On the Training of Artificial Neural Networks with Radial Basis Function Using Optimum-Path Forest Clustering

Gustavo H. Rosa, Kelton A. P. Costa,
Leandro A. Passos Júnior, João P. Papa
Department of Computing
São Paulo State University
Bauru, São Paulo, Brazil
{gustavo.rosa,kelton,leandropassosjr,papa}@fc.unesp.br

Alexandre X. Falcão
Institute of Computing
University of Campinas
Campinas, São Paulo, Brazil
afalcao@ic.unicamp.br

João Manuel R. S. Tavares
Faculdade de Engenharia
Universidade do Porto
Porto, Portugal
tavares@fe.up.pt

***Abstract*—In this paper, we show how to improve the Radial Basis Function Neural Networks effectiveness by using the Optimum-Path Forest clustering algorithm, since it computes the number of clusters on-the-fly, which can be very interesting for finding the Gaussians that cover the feature space. Some commonly used approaches for this task, such as the well-known k-means, require the number of classes/clusters previous its performance. Although the number of classes is known in supervised applications, the real number of clusters is extremely hard to figure out, since one class may be represented by more than one cluster. Experiments over 9 datasets together with statistical analysis have shown the suitability of OPF clustering for the RBF training step.**

***Index Terms*—Artificial Neural Networks, Radial Basis Function, Optimum-Path Forest**

I. Introduction

Machine learning techniques have been extensively studied in the last decades. Improvements in their mathematical formulation and implementation through computer programs have led to the development of even better approaches to handle the problem of finding separating decision functions, mainly in the context of large datasets.

The introduction of the well-known *Perceptron* by the seminal work of Rosenblatt [1] has started a research race to answer the following question: "How does the brain work?". However, even after several years, this concern remains an open problem. Artificial Neural Networks (ANNs) have arisen to boost the original idea of Perceptron, in which a collection of neurons can process the information and propagate input signals from one layer to another. A famous approach is the ANN with Multilayer Perceptrons (ANN-MLP), in which a common architecture composed by three main layers (input, hidden and output) tries to model the complex and intrinsic brain working model [2]. Since the idea is to separate the feature space such that samples with similar properties will belong to the same cluster, an ANN-MLP employ linear decision boundaries to address such a task. Complex separating functions can be obtained using hidden layers with more neurons [3].

Neural networks with Radial Basis Function (RBF) are also another interesting kind of networks which employs a three-layered architecture for pattern recognition and regression problems. The idea is to submit the input data to a non-linear mapping performed by radial basis functions in the hidden layer, and after that a linear combination of the hidden layer outputs is then employed in the output layer [3]. Artificial Neural Networks with Radial Basis Function (ANN-RBF) also seek to separate the feature space as ANN-MLP does, but now different decision boundaries are designed. Usually, such sort of neural networks can be faster than ANN-MLP ones, since their training step is simpler and can be conducted by using several approaches, being one of them a single-step solution based on pseudo-inverse solution.

The basic idea of an artificial neural network with radial basis function is to perform a training step followed by classification, being the former procedure composed by two phases: (i) an unsupervised one, which is responsible to find out the radial function's parameters, and (ii) a supervised step, which performs the non-linear mapping of the input vector for further linear weight combination. Among the wide variety of radial basis functions, the most employed is the well-known Gaussian function, which has a simple and effective formulation. However, the main shortcoming of ANN-RBF networks is the requirement of a good space coverage by Gaussian functions, which turns ANN-RBF extremely dependent on the effectiveness of the clustering approach that aims to find the parameters of the Gaussian functions, i.e., mean and variance (the later one is commonly estimated though a closed equation). Despite most implementations employ the well-known k-means for such task, which is simple and easy to be implemented, in real applications, it is not so easy to know the number of clusters, as required by k-means. Although the

The authors would like to thank FAPESP grants #2009/16206-1, #2013/05513-6 and 2013/20387-7, CAPES and also CNPq grants #303182/2011-3, #470571/2013-6, #3479070/2013-0 and #303673/2010-9. This work was partially done in the scope of the project "A novel framework for Supervised Mobile Assessment and Risk Triage of Skin lesions via Non-invasive Screening", with reference PTDC/BBB-BMD/3088/2012, financially supported by Fundação para a Ciência e a Tecnologia (FCT) in Portugal.

