Pattern Recognition Letters 87 (2017) 117–126

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Optimum-Path Forest based on k-connectivity: Theory and applications[☆]

João Paulo Papa[a,*], Silas Evandro Nachif Fernandes[b], Alexandre Xavier Falcão[c]

[a] Department of Computing, São Paulo State University, Av. Eng. Luiz Edmundo Carrijo Coube, 14-01, Bauru, SP 17033-360, Brazil
[b] Department of Computing, Federal University of São Carlos, Rod. Washington Luís, Km 235, São Carlos, SP 13565-905, Brazil
[c] Institute of Computing, University of Campinas, Av. Albert Einstein, 1251, Campinas, SP 13083-852, Brazil

ARTICLE INFO

Article history:
Available online 10 August 2016

Keywords:
Pattern classification
Optimum-Path Forest
Supervised learning

ABSTRACT

Graph-based pattern recognition techniques have been in the spotlight for many years, since there is a constant need for faster and more effective approaches. Among them, the Optimum-Path Forest (OPF) framework has gained considerable attention in the last years, mainly due to the promising results obtained by OPF-based classifiers, which range from unsupervised, semi-supervised and supervised learning. In this paper, we consider a deeper theoretical explanation concerning the supervised OPF classifier with k-neighborhood (OPF$_k$), as well as we proposed two different training and classification algorithms that allow OPF$_k$ to work faster. The experimental validation against standard OPF and Support Vector Machines also validates the robustness of OPF$_k$ in real and synthetic datasets.

1. Introduction

Roughly speaking, pattern recognition techniques aim at learning a function that maps the input data to a set of predicted labels or continuous-valued outputs. Depending on the amount of knowledge we have about the training set, we can classify pattern recognition techniques in two main approaches: (i) supervised learning, which refers to situations one has full knowledge about the training data, and (ii) unsupervised learning, where we have no information about the dataset [7]. Recently, a new sort of approaches have been referred to semi-supervised ones, which feature some knowledge about a small subset of the training data. Such approaches make use of the active learning theory, which aims at improving data classification by means of user interaction.

Cutting edge research on pattern recognition may have contributed with its prominent works in the last years. Advances in hardware technology have allowed complex mathematical theories to be in lockstep with machine learning-based software development. Probabilistic models, techniques based on statistical learning theory and neural networks have been always the forerunners for pattern recognition-like applications. Support Vector Machines (SVM) [6], for instance, may be considered the hallmark with respect to kernel-based learning techniques. Since we can face complex and overlapped feature spaces, it might be interesting to employ kernel functions to map the data onto a higher dimensional representation.

Neural networks still play an important hole in the pattern recognition research field, since there is always room for improvements in the old fashion techniques [15,21,23]. In the last years, a special attention has been devoted to deep learning architectures [2,13], since they can be very robust to changes in scale, rotation and brightness in regard to image classification tasks. Recent advances in Bayesian networks [4,10], k-means [14] and the well-known Gaussian Mixture Models [5] have maintained the tradition of such techniques.

Another interesting framework that leads to a very interesting and powerful tool for pattern recognition concerns with graph-based methods. Basically, such methods model the machine learning task as a problem formulated in the graph theory: the dataset samples, which are represented by their corresponding feature vectors, are the graph nodes, that are further connected by an adjacency relation. Without loss of generality, a graph-based method aims at removing or adding edges using some heuristic in order to create connected components, which stand for a group of samples that share some similar characteristics [3].

Papa et al. [19,20] presented a new framework for graph-based pattern recognition named Optimum-Path Forest (OPF), which addresses the graph partition task as a competition process among some key (prototype) samples in order to conquer the remaining nodes according to a path-cost function. The idea is based on the Image Foresting Transform (IFT) [8], which works similarly to OPF,

☆ This paper has been recommended for acceptance by Cheng-Lin Liu.
* Corresponding author. Fax: +55 14 3103 6079.
E-mail addresses: papa@fc.unesp.br, papa.joaopaulo@gmail.com (J.P. Papa).

http://dx.doi.org/10.1016/j.patrec.2016.07.026
0167-8655/© 2016 Elsevier B.V. All rights reserved.

but in the context of designing image processing-like operators. Both OPF and IFT follow the idea of the ordered communities formation, in which an individual (node) will belong to the community (cluster) that gives him/her the best reward (path-cost function value).

In order to create an OPF-based classifier, we have to face three main questions: (i) how to connect samples, (ii) how to find out prototypes, and (iii) what sort of path-cost function one should employ in the competition process. Papa et al. [19,20] and Papa and Falcão [17,18] have addressed the above questions in two distinct ideas: (i) using a complete graph as adjacency relation, a Minimum Spanning Tree (MST)-based approach to find out prototypes, and a path-cost function (f_{max}) that computes the maximum arc-weight along a path (sequence of nodes) [19,20]; and (ii) using a k-nearest neighbors (k-nn) adjacency relation, a density-based approach to estimate prototypes, and a path-cost function (f_{min}) that computes the minimum value between the cost of a training node and the density of the test sample [17,18]. This latter formulation is based on the unsupervised OPF [22], which was proposed to handle data clustering problems. Recently, Souza et al. [24] proposed a new variant called k-OPF, which essentially assigns the most frequent label of the k-lowest path-costs to a given sample, instead of the lowest path-cost only.

The main differences regarding the OPF with complete graph and its version that employs an adjacency relation based on k-connectivity (OPF$_{knn}$) rely on: (i) the naïve OPF weights only edges, while (OPF$_{knn}$) weights both edges and nodes; (ii) the prototypes estimated by OPF are located at the frontier of the classes, and the key nodes estimated by OPF$_{knn}$ are positioned at the regions with highest density (center of clusters), and (iii) the classification process adopted by traditional OPF aims to *minimize* the cost of every sample using a path-cost function that computes the *maximum* arc-weight along a path; and the classification process of OPF$_{knn}$ tries to *maximize* the cost of every sample using a path-cost function that computes the *minimum* value between the cost of a training sample and the density of a test node. Notice k-OPF and OPF$_{knn}$ are different to each other, since the first one uses the complete graph as adjacency relation, it computes the prototypes using the Minimum Spanning Tree approach, and uses f_{max} as the path-cost function. The latter approach employs a k-nn graph, it computes prototypes based on a probability density function, and uses f_{min} as the path-cost function.

In this paper, we extend the research of Papa and Falcão [17,18] by addressing in more details the working mechanism of OPF$_{knn}$, as well as we propose to model the problem of finding the size of the k-neighborhood as an optimization task using meta-heuristics. Since Papa and Falcão [17] proposed to use an exhaustive search for finding the best value of k, i.e., the one that maximizes the accuracy over the training set, our approach can speed up the original work, as well as we can reduce the overtraining, since the proposed optimization process is conducted over a validating set. Another contribution of this work is to take advantage of the cost of each training sample, which has been computed during the training phase already, when classifying samples, i.e., we can simply halt the classification process earlier without affecting the theoretical basis of the algorithm. Therefore, the main contributions of this paper are three-fold: (i) to present a deeper formulation with respect to OPF$_{knn}$, (ii) to propose a meta-heuristic-based approach to automatically estimate the neighborhood size for density computation purposes, and (iii) to propose a faster classification process for the OPF$_{knn}$ technique. In addition, we have compared OPF$_{knn}$ against traditional OPF and Support Vector Machines.

The remainder of the paper is organized as follows. Sections 2 and 3 present the OPF$_{knn}$ background theory and the proposed approach to speed up both the training and testing phases, respectively. Experiments are discussed in Section 4, and Section 5 states conclusions and future works.

2. Optimum-Path Forest with *knn*-connectivity

In this section, we describe the theory related to OPF$_{knn}$, as well as the basis of OPF-based classifiers.

