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

1Graduate Program in Electrical Engineering, Federal University of Minas Gerais, Avenida Antônio Carlos 6627, 31270-901 Belo Horizonte, MG, Brazil
2Institute of Science and Technology, Federal University of Jequitinhonha and Mucuri Valleys, Rodovia MGT 367, Km 583, 3000 Alto da Jacuba, 39100-000 Diamantina, MG, Brazil
3Department of Electrical Engineering, Federal University of Minas Gerais, Avenida Antônio Carlos 6627, 31270-901 Belo Horizonte, MG, Brazil
4Department 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

Copyright © 2015 Euler Guimarães Horta et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 \((x_i, y_i)\) pairs obtained by sampling \(x_i\) from the input space \(\chi\) according to a probability function \(P(X)\) and by querying an oracle function \(f_g(x_i)\) for the labels \(y_i\). The final goal of learning is to obtain the parameters \(Z\) and \(w\) of the approximation function \(f(x, Z, w)\) so that \(f(x, Z, w) \approx f_g(x)\). Convergence conditions to guarantee that \(f(x, Z, w) \to f_g(x)\) in \(\chi\) depend on the representativeness and size of the learning set \(D_L = \{x_i, y_i\}_{i=1}^{N_L}\). Reliable labeling of the input samples \(D_u = \{x_i\}_{i=1}^{N_u}\) is of paramount importance to guarantee robustness of the approximation function \(f(x, Z, w)\). 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 \(P(X)\) 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 \( P(X) \). Nonetheless, the construction of learning machines that may influence \( P(X) \) 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 \( P(X) \) 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 \( f(x, Z, w) \) that is valid in the whole input domain by behaving as similar as possible to the label-generator function \( f_j(x) \). The goal is to select more representative samples that will result in \( f(x, Z, w) \rightarrow f_j(x) \). 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 \( f(x, Z, w) \) is composed of the hidden layer mapping function \( \psi(x, Z) \) and the output function \( \phi(\psi(x, Z), w) \). Since both functions are single layer, \( \phi(\cdot, \cdot) \) can only perform linear separation and \( \psi(x, Z) \) is expected to linearize the problem. Nevertheless, the difficulty with the nonlinear approach is that in order to obtain \( \psi(x, Z) \) 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 \( \psi(x, Z) \). The basic principle of ELM is to randomly sample the elements of \( Z \) and to expand the input space into a higher dimension in order to obtain \( \psi(x, Z) \). This is the most fundamental difference between ELM, feedforward neural networks, and SVM, since in these two models the function \( \psi(x, Z) \) 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 \( \hat{y}_i \) for the input pattern \( x_i \) is a linear combination of the labels \( y_j \) weighted by the kernel \( K(x_i, x_j) \) between \( x_i \) and all other learning patterns \( x_j \): \( \hat{y}_i = \text{sign}(\sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) + b) \). The linear combination coefficients are the Lagrange multipliers \( \alpha_j \) 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 \( \hat{y}_i \), SVM’s learning can be seen as a sample selection problem. Given the proper kernel parameters, the selection of margin patterns \( x_i \) and the Lagrange multipliers \( \alpha_i \) yields error minimization and margin maximization [10]. In such a scenario, “discarded” samples are those assigned to null Lagrange multipliers \( \alpha_i = 0 \) and the “selected” ones, the support vectors, are those with Lagrange multipliers in the range \( 0 \leq \alpha_i \leq C \), 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 \( \psi(x, Z) \).

The mapping \( H \) of \( X \) 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 \( H \) 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 regulari-
ation.

The method presented in this paper is a classifier com-
posed of a large size ELM hidden layer and an output layer
learned via a Hebbian Learning Perceptron with normal-
ized 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. Experi-
mental 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 Learn-
ing Machines. Section 3 presents the Hebbian Learning.
Section 4 discusses how overfitting can be controlled using
Hebbian Learning. Section 5 extends the Perceptron Con-
vergence 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 \( X \) 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
\( y \) contain the corresponding labels of each one of the \( N \) input
samples of \( X \):

\[
X = \begin{bmatrix}
X_1^T \\
\vdots \\
X_N^T
\end{bmatrix} = 
\begin{bmatrix}
x_{11} & \cdots & x_{1n} \\
\vdots & \ddots & \vdots \\
x_{N1} & \cdots & x_{Nn}
\end{bmatrix}_{N \times n}
\]

(1)

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

\[
H = \begin{bmatrix}
\psi(x_1^T, Z, b) \\
\vdots \\
\psi(x_N^T, Z, b)
\end{bmatrix} = \begin{bmatrix}
\text{sigmoid}(x_1^T \cdot z_1 + b_1) & \cdots & \text{sigmoid}(x_1^T \cdot z_p + b_p) \\
\vdots & \ddots & \vdots \\
\text{sigmoid}(x_N^T \cdot z_1 + b_1) & \cdots & \text{sigmoid}(x_N^T \cdot z_p + b_p)
\end{bmatrix}_{N \times p}
\]

(2)

In the particular case of ELM since the elements of \( Z \) and
\( b \) 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 \( X \) into \( H \) is assumed to be linearly separable.

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

\[
Hw = y.
\]

(3)

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

\[
w = H^+y,
\]

(4)

where \( H^+ \) is the Moore-Penrose pseudoinverse. The network
response \( \tilde{y} \) to an input pattern \( x \) is obtained by first calculating
\( H \) and then by estimating the output as \( \tilde{y} = \text{sign}(Hw) \) [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 result-
ing general function \( f(x, Z, b, w) \) is expected to be robust to
\( x_i \notin X \). 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
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 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]

$$minimize: \quad L_{\text{Primal}} = \frac{1}{2} \| w \|^2 + \frac{C}{2} \sum_{i=1}^{N} \xi_i^2$$

$$subject to: \quad \psi \left( x_i^T, Z, b \right) \cdot w = y_i - \xi_i, \quad \forall i = 1, \ldots, N,$$

where $\xi_i$ is the training error with respect to the training pattern $x$ and $C$ is a regularization parameter.

