FEMaR: A Finite Element Machine for Regression Problems

Danillo R. Pereira, Joao P. Papa
Department of Computing
São Paulo State University
Bauru, São Paulo 17033-360
Email: dpereira@ic.unicamp.br, papa@fc.unesp.br

Andre N. Souza
Department of Electrical Engineering
São Paulo State University
Bauru, São Paulo 17033-360
Email: andrejau@feb.unesp.br

***Abstract*—Regression-based tasks have been the forerunner regarding the application of machine learning tools in the context of data mining. Problems related to price and stock prediction, selling estimation, and weather forecasting are commonly used as benchmarking for the comparison of regression techniques, just to name a few. Neural Networks, Decision Trees and Support Vector Machines are the most widely used approaches concerning regression-oriented applications, since they can generalize well in a number of different applications. In this work, we propose an efficient and effective regression technique based on the Finite Element Method (FEM) theory, hereinafter called Finite Element Machine for Regression (FEMaR). The proposed approach has only one parameter and it has a quadratic complexity for both training and classification phases when we use basis functions that obey some properties, as well as we show the proposed approach can obtain very competitive results when compared against some state-of-the-art regression techniques.**

I. Introduction

Machine learning techniques have been actively pursued in the last decades, since there is a growing number of applications that require some sort of intelligent-based decision-making mechanism. Recently, deep learning-oriented works have pushed machine learning to the cutting-edge research related to unsupervised learning features from large datasets. For some applications, deeply learnable features work much better than handcrafted ones, but at the price of needing a considerable amount of data for learning purposes, otherwise the model can get overfitted.

Similarly to pattern classification, regression techniques aim at learning a function that can predict outputs given some input data, being the output not a single label, but any real-valued outcome instead. Problems related to weather forecasting, selling price and stock prediction are the most related ones when one talks about regression applications. As such, the reader can refer to a number of techniques to handle such problem, such as Neural Networks [1], Support Vector Regression (SVR) [2], Gaussian Processes [3], and Bayesian regression [4], just to name a few.

Although we can refer to a number of regression techniques in the literature, there is still room for improvements. Support Vector Regression, for instance, might be one of the most effective ones, but at the price of a high computational load to find out suitable parameters related to the kernel mapping process. On the other hand, Decision Trees are usually quite fast, but they may not generalize well for some complex problems. Such side effects, among others, have motivated us to think about alternatives and new solutions for regression-like problems.

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) [5], [6]. 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 FEMaR - Finite Element Machine for Regression, a new framework for the design of pattern classifiers based on finite element analysis. Depending on the basis function used, FEMaR can be parameterless. It also 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, FEMaR 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 FEMaR can obtain very competitive results when compared against some state-of-the-art regression techniques.

The remainder of this paper is organized as follows. Sections II and III introduce the theoretical background related to FEM and FEMaR, respectively. Section IV presents the methodology and experiments used to evaluate FEMa in the context of regression problems, 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

978-1-5090-6182-2/17/$31.00 ©2017 IEEE

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:

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

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

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

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:

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

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:

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

and

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

Such basis has smoothing properties, as follows:

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

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:

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

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.

In this work, we use a special class of finite element bases, which are defined by points (meshless) [7], [8]. 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 [9], each element is defined as follows:

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

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:

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

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.

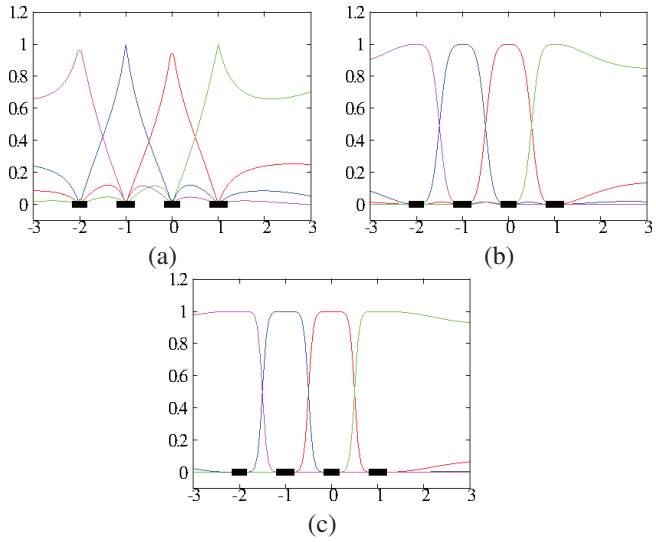

Fig. 1. Behaviour of different Shepard bases according to three values of k, where the black dots stand for the center of the basis: (a) $k = 1$, (b) $k = 3$ and (c) $k = 5$.