2.1. Theoretical background

Let $\mathcal{Z}$ be a labeled dataset such that $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \mathcal{Z}_3$, where $\mathcal{Z}_1$, $\mathcal{Z}_2$ and $\mathcal{Z}_3$ stand for a training, validating and testing sets, respectively. A graph $G = (\mathcal{V}, \mathcal{A})$ can be derived from $\mathcal{Z}$ such that each $s \in \mathcal{Z}$ becomes a graph node $v(s) \in \mathcal{V}$, where $v(\cdot)$ stands for a function that extracts the feature vector of given dataset sample (e.g., image, pixel, voxel or signal). Additionally, $\mathcal{A}$ denotes an adjacency relation that connects the samples in $\mathcal{V}$, and $d : \mathcal{V} \times \mathcal{V} \rightarrow \Re^+$ defines a function that is used to weight the edges in $\mathcal{A}$. Analogously to the construction of G, we can also derive $G_1 = (\mathcal{V}_1, \mathcal{A}_1)$, $G_2 = (\mathcal{V}_2, \mathcal{A}_2)$ and $G_3 = (\mathcal{V}_3, \mathcal{A}_3)$ from $\mathcal{Z}_1$, $\mathcal{Z}_2$ and $\mathcal{Z}_3$, respectively. However, as the adjacency relation is the same for the entire dataset, we can adopt $\mathcal{A}$ for all graphs.

Let π_s be a path in $\mathcal{G}$ with terminus in node $s \in \mathcal{V}$, and $\langle \pi_s \cdot (s, t)\rangle$ be the concatenation between path π_s and the arc $(s, t) \in \mathcal{A}$. We also denote $\langle t\rangle$ as being a trivial path. The idea of an OPF-based classifier is to use a smooth path-cost function f in order to rule a competition process in G among a set of prototype nodes $\mathcal{S} \subseteq \mathcal{V}$. The OPF algorithm aims at minimizing/maximizing $f(s)$ for every sample $s \in \mathcal{V}$, being the smoothness of f defined as follows [8]: for every sample $t \in \mathcal{V}$, there exists an optimum-path π_t which is trivial or can be represented by $\langle \pi_s \cdot (s, t)\rangle$, where

- $f(\pi_s) \leq f(\pi_t)$;
- π_s is optimum; and
- for every optimum-path τ_s, $f(\langle \tau_s \cdot (s, t)\rangle) = f(\pi_t)$.

The OPF proposed by Papa et al. [19,20] adopts $\mathcal{A}$ as a complete graph, the prototype set $\mathcal{S}$ is designed as being the connected samples in an MST computed over the training set, and f outputs the maximum arc-weight along a path (f_{max}). Such OPF configuration is motivated by the fact that an Optimum-Path Forest computed over a graph using f_{max} follows the shape of an MST computed over it, which means we can obtain the very same Optimum-Path Forest as previously computed using OPF by just removing the arcs that connect samples from different classes in the MST, and then propagating their costs using f_{max}. This behavior was observed by the work of Alléne et al. [1], and it has been used to make OPF training phase faster [11]. If one has an unique MST, i.e., all arc-weights are different to each other, the OPF classification error over the training set would be reduced to zero.

The main problem in reducing the error over the training set is related to a possible data overfitting. Therefore, motivated by such assumption, Papa and Falcão [17] proposed the OPF$_{knn}$, which models $\mathcal{A}$ as being an adjacency relation that connects each sample to its k-nearest neighbors (say that $\mathcal{A}_k$); the prototypes are now estimated as the nodes located at the highest density regions, and a path-cost function that aims at maximizing the cost of every sample is now employed. Roughly speaking, OPF$_{knn}$ has two phases: a training and a classification step. The former is responsible for computing the density of each training node using $\mathcal{A}_{k^*}$, being k^* the best value of k that maximizes some criterion, and then to start the competition process among prototypes. After that, we have an Optimum-Path Forest computed over the training set, which will be used to classify each test sample. The classification process just picks up a sample from the test set, connects it to its k^*-nearest neighbors in the Optimum-Path Forest generated by the training

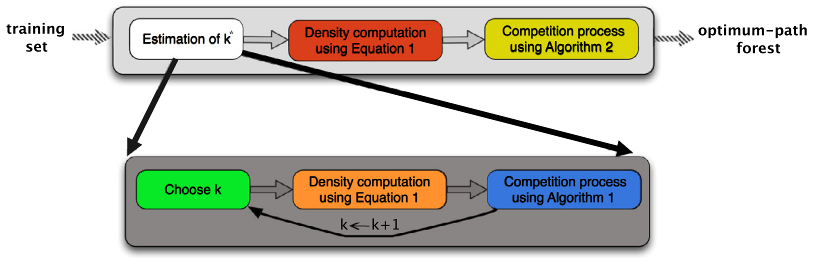
training set
Estimation of k*
Density computation using Equation 1
Competition process using Algorithm 2
optimum-path forest
Choose k
Density computation using Equation 1
Competition process using Algorithm 1
k←k+1

Fig. 1. OPF$_{knn}$ training step workflow. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

phase, and then uses the same OPF$_{knn}$ rule employed in the competition process to conquer that sample. After that, the test sample is then removed from the graph, and the above process starts over again for each test sample. The next sections describe in more details the aforementioned procedures.

2.2. Training

The training phase is responsible for generating the Optimum-Path Forest, which encodes the cost C and the predecessor map P of each sample, i.e., the optimum-path from each training sample to its most strongly connected prototype. The OPF$_{knn}$ training phase is composed of three modules: (i) the estimation of k^*, (ii) the computation of a probability density function for each training sample, and (iii) the competition process between prototypes. Fig. 1 depicts the above steps.

The first step is to compute k^* ("white" module in Fig. 1), which stands for the computation of the k value that maximizes some criterion in order to create the k-nearest neighbors graph. The approach proposed by Papa and Falcão [17] ("dark gray" module in Fig. 1) searches for a k value within the range $[1, k_{max}]$ that maximizes the OPF$_{knn}$ accuracy over the training set. First of all, an initial k value is chosen ("green" module in Fig. 1), usually $k = 1$ being increased by one unit up to k_{max}, and the k-nearest neighbors graph is built. After that, the next step is to compute a probability density function (pdf) $\rho(s)$ of each node $s \in \mathcal{V}_1$ ("orange" module in Fig. 1), as follows:

$$\rho(s) = \frac{1}{\sqrt{2\pi\sigma^2}|\mathcal{A}^*_k(s)|}\sum_{\forall t\in\mathcal{A}^*_k(s)} \exp\left(\frac{-d^2(s,t)}{2\sigma^2}\right), \tag{1}$$

where $d(s, t)$ stands for the distance between samples s and t, $|\mathcal{A}^*_k(s)| = k^*$, $\sigma = \frac{d_f}{3}$, and d_f is the maximum arc weight in G_1. Notice this parameter choice considers all adjacent nodes for density computation, since a Gaussian function covers most samples within $d(s, t) \in [0, 3\sigma]$.

Further, the competition process ("blue" module in Fig. 1) takes place in G_1. As aforementioned, a set of prototype samples $\mathcal{S} \subseteq \mathcal{V}_1$ compete among themselves in order to conquer the remaining samples in $\mathcal{V}_1$ offering to them optimum-paths according to a path-cost function f_{min} inspired by unsupervised OPF [22], given by:

$$f_{min}(\langle t\rangle) = \begin{cases}\rho(t) & \text{if } t\in S\\ \rho(t)-1 & \text{otherwise}\end{cases}$$
$$f_{min}(\pi_s\cdot\langle s,t\rangle) = min\{f_{min}(\pi_s), \rho(t)\}. \tag{2}$$

In short, the idea of assigning $\rho(t) - 1$ to a sample $t \in \mathcal{S}$ is to avoid plateaus nearby the maxima of the pdf. The OPF$_{knn}$ classifier tries to maximize Eq. (2) for each instance $s \in \mathcal{V}_1$ in order to partition $\mathcal{G}_1$ in an Optimum-Path Forest, being its roots (samples in $\mathcal{S}$) the nodes with highest density in $\mathcal{V}_1$. Algorithm 1 implements the above idea.

