# Efficient Supervised Optimum-Path Forest Classification for Large Datasets

João P. Papa[a], Alexandre X. Falcão[b], Victor Hugo C. de Albuquerque[c], João Manuel R. S. Tavares[d]

[a] *UNESP - Univ Estadual Paulista, Computing Department, Bauru, Brazil.*
*Email: papa@fc.unesp.br*
[b] *University of Campinas, Computing Institute, Campinas, Brazil.*
*Email: afalcao@ic.unicamp.br*
[c] *University of Fortaleza, Technological Research Center, Fortaleza, Brazil.*
*Email: victor.albuquerque@fe.up.pt*
[d] *University of Porto, Engineering Faculty, Porto, Portugal.*
*Email: tavares@fe.up.pt*

**Abstract**

Data acquisition technologies can provide large datasets with millions of samples for statistical analysis. This creates a tremendous challenge for pattern recognition techniques, which need to be more efficient without loosing their effectiveness. We have tried to circumvent the problem by reducing it into the fast computation of an *optimum-path forest* (OPF) in a graph derived from the training samples. In this forest, each class may be represented by multiple trees rooted at some representative samples. The forest is a classifier which assigns to any new sample the label of its most strongly connected root. This methodology has been successful with different graph topologies and learning techniques. In this work we have focused on one of the supervised approaches, which has offered considerable advantages over Support Vector Machines and Artificial Neural Networks to handle large datasets. We propose (i) a new algorithm that speeds up classification and (ii) a solution to reduce the training set size with negligible effects on the accuracy of classification, further increasing its efficiency. Experimental results show the improvements with respect to our previous approach and advantages over other existing methods, which make the new method a valuable contribution for large dataset analysis.

*Keywords:* Optimum-Path Forest Classifiers, Support Vector Machines, Artificial Neural Networks, Pattern Recognition, Machine Learning

## 1. Introduction

New data acquisition technologies can provide large datasets for statistical analysis. Especially imaging devices, such as digital cameras, multispectral remote sensors and tomographic scanners, create images with millions of pixels for classification (segmentation). Large image collections are also available for content-based image retrieval. In both cases, image and pixel classification may also require user feedback, retraining, and interactive time response during some iterations [1, 2, 3].

However, popular approaches, such as Support Vector Machines [4] (SVMs) and Artificial Neural Networks [5, 6] (ANNs), present a prohibitive computational time for large datasets, especially in the training phase. Although there are efforts, such as LASVM [7] and SVMs without kernel mapping [8], to speed up SVMs, for instance, the former is limited to binary classification and the latter considerably reduces the accuracy of classification in the case of overlapped classes. Therefore, it seems paramount to develop more efficient and effective pattern recognition methods for large datasets.

In order to circumvent this problem, we applied a methodology to develop pattern classifiers based on the fast computation of an *optimum-path forest* (OPF) in a graph derived from the training samples. This provided both effective supervised [9, 10] and unsupervised [11] learning techniques for training sets of reasonable sizes (thousands of samples). The training samples are interpreted as the nodes of a graph whose arcs are defined by an *adjacency relation*. Any path in the graph has a value given by a *connectivity function*. The maximization (minimization) of a connectivity map results in an optimum-path forest rooted at representative samples of classes/clusters. This methodology only assumes that two samples in a same cluster/class should be at least *connected* by a chain of nearby samples (transitive property). Classes/clusters may present arbitrary shapes and some degree of overlapping, and there is no need to use parametric models. The class/cluster label assignment to new samples is also efficiently performed based on a local processing of the forest's attributes and the distances between the new sample and some training nodes. By changing the adjacency relation, connectivity function and learning technique, one can derive different pattern classifiers. In this work, we focus on the supervised classification method proposed in [9], because it has been the most successfully used for different

applications [12, 13, 14, 15, 16, 17, 18, 19]. Here forth, we will refer to this approach as the OPF classifier for sake of simplicity.

The OPF classifier [9] interprets the training set as a complete graph, weighted by the distance between nodes in a given feature space, and assigns to any path the maximum arc weight along it. It does not require optimization of any parameter and its training phase can be considerably faster (from tens to thousands) than the training phases of SVMs and ANNs, with accuracies better than or equivalent to the ones obtained with these approaches. However, for a small number of support vectors, it might take more time for classification than SVMs [20]. In order to speed up its classification time, we propose here two improvements: (i) a faster classification algorithm, which avoids to visit all training nodes during classification, and (ii) a supervised *learning-with-pruning* approach, which can considerably reduce the training set size, providing faster classification with negligible loss in accuracy. The contribution in (i) was recently accepted for conference publication [21]. Here, we validate the new algorithm using more datasets and discuss its worst-case time complexity. We are also combining the new algorithm with the approach in (ii) to further speed up classification. The learning-with-pruning algorithm is also an improvement with respect to our previous work [12]. It shows better accuracy, robustness, and allows the user to specify a maximum loss in accuracy as stopping criterion for pruning. The experiments demonstrate the improvements of this enhanced OPF classifier (EOPF) with respect to our previous approach and its advantages over SVMs and ANNs. For large datasets, the results represent a significant contribution.

Section 2 provides an overview of the OPF methodology for pattern recognition and of its main contributions. Sections 3 and 4 present, respectively, the new methods (i)–(ii) above and their experimental results. Section 5 states conclusion.

## 2. Pattern recognition by optimum-path forest

Given a training set with samples from distinct classes, we wish to design a pattern classifier which can assign the true class label to any new sample. Each sample is represented by a set of features and a distance function measures their dissimilarity in the feature space. The training samples are then interpreted as the nodes of a graph, whose arcs are defined by a given adjacency relation and weighted by the distance function. It is expected that

samples from a same class/cluster are connected by a path of nearby samples. Therefore, the degree of connectedness for any given path is measured by a connectivity (path-value) function, which exploits the distances along the path. In supervised learning, the true label of the training samples is known and so it is exploited to identify key samples (prototypes) in each class. Optimum paths are computed from the prototypes to each training sample, such that each prototype becomes root of an optimum-path tree composed by its most strongly connected samples. The labels of these samples are assumed to be the same of their root. In unsupervised learning, each cluster is represented by an optimum-path tree rooted at a single prototype but we do not know the class label of the training samples. Therefore, we expect that each cluster contains only samples of a same class and some other information about the application is needed to complete classification. The basic idea is then to specify an adjacency relation and a path-value function, compute prototypes and reduce the problem into an optimum-path forest computation in the underlying graph. The training forest becomes a classifier which can assign to any new sample the label of its most strongly connected root. Essentially, this methodology extends a previous approach, called *Image Foresting Transform*[22], for the design of image processing operators from the image domain to the feature space.