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.

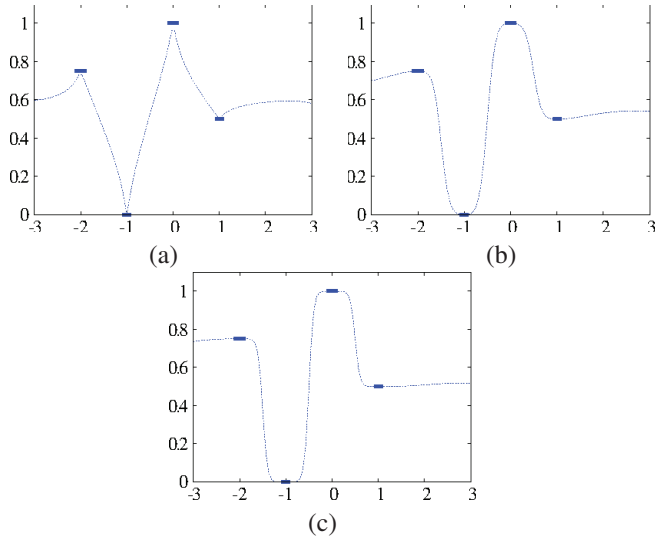

Fig. 2. Interpolated function using the Shepard basis for (a) $k = 1$, (b) $k = 3$ and (c) $k = 5$. The blue rectangles represent the center of the basis and their sampled values.

III. Finite Element Machine for Regression

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 (independent variables) $\mathbf{x}_i \in \Re^m$ extracted from sample i, and $y_i \in \Re$ stands for the value of dependent variables (output). Notice we adopted the very same formulation used in the previous section, i.e. a point in FEM formulation stands for a sample in FEMaR.

B. Manifold Learning

Depending on the basis function used to interpolate points, FEMaR 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 FEMaR 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 testing step. Given a sample $\mathbf{x} \in \mathcal{Z}_2$, we need to compute its dependent variable y, as follows:

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

where y_j stands for the sampled value of x_j. Roughly speaking, the estimation (testing) phase of FEMaR aims at assigning a real value to each test sample that is basically a weighted combination of all bases centered at the training samples. As a matter of fact, the weights are, essentially, the output of each training sample.

C. Toy Example

In this section, we present the FEMaR working mechanism on a bidimensional classification problem. Figure 3a shows a training set with samples distributed over a feature space. Concerning this example, we consider three different values to be estimated, which are represented by three different colors (i.e. red, green and blue). The task is to verify the influence region of each training sample in the image domain, i.e. we want "to interpolate" the remaining points (white ones) in the image frame displayed in Figure 3a. In this case, each sample (point) is described by its (x, y)-position, and the dependent value (output to be predicted) is a different color. The regression outputs of different regression methods are displayed in Figures 3b-g.

One can observe a very much clear difference among the methods: while decision trees generate discrete influence regions, SVR with Linear kernel and Bayesian Ridge regression seem to get confused, since the influence region of each training sample is not clearly defined. On the other hand, SVR with a Radial Basis Function kernel and FEMaR are capable to generate well-bounded influence regions for each training sample (colored points). However, FEMaR can also cover a larger area of the feature space, thus being able to generalize better when one lacks data.

Figures 4b, 4c and 4d depict the image frame after regression by FEMaR 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 image refers to the influence of the training samples and their estimated values (color). Notice that FEMaR can obtain quite good and smooth functions for different values of k (Equation 13). As matter of fact, the larger the value of k, the less points will influence the regression process.

D. Complexity Analysis

As aforementioned, depending on the basis function used to build the manifold function (i.e. interpolating and partition of unity properties), FEMaR 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 [10].

In regard to the regression 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 regression phase takes $\theta(|\mathcal{Z}_1| + |\mathcal{Z}_1||\mathcal{Z}_2|) \in \theta(|\mathcal{Z}_1|.|\mathcal{Z}_2|)$.