Algorithm 1 OPF$_{knn}$ algorithm.

```
Require: A $k$ neighborhood graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A}_k)$, and a path-cost
    function $f$.
Ensure: An oOtimum-Path Forest $P$, a label map $L$ and a cost map
    $C$.
 1: for every $s \in \mathcal{V}_1$ do
 2:   Compute $\rho(s)$ using Eq. (1).
 3:   $P(s) \leftarrow \emptyset$, $C(s) \leftarrow \rho(s) - 1$, $L(s) \leftarrow \lambda(s)$
 4:   Insert $s$ in the priority queue $Q$.
 5: end for
 6: while $Q \neq \emptyset$ do
 7:   Remove $s$ from $Q$ such that $C(s)$ is maximum.
 8:   if $P(s) = \emptyset$ then
 9:     $C(s) \leftarrow \rho(s)$.
10:   end if
11:   for each $t \in \mathcal{A}_k(s)$ do
12:     $tmp \leftarrow f(C(s), \rho(t))$.
13:     if $tmp > C(t)$ then
14:       $L(t) \leftarrow L(s)$, $P(s) \leftarrow s$, $C(t) \leftarrow tmp$.
15:     end if
16:   end for
17: end while
```

Lines 1–4 are responsible for initializing the cost map $C(s)$, $\forall s \in \mathcal{V}_1$, as well as the predecessor map $P(s)$, which contains the conqueror of s, is set to a null value. In addition, the label map L of each instance s is initialized with its true label, given by $\lambda(s)$. The core of Algorithm 1 is given by the main loop in Lines 6–17: Line 7 removes from Q the node with maximum density value, since OPF$_{knn}$ considers the prototypes as being the nodes with highest density values, i.e., the ones located in the center of the classes. If that removed instance has not been conquered so far, i.e., it has no predecessor (i.e., $P(s) = \emptyset$), its density is set to its original value (steps implemented in Lines 8 and 9 and described by the first part of Eq. (2)).

The next step concerns to analyze the neighborhood of s in Lines 11–16 of Algorithm 1, say that $\forall t \in \mathcal{A}_k(s)$, in order to check whether s is capable to conquer the instances that fall in $\mathcal{A}_k(s)$. Line 12 computes the minimum value between $C(s)$ and the density of t denoted as $\rho(t)$ (this procedure has been described in the second part of in Eq. (2)), and if this value is better (say that greater) than $C(t)$, it means s can conquer t assigning to it the label of s, as well as sample s becomes the predecessor of t (Line 14). The cost $C(t)$ is also updated[1].

However, Eq. (2) may not guarantee one maximum (at least) per class. Thus, the idea of OPF$_{knn}$ training step is to first find out k^* using Algorithm 1, and then to execute it again using a modified

[1] For the sake of explanation purposes, Line 12 of Algorithm 1 implements $tmp \leftarrow min\{f(C(s), \rho(s))\}$ when one considers the path-cost f_{min}.

version of function f_{min} called f'_{min}:

$$f'_{min}(\langle t\rangle) = \begin{cases} \rho(t) & \text{if } t \in \mathcal{S} \\ \rho(t) - 1 & \text{otherwise} \end{cases}$$

$$f'_{min}(\pi_s \cdot \langle s,t\rangle) = \begin{cases} -\infty \text{ if } \lambda(t) \neq \lambda(s) \\ min\{f'_{min}(\pi_s), \rho(t)\} \text{ otherwise.} \end{cases} \quad (3)$$

Therefore, Eq. (3) penalizes all arcs $(s,t) \in \mathcal{A}^*_k$ such that $\lambda(s) \neq \lambda(t)$, avoiding such arcs to belong to some optimum-path. In short, the OPF$_{knn}$ training step can be summarized as follows: after estimating k^* (“white” module in Fig. 1), we build the k^*-nearest neighbors graph and the pdf of each training node is then computed (“red” module in Fig. 1). After that, the OPF$_{knn}$ algorithm with f'_{min} is performed over the training set (“yellow” module in Fig. 1). Algorithm 2 implements the whole OPF$_{knn}$ training phase.

Algorithm 2 OPF$_{knn}$ training step.

```
Require: An λ-labeled training set $\mathcal{Z}_1$ and the parameter $k_{max}$.
Ensure: An Optimum-Path Forest $P$, a label map $L$ and a cost map $C$.
 1: $MaxAcc \leftarrow -\infty$, $k^* \leftarrow 0$.
 2: for $k = 1$ to $k_{max}$ do
 3:   Create a $k$-neighborhood graph $G_1 = (\mathcal{V}_1, \mathcal{A}_k)$ using $\mathcal{Z}_1$.
 4:   $[P, L, C] \leftarrow$ Algorithm 1$(G_1, f_{min})$.
 5:   $Acc \leftarrow$ recognition rate over $\mathcal{Z}_1$.
 6:   if $Acc > MaxAcc$ then
 7:     $MaxAcc \leftarrow Acc$, $k^* \leftarrow k$.
 8:   end if
 9: end for
10: Create a $k^*$-neighborhood graph $G_1 = (\mathcal{V}_1, \mathcal{A}_{k^*})$ using $\mathcal{Z}_1$.
11: $[P, L, C] \leftarrow$ Algorithm 1$(G_1, f'_{min})$.
```

The loop in Lines 2–9 is responsible for choosing k^* within the range [1, k_{max}], as described by the “dark” module in Fig. 1. Notice this first part of OPF$_{knn}$ training algorithm employs Algorithm 1 with the path-cost function f described by Eq. (2). Further, the k^*-neighborhood graph is built and OPF$_{knn}$ is trained once again over the same training set $\mathcal{Z}_1$ in Lines 10 and 11, but now the Algorithm 1 is used with path-cost function f'_{min} (Eq. (3)).

2.3. Classification

The OPF$_{knn}$ classification step is straightforward, and it employs the very same competition process adopted during the training phase: for each $t \in \mathcal{Z}_3$ (testing set), we connect t to its k^*-nearest neighbors in $\mathcal{Z}_1$, and then we find out the sample $s \in \mathcal{Z}_1$ that satisfies the equation below:

$$C(t) = \max_{s \in A_{k^*}(t)} \{\min\{C(s), \rho(t)\}\}. \quad (4)$$

The classification step simply assigns $L(t) = \lambda(s)$. Notice the testing samples are not permanently added to $\mathcal{Z}_1$.

3. Improvements on OPF$_{knn}$

In this section, we present the two main improvements proposed in this paper. First, we show how to perform training step faster by modeling the problem of finding suitable neighborhoods as an optimization task, and then we also show how to obtain a faster classification step.

3.1. Learning k*-connectivity by means of meta-heuristics*

The algorithm proposed by Papa and Falcão [17] to learn suitable values for k consists, basically, on an exhaustive search within the range [1, k_{max}] aiming at finding out the k value that maximizes the OPF$_{knn}$ accuracy over the training set, i.e., k^*. In this paper, we propose a fast and less prone to overfitting approach for finding suitable values for k based on meta-heuristics. The idea is to model the problem of finding k^* as an optimization task, in which any meta-heuristic-based approach can be employed. The reason for using such sort of algorithms relies on their simplicity and ability for solving optimization problems.

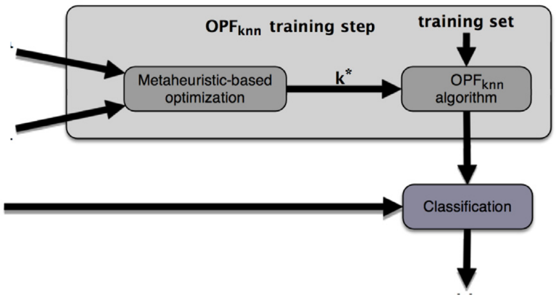

Fig. 2. Proposed optimization approach to estimate k^*.

In order to accomplish such purpose, we make use of a labeled validating set ($\mathcal{Z}_2$) to be employed as a guideline during the search of suitable k values: each agent (possible solution) a_i, $i = 1, 2, \ldots, M$, where M stands for the number of agents, is initialized with a random value $k_i \in$ [1, k_{max}]; then a k_i-neighborhood graph is built and the OPF$_{knn}$ is trained over $\mathcal{Z}_1$ and its accuracy rate is computed over $\mathcal{Z}_2$. Such accuracy is used as the fitness value to guide the optimization process, which aims at maximizing this accuracy. This procedure is performed for every agent a_i, thus defining one iteration of the optimization algorithm. The aforementioned steps are then repeated over again until a criterion be satisfied (usually, a number of T iterations is used for that purpose).

In this paper, we validated the proposed approach using two well-known meta-heuristic-based techniques: Particle Swarm Optimization (PSO) [12] and Harmony Search (HS) [9]. Although any other meta-heuristic technique can be employed, there are two main reasons for choosing them: (i) both techniques are well accepted by the scientific community, and (ii) as we dealing with a discrete optimization problem, HS fits well such situation, since it is a native discrete optimization technique, and also there are several discrete versions of PSO in the literature. Fig. 2 illustrates the proposed optimization approach.

