A Kernel-Based Optimum-Path Forest Classifier

Luis C. S. Afonso[1], Danillo R. Pereira[2], and João P. Papa[3(✉)]

[1] Department of Computing, UFSCar - Federal University of São Carlos, São Carlos, Brazil
sugi.luis@dc.ufscar.br
[2] University of Western São Paulo, Presidente Prudente, Brazil
danilopereira@unoeste.br
[3] School of Sciences, UNESP - São Paulo State University, Bauru, Brazil
papa@fc.unesp.br

Abstract. The modeling of real-world problems as graphs along with the problem of non-linear distributions comes up with the idea of applying kernel functions in feature spaces. Roughly speaking, the idea is to seek for well-behaved samples in higher dimensional spaces, where the assumption of linearly separable samples is stronger. In this matter, this paper proposes a kernel-based Optimum-Path Forest (OPF) classifier by incorporating kernel functions in both training and classification steps. The proposed technique was evaluated over a benchmark comprised of 11 datasets, whose results outperformed the well-known Support Vector Machines and the standard OPF classifier for some situations.

Keywords: Optimum-Path Forest · Kernel · Support Vector Machines

1 Introduction

The nature of real-world problems has driven their modeling to structured objects where samples are usually represented by nodes and their relationship are represented by connections (edges), similarly to a graph [1,2]. Among these problems, one can mention studies in bioinformatics, social networks, and data mining in documents, just to name a few, in which it is necessary to identify structures or elements that share similar properties.

Another very common characteristic present in real-world problems is the non-linear distribution, which requires complex models to identify the different existing patterns in the dataset [3]. In this matter, kernels have been proposed as a tool to overcome this issue by providing a way to map the data into a space of higher dimension, where the input data may be linearly separable.

Support Vector Machines (SVM) is perhaps the most well-known technique that makes use of kernels for data embedding into higher dimensional feature spaces [4]. There are also the kernel-PCA (Principal Component Analysis) [5],

The authors are grateful to Capes, CNPq grant #306166/2014-3, and FAPESP grants #2014/16250-9, #2014/12236-1, and #2016/19403-6.

M. Mendoza and S. Velastín (Eds.): CIARP 2017, LNCS 10657, pp. 652–660, 2018.
https://doi.org/10.1007/978-3-319-75193-1_78

and the kernel-Fisher discriminant analysis [6], just to mention a few. Although the mapping can take a feature space to a very high dimension that increases the computing cost, both training and classification steps can be performed by means of a dot product, the so-called *kernel trick*. Such procedure consists in computing the dot product without the need of mapping the samples to a higher dimensional space. The combination of the these two worlds, i.e., representations based on graphs and non-linear distributions, has motivated studies in the application of kernels in structured data [1,7]. In this research area, it is typical to find two different approaches, i.e., applications that make use of kernels to identify similarity among graphs, while others focus on the similarity among the graph nodes.

One of the first works on kernels among graphs was proposed by Gärtner et al. [8] and Borgwardt et al. [9] using the random walk, and marginalized kernels by Tsuda et al. [10], Kashima et al. [11,12] and Mahé et al. [13]. The basic idea of random-walk graph kernels is to perform a random walk in a pair of graphs and sum up the number of matching walks [2] on both graphs. The marginalized kernel is defined as the inner product of the count vectors averaged over all possible label paths [11]. Regarding kernels in graphs, interesting works were proposed by Kondor and Lafferty [1] and Smola and Kondor [14]. In [1], the authors proposed the diffusion kernels, which is a special class of exponential kernels based on the heat equation.

Graph-based machine learning techniques can be noticed as well. The Optimum-Path Forest (OPF) framework [15–18] is a graph-based classification algorithm where samples are represented as graph nodes and each node can be also weighted by a probability density function that takes into account its k-nearest neighbors, and the edges are weighted by a distance function. The partitioning using the OPF algorithm is performed by a competition process, in which prototypes try to conquer the remaining samples by offering them paths with optimum cost. The result is a set of trees (forest) rooted at a set of prototypes, where each tree may represent a single class or cluster. The OPF algorithm has been applied in a large variety of applications, such as network security [19], Parkinson's disease identification [20], clustering [21,22], and biometrics [23,24], just to name a few, and it has showed competitive results against some state-of-art and well-known machine learning algorithms.

