FEMa: A Finite Element Machine for Fast Learning

Danillo Roberto Pereira, Marco Antônio Piteri, André Nunes Souza, João Paulo Papa, *Member, IEEE,*
and Hojjat Adeli, *Fellow, IEEE, Distinguished Member,*
ASCE, Fellow, AAAS, Fellow AIMBE, Fellow American Neurological Association

***Abstract*—Machine learning has played an important role in the past decades, being in lockstep with the main advances in computer technology. Given the massive amount of data generated daily, there is a need for even faster and effective learning algorithms that can provide updated models for real-time applications and on-demand tools. In this paper, we propose FEMa - A Finite Element Machine classifier - for supervised learning problems, where each training sample is the center of a basis function, and the whole training set is modeled as a probabilistic manifold for classification purposes. FEMa has its theoretical basis in the Finite Element Method, which is widely used for numeral analysis in engineering problems. We show FEMa is parameterless and it has a quadratic complexity for both training and classification phases when we use basis functions that obey some properties, as well as the proposed classifier can obtain very competitive results when compared against some state-of-the-art supervised pattern recognition techniques.**

***Index Terms*—Finite element methods, Pattern classification, Pattern recognition**

I. Introduction

THE "Big Data" era has flooded researchers and the whole community with tons of data daily. Multimedia-based applications are in charge of generating an unsurmountable amount of data, which end up at the screens of mobile phones and tablets. Home-made videos are usually referred as the bottleneck of any network traffic analyzer, since they are uploaded to cloud-driven servers as soon as they are generated or forwarded by someone else via the so-called social networks.

The huge amount of data requires to be processed and mined efficiently. Former versions of well-known machine learning techniques such as Support Vector Machines (SVMs) [1], Artificial Neural Networks (ANNs) [2], [3], Polynomial Neural Networks [4], Recurrent Networks [5], [6], and Adaptive Conjugate Gradient Neural Networks [7], [8] are now being implemented in General-Purpose Computing on Graphics Processing Units (GPGPU) to cope with streams of data that need to be analyzed daily.

Active learning is another research area that needs fast techniques for learning and classification. One very usual example concerns interactive and semi-supervised learning tools for image classification and annotation. Suppose a physician wants to classify a Magnetic Resonance image of the brain, which may contain hundreds of thousands of pixels. The user shall mark a few positive and negative samples (pixels) that will be used to train the classifier, which then classifies the remaining image. Further, the user shall refine the results by marking some misclassified regions for training once more. Notice the whole process should take a few seconds/iterations. In this context, the user feedback is crucial to obtain a concise/reliable labeled image.

Considering the aforementioned situation, some techniques may not be appropriate to be employed, since they can hardly handle the problem of updating the model learned previously when new training samples come to the problem. Support Vector Machines are known to be costly, since they require a fine-tuning parameter step, which turns out to be the bottleneck for efficient implementations [9]. Although different variations and GPU-based implementations are published monthly, it is not straightforward to use them, which makes them far from being user-friendly. Additionally, SVM training step is quadratic with respect to the number of training examples.

Deep learning techniques have have received a lot of attention in recent years [10], [11], since they can learn features from images/signals without label information. Although such approaches have obtained outstanding results in a number of applications, they usually overfit under small training sets. Also, some architectures require hundreds of parameters for fine-tuning resulting in very costly training.

Graph-based pattern recognition techniques took their place in the scientific community as well. Papa et al. [12], [13], [14], [15] proposed the Optimum-Path Forest (OPF), a framework for the design of classifiers. OPF has obtained promising results in a number of applications, being much faster than SVM for training, since its original version is parameterless [13], [14] and does not require fine-tuning parameters. However, OPF-based classifiers are usually affected by high-dimensional spaces, a shortcoming for techniques that make use of distances for classification purposes.

Artificial Neural Networks have been reinvented in the last decades. From the original Backpropagation learning algorithm [16] to faster approaches such as the Levenberg-Marquardt [17], the reader can refer to a number of variants that somehow try to deal with the problem of avoiding getting trapped from local optima during training, as well as to make their convergence step faster [18]. Polynomial neural networks [19], hybrid networks [20], and probabilistic ones [21], [22] have been used in a number of different applications in the literature.

D. Pereira and J. Papa are with the Department of Computing, São Paulo State University, Bauru, SP, 17033-360 Brazil e-mail: dpereira@ic.unicamp.br, papa@fc.unesp.br

M. Piteri is with the Department of Computing, São Paulo State University, Presidente Prudente, SP Brazil e-mail: piteri@fct.unesp.br

A. Souza is with the Department of Electrical Engineering, São Paulo State University, Bauru, SP, 17033-360 Brazil e-mail: andrejau@feb.unesp.br

H. Adeli is with the Department of Civil, Environmental and Geodetic Engineering, The Ohio State University, Columbus, OH 43210 USA e-mail: adeli.1@osu.edu

In early 90's, Specht [21] proposed the Probabilistic Neural Networks (PNNs), which basically replaces the sigmoid activation function by an exponential one. Since PNNs do not require using Backpropagation, they are usually much faster than traditional ANNs [23], [24]. PNNs are composed of four layers: input, pattern, summation and output. The first layer is responsible for feeding the network with features extracted from samples, and the pattern layer aims at encoding all training data patterns, i.e. the number of pattern units (Gaussian probability distribution functions) is the very same number of training samples. The summation layer contains one unit for each class, and the output layer uses a Bayesian rule to compute the probability in assigning a certain class to a given input data. Since standard PNNs use an exponential activation function, one needs to set the variance (spread) of the Gaussian function, which can considerably influence the effectiveness of the network.

Some years later, Ahmadlou and Adeli [22] proposed the Enhanced Probabilistic Networks (EPNNs), a clever way to penalize outliers when computing the influence of the Gaussian distribution over the training samples. Actually, the authors proposed to compute a variance for each training sample based on a neighborhood, and depending on the class labels of its neighbours, the Gaussian function centered at an outlier pattern can barely influence other points. Papers that make use of EPNNs have appeared in the literature [25], [26], since EPNNs are fast and very suitable for large-scale datasets.

Moving from machine learning to numerical analysis, one of the most widely used approaches for finding approximate solutions to boundary-value problems in partial differential equations is the Finite Element Method (FEM) [27], [28]. Roughly speaking, FEM divides the original problem into smaller pieces called finite elements, and the simple equations that describe each element are assembled in a complex one that should describe the whole problem. Therefore, given a set of points, FEM can interpolate them using basis functions in order to build a manifold that contains all these points. In this paper, we borrow some ideas related to FEM to propose FEMa - Finite Element Machine, a new framework for the design of pattern classifiers based on finite element analysis. Depending on the basis function used, FEMa can be parameterless. It features a quadratic complexity for both training and classification phases, which turns out to be its main advantage when dealing with massive amount of data. In short, FEMa learns a probabilistic manifold built over the training samples, which are the center of a finite element basis. Therefore, the problem of learning a manifold using one finite element basis is broken into a surface composed of several bases, centered at each training sample. In this paper, we show that FEMa can obtain very competitive results when compared against some state-of-the-art supervised pattern recognition techniques.

The remainder of this paper is organized as follows. Sections II and III introduce the theoretical background related to FEM and FEMa, respectively. Section IV presents the methodology and experiments used to evaluate FEMa in the context of big data environments, and Section V states conclusions and future works.

II. Finite Element Method

In this section, we present the main concepts related to the Finite Element Method. Broadly speaking, FEM aims at approximating functions given a set of sampled points by means of basis functions. In a first step, the basis functions are used to interpolate the manifold based on the sampled points (domain) and their respective responses to that functions (image). Further, the approximation step aims at interpolating new points to the learned manifold.