3.2. Speeding-up classification using cost queue

The standard OPF$_{knn}$ classification approach described in Section 2.3 performs the very same operation for every test sample, i.e., the idea is to connect that sample to its k^*-nearest neighbors, to compute its density using Eq. (1), and further to evaluate which training node satisfies Eq. (4). Notice this last step needs to be executed for all neighbors of the sample that is going to be classified.

However, when $k^* \rightarrow \infty$, the computational load might be somehow prohibitive, since one need first to find out the k^*-nearest neighbors, and then to estimate the cost that each neighbor will offer to that testing sample. Basically, the classification step runs over all training samples that are connected to the testing node, and then takes the one that offers the maximum cost, which is computed using f'_{min}. For the sake of explanation, consider the classification process displayed in Fig. 3a, in which one can observe a testing sample (“white”), as well as training samples from classes “red” and “blue”[2].

[2] The values over the nodes stand for their costs estimated though the training step.

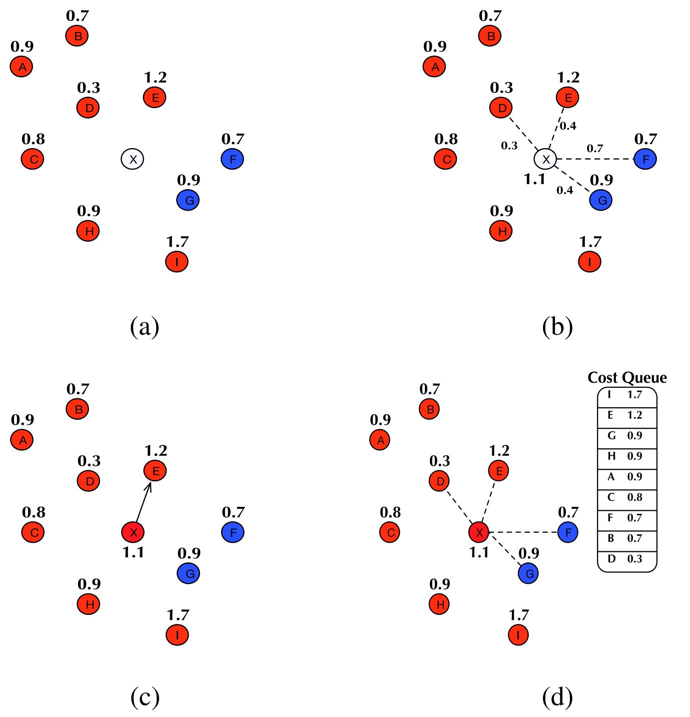

Fig. 3. OPF$_{knn}$ classification process: (a) training samples from classes "red" and "blue", and one testing sample ("white") to be classified, (b) testing sample is connected to its 4-nearest neighbors, for the further computation of its density, (c) testing sample is conquered by sample '*E*', and then labeled as belonging to class "red", and (d) cost queue used to speed up the classification step. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

As aforementioned, the first step is responsible for connecting the test sample to its k^*-nearest neighbors (Fig. 3b)[3], for further computing its density using Eq. (1). In this case, the values near the arcs stand for their weights (i.e., the distance among the nodes), and the value above/below the testing sample denotes its density (dummy value). The next step concerns with computing the cost that each training sample will offer to testing node '*X*'. Let $C_A(X)$ be the cost that training sample '*A*' offers to '*X*', and $C(A)$ be the cost associated with sample '*A*' (i.e., the cost computed after the training step). Therefore, if we consider the 4-neighborhood of sample '*X*', we have the following costs :

- $C_D(X) = min\{C(D), \rho(X)\} = min\{0.3, 1.1\} = 0.3$
- $C_E(X) = min\{C(E), \rho(X)\} = min\{1.2, 1.1\} = 1.1$
- $C_F(X) = min\{C(F), \rho(X)\} = min\{0.7, 1.1\} = 0.7$
- $C_G(X) = min\{C(G), \rho(X)\} = min\{0.9, 1.1\} = 0.9$

Finally, the sample that will conquer '*X*' is the one that offers the maximum cost among all costs within the neighborhood of '*X*', i.e.:

$$\begin{aligned} C(X) &= \max\{C_D(X), C_E(X), C_F(X), C_G(X)\} \\ &= \max\{0.3, 1.1, 0.7, 0.9\} \\ &= 1.1. \end{aligned} \tag{5}$$

Based on Eq. (5), we can thus conclude sample '*X*' will be conquered by training node 'E', and then associated to the label "red" (Fig. 3c). However, one can clearly observe there is no need to run through all training samples sequentially if we keep an ordered queue sorted by a decreasing order of costs, as displayed in Fig. 3d. In fact, this queue can be maintained in $\theta(1)$ during the training phase. Roughly speaking, we just need to evaluate the very first training node in the cost queue that is neighbor of the testing sample. For the sake of explanation, consider the following two possible situations:

- The first node in the queue offers a cost that is smaller or equal than the density of the test sample: consider the example depicted in Fig. 3. In this case, $C_E(X) = 1.1 \leq \rho(X)$. In this case, the next sample in the queue has a smaller (or equal) cost than '*E*', and thus it will not offer a cost that is greater than the one offered by sample '*E*' already. Since we are looking for to solve Eq. (4), i.e., to find the training node that offers the maximum cost among those computed using f'_{min}, there is no need to evaluate any other node in the cost queue.
- The first node in the queue offers a cost that is greater than the density of the test sample: suppose sample '*E*' now offers the cost $C_E(X) = 1.7 \geq \rho(X)$. In this case, the next sample in the queue that is neighbor of '*X*' will offer a cost that is smaller or equal than $C_E(X)$, and thus it will not conquer '*X*' with a better cost. Once again, there is no need to evaluate any other node in the cost queue.

[3] In this example, we assume $k^* = 4$.

Table 1
Description of the datasets.

Dataset	# samples	# features	# classes
aflw	8193	4096	2
Colon-cancer	62	2000	2
dmoz-web-directory	1329	10,629	5
Leukemia	72	7129	2
Pima-Indians-Diabetes	768	8	2
scene-classification	2407	294	15
Statlog-Australian	690	14	2
Statlog-dna	5186	180	3
Statlog-Heart	270	13	2
Statlog-Letter	35,000	16	26
Statlog-Shuttle	101,500	9	7
Synthetic1	500	2	2
Synthetic2	1000	2	2
Synthetic3	100,000	4	4
UCI-a1a	32,561	123	2
UCI-Breast-Cancer	683	10	2
UCI-Connect-4	67,557	126	3
UCI-Ionosphere	351	34	2
UCI-Liver-disorders	345	6	2
UCI-Mushrooms	8124	112	2
UCI-Pendigits	10,992	16	10
UCI-vowel	990	10	11
usps	9298	256	10
w1a	49,749	300	4
yahoo-web-directory	1106	10,629	4

4. Experimental section

In this section, we presented the methodology and experiments employed to validate the effectiveness and efficiency of OPF$_{knn}$, as well as its enhanced training and testing algorithms.

4.1. Datasets

We performed experiments over 25 real and synthetic datasets[4–7], whose main characteristics are presented in Table 1. Notice the datasets differ in the number of samples, features and classes. The choice of these datasets was motivated by their level

[4] http://mldata.org
[5] http://archive.ics.uci.edu/ml
[6] http://pages.bangor.ac.uk/~mas00a/activities/artificial_data.htm
[7] http://lrs.icg.tugraz.at/research/aflw

of complexity (overlapped samples), which turns the classification process more sensitive to misclassification[8].

4.2. Experimental setup

We divided the experiments in three distinct rounds: (i) in the first one, we evaluated the effectiveness and efficiency of OPF$_{knn}$ against traditional OPF and Support Vector Machines with Radial Basis Function kernel; (ii) further, the proposed approach to speed up the OPF$_{knn}$ training step by means of meta-heuristics (Section 3.1) is compared against naïve OPF$_{knn}$; and (iii) the last experiment aimed at showing the efficiency of the enhanced classification algorithm (Section 3.2), as well as we also showed the recognition rates between this new approach and standard OPF$_{knn}$ are statistically equivalent to each other.

4.3. Experiments

In this section, we first evaluated the robustness of OPF$_{knn}$ against naïve OPF (Section 4.3.1) and SVM, and then we assessed both the efficiency and effectiveness of the proposed approach based on a cost queue to speed up the classification phase (Section 4.3.2), hereinafter called pOPF$_{knn}$. Finally, we evaluated the robustness of the proposed approach based on meta-heuristics to find out k^* (Section 4.3.3), hereinafter called mOPF$_{knn}$.

