Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 126452, 17 pages
http://dx.doi.org/10.1155/2015/126452

Research Article

Stream-Based Extreme Learning Machine Approach for Big Data Problems

Euler Guimarães Horta,[1,2] Cristiano Leite de Castro,[1,3] and Antônio Pádua Braga[1,4]

[1] *Graduate Program in Electrical Engineering, Federal University of Minas Gerais, Avenida Antônio Carlos 6627, 31270-901 Belo Horizonte, MG, Brazil*
[2] *Institute of Science and Technology, Federal University of Jequitinhonha and Mucuri Valleys, Rodovia MGT 367, Km 583, 5000 Alto da Jacuba, 39100-000 Diamantina, MG, Brazil*
[3] *Department of Electrical Engineering, Federal University of Minas Gerais, Avenida Antônio Carlos 6627, 31270-901 Belo Horizonte, MG, Brazil*
[4] *Department of Electronics Engineering, Federal University of Minas Gerais, Avenida Antônio Carlos 6627, 31270-901 Belo Horizonte, MG, Brazil*

Correspondence should be addressed to Euler Guimarães Horta; euler.horta@ict.ufvjm.edu.br

Received 15 May 2015; Accepted 17 August 2015

Academic Editor: Huaguang Zhang

Big Data problems demand data models with abilities to handle time-varying, massive, and high dimensional data. In this context, Active Learning emerges as an attractive technique for the development of high performance models using few data. The importance of Active Learning for Big Data becomes more evident when labeling cost is high and data is presented to the learner via data streams. This paper presents a novel Active Learning method based on Extreme Learning Machines (ELMs) and Hebbian Learning. Linearization of input data by a large size ELM hidden layer turns our method little sensitive to parameter setting. Overfitting is inherently controlled via the Hebbian Learning crosstalk term. We also demonstrate that a simple convergence test can be used as an effective labeling criterion since it points out to the amount of labels necessary for learning. The proposed method has inherent properties that make it highly attractive to handle Big Data: incremental learning via data streams, elimination of redundant patterns, and learning from a reduced informative training set. Experimental results have shown that our method is competitive with some large-margin Active Learning strategies and also with a linear SVM.

1. Introduction

The induction of Supervised Learning models relies on a large enough set of $(\mathbf{x}_i, y_i)$ pairs obtained by sampling $\mathbf{x}_i$ from the input space χ according to a probability function $P(\mathbf{X})$ and by querying an oracle function $f_g(\mathbf{x}_i)$ for the labels y_i. The final goal of learning is to obtain the parameters $\mathbf{Z}$ and $\mathbf{w}$ of the approximation function $f(\mathbf{x}, \mathbf{Z}, \mathbf{w})$ so that $f(\mathbf{x}, \mathbf{Z}, \mathbf{w}) \approx f_g(\mathbf{x})$. Convergence conditions to guarantee that $f(\mathbf{x}, \mathbf{Z}, \mathbf{w}) \to f_g(\mathbf{x})$ in χ depend on the representativeness and size of the learning set $D_L = \{\mathbf{x}_i, y_i\}_{i=1}^{N_L}$. Reliable labeling of the input samples $D_u = \{\mathbf{x}_i\}_{i=1}^{N_u}$ is of paramount importance to guarantee robustness of the approximation function $f(\mathbf{x}, \mathbf{Z}, \mathbf{w})$. In any case, N_L should be large enough to guarantee convergence conditions.

In Big Data problems, however, the availability of a large amount of data reveals itself as another important challenge for the induction of supervised models [1]. Learning using the entire dataset can be impracticable for most current supervised classifiers due to their time-consuming training procedures. The problem becomes more evident when labeling cost is difficult or expensive and data is presented to the learner via data streams [2]. Dealing with Big Data requires some technique to circumvent the need of considering the entire data in the learning process. In this context, sampling probability $P(\mathbf{X})$ can be controlled in order to induce good learning models using fewer patterns.

In Supervised Learning it is assumed that the learner has no control of the sampling probability $P(\mathbf{X})$. Nonetheless, the construction of learning machines that may influence $P(\mathbf{X})$ has become a central problem in recent years. This new subfield of Machine Learning, known as *Active Learning* [2], has received special attention due to the new learning settings that have appeared in application areas such as bioinformatics [3], electronic commerce [4], and video classification [5].

In the new setting, the learner is *active* and may actually choose the samples from a stream [6] or pool [7] of data to be labeled. The sample selection strategy embodied into the active learner determines the probability $P(\mathbf{X})$ of an input sample to be selected for labeling and learning. In the end, the goal of Active Learning is similar to that of Supervised Learning: to induce a learning function $f(\mathbf{x}, \mathbf{Z}, \mathbf{w})$ that is valid in the whole input domain by behaving as similar as possible to the label-generator function $f_g(\mathbf{x})$. The goal is to select more representative samples that will result in $f(\mathbf{x}, \mathbf{Z}, \mathbf{w}) \rightarrow f_g(\mathbf{x})$. For instance, in a classification problem those samples that are near the separation margin between classes may suffice [8, 9] if discriminative models like Support Vector Machine (SVM) [10] and Perceptron-based neural networks [11] are used.

Margin-based Active Learning has been usually accomplished by considering the simplistic linear separability of patterns in the input space [12–14]. Once a linear separator is obtained from the initial samples, further labeling is accomplished according to a preestablished criterion, usually related to sample proximity to the separator, which is simpler to calculate if the separator is linear. In a more realistic and general approach, however, a nonlinear separator should be considered, which requires that linearization be carried out by mapping the input data into a feature space, where sample selection is actually accomplished. The overall function $f(\mathbf{x}, \mathbf{Z}, \mathbf{w})$ is composed of the hidden layer mapping function $\psi(\mathbf{x}, \mathbf{Z})$ and the output function $\phi(\psi(\mathbf{x}, \mathbf{Z}), \mathbf{w})$. Since both functions are single layer, $\phi(\cdot, \cdot)$ can only perform linear separation and $\psi(\mathbf{x}, \mathbf{Z})$ is expected to linearize the problem. Nevertheless, the difficulty with the nonlinear approach is that in order to obtain $\psi(\mathbf{x}, \mathbf{Z})$ some sort of user interaction may be required.

In order to overcome the difficulty to obtain a user-independent nonlinear feature space mapping, in this paper we present a method that is based on the principles of Extreme Learning Machines (ELMs) [15] to obtain the mapping function $\psi(\mathbf{x}, \mathbf{Z})$. The basic principle of ELM is to randomly sample the elements of $\mathbf{Z}$ and to expand the input space into a higher dimension in order to obtain $\psi(\mathbf{x}, \mathbf{Z})$. This is the most fundamental difference between ELM, feedforward neural networks, and SVM, since in these two models the function $\psi(\mathbf{x}, \mathbf{Z})$ is obtained by minimizing the output error. In practice, the only parameter required by the ELM projection is the dimension (number of neurons) of the feature space to which its final performance is not much sensitive.

Although both ELM and SVM are based on two-layer mapping, SVM's kernel provides an implicit mapping whereas ELM is based on the explicit mapping by the hidden layer sigmoidal functions [16, 17]. The two models also differ on the way that smoothing of the approximation function is treated, since SVM's performance relies on support vectors and ELM's output is computed considering the whole dataset. The Lagrangian solution of SVM's quadratic programming learning problem yields Lagrange multipliers that, in practice, point out to the patterns (the support vectors) that will be used to compute SVM's output. In fact, SVM's output $\widehat{y}_i$ for the input pattern $\mathbf{x}_i$ is a linear combination of the labels y_j weighted by the kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ between $\mathbf{x}_i$ and all other learning patterns $\mathbf{x}_j$: $\widehat{y}_i = \text{sign}(\sum_{i=1}^{N} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) + b)$. The linear combination coefficients are the Lagrange multipliers α_j resulting from the solution of the quadratic programming problem, which was formulated with the objective of minimizing the empirical risk and maximizing the separation margin [10]. Since only those patterns with nonzero Lagrange multipliers effectively contribute to the computation of $\widehat{y}_i$, SVM's learning can be seen as a sample selection problem. Given the proper kernel parameters, the selection of margin patterns $\mathbf{x}_j$ and the Lagrange multipliers α_j yields error minimization and margin maximization [10]. In such a scenario, "discarded" samples are those assigned to null Lagrange multipliers $\alpha_j = 0$ and the "selected" ones, the support vectors, are those with Lagrange multipliers in the range $0 \leq \alpha_j \leq C$, where C is a regularization parameter.

SVM's learning approach, however, can not be directly applied to the Active Learning problem since the whole dataset must be available at learning time so that the optimization problem can be solved. Active Learning methods should be capable of dealing with incremental and online learning [2, 14, 18], which is particularly convenient to Big Data problems. Nonetheless, the selection strategy presented in this paper aims at patterns near the class separation boundaries, which is expected to result in large-margin separators and in output function smoothing that may compensate for overparametrization [19] of the projection function $\psi(\mathbf{x}, \mathbf{Z})$.

The mapping $\mathbf{H}$ of $\mathbf{X}$ into the feature space is expected to embody a linearly separable problem given a large enough number of projection neurons p [20]. Once the mapping matrix $\mathbf{H}$ is obtained, the learning problem is reduced to selecting patterns and to inducing the parameters of the linear separator.

The pseudoinverse approach adopted by original formulation of ELM [15] to obtain the linear separator results in overfitting when the number of selected patterns tends to the number of neurons ($N \rightarrow p$). In such a situation, the number of equations is the same as the number of unknowns and the pseudoinverse yields a zero-error solution [15]. Consequently an overfitted model is obtained due to the large number of hidden neurons p required to separate the data with a random projection. Since in Active Learning the training set size will most likely reach the number of neurons as more patterns are labeled, the zero-error solution effect of the pseudoinverse is unwanted because it may result in a sudden decline in performance near the limit $N \approx p$. Because of that, an alternative to the pseudoinverse solution should be adopted in Active Learning problems.

Recently, Huang et al. [21] proposed a regularized version of ELM that can avoid the zero-error solution for $N \approx p$.

For this formulation a regularization parameter should be fine-tuned, which can increase the costs to perform Active Learning, because some labeled patterns should be separated to the parameter tuning. In addition, since Active Learning is incremental, relearning the whole dataset for every new pattern can be prohibitive. At first sight, the Online Sequential Extreme Learning Machine (OS-ELM) [22] could be a good candidate to Active Learning, because it can learn data one by one or chunk by chunk. However, its formulation demands that the initial model must be calculated using at least $N = p$ patterns. In this case p can be large, which implies that the initial learning set should also be large. So, this is not the best option, because the main objective of Active Learning is to minimize the number of labeled patterns necessary to learn [2]. Because of that, in this paper we propose a new incremental learning approach to replace the pseudoinverse-based solutions. The method has an inherent residual term that compensates the zero-error solution of the pseudoinverse and that can be viewed as implicit regularization.

The method presented in this paper is a classifier composed of a large size ELM hidden layer and an output layer learned via a Hebbian Learning Perceptron with normalized weights [23]. The Active Learning strategy relies on a convergence test adapted from the Convergence Theorem of Perceptron [11, 24]. The learning process is stream-based and each pattern is analyzed once. It is also incremental and online, which is particularly suitable for Big Data. Experimental results have shown that the proposed Active Learning strategy achieved a performance similar to linear SVM with ELM kernel and to regularized ELM. Our approach, however, has shown learning only a small part of the dataset.