A. Function Approximation

Let $\mathcal{D}$ and $\mathcal{V}$ be an infinite and a non-trivial set, respectively, and $F : \mathcal{D} \rightarrow \mathcal{V}$ be a function that contains an infinite number of mappings. Therefore, F can not be represented as a generic element in computers, and thus one needs to replace F by an approximation function $\tilde{F}$ in some finite subspace. Additionally, the quality of the approximation function $\tilde{F}$ can be measured by the norm $\|\tilde{F} - F\|$, where $\|\cdot\|$ can be any norm defined on some finite space. Also, that norm is often called approximation error.

1) Approximation Basis: A basis ϕ of the space $\mathcal{V}$ is an array $\phi = [\phi_1, \phi_2, \ldots, \phi_n]$ of functions whose elements are linearly independent. Also, every element $v \in \mathcal{V}$ can be obtained by a linear combination of those functions as follows:

$$v = \sum_{i=1}^{n} a_i \phi_i, \tag{1}$$

where $\mathbf{a} = [a_1, a_2, \ldots, a_n]$ such that $a_i \in \Re$. Notice the approximation function $\tilde{F}$ can be represented in computers by the real coefficients $\mathbf{a}$ when ϕ is a basis of some finite space.

2) Interpolation: One basic application of approximation spaces is the interpolation of discrete data. In this context, given a set of points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ such that $\mathcal{X} \subset \mathcal{D}$, and their respective set of associated values $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$, such that $\mathcal{Y} \subset \mathcal{V}$, the goal is to find an approximation function $\tilde{F}$ that interpolates the pairs $(\mathbf{x}_i, y_i)$ such that:

$$\tilde{F}(\mathbf{x}_i) = y_i, \forall i \in \{1, 2, \ldots, n\}. \tag{2}$$

In order to describe $\tilde{F}$ by the basis ϕ one needs to find the coefficients $\mathbf{a}$ such that:

$$\tilde{F}(\mathbf{x}_i) = \sum_{j=1}^{n} a_j \phi_j(\mathbf{x}_i) = y_i, \forall i \in \{1, 2, \ldots, n\}. \tag{3}$$

The above equation means each element $y_i \in \mathcal{Y}$ is generated from the linear combination between all basis functions and their respective coefficients.

The above formulation is equivalent to solve the following linear system in the matrix notation:

$$\mathbf{Z}\mathbf{a} = \mathbf{y}, \tag{4}$$

where $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$, and $\mathbf{Z}$ is an $n \times n$ matrix that stores the influence of each basis element ϕ_i concerning the point x_j, as follows:

$$Z_{ij} = \phi_i(\mathbf{x}_j). \quad (5)$$

3) Interpolating Bases: A basis ϕ is an interpolating basis regarding the points in $\mathcal{X}$ iff:

$$\phi_i(\mathbf{x}_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

For such a basis, $\mathbf{Z}$ stands for the identity matrix, which means $a_i = y_i, \forall i \in \{1, 2, \ldots, n\}$.

However, one can face bases that are not interpolating natively. In this case, given a non-interpolating basis, we can obtain a new interpolating one $\hat{\phi}$ where each element $\hat{\phi}_i$ is a linear combination of the elements ϕ_i, as follows:

$$\hat{\phi}_i(\mathbf{x}) = \sum_{j=0}^{n} Z_{ij}^{-1} \phi_j(\mathbf{x}), \quad (7)$$

where $\mathbf{Z}^{-1}$ is the inverse of matrix $\mathbf{Z}$.

B. Partition of Unity Basis

A basis ϕ is a partition of unity iff:

$$\phi_i(\mathbf{x}) \geq 0, \forall i \text{ and } \forall \mathbf{x} \in \mathcal{D}, \quad (8)$$

and

$$\sum_{i=1}^{n} \phi_i(\mathbf{x}) = 1, \forall \mathbf{x} \in \mathcal{D}. \quad (9)$$

Such basis has smoothing properties, as follows:

$$a_l \geq \sum_{i=1}^{n} a_i \phi_i(\mathbf{x}) \geq a_h, \quad (10)$$

where a_l and a_h stand for the minimum and maximum coefficients of $\mathbf{a}$. The smoothness in interpolation-driven computations is often desired to avoid discontinuities.

Given a basis ϕ that satisfies Equation 8 only, we can easily define a new basis $\tilde{\phi}$ in order to satisfy Equation 9 either. Such new basis can be obtained by means of the following normalization step:

$$\tilde{\phi}_i(\mathbf{x}) = \frac{\phi_i(\mathbf{x})}{\sum_{j=1}^{n} \phi_j(\mathbf{x})}. \quad (11)$$

C. Finite Element Basis

Let $S(\phi(\mathbf{x}))$ be the support of a given basis $\phi(\mathbf{x})$, which represents the set of points $\mathbf{x} \in \mathcal{D}$ such that $\phi(\mathbf{x}) \neq 0$. A finite element basis ϕ for an approximation space requires $S(\phi(\mathbf{x}))$ be small and compact enough. The meaning of "small" depends on the context, but usually means the value (e.g. length, area, and volume) of $S(\phi(\mathbf{x}))$ is about $1/n$ of the measurements of $\mathcal{D}$.

The union of all supports of basis ϕ should cover the entire domain $\mathcal{D}$ of the points where the function F (function to be approximated) is nonzero. The use of such bases of finite elements to the approximation of functions concerns the so-called Finite Element Method (FEM).

In this work, we use a special class of finite element bases, which are defined by points (meshless) [29], [30]. In such basis, each finite element ϕ_i has a central point $\mathbf{x}_i$ located at the center of $S(\phi(\mathbf{x_i}))$. In other words, we are just centering the basis at the point $\mathbf{x}_i$. Next, we present the basis used in this work, which is quite popular in the context FEM.

1) Shepard Basis: In the Shepard basis [31], each element is defined as follows:

$$\phi_i(\mathbf{x}) = \frac{w(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^{n} w(\mathbf{x}, \mathbf{x}_j)}, \quad (12)$$

where $w : \mathcal{D} \times \mathcal{D} \rightarrow \Re$ is a non-negative function, such that $w(\mathbf{x}, \mathbf{x}_i) \rightarrow \infty$ when $\mathbf{x} \rightarrow \mathbf{x}_i$. Roughly speaking, the closer is $\mathbf{x}$ from $\mathbf{x}_i$, the larger is the value of function w. Such property implies that a Shepard basis holds the interpolating and partition of unity assumptions.

Usually, function w is chosen as a power $k \geq 1$ of the inverse of the Euclidean distance, as follows:

$$w(\mathbf{x}, \mathbf{x}_i) = \frac{1}{|\mathbf{x}, \mathbf{x}_i|^k}, \quad (13)$$

where $|\mathbf{x}, \mathbf{x}_i|$ denotes the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}_i$. Notice parameter k controls the smoothness of the interpolation process, and it should be chosen according to the user needs. Figure 1 shows different Shepard bases using three values of k. One can observe the behaviour of the basis centered at the black dots according to different values of k: the greater the value of k, the more sloppy is the function. Clearly, $k = 1$ results in a steep function.

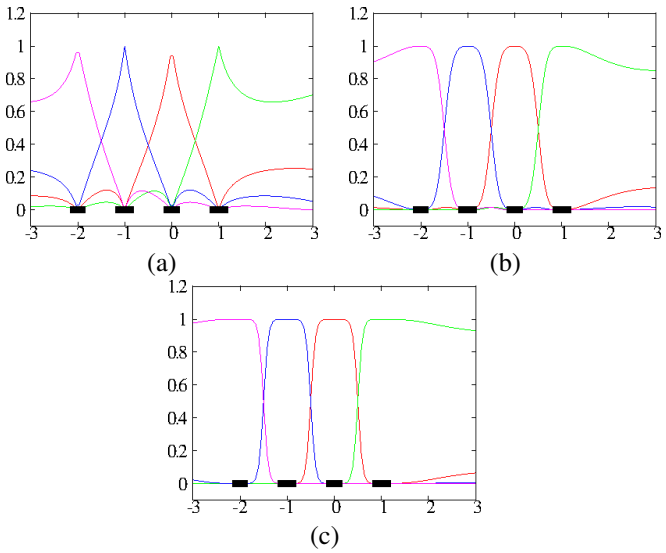

Fig. 1. Behaviour of different Shepard bases according to three values of k, where the black dots stand for the center of the basis: (a) $k = 1$, (b) $k = 3$ and (c) $k = 5$.