4.3.1. Evaluating the robustness of OPF$_{\mathrm{knn}}$

In this section, we evaluated both the efficiency and effectiveness of OPF$_{knn}$ against traditional OPF and SVM. In regard to OPF-based classifiers, we used the open-source library LibOPF[9,10].

The experiments were conducted as follows: each dataset was partitioned into 50% of the samples for training, and the remaining 50% for classification purposes. As OPF$_{knn}$ has the neighborhood size to be optimized, we ended up partitioning the training set once more. Therefore, we have 30% of the entire datasets used for training, 20% employed for validation purposes (i.e., parameter fine-tuning), and the remaining 50% were used to assess the recognition rate over the testing set. Notice standard OPF used a training set with 50% of the samples (i.e., training and validating), since it does not have parameters to be fine-tuned. After learning, OPF$_{knn}$ was trained once more using the original training set (i.e., the very same one used by OPF).

In regard to SVM, we used a Radial Basis Function kernel with parameters optimized by means of a grid-search over the range $\gamma \in \{0.001, 0.01, 0.1, 1\}$ and $C \in \{1, 10, 100, 1000\}$. The parameters were optimized over the very same validating set used by OPF$_{knn}$, for further training SVM once more with the fine-tuned parameters over the merged training and validating sets.

In order to provide a statistical evaluation, we conducted a cross-validation procedure with 20 runnings, being the training, validating and testing sets the very same ones for all compared techniques. The final results were evaluated trough the Wilcoxon signed-rank test with significance of 0.05 [25]. In regard to the OPF$_{knn}$ parameter, we employed a neighborhood size of $k \in [1, 20]$, i.e., $k_{max} = 20$. Finally, we used an accuracy measure proposed by Papa et al. [20], which considers unbalanced datasets, as well as we considered the well-known F-measure. Notice such ranges were empirically set.

Table 2 shows the mean recognition rates over the testing set, in which the values in bold stand for the most accurate techniques according to Wilcoxon test. OPF$_{knn}$ obtained the best results in 11 out 25 datasets, being some recognition rates very similar to the ones obtained by OPF and SVM for some situations, and with a clear difference in others (e.g., “dmoz-web-directory-topics”, “Statlog-Shuttle” and “UCI-Pendigits”). Obviously, there are other datasets in which OPF$_{knn}$ performs worse than naïve OPF and SVM, as stated by Papa and Falcão [17,18] and Papa et al. [19,20]. Once again, the main purpose of this work is to highlight OPF$_{knn}$ and OPF are complementary to each other, which means one can use each of them in different situations.

Table 2
Mean recognition rates according to the approach proposed by Papa et al. [20].

Dataset	OPF$_{knn}$	OPF	SVM
aflw	89.57 ± 0.46	88.81 ± 0.70	**94.44** ± 0.31
Colon-cancer	**69.31** ± 8.46	**69.31** ± 8.41	51.91 ± 3.74
dmoz-web-directory-topics	**62.68** ± 2.26	62.11 ± 3.06	52.66 ± 2.21
Leukemia	**78.57** ± 5.22	**77.56** ± 5.48	50.00 ± 0.00
Pima-Indians-Diabetes	**64.89** ± 1.72	**64.38** ± 2.72	**64.89** ± 4.88
scene-classification	67.58 ± 1.18	67.13 ± 1.31	**69.50** ± 0.54
Statlog-Australian	65.33 ± 2.00	64.45 ± 2.00	**69.21** ± 2.16
Statlog-dna	88.96 ± 0.60	86.35 ± 2.21	**97.64** ± 0.42
Statlog-Heart	**59.76** ± 3.16	**59.53** ± 3.09	**60.75** ± 4.07
Statlog-Letter	98.60 ± 0.10	98.07 ± 0.45	**99.12** ± 0.07
Statlog-Shuttle	**97.33** ± 1.22	96.50 ± 1.58	95.20 ± 1.55
Synthetic1	52.62 ± 3.70	52.46 ± 3.44	**56.20** ± 2.59
Synthetic2	73.49 ± 1.21	72.48 ± 1.20	**81.19** ± 1.17
Synthetic3	86.10 ± 0.08	85.40 ± 0.13	**89.58** ± 0.11
UCI-a1a	70.96 ± 0.34	65.40 ± 1.20	**75.48** ± 0.52
UCI-Breast-Cancer	**95.36** ± 1.08	94.69 ± 1.19	**95.90** ± 0.84
UCI-Connect-4	64.21 ± 0.23	62.96 ± 0.42	**78.76** ± 2.02
UCI-Ionosphere	81.37 ± 2.46	80.37 ± 2.15	**91.73** ± 2.19
UCI-Liver-disorders	60.98 ± 3.04	60.13 ± 3.32	**63.58** ± 4.65
UCI-Mushrooms	**100.00** ± 0.0	94.32 ± 9.21	**99.99** ± 0.04
UCI-Pendigits	**99.58** ± 0.06	99.44 ± 0.31	98.87 ± 0.11
UCI-vowel	**97.06** ± 0.92	96.43 ± 0.82	**96.43** ± 1.58
usps	98.02 ± 0.14	97.73 ± 0.27	**98.47** ± 0.15
w1a	82.40 ± 0.24	80.86 ± 1.03	**82.81** ± 0.25
yahoo-web-directory-topics	**51.44** ± 1.43	50.86 ± 1.77	**52.50** ± 1.47

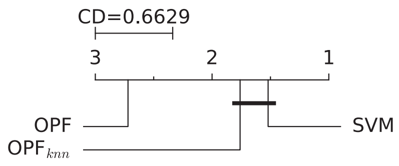

Fig. 4. Nemenyi statistical test concerning the recognition rates displayed in Table 2.

Additionally, we performed the non-parametric Friedman test, which is used to rank the algorithms for each dataset separately. In case of Friedman test provides meaningful results to reject the null-hypothesis (h_0: all techniques are equivalent), we can perform a post-hoc test further. For this purpose, we conducted the Nemenyi test, proposed by Nemenyi [16], which allows us to verify whether there is a critical difference (CD) among techniques or not. The results of the Nemenyi test can be represented in a simple diagram, in which the average ranks of the methods are plotted on the horizontal axis, where the lower the average rank is, the better the technique is. Moreover, the groups with no significant difference are then connected with a horizontal line.

Fig. 4 displays the Nemenyi test concerning the accuracy rate. The more accurate the technique is, the farthest right it is placed. Although SVM have obtained the first place, the horizontal bar indicates both OPF$_{knn}$ and SVM are statistically similar to each other with respect to the accuracy rate. Notice the Nemenyi test considers all datasets as a single experiment

Table 3 presents the mean F-measure results concerning the very same group of datasets. In this case, OPF$_{knn}$ obtained the best

[8] The experiments were conducted on a computer with a Pentium Intel Core i5® 3.2GHz processor, 4GB of memory RAM and Linux Ubuntu Desktop LTS 64 bits 14.04.1 as the operational system.

[9] http://www.ic.unicamp.br/~afalcao/libopf

[10] https://github.com/jppbsi/LibOPF

Table 3
Mean *F*-measure.

