley.com/10.1002/ima.20188

Abstract

We present a supervised classification method which represents each class by one or more optimum-path trees rooted at some key samples, called *prototypes*. The training samples are nodes of a complete graph, whose arcs are weighted by the distances between the feature vectors of their nodes. Prototypes are identified in all classes and the minimization of a *connectivity function* by dynamic programming assigns to each training sample a minimum-cost path from its most strongly connected prototype. This competition among prototypes partitions the graph into an optimum-path forest rooted at them. The class of the samples in an optimum-path tree is assumed to be the same of its root. A test sample is classified similarly, by identifying which tree would contain it, if the sample were part of the training set. By choice of the graph model and connectivity function, one can devise other optimum-path forest classifiers. We present one of them, which is fast, simple, multi-class, parameter independent, does not make any assumption about the shapes of the classes, and can handle some degree of overlapping between classes. We also propose a general algorithm to learn from errors on an evaluation set without increasing the training set, and show the advantages of our method with respect to SVM, ANN-MLP, and k-NN classifiers in several experiments with datasets of various types.

1 Introduction

Patterns are usually represented by feature vectors (set of measures or observations) obtained from samples of a dataset [1]. Two fundamental problems in pattern recognition are: (i) the identification of natural groups (clustering) composed by samples with similar patterns and (ii) the classification of each sample in one of c possible classes (labels). The dataset is usually divided in two parts, a training set and a test set, being the first used to project the classifier and the second used for validation, by measuring its classification errors (accuracy). This process must be also repeated several times with randomly selected training and test samples to achieve a conclusion about the statistics of its accuracy (robustness and precision). While problem (i) has no prior information about the labels of the samples, the training in problem (ii) can count with unlabeled samples (unsupervised learning), labeled samples (supervised learning) or part of the samples labeled and the other part unlabeled (semi-supervised learning [2–4]). Our focus is on the supervised learning approaches.

Figure 1 illustrates three typical cases in 2D feature spaces using two classes: (a) linearly separable, (b) piecewise linearly separable, and (c) overlapping classes with arbitrary shapes. Any reasonable approach should handle (a) and (b), being (c) the most interesting challenge. An artificial neural network with multi-layer perceptrons (ANN-MLP), for example, can address (a) and (b), but not (c) [5]. As an unstable classifier, collections of ANN-MLP [6] can improve its performance up to some unknown limit of classifiers [7]. Support vector machines (SVMs) have been proposed to overcome the problem, by assuming linearly separable classes in a higher-dimensional feature space [8]. Its computational cost rapidly increases with the training set size and the number of support vectors. As a binary classifier, multiple SVMs are required to solve a multi-class problem [9]. Tang and Mazzoni [10] proposed a method to reduce the number of support vectors in the multi-class problem. Their approach suffers from slow convergence and high computational cost, because they first minimize the number of support vectors in several binary SVMs, and then share these vectors among the machines. Panda et al. [11] presented a method to reduce the training set size before computing the SVM algorithm. Their approach aims to identify and remove samples likely related to non-support vectors. However, in all SVM approaches, the assumption of separability may also not be valid in any space of finite dimension [12].

We propose a supervised classifier based on *optimum-path forest* (OPF), which is fast, simple, multi-class, parameter independent, does not make any assumption about the shapes of the classes, and can handle some degree of overlapping between classes. The training set is thought of as a complete graph, whose nodes are the samples and arcs link all pairs of nodes. The arcs are weighted by the distances between the feature vectors of their corresponding nodes. Any sequence of distinct samples forms a path connecting the terminal nodes and a *connectivity function* assigns a cost to that path (e.g., the maximum arc-weight along it). The idea is to identify prototypes in each class such that every sample is assigned to the class of its most strongly connected prototype. That is, the one which offers to it a minimum-cost path, considering all possible paths from the prototypes. Figure 1 shows two sets of prototypes, S_1 and S_2, in classes 1 and 2. The connection from S_i to a sample t is represented by a path $\pi_t^{(i)}$ with terminus t and root in some prototype of S_i, $i = 1, 2$. In all cases, the optimum path (to which the maximum arc-weight is minimum) comes from a prototype of the same class of t. Our approach can handle all three cases with the maximum arc-weight function and prototypes estimated as the closest samples from distinct classes. In the case of overlapping between classes, these prototypes work as class defenders in the overlapped regions of the feature space (Figure 1c).

The classifier is an optimum-path forest rooted at the prototypes. That is, each training sample belongs to one optimum-path tree rooted at its most strongly connected prototype. The classification of a test sample evaluates the optimum paths from the prototypes to this sample incrementally, as though it were part of the forest, and assigns to it the label of the most strongly connected root. Note the difference between the proposed method with the maximum arc-weight function and the nearest neighbor approach [13]. A test/training sample may be assigned to a given class, even when its closest labeled sample is from another class (Figure 1b).

The optimum paths from the prototypes to the other samples are computed by the algorithm of the image foresting transform (IFT) — a tool for the design of image processing operators based on connectivity [14] — which is extended here from the image domain to the feature space. The IFT algorithm is essentially Dijkstra's algorithm [15] modified for multiple sources and more general path-value functions [14]. It first identifies the minima (maxima) of the path-value function as source nodes and then propagates optimum paths from those sources in a non-decreasing (non-increasing) order of optimum-path values, partitioning the graph into an optimum-path forest rooted at the source nodes. It is a dynamic programming strategy in which, by choice of the path-value function (Equation 1 in Section 2.1), we force the prototypes to be the roots of the forest.

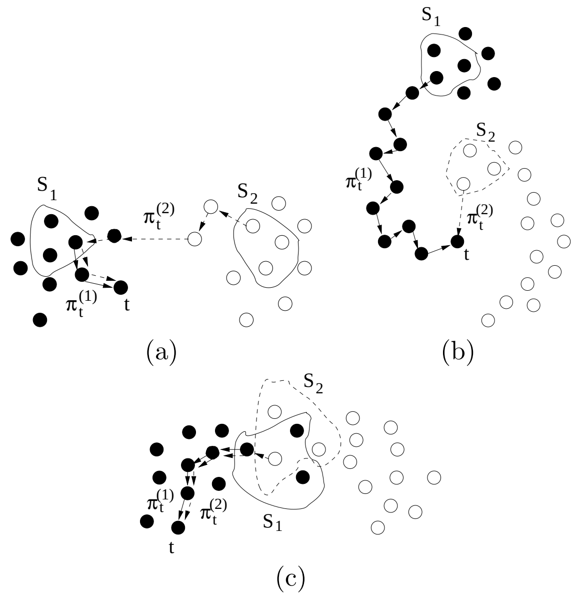

Figure 1: Examples of 2D feature spaces using two classes: (a) linearly separable, (b) piecewise linearly separable, and (c) overlapping classes with arbitrary shapes. Prototypes can be identified in each class, forming the sets S_1 and S_2. Every sample t can be connected to a prototype in S_i, $i = 1, 2$, by a sequence $\pi_t^{(i)}$ of distinct samples. The classification is done based on optimal connections to the prototypes.

The dataset partition by the proposed classifier in the feature space is equivalent to an image segmentation by the IFT-watershed transform from labeled markers [16, 17] in the image domain. Similar important relations can be obtained with other image operators, such as relative-fuzzy connected segmentation [18–21]. In our case, the markers are the prototypes and we have a special way to estimate them. Figure 2 helps to understand this comparison and why the proposed method works in the feature space, when prototypes are estimated as the closest samples from distinct classes. Figure 2a shows an image with one internal marker (white) and one external marker (black) for an object of interest. The pixels are the nodes of a graph whose arcs link the 8-neighbors of each pixel. The arc weights are dissimilarity values between pixels, computed based on their image properties. The dissimilarity function between pixels plays the same role of the distance function between samples and distinct classes are represented by object and background. The connectivity function is the maximum arc-weight along the path. Figure 2b gives an idea of the arc weights by displaying the complement of a gradient-like image, which is created by assigning to each pixel the maximum among the arc weights between it and its eight neighbors. By selecting markers around the weaker parts (lower arc weights) of the boundary (Figure 2a), we force the minimum-cost paths from internal and external markers to meet first at the weaker parts of the object's boundary, blocking these passages for paths from the other side. Therefore, possible paths from one side to the other will have costs higher than paths from the same side with respect to each marker. The optimum-path propagation from both markers describes an ordered region growing (flooding) process where the wavefronts from each marker meet at the object's boundary (Figure 2c). The object is defined by the optimum-path forest rooted at the pixels of the internal marker. In the case of multiple internal and external markers, the object is composed by multiple internal forests. Three frames of this process are presented in Figures 2d- 2f. Note that internal (external) pixels, which are only reachable by high-cost paths, are initally surrounded by optimum paths from the internal (external) marker and finally conquered by this marker. We can also exploit other connectivity functions, but this work presents only the results for the maximum arc-weight function.