The remainder of this paper is organized as follows: Section 2 describes the foundations of Extreme Learning Machines. Section 3 presents the Hebbian Learning. Section 4 discusses how overfitting can be controlled using Hebbian Learning. Section 5 extends the Perceptron Convergence Theorem [11, 24] to the Hebbian Learning with normalized weights [23]. Section 6 presents the principles of our Active Learning strategy. Experimental results are shown in Section 7. At last, the final discussions and conclusions are provided in Section 8.

2. Extreme Learning Machines

ELM can be seen as a learning approach to train a two-layer feedforward neural network, Multilayer Perceptron (MLP) type [15]. The method has basically the following main characteristics: (1) number of hidden neurons is large, (2) training of hidden and output layers is made separately, (3) hidden nodes parameters are not learned according to a general objective function but randomly chosen, and (4) output weights are not learned iteratively but obtained directly with the pseudoinverse method.

The input matrix $\mathbf{X}$ with N rows and n columns contains the input training data, where N is the number of samples and n is the input space dimension. The rows of the $N \times 1$ vector $\mathbf{y}$ contain the corresponding labels of each one of the N input samples of $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & \vdots \\ x_{N1} & \cdots & x_{Nn} \end{bmatrix}_{N \times n}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}. \tag{1}$$

Function $\psi(\mathbf{x}, \mathbf{Z}, \mathbf{b})$, with argument $\mathbf{x}$, matrix of weights $\mathbf{Z}$, and vector of bias $\mathbf{b}$, maps each one of the rows of $\mathbf{X}$ into the rows of the $N \times p$ mapping matrix $\mathbf{H}$, where p is the number of hidden layer neurons ($\mathbf{H} = \psi(\mathbf{X}, \mathbf{Z}, \mathbf{b})$). Activation functions of all neurons are regular sum-and-sigmoid functions:

$$\mathbf{H} = \begin{bmatrix} \psi\left(\mathbf{x}_1^T, \mathbf{Z}, \mathbf{b}\right) \\ \vdots \\ \psi\left(\mathbf{x}_N^T, \mathbf{Z}, \mathbf{b}\right) \end{bmatrix} = \begin{bmatrix} \text{sigmoid}\left(\mathbf{x}_1^T \cdot \mathbf{z}_1 + b_1\right) & \cdots & \text{sigmoid}\left(\mathbf{x}_1^T \cdot \mathbf{z}_p + b_p\right) \\ \vdots & \vdots & \vdots \\ \text{sigmoid}\left(\mathbf{x}_N^T \cdot \mathbf{z}_1 + b_1\right) & \cdots & \text{sigmoid}\left(\mathbf{x}_N^T \cdot \mathbf{z}_p + b_p\right) \end{bmatrix}_{N \times p}. \tag{2}$$

In the particular case of ELM since the elements of $\mathbf{Z}$ and $\mathbf{b}$ are randomly sampled the number of hidden neurons p is expected to be large enough to meet the linear separability conditions of Cover's theorem [20], so the projected data from $\mathbf{X}$ into $\mathbf{H}$ is assumed to be linearly separable.

Matrix $\mathbf{H}$ is then mapped into the output space by the function $\phi(\mathbf{H}, \mathbf{w})$ in order to approximate the label-vector $\mathbf{y}$. The $p \times 1$ vector $\mathbf{w}$ contains the parameters of the linear separator in the hidden layer and is obtained solving a linear system of N equations:

$$\mathbf{H}\mathbf{w} = \mathbf{y}. \tag{3}$$

The smallest norm least-squares solution of the above linear system is [15]

$$\mathbf{w} = \mathbf{H}^{\dagger}\mathbf{y}, \tag{4}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose pseudoinverse. The network response $\hat{\mathbf{y}}$ to an input pattern $\mathbf{x}$ is obtained by first calculating $\mathbf{H}$ and then by estimating the output as $\hat{\mathbf{y}} = \text{sign}(\mathbf{H}\mathbf{w})$ [21] for binary classification. For multiclass classification the output is estimated choosing the highest output neuron. In this paper we focus only in binary classification.

Like in any function approximation problem, the resulting general function $f(\mathbf{x}, \mathbf{Z}, \mathbf{b}, \mathbf{w})$ is expected to be robust to $\mathbf{x}_i \notin \mathbf{X}$. However, the pseudoinverse zero-error least-squares solution results in overfitting of the oversized ELM when N is close to p. Since the learning set is formed incrementally

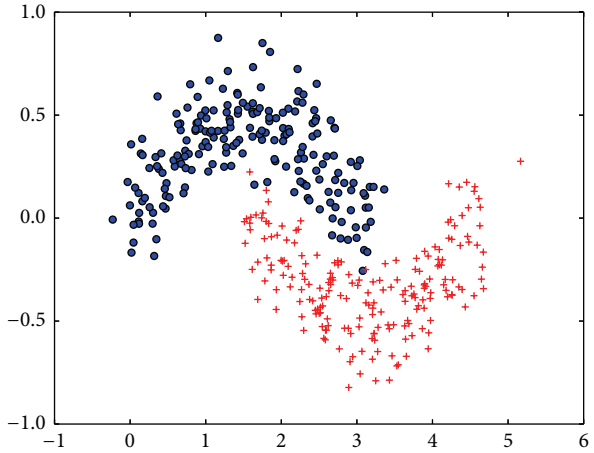

Figure 1: Binary classification dataset used in the random selection Active Learning example of Figure 2.

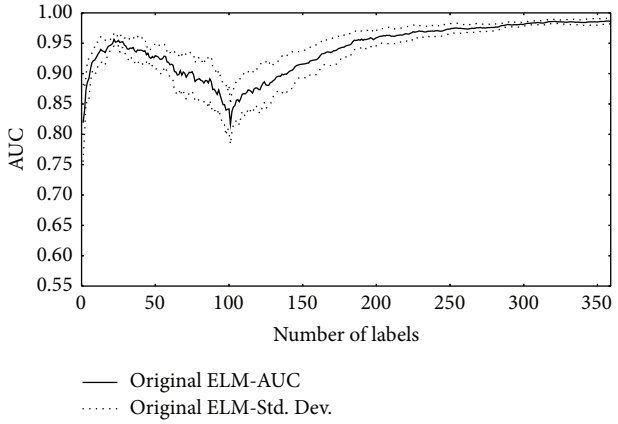

Figure 2: The use of the pseudoinverse causes sharp decline in performance when the number of selected patterns approaches the number of hidden neurons ($N \approx p$).

in Active Learning, N will eventually reach p as learning develops, which makes the use of original formulation of ELM in this context impracticable, even if the heuristics of Schohn and Cohn [9] and Tong and Koller [8] are applied.

In order to show performance degradation when using the pseudoinverse, a sample selection strategy using ELM with 100 hidden neurons was applied to the dataset of Figure 1, which is a nonlinear binary classification problem with 180 samples of each class. The experiment was performed with 10-fold cross-validation and 10 runs. The learning process started using only one randomly chosen pattern, which was then projected into the hidden layer with random bias and weights. Output weights were obtained with the pseudoinverse followed by the calculation of Area Under the ROC Curve (AUC) [25] performance on the test set. Active Learning continues with the random selection strategy and as new random patterns are added to the learning set the projection procedures and pseudoinverse calculation are repeated. Figure 2 shows the yielded average AUC on all experiments. As can be observed, AUC performance degrades sharply in the region $N \approx p$ when the number of equations reaches the number of unknowns and the pseudoinverse solution of the linear system is exact.

Active Learning could benefit from the linearization accomplished by the hidden layer projection of ELM, since it is practically a parameterless mapping. The number of hidden neurons p does not require fine-tuning and hidden nodes parameters are randomly sampled, so no optimization parameters need to be set and no interaction is required with user at learning time. The payoff is that the large value required for p and the pseudoinverse solution will result in the unwanted behavior of Figure 2.

The OS-ELM seems to be a good candidate to perform Active Learning, because it can learn data one by one or chunk by chunk. However, its formulation demands that in initialization phase the number of training patterns should be at least equal to the number of hidden nodes ($N = p$) [22]. As discussed before, the learning set is formed incrementally in Active Learning and N could be less than p at the beginning, so OS-ELM is not the best option to perform Active Learning.

The new regularized formulations of ELM proposed by Huang et al. [21] can generalize well even if $N \leq p$. The classification problem for the new regularized formulations of ELM with a single-output node can be formulated as [21]

$$\begin{aligned} &\text{Minimize:} \quad L_{\text{Primal}} = \frac{1}{2}\|\mathbf{w}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 \\ &\text{Subject to:} \quad \psi\left(\mathbf{x}_i^T, \mathbf{Z}, \mathbf{b}\right)\cdot\mathbf{w} = y_i - \xi_i, \\ &\qquad\qquad\qquad\qquad i = 1, \ldots, N, \end{aligned} \tag{5}$$

where ξ_i is the training error with respect to the training pattern $\mathbf{x}_i$ and C is a regularization parameter.

In this formulation, the ELM training is equivalent to solving the following dual optimization problem [21]:

$$\begin{aligned} L_{\text{Dual}} = {} & \frac{1}{2}\|\mathbf{w}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 \\ & - \sum_{i=1}^{N}\alpha_i\left(\psi\left(\mathbf{x}_i^T, \mathbf{Z}, \mathbf{b}\right)\cdot\mathbf{w} - y_i + \xi_i\right), \end{aligned} \tag{6}$$

where each Lagrange multiplier α_i corresponds to the ith training pattern [21].

Huang et al. [21] proposed two solutions for this optimization problem:

(1) For the case where the learning set has small or medium size:

$$\mathbf{w} = \mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{y}. \tag{7}$$

(2) For the case where the learning set has large size:

$$\mathbf{w} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{y}. \tag{8}$$

For binary classification the output for a input pattern $\mathbf{x}_i$ is estimated as $\hat{\mathbf{y}} = \text{sign}(\psi(\mathbf{x}_i, \mathbf{Z}, \mathbf{b})\mathbf{w})$ [21]. For this ELM formulation a regularization parameter should be fine-tuned. In this case, some patterns should be separated and labeled for that purpose, which can increase the Active Learning costs. The main objective of Active Learning is to induce a Supervised Learning model using the fewest number of labeled patterns, so in this case the parameter tuning can be prohibitive.

With the purpose of showing the characteristics of these ELM alternatives (all ELM implementations are accomplished using the source codes available in the website http://www.ntu.edu.sg/home/egbhuang/elm_codes.html) the same problem of Figure 2 was applied to OS-ELM and the new regularized formulation of ELM that we called ELM2012. For ELM2012 thirty percent of the training set was used for fine-tuning the regularization parameter C. For OS-ELM it was used in the initialization phase of a number of labeled patterns equal to the number of hidden nodes (100). The subsequent patterns are learned one by one in a randomized way. The results are presented in Figure 3. As can be seen, the OS-ELM results are affected by the overfitted pseudoinverse solution of the initialization phase ($N = p$). This problem is propagated during the one by one learning. The ELM2012 solves the pseudoinverse limitations when $N \approx p$, but a regularization parameter should be fine-tuned, which can be prohibitive in the Active Learning scenario [18].

In this paper we present a Hebbian Learning [23, 26] approach to compensate the unwanted behavior of Figure 2 and to avoid fine-tuning regularization parameters as will be discussed in the next sections.