1051-4651/14 $31.00 © 2014 IEEE
DOI 10.1109/ICPR.2014.262

user may have the knowledge of the number of classes, each of them may be represented by more than one Gaussian function. Additionally, the k value can be estimated through a validating set, but it may be impractical for large datasets.

Nowadays, many works have addressed the problem of finding Gaussian's parameters using optimization approaches. Esmaeili and Mozayani [4], for instance, employed Particle Swarm Optimization (PSO) for this task. Tsekouras and Tsimikas [5] proposed a hybrid approach for ANN-RBF training using PSO and a fuzzy-based clustering approach. A Memetic-based algorithm using the concept of Differential Evolution has been also used for the same context by Qasema and Shamsuddina [6]. Although such optimization-based approaches are interesting and widely employed for RBF training, it is important to point out some drawbacks: (i) they can be trapped in local optima, (ii) some of them have a high computational burden and may require several iterations for convergence, and (iii) evolutionary-based algorithms often require several parameters as input, making necessary to find out reasonable values for each of them, which can be empirically done or even with meta-optimization techniques, increasing the complexity of the whole system.

Recently, Rocha et al. [7] proposed an interesting approach for data clustering based on the Optimum-Path Forest (OPF) methodology, which models the clustering task as a graph partition problem, where the samples are the nodes and a predefined adjacency relation connects them. The graph partition is ruled by a competition between key nodes (prototypes) using a path-cost function, being each of them the root of its optimum-path tree (cluster), which contains the conquered nodes. The OPF has demonstrated better results than traditional Mean-Shift algorithm [7], and can find out the number of clusters on-the-fly, i.e., it does not require the number of clusters as an input, such as k-means, for instance, being an interesting tool for RBF networks training. Additionally, the prototypes estimated by OPF, which will encode the mean values of the Gaussian distributions, are positioned in the center (or nearby regions) of the clusters (high density regions), being located at more representative positions than the ones computed by k-means.

Therefore, this paper introduces the OPF clustering for RBF networks training, which is compared against with traditional k-means in several synthetic and public datasets. The experiments have been statistically analyzed, showing the robustness of OPF clustering in situations in which k-means would fail, such as in classes represented by more than one Gaussian distribution (which may really happen in practice). The remainder of this paper is organized as follows. Section II presents the background theory related with RBF networks and OPF clustering. Sections III and IV describe the proposed approach for finding suitable Gaussian's centers using OPF and the experimental section, respectively. Finally, conclusions are stated in Section V.

II. Theoretical Background

A. Artificial Neural Networks with Radial Basis Function

Artificial Neural Networks with Radial Basis Function are one of the most used machine learning techniques, which can be seen as a multilayer neural network composed by three layers: input, hidden and an output. The first layer receives the input vector, which is submitted to a non-linear transformation in the hidden layer. Finally, a linear combination of the outputs of the hidden layer is performed in the last layer.

Given an input vector $x \in \mathbb{R}^n$, the outputs of the RBF neural network can be computed as follows:

$$y_j = \sum_{i=1}^{m} h_i w_{i,j}, \ j = 1, \ldots, p, \tag{1}$$

in which m and p denote the number of hidden neurons (gaussians) and the number of output neurons, respectively, $w_{i,j} \in \mathbb{R}$ stands for the weight of the connection between between the hidden neuron i and the output neuron j, and $h \in \mathbb{R}^m$ denotes a vector of the outputs of the radial basis functions. In regard to RBF neural networks, h can be composed by any sort of radial basis function, being the most common the n-dimensional Gaussian:

$$h_j(x|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{n/2} \left|\Sigma_j\right|^{1/2}} e^{\left(-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)\right)}, \tag{2}$$

where $\mu_j \in \mathbb{R}^n$ and Σ_j stand for the centers (mean vector) and covariance matrix of gaussian j, respectively. Notice $|\Sigma|$ stands for the determinant of Σ. As such, the ANN-RBF training step aims to find out suitable values for μ, Σ and the weights w.