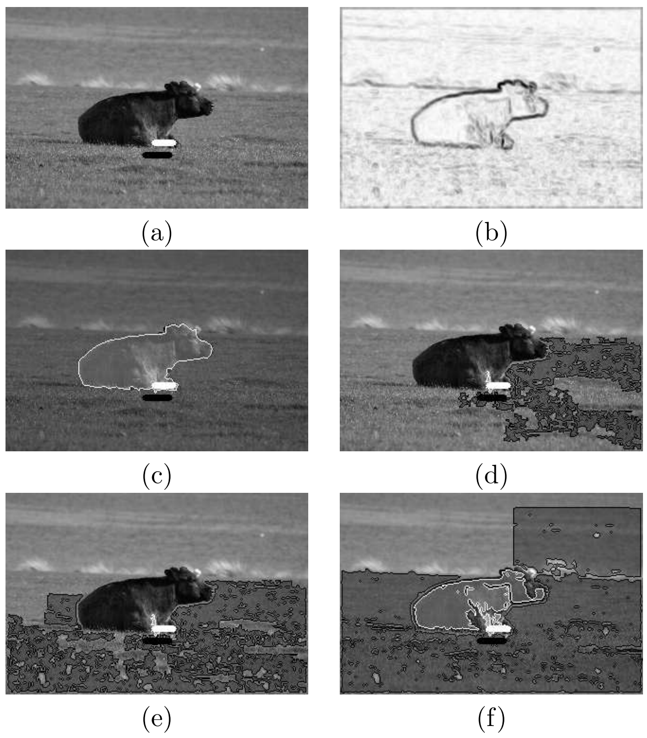
(a) (b) (c) (d) (e) (f)

Figure 2: IFT-watershed segmentation. (a) Image with internal (white) and external (black) markers. (b) The complement of a gradient-like image which gives an idea of the arc weigths. The markers are selected around the wearker parts of the boundary (brighter values in b). (c) The result of segmentation and (d-f) three frames of the IFT flooding process that leads to (c).

Supervised classification based on prototypes is not new. For example, methods such as the k-nearest neighbors (k-NN) use all training samples as prototypes [22]. Its classification relies on the direct distance between samples. As far as we know, our approach is the first to consider optimum-path forests rooted at automatically selected prototypes in the feature space. Besides, by changing the graph model and path-value function, one can derive other types of optimum-path forest classifiers, such as the unsupervised learning approach proposed in [23, 24], which also relies on a different strategy to estimate prototypes. Most approaches for pattern classification based on graphs and/or paths in graphs are either unsupervised [25–28] or semi-supervised [29–32]. The proposed method can be easily extended to semi-supervised classification, given that the optimum-path forest can include unlabeled non-prototype samples. Previous versions of it have also been published [33–37]. We have simplified the learning procedure with better results, corrected some mistakes, improved explanations and added several experiments using more datasets, baseline classifiers, and image descriptors based on texture, shape and color.

Other contribution of this work concerns learning algorithms, which can teach a classifier from its errors on a third evaluation set without increasing the size of the training set. As the samples in the test set can not be seen during the project, the evaluation set is necessary for this purpose. The basic idea is to randomly interchange samples of the training set with misclassified samples of the evaluation set, retrain the classifier and evaluate it again, repeating this procedure during a few iterations. The effectiveness is measured by comparing the results on the unseen test set before and after the learning algorithm. It is expected an improvement in performance for any stable classifier.

The learning with fixed training set size is usually required in large datasets with thousands/millions of samples (e.g., pixels/voxels in 2D/3D images). It also stems from applications where the classifier is part of an expert system, which performs a laborious data analysis (sometimes inviable for human beings) and emits its opinion to a human expert. The human expert may agree or not based on other evidences, but the feedback about the classification errors is important to improve performance in a future analysis. The diagnosis of parasites from microscopy images of biological slides is an example [38]. The human visual inspection is very difficult and error prone in several situations due to the amount of impurities and small sizes of some parasites (e.g., protozoa in samples of feces). We aim to improve the performance of the expert system along time of use and we are taking into account the fact that computers have a limited storage and processing capacity for the training set.

This paper describes the supervised OPF classifier in

Section 2, presents a general learning algorithm in Section 3, which follows the same aforementioned strategy for all classifiers, shows results that compare the OPF classifier with SVM [8], ANN-MLP [5] and k-NN [22] in Section 4, and states conclusions in Section 5.

2 Optimum-path forest classifier

Let Z_1, Z_2, and Z_3 be training, evaluation, and test sets with $|Z_1|$, $|Z_2|$, and $|Z_3|$ samples of a given dataset. We use samples as points, images, voxels, and contours in this paper. As already explained, this division of the dataset is necessary to validate the classifier and evaluate its learning capacity from the errors. Z_1 is used to project the classifier and Z_3 is used to measure its accuracy, being the labels of Z_3 kept unseen during the project. A pseudo-test on Z_2 is used to teach the classifier by randomly interchanging samples of Z_1 with misclassified samples of Z_2. After learning, it is expected an improvement in accuracy on Z_3.

Let $\lambda(s)$ be the function that assigns the correct label i, $i = 1, 2, \ldots, c$, of class i to any sample $s \in Z_1 \cup Z_2 \cup Z_3$, $S \subset Z_1$ be a set of prototypes from all classes, and v be an algorithm which extracts n features (color, shape, texture properties) from any sample $s \in Z_1 \cup Z_2 \cup Z_3$ and returns a vector $\vec{v}(s)$. The distance $d(s,t) \geq 0$ between two samples, s and t, is the one between their feature vectors $\vec{v}(s)$ and $\vec{v}(t)$. One can use any distance function suitable for the extracted features. The most common is the Euclidean norm $\|\vec{v}(t) - \vec{v}(s)\|$, but some image features require special distance algorithms [39, 40]. A pair (v, d) then describes how the samples of a dataset are distributed in the feature space. Therefore, we call (v, d) a *descriptor* and the experiments in Section 4 use shape [41], texture [35] and color [42] descriptors based on this definition.

Our problem consists of projecting a classifier which can predict the correct label $\lambda(s)$ of any sample $s \in Z_3$. Training consists of finding a special set $S^* \subset Z_1$ of prototypes and a discrete optimal partition of Z_1 in the feature space (i.e., an optimum-path forest rooted in S^*). The classification of a sample $s \in Z_3$ (or $s \in Z_2$) is done by evaluating the optimum paths incrementally, as though it were part of the forest, and assigning to it the label of the most strongly connected prototype.

2.1 Training

Let (Z_1, A) be a complete graph whose nodes are the training samples and any pair of samples defines an arc in $A = Z_1 \times Z_1$ (Figure 3a). The arcs do not need to be stored and so the graph does not need to be explicitly represented. A path is a sequence of distinct samples $\pi_t = \langle s_1, s_2, \ldots, t\rangle$ with terminus at a sample t. A path is said *trivial* if $\pi_t = \langle t\rangle$. We assign to each path π_t a cost $f(\pi_t)$ given by a connectivity function f. A path π_t is said optimum if $f(\pi_t) \leq f(\tau_t)$ for any other path τ_t. We also denote by $\pi_s \cdot \langle s,t\rangle$ the concatenation of a path π_s and an arc (s,t).