3. Hebbian Learning

In neural networks learning, the update of weight w_{ij} that connects neurons i and j according to Hebb's rule [26], is proportional to the cross product of their activation values for any input-output association k or in other words $w_{ij} \propto x_{ik} y_{jk}$. The rule can be written in matrix form as in

$$\mathbf{w} = \mathbf{X}^T \mathbf{y}, \tag{9}$$

where $\mathbf{w}$ is the $n \times 1$ weight vector, $\mathbf{X}$ is the $N \times n$ input data matrix, and $\mathbf{y}$ is the $N \times 1$ vector containing the output values.

Since the estimation $\hat{\mathbf{y}}$ of vector $\mathbf{y}$ is given by $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, the substitution of $\mathbf{w}$ from (9) into this expression leads to $\hat{\mathbf{y}} = \mathbf{X}\mathbf{X}^T\mathbf{y}$. Therefore, in order to perfectly retrieve $\mathbf{y}$ we must have $\mathbf{X}\mathbf{X}^T = \mathbf{I}$. If such a condition is met the training error is zero, since $\hat{\mathbf{y}} = \mathbf{y}$, and the learning rule is equivalent to the pseudoinverse method of (4). This condition can only happen if $\mathbf{x}_i \perp \mathbf{x}_j$ and $\mathbf{x}_i^T \mathbf{x}_i = 1 \; \forall ij$. In such a situation $\mathbf{X}$ forms an orthonormal basis and $\mathbf{X}^T = \mathbf{X}^{-1}$ [27] which leads (9) and (4) to be equivalent. However, in most real situations $\mathbf{X}$ will not be an orthonormal basis and a residual term because the product $\mathbf{X}\mathbf{X}^T$ will cause a shift of $\hat{\mathbf{y}}$ in relation to $\mathbf{y}$. Since

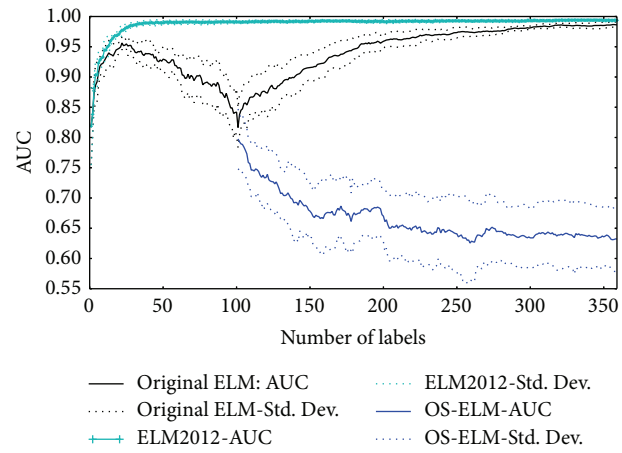

Figure 3: Comparison between the original formulation of ELM [15], OS-ELM [22], and the new regularized formulation of ELM [21] here called ELM2012. Thirty percent of the learning set was used to fine-tune the regularization parameter of ELM2012.

an individual $\hat{y}_k$ is obtained as $\hat{y}_i = \sum_{j=1}^{N} \mathbf{x}_k^T \mathbf{x}_j y_j$, the term corresponding to $j = k$ can be separated from the summation which leads to the following:

$$\hat{y}_k = \mathbf{x}_k^T \mathbf{x}_k y_k + \sum_{j=1, j\neq k}^{N} \mathbf{x}_k^T \mathbf{x}_j y_j. \tag{10}$$

In the particular situation when the input data is normalized, that is, $\mathbf{x}_k^T \mathbf{x}_k = 1$, (10) can be simplified into

$$\hat{y}_k = y_k + \overbrace{\sum_{j=1, j\neq k}^{N} \mathbf{x}_k^T \mathbf{x}_j y_j}^{\text{crosstalk}}. \tag{11}$$

The interpretation of (11) [28] is that the estimated response $\hat{y}$ for the input pattern $\mathbf{x}_k$ is the exact response y_k plus the residual crosstalk term.

The crosstalk of (11) is inherent to Hebbian Learning that is due to the nonorthogonality of the input data, so it depends on how the input samples are spatially related to each other. For usual inductive learning approaches data is provided to learning without any interference on the sampling probability of a given $\mathbf{x}_k$. Therefore, crosstalk is innate to a given dataset and can be calculated directly given the samples. From (11) it can be noted that it represents a shift in relation to the zero-error pseudoinverse solution, which is in fact unwanted in ELM learning since it yields overfitting, as discussed in Section 2. The degree on which the Hebbian solution differs from the zero-error solution is in fact determined by the sample selection strategy in Active Learning. This has a penalization effect in relation to the pseudoinverse outcome that is expected to result in a smoother learning curve than the one presented in Figure 2.

We argue in this paper that the use of Hebbian Learning as represented in (9) in replacement to (4) yields better

generalization due to the residual term, which alleviates the overfitting effects of the pseudoinverse, as will be discussed in the next sections. In addition, Hebbian Learning is particularly suitable to Active Learning, since the selection of a new pattern does not require that previously selected ones be learned again. The contribution of a new pattern can be simply added to the current weight vector, as shown in (12). Unlearning, which is appropriate to remove redundant patterns in Big Data problems, can also be accomplished by removing the data from the summation

$$\mathbf{w}_{\text{hebb}} = \overbrace{\mathbf{x}_0 y_0 + \mathbf{x}_1 y_1 + \cdots + \mathbf{x}_t y_t}^{\text{Current weight}} + \overbrace{\mathbf{x}_{t+1} y_{t+1}}^{\text{New pattern}}. \quad (12)$$

It is clear that the crosstalk may increase as new patterns are selected to be learned and are added to the summation of (12). The estimation of the crosstalk term according to varied sampling scenarios was the subject of many papers that aimed at estimating the storage capacity of Hopfield Networks [29] in the late 1980s [30–33]. The limit number of associations that can be stored is clearly dependent on the crosstalk. In Active Learning, however, the magnitude of the crosstalk is a result of the patterns selected for learning, so the selection strategy has an indirect control of how much $\widehat{y}_k$ deviates from y_k. In the next sections a selection strategy as well as a method to estimate the maximum number of patterns that can be stored in ELMs trained with Hebbian Learning in the Active Learning scenario is presented. It is also shown in the next section that crosstalk has a regularization effect in Hebbian Learning.

4. Avoiding Overfitting with Hebbian Learning

As discussed in the previous sections, the very nature of ELM's design results in an oversized network that is likely to yield overfitting with pseudoinverse learning. This happens because the pseudoinverse solution is optimal and results in null error when $N = p$, so in order to avoid the associated overfitting effect of single-objective error minimization learning, many methods aim at displacing the network solution from the null error region. Regularization methods [34], for instance, include a regularization term in the objective function, which is usually represented as a linear combination of the square error and an additional penalization function. The objective function is usually described in the form $J(\mathbf{w}) = \sum e(\mathbf{w})^2 + \lambda\|\mathbf{w}\|$ with the norm of the weights $\|\mathbf{w}\|$ often used as penalization [35]. The effect of the regularized objective function is to shift the solution from the null training set error for $\lambda \neq 0$. In other words, the training set error should increase in order to avoid the overfitting effect of pseudoinverse learning combined with an intrinsically oversized network.

The crosstalk term of (11) implicitly contributes with a penalization term to the zero-error solution, which can be made clearer by the expression that follows. After some algebraic manipulations, which are presented in Appendix A, the square error $\epsilon_k^2 = (y_k - \widehat{y}_k)^2$ due to an arbitrary association

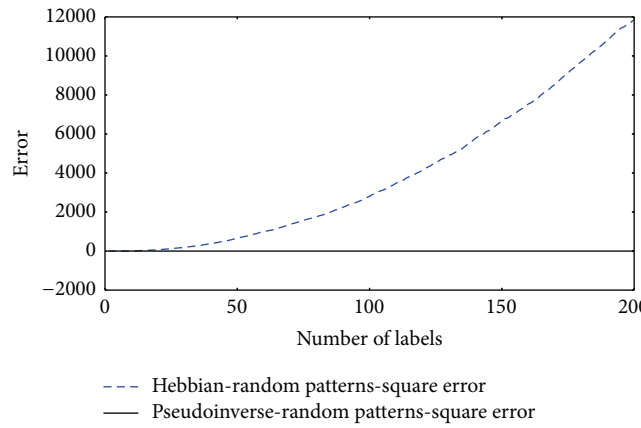

Figure 4: Training set error for ELMs trained with (4) and (9), pseudoinverse, and Hebbian Learning.

k trained with Hebbian Learning, as described in Section 3, can be represented by

$$\epsilon_k^2 = \overbrace{\left[2y_k - \widehat{y}_k\right]^2}^{\text{Error}} + \overbrace{y_k\left(2\mathbf{x}_k^T\mathbf{x}_k \sum_{i=1, i\neq k}^{N} y_i \mathbf{x}_i^T \mathbf{x}_k - y_k\right)}^{\text{Penalization}}. \quad (13)$$

The penalization term of (13) was obtained from the original error $(y_k - \widehat{y}_k)^2$ with the objective of showing the residual effect of Hebbian Learning. The first term is proportional to the square error, whereas the second one has a penalization effect due to the spatial relation of patterns within the learning set, since it is related to the crosstalk of (11). As more patterns are selected for learning the interference among them is likely to increase and so is the magnitude of the penalization term of (13). The sample selection strategy in Active Learning has, therefore, a direct impact on penalization and on overfitting smoothing.

The graph of Figure 4 shows ELM's training set error due to Hebbian and pseudoinverse learning, as the number of (random) learning patterns increases. Training was accomplished with (4) and (9). As expected, Hebbian Learning error increases quadratically, due to the first term of (13), whereas pseudoinverse error is null regardless of the number of selected learning patterns. This kind of behavior changes drastically when the error is calculated for the test set as can be seen in Figure 5. It is important to emphasize at this point that Figures 4 and 5 are in different y-axis scales, because of the discrepancy of magnitude of errors; however, the scales do not affect the current qualitative analysis. It can be observed that the pseudoinverse error increases drastically near $N = p$, a behavior that is compatible with the one presented also for the AUC in Figure 2. Although Hebbian test error has still a quadratic behavior, which is very smooth in the figure because of the scale, pseudoinverse error for the test set is much higher, specially near $N = p$ due to overfitting.

This kind of behavior indicates that crosstalk, which is often considered as a limitation of Hebbian Learning [30–33], may have a positive effect on ELM learning. Its contribution to output smoothing will, however, depend on the learning

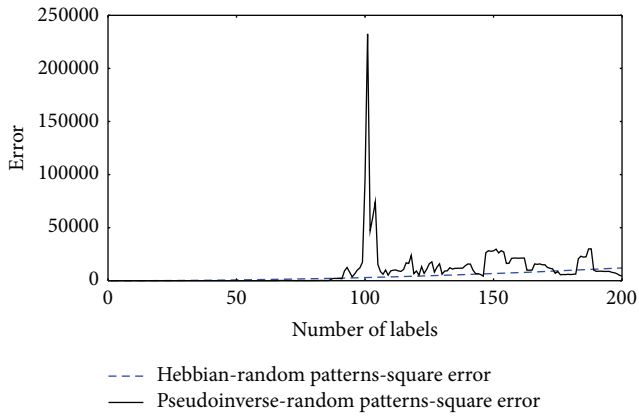

Figure 5: Test set error for ELMs trained with (4) and (9), pseudoinverse, and Hebbian Learning. Pseudoinverse error increases abruptly near $N = p$.

method's ability to control its magnitude, which is one of the goals of the proposed method that will be detailed in the next sections.