Basically, μ and Σ are computed through unsupervised learning. Commonly, it is assumed an isotropic distribution, which means we can use the same variance for all Gaussians. Among several possibilities for that purpose, we adopted the following:

$$\sigma = 2D, \tag{3}$$

in which D stands for the average distance between centers. As we are working with n-dimensional Gaussian distributions (Equation 2), we can build diagonal covariance matrices with size $n \times n$, as follows:

$$\Sigma_i = \begin{bmatrix} \sigma & 0 & \ldots & 0 \\ 0 & \sigma & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & \sigma \end{bmatrix}, \tag{4}$$

in which Σ_i denotes the covariance matrix for Gaussian distribution i. Regarding the weights w, there are some interesting approaches such as the pseudo-inverse method and the generalized delta rule. As in this paper we have employed the former approach, next section describes it in more details.

Pseudo-Inverse Method: The pseudo-inverse method can be seen as the solution for a least square problem, in which we can compute the matrix of weigths $W_{m\times p}$. Let $X = \{x^1, x^2, \ldots, x^r\}$ be a training dataset, such that $x^i \in \mathbb{R}^n$. We can find W by using X as the inputs to the neural network. Let Φ be defined as follows:

$$\Phi = \begin{bmatrix} h_1(x^1|\mu_1,\sigma_1) & h_2(x^1|\mu_2,\sigma_2) & \ldots & h_m(x^1|\mu_m,\sigma_m) \\ h_1(x^2|\mu_1,\sigma_1) & h_2(x^2|\mu_2,\sigma_2) & \ldots & h_m(x^2|\mu_m,\sigma_m) \\ \vdots & \ddots & \vdots & \vdots \\ h_1(x^r|\mu_1,\sigma_1) & h_2(x^r|\mu_2\sigma_2) & \ldots & h_m(x^r|\mu_m,\sigma_m) \end{bmatrix}, \quad (5)$$

where $\Phi_{r\times m}$ is a matrix with the responses of the m hidden layer neurons to the r training input samples. We also have:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \ldots & w_{1,p} \\ w_{2,1} & w_{2,2} & \ldots & w_{2,p} \\ \vdots & \ddots & \vdots & \vdots \\ w_{m,1} & w_{m,2} & \ldots & w_{m,p} \end{bmatrix}, \quad (6)$$

that is our already known weight's matrix, in which $w_{i,j}$ means the weight of the connection between hidden neuron i to the output neuron j. Finally, let $Y_{r\times p}$ be the output matrix:

$$Y = \begin{bmatrix} y_{1,1} & y_{1,2} & \ldots & y_{1,p} \\ y_{2,1} & y_{2,2} & \ldots & y_{2,p} \\ \vdots & \ddots & \vdots & \vdots \\ y_{r,1} & y_{r,2} & \ldots & y_{r,p} \end{bmatrix}, \quad (7)$$

in which $y_{i,j}$ means the desired value for the training sample i at output neuron j. Notice y_j has been defined in Equation 1, but for only one sample $x \in \mathbb{R}^n$. Now, we have a set X with r samples, and each $y_{i,j}$ is now redefined to be part of the matrix Y, but it can be computed as the same way before, i.e., using Equation 1, which can be rewritten to handle the whole training set:

$$Y = \Phi W. \quad (8)$$

By now, we have to discover W, which can be computed as follows:

$$W = \Phi^{-1} Y, \quad (9)$$

in which Φ^{-1} stands for the inverse of Φ. As Φ may not be square, we need to compute its pseudo-inverse Φ^{+}, which can be performed as follows:

$$\Phi^{+} = V S^{+} U^{T}, \quad (10)$$

which is the so-called Moore-Penrose pseudo-inverse. In this case, Equation 9 can be rewritten as:

$$W = \Phi^{+} Y. \quad (11)$$

Therefore, W can be computed and the weights estimated.