In [9], we presented a first method for supervised classification using a complete graph (implicit representation) and the maximum arc weight along a path as connectivity function. The prototypes were chosen as samples that share an arc between distinct classes in a minimum spanning tree of the training set [23]. This OPF classifier has been widely used in several applications, such as remote sensing [12], pathology detection by means of biomedical signal recognition [13], emotion recognition through speech processing [14], automatic vowel classification [15], biometrics [16, 17], petroleum well drilling monitoring [18], medical image segmentation [24], and robust object tracking [19]. In the present paper, we propose considerable improvements to make this OPF classifier efficient for large datasets.

Another supervised learning method was proposed in [10]. In this case, the arcs connect $k$-nearest neighbors ($k$-nn) in the feature space. The distances between adjacent nodes are used to estimate a probability density value of each node and optimum paths are computed from the maxima of this probability density function (pdf). For large datasets, we usually use a smaller training set and a much larger evaluation set to learn the most representative samples from the classification errors in the evaluation set. This

considerably improves classification accuracy of new samples. This strategy was assessed with $k$-nn graphs in [25]. The accuracy results can be better than using similar strategy with complete graph [9] for some situations, but the latter is still preferred because it is faster and does not require the optimization of the parameter $k$.

For unsupervised learning, the $k$-nn graph has presented excellent results for medical image segmentation by computing clusters as optimum-path trees rooted at the maxima of the pdf [11, 26]. In this work, however, our focus is on improving efficiency of the supervised OPF classifier based on complete graph and maximum arc weight function. Given that it may take more time to classify a large testing set than SVMs [20], depending on the number of support vectors, we have proposed strategies to reduce the training set size [12] and increase efficiency of classification [21]. As explained in Section 1, we are now revisiting, combining, and extending both previous works with a new learning-with-pruning algorithm. The new method is called *Enhanced OPF Classifier* (EOPF).

## 3. The Enhanced OPF Classifier (EOPF)

We present here two ideas to speed up the supervised OPF classifier with complete graph and maximum arc weight function: (i) a new classification algorithm (Section 3.2), which avoids to visit all training nodes during classification, and (ii) a supervised *learning-with-pruning* approach (Section 3.3), which can considerably reduce the training set size, providing faster classification with negligible loss in accuracy.

### *3.1. Training the EOPF classifier*

In large datasets, the number of labeled samples for training is usually large. Therefore, a first strategy to make a classifier more efficient is the use of two labeled and disjoint sets, $Z_1$ and $Z_2$, $|Z_1| \ll |Z_2|$, being the first the actual training set and the second an evaluation set. The purpose of the evaluation set is to improve the quality of the samples in the training set, without increasing its size, by replacing classification errors in $Z_2$ by non-prototype samples of $Z_1$ [9]. After this learning process, the classifier is ready to be tested on any unseen dataset $Z_3$. For validation, this process must also be repeated several times, with different random and disjoint sets $Z_1$, $Z_2$, and $Z_3$, in order to obtain the average accuracy results. The training

of the EOPF classifier is very similar to the one used for the OPF classifier [9], except for an important detail that will be explained in Algorithm 1.

Let $(Z_1, A)$ be a complete graph whose nodes are the samples in $Z_1$ and any pair of samples defines an arc in $A = Z_1 \times Z_1$. The arcs do not need to be stored and so the graph representation is implicit. A path is a sequence of distinct samples $\pi_t = \langle s_1, s_2, \ldots, s_{k-1}, t\rangle$ with terminus $t$, where $(s_i, s_{i+1}) \in A$ for $1 \leq i \leq k-1$. A path is said *trivial* if $\pi_t = \langle t\rangle$. We assign to each path $\pi_t$ a cost $f(\pi_t)$ given by a path-value function $f$. A path $\pi_t$ is considered optimum if $f(\pi_t) \leq f(\tau_t)$ for any other path $\tau_t$ with the same terminus $t$. We also denote by $\pi_s \cdot \langle s, t\rangle$ the concatenation of a path $\pi_s$ and arc $(s, t)$.

Training essentially consists of finding an optimum-path forest in $(Z_1, A)$, which is rooted in a special set $S \subset Z_1$ of prototypes. As proposed in [9], the set $S$ is represented by samples that share arcs between distinct classes in a minimum-spanning tree (MST) of $(Z_1, A)$ [23]. For path-value function $f_{\max}$, these prototypes (roots of the forest) tend to minimize the classification errors in $Z_1$, when their labels are propagated to the nodes of their trees.

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

such that $f_{max}(\pi_s)$ computes the maximum distance between adjacent samples in a non-trivial path $\pi_s$.

The training algorithm for the OPF classifier [9] assigns one optimum path $P_1^*(s)$ from $S$ to every sample $s \in Z_1$, forming an optimum path forest $P_1$ (a function with no cycles which assigns to each $s \in Z_1 \backslash S$ its predecessor $P_1(s)$ in $P_1^*(s)$ or a marker $nil$ when $s \in S$). Let $R_1(s) \in S$ be the root of $P_1^*(s)$ (which can be reached from $P_1(s)$), the OPF algorithm computes for each $s \in Z_1$, the minimum cost $C_1(s)$ of $P_1^*(s)$, the class label $L_1(s) = \lambda(R_1(s))$, and the predecessor $P_1(s)$.

An important difference here is that the training algorithm for the EOPF classifier also outputs a new set $Z_1'$ with all training nodes in a non-decreasing order of optimum cost. As we will see in the next section, this information can be exploited to speed up classification. Algorithm 1 implements this training procedure.