Figure 2 depicts some interpolated functions using FEM with Shepard basis. Analogously to the behaviour of the aforementioned basis, the interpolated functions tend to become less smooth. Once more, the rectangles stand for the center of the basis.

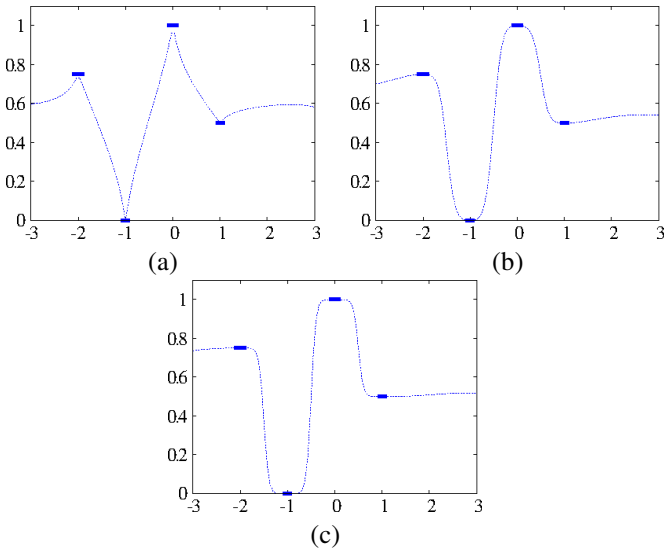

Fig. 2. Interpolated function using the Shepard basis for (a) $k = 1$, (b) $k = 3$ and (c) $k = 5$. The blue rectangles represent the center of the basis and their sampled values.

III. Finite Element Machine

In this section, we present the Finite Element Machine classifier, as well as how it can cope with the problem of supervised pattern classification efficiently.

A. Background Theory

Let $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2$ be a dataset partitioned into a training ($\mathcal{Z}_1$) and a test ($\mathcal{Z}_2$) set. In this case, the pair $(\mathbf{x}_i, y_i) \in \mathcal{Z}$ denotes the feature vector $\mathbf{x}_i \in \Re^m$ extracted from sample i, and y_i stands for its label. Notice we adopted the very same formulation used in the previous section, i.e. a point in FEM formulation stands for a sample in FEMa.

Roughly speaking, FEMa learns a set of probability functions $\mathcal{P}(\mathbf{x}) = \{P_1(\mathbf{x}), P_2(\mathbf{x}), \ldots, P_c(\mathbf{x})\}$, where c stands for the number of classes, and $P_i(\mathbf{x})$ represents the probability of a given sample $\mathbf{x}$ to be assigned to class i. In other words, FEMa aims at learning a probabilistic manifold from the training set.

B. Probabilistic Manifold Learning

Depending on the basis function used to interpolate points, FEMa does not require a training step, which turns out to be quite interesting when dealing with big data. Precisely, this assumption is true concerning bases that are natively interpolating, such as Shepard basis. On the other hand, with respect to non-interpolating basis, e.g. radial functions, one needs to compute $\mathbf{Z}^{-1}$ in Equation 7. Also, if the basis function does not hold the partition of unity property, one shall compute Equation 11 either. Therefore, although FEMa can be used with any basis function, we shed light over that bases holding both the interpolating and partition of unity properties are much more appealing when dealing with massive amount of data. As such, we can consider the calculation of $\mathbf{Z}^{-1}$ and Equation 11 as the training steps when using non-interpolating and non-partition of units bases.

Assuming we are using an interpolating and partition of unity basis (e.g Shepard), we can move to the classification step. Given a sample $\mathbf{x} \in \mathcal{Z}_2$, we need to compute its probability of belonging to each class i, $i = 1, 2, \ldots, c$, as follows:

$$P_i(\mathbf{x}) = \sum_{j=1}^{|\mathcal{Z}_1|} \rho_i^j \phi_j(\mathbf{x}), \tag{14}$$

where $\rho_i^j \in [0, 1]$ stands for the probability of training sample j belonging to class i. An interesting property concerning FEMa relates to the possibility in assigning a probability to each training sample, which means we have an uncertainty associated to those samples, thus having an important role when dealing with data overfitting. This capability is extremely important in medical-driven applications, where physicians usually have different opinions with respect to the very same data (e.g. cancer detection in images).

The probability $\rho_i^j \in [0, 1]$ can be computed using the following formulation:

$$\rho_i^j = \begin{cases} 1 & \text{if } y_j = i \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Since we have labeled datasets (i.e. we are assuming the labeling process is errorless), we can use $\rho_i^j \in \{0, 1\}$. Therefore, we generate the set of probabilities $\mathcal{P}(\mathbf{x})$ for each sample $\mathbf{x} \in \mathcal{Z}_2$.

In short, FEMa classifies a given sample $\mathbf{x} \in \mathcal{Z}_2$ as belonging to the class $\hat{y}$ that satisfies the above equation:

$$\hat{y} = \arg\max_i P_i(\mathbf{x}). \tag{16}$$

Also, FEMa allows us to infer the certainty $C(\mathbf{x})$ as follows:

$$C(\mathbf{x}) = \frac{P_{\hat{y}}(\mathbf{x})}{\sum_{j=1}^{c} P_j(\mathbf{x})}. \tag{17}$$

Therefore, FEMa can produce both hard and soft (probability) outputs without any modification. Figure 3 illustrates the process of learning the probability functions of each class in a one-dimensional and two-class problem. For the sake of explanation, the x-axis stands for a test set with samples within the interval $[-3, 3]$, and the y-axis denotes their probability values with respect to the class 1 (Figure 3a) and class 2 (Figure 3b). Also, the red dots stand for the training samples, i.e. the centers of the basis functions.

Let us consider a test sample with value -2 in Figure 3c. As one can observe, such sample has been used as a center for the basis function in Figure 3a already (it is a training sample). In this case, the classification process will assign class 1 to this sample, since $P_1(-2) \approx 1$, and $P_2(-2) \approx 0$. Now, consider a sample with value 2 that does not belong to the training set, i.e. it is not a basis center. In this case, $P_1(2) \approx 0.15$ and $P_2(2) \approx 0.85$, which leads FEMa to assign class 2 to that sample.