B. Optimum-Path Forest Clustering

The design of classifiers based on Optimum-Path Forest has been proposed as a graph-based methodology to exploit connectivity relations between data samples in a given feature space. The methodology interprets a training set as a graph, whose nodes are the samples and the arcs connect pairs of samples that satisfy a given *adjacency relation*. For a suitable *path-value (connectivity) function*, the optimum-path forest algorithm [8] partitions the graph into optimum-path trees rooted at some key samples, named *prototypes*. The prototypes compete among themselves for the most closely connected samples in the training set, such that each sample is assigned to the tree whose prototype offers to it an optimum path. Classification of a new sample is done by finding its most closely connected root in an incremental way through the evaluation of the optimum-path values of the training samples.

Let $\mathcal{Z}$ be a dataset such that for every sample $s \in \mathcal{Z}$ there exists a feature vector $\vec{v}(s)$. Let $d(s,t)$ be the distance between s and t in the feature space. A graph $(\mathcal{Z}, \mathcal{A}_k)$ can be defined such that the arcs $(s,t) \in \mathcal{A}_k$ connect k-nearest neighbors (k-nn) in the feature space ($\mathcal{A}_k$ stands for the k-nn adjacency relation). The arcs are weighted by $d(s,t)$ and the nodes $s \in \mathcal{Z}$ are weighted by a probability density value $\rho(s)$:

$$\rho(s) = \frac{1}{\sqrt{2\pi\sigma^2}|\mathcal{A}_k(s)|} \sum_{\forall t \in \mathcal{A}_k(s)} \exp\left(\frac{-d^2(s,t)}{2\sigma^2}\right), \quad (12)$$

where $|\mathcal{A}_k(s)| = k$, $\sigma = \frac{d_f}{3}$, and d_f is the maximum arc weight in $(\mathcal{Z}, \mathcal{A}_k)$. This parameter choice considers all adjacent nodes for density computation, since a Gaussian function covers most samples within $d(s,t) \in [0, 3\sigma]$. Moreover, since $\mathcal{A}_k$ is asymmetric, symmetric arcs must be added to it on the plateaus of the probability density function (PDF) in order to guarantee a single root per maximum.

The traditional method to estimate a PDF is by Parzen-window. Equation 12 can provide Parzen-window estimation based on an isotropic Gaussian kernel when we define the arcs by $(s,t) \in \mathcal{A}_k$ if $d(s,t) \leq d_f$. This choice, however, presents problems with the differences in scale and sample concentration. Solutions for this problem lead to adaptive choices of d_f depending on the region of the feature space [9]. By taking into account the k-nearest neighbors, the method handles different concentrations and reduces the scale problem to the one of finding the best value of k, say k^* within $[k_{\min}, k_{\max}]$, for $1 \leq k_{\min} < k_{\max} \leq |\mathcal{Z}|$.

The solution proposed by Rocha *et al.* [7] to find k^* considers the minimum graph cut among all clustering results for $k \in [1, k_{\max}]$ ($k_{\min} = 1$), according to the normalized

measure $GC(\mathcal{A}_k, L, d)$ suggested by Shi and Malik [10]:

$$GC(\mathcal{A}_k, L, d) = \sum_{i=1}^{c} \frac{W_i'}{W_i + W_i'}, \quad (13)$$

$$W_i = \sum_{\forall (s,t) \in \mathcal{A}_k | L(s)=L(t)=i} \frac{1}{d(s,t)}, \quad (14)$$

$$W_i' = \sum_{\forall (s,t) \in \mathcal{A}_k | L(s)=i, L(t)\neq i} \frac{1}{d(s,t)}, \quad (15)$$

where $L(t)$ is the label of sample t, W_i' uses all arc weights between cluster i and other clusters, and W_i uses all arc weights within cluster $i = 1, 2, \ldots, c$.