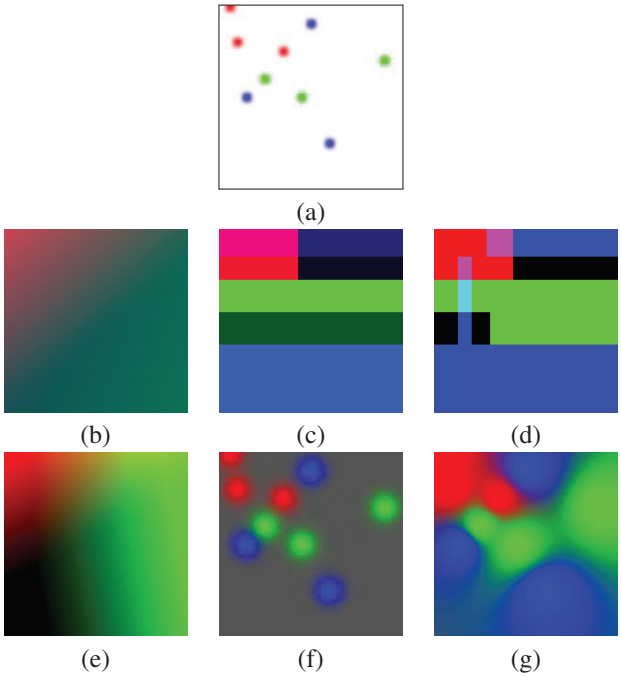

Fig. 3. Regression example: (a) samples in the feature space, (b) Bayesian Ridge Regression, (c) Decision Tree with maximum depth of 2, (d) Decision Tree with maximum depth of 5, (e) SVR with Linear kernel (parameter $C = 0.0001$), (f) SVR with Radial Basis Function kernel ($C = 0.001$ and $\gamma = 0.1$, and (g) FEMaR.

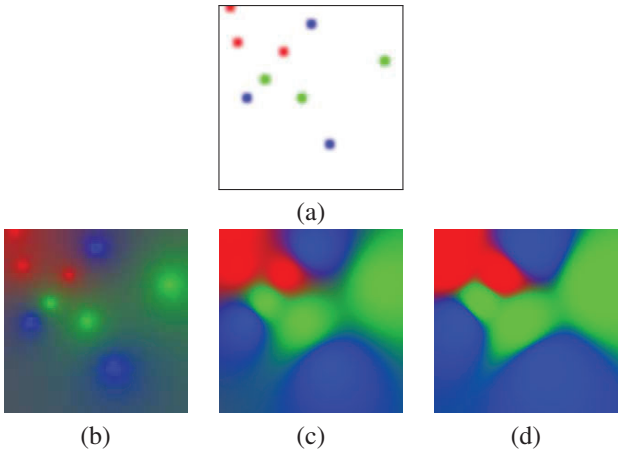

Fig. 4. FEMaR working mechanism: (a) training set with samples distributed in three classes, and the regression outputs generated by FEMaR using (b) $k = 1$, (c) $k = 3$ and (d) $k = 5$.

Therefore, by using an interpolating basis function, the whole FEMaR learning and classification processes require a quadratic complexity with respect to the training/testing set size (i.e. when $|\mathcal{Z}_1| = |\mathcal{Z}_2|$).

IV. Experiments

In this section, we present the methodology and the experimental setup used to asses the robustness and efficiency of FEMaR against four other regression methods: (i) Bayesian Ridge Regression (Bayes-R), (ii) Decision Trees (DT), (iii)

Support Vector Regression with Linear basis function (SVR-L), and (iv) Support Vector Regression with Radial Basis Function (SVR-RBF). Such approaches were selected for comparison purposes since they have been commonly applied in a wide number of applications in the literature.

Concerning Decision Trees, we used two distinct versions: one composed of trees with maximum depth of 2, and another composed of trees with maximum depth 5. With respect to SVR-L, the parameter C was set to 0.0001, and concerning SVR-RBM the parameter C was defined as 0.001, and γ as 0.1. Finally, with respect to FEMaR, parameter k was defined as 3[1]. In regard to the implementation, the four regression techniques used in this work come from scikit-learning toolbox [11], and concerning FEMaR we used our own implementation.

In order to validate the experiments, we employed 14 public benchmarking datasets[2] that have been frequently used for the evaluation of regression methods. Table I presents the main characteristics of the datasets, which were selected in order to model distinct scenarios, which comprise datasets with different number of features, sizes and normalized/non-normalized features.

TABLE I
INFORMATION ABOUT THE USED IN THE EXPERIMENTS.