C. Toy Example

In this section, we present the FEMa working mechanism on a bidimensional classification problem. Figure 4a shows a

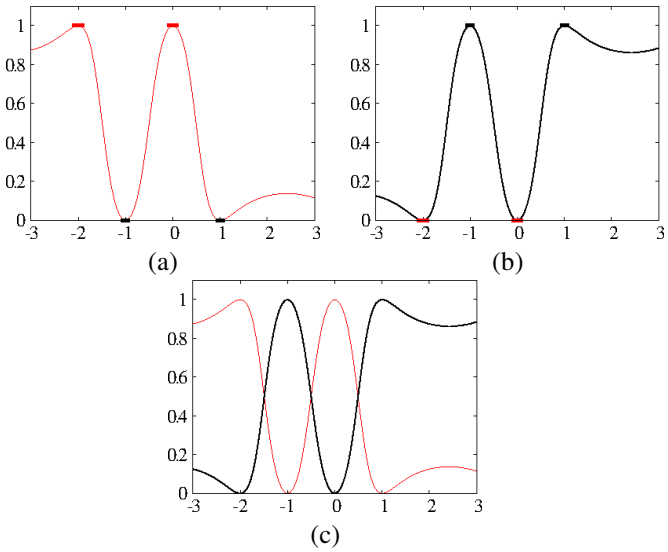

Fig. 3. The Shepard approximation of the probability function of a two-class problem using $k = 3$ considering a given sample $\mathbf{x}$: (a) $P_1(\mathbf{x})$ and (b) $P_2(\mathbf{x})$. The red dots and the red curve denote the samples and the probability function of class 1, respectively, and the black dots and the black curve stand for the samples and the probability function of class 2, respectively. In (c), we have the two probability functions together. Notice each real number in $[-3, 3]$ (i.e. x-axis) is classified according to the class that has the higher probability value (i.e. y-axis).

training set with samples distributed over three classes (red, green and blue). The task is to verify the influence region of each training sample in the image domain, i.e. to classify the remaining points (white ones) in the image frame displayed in Figure 4a. In this case, the feature of each sample (point) is just its (x, y)-position.

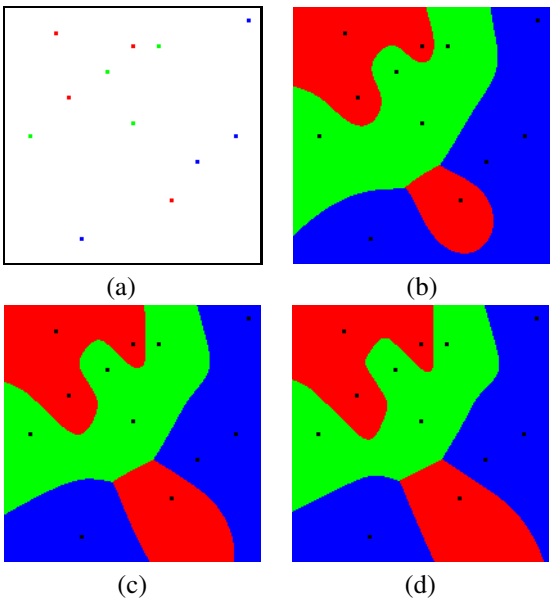

Fig. 4. FEMa working mechanism: (a) training set with samples distributed in three classes, and the image classified by FEMa using (b) $k = 1$, (c) $k = 3$ and (d) $k = 5$.

Figures 4b, 4c and 4d depict the image frame classified by FEMa using the Shepard basis with $k = 1$, $k = 3$ and $k = 5$, respectively. Since we are using the (x, y) coordinates to describe each sample, the labeled image refers to the influence region of each training sample, which ends up generating the boundaries of each class. Notice that FEMa can obtain quite good and smooth decision boundaries for different values of k (Equation 13). As matter of fact, the larger the value of k, the less points will influence the interpolating process of the probability function. For the sake of clarification purposes, when $k \to \infty$, FEMa tends to behave similarly to the well-known nearest neighbor classifier.

Figure 5a displays the degree of certainty (Equation 17) computed by FEMa with $k = 3$ for each test sample with respect to Figure 4a. The brighter the pixel, the greater its degree of certainty to be assigned to some class. Notice the darker pixels fall in the boundary among classes (Figure 4c). Figure 5b represents each test sample by its label color weighted by its degree of certainty.

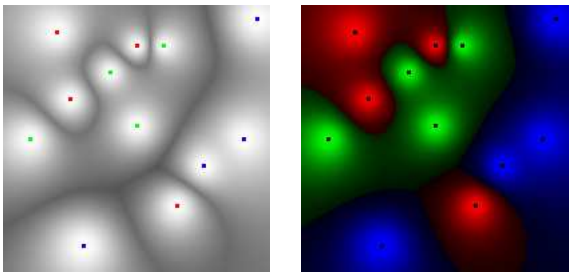

Fig. 5. Probability map (degree of certainty) computed by FEMa in (a), and the test samples with their class label weighted by their respective degree of certainty.

D. Complexity Analysis

As aforementioned, depending on the basis function used to build the probabilistic manifold (i.e. interpolating and partition of unity properties), FEMa does not require an explicit training step, since we just need to place the training points, thus taking $\theta(1)$. However, if one uses a non-interpolating basis function, we need to compute the inverse matrix $\mathbf{Z}^{-1}$ in Equation 7, which requires $\theta(|\mathcal{Z}_1|^{2.37})$ using the Coppersmith-Winograd algorithm [32].

In regard to the classification phase, for each test sample $\mathbf{x}$, we need to compute Equation 12, which requires $\theta(|\mathcal{Z}_1|)$. However, the denominator of such equation considers all training samples, thus becoming a constant, and we need to compute it only once. Since the test set contains $|\mathcal{Z}_2|$ samples, the overall classification phase takes $\theta(|\mathcal{Z}_1| + |\mathcal{Z}_1|\,|\mathcal{Z}_2|) \in \theta(|\mathcal{Z}_1|\,.\,|\mathcal{Z}_2|)$. Therefore, by using an interpolating basis function, the whole FEMa learning and classification processes require a quadratic complexity with respect to the training/testing set size (i.e. when $|\mathcal{Z}_1| = |\mathcal{Z}_2|$).

However, when we have unbalanced datasets, samples from the majority classes will have a stronger influence when computing the probability functions. Suppose a two-class classification problem, i.e. we have samples from the positive and negative samples. Also, suppose samples from the negative class comprise only 1% of the number of positive samples. When we are computing the probability function

of test sample, the positive samples will play a major role during this computation process. In order to overcome this problem, we can use only the T nearest training samples from each class, where $T \in O(\alpha)$ and α stands for the number of elements from the smallest class[1]. In this case, since we need to sort the training samples according to their distances for each test sample, the classification phase now takes $\theta((|\mathcal{Z}_1| \log |\mathcal{Z}_1|).|\mathcal{Z}_2|)$. Notice we can make it better by using some special data structures, such as kd-trees, which require $\theta(|\mathcal{Z}_1| \log |\mathcal{Z}_1|)$ for loading the whole data only once during training. Now, with respect to the classification phase, we do not need the sorting step, since to obtain the nearest T samples takes $O(T. \log |\mathcal{Z}_1|)$, and thus the classification phase requires $\theta((T. \log |\mathcal{Z}_1|).|\mathcal{Z}_2|)$.

IV. Experiments

In this section, we present the methodology and the experiments used to asses the robustness and efficiency of FEMa against six other classifiers: (i) ANN trained with Backpropation, (ii) Bayes, (iii) EPNN, (iv) OPF, (v) k-NN (k-nearest neighbors) and (vi) SVM. Such approaches were selected for comparison purposes since they have been commonly applied in a number of classification tasks in the literature, being some of them referred as state-of-the-art by the machine learning community.

In order to validate the experiments, we employed 23 public benchmarking datasets[2] that have been frequently used for the evaluation of supervised classification methods. We divided the dataset into two groups: (i) small datasets and (ii) medium-to-large datasets. Tables I and II present the main characteristics of the datasets concerning the small and the medium-to-large group, respectively. The datasets were selected in order to represent distinct scenarios, which comprise datasets with different number of features, sizes and classes.

TABLE I
INFORMATION ABOUT THE SMALL DATASETS USED IN THE EXPERIMENTS.