The method defines a path π_t as a sequence of adjacent samples starting from a root $R(t)$ and ending at a sample t, being $\pi_t = \langle t \rangle$ a trivial path and $\pi_s \cdot \langle s, t \rangle$ the concatenation of π_s and arc (s, t). It assigns to each path π_t a value $f(\pi_t)$ given by a connectivity function f. A path π_t is considered optimum if $f(\pi_t) \geq f(\tau_t)$ for any other path τ_t.

Among all possible paths π_t from the maxima of the PDF, the method assigns to t a path whose minimum density value along it is maximum. That is, the method finds $V(t) = \max_{\forall \pi_t \in (\mathcal{Z}, \mathcal{A}_k)} \{f(\pi_t)\}$ for $f(\pi_t)$ defined by:

$$f(\langle t \rangle) = \begin{cases} \rho(t) & \text{if } t \in \mathcal{R} \\ \rho(t) - \delta & \text{otherwise} \end{cases}$$

$$f(\langle \pi_s \cdot \langle s, t \rangle \rangle) = \min\{f(\pi_s), \rho(t)\}, \quad (16)$$

for $\delta = \min_{\forall (s,t) \in \mathcal{A}_k | \rho(t) \neq \rho(s)} |\rho(t) - \rho(s)|$ and $\mathcal{R}$ being a root set, discovered on-the-fly, with one element per each maximum of the PDF. Note that higher values of δ reduce the number of maxima. We are setting $\delta = 1.0$ and scaling real numbers $\rho(t) \in [1, 1000]$ in this work. The OPF algorithm maximizes the connectivity map $V(t)$ by computing an optimum-path forest — a predecessor map P with no cycles that assigns to each sample $t \notin \mathcal{R}$ its predecessor $P(t)$ in the optimum path from $\mathcal{R}$ or a marker nil when $t \in \mathcal{R}$.

III. Optimum-Path Forest-Based Training

In this section, we present the proposed approach that employs OPF clustering to find out the centers of Gaussian distributions. Given a supervised dataset, we partition it in a training and testing sets, being the former used by OPF to cluster samples and then to compute prototypes. After that, their positions are stored and used as the mean values for the Gaussian distributions (the covariance matrices are computed using Equation 4).

Let $\mathcal{C}$ be the number of prototypes (clusters) found by OPF in the ANN-RBF unsupervised training step. After building the Gaussian models, we employ a neural architecture with n input neurons and $|\mathcal{C}|$ neurons for the hidden and output layers. Although we may have more hidden layers than the number of classes (it may have more Gaussian distributions than classes), OPF can find out a suitable number of clusters (Gaussians), and then we can use the position of the root of each them (i.e., the prototypes) to encode the mean values of each Gaussian. Figure 1 illustrates the proposed procedure.

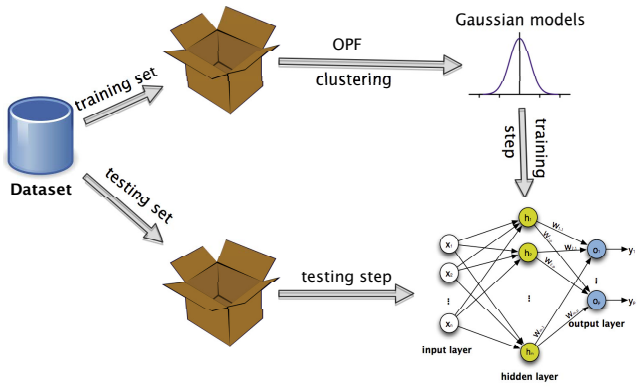

Fig. 1. Proposed pipeline for OPF-based neural network training.

IV. Experimental Results

The experiments have been carried out in two distinct phases: (i) in the former one, we have shown the similarity (location) between prototypes found by OPF and the means computed by k-means in four two-dimensional synthetic datasets, as well as we discussed the robustness of OPF with respect to k_{max} parameter; (ii) in the latter experiment, we employed the proposed approach (Section III) for the same synthetic datasets used in the previous step and also for three more real problems. Notice for all experiments we have employed a computer equipped with an Intel I5 ® processor, 8Gb RAM and OS X 10.8.5 as the operational system.