5. Limit Number of Training Patterns

The resulting model from (9) is in fact a Simple Perceptron [36] trained with Hebbian Learning. A variation with normalized weights, as presented in (14), has been shown to yield margin maximization [23]. In our context the question that remains, however, is to estimate the minimum number of patterns that are needed for learning. This is particularly important to smooth the effect of crosstalk, to control the labeling cost in Active Learning, and to reduce training set size of Big Data problems:

$$\mathbf{w} = \frac{\sum_{i=1}^{N} \mathbf{x}_i y_i}{\left\| \sum_{i=1}^{N} \mathbf{x}_i y_i \right\|}. \tag{14}$$

Perceptron Convergence Theorem states that Rosenblatt's algorithm [36] converges with a limit number of iterations that is less than or equal to the maximum number of misclassifications if the problem is linearly separable [11]. In our context, for ELM learning, we assume that the resulting problem in ELM's hidden layer is linearly separable due to ELM's hidden layer nonlinear projection. Based on this assumption, we show next that Nilsson's proof of convergence [11, 24] can be extended to Hebbian Learning. The proof of the theorem that follows, which is presented in Appendix B, assures the estimation of the limit number of patterns that is sufficient to find a linear separator and, consequently, yields the upper limit that ensures convergence. When considering such a reduced training set, learning can be interrupted to avoid increase in the penalization term of (13) due to crosstalk. As will be shown in the next sections, the limit number of patterns given by the following theorem does in fact minimize the crosstalk.

Theorem 1. *For two linearly separable classes the maximum number of labels needed for convergence of a Hebbian Perceptron is given by*

$$t_{\max} = \frac{\beta + 2\theta}{\alpha^2}, \tag{15}$$

where β is the maximum square norm of the training patterns, θ is the margin of the most distant pattern from the separating hyperplane, and α is the margin of the closest pattern to the separating hyperplane.

The deductions of α, β and θ are presented in Appendix B. Considering the learning set ζ, α, β and θ can be defined as presented in the following equations:

$$\begin{aligned} \alpha &= \min_{\mathbf{x} \in \zeta} \left| \mathbf{w}^T \mathbf{x} \right|, \\ \beta &= \max_{\mathbf{x} \in \zeta} \|\mathbf{x}\|^2, \\ \theta &= \max_{\mathbf{x} \in \zeta} \left| \mathbf{w}^T \mathbf{x} \right|. \end{aligned} \tag{16}$$

Equation (15) indicates that Hebbian Learning Perceptron with normalized weights [23] converges using at most $(\beta + 2\theta)/\alpha^2$ labels. As will be shown in the next section, the value of $t_{\max}$ can be estimated during learning and used as part of the sample selection criterion and also to determine the number of patterns to be learned. This is specially interesting to Big Data problems because only the most informative patterns need to be selected and learned.

6. Convergence Test as Active Learning Strategy

The theorem presented in the previous section can be used as a labeling criterion for Active Learning since it points out to the amount of labels necessary to ensure convergence of the Hebbian Learning. The decision on labeling is based on successive estimates of $t_{\max}$ along the learning process. Since the convergence should occur with a reduced number of patterns, the influence of crosstalk term can be controlled and smoothing can be achieved.

Our classifier is composed of a single ELM hidden layer and an output layer learned via Hebbian Learning Perceptron with normalized weights [23]. Fernández-Delgado et al. [23] demonstrated that this Perceptron works as an SVM in which all training patterns are considered as support vectors with Lagrange multipliers equal to 1.

In order to select only the most informative patterns our Active Learning strategy uses a convergence test as a labeling criterion at each time a new pattern is presented. Each new pattern is propagated through ELM hidden layer and then its margin is calculated in relation to the current hyperplane. This value is assigned to α in (15). The variable θ corresponds to the maximum between the calculated α and the previous αs. The variable β is the maximum between the norm of the current vector and the largest norm of the previous training vectors. Using these variables, $t_{\max}$ is calculated using (15).

Input: Initial size of the training set m, maximum number of labels L, ELM random weights $\mathbf{Z}$ and random bias $\mathbf{b}$, generator function F
Output: Weights vector $\mathbf{w}$
Method:
Take at random m pattens $\mathbf{x}_k$ from the generator function F;
Propagate the m pattens through the ELM layer ($\psi(\mathbf{x}_k^T, \mathbf{Z}, \mathbf{b})$) and query its labels y_k;

$$\mathbf{w} = \frac{\sum_{k=1}^{m} \psi\left(\mathbf{x}_k^T, \mathbf{Z}, \mathbf{b}\right) y_k}{\left\| \sum_{k=1}^{m} \psi\left(\mathbf{x}_k^T, \mathbf{Z}, \mathbf{b}\right) y_k \right\|};$$

$$\beta = \max_{k=1,\ldots,m} \left(\left\| \psi\left(\mathbf{x}_k^T, \mathbf{Z}, \mathbf{b}\right) \right\|^2 \right);$$

$$\theta = \max_{k=1,\ldots,m} \left(\left| \mathbf{w}^T \psi\left(\mathbf{x}_k^T, \mathbf{Z}, \mathbf{b}\right) \right| \right);$$

Repeat
  Take at random a pattern $\mathbf{x}$ from F;

$$\alpha = \left| \mathbf{w}^T \psi\left(\mathbf{x}^T, \mathbf{Z}, \mathbf{b}\right) \right|;$$

$$\beta = \max\left(\left\| \psi\left(\mathbf{x}^T, \mathbf{Z}, \mathbf{b}\right) \right\|^2, \beta \right);$$

$$\theta = \max\left(\left| \mathbf{w}^T \psi\left(\mathbf{x}^T, \mathbf{Z}, \mathbf{b}\right) \right|, \theta \right);$$

$$t = \frac{\beta + 2\theta}{\alpha^2};$$

  if $t > m$ **then**
    Query the label y;
    Update $\mathbf{w} = (\mathbf{w} + \psi(\mathbf{x}^T, \mathbf{Z}, \mathbf{b})y)/\|\mathbf{w} + \psi(\mathbf{x}^T, \mathbf{Z}, \mathbf{b})y\|$;
    $m = m + 1$;
  end
Until $(m = L)$ or $(F = \varnothing)$;

Algorithm 1: Active Learning strategy.

The convergence test is accomplished using $t_{\max}$. If $t_{\max}$ is larger than the number of used training patterns, then the algorithm did not converge from the current pattern, so that this pattern must be learned. Its label is queried to the specialist and the Perceptron weights are adjusted using the following equation:

$$\mathbf{w}(t+1) = \frac{\mathbf{w}(t) + \psi(\mathbf{x}, \mathbf{Z}, \mathbf{b})\, y}{\left\| \mathbf{w}(t) + \psi(\mathbf{x}, \mathbf{Z}, \mathbf{b})\, y \right\|}. \tag{17}$$

If $t_{\max}$ is lower than or equal to the number of used training patterns then the algorithm converges and therefore it is not necessary to query the current label since it is probably redundant. The process continues until new patterns are presented or until the maximum number of labels is reached. Algorithm 1 presents the pseudocode of our approach.

The proposed Active Learning strategy is able to find a solution that maximizes AUC and controls the crosstalk term. In order to illustrate these characteristics, three sample selection strategies using ELM with 100 hidden neurons were applied to the dataset of Figure 1. The first strategy uses the Hebbian Learning with normalized weights to learn, iteratively, the patterns closest to the separating hyperplane as proposed in the heuristic of Schohn and Cohn [9] (shown in Figure 6 as ELMPCP). The second strategy uses the Hebbian Learning with normalized weights to learn patterns selected at random (shown in Figure 6 as ELMPRP). The last strategy is our Active Learning method based on convergence test, named here as Extreme Active Learning Machine (EALM). We also included the ELMs results of Figure 3. The experiment was performed with 10-fold cross-validation and 10 runs. The results are presented in Figure 6 in terms of average AUC.

Considering a reduced training set, the heuristic of Schohn and Cohn [9] achieved better results than random selection. Nevertheless, Schohn's heuristic is considerably more time-consuming since it requires at each iteration the hyperplane distance estimates for all patterns. Our Active Learning strategy, in turn, achieves a good AUC performance calculating the margin once for each pattern. It is worth noting that although our strategy is stream-based [6], its performance was similar to the Schohn's pool-based strategy [9]. The regularized ELM achieved better results than the other strategies, but for this method a regularization parameter was fine-tuned using thirty percent of the learning set, which can be prohibitive in a real Active Learning scenario.

In order to illustrate the effect of crosstalk control, average square errors were calculated for each training set via (13). Each term of this equation is presented as a separated curve in Figure 7. As can be seen the error term related to the crosstalk is dominant for all strategies. In contrast with random pattern selection strategy, Schohn's heuristic was able to reduce the total error (square error) for reduced training sets. Our Active Learning strategy found a solution with nonzero total error and reduced crosstalk term, as expected. The next section presents experimental results for real problems, including Big Data, and compares our strategy with others reported in literature.

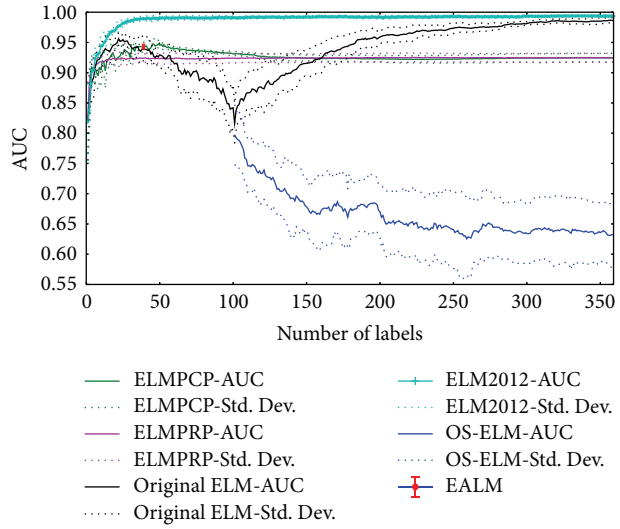

Figure 6: 10-fold cross-validation results for different sample selection strategies applied to the dataset of Figure 1.

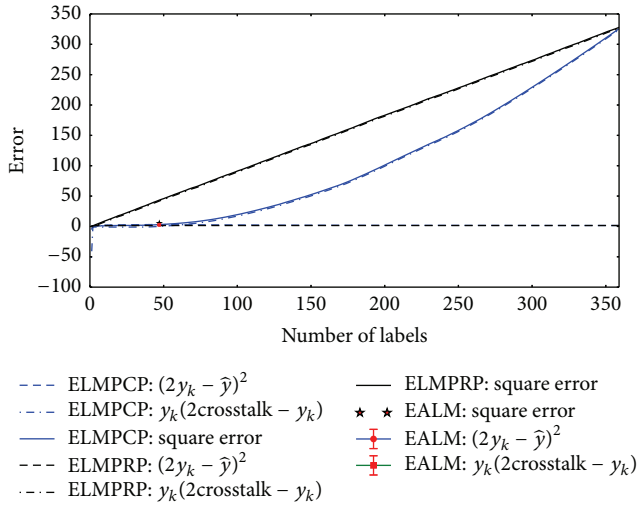

Figure 7: Training set error calculated using (13).