Dataset	# samples	# features
abalone	4.177	8
abalone-scale	4.177	8
bodyfat	252	14
bodyfat-scale	252	14
housing	506	13
housing-scale	506	13
mg	1.385	6
mg-scale	1.385	6
mpg	392	7
mpg-scale	392	7
pyrim	47	27
pyrim-scale	47	27
triazines	186	60
triazines-scale	186	60

The datasets were partitioned at random using 25%, 50% and 75% for training purposes, being the remaining samples for testing purposes. Notice the aforementioned protocol was repeated under 15 runnings for the computation of the Mean Squared Error (MSE) and computational load in seconds. The idea is to verify the behavior of FEMaR under training sets with different sizes. Additionally, the Wilcoxon signed-rank statistical test [12] with significance of 0.05 was used to validate the results.

Table II presents the MSE results concerning a training set with 25% of the entire image for learning purposes. The most accurate results according to the statistical test are in bold. FEMaR obtained the best results in 6 out 14 datasets, Bayes-R obtained and SVR-RBF achieved the best results in 10 and 8 datasets, respectively. However, for some situations (i.e. "abalone-scale" and "housing-scale") FEMaR obtained much better results than all techniques, as the opposite has happened either (e.g. "bodyfat-scale" and "mpg-scale"). In regard to normalized/non-normalized datasets, some techniques seem to be highly affected, such as SVR-L in "housing" and "mpg" datasets, for instance. In our case, FEMaR does not appear to be considerably affected by non-normalized features.

Table III presents the mean computational load in seconds concerning the training and testing steps. The fastest techniques where Bayes-R and the ones based on Decision Trees, followed by FEMaR and SVR-RBF. SVR-L took longer in the situations where its results were the worst ones in Table II. Since SVR is essentially an optimization problem, SVR-L did not find a reasonable result for some datasets, thus taking longer in the search for better solutions (e.g. "housing" and "mpg").

Tables IV and V present the MSE values concerning training sets with 50% and 75% of the entire dataset, respectively. Concerning 50% of the dataset for training purposes, FEMaR obtained the best results in 6 out of 14 datasets, and with respect to 75%, the proposed approach obtained the best results in 7 out of 14 datasets. Roughly speaking, FEMaR also benefit from larger datasets, which is usually expected in machine learning applications.

Figure 5 depicts the robustness of FEMaR with respect to different values of k concerning "triazines" dataset. One can observe that FEMaR gets little affected by different values of k, which is interesting, since we can save time when looking for the k-nearest neighbors of a given testing sample. We opted to use that dataset due to their small size, thus requiring low computational burden.

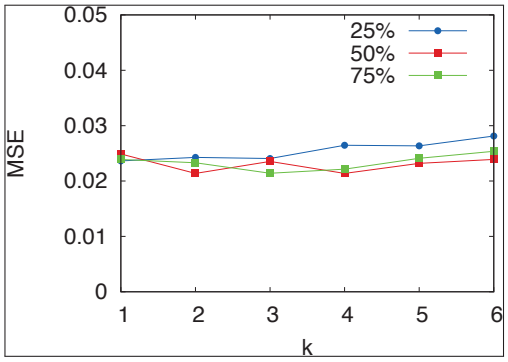

Fig. 5. Robustness of FEMaR with respect to different values of k concerning "triazines" dataset.

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. In this paper, we proposed FEMaR - A Finite Element Machine for Regression problems based on the Finite Element Method theory, which has been extensively used

[1]All parameters were empirically defined by means of a cross-validation procedure.