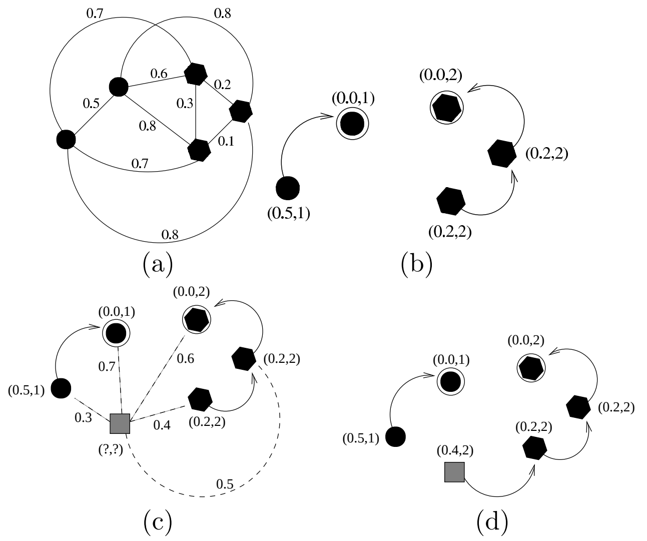

Figure 3: (a) Complete weighted graph for a simple training set. (b) Resulting optimum-path forest for $f_{\max}$ and two given prototypes (circled nodes). The entries (x, y) over the nodes are, respectively, the cost and the label of the samples. The directed arcs indicate the predecessor nodes in the optimum path. (c) Test sample (gray square) and its connections (dashed lines) with the training nodes. (d) The optimum path from the most strongly connected prototype, its label 2, and classification cost 0.4 are assigned to the test sample. The test sample is classified in the class hexagon, although its nearest training sample is from the class circle.

We will address the connectivity function $f_{\max}$.

$$\begin{aligned} f_{\max}(\langle s\rangle) &= \begin{cases} 0 & \text{if } s \in S, \\ +\infty & \text{otherwise} \end{cases} \\ f_{\max}(\pi_s \cdot \langle s,t\rangle) &= \max\{f_{\max}(\pi_s), d(s,t)\} \end{aligned} \quad (1)$$

such that $f_{\max}(\pi_s \cdot \langle s,t\rangle)$ computes the maximum distance between adjacent samples along the path $\pi_s \cdot \langle s,t\rangle$. The minimization of $f_{\max}$ assigns to every sample $t \in Z_1$ an optimum path $P^*(t)$ from the set $S \subset Z_1$ of prototypes, whose minimum cost $C(t)$ is

$$C(t) = \min_{\forall \pi_t \in (Z_1, A)} \{f_{\max}(\pi_t)\}. \quad (2)$$

The minimization of $f_{\max}$ is computed by Algorithm 1, called OPF algorithm, which is an extension of the general image foresting transform (IFT) algorithm [14] from the image domain to the feature space, here specialized for $f_{\max}$. As explained in Section 1, this process assigns one optimum path from S to each training sample t in a non-decreasing order of minimum cost, such that the graph is partitioned into an optimum-path forest P (a function with no cycles which assigns to each $t \in Z_1 \backslash S$ its predecessor $P(t)$ in $P^*(t)$ or a marker *nil* when $t \in S$, as shown in Figure 3b). The root $R(t) \in S$ of $P^*(t)$ can be obtained from $P(t)$ by following the predecessors backwards along the path, but its label is propagated during the algorithm by setting $L(t) \leftarrow \lambda(R(t))$.

Algorithm 1 – OPF ALGORITHM

INPUT:	A training set Z_1, λ-labeled prototypes $S \subset Z_1$ and the pair (v, d) for feature vector and distance computations.
OUTPUT:	Optimum-path forest P, cost map C and label map L.
AUXILIARY:	Priority queue Q and cost variable *cst*.

```
1.  For each $s \in Z_1 \backslash S$, set $C(s) \leftarrow +\infty$.
2.  For each $s \in S$, do
3.      $C(s) \leftarrow 0$, $P(s) \leftarrow nil$, $L(s) \leftarrow \lambda(s)$, and insert $s$ in $Q$.
4.  While $Q$ is not empty, do
5.      Remove from $Q$ a sample $s$ such that $C(s)$ is minimum.
6.      For each $t \in Z_1$ such that $t \neq s$ and $C(t) > C(s)$, do
7.          Compute $cst \leftarrow \max\{C(s), d(s,t)\}$.
8.          If $cst < C(t)$, then
9.              If $C(t) \neq +\infty$, then remove $t$ from $Q$.
10.             $P(t) \leftarrow s$, $L(t) \leftarrow L(s)$ and $C(t) \leftarrow cst$.
11.             Insert $t$ in $Q$.
```

Lines $1-3$ initialize maps and insert prototypes in Q. The main loop computes an optimum path from S to every sample s in a non-decreasing order of minimum cost (Lines $4-11$). At each iteration, a path of minimum cost $C(s)$ is obtained in P when we remove its last node s from Q (Line 5). Ties are broken in Q using first-in-first-out policy. That is, when two optimum paths reach an ambiguous sample s with the same minimum cost, s is assigned to the first path that reached it. Note that $C(t) > C(s)$ in Line 6 is false when t has been removed from Q and, therefore, $C(t) \neq +\infty$ in Line 9 is true only when $t \in Q$. Lines $8-11$ evaluate if the path that reaches an adjacent node t through s is cheaper than the current path with terminus t and update the position of t in Q, $C(t)$, $L(t)$ and $P(t)$ accordingly.

One can use other *smooth* connectivity functions, as long as they group samples with similar properties [14]. A function f is smooth in (Z_1, A) when for any sample $t \in Z_1$, there exists an optimum path π_t which either is trivial or has the form $\pi_s \cdot \langle s, t\rangle$, where

(a) $f(\pi_s) \leq f(\pi_t)$,

(b) π_s is optimum,

(c) for any optimum path τ_s, $f(\tau_s \cdot \langle s, t\rangle) = f(\pi_t)$.

We say that S^* is an optimum set of prototypes when Algorithm 1 minimizes the classification errors in Z_1. S^* can be found by exploiting the theoretical relation between minimum-spanning tree (MST) [15] and optimum-path tree for $f_{\max}$ [21, 43].

By computing a MST in the complete graph (Z_1, A), we obtain a connected acyclic graph whose nodes are all samples of Z_1 and the arcs are undirected and weighted by the distances d between adjacent samples (Figure 4a). The spanning tree is optimum in the sense that the sum of its arc weights is minimum as compared to any other spanning tree in the complete graph. In the MST, every pair of samples is connected by a single path which is optimum according to $f_{\max}$. That is, the minimum-spanning tree contains one optimum-path tree for any selected root node.

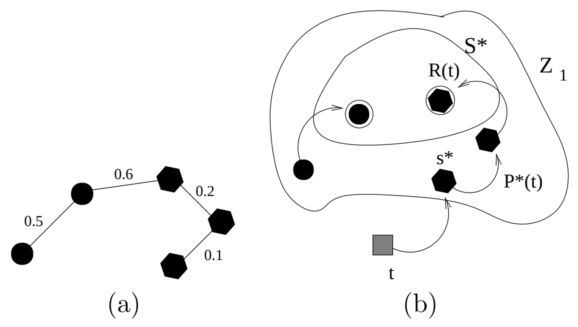

Figure 4: (a) MST of the graph shown in Figure 3a where the optimum prototypes share the arc of weight 0.6. (b) The classification of a test sample (gray square) t as in Figure 3c assigns the optimum path $P^*(t)$ from $R(t) \in S^*$ to t passing through s^*.

The optimum prototypes are the closest elements of the MST with different labels in Z_1. By removing the arcs between different classes, their adjacent samples become prototypes in S^* and Algorithm 1 can compute an optimum-path forest in Z_1 (Figure 3b). Note that, a given class may be represented by multiple prototypes (i.e., optimum-path trees) and there must exist at least one prototype per class.

It is not difficult to see that the optimum paths between classes tend to pass through the same removed arcs of the minimum-spanning tree. The choice of prototypes as described above aims to block these passages, reducing the chances of samples in any given class be reached by optimum paths from prototypes of other classes.

2.2 Classification

For any sample $t \in Z_3$, we consider all arcs connecting t with samples $s \in Z_1$, as though t were part of the training graph (Figure 3c). Considering all possible paths from S^* to t, we find the optimum path $P^*(t)$ from S^* and label t

with the class $\lambda(R(t))$ of its most strongly connected prototype $R(t) \in S^*$ (Figure 4b). This path can be identified incrementally, by evaluating the optimum cost $C(t)$ as

$$C(t) = \min\{\max\{C(s), d(s,t)\}\}, \ \forall s \in Z_1. \tag{3}$$

Let the node $s^* \in Z_1$ be the one that satisfies Equation 3 (i.e., the predecessor $P(t)$ in the optimum path $P^*(t)$). Given that $L(s^*) = \lambda(R(t))$, the classification simply assigns $L(s^*)$ as the class of t (Figure 3d). An error occurs when $L(s^*) \neq \lambda(t)$.