Table I presents the datasets employed in this paper. Notice the top four datasets were created (synthetic) for this work, and the remaining ones are public available datasets.

TABLE I
Description of the datasets employed in this work.

Dataset	# samples	# features	# classes
Db1	500	2	2
Db2	200	2	2
Db3	150	2	3
Db4	445	2	3
Wine [11]	178	13	4
Iris [11]	150	4	3
Breast-cancer (BC) [11]	683	10	2
Saturn [12]	200	2	2
Cone-Torus (CT) [12]	200	2	3

A. Experiments

1) OPF versus k-means: As aforementioned, the first round of experiments aimed to compare the OPF prototypes' position with the centers obtained by k-means. Figure 2 depicts such results, in which the left and right columns stand for k-means (centers in red) and OPF (prototypes in orange) results, being $k_{max} = 50$ used for all datasets. One can see that both k-means and OPF achieved similar results for Db1 (Figures 2a and 2b) and Db3 (Figures 2e and 2f) datasets. However, in situations when we have more than one cluster per class, k-means may not achieve reasonable results, as one can notice

for Db2 (Figures 2c and 2d) and Db4 (Figures 2f and 2g) datasets. Therefore, such behavior can degrade the ANN-RBF effectiveness, since the feature space will not be properly covered by the Gaussians distributions. Additionally, one can check that OPF found out the centers of all clusters for both datasets.

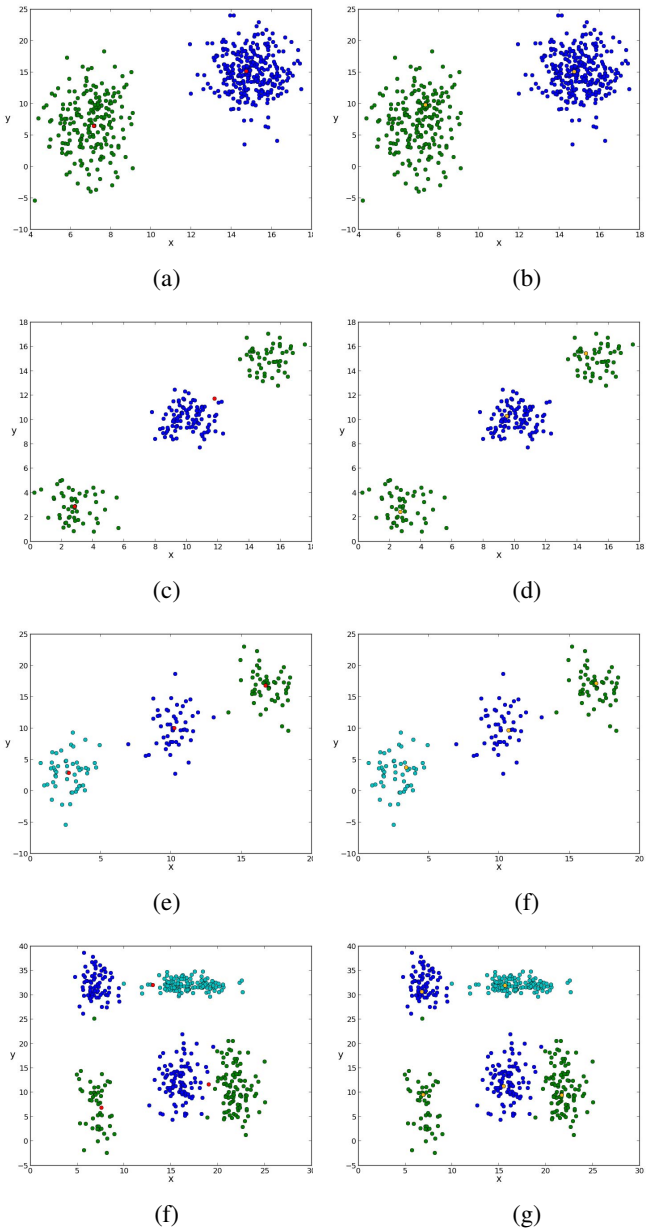

Fig. 2. Synthetic datasets: OPF prototypes in orange for datasets Db1, Db2, Db3 and Db4 in the right column, respectively, and k-means centers found in red for datasets Db1, Db2, Db3 and Db4 in the left column, respectively.