Dataset	# samples	# features	# classes
australian	690	14	2
boat	100	2	3
breast	683	10	2
cone-torus	400	2	3
data1	1,423	2	2
data2	283	2	2
data3	340	2	5
data4	698	2	3
data5	1,850	2	2
diabetes	768	8	2
fourclass	862	2	2
glass	214	9	6
heart	270	13	2
petals	100	2	4
saturn	200	2	2
segment	2,310	19	7
vehicle	846	18	4
wine	178	13	3

[1]In this paper, we use $T = \alpha$.
[2]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

TABLE II
INFORMATION ABOUT THE MEDIUM-TO-LARGE DATASETS USED IN THE EXPERIMENTS.

Dataset	# training samples	# testing samples	# features	# classes
a1a	1,605	30,956	123	2
a2a	2,265	30,296	123	2
a3a	3,185	29,376	123	2
a4a	4,781	27,780	123	2
a5a	6,414	26,147	123	2

Since the medium-to-large datasets are partitioned into training and testing sets already, we decided to partition the small datasets at random using 50% for training purposes and the remaining 50% for classification. Notice the aforementioned protocol was repeated for both normalized and non-normalized versions of the datasets under 15 runnings for the computation of mean accuracy and computational load. The idea is to verify the behavior of FEMa under such circumstance. The normalization process is the same adopted by LibOPF [33], which is used to implement the OPF classifier:

$$\hat{f}_i = \frac{(f_i - \tilde{f}_i)}{s_i}, \tag{18}$$

where f_i, $\tilde{f}_i$ and s_i are, respectively, the i-th feature, the average of f_i, and the standard deviation of f_i in the dataset. Also, $\hat{f}_i$ stands for the normalized version of f_i. In order to compare the classification methods, we computed the mean accuracy and standard deviation for each one. Further, we employed the Wilcoxon signed-rank test [34] with significance of 0.05 to provide a more robust statistical evaluation.

Methods that require fine-tuning parameters (i.e. SVM, k-NN and EPNN) are optimized differently. In regard to SVM, since we used a Radial Basis Function kernel, the searching range of parameter C (optimization function) was defined within the interval $[-32, 32]$, while the searching range of parameter γ (variance of the Gaussian kernel) was defined within the interval $[0, 32]$. For both parameters we used a step size of 2. With respect to k-NN, we defined the value of k as the best value of an exhaustive search in the range $[1, |\mathcal{Z}_1|]$ with step size of 2 (i.e. the best value of k is the one that maximizes the accuracy over the training set). For the EPNN classifier, the search space of parameter σ (variance of the Gaussian function used in the pattern layer) was defined within $[0, 1]$ with step size of 0.05, and the search space for the radius was defined within $[l_d, m_d]$, where l_d and m_d denote the lowest and greatest distance among two samples. The ANN architecture employs 4 hidden layers with 8 neurons on each, and the number of epochs and desired error were defined as $70,000$ and 0.0001, respectively. Notice all these experimental setup was defined empirically.

A. Small-sized Datasets

Table IV-A presents the mean recognition rates using 50% of the datasets for training purposes without feature normalization, where the most accurate results according to the

Wilcoxon statistical tool are in bold. One can observe that FEMa obtained the best results in 9 out of 18 datasets, being the sole best technique in three situations (i.e. "breast", "data2" and "data3"). Additionally, concerning five other datasets ("cone-torus", "diabetes", "glass", "segment" and "wine"), FEMa obtained recognition rates quite close to the best ones. The worst performance appears to be in the "vehicle" dataset, but the same has happened to all classifiers, except for SVM, which obtained the best results for this dataset so far.

Table IV presents the mean recognition rates with non-normalized features. Once again, FEMa obtained the best results in 9 out 18 datasets, with recognition rates close to the best ones in three more datasets. Actually, the only situation that seems to be affected by non-normalized features is the "breast" dataset, though all techniques were affected either. In this new experiment, FEMa obtained the sole best result in "wine" dataset only.

Table V presents the mean computational load for training purposes. Notice we did not show the results concerning FEMa, since it does not have training step. The most expensive techniques are the ones that require parameter fine-tuning (i.e. k-NN, SVM and EPNN), since we considered the time spent on this step to the final training procedure computational load. Our implementation of the Bayesian classifier is considerably fast for training, since it basically consists into finding the maximum arc-weight among training samples to be used as a normalization factor in the exponential function (probability estimates).

Table VI presents the mean computational load concerning the classification time over the small-sized datasets. Clearly, one can observe all techniques are considerably fast, since the datasets do not comprise so many samples. In this experiment, FEMa seems to be the slowest one, but if one considers the whole procedure (i.e. training+classification), FEMa and Bayes are the fastest ones, being FEMa more accurate than Bayes in a larger number of situations.

B. Medium-to-large-sized Datasets

Table VII presents the recognition rates of the medium-to-large datasets used in this work with normalized features, where the best results according to Wilcoxon signed-rank test are in bold. In this case, SVM obtained the best results for all datasets, followed by FEMa, k-NN and Bayes. Since the medium-to-sized datasets have a considerable number of samples for training purposes, SVM can benefit from that, since the samples will be mapped to a higher dimensional space for learning the maximum-marge hyperplane. However, its training step is too costly, as showed in Table VIII. Actually, except for Bayes and OPF, all other classifiers required a considerable computational load for training, which is prohibitive in real-time learning systems, where the training set dynamics changes over time. In this situation, FEMa seems to be most suitable approach, since it does not require the training step.

Table IX presents the classification load for all techniques. One can observe ANN as the fastest approach, since it basically needs to forward the input data to the layers computing inner products between the activation values and the weights. However, if one considers the whole computational time (i.e. training+classification), Bayes was the fastest approach followed by FEMa, though the latter being more accurate.

As aforementioned in Section III-D, FEMa uses a k-neighborhood to compute the probability function of each test sample to avoid problems with unbalanced datasets. By using kd-trees, for instance, we can make FEMa faster by a factor of $|\mathcal{Z}_1|/\alpha$, where α is the number of elements of the smallest class.

C. Discussion

The proposed FEMa classifier was compared against six other supervised pattern recognition techniques in two distinct scenarios: small and medium-to-large datasets. Also, with respect to the former situation, we also considered normalized and non-normalized datasets.

From both results, we can observe that FEMa has been placed in the top two first positions for almost all datasets, and it seems to not be affected by non-normalized features. Since FEMa does not require a training step, it has been placed as the second fastest approach (i.e. training+classification), just behind the Bayesian classifier, which has a very fast and simple training phase either. While FEMa has obtained the best results in 9 datasets (Table IV), Bayes achieved the most accurate results in 6 situations, being the sole best technique in only one dataset (FEMa obtained the best results solely in 3 datasets).

Another interesting point about FEMa concerns its possibility to be extended, since the reader can evaluate other basis functions to interpolate the probabilistic manifold, as well as we can try to make FEMa even faster by means of kd-trees, which are often used to speed up the k-nearest neighbours classifier. Therefore, we believe a framework to the development of pattern recognition techniques based on Finite Element Method has been proposed, instead of a single supervised classifier only. Since this version is parameterless, it becomes easier to use and less prone to errors, besides being a deterministic classifier.

V. Conclusions and Future Works

Supervised pattern recognition techniques have been paramount in the last years, mainly due to the increasing number of applications that make use of some decision-making mechanism. Also, the number of new data available at the internet every single day makes some techniques unfeasible to be trained online. That is a crucial shortcoming in several situations, such as active and semi-supervised learning, and intrusion detection in computer networks, for instance. Recommendation systems may be affected, since such models need to be dynamic enough to handle the so-called "concept drift", i.e. when a user suddenly changes its expected behaviour, thus requiring a new training procedure with the updated data.

In this paper, we proposed FEMa - A Finite Element Machine classifier based on the Finite Element Method theory, which has been extensively used for several purposes in engineering and sciences, but never for classification purposes. The main idea is to learn a probabilistic manifold built upon

TABLE III
MEAN CLASSIFICATION RATES USING 50% OF THE SAMPLES FOR TRAINING WITH NORMALIZED FEATURES.