[2]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

Dataset	Bayes-R	DTree2	DTree5	SVR-L	SVR-RBF	FEMaR
abalone	4.98 ± 0.08	6.72 ± 0.10	6.00 ± 0.22	5.18 ± 0.11	$\mathbf{4.71 \pm 0.08}$	5.68 ± 0.19
abalone-scale	4.95 ± 0.12	6.68 ± 0.16	6.01 ± 0.22	5.12 ± 0.17	4.77 ± 0.12	$\mathbf{4.54 \pm 0.26}$
bodyfat	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
bodyfat-scale	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	6.02 ± 1.18
housing	$\mathbf{29.34 \pm 2.24}$	35.75 ± 5.11	31.12 ± 4.47	2521.17 ± 1338.29	82.11 ± 3.86	38.97 ± 5.92
housing-scale	26.34 ± 2.47	32.18 ± 4.37	28.49 ± 7.75	27.98 ± 4.91	20.28 ± 5.10	$\mathbf{11.72 \pm 2.93}$
mg	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$
mg-scale	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$
mpg	$\mathbf{12.04 \pm 0.71}$	20.89 ± 1.93	16.19 ± 1.84	$4132207.71 \pm 3463504.50$	59.92 ± 3.03	21.44 ± 1.55
mpg-scale	11.73 ± 0.67	21.33 ± 1.81	15.12 ± 2.18	12.59 ± 0.58	$\mathbf{10.05 \pm 1.00}$	49.56 ± 34.51
pyrim	$\mathbf{0.01 \pm 0.00}$	0.02 ± 0.00	0.02 ± 0.01	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.01 \pm 0.00}$
pyrim-scale	$\mathbf{0.01 \pm 0.00}$	0.02 ± 0.01	0.02 ± 0.00	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.01 \pm 0.00}$	7.77 ± 4.20
triazines	$\mathbf{0.02 \pm 0.00}$	0.03 ± 0.01	0.03 ± 0.01	0.08 ± 0.04	0.03 ± 0.01	0.03 ± 0.00
triazines-scale	$\mathbf{0.02 \pm 0.00}$	0.03 ± 0.00	0.03 ± 0.01	0.18 ± 0.10	0.03 ± 0.01	56.99 ± 28.24

TABLE II
MSE CONSIDERING A TRAINING SIZE WITH 25% OF THE ENTIRE DATASET.

Dataset	Bayes-R	DTree2	DTree5	SVR-L	SVR-RBF	FEMaR
abalone	0.01 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	1.73 ± 0.29	0.38 ± 0.08	0.81 ± 0.17
abalone-scale	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	2.54 ± 0.50	0.45 ± 0.07	0.79 ± 0.16
bodyfat	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
bodyfat-scale	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
housing	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	60.24 ± 17.09	0.01 ± 0.00	0.02 ± 0.01
housing-scale	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.29 ± 0.34	0.03 ± 0.01	0.01 ± 0.00
mg	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.55 ± 0.09	0.11 ± 0.02	0.08 ± 0.01
mg-scale	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	2.36 ± 0.68	0.33 ± 0.06	0.08 ± 0.01
mpg	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	19.54 ± 10.09	0.00 ± 0.00	0.00 ± 0.00
mpg-scale	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.08 ± 0.05	0.02 ± 0.00	0.00 ± 0.00
pyrim	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
pyrim-scale	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
triazines	0.02 ± 0.02	0.00 ± 0.00	0.00 ± 0.00	0.20 ± 0.20	0.00 ± 0.00	0.01 ± 0.00
triazines-scale	0.01 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.68 ± 0.41	0.00 ± 0.00	0.00 ± 0.00

TABLE III
MEAN COMPUTATIONAL LOAD (SECONDS) CONSIDERING A TRAINING SET WITH 25% OF THE ENTIRE DATASET.

Dataset	Bayes-R	DTree2	DTree5	SVR-L	SVR-RBF	FEMaR
abalone	4.92 ± 0.16	6.66 ± 0.16	5.66 ± 0.29	5.13 ± 0.21	$\mathbf{4.63 \pm 0.18}$	5.45 ± 0.19
abalone-scale	4.89 ± 0.16	6.63 ± 0.22	5.50 ± 0.27	5.11 ± 0.23	$\mathbf{4.63 \pm 0.18}$	5.68 ± 2.37
bodyfat	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
bodyfat-scale	$\mathbf{0.00 \pm 0.00}$	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	0.00 ± 0.00	0.00 ± 0.00	5.19 ± 1.27
housing	26.03 ± 3.21	32.29 ± 4.09	$\mathbf{23.67 \pm 6.53}$	1956.37 ± 909.42	77.86 ± 6.53	32.22 ± 6.29
housing-scale	23.26 ± 2.22	29.78 ± 3.06	20.13 ± 5.44	24.00 ± 3.34	14.01 ± 4.70	$\mathbf{10.03 \pm 2.47}$
mg	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$
mg-scale	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$
mpg	$\mathbf{11.50 \pm 1.02}$	19.16 ± 1.46	12.48 ± 2.11	106960.37 ± 56646.27	58.02 ± 4.17	18.65 ± 1.16
mpg-scale	11.88 ± 0.90	20.69 ± 2.13	13.23 ± 1.91	12.61 ± 1.03	$\mathbf{7.97 \pm 1.12}$	43.36 ± 17.37
pyrim	$\mathbf{0.01 \pm 0.00}$	0.02 ± 0.01	0.02 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	$\mathbf{0.01 \pm 0.00}$
pyrim-scale	$\mathbf{0.01 \pm 0.00}$	0.02 ± 0.01	0.01 ± 0.01	$\mathbf{0.01 \pm 0.00}$	0.01 ± 0.01	4.32 ± 2.08
triazines	$\mathbf{0.02 \pm 0.00}$	0.03 ± 0.01	0.02 ± 0.01	0.03 ± 0.00	0.03 ± 0.01	$\mathbf{0.02 \pm 0.00}$
triazines-scale	$\mathbf{0.02 \pm 0.00}$	0.03 ± 0.00	$\mathbf{0.02 \pm 0.00}$	0.03 ± 0.01	0.03 ± 0.00	60.45 ± 30.35