There is one more question regarding OPF clustering that one may face: "What about k_{max} parameter?". One can argue OPF does not require the number of classes (clusters), but the user needs to input k_{max}. This parameter controls the area of coverage of one sample during the competition process, and its value may previously define the number of clusters (Section II-B). However, k_{max} is much less prone to error for clusters' estimation than k value for k-means. Figure 3 shows an experiment over Db1, Db2, Db3 and Db4 datasets using different values of k_{max}. One can see there that are several plateus in the curves, i. e., there are several k_{max} values in which the number of clusters do not change. Notice the number of clusters increases drastically with small values of k_{max}, since the compared datasets have hundreds of samples. For smaller datasets, this may not happen. Therefore, an user with a low level of experience with OPF can learn suitable input vales for k_{max}.

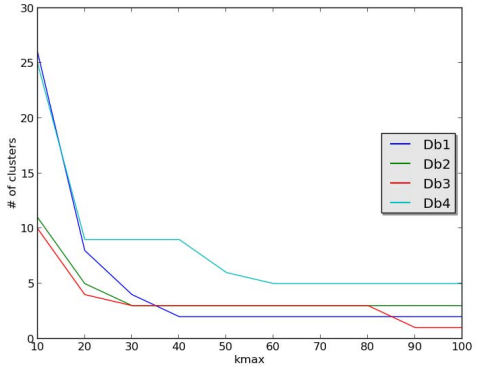

Fig. 3. Robustness of OPF regarding different k_{max} values.

2) Evaluating OPF effectiveness for ANN-RBF neurons estimation: In this set of experiment, we have evaluated the quality of the neurons found out by OPF against with the ones computed by k-means. For such purpose, we employed the datasets presented in Table I in a cross-validation procedure with 10 runnings, being 50% of the dataset used to compose the training set, and the remaining samples for testing. After that, we assessed the robustness (both efficiency and effectiveness) of experiments over a paired-sample t-test with 95% of confidence. Table II presents the mean accuracy of ANN-RBF with the neurons obtained by OPF and k-means. Notice we employed an accuracy measure proposed by Papa et al. [13], which considers imbalanced classes.

One can verify that the results using the neurons obtained by OPF are clearly better than the ones computed by k-means. In some cases, the ANN-RBF recognition rate using OPF neurons has been 48.13% better than using k-means (Db4 dataset, for instance). Such difference mainly rely on situations in which the number of Gaussians that cover the feature space is different from the number of classes. In practice, as one can found in all real applications, OPF presented a considerable advantage over k-means for neurons' estimation.

In regard to the statistical evaluation, the paired-sample t-test with 95% of confidence rejected the null hypothesis, which states OPF and k-means are similar to each other (i.e., both approaches can lead ANN-RBF to results over all datasets with

TABLE II
MEAN ACCURACIES FOR ANN-RBF CLASSIFICATION USING OPF AND k-MEANS. IT SHOULD BE NOTICED THAT THE STANDARD DEVIATIONS ARE TOO SMALL TO BE INDICATED. THE SQUARE BRACKETS IN OPF COLUMN STAND FOR k_{max}.

Dataset	**OPF**	**k-means**
Db1	98.33%[50]	97.16%
Db2	100.00%[50]	74.00%
Db3	99.00%[50]	99.00%
Db4	97.49%[50]	65.81%
Wine	81.99%[5]	74.38%
Iris	99.00%[10]	94.00%
Breast-cancer	96.34%[50]	80.24%
Saturn	53.99%[50]	44.00%
Cone-Torus	84.33%[10]	73.17%

the same mean accuracies). Therefore, one can conclude OPF provides more suitable neurons for ANN-RBF training than k-means considering the datasets employed in this work.