Dataset	OPF_{knn}	OPF	SVM
aflw	0.889858	0.881942	**0.940952**
Colon-cancer	**0.815060**	**0.815452**	0.781411
dmoz-web-directory	0.544423	0.529464	**0.592993**
Leukemia	**0.721098**	**0.707009**	0.000000
Pima-Indians-Diabetes	**0.542961**	**0.538063**	**0.504691**
scene-classification	0.611765	0.603329	**0.714909**
Statlog-Australian	0.698748	0.688042	**0.716649**
Statlog-dna	0.840371	0.802822	**0.969121**
Statlog-Heart	**0.549389**	**0.541723**	**0.513663**
Statlog-Letter	0.973181	0.963048	**0.983192**
Statlog-Shuttle	**0.999543**	0.999429	0.998524
Synthetic1	**0.528762**	**0.530372**	**0.497236**
Synthetic2	0.731743	0.721496	**0.809234**
Synthetic3	0.791504	0.780973	**0.843795**
UCI-a1a	0.861291	0.802861	**0.901493**
UCI-Breast-Cancer	**0.969589**	0.966384	**0.971183**
UCI-Connect-4	0.663644	0.589268	**0.832589**
UCI-Ionosphere	0.767735	0.753203	**0.896202**
UCI-Liver-disorders	**0.540719**	0.530601	**0.500406**
UCI-Mushrooms	**1.000000**	0.930392	**0.999910**
UCI-Pendigits	**0.992380**	0.989935	0.980136
UCI-vowel	**0.946560**	0.935293	**0.934548**
usps	0.968006	0.963450	**0.975141**
w1a	0.976360	0.921680	**0.984757**
yahoo-web-directory	0.173338	0.172730	**0.667598**

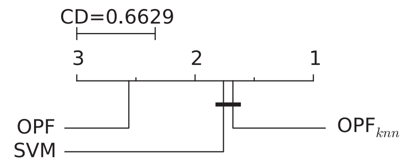

Fig. 5. Nemenyi statistical test concerning the *F*-measure values displayed in Table 3.

results in 11 out of 25 datasets, though not the very same ones with respect to Table 2. Fig. 5 displays the Nemenyi test concerning the *F*-measure. In this case, we can observe OPF_{knn} obtained the best results, but being similarly to SVM. The naïve OPF obtained the last position, although it has been the fastest approach for training purposes, as discussed later.

Fig. 6 depicts the mean computational load considering the training time (log [s]). Clearly, standard OPF is faster than OPF_{knn}, since the latter one needs to estimate the neighborhood size (k^*), and naïve OPF does not have any parameter to be optimized. However, both approaches are faster than SVM, since it requires the fine-tuning parameter step. On average, OPF is about 14.15 times faster than OPF_{knn} for training, and the latter is about 1.17 times faster than SVM for training as well. In addition, Fig. 7 displays the mean execution time considering now the testing time. In this case, OPF_{knn} has demonstrated to be faster in some datasets (i.e., "shuttle" and "Synthetic3"), though with a small difference (actually, OPF_{knn} has been about 1.09 faster than OPF considering the testing phase for all datasets). Although both OPF variants need to run over all training samples first (naïve OPF to check the training sample that is going to conquer the testing node, and OPF_{knn} to find the k-nearest neighbors), OPF_{knn} needs an additional step to find out the training sample that shall offer the optimum-path cost among those in the neighborhood. Since we are using a small k_{max} value ($k_{max} = 20$), such additional step may not take longer. However, for larger datasets, it might be necessary to employ larger k_{max} values, which can lead OPF_{knn} to a more expensive testing step.

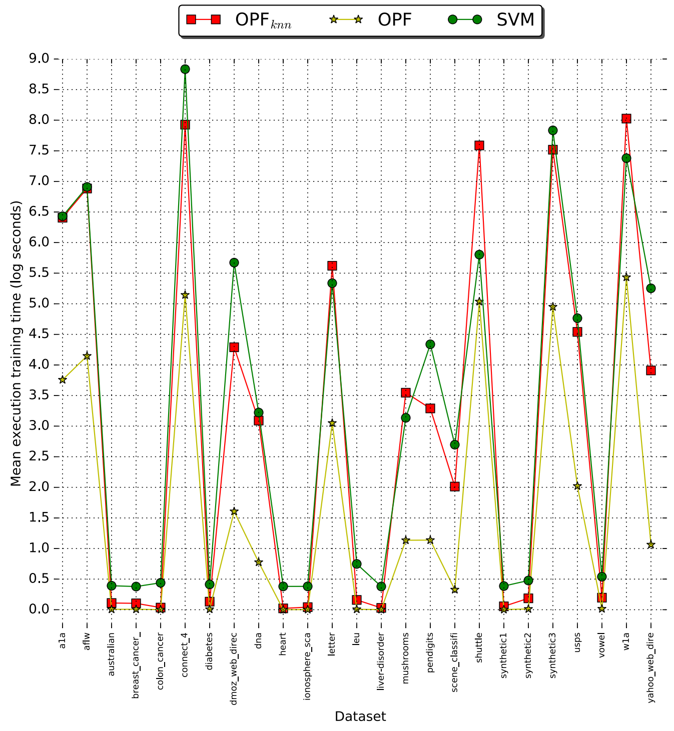

Fig. 6. Mean computational load considering the training phase.

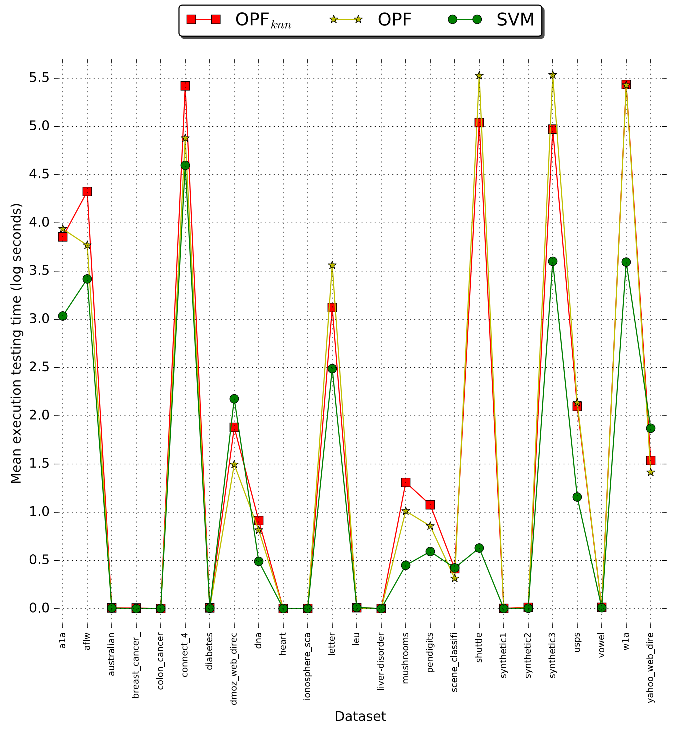

Fig. 7. Mean computational load considering the testing phase.

4.3.2. Evaluating the robustness of $pOPF_{knn}$

We assessed the robustness of $pOPF_{knn}$ (Section 3.2) against OPF_{knn} in two datasets, which were chosen based on their size. The idea is to evaluate $pOPF_{knn}$ in small- and medium/large-sized datasets, say that "Synthetic2" and "scene-classification", respectively. Since the proposed $pOPF_{knn}$ aims at considering only the first training sample in the cost queue for classification purposes, we expect to obtain better results when $k_{max} \rightarrow \infty$. Indeed, such

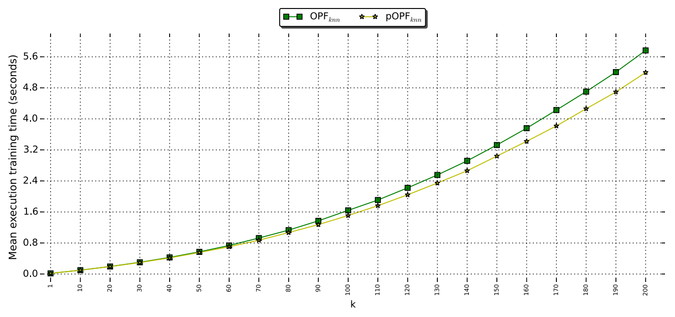

Fig. 8. Mean computational load considering the training phase for pOPF$_{knn}$ over "Synthetic2" dataset.

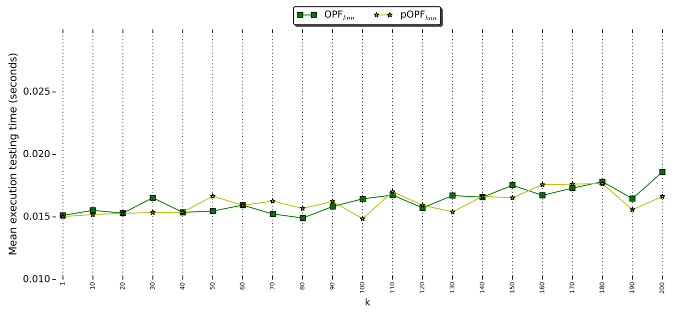

Fig. 9. Mean computational load considering the testing phase for pOPF$_{knn}$ over "Synthetic2" dataset.