**Algorithm 1.** – Training Algorithm for the EOPF Classifier

| | |
|---|---|
| Input: | A $\lambda$-labeled training set $Z_1$ and the pair $(v, d)$ for feature vector and distance computations. |
| Output: | Optimum-path forest $P_1$, cost map $C_1$, label map $L_1$, and ordered set $Z_1'$. |
| Auxiliary: | Priority queue $Q$, set $S$ of prototypes, and cost variable *cst*. |

1. *Set* $Z_1' \leftarrow \emptyset$ *and compute by MST the prototype set* $S \subset Z_1$.
2. *For each* $s \in Z_1 \backslash S$, *set* $C_1(s) \leftarrow +\infty$.
3. *For each* $s \in S$, *do*
4. $\quad$ $C_1(s) \leftarrow 0$, $P_1(s) \leftarrow nil$, $L_1(s) \leftarrow \lambda(s)$, *and insert* $s$ *in* $Q$.
5. *While* $Q$ *is not empty, do*
6. $\quad$ *Remove from* $Q$ *a sample* $s$ *such that* $C_1(s)$ *is minimum.*
7. $\quad$ *Insert* $s$ *in* $Z_1'$.
8. $\quad$ *For each* $t \in Z_1$ *such that* $t \neq s$ *and* $C_1(t) > C_1(s)$, *do*
9. $\quad\quad$ *Compute* $cst \leftarrow \max\{C_1(s), d(s,t)\}$.
10. $\quad\quad$ *If* $cst < C_1(t)$, *then*
11. $\quad\quad\quad$ *If* $C_1(t) \neq +\infty$, *then remove* $t$ *from* $Q$.
12. $\quad\quad\quad$ $P_1(t) \leftarrow s$, $L_1(t) \leftarrow L_1(s)$, $C_1(t) \leftarrow cst$.
13. $\quad\quad\quad$ *Insert* $t$ *in* $Q$.
14. *Return a classifier* $[P_1, C_1, L_1, Z_1']$.

The time complexity for training is $O(|Z_1|^2)$, due to the main (Lines 5-13) and inner loops (Lines 8-13) in Algorithm 1, which run $O(|Z_1|)$ times each.

### *3.2. EOPF Classification*

In [9], the classification of each new sample $t \in Z_2$ (or $Z_3$) is done based on the distance $d(s,t)$ between $t$ and each training node $s \in Z_1$ and on the evaluation of the following equation.

$$C_2(t) = \min\{\max\{C_1(s), d(s,t)\}\}, \ \forall s \in Z_1. \tag{2}$$

Let $s^* \in Z_1'$ be the node $s$ that satisfies this equation. It essentially considers all possible paths $\pi_s$ from $S$ in $(Z_1, A)$ extended to $t$ by an arc $(s,t)$, finds the optimum path $P_1^*(s^*) \cdot \langle s^*, t \rangle$, and label $t$ with the class $\lambda(R_1(s^*))$ of its most strongly connected prototype $R_1(s^*) \in S$ (i.e., $L_2(t) \leftarrow L_1(s^*) = \lambda(R_1(s^*))$).

Note that $Z_1$ can be replaced by $Z_1'$ in Equation 2 and its evaluation can halt when $\max\{C_1(s), d(s,t)\} < C_1(s')$ for a node $s'$ whose position in $Z_1'$ succeeds the position of $s$. This avoids to visit all nodes in $Z_1'$ in many

cases and the efficiency gain increases with the time complexity of $d(s,t)$. Algorithm 2 implements this faster classification procedure.

**Algorithm 2.** – EOPF CLASSIFICATION

| | |
|---|---|
| INPUT: | Classifier $[P_1, C_1, L_1, Z_1']$, evaluation set $Z_2$ (or test set $Z_3$), and the pair $(v, d)$ for feature vector and distance computations. |
| OUTPUT: | Label $L_2$ and predecessor $P_2$ maps defined for $Z_2$. |
| AUXILIARY: | Cost variables *tmp* and *mincost*. |

1. *For each* $t \in Z_2$, *do*
2. $\quad i \leftarrow 1$, $mincost \leftarrow max\{C_1(k_i), d(k_i, t)\}$.
3. $\quad L_2(t) \leftarrow L_1(k_i)$ *and* $P_2(t) \leftarrow k_i$.
4. $\quad$ *While* $i < |Z_1'|$ *and* $mincost > C_1(k_{i+1})$, *do*
5. $\quad\quad$ *Compute* $tmp \leftarrow \max\{C_1(k_{i+1}, d(k_{i+1}, t)\}$.
6. $\quad\quad$ *If* $tmp < mincost$, *then*
7. $\quad\quad\quad mincost \leftarrow tmp$.
8. $\quad\quad\quad L_2(t) \leftarrow L(k_{i+1})$ *and* $P_2(t) \leftarrow k_{i+1}$.
9. $\quad\quad i \leftarrow i + 1$.
10. *Return* $[L_2, P_2]$.

In Algorithm 2, the main loop (Lines $1-9$) performs classification of all nodes in $Z_2$. The inner loop (Lines $4-9$) visits each node $k_{i+1} \in Z_1'$, $i = 1, 2, \ldots, |Z_1'|$ until an optimum path $\pi_{k_{i+1}} \cdot \langle k_{i+1}, t\rangle$ be found. In the worst case, it visits all nodes in $Z_1'$ (Line 4). Line 5 evaluates $f_{max}(\pi_{k_{i+1}} \cdot \langle k_{i+1}, t\rangle)$ and Lines $7-8$ updates cost, label and predecessor of $t$ whenever $\pi_{k_{i+1}} \cdot \langle k_{i+1}, t\rangle$ is better than the current path $\pi_t$ (Line 6). Figure 1 illustrates this process.

In the worst case, when unbalanced classes present elongated shapes for instance, the time complexity of Algorithm 2 is the same $O(|Z_1'||Z_2|)$ of the previous one. However, in practice, its time complexity is $O(p|Z_2|)$, for some $p \ll |Z_1'|$. Figure 2a illustrates a bad case using two unbalanced classes of white and black nodes, in which four prototypes will be elected as nodes that share an arc between distinct classes in the MST of the complete graph (bounded nodes in Figure 2b). An optimum-path forest from these prototypes is shown in Figure 2c. Note that, almost all training nodes will be visited in $Z_1'$ in order to find the training node $s^*$ that reaches a new sample $t \in Z_2$ with minimum cost for classification, as shown in Figure 2d, because $C_1(s^*)$ is high due to that long arc before its predecessor node and the access of $t$ through other nodes would be even more expensive.