7. Experimental Results

In this section we report the results of four experiments. Experiment 1 demonstrates the limitation of original formulation of ELM when the number of training patterns is closer to the hidden layer size. It demonstrates the effect of using Hebbian Learning Perceptron with normalized weights [23] and it also compares all strategies with the regularized formulation of ELM. Experiment 2 shows that the number of hidden neurons does not need to be fine-tuned if it is much larger than the input size. Experiment 3 compares our Active Learning strategy with known Active Learning methods in literature, with a linear SVM (all linear SVM implementations are accomplished using LibLinear [38] available in the Shogun package [39]) and also with the regularized formulation of ELM. All these 3 experiments were performed over datasets of the UCI repository [40]. Such datasets are listed in Table 1 along their characteristics.

Table 1: Datasets for Experiments 1, 2, and 3.

Name	Key	Number of patterns	Number of inputs
Heart disease	HRT	297	13
Wisc. breast cancer original	WBCO	699	9
Wisc. breast cancer diagnostic	WBCD	569	31
Pima diabetes	PIMA	768	8
Sonar	SNR	208	60
Ionosphere	ION	351	34
Australian credit	AUST	690	14
Liver disorder	LIV	345	6
German credit	GER	1000	24
Spam	SPAM	4601	58

Table 2: Datasets for Experiment 4.

Name	Training set	Test set	Number of inputs
Splice	1000	2175	60
IJCNN1	49990	91701	22
Web Gaussian	49749	14951	300
Adult	22696	9865	123

At last, Experiment 4 shows the effectiveness of our Active Learning strategy on Big Data problems. The datasets were extracted from [41] and have their characteristics listed in Table 2. As can be observed, they present a large amount of patterns, which may hinder learning of traditional supervised models. Our results demonstrate, however, that it is possible to obtain effective models for Big Data with a reduced number of informative patterns.

7.1. Results with UCI Datasets. The following models were compared in Experiment 1: original formulation of ELM (original ELM); regularized version of ELM (ELM2012); ELM hidden layer with Hebbian Learning Perceptron and normalized weights [23] (ELMP); and our Active Learning strategy (Extreme Active Learning Machine—EALM). Data was presented at random for ELM and EALM models. In the particular case of ELMP, two tests were performed: (i) learning from data presented at random (ELMPRP); (ii) learning from patterns which are closest to the separating hyperplane as proposed in Schohn and Cohn [9] (ELMPCP). For all models, ELM hidden layer was set with the same random parameters (weights and bias) and 100 hidden neurons. This number was chosen to demonstrate ELM's limitation when the number of training patterns is close to the hidden layer size. For ELM2012 thirty percent of the learning set was used to fine-tune the regularization parameter.

The average performance of a particular algorithm was calculated over 10 runs of 10-fold cross-validation with AUC. The results of Experiment 1 are presented in Figures 8 and 9. As expected, the original formulation of ELM did not present good generalization when training set size is near the hidden

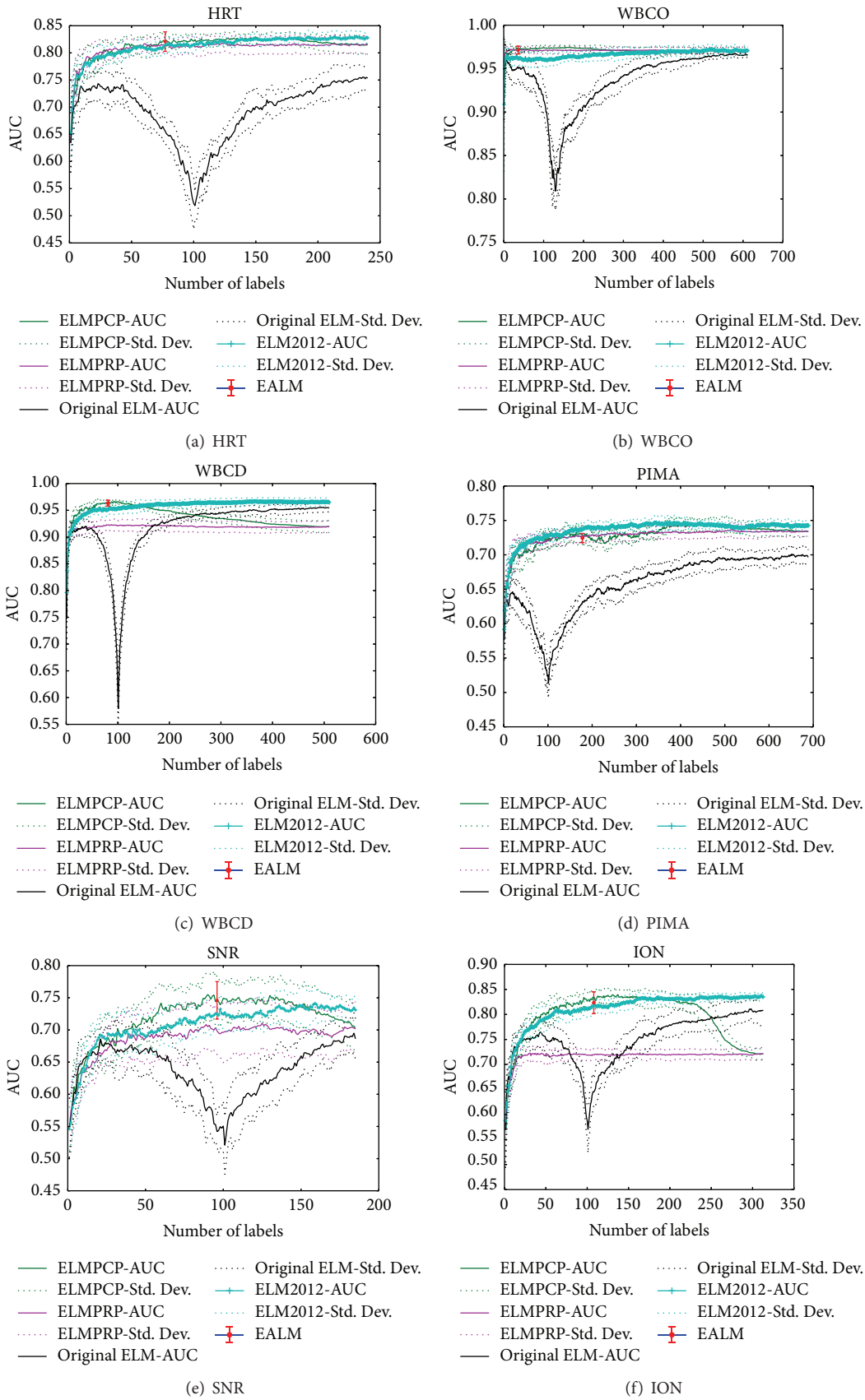

Figure 8: Average results of 10 runs of 10-fold cross-validation for different ELM methods in the datasets: HRT, WBCO, WBCD, PIMA, SNR, and ION.

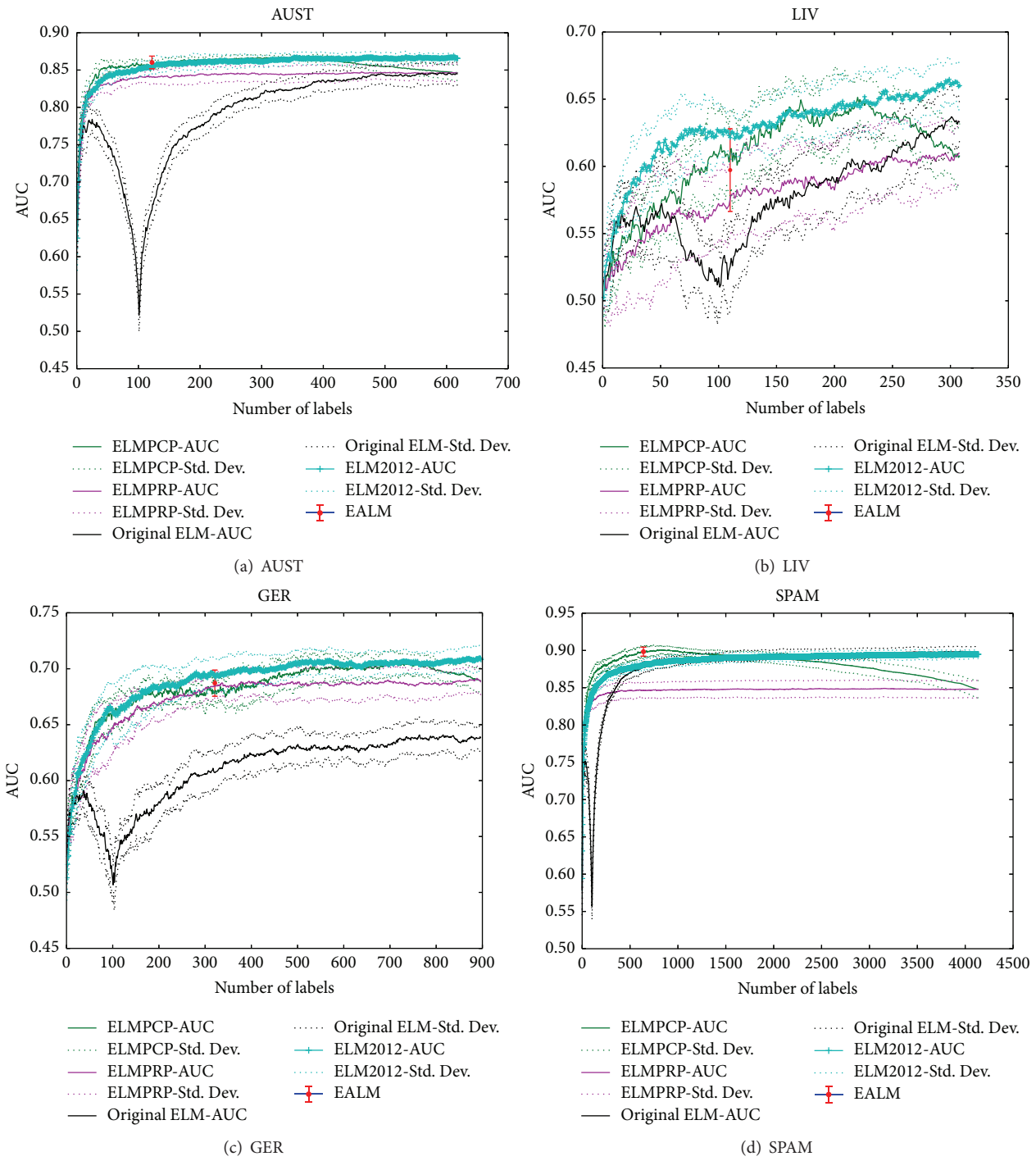

Figure 9: Average results of 10 runs of 10-fold cross-validation for different ELM methods in the datasets: AUST, LIV, GER, and SPAM.

layer size. This is because ELM's output weights are calculated solving a system of linear equations [15]. One can also observe that our Active Learning solution achieved a performance similar to ELMP trained with the closest patterns (ELMPCP). This shows that our labeling criterion based on the converge test indeed works as a "margin-based filter" but without the need to calculate the distances for all patterns. The EALM solutions are also close to the best solutions obtained using ELM2012, which confirms that our model uses the crosstalk term as implicit regularization.