TABLE IV
MSE CONSIDERING A TRAINING SIZE WITH 50% OF THE ENTIRE DATASET.

for several purposes in engineering and sciences, but not for regression purposes. The main idea is to learn a probabilistic manifold built upon the training samples, which will become the center of a basis function each. Further, the regression process simply inserts a test sample into the manifold, and computes the output given by FEMaR.

Experiments against five other well-known regression techniques in 14 datasets showed that FEMaR can obtain very competitive results, though being considerably fast and, in practice, it does not have a training phase. In regard to future works, we aim at extending FEMaR for clustering problems, as well as to evaluate the influence of other basis functions. In addition, we shall implement its optimized version based on kd-trees to estimate the parameter k.

ACKNOWLEDGMENT

The authors are grateful to FAPESP grants #2013/07375-0, #2014/16250-9 and #2014/12236-1, as well as CNPq grant #306166/2014-3.

REFERENCES

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007.

Dataset	Bayes-R	DTree2	DTree5	SVR-L	SVR-RBF	FEMaR
abalone	4.86 ± 0.40	6.78 ± 0.47	5.55 ± 0.50	5.07 ± 0.44	$\mathbf{4.58 \pm 0.43}$	5.48 ± 0.46
abalone-scale	4.95 ± 0.26	6.57 ± 0.45	5.53 ± 0.35	5.04 ± 0.28	$\mathbf{4.49 \pm 0.22}$	$\mathbf{4.45 \pm 0.17}$
bodyfat	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
bodyfat-scale	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	5.43 ± 2.82
housing	25.69 ± 3.91	30.29 ± 5.47	$\mathbf{22.84 \pm 5.61}$	2602.42 ± 767.78	70.09 ± 9.03	27.64 ± 3.14
housing-scale	26.31 ± 7.18	32.14 ± 8.37	22.48 ± 9.26	28.64 ± 8.78	15.87 ± 9.91	$\mathbf{8.44 \pm 3.45}$
mg	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$
mg-scale	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$
mpg	$\mathbf{11.10 \pm 1.63}$	19.24 ± 2.30	13.27 ± 3.46	13365 ± 5828291.74	56.99 ± 5.23	16.87 ± 2.26
mpg-scale	12.03 ± 2.29	20.38 ± 2.25	13.58 ± 2.51	12.86 ± 2.73	$\mathbf{7.96 \pm 2.45}$	59.30 ± 40.61
pyrim	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$
pyrim-scale	0.02 ± 0.01	0.02 ± 0.01	0.02 ± 0.01	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	7.43 ± 9.22
triazines	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.02 \pm 0.01}$	-	0.03 ± 0.01	$\mathbf{0.02 \pm 0.01}$
triazines-scale	$\mathbf{0.02 \pm 0.00}$	0.03 ± 0.01	$\mathbf{0.02 \pm 0.01}$	0.03 ± 0.01	0.03 ± 0.01	56.33 ± 30.06

TABLE V

MSE CONSIDERING A TRAINING SIZE WITH 75% OF THE ENTIRE DATASET. THE SYMBOL '-' MEANS THE METHOD DID NOT CONVERGE AT A REASONABLE TIME.

[2] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in Neural Information Processing Systems 9*, M. I. Jordan and T. Petsche, Eds. MIT Press, 1997, pp. 155–161.

[3] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[4] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York, USA: Wiley-Interscience, 2000.

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

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

[7] 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.

[8] 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.

[9] 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.

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

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

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