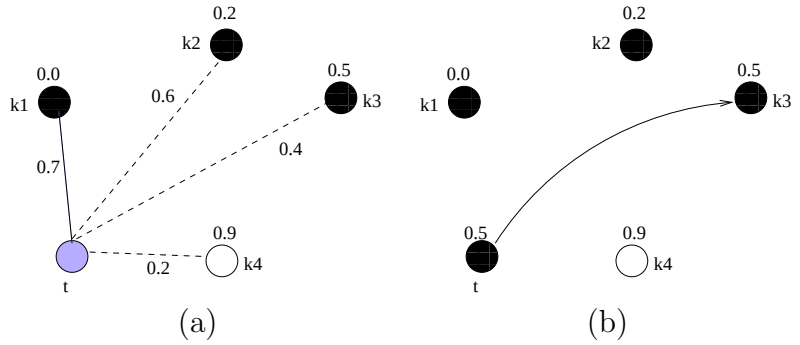



Figure 1: (a) Example graph with two classes represented by black and white nodes. The values over the nodes indicate their minimum costs after training. A new gray sample $t$ is assumed connected to all training nodes. The previous algorithm would visit all nodes $k_i \in Z_1'$, $i = 1, 2, 3, 4$, to classify $t$. The new algorithm halts when the cost value of the next node in $Z_1'$ is greater than or equal to the minimum cost already offered to $t$. (b) The classification process of sample $t$ is then terminated at node $k_3$, because $C_1(k_4) = 0.9 > 0.5\}$ and so $L_2(t) \leftarrow L_1(k_3) = black$.

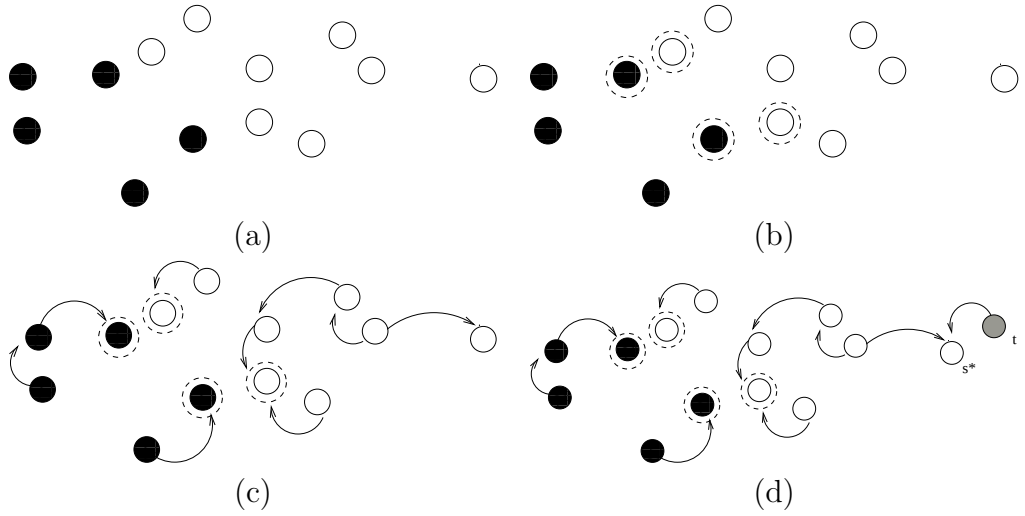



Figure 2: Example of a bad case for the EOPF classifier: (a) training set where classes present elongated shapes, (b) prototypes (bounded nodes) obtained from the MST, (c) optimum-path forest generated from the prototypes, (d) classification of a new sample $t \in Z_2$ through its best predecessor node $s^* \in Z_1'$.

### *3.3. Pruning of irrelevant patterns*

Large datasets usually present redundancy, so at least in theory it should be possible to estimate a reduced training set with the most relevant patterns

for classification. The use of a training set $Z_1$ and an evaluation set $Z_2$ has allowed us to learn relevant samples for $Z_1$ from the classification errors in $Z_2$, by swapping misclassified samples of $Z_2$ and non-prototype samples of $Z_1$ during a few iterations [9]. In this learning strategy, $Z_1$ remains with the same size and the classifier instance with the highest accuracy is selected to be tested in the unseen set $Z_3$. In this section, we use this learning procedure (as described in Algorithm 3) within a new method (Algorithm 4) to reduce the training set size by identifying and eliminating irrelevant samples from $Z_1$.

**Algorithm 3.** – OPF Learning Algorithm

| | |
|---|---|
| Input: | A $\lambda$-labeled training and evaluating sets $Z_1$ and $Z_2$, respectively, number $T$ of iterations, and the pair $(v, d)$ for feature vector and distance computations. |
| Output: | Optimum-path forest $P_1$, cost map $C_1$, label map $L_1$, and ordered set $Z'_1$. |
| Auxiliary: | Arrays $FP$ and $FN$ of sizes $c$ for false positives and false negatives, set $S$ of prototypes, and list $LM$ of misclassified samples. |

1. *Set* $MaxAcc \leftarrow -1$.
2. *For each iteration* $I = 1, 2, \ldots, T$, *do*
3. $\quad LM \leftarrow \emptyset$ *and compute the set* $S \subset Z_1$ *of prototypes.*
4. $\quad [P_1, C_1, L_1, Z'_1] \leftarrow$ *Algorithm1*$(Z_1, S, (v, d))$.
5. $\quad$ *For each class* $i = 1, 2, \ldots, c$, *do*
6. $\quad\quad FP(i) \leftarrow 0$ *and* $FN(i) \leftarrow 0$.
7. $\quad [L_2, P_2] \leftarrow$ *Algorithm2*$(Z'_1, Z_2, (v, d))$
8. $\quad$ *For each sample* $t \in Z_2$, *do*
9. $\quad\quad$ *If* $L_2(t) \neq \lambda(t)$, *then*
10. $\quad\quad\quad FP(L_2(t)) \leftarrow FP(L_2(t)) + 1$.
11. $\quad\quad\quad FN(\lambda(t)) \leftarrow FN(\lambda(t)) + 1$.
12. $\quad\quad\quad LM \leftarrow LM \cup t$.
13. $\quad$ *Compute accuracy Acc according to [9].*
14. $\quad$ *If* $Acc > MaxAcc$ *then save the current instance* $[P_1, C_1, L_1, Z'_1]$
15. $\quad$ *of the classifier and set* $MaxAcc \leftarrow Acc$.
16. $\quad$ *While* $LM \neq \emptyset$
17. $\quad\quad LM \leftarrow LM \backslash t$.
18. $\quad\quad$ *Replace* $t$ *by a non-prototype sample, randomly selected from* $Z_1$.
19. *Return the classifier instance* $[P_1, C_1, L_1, Z'_1]$ *with the highest accuracy in* $Z_2$.

The efficacy of Algorithm 3 increases with the size of $Z_1$, because more non-prototype samples can be swapped by misclassified samples of $Z_2$. However, for sake of efficiency, we need to choose some reasonable maximum size for $Z_1$. After learning the best training samples for $Z_1$, we may also mark paths in $P_1$ used to classify samples in $Z_2$ and define their nodes as *relevant samples* in a set $\mathcal{R}$. The "irrelevant" training samples in $Z_1 \backslash \mathcal{R}$ can then be moved to $Z_2$. Algorithm 4 applies this idea repetitively, while the loss in accuracy on $Z_2$ with respect to the highest accuracy obtained by Algorithm 3 (using the initial training set size) is less or equal to a maximum value $MLoss$ specified by the user.