Experiment 2 compares EALMs with different hidden layer sizes: 100, 500, 1000, 2500, 5000, and 10000. Results are listed in Table 3 in terms of the average number of labels selected by EALM (AL Labels) and the average AUC (AUC). The best values are marked in bold. From Table 3, one can notice that when the number of hidden neurons

Table 3: Comparison of the hidden layer size.

Key	EALM					
	100 neurons		500 neurons		1000 neurons	
	AL Labels	AUC	AL Labels	AUC	AL Labels	AUC
HRT	76.92 ± 3.36	0.82 ± 0.02	80.09 ± 1.58	0.83 ± 0.01	78.17 ± 1.09	0.83 ± 0.01
WBCO	36.47 ± 2.82	0.97 ± 0.01	36.98 ± 0.93	0.97 ± 0.00	37.16 ± 0.70	0.97 ± 0.00
WBCD	81.28 ± 5.36	0.96 ± 0.01	76.71 ± 1.01	0.97 ± 0.00	**73.65 ± 0.97**	**0.97 ± 0.00**
PIMA	177.63 ± 8.64	0.72 ± 0.01	172.96 ± 3.42	0.71 ± 0.01	168.69 ± 2.38	0.72 ± 0.01
SNR	95.97 ± 4.19	0.75 ± 0.03	100.00 ± 1.34	0.78 ± 0.01	98.42 ± 1.74	0.75 ± 0.01
ION	107.83 ± 4.44	0.82 ± 0.02	102.05 ± 1.04	0.85 ± 0.01	103.59 ± 1.37	0.84 ± 0.01
AUST	122.04 ± 5.80	0.86 ± 0.01	**119.59 ± 2.30**	**0.86 ± 0.01**	124.88 ± 1.88	0.86 ± 0.01
LIV	**109.73 ± 6.23**	**0.60 ± 0.03**	107.20 ± 4.14	0.58 ± 0.03	109.84 ± 7.10	0.59 ± 0.04
GER	291.21 ± 4.64	0.63 ± 0.01	302.07 ± 3.09	0.67 ± 0.01	315.55 ± 3.11	0.68 ± 0.01
SPAM	675.86 ± 4.33	0.89 ± 0.00	597.17 ± 4.89	0.92 ± 0.00	557.18 ± 4.68	0.92 ± 0.00
	2500 neurons		5000 neurons		10000 neurons	
HRT	76.94 ± 1.96	0.83 ± 0.01	76.53 ± 1.75	0.83 ± 0.01	**76.34 ± 1.11**	**0.84 ± 0.01**
WBCO	36.90 ± 0.50	0.97 ± 0.00	36.70 ± 1.07	0.97 ± 0.00	**36.12 ± 0.87**	**0.97 ± 0.00**
WBCD	75.44 ± 1.14	0.97 ± 0.00	75.35 ± 1.15	0.97 ± 0.00	75.76 ± 0.95	0.97 ± 0.01
PIMA	**168.63 ± 2.73**	**0.72 ± 0.01**	171.63 ± 3.59	0.73 ± 0.01	172.43 ± 2.83	0.71 ± 0.01
SNR	**97.54 ± 1.53**	**0.78 ± 0.02**	98.08 ± 1.49	0.78 ± 0.02	97.75 ± 2.40	0.77 ± 0.02
ION	**101.56 ± 1.65**	**0.85 ± 0.01**	101.72 ± 1.67	0.85 ± 0.01	102.97 ± 0.78	0.85 ± 0.01
AUST	119.91 ± 4.19	0.86 ± 0.01	120.75 ± 3.46	0.86 ± 0.00	121.18 ± 3.17	0.86 ± 0.01
LIV	109.22 ± 6.94	0.59 ± 0.03	106.53 ± 5.12	0.58 ± 0.03	105.84 ± 5.22	0.58 ± 0.02
GER	**313.04 ± 3.09**	**0.68 ± 0.01**	313.45 ± 3.11	0.68 ± 0.02	314.67 ± 2.90	0.68 ± 0.01
SPAM	559.81 ± 3.67	0.92 ± 0.00	561.52 ± 6.31	0.92 ± 0.00	**549.60 ± 2.95**	**0.92 ± 0.00**

reaches values larger than 500, the performance of all models becomes similar. This demonstrates that EALM is little sensitive to the hidden layer size and therefore it is not necessary to perform any tuning procedure for this parameter. The use of a larger value such as 1000 is enough, as suggested in [17, 42].

Experiment 3 [37] compares EALM with the following methods: Perceptron of Dasgupta et al. (*PDKCM*) [13], Perceptron of Cesa-Bianchi et al. (*PCBGZ*) [12]; SVM of Tong and Koller (*SVMTK*) [8]; a linear SVM trained with all patterns (*SVMALL*) and the regularized formulation of ELM trained with all patterns (*ELM2012)*. All these methods were used as linear outputs for an ELM hidden layer projection with 1000 neurons and weights and bias randomly selected in the range $[-3, 3]$, as proposed by [17]. Thirty percent of each dataset was separated in order to set the free parameters of *PDKCM*, *PCBGZ*, *ELM2012*, and *SVMs*. This was accomplished via a grid search with 10-fold cross-validation procedure, as suggested in Monteleoni and Kääriäinen [14]. *PCBGZ* was also tested with the optimal parameter $b = (\max_{x \in C} \|x\|^2)/2$ [12] and this variation was named as *PCBGZ-OPT*. Regularization parameter C of SVM was configured according to the range $\{2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 1, 2, 2^2, \ldots, 2^{14}\}$, as suggested in [23]. Regularization parameter C of ELM2012 was configured according to the range $\{2^{-24}, 2^{-23}, \ldots, 2^{24}, 2^{25}\}$, as suggested in [21].

Ten runs of 10-fold cross-validation were performed in order to calculate the average accuracy (Ac), the average AUC (AUC), and the average number of selected labels (AL Labels). All datasets were normalized to mean 0 and standard deviation 1. *EALM*, *PDKCM*, *PCBGZ*, and *PCBGZ-OPT* were initialized with one pattern selected at random. *SVMTK* was initialized using a set composed of two patterns, one of each class, and randomly chosen, as proposed by [8].

Results of Experiment 3 are shown in Table 4. For a particular model, the effective amount of labels consists of those used to configure its free parameters added to the labels selected by Active Learning (AL Labels). Table 4 shows the number of labels achieved by Active Learning (AL Labels) and the number of labels effectively used (Effective Labels). As can be observed, the best results were obtained for *EALM*, *SVMTK*, *SVMALL*, and *ELM2012*. Although *SVMTK* had selected a smaller number of labels during Active Learning, its computational cost is higher than EALM. Furthermore, it was verified that the effective number of labels used by *EALM* is smaller than those used by other models. The results of EALM are close to those of ELM2012 which indicates that the crosstalk term of EALM has a similar regularization effect to the regularization parameter of ELM2012 but with the advantage that it is not fine-tuned because the crosstalk term is automatically controlled by our Active Learning strategy.

7.2. Results with Big Data. Experiment 4 shows the ability of our Active Learning strategy (EALM) in dealing with problems containing a large amount of data (see Table 2). As mentioned earlier, EALM has inherent properties that make it

Table 4: Average results of 10 runs of 10-fold cross-validation [37].

Key	AL Labels	Effective Labels	Ac	AUC
		EALM		
HRT	88.58 ± 5.97	88.58	0.85 ± 0.01	0.84 ± 0.01
WBCO	39.10 ± 2.79	39.10	0.98 ± 0.00	0.98 ± 0.00
WBCD	80.07 ± 3.96	80.07	0.97 ± 0.01	0.97 ± 0.01
PIMA	188.16 ± 6.18	188.16	0.76 ± 0.01	0.72 ± 0.01
SNR	103.89 ± 2.88	103.89	0.71 ± 0.03	0.72 ± 0.03
ION	105.30 ± 5.86	105.30	0.89 ± 0.01	0.87 ± 0.02
AUST	137.92 ± 8.34	137.92	0.86 ± 0.01	0.86 ± 0.01
LIV	163.54 ± 5.43	163.54	0.60 ± 0.02	0.60 ± 0.02
GER	295.71 ± 8.26	295.71	0.75 ± 0.01	0.68 ± 0.01
SPAM	515.48 ± 18.56	515.48	0.92 ± 0.00	0.91 ± 0.00
		PDKCM		
HRT	65.18 ± 1.43	146.18	0.77 ± 0.02	0.77 ± 0.02
WBCO	63.46 ± 2.50	268.46	0.97 ± 0.00	0.97 ± 0.01
WBCD7	36.01 ± 0.87	207.01	0.93 ± 0.02	0.92 ± 0.02
PIMA	131.02 ± 3.86	361.02	0.72 ± 0.02	0.70 ± 0.02
SNR	77.01 ± 1.85	139.01	0.71 ± 0.03	0.72 ± 0.03
ION	51.01 ± 1.77	156.01	0.83 ± 0.03	0.80 ± 0.03
AUST	145.56 ± 3.18	352.56	0.83 ± 0.01	0.83 ± 0.01
LIV	142.22 ± 3.56	246.22	0.61 ± 0.02	0.61 ± 0.02
GER	247.85 ± 5.07	547.85	0.71 ± 0.01	0.64 ± 0.01
SPAM	314.13 ± 6.07	1694.13	0.87 ± 0.01	0.87 ± 0.01
		PCBGZ		
HRT	50.83 ± 2.10	131.83	0.78 ± 0.02	0.78 ± 0.02
WBCO	182.32 ± 2.98	387.32	0.97 ± 0.01	0.97 ± 0.01
WBCD	203.67 ± 2.79	374.67	0.96 ± 0.01	0.96 ± 0.01
PIMA	193.01 ± 4.35	423.01	0.71 ± 0.01	0.69 ± 0.02
SNR	93.28 ± 1.77	155.28	0.71 ± 0.03	0.70 ± 0.02
ION	116.28 ± 3.33	221.28	0.86 ± 0.01	0.83 ± 0.02
AUST	170.40 ± 2.75	377.40	0.82 ± 0.02	0.81 ± 0.02
LIV	156.05 ± 2.65	260.05	0.59 ± 0.03	0.59 ± 0.03
GER	180.43 ± 3.37	480.43	0.70 ± 0.01	0.63 ± 0.01
SPAM	1168.10 ± 12.43	2548.10	0.89 ± 0.01	0.89 ± 0.01
		PCBGZ-OPT		
HRT	168.47 ± 0.36	249.47	0.77 ± 0.03	0.76 ± 0.03
WBCO	424.06 ± 0.74	629.06	0.97 ± 0.01	0.96 ± 0.01
WBCD	353.04 ± 0.54	524.04	0.96 ± 0.01	0.96 ± 0.01
PIMA	480.08 ± 0.51	710.08	0.69 ± 0.02	0.67 ± 0.02
SNR	130.46 ± 0.30	192.46	0.71 ± 0.03	0.70 ± 0.02
ION	219.55 ± 0.33	324.55	0.86 ± 0.03	0.83 ± 0.03
AUST	430.49 ± 0.61	637.49	0.79 ± 0.01	0.79 ± 0.01
LIV	215.38 ± 0.29	319.38	0.59 ± 0.03	0.59 ± 0.03
GER	624.64 ± 0.63	924.64	0.70 ± 0.01	0.64 ± 0.02
SPAM	2860.56 ± 1.69	4240.56	0.89 ± 0.01	0.89 ± 0.01
		SVMTK		
HRT	45.70 ± 4.24	126.70	0.82 ± 0.01	0.82 ± 0.01
WBCO	18.70 ± 1.83	223.70	0.97 ± 0.00	0.97 ± 0.00
WBCD	40.00 ± 2.79	211.00	0.98 ± 0.00	0.97 ± 0.00
PIMA	138.60 ± 26.06	368.60	0.75 ± 0.01	0.72 ± 0.01
SNR	59.60 ± 9.71	121.60	0.75 ± 0.02	0.75 ± 0.02
ION	55.80 ± 5.77	160.80	0.90 ± 0.01	0.87 ± 0.01
AUST	68.90 ± 13.90	275.90	0.85 ± 0.01	0.85 ± 0.01
LIV	96.40 ± 26.01	200.40	0.64 ± 0.02	0.64 ± 0.02
GER	207.40 ± 17.33	507.40	0.74 ± 0.01	0.66 ± 0.01
SPAM	340.00 ± 25.26	1720.00	0.92 ± 0.00	0.92 ± 0.00
		SVMALL		
HRT	170.00	251.00	0.81 ± 0.01	0.83 ± 0.08
WBCO	430.00	635.00	0.97 ± 0.00	0.97 ± 0.02
WBCD	358.00	529.00	0.98 ± 0.00	0.99 ± 0.02
PIMA	485.00	715.00	0.74 ± 0.01	0.71 ± 0.08
SNR	131.00	193.00	0.81 ± 0.01	0.82 ± 0.10
ION	221.00	326.00	0.91 ± 0.01	0.90 ± 0.07
AUST	434.00	641.00	0.84 ± 0.01	0.84 ± 0.04
LIV	217.00	321.00	0.67 ± 0.02	0.62 ± 0.11
GER	630.00	930.00	0.75 ± 0.01	0.64 ± 0.04
SPAM	2898.00	4278.00	0.93 ± 0.00	0.93 ± 0.02
		ELM2012		
HRT	170.00	251.00	0.82 ± 0.01	0.82 ± 0.01
WBCO	430.00	635.00	0.97 ± 0.00	0.97 ± 0.00
WBCD	358.00	529.00	0.97 ± 0.00	0.97 ± 0.00
PIMA	484.00	714.00	0.73 ± 0.00	0.74 ± 0.00
SNR	131.00	193.00	0.76 ± 0.01	0.76 ± 0.02
ION	221.00	326.00	0.87 ± 0.01	0.84 ± 0.01
AUST	434.00	641.00	0.85 ± 0.00	0.85 ± 0.00
LIV	217.00	321.00	0.64 ± 0.02	0.63 ± 0.02
GER	630.00	930.00	0.70 ± 0.01	0.72 ± 0.01
SPAM	2898.00	4278.00	0.93 ± 0.00	0.92 ± 0.00