The current OPF algorithm implementation works naturally with non-linear situations, but it does not map samples from one space to another. In this paper, we take one step further by modifying the OPF algorithm to work with kernels on graphs in order to improve its training and classification results, since such approach has not been applied so far. The performance of the proposed approach is assessed under three different kernels, and it is compared against the original OPF algorithm and the well-known SVM in 11 different datasets. The remainder of this paper is organized as follows. Sections 2 and 3 present the theoretical background concerning OPF and its kernel-based variant, respectively. Section 4 discusses the methodology and experiments, and Sect. 5 states the conclusions.

2 Optimum-Path Forest

In this section, we explain the OPF working mechanism considering the first proposed version [16,17]. Roughly speaking, the OPF classifier models the problem of pattern recognition as a graph partition in a given feature space. The nodes are represented by the feature vectors and the edges connect all pairs of them, defining a full connectedness graph. The partition of the graph is performed through a competition process among some key samples (prototypes), which offer optima paths to the remaining nodes of the graph. Each prototype sample defines its own optimum-path tree (OPT), and the collection of all OPTs defines an optimum-path forest, which gives the name to the classifier.

Let $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2$ be a dataset labeled with a function λ, in which $\mathcal{Z}_1$ and $\mathcal{Z}_2$ stand for the training and test sets, respectively, such that $\mathcal{Z}_1$ is used to train a given classifier and $\mathcal{Z}_2$ is used to assess its accuracy. Let $\mathcal{S} \subseteq \mathcal{Z}_1$ be a set of prototype samples. Essentially, the OPF classifier creates a discrete optimal partition of the feature space such that any sample $\mathbf{s} \in \mathcal{Z}_2$ can be classified according to this partition.

The OPF algorithm may be used with any *smooth* path-cost function which can group samples with similar properties [25]. Papa et al. [16,17] employed the path-cost function f_{max}, which is computed as follows:

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

in which $d(\mathbf{s}, \mathbf{t})$ denotes the distance between samples $\mathbf{s}$ and $\mathbf{t}$, and a path π is defined as a sequence of adjacent samples. As such, we have that $f_{max}(\pi)$ computes the maximum distance among adjacent samples in π, when π is not a trivial path.

The OPF algorithm assigns one optimum path $P^*(\mathbf{s})$ from $\mathcal{S}$ to every sample $\mathbf{s} \in \mathcal{Z}_1$, forming an optimum path forest P (a function with no cycles that assigns to each $s \in \mathcal{Z}_1 \backslash \mathcal{S}$ its predecessor $P(\mathbf{s})$ in $P^*(\mathbf{s})$ or a marker *nil* when $\mathbf{s} \in \mathcal{S}$. Let $R(\mathbf{s}) \in \mathcal{S}$ be the root of $P^*(\mathbf{s})$ that can be reached from $P(\mathbf{s})$. The OPF algorithm computes for each $\mathbf{s} \in \mathcal{Z}_1$, the cost $C(\mathbf{s})$ of $P^*(\mathbf{s})$, the label $L(\mathbf{s}) = \lambda(R(\mathbf{s}))$, and the predecessor $P(\mathbf{s})$.

2.1 Training

In the training phase, the OPF algorithm aims to find the set $\mathcal{S}^*$, that is the optimum set of prototypes, by minimizing the classification errors for every $\mathbf{s} \in \mathcal{Z}_1$ through the exploitation of the theoretical relation between minimum-spanning tree (MST) and optimum-path tree (OPT) for f_{max} [26]. The training essentially consists in finding $\mathcal{S}^*$ from $\mathcal{Z}_1$ and an OPF classifier rooted at $\mathcal{S}^*$.

By computing a MST, we obtain a connected acyclic graph whose nodes are all samples of $\mathcal{Z}_1$ and the arcs are undirected and weighted by the distances d between adjacent samples. The spanning tree is optimum in the sense that

the sum of its arc weights is minimum as compared to any other spanning tree in the complete graph. In the MST, every pair of samples is connected by a single path which is optimum according to f_{max}. That is, the minimum-spanning tree contains one optimum-path tree for any selected root node. The optimum prototypes are the closest elements of the MST with different labels in $\mathcal{Z}_1$ (i.e., elements that fall in the frontier of the classes). After finding the prototypes, the competition process takes place to build the optimum-path forest.