In addition, we have computed the execution times (seconds) for ANN-RBF training step through OPF and k-means. Figure 4 illustrates this information for each dataset. The mean execution time for k-means (considering all datasets) was about 0.0041s, while OPF-based training time was about 0.0348s. Therefore, the training step using k-means is 8.48 times faster than employing OPF. Once again, we applied the paired-sample t-test with 95% of confidence in the execution times data, which evidenced that k-means is statistically faster than OPF, i.e., the t-test rejected the null hypothesis. However, OPF can be a suitable alternative to k-means, since the former recognition rates are considerable better than the last one.

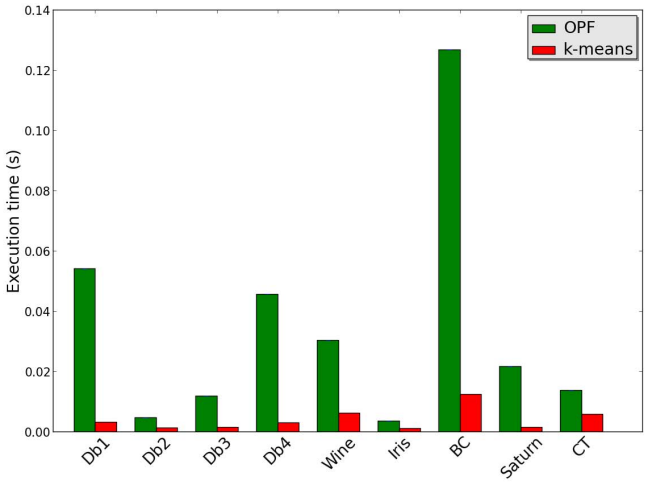

Fig. 4. Execution times for OPF- and k-means-based ANN-RBF training.

V. CONCLUSIONS

The RBF training procedure is composed by two steps: (i) an unsupervised one, which aims to find the hidden neurons; and (ii) a supervised step, that properly computes the weights that will be used to combine the output of hidden neurons. The most commonly used approach to find out Gaussian's neurons is to employ k-means for such purpose, since the neurons encode the mean parameter of the Gaussian distribution. Therefore, as k-means tries to find the center point of a given cluster, such approach is a good candidate for this task. However, the user needs to input the number of means (k), which may not correspond to the real number of clusters, since a class can be represented by more than one cluster. In this paper, we proposed to introduce the OPF clustering approach for such task, since it computes the number of clusters (neurons) on-the-fly, being more intuitive to be used than k-means.

The experiments have shown that the OPF neurons' position are similar to the ones found out by k-means, and OPF can outperform k-means in situations that require more Gaussians than classes to cover the feature space. Experimental results over 9 datasets showed the robustness of OPF-based RBF training step. The main drawback of using OPF concerns with its computational load, which is greater than k-means one.

REFERENCES

[1] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[3] S. Haykin, *Neural Networks: A comprehensive foundation*. Prentice-Hall, 1998.

[4] A. Esmaeili and N. Mozayani, "Adjusting the parameters of radial basis function networks using particle swarm optimization," in *Proceedings of the IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, 2009, pp. 179–181.

[5] G. E. Tsekouras and J. Tsimikas, "On training RBF neural networks using input-output fuzzy clustering and particle swarm optimization," *Fuzzy Sets and Systems*, vol. 221, pp. 65–89, 2013.

[6] S. N. Qasem and S. M. Shamsuddin, "Memetic elitist pareto differential evolution algorithm based radial basis function networks for classification problems," *Applied Soft Computing*, vol. 11, no. 8, pp. 5565–5581, 2011.

[7] L. M. Rocha, F. A. M. Cappabianco, and A. X. Falcão, "Data clustering as an optimum-path forest problem with applications in image analysis," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 50–68, 2009.

[8] A. Falcão, J. Stolfi, and R. Lotufo, "The image foresting transform theory, algorithms, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 19–29, 2004.

[9] D. Comaniciu, "An algorithm for data-driven bandwidth selection," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 281–288, 2003.

[10] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug 2000.

[11] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

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

[13] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki, "Supervised pattern classification based on optimum-path forest," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 120–131, 2009.