**Algorithm 4.** – Learning-with-Pruning Algorithm

| | |
|---|---|
| Input: | Training and evaluation sets, $Z_1$ and $Z_2$, labeled by $\lambda$, the pair $(v, d)$ for feature vector and distance computations, maximum loss $MLoss$ in accuracy on $Z_2$, and number $T$ of iterations. |
| Output: | EOPF classifier $[P_1, C_1, L_1, Z_1']$ with reduced training set. |
| Auxiliary: | Set $\mathcal{R}$ of relevant samples, and variables $Acc$ and $tmp$. |

1. $[P_1, C_1, L_1, Z_1'] \leftarrow$ *Algorithm3*$(Z_1, Z_2, T, (v, d))$.
2. $[L_2, P_2] \leftarrow$ *Algorithm2*$(Z_1', Z_2, (v, d))$ *and store accuracy in* $Acc$.
3. $tmp \leftarrow Acc$ *and* $\mathcal{R} \leftarrow \emptyset$.
4. *While* $|Acc - tmp| \leq MLoss$ *and* $\mathcal{R} \neq Z_1$ *do*
5. $\quad \mathcal{R} \leftarrow \emptyset$.
6. $\quad$ *For each sample* $t \in Z_2$, *do*
7. $\quad\quad s \leftarrow P_2(t) \in Z_1$.
8. $\quad\quad$ *While* $s \neq nil$, *do*
9. $\quad\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup s$.
10. $\quad\quad\quad s \leftarrow P_1(s)$.
11. $\quad$ *Move samples from* $Z_1 \backslash \mathcal{R}$ *to* $Z_2$.
12. $\quad [P_1, C_1, L_1, Z_1'] \leftarrow$ *Algorithm3*$(Z_1, Z_2, T, (v, d))$.
13. $\quad [L_2, P_2] \leftarrow$ *Algorithm2*$(Z_1', Z_2, (v, d))$ *and store accuracy in* $tmp$.
14. *Return* $[P_1, C_1, L_1, Z_1']$.

Lines $1-3$ compute learning and classification using the highest accuracy classifier obtained for an initial training set size. Its accuracy is stored in $Acc$ and used as reference value $Acc$ in order to stop the pruning process, when the loss in accuracy is greater than an user-specified value $MLoss$ or all training samples are considered relevant. The main loop in Lines

$4 - 13$ essentially marks the relevant samples in $Z_1$ by following backwards the optimum paths used for classification (Lines $5 - 10$), moves irrelevant samples to $Z_2$, and repeats learning and classification from a reduced training set until it reaches the above stopping criterion. This process is actually different from and better than the one previously proposed method in [12]. The previous method is good when there is a lot of redundancy in $Z_1$, but it does not provide any mechanism to avoid a significant loss in accuracy. The present algorithm consistently provides higher accuracy than the previous one independently of the redundancy degree in $Z_1$.

## 4. Experimental results

We first demonstrate the need for more efficient classifiers with minimum loss in accuracy (Section 4.1) by comparing the previous OPF with other popular approaches. The efficiency gain of the EOPF classifier over the previous OPF is evaluated in Section 4.2 using several datasets from different applications. Finally, we extend this evaluation in Section 4.3 by considering EOPF with and without the proposed pruning approach.

### *4.1. Why do we need more efficient classifiers?*

Popular methods, such as Support Vector Machines (SVM) and Artificial Neural Networks (ANN), present high computational cost for training, being impractical in the case of training sets with thousands of samples or in applications that require multiple retraining with interactive response times (e.g., interactive segmentation [1, 2]). In the case of redundant training sets, it is still possible to reduce them in order to improve efficiency of these approaches. However, reduced training sets usually affect the efficacy of them in a significant way.

By choosing a small dataset, the MPEG-7 Shape Database Part B [27], which contains only 1,400 shape samples and 70 classes, it is already possible to understand the shortcomings of the popular approaches with respect to the EOPF method. For this database, the BAS (Beam Angle Statistics) shape descriptor is known as one of the best to represent and distinguish its classes [28]. We then compared the performance of the following classifiers in this database using the BAS descriptor with 180 features. The previous OPF classifier using the C code available in [29], SVM-RBF (with Radial Basis Function) using LibSVM [30], SVM no-kernel (without kernel mapping) using LibLINEAR [8], ANN-MLP (Multilayer Perceptrons) using the FANN

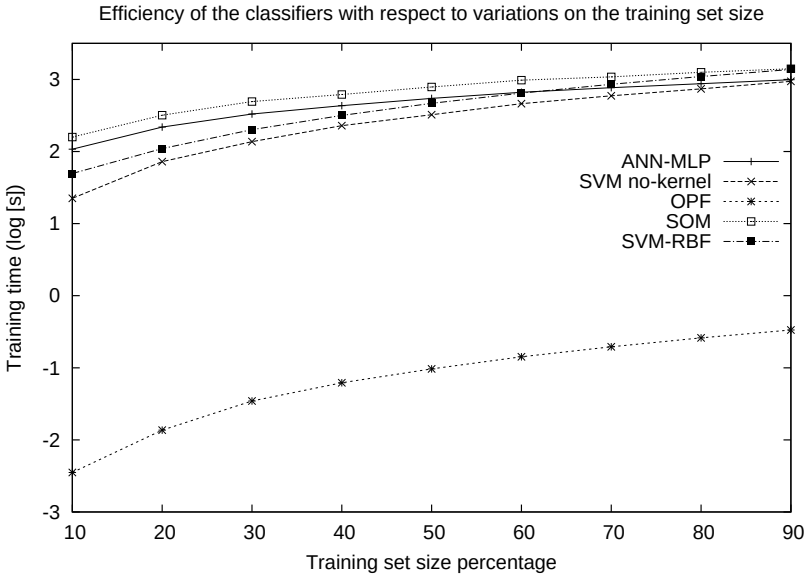
Library [31], and our own implementation of SOM (Kohonen Self Organizing Maps).

For both, SVM-RBF and SVM no-kernel, we applied cross validation for parameter optimization. In ANN-MLP, we used the following empirically chosen architecture: 180:8:8:70, i.e., 180 neurons in the input layer, two hidden layers with 8 neurons on each one and 70 neurons on the output layer. For SOM, we used a neural lattice with 100×100 neurons with 10 learning iterations.