2.2 Classification

For any sample $\mathbf{t} \in \mathcal{Z}_2$, we consider all arcs connecting $\mathbf{t}$ with samples $\mathbf{s} \in \mathcal{Z}_1$, as though $\mathbf{t}$ were part of the training graph. Considering all possible paths from $\mathcal{S}^*$ to $\mathbf{t}$, we find the optimum path $P^*(\mathbf{t})$ from $\mathcal{S}^*$ and label $\mathbf{t}$ with the class $\lambda(R(\mathbf{t}))$ of its most strongly connected prototype $R(\mathbf{t}) \in \mathcal{S}^*$. This path can be identified incrementally by evaluating the optimum cost $C(\mathbf{t})$ as

$$C(\mathbf{t}) = \min\{\max\{C(\mathbf{s}), d(\mathbf{s}, \mathbf{t})\}\}, \ \forall \mathbf{s} \in \mathcal{Z}_1. \tag{2}$$

Let the node $\mathbf{s}^* \in \mathcal{Z}_1$ be the one that satisfies Eq. 2 (i.e., the predecessor $P(\mathbf{t})$ in the optimum path $P^*(\mathbf{t})$). Given that $L(\mathbf{s}^*) = \lambda(R(\mathbf{t}))$, the classification simply assigns $L(\mathbf{s}^*)$ as the class of $\mathbf{t}$. An error occurs when $L(\mathbf{s}^*) \neq \lambda(\mathbf{t})$.

3 Proposed Approach

The proposed kernel-based OPF, hereinafter called kOPF, works similarly to SVM algorithm, in which samples are mapped into a feature space of higher dimension. In the SVM context, such mapping is performed as an attempt to make the data linearly separable. The OPF, on the other hand, naturally works with non-linear data. Therefore, the main idea of this work is to evaluate OPF's behavior under such assumption of samples' separability in higher dimensional spaces.

Let $\Phi(\cdot, \cdot)$ be a kernel function that generates a new dataset $\mathcal{X} = \Phi(\mathcal{Z})$. Given a sample $\mathbf{p} \in \mathcal{Z}$, such that $\mathbf{p} \in \Re^n$, its new representation $\hat{\mathbf{p}} \in \mathcal{X}$ is defined as follows.

$$\hat{\mathbf{p}} = (\Phi_1, \Phi_2, \ldots, \Phi_{|\mathcal{Z}_1|}), \tag{3}$$

where $\Phi_i = (\mathbf{p}, \mathbf{s}_i)$, $\mathbf{s}_i \in \mathcal{Z}_1$. Notice that $\hat{\mathbf{p}} \in \Re^{|\mathcal{Z}_1|}$, which means the new sample $\mathbf{p}$ contains as many dimensions as the number of training samples.

In short, $\Phi(\mathbf{p}, \mathbf{s}_i)$ makes use of a distance function (i.e., Euclidean, Mahalanobis, among others) to compute a term that replaces the norm in kernel functions, such as Radial Basis Function (RBF) and Sigmoid, for instance. The aforementioned term corresponds to the distance between a sample to be mapped $\mathbf{p}$ and a training sample $\mathbf{s}$. Basically, the mapping performed by kOPF is carried out by computing a feature vector, where each component has the distance value from the sample to be mapped (either training or testing sample) to a different

training sample applied to a kernel function. It is important to highlight that large training sets may cause a significant increase on the size of the new feature vector, since the size is dependent on the number of training samples $|\mathcal{Z}_1|$.

4 Methodology and Experiments

The kOPF classifier has both its performance and accuracy assessed by means of 11 public benchmarking datasets[1] that provide different classification scenarios. The implementation of our proposed approach is developed over the LibOPF [27], being standard OPF and SVM used as baselines for the experiments. With respect to SVM, we used the well-known LibSVM[2]. Table 1 presents detailed information from the datasets.

Table 1. Information about the datasets used in the experiments.

Dataset	No. samples	No. features	No. classes
boat (bt)	100	2	3
cone-torus (ct)	400	2	3
data1 (d1)	1,423	2	2
data2 (d2)	283	2	2
data3 (d3)	340	2	5
data4 (d4)	698	2	3
data5 (d5)	1,850	2	2
mpeg7-BAS (m-B)	1,400	180	70
mpeg7-Fourier (m-F)	1,400	126	70
petals (ps)	100	2	4
saturn (sn)	200	2	2

Since the kernel function can influence the final accuracy, we evaluated its impact by applying three different kernel functions for kOPF as follows:

- Identity: $\Phi(\mathbf{p}, \mathbf{s}) = \|\mathbf{p}, \mathbf{s}\|$
- RBF: $\Phi(\mathbf{p}, \mathbf{s}) = e^{-(\gamma\|\mathbf{p}, \mathbf{s}\|^2)}$
- Sigmoid: $\Phi(\mathbf{p}, \mathbf{s}) = \tanh(\gamma\|\mathbf{p}, \mathbf{s}\| + C)$

where $\|\mathbf{p}, \mathbf{s}\|$ denotes the Euclidean distance. Notice the Identity kernel is parameterless. The SVM was also evaluated using three different kernel functions: linear, RBF and Sigmoid. The situations in which parameters C and γ are required, it is performed their optimization using the intervals $C = [-32, 32]$ and $\gamma = [0, 32]$ with steps equals to 2 for both of them.