Similar procedure is applied for samples in the evaluation set Z_2. In this case, however, we would like to use misclassified samples of Z_2 to learn the distribution of the classes in the feature space and improve the classification performance on Z_3.

3 Learning from errors on the evaluation set

There are many situations that limit the size of Z_1: large datasets, limited computational resources, and high computational time as required by some approaches. Mainly in applications with large datasets, it would be interesting to select for Z_1 the most informative samples, such that the accuracy of the classifier is little affected by this size limitation. It is also important to show that a classifier can improve its performance along time of use, when we are able to teach it from its errors. This section presents a general learning algorithm which uses a third evaluation set Z_2 to improve the composition of samples in Z_1 without increasing its size.

From an initial choice of Z_1 and Z_2, the algorithm projects an instance I of a given classifier from Z_1 and evaluates it on Z_2. The misclassified samples of Z_2 are randomly selected and replaced by samples of Z_1 (under certain constraints). This procedure assumes that the most informative samples can be obtained from the errors. The new sets Z_1 and Z_2 are then used to repeat the process during a few iterations T. The instance of classifier with highest accuracy is selected along the iterations. The accuracy values $\mathcal{L}(I)$ obtained for each instance I form a *learning curve*, whose non-decreasing monotonic behavior indicates a positive learning rate for the classifier. Afterwards, by comparing the accuracies of the classifier on Z_3, before and after the learning process, we can evaluate its learning capacity from the errors.

The accuracies $\mathcal{L}(I)$, $I = 1, 2 \ldots, T$, are measured by taking into account that the classes may have different sizes in Z_2 (similar definition is applied for Z_3). If there are two classes, for example, with very different sizes and a classifier always assigns the label of the largest class, its accuracy will fall drastically due to the high error rate on the smallest class.

Let $NZ_2(i)$, $i = 1, 2, \ldots, c$, be the number of samples in Z_2 from each class i. We define

$$e_{i,1} = \frac{FP(i)}{|Z_2| - |NZ_2(i)|} \quad \text{and} \quad e_{i,2} = \frac{FN(i)}{|NZ_2(i)|}, \ i = 1, \ldots, c \tag{4}$$

where $FP(i)$ and $FN(i)$ are the false positives and false negatives, respectively. That is, $FP(i)$ is the number of samples from other classes that were classified as being from the class i in Z_2, and $FN(i)$ is the number of samples from the class i that were incorrectly classified as being from other classes in Z_2. The errors $e_{i,1}$ and $e_{i,2}$ are used to define

$$E(i) = e_{i,1} + e_{i,2}, \tag{5}$$

where $E(i)$ is the partial sum error of class i. Finally, the accuracies $\mathcal{L}(I)$, $I = 1, 2 \ldots, T$, are written as

$$\mathcal{L}(I) = \frac{2c - \sum_{i=1}^{c} E(i)}{2c} = 1 - \frac{\sum_{i=1}^{c} E(i)}{2c}. \tag{6}$$

Algorithm 2 presents this learning procedure which has been used for OPF, SVM, ANN-MLP and k-NN, by changing Lines 4 and $19 - 20$.

Algorithm 2 – General Learning Algorithm

```
INPUT:      Training and evaluation sets, $Z_1$ and $Z_2$, labeled
            by $\lambda$, number $T$ of iterations, and the pair $(v, d)$
            for feature vector and distance computations.
OUTPUT:     Learning curve $\mathcal{L}$ and the OPF/SVM/ANN-
            MLP/$k$-NN classifier with highest accuracy.
AUXILIARY:  Arrays $FP$ and $FN$ of sizes $c$ for false positives
            and false negatives and list $LM$ of misclassified
            samples.

1.  Set $MaxAcc \leftarrow -1$.
2.  For each iteration $I = 1, 2, \ldots, T$, do
3.      $LM \leftarrow \emptyset$
4.      Train OPF/SVM/ANN-MLP/$k$-NN with $Z_1$.
5.      For each class $i = 1, 2, \ldots, c$, do
6.          $FP(i) \leftarrow 0$ and $FN(i) \leftarrow 0$.
7.      For each sample $t \in Z_2$, do
8.          Use the classifier obtained in Line 3 to classify $t$
9.          with a label $L(t)$.
10.         If $L(t) \neq \lambda(t)$, then
11.             $FP(L(t)) \leftarrow FP(L(t)) + 1$.
12.             $FN(\lambda(t)) \leftarrow FN(\lambda(t)) + 1$.
13.             $LM \leftarrow LM \cup t$.
14.     Compute accuracy $\mathcal{L}(I)$ by Equation 6.
15.     If $\mathcal{L}(I) > MaxAcc$ then save the current instance
16.     of the classifier and set $MaxAcc \leftarrow \mathcal{L}(I)$.
17.     While $LM \neq \emptyset$
18.         $LM \leftarrow LM \backslash t$
19.         Replace $t$ by a randomly selected sample of the
20.         same class in $Z_1$, under some constraints.
```

In OPF, Line 4 is implemented by computing $S^* \subset Z_1$ as described in Section 2.1 and the predecessor map P, label map L and cost map C by Algorithm 1. The classification is done by setting $L(t) \leftarrow L(s^*)$, where $s^* \in Z_1$ is the sample that satisfies Equation 3. The constraints in Lines $19-20$ refer to keep the prototypes out of the sample interchanging process between Z_1 and Z_2. We do the same with the support vectors in SVM. However, they may be selected for interchanging in future iterations if they are no longer prototypes or support vectors. For SVM, we use the latest version of the LibSVM package [44] with Radial Basis Function (RBF) kernel, parameter optimization and the one-versus-one strategy for the multi-class problem to implement Line 4.

We use the Fast Artificial Neural Network Library (FANN) [45] to implement the ANN-MLP. The network configuration is x:y:z, where $x=n$ (number of features), $y=|Z_1|-1$ and $z=c$ (number of classes) are the number of neurons in the input, hidden and output layers, respectively [46]. In Line 4, the ANN-MLP is trained by back propagation. There is no constraint in Lines $19-20$. However, we keep the weights of the neurons as initial setting for training in the next iteration. For k-NN, training in Line 4 involves the computation of the value of k which provides the highest accuracy on Z_1 according to the Leave-One-Out approach [47]. Lines $19-20$ are implemented without constraints.

Lines $5-6$ initialize the false positive and false negative arrays for accuracy computation. The classification of each sample is performed in Lines $7-13$, updating the false positive and false negative arrays. Misclassified samples are stored in the list LM (Line 13). Line 14 computes the accuracy $\mathcal{L}(I)$ and Lines $15-16$ save the best instance of classifier so far. The inner loop in Lines $17-20$ changes the misclassified samples of Z_2 by randomly selected samples of Z_1, under the aforementioned constraints.

Figure 5 illustrates the learning curve of each classifier for the same dataset and descriptor. Oscillations indicate instability of the classifier (e.g., $ANN-MLP$) or presence of outliers. The monotonic behavior of the OPF's learning curve is usually observed. Nevertheless, the choice of the classifier instance with highest accuracy aims to avoid outliers in Z_1.

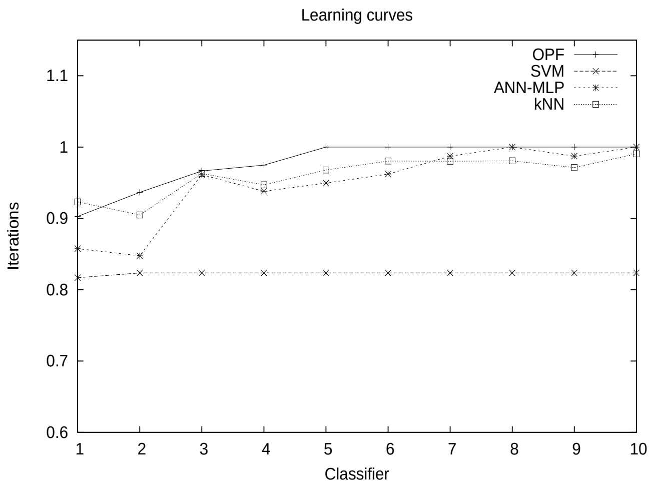

Figure 5: Learning curve of each classifier for the dataset B_8 using descriptor D_{10} (Tables 1 and 2).