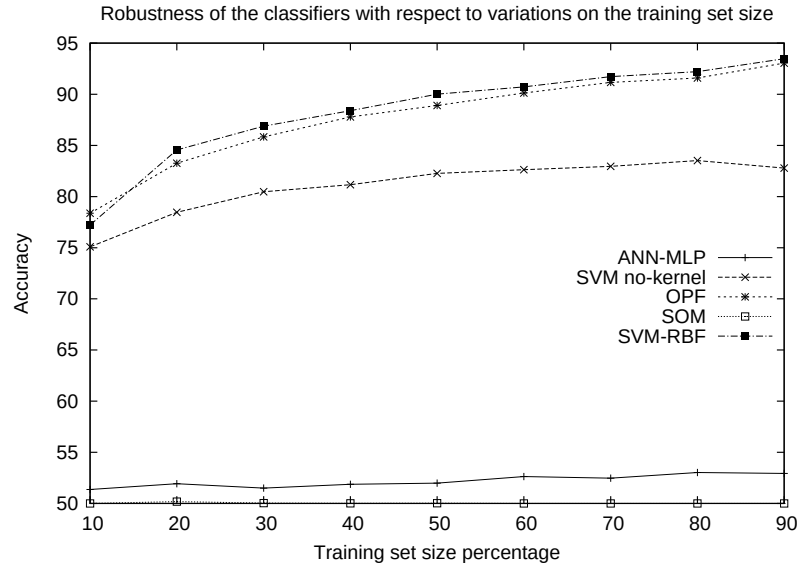
The experiments were executed 10 times with randomly generated training $Z_1$ and test $Z_3$ sets for each classifier and for different training set sizes. Figure 3 displays the average results of training time (using the logarithmic scale in seconds) and accuracy with different training set sizes. These experiments were performed using a PC with Intel® Core I5 processor and 4Gb RAM. Note that OPF is much faster than all other classifiers (Figure 3a). Using 90% of the samples for training, OPF was 4,107.48 times faster than SVM-RBF, for instance. This becomes really relevant in the case of large training sets. Besides its advantage in efficiency for training, the accuracy of OPF is also equivalent to the one of SVM-RBF, being both the most effective classifiers (Figure 3b).

One may argue that reduced training sets would be enough to increase efficiency of the SVM approaches, for example, with negligible loss in accuracy. Table 1 shows that this is not possible in the case of this database. Mean accuracy ($x$) on the test set and training time ($y$ in seconds) are presented using the $x : y$ format for training sets with 10%, 5%, 2.5%, and 0.8% of the data samples. SVM-RBF and SVM no-kernel considerably loose accuracy with respect to their initial values using 10% of the samples. OPF obtains the highest accuracy with 10% of the samples, being still 10,000 faster than SVM-RBF and 3,000 times faster than SVM no-kernel when they use only 0.8% of the samples for training.

Hence, it should be clear that OPF can achieve higher accuracy and efficiency using much larger training sets than those that are feasible for the SVM approaches. Now it is important to show that EOPF improves the classification time with respect to OPF and the pruning of training samples is possible to increase efficiency of the EOPF classification with negligible loss in accuracy.




(a)




(b)

Figure 3: Results for increasing training set sizes: (a) Mean training time in seconds using the logarithmic scale for OPF, ANN-MLP, SVM-RBF, SVM no-kernel and SOM. (b) Mean accuracy of the classifiers on a test set.

| Training set percentage | OPF | SVM-RBF | SVM no-kernel |
|---|---|---|---|
| 10% | 78.40%:0.0027 | 77.12%:50.62 | 75.10%:22.86 |
| 5% | — | 72.25%:27.88 | 71.16%:7.86 |
| 2.5% | — | 72.96%:28.02 | 71.69%:7.82 |
| 0.8% | — | 72.50%:27.46 | 71.04%:8.02 |

Table 1: The training efficiency of an OPF classifier with the highest accuracy is still much higher than the training efficiency of the SVM approaches using considerably reduced training set sizes.

### *4.2. EOPF versus the previous OPF classifier*

The experiments from now on use 5 datasets from different application domains.

- IBSR - Internet Brain Segmentation Repository[1]: this dataset is composed of 18 T1 weighted images with 1.5mm slice thickness. In this dataset, each voxel is considered as a sample for training and classification, and each image contains more than 1 million of voxels. The feature vector of each voxel is composed of its intensity value and the five intensities around the median value of the voxels within its 26-neighborhood. This feature vector preserves the border of the tissues and is robust to high-frequency noise. The idea is to automatic distinguish white matter from gray matter voxels. We used only one image of this dataset, which contains 1,079,258 voxels (samples), with 6 features each, distributed in two classes (white and gray matter). Figures 4a and 4b display a 3D image slice of this dataset and its corresponding ground truth.

- Poker: the purpose of this dataset, which contains 1,025,010 samples, is to predict poker hands. Each sample is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes. The number of classes is 10 [32].

- IJCNN: this dataset was used in the International Joint Conference

[1]URL: www.cma.mgh.harvard.edu/ibsr/

on Neural Networks (IJCNN) competition. It consists into 141,691 samples with 13 features each, distributed in 2 classes [32].

- Chess: this dataset contains 28,056 samples with 6 features each, distributed in 18 classes. The idea is to predict players's movements, and the details about this dataset can be found in [32].

- Metallographic: this dataset is composed by pixels of one labeled metallographic image with 76,800 pixels to be classified into 2 classes: cast iron and background. The number of features is 3, which corresponds to the RGB values of each pixel. Figures 4c and 4d display one image of this dataset and its corresponding ground truth.

  For each dataset, we used different percentages of samples in the training $Z_1$, evaluation $Z_2$, and test $Z_3$ sets:

  - IBSR: $|Z_1| = 1\%$, $|Z_2| = 49\%$, and $Z_3 = 50\%$ of the samples.
  - Poker: $|Z_1| = 1\%$, $|Z_2| = 49\%$, and $Z_3 = 50\%$ of the samples.
  - IJCNN: $|Z_1| = 5\%$, $|Z_2| = 45\%$, and $Z_3 = 50\%$ of the samples.
  - Chess: $|Z_1| = 5\%$, $|Z_2| = 45\%$, and $Z_3 = 50\%$ of the samples.
  - Metallographic: $|Z_1| = 5\%$, $|Z_2| = 45\%$, and $Z_3 = 50\%$ of the samples.