[1] https://github.com/jppbsi/LibOPF.
[2] https://www.csie.ntu.edu.tw/~cjlin/libsvm.

The classification experiments were conducted by means of a hold-out process using 15 runs, in which both training and testing sets were randomly generated in each run and always having a number of samples equals to 50% of the dataset size. The experiments also evaluated the impact in the accuracy rate when features are normalized. Tables 2 and 3 present the mean accuracy results considering non-normalized and normalized datasets, respectively. The accuracy rates were computed using the accuracy measure proposed by Papa et al. [16], which considers unbalanced data. The best results according to the Wilcoxon signed-rank test with significance 0.05 are shown in bold.

Table 2. Mean classification rates for non-normalized features.

Dataset	OPF	*k*OPF			SVM		
		Identity	RBF	Sigmoid	Linear	RBF	Sigmoid
bt	98.5 ± 0.0	96.8 ± 3.0	**100.0 ± 0.0**	99.1 ± 1.2	76.9 ± 1.8	**100.0 ± 0.0**	76.9 ± 1.8
ct	86.2 ± 0.0	84.3 ± 2.6	84.5 ± 2.7	84.2 ± 2.4	74.9 ± 0.2	**86.5 ± 1.1**	74.4 ± 0.4
d1	99.5 ± 0.0	**99.4 ± 0.2**	67.2 ± 8.6	68.5 ± 14.5	94.8 ± 0.8	**99.4 ± 0.3**	94.0 ± 0.8
d2	**99.3 ± 0.0**	98.1 ± 1.1	57.0 ± 2.6	62.0 ± 6.1	97.1 ± 0.5	98.2 ± 0.2	81.3 ± 3.6
d3	99.6 ± 0.0	**99.7 ± 0.4**	64.0 ± 3.6	71.1 ± 5.4	98.4 ± 0.8	**99.7 ± 0.4**	97.9 ± 0.7
d4	**100.0 ± 0.0**	**100.0 ± 0.0**	60.8 ± 2.6	67.8 ± 6.4	**100.0 ± 0.0**	**100.0 ± 0.0**	50.9 ± 1.3
d5	**100.0 ± 0.0**	**100.0 ± 0.0**	62.5 ± 3.2	67.5 ± 9.3	50.0 ± 0.0	**100.0 ± 0.0**	50.0 ± 0.0
m-B	89.4 ± 0.0	89.0 ± 0.30	50.1 ± 0.1	50.1 ± 0.1	87.9 ± 0.2	**90.5 ± 0.2**	50.0 ± 0.0
m-F	**73.0 ± 0.0**	**73.0 ± 0.5**	64.4 ± 0.6	66.2 ± 0.5	69.0 ± 0.8	**73.0 ± 0.5**	68.1 ± 0.5
ps	98.7 ± 0.0	**100.0 ± 0.0**	99.2 ± 0.6	98.9 ± 1.0	99.6 ± 0.6	**100.0 ± 0.0**	**100.0 ± 0.0**
sn	**93.0 ± 0.0**	90.2 ± 2.0	80.8 ± 2.6	81.8 ± 4.0	42.7 ± 3.7	91.3 ± 1.9	49.0 ± 3.3

Table 3. Mean classification rates for normalized features.