Dataset Code	Dataset Name	objects number	classes number
B_1	MPEG-7	1400	70
B_2	COREL	1607	49
B_3	Brodatz	208	13
B_4	WBC	699	2
B_5	IS	2310	7
B_6	LR	5000	26
B_7	Brain	1578	2
B_8	Cone-torus	400	3
B_9	Saturn	200	2
B_{10}	Petals	100	4
B_{11}	Boat	100	3

Table 1: Description of the datasets.

4 Evaluation

This section presents the datasets, descriptors, and experiments that compare OPF with SVM, ANN-MLP and k-NN in accuracy and efficiency (computational time).

Table 1 presents the 11 datasets used in the experiments, with diverse types of samples. The dataset MPEG-7 [48] uses shape images (Figure 6), COREL [49] uses color images (Figure 7), presented here in grayscale, and Brodatz [50] uses texture images (Figure 8). These datasets allow to evaluate the performance of the classifiers using shape, color and texture descriptors, respectively. The remaining datasets already provide their feature vectors: WBC - Wisconsin Breast Cancer, IS - Image Segmentation, and LR - Letter Recognition [51]; Brain [52]; and Cone-torus, Saturn, Petals, and Boat [53]. The dataset Brain uses voxels as samples from gray and white matter in magnetic resonance images of brain phantoms, with various levels of noise and inhomogeneity that produce outliers. The features are the minimum, maximum, and intensity within a small 3D neighborhood of each voxel. The last four datasets use the (x, y) coordinates of 2D points as features (Figure 9).

Table 2 shows 10 different possibilities of combining feature extraction v and distance function d to form descriptors (v, d). Some descriptors were designed for shape (D_1-D_5), color (D_6-D_7) and texture (D_8) images. Descriptors D_1, D_2 and D_3 use the Fourier coefficients (FC) [54], Moment Invariants (MI) [55] and multiscale fractal dimensions (MSF) [56] as shape features, respectively, and Eu-

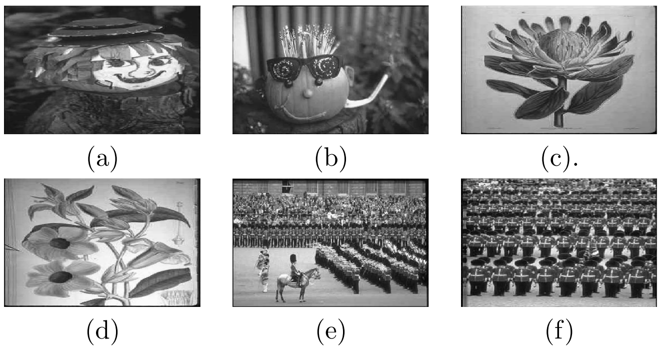

Figure 6: Examples of the MPEG-7 shapes from the classes (a)-(c) fish and (d)-(f) camel.

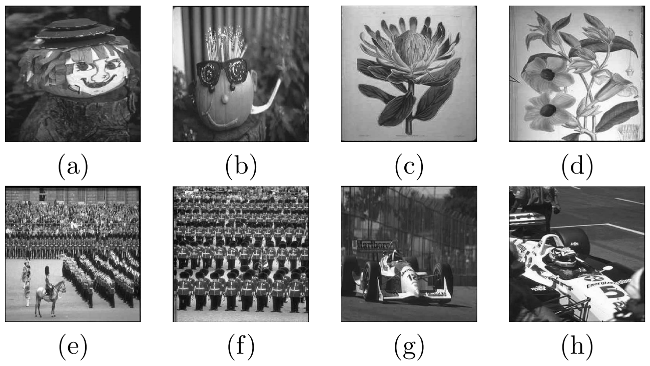

Figure 7: Examples of the Corel images from the classes (a)-(b) pumpkin, (c)-(d) flowers, (e)-(f) British army, and (g)-(h) race cars.

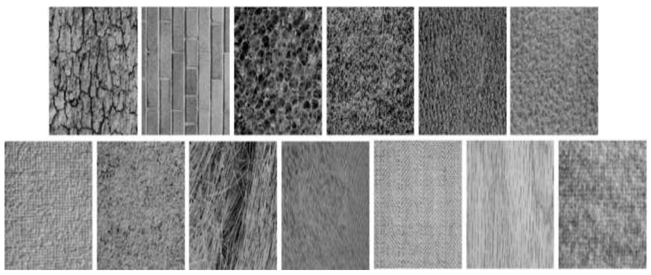

Figure 8: Texture images from the Brodatz dataset. Each image, from left to right and from top to bottom, represents a class: bark, brick, bubbles, grass, leather, pigskin, raffia, sand, straw, water, weave, wood, and wool.

clidean norm (L2) as distance function. Descriptors D_4 and D_5 compute three statistical measures, called bean angle statistics (BAS), for each sample on a contour [41]. They use L2 metric and optimal correspondence subsequence (OCS) [39], respectively, for comparison between feature vectors, illustrating the importance of special distance functions such as OCS. The comparison among descriptors from D_1 to D_5 using a same classifier illustrates their ability in representing the shapes of a given dataset.

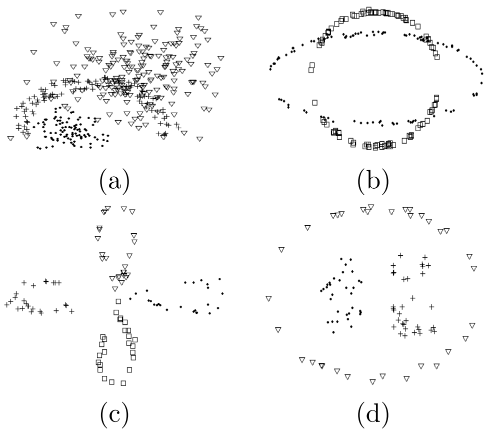

Figure 9: Datasets of 2D points: (a) Cone-torus, (b) Saturn, (c) Petals, and (d) Boat.

Descriptor Code	Feature extraction algorithm	Distance function
D_1	FC	L2
D_2	MI	L2
D_3	MSF	L2
D_4	BAS	L2
D_5	BAS	OCS
D_6	BIC	dLog
D_7	CHIST	L1
D_8	TEX	RIM
D_9	OWN	L2
D_{10}	XY	L2

Table 2: Descriptors used in the experiments.

Descriptor D_6 classifies pixels into border/interior regions and computes color histograms for each region [42]. It uses as distance function the L1 metric between the logarithm of the histograms (dLog). Color images are also represented by color histograms (CHIST) [57] and compared with L1 metric in the descriptor D_7. Descriptor D_8 uses steerable pyramid decomposition to create texture features (TEX), which are compared by a rotation-invariant texture matching (RIM) [35]. Descriptor D_9 represents all feature vectors (OWN) already available in the datasets from B_4 to B_7 and D_{10} represents the 2D-point (XY) features of the datasets from B_8 to B_{11} (Table 1). Their distance function is L2. Finally, the combinations between datasets and descriptors are summarized in Table 3.

Some classifiers assume the Euclidean feature space implicitly, as the ANN-MLP, and others have the distance function embedded in the model, as in the radial basis function (RBF) of the SVM. When the distance function

Dataset Code	Descriptor Code
B_1	D_1,D_2,D_3,D_4,D_5
B_2	D_6,D_7
B_3	D_8
B_4	D_9
B_5	D_9
B_6	D_9
B_7	D_9
B_8	D_{10}
B_9	D_{10}
B_{10}	D_{10}
B_{11}	D_{10}

Table 3: Datasets and the respective descriptors used in the experiments.

is L2, we use as RBF for SVM

$$K(s,t) = \exp^{-\gamma \|(\vec{v}(s) - \vec{v}(t))\|^2}, \tag{7}$$

where s and t are two samples (one is support vector) and $\vec{v}(s)$ and $\vec{v}(t)$ are their feature vectors. The constant γ is found by parameter optimization. In the case of special distances d, we have observed a considerable improvement in the SVM's performance when we replace its RBF by

$$K'(s,t) = \exp^{-\gamma d^2(s,t)}. \tag{8}$$

This can be observed in Tables 4 and 5 for dataset B_1 (MPEG-7) with D_4 (BAS with L2) and D_5 (BAS with OCS). Therefore, K' was used in all experiments involving SVM and special distance functions d, and K was used for L2.