Dataset	ANN	Bayes	OPF	FEMa	k-NN	SVM	EPNN
australian	80.71 ± 2.67	78.62 ± 1.25	77.71 ± 1.71	81.61 ± 1.46	$\mathbf{84.54 \pm 1.06}$	$\mathbf{85.46 \pm 1.02}$	79.08 ± 2.96
boat	79.41 ± 7.40	96.08 ± 2.59	95.83 ± 2.67	95.59 ± 1.90	96.08 ± 2.59	$\mathbf{99.02 \pm 1.96}$	95.29 ± 2.35
breast	96.77 ± 0.96	95.30 ± 1.13	94.99 ± 1.28	$\mathbf{97.08 \pm 0.27}$	96.85 ± 0.44	90.06 ± 0.61	90.06 ± 0.61
cone-torus	62.32 ± 4.52	82.25 ± 1.56	81.66 ± 1.39	82.26 ± 1.79	82.26 ± 2.55	$\mathbf{82.65 \pm 4.01}$	81.19 ± 2.05
data1	98.97 ± 0.25	99.45 ± 0.25	99.40 ± 0.21	$\mathbf{99.59 \pm 0.18}$	$\mathbf{99.53 \pm 0.18}$	99.35 ± 0.24	99.35 ± 0.24
data2	98.16 ± 0.49	98.35 ± 0.85	98.07 ± 0.68	$\mathbf{98.62 \pm 0.72}$	98.50 ± 0.91	98.50 ± 0.76	98.03 ± 0.82
data3	91.08 ± 8.81	98.83 ± 1.01	98.55 ± 1.29	$\mathbf{99.29 \pm 0.58}$	98.74 ± 1.00	99.09 ± 0.85	90.06 ± 2.09
data4	99.50 ± 0.79	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	99.89 ± 0.14
data5	99.40 ± 1.03	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	97.73 ± 0.00
diabetes	$\mathbf{71.78 \pm 1.42}$	68.17 ± 1.79	66.92 ± 1.61	68.47 ± 1.86	66.64 ± 1.34	70.30 ± 1.11	65.68 ± 0.45
fourclass	74.68 ± 9.85	$\mathbf{99.90 \pm 0.14}$	99.45 ± 0.32	$\mathbf{99.90 \pm 0.14}$	$\mathbf{99.94 \pm 0.12}$	$\mathbf{99.91 \pm 0.13}$	98.75 ± 0.65
glass	34.79 ± 12.25	$\mathbf{63.08 \pm 5.59}$	62.65 ± 5.31	61.76 ± 6.18	58.57 ± 9.66	58.45 ± 6.21	62.49 ± 1.87
heart	77.60 ± 3.63	74.52 ± 2.93	73.02 ± 3.92	79.73 ± 2.14	$\mathbf{82.38 \pm 2.20}$	$\mathbf{82.44 \pm 2.01}$	75.26 ± 3.30
ionosphere	86.76 ± 3.25	80.45 ± 2.40	80.06 ± 2.45	60.83 ± 1.58	77.97 ± 2.90	93.51 ± 2.56	66.93 ± 0.23
petals	99.04 ± 1.36	$\mathbf{99.28 \pm 0.93}$	$\mathbf{99.28 \pm 0.93}$	$\mathbf{99.28 \pm 0.93}$	$\mathbf{99.28 \pm 0.93}$	98.56 ± 1.86	99.03 ± 0.94
saturn	58.88 ± 4.14	$\mathbf{87.88 \pm 3.06}$	87.50 ± 2.69	$\mathbf{87.85 \pm 3.11}$	$\mathbf{87.88 \pm 3.06}$	87.50 ± 3.28	86.80 ± 3.31
segment	48.29 ± 10.79	95.16 ± 0.69	94.84 ± 0.60	95.03 ± 0.56	95.16 ± 0.69	$\mathbf{95.98 \pm 0.78}$	95.48 ± 0.24
vehicle	50.16 ± 9.50	68.37 ± 1.17	67.30 ± 1.10	70.00 ± 1.33	68.18 ± 1.88	$\mathbf{81.81 \pm 1.29}$	69.29 ± 1.10
wine	94.81 ± 2.79	95.66 ± 1.48	94.85 ± 1.21	97.57 ± 1.13	96.76 ± 2.35	$\mathbf{98.53 \pm 0.92}$	93.56 ± 2.67

TABLE IV
MEAN CLASSIFICATION RATES USING 50% OF THE SAMPLES FOR TRAINING WITH NON-NORMALIZED FEATURES.

Dataset	ANN	Bayes	OPF	FEMa	k-NN	SVM	EPNN
australian	75.61 ± 7.70	80.20 ± 2.06	79.12 ± 1.63	81.85 ± 1.99	$\mathbf{85.53 \pm 1.64}$	$\mathbf{85.59 \pm 1.52}$	80.21 ± 0.92
boat	87.75 ± 5.94	95.34 ± 2.93	94.61 ± 3.06	93.87 ± 3.97	95.34 ± 2.93	$\mathbf{100.0 \pm 0.0}$	94.02 ± 1.88
breast	50.00 ± 0.00	$\mathbf{56.49 \pm 1.63}$	$\mathbf{56.55 \pm 1.64}$	$\mathbf{56.57 \pm 1.59}$	52.95 ± 2.37	51.39 ± 1.08	54.04 ± 2.01
cone-torus	56.85 ± 23.66	81.48 ± 2.87	81.00 ± 3.12	81.55 ± 3.16	81.58 ± 3.66	$\mathbf{84.03 \pm 3.26}$	79.98 ± 1.21
data1	91.57 ± 15.78	$\mathbf{99.45 \pm 0.28}$	99.33 ± 0.42	$\mathbf{99.45 \pm 0.28}$	$\mathbf{99.49 \pm 0.28}$	99.22 ± 0.10	97.59 ± 3.85
data2	96.15 ± 3.54	98.40 ± 0.77	$\mathbf{98.65 \pm 0.79}$	98.40 ± 0.77	98.38 ± 0.85	$\mathbf{98.65 \pm 0.73}$	97.21 ± 2.07
data3	43.88 ± 22.45	$\mathbf{99.55 \pm 0.94}$	$\mathbf{99.55 \pm 0.94}$	$\mathbf{99.55 \pm 0.94}$	99.27 ± 0.87	99.27 ± 0.63	98.55 ± 0.74
data4	85.02 ± 19.50	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	99.16 ± 1.01
data5	73.77 ± 23.83	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
diabetes	63.48 ± 7.29	66.03 ± 1.43	64.92 ± 1.52	67.05 ± 1.94	66.76 ± 2.61	$\mathbf{71.13 \pm 2.39}$	68.17 ± 0.87
fourclass	80.19 ± 11.34	$\mathbf{99.92 \pm 0.14}$	99.79 ± 0.18	$\mathbf{99.92 \pm 0.14}$	$\mathbf{99.90 \pm 0.19}$	$\mathbf{99.91 \pm 0.13}$	89.99 ± 10.02
glass	32.67 ± 3.92	62.68 ± 5.07	62.68 ± 5.00	61.73 ± 4.85	52.47 ± 10.95	$\mathbf{63.79 \pm 4.62}$	62.98 ± 1.08
heart	74.27 ± 4.08	75.19 ± 3.00	75.21 ± 2.62	77.92 ± 2.30	$\mathbf{82.04 \pm 1.97}$	$\mathbf{82.12 \pm 1.95}$	80.74 ± 1.22
petals	98.80 ± 1.91	$\mathbf{99.52 \pm 1.27}$	$\mathbf{99.52 \pm 1.27}$	$\mathbf{99.52 \pm 1.27}$	$\mathbf{99.52 \pm 1.27}$	99.04 ± 0.96	98.94 ± 0.62
saturn	64.38 ± 11.38	$\mathbf{89.62 \pm 6.08}$	$\mathbf{89.50 \pm 6.16}$	$\mathbf{89.68 \pm 6.03}$	$\mathbf{89.62 \pm 6.08}$	$\mathbf{89.50 \pm 6.42}$	85.28 ± 3.39
segment	44.09 ± 14.78	95.43 ± 0.70	95.11 ± 0.73	95.60 ± 0.58	95.43 ± 0.70	$\mathbf{96.09 \pm 0.60}$	68.92 ± 3.05
vehicle	53.28 ± 9.48	68.28 ± 1.65	67.45 ± 1.46	68.81 ± 1.51	68.66 ± 2.17	$\mathbf{82.71 \pm 1.30}$	63.33 ± 1.35
wine	93.06 ± 5.41	96.04 ± 1.34	94.94 ± 1.80	$\mathbf{98.09 \pm 0.22}$	97.43 ± 1.34	96.17 ± 1.23	95.28 ± 2.06