highly attractive to handle Big Data. EALM algorithm enables stream-based incremental learning, elimination of redundant patterns, and learning from a reduced informative dataset. Although the regularized formulations of ELM (ELM2012) and linear SVM do not implement such aspects, their results over the same conditions were considered here as a baseline for comparison.

Regularization parameters C of linear SVM and of ELM2012 were configured using a grid search with 10-fold cross-validation procedure on 30% of the training set. All datasets were normalized to mean 0 and standard deviation 1. Table 5 lists the average values achieved by EALM, ELM2012, and SVM for the test subsets. These values were calculated over 10 runs (training/test cases) with the metrics accuracy (Ac) and AUC. The average number of selected labels by EALM (Used Labels) is also shown.

As can be verified in Table 5, EALM, ELM2012, and linear SVM achieved similar performances for Splice and Adult datasets. Nevertheless, EALM used a reduced number of labels (42.28% and 18.88%, resp.), which can be highly positive in a Big Data setting in which the labels have a high cost. Datasets IJCNN1 and Web Gaussian, in turn, are highly imbalanced and this issue may have affected the AUC performance for all models. In the particular case of Web Gaussian, linear SVM achieved better results than EALM. This is because SVM used all labels in the optimization

Table 5: Big Data results.

Key	Used Labels	Ac	AUC
	EALM		
Splice	422.80 ± 14.86 (42.28%)	0.83 ± 0.01	0.83 ± 0.01
IJCNN1	3244.70 ± 112.04 (6.49%)	0.96 ± 0.01	0.86 ± 0.02
Web Gaussian	2108.50 ± 82.46 (4.23%)	0.98 ± 0.01	0.77 ± 0.01
Adult	4285.70 ± 88.35 (18.88%)	0.84 ± 0.01	0.76 ± 0.01
	ELM2012		
Splice	1000 (100%)	0.85 ± 0.01	0.85 ± 0.01
IJCNN1	49990 (100%)	0.95 ± 0.01	0.79 ± 0.01
Web Gaussian	49749 (100%)	0.98 ± 0.01	0.79 ± 0.01
Adult	22696 (100%)	0.84 ± 0.00	0.74 ± 0.00
	SVM		
Splice	1000 (100%)	0.81 ± 0.02	0.81 ± 0.02
IJCNN1	49990 (100%)	0.97 ± 0.01	0.91 ± 0.01
Web Gaussian	49749 (100%)	0.99 ± 0.01	0.87 ± 0.01
Adult	22696 (100%)	0.84 ± 0.01	0.75 ± 0.01

process. EALM used only 6.49% and 4.23%, respectively, and yet its results were satisfactory.

8. Conclusion

Big Data problems demand data models with abilities to handle time-varying, massive, and high dimensional data. In this scenario, Active Learning emerges as an attractive technique for the development of high performance classifiers using few data. The importance of Active Learning for Big Data becomes more evident when labeling cost is high and when data is presented to the learner via data streams. Some researches have developed linear models to perform Active Learning which take some unrealistic assumptions about data distribution and require setting of free parameters. These aspects directly impact the main reason of Active Learning, which is cost reduction, and their use could be almost prohibitive for massive datasets.

Our stream-based Active Learning strategy is little sensitive to parameter setting. This is achieved by using an ELM hidden layer projection whose size is larger than the dimension of input data. This issue has already been claimed in others ELM-based studies in literature [17, 42]. Overfitting is inherently avoided via the Hebbian Learning crosstalk term, which leads to smooth linear separations in the ELM output layer. This fact was confirmed with similar performances achieved by our Active Learning strategy and large-margin strategies such as ELM+linear SVM and ELM+Schohn's heuristic. Our results are also close to those of regularized ELM which indicates that the crosstalk term of our model has a similar regularization effect to the regularization parameter of ELM, but with the advantage that the crosstalk term is automatically controlled by our Active Learning strategy.

The use of the convergence test as a labeling criterion enables the iterative selection of the most informative patterns without the need to calculate the distances for any previous pattern, as is usually accomplished by the margin-based Active Learning approaches. This simply means that our strategy does not hold any pattern previously learned, so that it is particularly suitable to scenarios that require online learning, such as stream-based Big Data domains.

The results of Experiment 4 show that our approach is effective when applied to massive data. Our approach with a reduced training dataset was able to achieve a performance similar to an SVM trained with all patterns. This fact highlights the importance of our approach to current Big Data applications.

Appendices

A. Deduction of (13)

The error $\epsilon_k^2 = (y_k - \widehat{y}_k)^2$ due to an arbitrary sample k, after expansion, is given by

$$\epsilon_k^2 = y_k^2 - 2y_k\widehat{y}_k + \widehat{y}_k^2. \tag{A.1}$$

Substituting $\widehat{y}_k = (\sum_{i=1}^{N} y_i\mathbf{x}_i)^T\mathbf{x}_k$ in (A.1) results in

$$\epsilon_k^2 = y_k^2 - 2y_k\left(\sum_{i=1}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k + \left(\left(\sum_{i=1}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k\right)^2. \tag{A.2}$$

Removing the term $y_k\mathbf{x}_k^T\mathbf{x}_k$ due to k_{th} sample from the summation results in

$$\begin{aligned}\epsilon_k^2 = {} & y_k^2 - 2y_k\left(y_k\mathbf{x}_k\right)^T\mathbf{x}_k - 2y_k\left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k \\ & + \left[\left(y_k\mathbf{x}_k\right)^T\mathbf{x}_k + \left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k\right]^2,\end{aligned} \tag{A.3}$$

$$\begin{aligned}\epsilon_k^2 = {} & y_k^2 - 2y_k\left(y_k\mathbf{x}_k\right)^T\mathbf{x}_k - 2y_k\left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k \\ & + \left[\left(y_k\mathbf{x}_k\right)^T\mathbf{x}_k\right]^2 \\ & + 2\left(y_k\mathbf{x}_k\right)^T\mathbf{x}_k\left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k \\ & + \left[\left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k\right]^2,\end{aligned} \tag{A.4}$$

$$\begin{aligned}\epsilon_k^2 &= \left[y_k - \left(y_k\mathbf{x}_k\right)^T\mathbf{x}_k\right]^2 \\ &+ \left[y_k - \left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k\right]^2 - y_k^2 \\ &+ 2\left(y_k\mathbf{x}_k\right)^T\mathbf{x}_k\left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k,\end{aligned} \tag{A.5}$$

$$\begin{aligned}\epsilon_k^2 &= \left[y_k - \left(y_k\mathbf{x}_k\right)^T\mathbf{x}_k\right]^2 \\ &+ \left[y_k - \left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k\right]^2 \\ &- \left[y_k - \mathbf{x}_k^T\mathbf{x}_k\left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k\right]^2 \\ &+ \left[\mathbf{x}_k^T\mathbf{x}_k\left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k\right]^2.\end{aligned} \tag{A.6}$$

Considering that the input data is normalized $(\mathbf{x}_k)^T\mathbf{x}_k = 1$ we have that

$$\begin{aligned}\epsilon_k^2 &= \overbrace{\left[y_k - \left(y_k\mathbf{x}_k\right)^T\mathbf{x}_k\right]^2}^{=0} \\ &+ \overbrace{\left[y_k - \left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k\right]^2}^{(y_k-\widehat{y}_k+y_k)^2} \\ &- \overbrace{\left[y_k - \mathbf{x}_k^T\mathbf{x}_k\left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k\right]^2}^{(y_k-(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i)^T\mathbf{x}_k)^2} \\ &+ \overbrace{\left[\mathbf{x}_k^T\mathbf{x}_k\left(\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i\right)^T\mathbf{x}_k\right]^2}^{((\sum_{i=1,i\neq k}^{N} y_i\mathbf{x}_i)^T\mathbf{x}_k)^2},\end{aligned} \tag{A.7}$$

$$\epsilon_k^2 = \left[2y_k - \widehat{y}_k\right]^2 - \left(y_k - \text{crosstalk}\right)^2 + \text{crosstalk}^2,$$

$$\epsilon_k^2 = \left[2y_k - \widehat{y}_k\right]^2 - y_k^2 + 2y_k\text{crosstalk} - \text{crosstalk}^2 + \text{crosstalk}^2,$$

$$\epsilon_k^2 = \left[2y_k - \widehat{y}_k\right]^2 - y_k^2 + 2y_k\text{crosstalk},$$

$$\epsilon_k^2 = \left[2y_k - \widehat{y}_k\right]^2 + y_k\left(2\text{crosstalk} - y_k\right).$$

B. Proof of Theorem 1

Theorem 1. *For two linearly separable classes the maximum number of labels needed for convergence of a Hebbian Perceptron is given by $t_{\max} = (\beta + 2\theta)/\alpha^2$, where β is the maximum square norm of the training patterns, θ is the margin of the most distant pattern from the separating hyperplane, and α is the margin of the closest pattern to the separating hyperplane.*