The experiments evaluate the accuracy on Z_3 and the computational time of each classifier, OPF, SVM, ANN-MLP, and k-NN, for each pair dataset and descriptor presented in Table 3. In all experiments, the datasets were divided into three parts: a training set Z_1 with 30% of the samples, an evaluation set Z_2 with 20% of the samples, and a test set Z_3 with 50% of the samples. These samples were randomly selected and each experiment was repeated 10 times with different sets Z_1, Z_2 and Z_3 to compute mean (robustness) and standard deviation (precision) of the accuracy values and mean value of kappa [58]. Section 4.1 presents the accuracy results of training on Z_1 and testing on Z_3. The accuracy results of training on Z_1, with learning from the errors in Z_2, and testing on Z_3 are presented in Section 4.2. The average computational time of each classifier for training and classification is divided by the number of samples and reported in Section 4.3.

4.1 Accuracy results on Z_3 without using Z_2

The results in Table 4 are presented as $x \pm y(z)[k]$, where x, y, z and k are the mean accuracy, its standard deviation, mean kappa coefficient [58] and the best value of k obtained for k-NN, respectively. Values of kappa below 0.80 indicate the difficulty in classifying some datasets using the respective descriptors. Good descriptors tend to better separate the classes in the feature space, reducing overlap and so facilitating the classification. The results in B_1 (MPEG-7), for example, indicate that D_5 outperforms the remaining descriptors. Besides, D_4 and D_5 differ only in the distance function and the results indicate that OCS [39] outperforms the Euclidean metric. Similarly, one may conclude that D_6 (BIC [42]) outperforms D_7, (color histogram [57]) in B_2 (COREL). Irrespective of that, we are comparing the relative performance of the classifiers.

Most accuracies of OPF and SVM were clearly higher than those of ANN-MLP and k-NN. OPF and SVM presented equivalent overall performances, being one better than the other depending on the case. Considering only the cases where the best k is 1 in k-NN, we can observe that the criterion of OPF to assign the label of the most strongly connected root to a sample is really more accurate than the label of the closest sample. The instability of ANN-MLP is reflected by the standard deviations, which are about 10 times higher than the standard deviations obtained by the other classifiers. Due to the overlapping between classes (Figure 9), the accuracies of the classifiers in B_8 and B_9 are lower than their accuracies in B_{10} and B_{11}. Due to the quality of the descriptors, similar observation explains the increasing order of accuracy in B_1 with D_1, D_3, D_2, D_4 and D_5.

4.2 Accuracy results on Z_3 with learning on Z_2

In order to evaluate the ability of each classifier in learning from the errors in Z_2 without increasing the size of Z_1, we executed Algorithm 2 for $T = 3$ iterations. The results are presented in Table 5.

We can observe that the conclusions drawn from Table 4 remain the same with respect to the overall performance of the classifiers. In most cases, the general learning algorithm improved the performance of the classifiers with respect to their results in Table 4.

4.3 Efficiency results

Table 6 shows the mean execution time in seconds divided by the number of samples that each classifier takes for

Dataset (Descriptor)	Classifiers			
	OPF	SVM	ANN-MLP	k-NN
B_1 (D_1)	**71.71±0.01(0.49)**	70.07±0.01(0.40)	57.28±0.44(0.14)	59.38±0.01(0.17)[1]
B_1 (D_2)	79.48±0.01(0.59)	**82.15±0.01(0.64)**	71.48±0.26(0.46)	72.04±0.01(0.64)[1]
B_1 (D_3)	**75.95±0.01(0.51)**	74.49±0.01(0.50)	62.98±0.39(0.25)	60.16±0.01(0.19)[1]
B_1 (D_4)	**87.37±0.01(0.74)**	87.05±0.01(0.75)	77.99±0.34(0.57)	66.55±0.01(0.67)[1]
B_1 (D_5)	**95.72±0.01(0.89)**	94.92±0.01(0.88)	76.29±0.04(0.55)	92.14±0.01(0.89)[1]
B_2 (D_6)	86.74±0.01(0.75)	**90.65±0.01(0.83)**	83.07±0.10(0.64)	82.83±0.01(0.70)[1]
B_2 (D_7)	80.25±0.01(0.63)	**83.37±0.01(0.70)**	80.07±0.10(0.61)	78.03±0.01(0.61)[1]
B_3 (D_8)	**88.85±0.02(0.77)**	84.27±0.01(0.68)	86.97±0.21(0.73)	84.52±0.01(0.80)[1]
B_4 (D_9)	93.87±0.01(0.88)	**95.46±0.01(0.90)**	92.83±0.20(0.86)	91.85±0.01(0.81)[3]
B_5 (D_9)	**79.37±0.01(0.68)**	78.35±0.01(0.59)	73.35±0.10(0.68)	65.89±0.01(0.41)[2]
B_6 (D_9)	90.35±0.01(0.80)	**93.35±0.01(0.90)**	84.72±0.10(0.73)	87.20±0.01(0.79)[2]
B_7 (D_9)	90.53±0.01(0.81)	**93.86±0.01(0.88)**	92.94±0.09(0.85)	86.39±0.01(0.73)[1]
B_8 (D_{10})	**87.29±0.01(0.71)**	78.41±0.24(0.71)	85.33±0.02(0.69)	81.34±0.01(0.65)[7]
B_9 (D_{10})	**88.10±0.03(0.76)**	86.90±0.05(0.73)	83.60±0.54(0.67)	81.90±0.02(0.62)[1]
B_{10} (D_{10})	**1.0±0.0(1.0)**	**1.0±0.0(1.0)**	**1.0±0.0(1.0)**	**1.0±0.0(1.0)[21]**
B_{11} (D_{10})	96.76±0.01(0.93)	**99.55±0.01(0.99)**	97.20±0.36(0.94)	93.19±0.01(0.89)[1]

Table 4: Accuracy results $x \pm y(z)$ on Z_3 without using Z_2: x - mean accuracy, y - its standard deviation and z - mean kappa. The best accuracies are indicated in bold and the best value of k is shown in brackets for the k-NN.

Dataset (Descriptor)	Classifiers			
	OPF	SVM	ANN-MLP	k-NN
B_1 (D_1)	**75.94±0.01(0.51)**	74.42±0.01(0.48)	61.39±0.06(0.22)	59.38±0.01(0.17)[1]
B_1 (D_2)	81.20±0.01(0.60)	**82.03±0.01(0.64)**	75.06±0.28(0.50)	72.88±0.01(0.44)[3]
B_1 (D_3)	**76.72±0.01(0.53)**	76.52±0.01(0.53)	64.76±0.18(0.29)	61.48±0.01(0.26)[3]
B_1 (D_4)	**87.65±0.01(0.75)**	87.18±0.01(0.73)	79.23±0.22(0.58)	67.14±0.01(0.68)[1]
B_1 (D_5)	**96.08±0.01(0.90)**	95.87±0.01(0.90)	78.65±0.04(0.57)	90.70±0.01(0.90)[1]
B_2 (D_6)	86.90±0.01(0.76)	**90.80±0.02(0.84)**	85.70±0.06(0.75)	82.83±0.01(0.73)[1]
B_2 (D_7)	80.29±0.01(0.63)	83.66±0.01(0.71)	**84.70±0.08(0.79)**	79.73±0.01(0.61)[1]
B_3 (D_8)	88.54±0.02(0.77)	84.37±0.01(0.68)	**92.29±0.16(0.84)**	86.26±0.02(0.70)[1]
B_4 (D_9)	94.17±0.01(0.89)	**96.07±0.01(0.91)**	93.26±0.19(0.87)	91.26±0.01(0.81)[9]
B_5 (D_9)	**79.90±0.01(0.70)**	78.65±0.01(0.57)	74.35±0.10(0.70)	67.06±0.01(0.24)[2]
B_6 (D_9)	92.07±0.01(0.84)	**94.31±0.01(0.93)**	87.72±0.10(0.79)	89.54±0.01(0.81)[9]
B_7 (D_9)	91.53±0.01(0.83)	**94.00±0.01(0.88)**	93.73±0.06(0.87)	88.99±0.01(0.76)[1]
B_8 (D_{10})	**88.38±0.01(0.72)**	87.95±0.17(0.74)	81.58±0.01(0.70)	83.43±0.01(0.68)[11]
B_9 (D_{10})	**89.30±0.02(0.78)**	**89.30±0.03(0.78)**	88.10±0.43(0.76)	83.90±0.02(0.63)[1]
B_{10} (D_{10})	**1.0±0.0(1.0)**	**1.0±0.0(1.0)**	**1.0±0.0(1.0)**	**1.0±0.0(1.0)**[5]
B_{11} (D_{10})	97.35±0.02(0.94)	**99.85±0.01(0.99)**	98.23±0.26(0.96)	94.81±0.03(0.86)[1]