Table 2 displays the results of the experiments using Algorithm 1 on $Z_1$ for training both OPF and EOPF classifiers and their respective classification algorithms for testing on $Z_3$. Note that $Z_2$ is used only in the next section, where the learning-with-pruning algorithm is applied. These experiments were also repeated 10 times with randomly selected sets to compute the average accuracies. The results are displayed as follows: $x \pm y[z]$, in which $x$, $y$ and $z$ indicate, respectively, the average accuracy, standard deviation and mean classification time (in seconds).

Note that, EOPF and OPF provide similar accuracies, being the former from 1.5 to 5 times faster than the latter for the same training and test sets.

### *4.3. EOPF with and without pruning*

The experiments in this section show that it is possible to improve the quality of samples in $Z_1$ by learning from the classification errors in $Z_2$ and, at the same time, considerably reduce the size of $Z_1$ (increasing classification

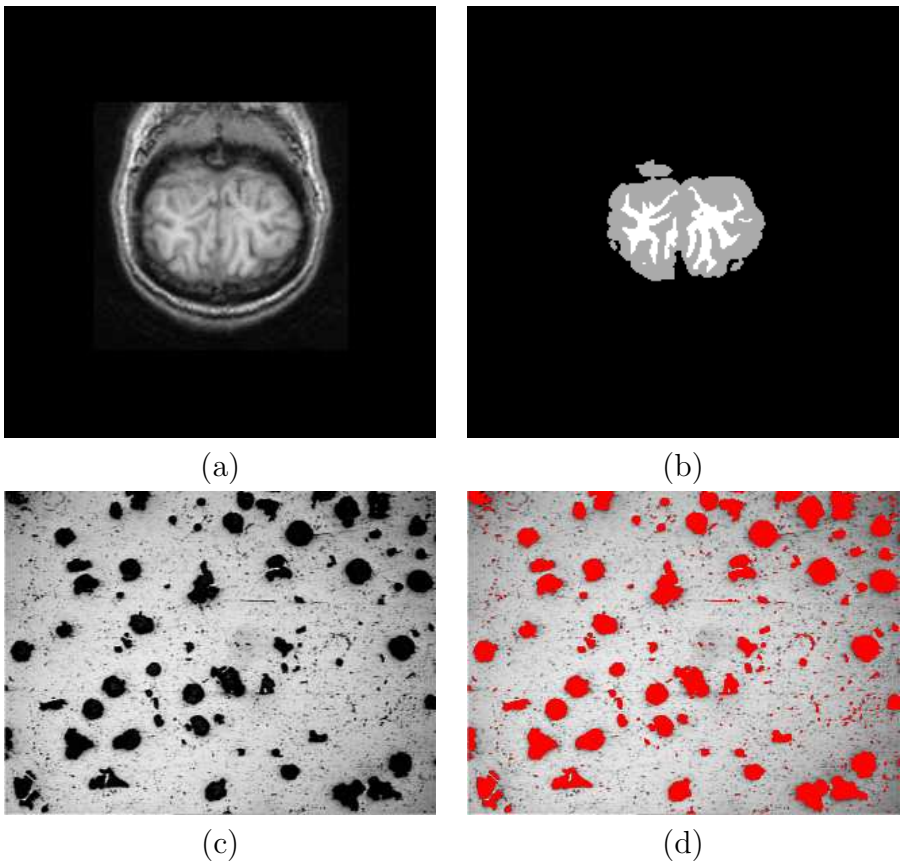



Figure 4: Some images used in the experiments. (a)-(b) MRI-T1 images from a human brain: (a) slice number 10 from IBSR-01 image and its corresponding labeled image in (b), in which the white matter is represented by the white pixels and the gray matter by the gray ones. These are $256 \times 256$ images encoded with 256 bits per pixel. (c)-(d) Metallographic images from nodular cast iron: (c) original and (d) labeled, in which the red pixels indicate the cast iron. These are $256 \times 300$ images encoded with 256 bits per pixel. The red pixels were used as the positive samples (cast iron) and the remaining ones were associated to the background.

efficiency) with negligible loss in accuracy. For a limit $MLoss = 5\%$ in Algorithm 4, for example, the results are shown in Table 3. The average numbers of accuracy, standard deviation and mean classification time (in seconds) were obtained after 10 runs of the methods with different training, evaluating and test sets.

The results indicate that the EOPF classifier using the learning-with-pruning approach can be from 1.1 to 6.19 times faster in classification with

| Dataset | OPF | EOPF |
|---|---|---|
| Brain | 71.95%±2.22[393.00] | 71.22%±2.16[292.84] |
| Poker | 53.61%±0.10[295.80] | 53.96%±0.12[266.85] |
| IJCNN | 88.10%±0.43[31.65] | 88.10%±0.43[23.39] |
| Chess | 64.88%±0.48[1.03] | 64.13%±0.62[0.93] |
| Metallographic | 100%±0.0[7.98] | 100%±0.0[1.61] |

Table 2: Mean accuracy and classification times for OPF and EOPF.

| Dataset | EOPF | EOPF+Pruning | Pruning rate |
|---|---|---|---|
| Brain | 71.22%±2.16[292.84] | 71.22%±2.36[307.12] | 0.0% |
| Poker | 53.96%±0.12[295.80] | 53.02%±0.08[266.85] | 0.58% |
| IJCNN | 88.10%±0.43[23.39] | 93.92%±0.25[13.85] | 58.87% |
| Chess | 64.13%±0.62[0.93] | 65.03%±0.65[1.06] | 0.55% |
| Metallographic | 100%±0.0[1.61] | 100%±0.0[0.26] | 94.69% |

Table 3: Mean accuracy, classification times for EOPF and EOPF+pruning. We also show the mean pruning rate for EOPF+pruning.

respect to EOPF without pruning. This speed up was not possible in the first case only of the Brain dataset using $MLoss = 5\%$, because all samples were considered relevant.

## 5. Conclusion

We presented an enhanced version of the OPF classifier (EOPF) and its combination with a new approach to learn relevant training samples from an evaluation set and, at the same time, eliminate irrelevant samples from the training set, reducing its size and increasing efficiency of classification with negligible loss in accuracy.

Experiments with large datasets from distinct applications showed that the new approach achieves the desired result. We conclude that OPF is more suitable to handle large datasets than popular methods, such as SVMs and ANNs, by showing that OPF can provide better accuracy and at the same time be extremely faster than SVM-RBF and SVM no-kernel for training. We also demonstrated that EOPF classification can be from 1.5 to 5 times faster than the OPF classification due to a new algorithm which reduces the

original time complexity from $O(|Z1|\,|Z3|)$ to $O(p\,|Z3|)$ for some $p << |Z1|$. The experiments also demonstrated that it is possible to reduce the training set size up to 95%, increasing efficiency in classification from 1.1 to 6.19 times faster.