Proof. Suppose that we wish to classify two linearly separable classes ζ_1 and ζ_2 belonging to dataset ζ and that the initial weight $\mathbf{w}_0 = \mathbf{0}$. For patterns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$ with labels $y_1, y_2, \ldots, y_t \in \{-1, +1\}$ the updated weight $\mathbf{w}_{t+1}$ is given by

$$\mathbf{w}_{t+1} = \mathbf{x}_1y_1 + \mathbf{x}_2y_2 + \cdots + \mathbf{x}_ty_t. \tag{B.1}$$

Since classes ζ_1 and ζ_2 are linearly separable, there is a solution $\mathbf{w}^*$ that correctly classifies all patterns. Thus, for a fixed solution $\mathbf{w}^*$ we can define a positive number α according to

$$\alpha = \min_{\mathbf{x}_t\in\zeta}\left|\mathbf{w}^{*T}\mathbf{x}_ty_t\right| = \min_{\mathbf{x}_t\in\zeta}\left|\mathbf{w}^{*T}\mathbf{x}_t\right|. \tag{B.2}$$

Multiplying both sides of (B.1) by $\mathbf{w}^{*T}$, we have

$$\mathbf{w}^{*T}\mathbf{w}_{t+1} = \mathbf{w}^{*T}\mathbf{x}_1y_1 + \mathbf{w}^{*T}\mathbf{x}_2y_2 + \cdots + \mathbf{w}^{*T}\mathbf{x}_ty_t. \tag{B.3}$$

According to the definition of (B.2) we can ensure that

$$\mathbf{w}^{*T}\mathbf{w}_{t+1} \geq t\alpha. \tag{B.4}$$

Using now Cauchy-Schwarz inequality [43] we reach

$$\left\|\mathbf{w}^*\right\|^2\left\|\mathbf{w}_{t+1}\right\|^2 \geq \left[\mathbf{w}^{*T}\mathbf{w}_{t+1}\right]^2. \tag{B.5}$$

Observing (B.4) we can conclude that $[\mathbf{w}^{*T}\mathbf{w}_{t+1}]^2 \geq t^2\alpha^2$, or equivalently

$$\left\|\mathbf{w}_{t+1}\right\|^2 \geq \frac{t^2\alpha^2}{\left\|\mathbf{w}^*\right\|^2}. \tag{B.6}$$

We can also write (B.1) as $\mathbf{w}_{k+1} = \mathbf{w}_k + y_k\mathbf{x}_k$, where $k = 1, \ldots, t$. Taking the square of the Euclidean norm on both sides of this equation we obtain

$$\left\|\mathbf{w}_{k+1}\right\|^2 = \left\|\mathbf{w}_k\right\|^2 + \left\|\mathbf{x}_ky_k\right\|^2 + 2\mathbf{w}_k^T\mathbf{x}_ky_k. \tag{B.7}$$

Equation (B.7) can now be rewritten as follows:

$$\left\|\mathbf{w}_{k+1}\right\|^2 - \left\|\mathbf{w}_k\right\|^2 = \left\|\mathbf{x}_ky_k\right\|^2 + 2\mathbf{w}_k^T\mathbf{x}_ky_k. \tag{B.8}$$

Summing up the terms $k = 1, \ldots, t$ of (B.8) and considering the initial condition $\mathbf{w}_0 = \mathbf{0}$ we have

$$\left\|\mathbf{w}_{t+1}\right\|^2 = \sum_{k=1}^{t}\left\|\mathbf{x}_ky_k\right\|^2 + 2\sum_{k=1}^{t}\mathbf{w}_k^T\mathbf{x}_ky_k. \tag{B.9}$$

Since $\|\mathbf{x}_ky_k\|^2 = \|\mathbf{x}_k\|^2$, regardless of the value of y_k, we can define

$$\begin{aligned}\beta &= \max_{\mathbf{x}_k\in\zeta}\left\|\mathbf{x}_k\right\|^2, \\ \theta &= \max_{\mathbf{x}_k\in\zeta}\left|\mathbf{w}_k^T\mathbf{x}_ky_k\right| = \max_{\mathbf{x}_k\in\zeta}\left|\mathbf{w}_k^T\mathbf{x}_k\right|.\end{aligned} \tag{B.10}$$

Using the definitions of (B.10), we can ensure that the following inequality is true:

$$\left\|\mathbf{w}_{t+1}\right\|^2 \leq t\beta + 2t\theta. \tag{B.11}$$

From (B.11) we can conclude that the square of the norm of weights vector grows linearly with the value of t. This result conflicts with that presented in (B.6) for values sufficiently large of t.

In this case the value of t can not be greater than a certain value $t_{\max}$ where (B.6) and (B.11) are equal:

$$\frac{t_{\max}^2\alpha^2}{\left\|\mathbf{w}^*\right\|^2} = t_{\max}\left(\beta + 2\theta\right). \tag{B.12}$$

Thus the value of $t_{\max}$ can be obtained by

$$t_{\max} = \frac{\left(\beta + 2\theta\right)\left\|\mathbf{w}^*\right\|^2}{\alpha^2}. \tag{B.13}$$

For normalized weights [23] we have $\|\mathbf{w}^*\| = 1$ and (B.13) is rewritten as

$$t_{\max} = \frac{\beta + 2\theta}{\alpha^2}. \tag{B.14}$$

□

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors would like to acknowledge FAPEMIG for the financial support.

References

[1] A. Jacobs, "The pathologies of big data," *Communications of the ACM*, vol. 52, no. 8, pp. 36–44, 2009.

[2] B. Settles, "Active learning literature survey," Tech. Rep., University of Wisconsin-Madison, Madison, Wis, USA, 2010.

[3] S. Doyle, J. Monaco, M. Feldman, J. Tomaszewski, and A. Madabhushi, "An active learning based classification strategy for the minority class problem: application to histopathology annotation," *BMC Bioinformatics*, vol. 12, no. 1, article 424, 14 pages, 2011.

[4] F. Provost, P. Melville, and M. Saar-Tsechansky, "Data acquisition and cost-effective predictive modeling: targeting offers for electronic commerce," in *Proceedings of the 9th International Conference on Electronic Commerce (ICEC '07)*, pp. 389–398, ACM, New York, NY, USA, August 2007.

[5] A. G. Hauptmann, W. H. Lin, R. Yan, J. Yang, and M. Y. Chen, "Extreme video retrieval: joint maximization of human and computer performance," in *Proceedings of the 14th Annual ACM International Conference on Multimedia (MULTIMEDIA '06)*, pp. 385–394, ACM, Santa Barbara, Calif, USA, October 2006.

[6] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, no. 2, pp. 201–221, 1994.

[7] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR '94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Organised by Dublin City University*, pp. 3–12, Springer, London, UK, 1994.

[8] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *The Journal of Machine Learning Research*, vol. 2, pp. 45–66, 2002.

[9] G. Schohn and D. Cohn, "Less is more: active learning with support vector machines," in *Proceedings of the 17th International Conference on Machine Learning*, San Francisco, Calif, USA, 2000.

[10] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.

[11] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan, New York, NY, USA, 1994.

[12] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Worst-case analysis of selective sampling for linear classification," *Journal of Machine Learning Research*, vol. 7, pp. 1205–1230, 2006.

[13] S. Dasgupta, A. T. Kalai, and C. Monteleoni, "Analysis of perceptron-based active learning," *Journal of Machine Learning Research*, vol. 10, pp. 281–299, 2009.

[14] C. Monteleoni and M. Kääriäinen, "Practical online active learning for classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, vol. 328, pp. 1–8, IEEE, Minneapolis, Minn, USA, June 2007.

[15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[16] Q. Liu, Q. He, and Z. Shi, "Extreme support vector machine classifier," in *Advances in Knowledge Discovery and Data Mining*, T. Washio, E. Suzuki, K. Ting, and A. Inokuchi, Eds., vol. 5012 of *Lecture Notes in Computer Science*, pp. 222–233, Springer, Berlin, Germany, 2008.

[17] B. Frénay and M. Verleysen, "Using SVMs with randomised feature spaces: an extreme learning approach," in *Proceedings of the European Symposium on Artificial Neural Networks—Computational Intelligence and Machine Learning*, pp. 315–320, Bruges, Belgium, April 2010.

[18] A. Guillory, E. Chastain, and J. Bilmes, "Active learning as non-convex optimization," in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS '09)*, vol. 5, Clearwater Beach, Fla, USA, 2009.

[19] I. Guyon and D. Stork, "Linear discriminant and support vector classifiers," in *Advances in Large Margin Classifiers*, A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, Eds., pp. 147–169, MIT Press, Cambridge, Mass, USA, 2000.

[20] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transactions on Electronic Computers*, vol. EC-14, no. 3, pp. 326–334, 1965.

[21] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[22] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[23] M. Fernández-Delgado, J. Ribeiro, E. Cernadas, and S. B. Ameneiro, "Direct parallel perceptrons (DPPs): fast analytical calculation of the parallel perceptrons weights with margin control for classification tasks," *IEEE Transactions on Neural Networks*, vol. 22, no. 11, pp. 1837–1848, 2011.

[24] N. Nilsson, *Learning Machines*, McGraw-Hill, New York, NY, USA, 1965.

[25] W. S. Andrus and K. T. Bird, "Radiology and the receiver operating characteristic (ROC) curve," *CHEST*, vol. 67, no. 4, pp. 378–379, 1975.

[26] D. O. Hebb, *The Organization of Behavior*, Wiley, New York, NY, USA, 1949.

[27] H. Anton and R. Busby, *Contemporary Linear Algebra*, Wiley, 2002.

[28] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation*, Advanced Book Program, Addison-Wesley, Boston, Mass, USA, 1991.

[29] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, 1982.

[30] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Transactions on Information Theory*, vol. 33, no. 4, pp. 461–482, 1987.

[31] P. Peretto, "On learning rules and memory storage abilities of asymmetrical neural networks," *Journal de Physique*, vol. 49, pp. 711–726, 1988.

[32] L. Personnaz, I. Guyon, and G. Dreyfus, "Information storage and retrieval in spin-glass like neural networks," *Journal de Physique Lettres*, vol. 46, no. 8, pp. 359–365, 1985.

[33] E. Gardner, "Maximum sotorage capacity in neural networks," *Europhysics Letters*, vol. 4, no. 4, pp. 481–485, 1987.

[34] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, no. 2, pp. 219–269, 1995.

[35] P. L. Bartlett, "For valid generalization the size of the weights is more important than the size of the network," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds., vol. 9, MIT Press, Cambridge, Mass, USA, 1997.

[36] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan, Washington, DC, USA, 1962.

[37] E. G. Horta and A. P. Braga, "An extreme learning approach to active learning," in *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 613–618, i6doc.com, Bruges, Belgium, 2014.

[38] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: a library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[39] S. Sonnenburg, G. Rätsch, S. Henschel et al., "The SHOGUN machine learning toolbox," *Journal of Machine Learning Research*, vol. 11, pp. 1799–1802, 2010.

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

[41] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, article 27, 2011.

[42] B. Frénay and M. Verleysen, "Parameter-insensitive kernel in extreme learning for non-linear support vector regression," *Neurocomputing*, vol. 74, no. 16, pp. 2526–2531, 2011.

[43] J. M. Steele, *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*, Cambridge University Press, New York, NY, USA, 2004.