Table 5: Accuracy results $x \pm y(z)$ on Z_3 with learning on Z_2: x - mean accuracy, y - its standard deviation and z - mean kappa. The best accuracies are indicated in bold and the best value of k is shown in brackets for the k-NN.

training and classification (without learning on Z_2) and for each dataset and descriptor.

Dataset (Descriptor)	Classifiers			
	OPF	SVM	ANN-MLP	k-NN
B_1 (D_1)	**0.0052**	1.304	0.8973	0.0450
B_1 (D_2)	**0.0010**	0.0599	0.3453	0.0034
B_1 (D_3)	**0.0012**	0.0742	0.3603	0.0038
B_1 (D_4)	**0.0019**	0.2859	0.5043	0.0126
B_1 (D_5)	5.8400	7.3926	**0.5031**	5.8432
B_2 (D_6)	**0.0185**	0.1813	0.2553	0.0199
B_2 (D_7)	**0.0010**	0.0997	0.2491	0.0254
B_3 (D_8)	**0.2163**	0.2333	0.2891	0.2164
B_4 (D_9)	**0.0019**	0.0091	0.01505	0.0042
B_5 (D_9)	0.0018	0.0693	0.400	**0.0010**
B_6 (D_9)	**0.0012**	0.0320	0.0800	0.0017
B_7 (D_9)	**0.0011**	0.1662	3.5625	0.01960
B_8 (D_{10})	**0.0010**	0.1281	1.8540	0.0011
B_9 (D_{10})	0.0011	0.1445	0.3828	**0.0002**
B_{10} (D_{10})	0.0018	0.0078	0.0020	**0.0003**
B_{11} (D_{10})	**0.0019**	0.0594	0.0039	**0.0019**

Table 6: Mean execution times in seconds for training and classification divided by the number of samples. The best times are indicated in bold.

Note that OPF is extremely fast, except when it uses descriptor D_5 (Table 2) because of the respective distance function computation. Similar effect can be observed in SVM and k-NN. Given that ANN-MLP does not use distance functions, it is free of this problem. On average, the results indicate that our most recent implementation of OPF was about 72 times faster than the latest implementation of SVM [44], 443 times faster than the fast ANN-MLP [45], and 1.3 times faster than our implementation of a k-NN classifier.

The importance of speed in pattern recognition seems to not have caught much attention. Most of the computational time is spent in training, which is done only once in many applications. However, take the first case of B_1 and D_1, for example, where OPF spent 0.0052 second per sample and SVM spent 1.304. For 100,000 samples, this represents 8.67 minutes using OPF and 36.22 hours using SVM. In the case of 2D/3D images, for example, the number of pixels/voxels ranges from thousands to millions, and the time for training becomes a burden. In medical imaging, it is very likely that a new training has to be done for every 3D image, due to their variations in noise, inhomogeneity, and protocols.

Acknowledgments

The authors thank Paulo Miranda (IC-UNICAMP) and Nelson Mascarenhas (UFSCar) for their previous contributions and the financial support from CNPq and FAPESP.

5 Conclusions

We presented a discrete approach for supervised classification (OPF) which computes an optimum-path forest on a training set and classifies samples with the label of their most strongly connected root in the forest. We also proposed a general learning algorithm, which usually improves performance of the classifiers without increasing the training set. The source code of the supervised OPF is available in `www.ic.unicamp.br/~afalcao/libopf`.

We compared OPF with SVM, ANN-MLP, and k-NN using several datasets and descriptors. These experiments involved datasets with shape, color and texture properties, and datasets commonly used by the machine learning community. The advantage of OPF over the others in computational time is notorious and impressive, which is crucial in the case of large datasets. It can be more or less accurate than SVM, depending on the case, but its accuracy is usually superior to those of ANN-MLP and k-NN. OPF also presents some interesting properties. It is fast, simple, multi-class, parameter independent, does not make any assumption about the shape of the classes, and can handle some degree of overlapping between classes.

The OPF classifiers are being successfully used in some real applications: the supervised approach is being used for oropharyngeal dysphagia identification [37], laryngeal pathology detection [36], and diagnosis of parasites from optical microscopy images [38], and the unsupervised approach is being used for the separation of grey-matter and white-matter in Magnetic Resonance images of the brain [24]. In the first three applications, the supervised OPF outperforms SVM in accuracy and efficiency. In all cases, there is no human interaction, however, we also intend to evaluate the supervised OPF for interactive segmentation of brain tissues, where the user selects training markers. In this case the method becomes similar to an IFT-watershed approach, except for the fact that it works in the feature space with no spatial connectivity constraint, which is important for tissues with disconnected voxels.

Applications with large datasets definitely favor OPF with respect to SVM. We must say that, as a discrete approach, the performance of OPF may be reduced for small training sets, if the number of samples are not enough to represent the classes. In SVM, this may also be a problem, but as it estimates a decision hyper-plane, it has a chance to divide the feature space with separation between classes. Too much overlapping between classes may also represent an advantage for SVM with respect to OPF, because its transformation to a higher-dimensional space may separate the classes, solving the problem.

The OPF classifier is an important contribution for pattern recognition and related fields, which also opens new research problems. One can investigate the optimum-path forest classification using incomplete graphs (e.g., graphs where the arcs are between k-nearest neighbors), different connectivity functions, and other algorithms to estimate prototypes and to learn from the errors in the evaluation set. The use of Genetic Programming [59] (GP) for arc-weight estimation in OPF is also an alternative to deal with class overlapping, by combining the distances from multiple descriptors in a non-linear way [60, 61].

References

[1] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* Wiley-Interscience, 2 edition, 2000.

[2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 9th Annual Conference on Computational Learning Theory*, pages 92–100, New York, NY, USA, 1998. ACM Press.

[3] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[4] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2006.

[5] S. Haykin. *Neural networks: a comprehensive foundation.* Prentice Hall, 1994.

[6] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms.* Wiley-Interscience, 2004.

[7] L. Reyzin and R. E. Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the 23th International Conference on Machine learning*, pages 753–760, New York, NY, USA, 2006. ACM Press.

[8] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Workshop on Computational Learning Theory*, pages 144–152, New York, NY, USA, 1992. ACM Press.

[9] K. Duan and S. S. Keerthi. Which is the best multiclass svm method? an empirical study. *Multiple Classifier Systems*, pages 278–285, 2005.

[10] B. Tang and D. Mazzoni. Multiclass reduced-set support vector machines. In *Proceedings of the 23th International Conference on Machine learning*, pages 921–928, New York, NY, USA, 2006. ACM Press.

[11] N. Panda, E. Y. Chang, and G. Wu. Concept boundary detection for speeding up svms. In *Proceedings of the 23th International Conference on Machine learning*, pages 681–688, New York, NY, USA, 2006. ACM Press.

[12] R. Collobert and S. Bengio. Links between perceptrons, mlps and svms. In *Proceedings of the 21th International Conference on Machine learning*, page 23, New York, NY, USA, 2004. ACM Press.

[13] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[14] A.X. Falcão, J. Stolfi, and R.A. Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):19–29, Jan 2004.

[15] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* MIT, 1990.

[16] R.A. Lotufo and A.X. Falcão. The ordered queue and the optimality of the watershed approaches. In *Mathematical Morphology and its Applications to Image and Signal Processing (ISMM'00)*, volume 18, pages 341–350. Kluwer, Jun 2000.