Considering that new technologies have provided large datasets for many applications and the popular approaches present prohibitive training times in such situations, we may then conclude that the proposed methods are important contributions for pattern recognition. For future works, we will try to speed up OPF training phase by studying other algorithms to calculate the Minimum Spanning Tree, which is used to find prototypes and is responsible for almost 50% of the training step execution time. Other future work will be guided to adapt LibOPF to be executed in GPUs.

## Acknowledgments

The authors thank the financial support from FAPESP (grants 2009/16206-1 and 2007/52015-0) and CNPq (grant 481556/2009-5).

## References

[1] P. Miranda, A. Falcão, J. Udupa, Synergistic arc-weight estimation for interactive image segmentation using graphs, Computer Vision and Image Understanding 114 (1) (2010) 85–99.

[2] T. Spina, J. Montoya-Zegarra, A. Falcão, P. Miranda, Fast interactive segmentation of natural images using the image foresting transform, in: Proceedings of the 16th International Conference on Digital Signal Processing, IEEE, Santorini, Greece, 2009, pp. 1–8. doi:10.1109/ICDSP.2009.5201044.

[3] A. Silva, A. Falcão, L. Magalhães, A new CBIR approach based on relevance feedback and optimum-path forest classification, Journal of WSCG 18 (1-3) (2010) 73–80.

[4] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297.

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

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

[7] A. Bordes, S. Ertekin, J. Weston, , L. Bottou, Fast kernel classifiers with online and active learning, Journal of Machine Learning Research 6 (2005) 1579–1619.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: A library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.

[9] J. P. Papa, A. X. Falcão, C. T. N. Suzuki, Supervised pattern classification based on optimum-path forest, International Journal of Imaging Systems and Technology 19 (2) (2009) 120–131.

[10] J. P. Papa, A. X. Falcão, A new variant of the optimum-path forest classifier, in: Proceedings of the 4th International Symposium on Advances in Visual Computing, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 935–944.

[11] L. M. Rocha, F. A. M. Cappabianco, A. X. Falcão, Data clustering as an optimum-path forest problem with applications in image analysis, International Journal of Imaging Systems and Technology 19 (2) (2009) 50–68.

[12] J. Papa, A. Falcão, G. Freitas, A. Ávila, Robust pruning of training patterns for optimum-path forest classification applied to satellite-based rainfall occurrence estimation, IEEE Geoscience and Remote Sensing Letters 7 (2) (2010) 396–400.

[13] J. Papa, A. A. Spadotto, A. Falcão, R. C. Guido, J. C. Pereira, Robust biomedical signal recognition though optimum-path forest, Artificial Intelligence in Medicine(accepted for publication).

[14] A. Iliev, M. Scordilis, J. Papa, A. Falcão, Spoken emotion recognition through optimum-path forest classification using glottal features, Computer, Speech, and Language 24 (3) (2010) 445–460.

[15] J. Papa, A. N. Marana, A. A. Spadotto, R. C. Guido, A. Falcão, Robust and fast vowel recognition using optimum-path forest, in: Proceedings of the 35th International Conference on Acoustics, Speech, and Signal Processing, 2010, pp. 2190–2193.

[16] G. Chiachia, A. Marana, J. Papa, A. Falcão, Infrared face recogniton by optimum-path forrest, in: Proceedings of the 16th International Conference on Systems, Signals, and Image Processing, 2009, pp. 1–4.

[17] J. Papa, A. Falcão, A. Levada, D. Corrêa, D. Salvadeo, N. Mascarenhas, Fast and accurate holystic face recognition through optimum-path forest, in: Proceedings of the 16th International Conference on Digital Signal Processing, 2009, pp. 1–6.

[18] I. V. Guilherme, A. N. Marana, J. P. Papa, G. Chiachia, L. C. S. Afonso, K. Miura, M. V. D. Ferreira, F. Torres, Petroleum well drilling monitoring through cutting image analysis and artificial intelligence techniques, Engineering Applications of Artificial Intelligence-doi:http://dx.doi.org/10.1016/j.engappai.2010.04.002.

[19] R. Minetto, J. P. Papa, T. V. Spina, A. Falcão, N. J. Leite, J. Stolfi, Fast and robust object tracking using image foresting transform, in: Proceedings of the 16th International Conference on Digital Signal Processing, 2009, pp. 1–4.

[20] J. P. Papa, V. H. C. de Albuquerque, A. X. Falcão, J. M. R. S. Tavares, Fast automatic microstructural segmentation of ferrous alloy samples using optimum-path forest, in: Proc. of The International Symposium CompImage'10, Vol. 6026 of Lecture Notes in Computer Science, Springer, 2010, pp. 210–220.

[21] J. Papa, F. Cappabianco, A. Falcão, Optimizing optimum-path forest classification for huge datasets, in: 20th IEEE Intl. Conf. on Pattern Recognition, Istanbul, Turkey, 2010, to appear.

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

[23] T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, MIT, 1990.

[24] F. A. M. Cappabianco, A. Falcão, L. M. Rocha, Clustering by optimum-path forest and its applications to automatic gm/wm classification in mr-t1 images of the brain, in: Proceedings of the 2008 International Symposium of Biomedical Imaging: From Nano to Macro, 2008, pp. 428–431.

[25] J. Papa, A. Falcão, A learning algorithm for the optimum-path forest classifier, in: Graph-Based Representations in Pattern Recognition, Springer Berlin/Heidelberg, 2009, pp. 195–204.

[26] F. A. Cappabianco, A. Falcão, C. L. Yasuda, J. K. Udupa, MR-Image Segmentation of Brain Tissues based on Bias Correction and Optimum-Path Forest Clustering, Tech. Rep. IC-10-07, Institute of Computing, University of Campinas (March 2010).

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

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

[29] J. P. Papa, C. T. N. Suzuki, A. X. Falcão, LibOPF: A library for the design of optimum-path forest classifiers, software version 2.0 available at `http://www.ic.unicamp.br/~afalcao/LibOPF` (2009).

[30] C. C. Chang, C. J. Lin, LIBSVM: A Library for Support Vector Machines, software available at url http://www.csie.ntu.edu.tw/~cjlin/libsvm (2001).

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

[32] A. Frank, A. Asuncion, UCI machine learning repository (2010). URL `http://archive.ics.uci.edu/ml`