Dataset	OPF	*k*OPF			SVM		
		Identity	RBF	Sigmoid	Linear	RBF	Sigmoid
bt	97.1 ± 0.0	99.7 ± 0.6	99.4 ± 1.2	**100.0 ± 0.0**	76.9 ± 1.8	99.5 ± 0.7	76.5 ± 1.2
ct	89.3 ± 0.0	86.7 ± 1.7	88.0 ± 1.2	**89.9 ± 0.5**	75.9 ± 1.6	**89.4 ± 0.5**	76.1 ± 1.6
d1	**99.3 ± 0.0**	99.0 ± 0.3	99.0 ± 0.3	99.0 ± 0.3	94.7 ± 0.4	**99.3 ± 0.2**	94.7 ± 0.4
d2	97.3 ± 0.0	96.9 ± 1.5	97.1 ± 1.2	97.0 ± 1.3	**98.8 ± 0.9**	**98.6 ± 0.6**	**98.6 ± 0.6**
d3	**100.0 ± 0.0**	98.4 ± 1.4	**100.0 ± 0.0**	98.7 ± 0.8	99.5 ± 0.4	99.3 ± 0.7	99.5 ± 0.4
d4	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
d5	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	50.0 ± 0.0	**100.0 ± 0.0**	50.7 ± 0.9
m-B	88.8 ± 0.0	**90.8 ± 0.3**	53.2 ± 0.5	56.3 ± 0.4	87.7 ± 0.9	89.9 ± 0.6	87.6 ± 0.9
m-F	62.9 ± 0.0	62.5 ± 0.4	59.2 ± 0.7	61.1 ± 0.8	64.7 ± 0.7	**65.4 ± 0.5**	64.3 ± 0.9
ps	98.7 ± 0.0	**100.0 ± 0.0**	99.5 ± 0.6	99.5 ± 1.0	**100.0 ± 0.0**	99.6 ± 0.6	**100.0 ± 0.0**
sn	91.0 ± 0.0	**91.8 ± 0.2**	79.2 ± 4.8	84.6 ± 0.8	52.3 ± 1.7	90.7 ± 4.5	50.7 ± 3.4

In the non-normalized feature scenario, SVM-RBF achieved the best results (or similar) in 9 out of 11 datasets, followed by the *k*OPF (*k*OPF-Identity and *k*OPF-RBF) with 7 out of 11 (being *k*OPF-Identity the best in 6 out of 11).

The standard OPF obtained the best results in 5 out of 11 datasets. Considering normalized features, the kOPF (kOPF-Identity, kOPF-RBF and kOPF-Sigmoid) obtained the best results (or similar) in 8 out of 11 datasets (being kOPF-Identity the best in 5 out of 11), followed by SVM (SVM-Linear, SVM-RBF and SVM-Sigmoid) with 7 out of 11 (being SVM-RBF the best in 6 out of 11). The OPF obtained the best results in only 4 out of 11 datasets.

In both (non-normalized and normalized) scenarios, the proposed kOPF outperformed the traditional OPF in most datasets, and for normalized features kOPF outperformed the SVM in some datasets as well. The results are quite interesting, since kOPF was able to improve OPF and outperforming SVM in some datasets. Considering some other datasets, although kOPF did not outperform both OPF and SVM, their results were considerably close.

The experiments also comprised the analysis of computational load required by each technique in each dataset. The results showed OPF and kOPF require a considerably small computational load in the training phases when compared against SVM. The high training time consumption turn the SVM prohibitive in real-time learning systems, specially if the training set is very dynamic over time. In this situation, both OPF and kOPF seems to be the most suitable approach. Due to lack of space, it was not possible to insert the computational load results of each technique, but OPF-based approaches have been around 200 times faster than SVM for training in the larger datasets.

5 Conclusions

This paper introduced a kernel-based OPF, which is a modification made over the standard OPF classifier that allows the usage of different kernel functions for both learning and classification. In our proposed approach, the mapping makes use of distance metric whose results are applied to kernel functions, such as RBF and Sigmoid. The main goal of such modification is to improve the accuracy rate.

The evaluation using 11 benchmark datasets and three different kernels showed the proposed approach achieved very interesting results, in which the application of kernel functions improved the accuracy rate of the traditional OPF, and even outperformed the well-known SVM when features were normalized. In summary, kOPF achieved satisfactory results and is an interesting option for classification, specially when training sets are very dynamic due to its low computational load for training purposes.

References

1. Kondor, R.I., Lafferty, J.D.: Diffusion kernels on graphs and other discrete input spaces. In: Proceedings of the Nineteenth International Conference on Machine Learning, ser. ICML 2002, pp. 315–322. Morgan Kaufmann Publishers Inc., San Francisco (2002)
2. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. J. Mach. Learn. Res. **11**, 1201–1242 (2010)