[17] R. Audigier and R.A. Lotufo. Watershed by image foresting transform, tie-zone, and theoretical relationship with other watershed definitions. In *Mathematical Morphology and its Applications to Signal and Image Processing (ISMM)*, pages 277–288, Rio de Janeiro, RJ, Oct 2007. MCT/INPE.

[18] P. K. Saha and J. K. Udupa. Relative fuzzy connectedness among multiple objects: Theory, algorithms, and applications in image segmentation. *Computer Vision and Image Understanding*, 82(1):42–56, 2001.

[19] R. Audigier and R.A. Lotufo. Seed-relative segmentation robustness of watershed and fuzzy connectedness approaches. In *XX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, pages 61–68, Belo Horizonte, MG, Oct 2007. IEEE CPS.

[20] G.T. Herman and B.M. Carvalho. Multiseeded segmentation using fuzzy connectedness. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 23:460–474, May 2001.

[21] A. Rocha P.A.V. Miranda, A.X. Falcão and F.P.G. Bergo. Object delineation by κ-connected components. *EURASIP Journal on Advances in Signal Processing*, 2008. to appear.

[22] K. Fukunaga and P. M. Narendra. A branch and bound algorithms for computing k-nearest neighbors. *IEEE Transactions on Computers*, 24(7):750–753, 1975.

[23] L. M. Rocha, A. X. Falcão, and L. G. P. Meloni. A robust extension of the mean shift algorithm using optimum path forest. In *8th Intl. Workshop on Combinatorial Image Analysis*, pages 29–38, Buffalo-NY, USA, 2008. RPS (ISBN 978-981-08-0228-8).

[24] F.A.M. Cappabianco, A.X. Falcão, and L.M. Rocha. Clustering by optimum path forest and its application to automatic GM/WM classification in MR-T1 images of the brain. In *The Fifth IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*, pages 428–431, 2008.

[25] L. J. Hubert. Some applications of graph theory to clustering. *Psychometrika*, 39(3):283–309, 1974.

[26] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20(1):68–86, Jan. 1971.

[27] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[28] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000.

[29] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. Semi-supervised graph clustering: a kernel approach. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 457–464, New York, NY, USA, 2005. ACM.

[30] B. Schlkopf Zhou, D. and T. Hofmann. Semi-supervised learning on directed graphs. *Advances in Neural Information Processing Systems*, pages 1633–1640, 2005.

[31] J. Callut, K. Fançoisse, and M. Saerens. Semi-supervised classication in graphs using bounded random walks. In *Proceedings of the 17th Annual Machine Learning Conference of Belgium and the Netherlands (Benelearn)*, pages 67–68, 2008.

[32] Nimit Kumar and K. Kummamuru. Semisupervised clustering with metric learning using relative comparisons. *IEEE Transactions on Knowledge and Data Engineering*, 20(4):496–503, 2008.

[33] J.P. Papa, A.X. Falcão, C.T.N. Suzuki, and N.D.A. Mascarenhas. A discrete approach for supervised pattern recognition. In *12th International Workshop on Combinatorial Image Analysis*, volume 4958, pages 136–147. LNCS Springer Berlin/Heidelberg, 2008.

[34] J. P. Papa, A. X. Falcão, P. A. V. Miranda, C. T. N. Suzuki, and N. D. A. Mascarenhas. Design of robust pattern classifiers based on optimum-path forests. In *Mathematical Morphology and its Applications to Image and Signal Processing (ISMM'07)*, pages 337–348. MCT/INPE, 2007.

[35] J.A. Montoya-Zegarra, J.P. Papa, N.J. Leite, R.S. Torres, and A.X. Falcão. earning how to extract rotation-invariant and scale-invariant features from texture images. *EURASIP Journal on Advances in Signal Processing*, 2008:1–16, 2008.

[36] J.P. Papa, A.A. Spadotto, A.X. Falc ao, and J.C. Pereira. Optimum path forest classifier applied to laryngeal pathology detection. In *Proc. of the 15th Intl. Conf. on Systems, Signals, and Image Processing*, volume 1, pages 249–252, Bratislava, Slovakia, Jun 2008. IEEE.

[37] A.A. Spadotto, J.C. Pereira, R.C. Guido, J.P. Papa, A.X. Falc ao, A.R. Gatto, P.C. Cola, and A.O. Schelp. Oropharyngeal dysphagia identification using wavelets and optimum path forest. In *Proc. of the 3rd IEEE Intl. Symp. on Communications, Control and Signal Processing*, pages 735–740, St. Julians, Malta, Greece, Mar 2008.

[38] A.X. Falcão, C.T.N. Suzuki, J.F. Gomes, J.P. Papa, L.C.S. Dias, and S.H. Shimizu. A system for diagnosing intestinal parasites by computerized image analysis, Jun 2008. PCT WO/2008/064442.

[39] Y. P. Wang and T. Pavlidis. Optimal correspondence of string subsequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1080–1087, 1990.

[40] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the 6th International Conference on Computer Vision*, page 59, Washington, DC, USA, 1998. IEEE Computer Society.

[41] N. Arica and F. T. Y. Vural. BAS: A Perceptual Shape Descriptor Based on the Beam Angle Statistics. *Pattern Recognition Letters*, 24(9-10):1627–1639, June 2003.

[42] R. O. Stehling, M. A. Nascimento, and A. X. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 102–109, New York, NY, USA, 2002. ACM Press.

[43] C. Allène, J. Y. Audibert, M. Couprie, J. Cousty, and R. Keriven. Some links between min-cuts, optimal spanning forests and watersheds. In *Mathematical Morphology and its Applications to Image and Signal Processing*, pages 253–264. MCT/INPE, 2007.

[44] C. C. Chang and C. J. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at url http://www.csie.ntu.edu.tw/˜ cjlin/libsvm.

[45] S. Nissen. *Implementation of a Fast Artificial Neural Network Library (FANN)*, 2003. Department of Computer Science University of Copenhagen (DIKU). Software available at *http://leenissen.dk/fann/*.

[46] S. C. Huang and Y. F. Huang. Bounds on the number of hidden neurons in multilayer perceptrons. *IEEE Transactions on Neural Networks*, 2(1):47–55, 1991.

[47] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.

[48] MPEG-7. Mpeg-7: The generic multimedia content description standard, part 1. *IEEE MultiMedia*, 09(2):78–87, 2002.

[49] Corel Corporation. Corel stock photo images, -. http://www.corel.com.

[50] P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover, New York, 1966.

[51] D.J. Newman A. Asuncion. UCI machine learning repository, 2007.

[52] D. Collins, A. Zijdenbos, V. Kollokian, J. Sled, N. Kabani, C. Holmes, and A. Evans. Design and construction of a realistic digital brain phantom. *IEEE Transactions on Medical Imaging*, 17(3):463–468, 1998.

[53] L. Kuncheva. Artificial data. *School of Informatics, University of Wales, Bangor*, 1996. http://www.informatics.bangor.ac.uk/˜kuncheva.

[54] E. Persoon and K. Fu. Shape Discrimination Using Fourier Descriptors. *IEEE Transanctions on Systems, Man, and Cybernetics*, 7(3):170– 178, 1977.

[55] M.K. Hu. Visual Pattern Recognition by Moment Invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.

[56] R. Torres, A. X. Falcão, and L.F. Costa. A graph-based approach for multiscale shape analysis. *Pattern Recognition*, 37(6):1163–1174, 2004.

[57] M. Swain and D. Ballard. Color Indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[58] J. Cohen. A coefficient of agreement for nominal scales. In *Educational and Psychological Measurement*, volume 20, pages 37–46, 1960.

[59] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. The MIT Press, 1992.

[60] R.S. Torres, A.X. Falcão, M.A. Gonçalves, B. Zhang, W. Fan, E. A. Fox, and P. Calado. A new framework to combine descriptors for content-based image retrieval. In *ACM 14th Conference on Information and Knowledge Management*, pages 335–336, Bremen, Germany, Nov 2005.

[61] R.S. Torres, A.X. Falcão, M.A. Gonalves, J.P. Papa, B. Zhang, W. Fan, and E.A. Fox. A genetic programming framework for content-based image retrieval. *Pattern Recognition*, 2008. (accepted).