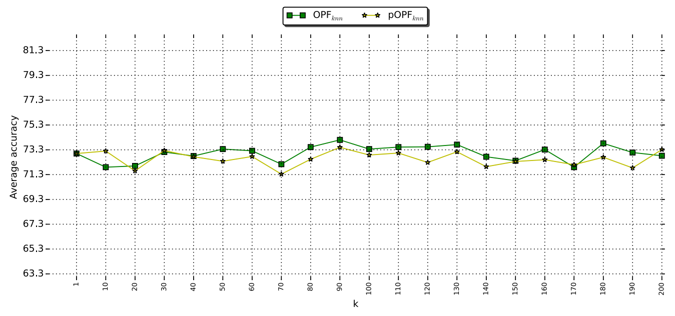

Fig. 10. Mean recognition rate considering the testing phase for pOPF$_{knn}$ over "Synthetic2" dataset.

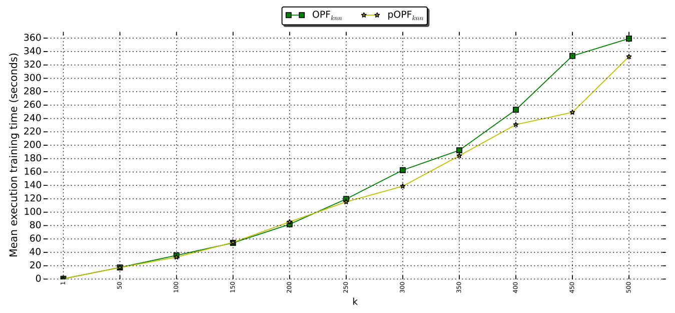

Fig. 11. Mean computational load considering the training phase for pOPF$_{knn}$ over "scene-classification" dataset.

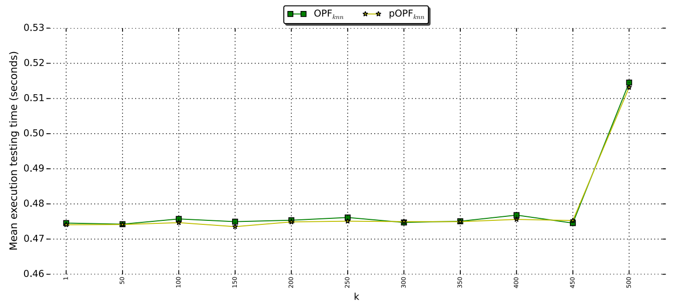

Fig. 12. Mean computational load considering the testing phase for pOPF$_{knn}$ over "scene-classification" dataset.

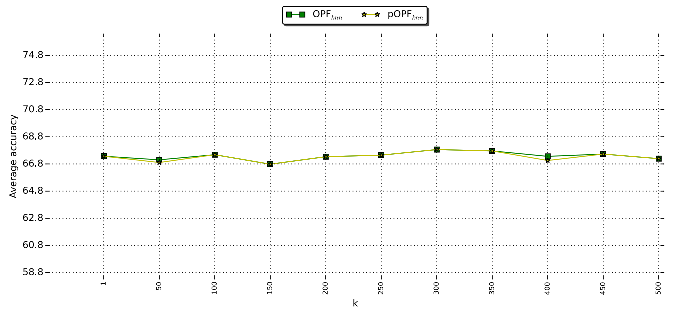

Fig. 13. Mean recognition rate considering the testing phase for pOPF$_{knn}$ over "scene-classification" dataset.

assumption can be observed in Fig. 8, which depicts the mean training time considering $k \in [1, 200]$, i.e., $k_{max} = 200$ over "Synthetic2" dataset. In regard to all experiments addressed in this section, we computed the mean training and testing times, as well as the mean recognition rates over a cross-validation procedure with 10 executions.

Considering Fig. 8, one can realize pOPF$_{knn}$ was around 10.93% faster than OPF$_{knn}$ for training purposes when using $k = 200$. In regard to the testing phase, pOPF$_{knn}$ was around 11.77% faster than OPF$_{knn}$ for the very same value of k (Fig. 9). One can also realize some oscillations in the testing time, since the dataset in charge of this experiment is small-sized, thus allowing small execution times for both pOPF$_{knn}$ e OPF$_{knn}$. This means any concurrent process may interfere in the execution time. However, we considered the average of the execution times over the whole range of k.

Fig. 10 depicts the mean recognition rate over "Synthetic2" dataset. In this case, the recognition rates were quite similar to each other, being the main difference obtained with $k = 120$, where OPF$_{knn}$ was around 1.25% more accurate then pOPF$_{knn}$ only. Although a statistical evaluation with Wilcoxon signed-rank test pointed out a small difference among them, we must stress they were quite close to each other.

In regard to a larger repository, we evaluated both pOPF$_{knn}$ and OPF$_{knn}$ in the "scene-classification" dataset. Fig. 11 displays the mean computational load concerning $k \in [1, 500]$. Since this dataset is larger than the previous one, we opted to employ a larger k_{max} value. Considering $k = 500$, pOPF$_{knn}$ was around 8.10% faster than OPF$_{knn}$.

Fig. 12 depicts the mean execution time concerning the testing phase. In this case, pOPF$_{knn}$ was around 0.26% faster than OPF$_{knn}$ only. In fact, the size of the dataset may not affect the differences between pOPF$_{knn}$ and OPF$_{knn}$, since both need to compute the k-nearest neighbors. The main difference related to pOPF$_{knn}$ concerns with the fact one does not need to run over all neighborhood again to verify the training sample that is going to conquer that testing sample, which is required by OPF$_{knn}$. Finally, Fig. 13 displays the mean recognition rates considering "scene-classification" dataset. Considering this experiment, OPF$_{knn}$ was around 0.3% more accurate than pOPF$_{knn}$ when using $k = 400$. In short, a statistical evaluation conducted with Wilcoxon signed-rank test with respect to the recognition rates pointed out both techniques can be considered similar to each other.

Table 4
Experimental results considering the proposed approach for learning the neighborhood size using meta-heuristics.

Dataset	Approach	Training time (s)	Mean k	Accuracy
Synthetic2	OPF$_{knn}$	5.76	1	72.82
Synthetic2	mOPF$_{knn}$ + *PSO*	1.53	104	72.80
Synthetic2	mOPF$_{knn}$ + *HS*	1.08	104	72.96
scene-classification	OPF$_{knn}$	359.37	1	67.19
scene-classification	mOPF$_{knn}$ + *HS*	21.96	253	67.65
scene-classification	mOPF$_{knn}$ + *PSO*	58.36	253	67.36

4.3.3. *Evaluating the robustness of mOPF*$_{\text{knn}}$

In this section, we evaluated the effectiveness of mOPF$_{knn}$, as well as we also showed it can be so accurate as OPF$_{knn}$. Since naïve OPF$_{knn}$ makes use of an exhaustive search within the range $k \in [1, k_{max}]$ to find out k^*, we proposed to model such problem as an optimization task by meta-heuristics. In fact, one can use any sort of optimization approach. We opted for such bundle of techniques due to their simplicity and elegant solutions. In regard to their configuration, we employed the following setup:

- Harmony Search: 5 harmonies with 30 iterations, $HMCR = 0.7$, $PAR = 0.7$ and $\sigma = 10$. Variables *HMCR* and *PAR*, which stand for "Harmony Memory Considering Rate" and "Pitch Adjusting Rate" are used to guide HS onto the search space, as well as to avoid traps from local optima. Variable σ denotes the "bandwidth" (step size) used within *PAR*.
- Particle Swarm Optimization: 5 particles with 30 iterations, $c_1 = 1.4$, $c_2 = 0.6$ and $w = 0.7$. Variables c_1 and c_2 are used to weight the importance of a possible solution being far or close to the local and global optimum, respectively. Variable w stands for the well-known "inertia weight", which is used as a step size towards better solutions.

Since PSO updates all possible solutions at each iteration, which means it needs to evaluate the fitness function (OPF$_{knn}$ training and classification) whenever a particle changes its position. As such, we shall have $5 \times 30 = 150$ evaluations of the fitness function, plus five more evaluations to initialize the swarm. Therefore, mOPF$_{knn}$+*PSO* will require 155 calls to the OPF$_{knn}$ training and classification functions, meanwhile naïve OPF$_{knn}$ requires 200 calls, since we set $k_{max} = 200$ in the experiments. In regard to mOPF$_{knn}$+*HS*, it requires $5 + 30 = 35$ evaluations of the fitness function only, plus five more to initialize the harmony memory (search space or swarm). Therefore, HS requires 40 calls to the fitness function, since it creates one possible solution at each iteration only.