3. Hofmann, T., Schölkopf, B., Smola, A.J.: Kernel methods in machine learning. Ann. Stat. **36**(3), 1171–1220 (2008)
4. Burges, C.J.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. **2**(2), 121–167 (1998)
5. Schölkopf, B., Smola, A., Müller, K.-R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**(5), 1299–1319 (1998)
6. Mika, S., Rtsch, G., Weston, J., Schlkopf, B., Mller, K.-R.: Fisher discriminant analysis with kernels (1999)
7. Gärtner, T.: A survey of kernels for structured data. SIGKDD Explor. Newsl. **5**(1), 49–58 (2003)
8. Gärtner, T., Flach, P., Wrobel, S.: On graph kernels: hardness results and efficient alternatives. In: Schölkopf, B., Warmuth, M.K. (eds.) COLT-Kernel 2003. LNCS (LNAI), vol. 2777, pp. 129–143. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45167-9_11
9. Borgwardt, K.M., Ong, C.S., Schnauer, S., Vishwanathan, S.V.N., Smola, A.J., Kriegel, H.-P.: Protein function prediction via graph kernels. Bioinformatics **21**, i47 (2005)
10. Tsuda, K., Kin, T., Asai, K.: Marginalized kernels for biological sequences. Bioinformatics **18**, S268 (2002)
11. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 321–328. AAAI Press (2003)
12. Kashima, H., Tsuda, K., Inokuchi, A.: Kernels for Graphs. MIT Press, Cambridge (2004)
13. Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L., Vert, J.-P.: Extensions of marginalized graph kernels. In: Proceedings of the Twenty-First International Conference on Machine Learning, ser. ICML 2004, p. 70. ACM, New York (2004)
14. Smola, A.J., Kondor, I.R.: Kernels and regularization on graphs. In: Proceedings of the Annual Conference on Computational Learning Theory (2003)
15. Rocha, L.M., Cappabianco, F.A.M., Falcão, A.X.: Data clustering as an optimum-path forest problem with applications in image analysis. Int. J. Imaging Syst. Technol. **19**(2), 50–68 (2009)
16. Papa, J.P., Falcão, A.X., Suzuki, C.T.N.: Supervised pattern classification based on optimum-path forest. Int. J. Imaging Syst. Technol. **19**(2), 120–131 (2009)
17. Papa, J.P., Falcão, A.X., Albuquerque, V.H.C., Tavares, J.M.R.S.: Efficient supervised optimum-path forest classification for large datasets. Pattern Recogn. **45**(1), 512–520 (2012)
18. Papa, J.P., Fernandes, S.E.N., Falcão, A.X.: Optimum-path forest based on k-connectivity: theory and applications. Pattern Recogn. Lett. **87**, 117–126 (2017)
19. Pereira, C.R., Nakamura, R.Y., Costa, K.A., Papa, J.P.: An optimum-path forest framework for intrusion detection in computer networks. Eng. Appl. Artif. Intell. **25**(6), 1226–1234 (2012)
20. Spadoto, A.A., Guido, R.C., Papa, J.P., Falco, A.X.: Parkinson's disease identification through optimum-path forest. In: 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology, pp. 6087–6090, August 2010
21. Pisani, R., Riedel, P., Ferreira, M., Marques, M., Mizobe, R., Papa, J.: Land use image classification through optimum-path forest clustering. In: 2011 IEEE International Geoscience and Remote Sensing Symposium, pp. 826–829, July 2011
22. Afonso, L., Papa, J., Papa, L., Marana, A., Rocha, A.: Automatic visual dictionary generation through optimum-path forest clustering. In: 2012 19th IEEE International Conference on Image Processing, pp. 1897–1900, September 2012

23. Chiachia, G., Marana, A.N., Papa, J.P., Falcao, A.X.: Infrared face recognition by optimum-path forest. In: 2009 16th International Conference on Systems, Signals and Image Processing, pp. 1–4, June 2009
24. Papa, J.P., Falcao, A.X., Levada, A.L.M., Correa, D.C., Salvadeo, D.H.P., Mascarenhas, N.D.A.: Fast and accurate holistic face recognition using optimum-path forest. In: 2009 16th International Conference on Digital Signal Processing, pp. 1–6, July 2009
25. Falcão, A., Stolfi, J., Lotufo, R.: The image foresting transform theory, algorithms, and applications. IEEE Trans. Pattern Anal. Mach. Intell. **26**(1), 19–29 (2004)
26. Allène, C., Audibert, J.-Y., Couprie, M., Keriven, R.: Some links between extremum spanning forests, watersheds and min-cuts. Image Vis. Comput. **28**(10), 1460–1471 (2010). Image Analysis and Mathematical Morphology
27. Papa, J.P., Suzuki, C.T.N., Falcão, A.X.: LibOPF: a library for the design of optimum-path forest classifiers (2014). http://www.ic.unicamp.br/~afalcao/libopf/