TABLE V
MEAN TRAINING TIME USING 50% OF THE SAMPLES FOR TRAINING WITH NORMALIZED FEATURES.

Dataset	ANN	Bayes	OPF	k-NN	SVM	EPNN
australian	12.61 ± 9.63	0.00 ± 0.00	2.61 ± 0.12	0.18 ± 0.01	11.01 ± 1.32	15.22 ± 3.11
boat	1.14 ± 1.36	0.00 ± 0.00	0.14 ± 0.00	0.00 ± 0.00	0.60 ± 0.03	2.10 ± 0.14
breast	18.40 ± 0.19	0.00 ± 0.00	0.97 ± 0.01	0.18 ± 0.00	3.87 ± 0.20	47.55 ± 8.43
cone-torus	11.05 ± 0.10	0.00 ± 0.00	1.02 ± 0.10	0.02 ± 0.00	4.76 ± 0.59	3.13 ± 0.88
data1	3.12 ± 3.79	0.00 ± 0.00	2.21 ± 0.07	1.47 ± 0.03	8.26 ± 0.20	3.81 ± 0.29
data2	0.17 ± 0.26	0.00 ± 0.00	0.19 ± 0.01	0.00 ± 0.00	0.82 ± 0.02	0.15 ± 0.02
data3	6.60 ± 5.21	0.00 ± 0.00	0.28 ± 0.00	0.01 ± 0.00	1.21 ± 0.02	1.68 ± 0.29
data4	3.94 ± 7.61	0.00 ± 0.00	0.41 ± 0.01	0.09 ± 0.02	2.12 ± 0.02	4.03 ± 0.97
data5	0.16 ± 0.01	0.00 ± 0.00	1.21 ± 0.02	5.06 ± 0.21	5.54 ± 0.08	0.86 ± 0.33
diabetes	21.14 ± 0.31	0.00 ± 0.00	6.29 ± 0.21	0.11 ± 0.01	26.52 ± 2.52	11.22 ± 1.08
fourclass	17.46 ± 8.52	0.00 ± 0.00	2.88 ± 0.09	0.31 ± 0.02	9.68 ± 0.24	13.22 ± 4.51
glass	8.29 ± 0.07	0.00 ± 0.00	0.27 ± 0.00	0.00 ± 0.00	1.97 ± 0.08	1.32 ± 0.11
heart	3.23 ± 3.84	0.00 ± 0.00	0.45 ± 0.04	0.01 ± 0.00	2.05 ± 0.14	4.02 ± 0.21
petals	0.01 ± 0.00	0.00 ± 0.00	0.18 ± 0.02	0.00 ± 0.00	0.59 ± 0.04	0.11 ± 0.01
saturn	5.18 ± 0.10	0.00 ± 0.00	0.49 ± 0.02	0.00 ± 0.00	2.09 ± 0.19	0.89 ± 0.07
segment	110.61 ± 4.29	0.03 ± 0.00	19.98 ± 0.22	17.93 ± 0.79	91.64 ± 3.76	298.13 ± 42.25
vehicle	32.39 ± 0.62	0.00 ± 0.00	5.96 ± 0.07	0.52 ± 0.04	20.72 ± 0.22	21.24 ± 2.14
wine	0.02 ± 0.01	0.00 ± 0.00	0.20 ± 0.00	0.00 ± 0.00	0.99 ± 0.02	0.09 ± 0.04

the training samples, which will become the center of a basis function each. Further, the classification process simply inserts a test sample into the manifold, and computes the probability of that sample to belong to each class.

Experiments against six other well-known supervised pattern recognition techniques showed that FEMa can obtain very competitive results, though being considerably faster than others, since it is parameterless and, in practice, it does not have a training phase. Also, FEMa do not seem to be affected by non-normalized features.

In regard to future works, we aim at extending FEMa for clustering and regression purposes, as well as to evaluate the influence of other basis functions. In addition, we shall implement its optimized version based on kd-trees.

TABLE VI
MEAN TESTING TIME.

Dataset	ANN	Bayes	OPF	EPNN	FEMa	KNN	SVM
australian	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.02 ± 0.00	0.15 ± 0.01	0.01 ± 0.00	0.01 ± 0.00
boat	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00
breast	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.09 ± 0.00	0.01 ± 0.00	0.02 ± 0.00
cone-torus	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.03 ± 0.00	0.02 ± 0.00	0.00 ± 0.00
data1	0.00 ± 0.00	0.01 ± 0.00	0.02 ± 0.00	0.03 ± 0.00	0.28 ± 0.00	0.06 ± 0.01	0.01 ± 0.00
data2	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
data3	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
data4	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.08 ± 0.01	0.04 ± 0.00	0.01 ± 0.00
data5	0.00 ± 0.00	0.02 ± 0.00	0.03 ± 0.00	0.04 ± 0.00	0.17 ± 0.00	0.09 ± 0.01	0.02 ± 0.00
diabetes	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.12 ± 0.02	0.01 ± 0.00	0.01 ± 0.00
fourclass	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.10 ± 0.00	0.01 ± 0.00	0.00 ± 0.00
glass	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
heart	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.00	0.01 ± 0.00	0.00 ± 0.00
petals	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00
saturn	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.00 ± 0.00
segment	0.00 ± 0.00	0.26 ± 0.00	0.07 ± 0.00	0.21 ± 0.00	4.53 ± 0.38	0.14 ± 0.02	0.08 ± 0.01
vehicle	0.00 ± 0.00	0.02 ± 0.00	0.01 ± 0.00	0.03 ± 0.00	0.38 ± 0.03	0.03 ± 0.00	0.05 ± 0.03
wine	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00

TABLE VII
RECOGNITION RATES CONCERNING THE MEDIUM-TO-LARGE DATASETS.

Dataset	ANN	Bayes	EPNN	FEMa	k-NN	OPF	SVM
a1a	81.43	79.32	76.10	81.83	82.58	71.27	**84.05**
a2a	81.84	79.05	76.15	81.69	82.44	71.72	**84.06**
a3a	79.46	79.23	76.08	81.74	82.71	72.53	**84.45**
a4a	81.50	79.58	79.73	81.94	83.02	73.77	**84.56**
a5a	82.05	79.61	79.56	81.99	83.06	73.43	**84.49**

TABLE VIII
TRAINING TIME CONCERNING THE MEDIUM-TO-LARGE DATASETS.