In any case, both mOPF$_{knn}$+*PSO* and mOPF$_{knn}$+*HS* would be faster than OPF$_{knn}$, since they shall require less computations of the fitness function. However, we would like to evaluate whether both variants have similar recognition rates with respect to OPF$_{knn}$, since they are not considering all possible range of k values within $[1, k_{max}]$. Therefore, that is the meaning of this experimental section. Additionally, the experimental setup was conducted as follows: we partitioned the datasets into three subsets, say that training (30%), validating (20%) and testing sets (50%). The former two sets are used to guide the optimization techniques onto the search space, which means both PSO and HS are looking for the k values that maximize the accuracy over the validating set. Soon after finding out k^*, OPF$_{knn}$ is trained once again over training + validating set. In order to provide a statistical validation, we employed a cross-validation procedure with 10 runnings.

Table 4 presents the mean accuracy over the test set, mean training time (we consider the step for learning parameter k in this computation), as well as the mean k value. Clearly, one can observe the proposed meta-heuristic-based optimization is much faster than naïve OPF$_{knn}$. If we consider "Synthetic" dataset, for instance, OPF$_{knn}$ + *PSO* and OPF$_{knn}$ + *HS* were 3.76 and 5.33 times faster than OPF$_{knn}$ concerning the training step. In regard to "scene-classification" dataset, OPF$_{knn}$ + *PSO* and OPF$_{knn}$ + *HS* were 6.15 and 16.36 times faster than OPF$_{knn}$. Last but not least, a statistical evaluation by means of the Wilcoxon signed-rank test pointed out mOPF$_{knn}$ and OPF$_{knn}$ are similar to each other considering the recognition rates for all variants and datasets considered in this experiment.

An interesting observation concerns with the k values, which are the same for mOPF$_{knn}$ + *PSO* and mOPF$_{knn}$ + *HS* considering the two datasets, but very far from the one found by OPF$_{knn}$. The reason for that concerns with different optima at the landscape of the fitness function. Another small optimization implemented in this work takes into account k values evaluated by OPF$_{knn}$+*PSO* and OPF$_{knn}$+*HS* already. Since we are working with an optimization problem with integer-valued variables, we do not need to recompute the fitness function for those values of k evaluated already. Therefore, mOPF$_{knn}$ can benefit from that.

5. Conclusions

Graph-based pattern recognition techniques are a powerful bundle of tools that can be applied to a number of problems. Since the scientific community has already a well-known and established theory related to those techniques, the applications can naturally benefit from that.

In this work, we considered the Optimum-Path Forest classifier, which is a framework to the development of pattern recognition techniques based on optimal graph partitions. Given an adjacency relation, a path-cost function and a methodology to estimate prototypes, OPF partitions the feature space into optimum-path trees rooted at each prototype. A sample that belongs to a given tree means it is more strongly connected to the root of that tree than to any other in the graph. One can understand OPF as a reward-based competition, in which the prototype that offers the most valuable reward (path-cost) shall conquer the sample.

A number of OPF variables have been proposed in the last years, which range from unsupervised, semi-supervised and supervised learning approaches. In this work, we coped with a supervised variant that employs a k-neighborhood graph, the so-called OPF$_{knn}$, being the main contributions as follows: (i) to provide a deeper explanation and discussion about OPF$_{knn}$, (ii) to propose a meta-heuristic-based approach to estimate the neighborhood size faster and also accurately, and (iii) to present an optimization approach based on a cost queue that makes both training and testing phase more efficient. Experiments over a number of synthetic and real datasets demonstrated the validity of the proposed approaches, as well as the robustness of OPF$_{knn}$ when compared against SVM and to its most used OPF version.

Acknowledgments

The authors are grateful to CAPES PROCAD 2966/2014 grant, FAPESP grants #2009/16206-1, #2013/20387-7 and #2014/2014/16250-9, as well as CNPq grants #303182/2011-3, #70571/2013-6 and #306166/2014-3.

References

[1] C. Alléne, J.-Y. Audibert, M. Couprie, R. Keriven, Some links between extremum spanning forests, watersheds and min-cuts, Image Vis. Comput. 28 (10) (2010) 1460–1471.
[2] Y. Bengio, A.C. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[3] H. Bunke, K. Riesen, Recent advances in graph-based pattern recognition with applications in document analysis, Pattern Recognit. 44 (5) (2011) 1057–1067.

[4] A.M. Carvalho, P. Adão, P. Mateus, Hybrid learning of Bayesian multinets for binary classification, Pattern Recognit. 47 (10) (2014) 3438–3450.
[5] Z. Chen, T. Ellis, A self-adaptive Gaussian mixture model, Comput. Vis. Image Underst. 122 (2014) 35–46.
[6] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[7] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley-Interscience, 2000.
[8] A. Falcão, J. Stolfi, R. de Alencar Lotufo, The image foresting transform: theory, algorithms, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 26 (1) (2004) 19–29.
[9] Z.W. Geem, Music-Inspired Harmony Search Algorithm: Theory and Applications, first ed., Springer Publishing Company, Incorporated, 2009.
[10] J. Hernández-González, I. Inza, J.A. Lozano, Learning Bayesian network classifiers from label proportions, Pattern Recognit. 46 (12) (2013) 3425–3440.
[11] A.S. Iwashita, J.P. Papa, A. Souza, A.X. Falcão, R.A. Lotufo, V.M. Oliveira, V.H.C. de Albuquerque, J.M.R. Tavares, A path- and label-cost propagation approach to speedup the training of the optimum-path forest classifier, Pattern Recognit. Lett. 40 (2014) 121–127.
[12] J. Kennedy, R.C. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.
[13] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[14] M.I. Malinen, R. Mariescu-Istodor, P. Fränti, K-means*: clustering by gradual data transformation, Pattern Recognit. 47 (10) (2014) 3376–3386.
[15] D. Martínez-Rego, O. Fontenla-Romero, A. Alonso-Betanzos, Nonlinear single layer neural network training algorithm for incremental, nonstationary and distributed learning scenarios, Pattern Recognit. 45 (12) (2012) 4536–4546.
[16] P. Nemenyi, Distribution-Free Multiple Comparisons, Princeton University, 1963.
[17] J.P. Papa, A.X. Falcão, A new variant of the optimum-path forest classifier, in: G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, F. Porikli, J. Peters, J. Klosowski, L. Arns, Y. Chun, T.-M. Rhyne, L. Monroe (Eds.), Advances in Visual Computing, Lecture Notes in Computer Science, 5358, Springer Berlin Heidelberg, 2008, pp. 935–944.
[18] J.P. Papa, A.X. Falcão, A learning algorithm for the optimum-path forest classifier, in: A. Torsello, F. Escolano, L. Brun (Eds.), Graph-Based Representations in Pattern Recognition, Lecture Notes in Computer Science, vol. 5534, Springer Berlin Heidelberg, 2009, pp. 195–204.
[19] J.P. Papa, A.X. Falcão, V.H.C. Albuquerque, J.M.R.S. Tavares, Efficient supervised optimum-path forest classification for large datasets, Pattern Recognit. 45 (1) (2012) 512–520.
[20] J.P. Papa, A.X. Falcão, C.T.N. Suzuki, Supervised pattern classification based on optimum-path forest, Int. J. Imaging Syst. Technol. 19 (2) (2009) 120–131.
[21] S. Qasem, S. Shamsuddin, Improving performance of radial basis function network based with particle swarm optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2009, pp. 3149–3156.
[22] L.M. Rocha, F.A.M. Cappabianco, A.X. Falcão, Data clustering as an optimum–path forest problem with applications in image analysis, Int. J. Imaging Syst. Technol. 19 (2) (2009) 50–68.
[23] G.H. Rosa, K.A.P. Costa, L.A. Passos Júnior, J.P. Papa, A.X. Falcão, J.M.R.S. Tavares, On the training of artificial neural networks with radial basis function using optimum-path forest clustering, in: Proceedings of the Twenty Second International Conference on Pattern Recognition, 2014, pp. 1472–1477.
[24] R. Souza, L. Rittner, R. Lotufo, A comparison between k-optimum path forest and k-nearest neighbors supervised classifiers, Pattern Recognit. Lett. 39 (2014) 2–10. Advances in Pattern Recognition and Computer Vision.
[25] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (6) (1945) 80–83.