Dataset	ANN	Bayes	OPF	EPNN	k-NN	SVM
a1a	477.41 ± 7.38	0.44 ± 0.04	1.60 ± 0.14	$7,439.90 \pm 39.26$	387.06 ± 4.00	$1,130.59 \pm 8.24$
a2a	706.92 ± 7.42	0.97 ± 0.07	2.72 ± 0.14	$17,758.61 \pm 757.12$	813.43 ± 6.59	$2,292.08 \pm 12.09$
a3a	964.27 ± 4.58	1.38 ± 0.03	5.76 ± 0.19	$11,336.21 \pm 334.14$	$2,592.87 \pm 10.57$	$4,557.10 \pm 1.24$
a4a	$1,449.42 \pm 1.17$	3.75 ± 0.11	12.68 ± 0.93	$45,847.83 \pm 842.12$	$10,470.55 \pm 18.88$	$10,870.21 \pm 19.07$
a5a	$1,926.00 \pm 22.93$	6.71 ± 0.23	24.36 ± 1.05	$59,763.13 \pm 192.11$	$20,256.74 \pm 38.72$	$21,870.21 \pm 75.19$

TABLE IX
TESTING TIME CONCERNING THE MEDIUM-TO-LARGE DATASETS.

Dataset	ANN	Bayes	OPF	EPNN	FEMa	KNN	SVM
a1a	0.05 ± 0.00	42.21 ± 1.59	16.46 ± 0.91	41.37 ± 0.00	89.22 ± 0.00	20.01 ± 0.00	25.82 ± 0.76
a2a	0.06 ± 0.01	58.15 ± 0.36	24.12 ± 1.83	55.34 ± 0.00	103.46 ± 0.00	30.42 ± 1.00	41.38 ± 2.57
a3a	0.06 ± 0.01	76.48 ± 3.51	35.02 ± 2.06	74.25 ± 0.00	150.07 ± 0.00	42.36 ± 0.24	48.44 ± 1.13
a4a	0.09 ± 0.01	108.04 ± 1.44	47.92 ± 1.96	128.91 ± 0.00	168.16 ± 0.00	60.81 ± 0.88	69.12 ± 0.21
a5a	0.05 ± 0.01	140.97 ± 7.23	58.49 ± 2.66	207.23 ± 0.00	239.13 ± 1.39	92.74 ± 1.66	108.98 ± 5.23

ACKNOWLEDGMENT

The authors are grateful to FAPESP grant #2014/16250-9, FAPESP/OSU grant #2015/50319-9, as well as CNPq grant #306166/2014-3.

REFERENCES

[1] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[2] S. L. Hung and H. Adeli, "Parallel backpropagation learning algorithms on {CRAY} y-mp8/864 supercomputer," *Neurocomputing*, vol. 5, no. 6, pp. 287–302, 1993, backpropagation, Part {II}.

[3] H. Adeli and S.-L. Hung, *Machine Learning: Neural Networks, Genetic Algorithms, and Fuzzy Systems*. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[4] C. T. Lin, M. Prasad, and A. Saxena, "An improved polynomial neural network classifier using real-coded genetic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 11, pp. 1389–1401, 2015.

[5] C. M. Lin and E. A. Boldbaatar, "Autolanding control using recurrent wavelet elman neural network," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 9, pp. 1281–1291, 2015.

[6] P. Liu, Z. Zeng, and J. Wang, "Multistability of recurrent neural networks with nonmonotonic activation functions and mixed time delays," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 4, pp. 512–523, 2016.

[7] H. Adeli and S.-L. Hung, "An adaptive conjugate gradient learning algorithm for effective training of multilayer neural networks," *Applied Mathematics and Computation*, vol. 62, pp. 81–102, 1994.

[8] ——, "A concurrent adaptive conjugate gradient learning algorithm on MIMD machines," *Journal of Supercomputer Applications*, vol. 7, pp. 155–166, 1993.

[9] J. S. Chou and A. D. Pham, "Smart artificial firefly colony-based support vector regression for enhanced forecasting in civil engineering," *Computer-Aided Civil and Infrastructure Engineering*, vol. 30, no. 9, pp. 715–732, 2015.

[10] Y. LeCun, Y.Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] M. H. Rafiei and H. Adeli, "A novel machine learning model for estimation of sale prices of real estate units," *Construction Engineering*

and Management, vol. 142, 2016.

[12] J. P. Papa and A. X. Falcão, "A new variant of the optimum-path forest classifier," in *Advances in Visual Computing*, ser. Lecture Notes in Computer Science, G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, F. Porikli, J. Peters, J. Klosowski, L. Arns, Y. Chun, T.-M. Rhyne, and L. Monroe, Eds. Springer Berlin Heidelberg, 2008, vol. 5358, pp. 935–944.

[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.

[14] J. P. Papa, A. X. Falcão, V. H. C. Albuquerque, and J. M. R. S. Tavares, "Efficient supervised optimum-path forest classification for large datasets," *Pattern Recognition*, vol. 45, no. 1, pp. 512–520, 2012.

[15] J. P. Papa, S. E. N. Fernandes, and A. X. Falcão, "Optimum-path forest based on k-connectivity: Theory and applications," *Pattern Recognition Letters*, vol. 87, pp. 117–126, 2017.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[17] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[18] Q. Liu and J. Wang, "A one-layer recurrent neural network for constrained nonsmooth optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 5, pp. 1323–1333, 2011.

[19] C. T. Lin, M. Prasad, and A. Saxena, "An improved polynomial neural network classifier using real-coded genetic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 11, pp. 1389–1401, 2015.

[20] N. Martinel, C. Micheloni, and G. L. Foresti, "The evolution of neural learning systems: A novel architecture combining the strengths of NTs, CNNs, and ELMs," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 1, no. 3, pp. 17–26, 2015.

[21] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990.

[22] M. Ahmadlou and H. Adeli, "Enhanced probabilistic neural network with local decision circles: A robust classifier," *Integrated Computer-Aided Engineering*, vol. 17, no. 3, pp. 197–210, 2010.

[23] H. Adeli and A. Panakkat, "A probabilistic neural network for earthquake magnitude prediction," *Neural Networks*, vol. 22, pp. 1018–1024, 2009.

[24] Z. Sankari and H. Adeli, "Probabilistic neural networks for eeg-based diagnosis of alzheimer's disease using conventional and wavelet coherence," *Journal of Neuroscience Methods*, vol. 197, pp. 165–170, 2011.

[25] ——, "Probabilistic neural networks for diagnosis of alzheimer's disease using conventional and wavelet coherence," *Journal of Neuroscience Methods*, vol. 197, no. 1, pp. 165–170, 2011.

[26] T. J. Hirschauer, H. Adeli, and J. A. Buford, "Computer-aided diagnosis of parkinson's disease using enhanced probabilistic neural network," *Journal of Medical Systems*, vol. 39, no. 11, pp. 1–12, 2015.

[27] O. C. Zienkiewicz and Y. K. Cheung, *The Finite Element Method in Structural and Continuum Mechanics*. McGraw-Hill, 1967.

[28] G. Yu and H. Adeli, "Object-oriented finite element analysis using EER model," *Journal of Structural Engineering*, vol. 119, pp. 2763–2781, 1993.

[29] J. Lehtinen, M. Zwicker, E. Turquin, J. Kontkanen, F. Durand, F. Sillion, and T. Aila, "A meshless hierarchical representation for light transport," *ACM Trans. Graph.*, vol. 27, no. 3, 2008.

[30] D. R. Pereira, J. Stolfi, and A. Gomide, "Comparison of finite element bases for global illumination in image synthesis," in *23rd SIBGRAPI Conference on Graphics, Patterns and Images*. IEEE Computer Society, 2010, pp. 287–294.

[31] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM national conference*. ACM Press, 1968, pp. 517–524.

[32] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *Journal of Symbolic Computation*, vol. 9, pp. 251–280, 1990.

[33] J. P. Papa, C. T. N. Suzuki, and A. X, "LibOPF: A library for the design of optimum-path forest classifiers," software version 2.1 available at http://www.ic.unicamp.br/ afalcao/libopf/index.html.

[34] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.