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

$$L_{\text{Dual}} = \frac{1}{2} \| w \|^2 + \frac{C}{2} \sum_{i=1}^{N} \xi_i^2$$

$$- \sum_{i=1}^{N} \alpha_i \left( \psi \left( x_i^T, Z, b \right) \cdot w - y_i + \xi_i \right),$$

where each Lagrange multiplier $\alpha_i$ corresponds to the $i$th 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:
$$w = H^T \left( \frac{1}{C} + HH^T \right)^{-1} y.$$  \hspace{1cm} (7)

2. For the case where the learning set has large size:
$$w = \left( \frac{1}{C} + H^T H \right)^{-1} H^T y.$$  \hspace{1cm} (8)
For binary classification the output for a input pattern \( x_i \) is estimated as \( \hat{y} = \text{sign}(\langle \psi(x_i, Z, b), w \rangle) \) [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 \( N = p \), 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

\[
w = X^T y, \quad (9)
\]

where \( w \) is the \( n \times 1 \) weight vector, \( X \) is the \( N \times n \) input data matrix, and \( y \) is the \( N \times 1 \) vector containing the output values.

Since the estimation \( \hat{y} \) of vector \( y \) is given by \( \hat{y} = Xw \), the substitution of \( w \) from (9) into this expression leads to \( \hat{y} = XX^T y \). Therefore, in order to perfectly retrieve \( y \) we must have \( XX^T = I \). If such a condition is met the training error is zero, since \( \hat{y} = y \), and the learning rule is equivalent to the pseudoinverse method of (4). This condition can only happen if \( x_i \perp x_j \) and \( x_i^T x_j = 1 \forall i,j \). In such a situation \( X \) forms an orthonormal basis and \( X^T = X^{-1} \) [27] which leads (9) and (4) to be equivalent. However, in most real situations \( X \) will not be an orthonormal basis and a residual term because the product \( XX^T \) will cause a shift of \( \hat{y} \) in relation to \( y \). Since an individual \( \hat{y}_k \) is obtained as \( \hat{y}_j = \sum_{j=1}^{N} x_k^T x_j y_j \), the term corresponding to \( j = k \) can be separated from the summation which leads to the following:

\[
\hat{y}_k = x_k^T x_k y_k + \sum_{j=1, j\neq k}^{N} x_k^T x_j y_j, \quad (10)
\]

In the particular situation when the input data is normalized, that is, \( x_k^T x_k = 1 \), (10) can be simplified into

\[
\hat{y}_k = y_k + \frac{\text{crosstalk}}{\sum_{j=1, j\neq k}^{N} x_k^T x_j y_j}. \quad (11)
\]

The interpretation of (11) [28] is that the estimated response \( \hat{y} \) for the input pattern \( x_k \) 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 \( x_k \). 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

$$\mathbf{w}_{\text{hebb}} = x_0 y_0 + x_1 y_1 + \cdots + x_i y_i + \sum_{t+1}^N y_{t+1}.$$  \hspace{1cm} (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 ̂\(y_k\) 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 \(e_k^2 = (y_k - ̂y_k)^2\) due to an arbitrary association

$$e_k^2 = \frac{1}{N} \sum_{i=1, i \neq k}^N y_i x_i^T x_k - y_k^2.$$  \hspace{1cm} (13)

The penalization term of (13) was obtained from the original error \((y_k - ̂y_k)^2\) 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 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
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.

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

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

(15)

where $\beta$ is the maximum square norm of the training patterns, $\theta$ is the margin of the most distant pattern from the separating hyperplane, and $\alpha$ is the margin of the closest pattern to the separating hyperplane.

The deductions of $\alpha$, $\beta$ and $\theta$ are presented in Appendix B. Considering the learning set $\xi$, $\alpha$, $\beta$ and $\theta$ can be defined as presented in the following equations:

$$\alpha = \min_{x \in \xi} |w^T x|,$$

$$\beta = \max_{x \in \xi} ||x||^2,$$

$$\theta = \max_{x \in \xi} |w^T x|.$$  

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 $t_{\text{max}}$ 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 $t_{\text{max}}$ 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 $\alpha$ in (15). The variable $\theta$ corresponds to the maximum between the calculated $\alpha$ and the previous $\alpha$. The variable $\beta$ is the maximum between the norm of the current vector and the largest norm of the previous training vectors. Using these variables, $t_{\text{max}}$ is calculated using (15).
The convergence test is accomplished using \( t_{\text{max}} \). If \( t_{\text{max}} \) 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:

\[
\alpha = \left\lVert \Psi(x^T, Z, b) \right\rVert_2^2;
\beta = \max_{k=1...m} \left( \left\lVert \Psi(x^T, Z, b) \right\rVert_2^2, \beta \right);
\theta = \max_{k=1...m} \left( \left\lVert w^T \Psi(x^T, Z, b) \right\rVert_2, \theta \right);
\]

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

if \( t > m \) then

Query the label \( y \);

Update \( w = (w + \Psi(x^T, Z, b))y/\left\lVert w + \Psi(x^T, Z, b) \right\lVert_2 \);

\( m = m + 1 \);

end

Until \( (m = L) \) or \( (F = \emptyset) \);

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.
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.

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 (ELMPCP); (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

\[ \text{AUC} = \frac{1}{n} \sum_{i=1}^{n} \frac{\text{TP}_i + \text{TN}_i}{\text{TP}_i + \text{TN}_i + \text{FP}_i + \text{FN}_i} \]

where TP, TN, FP, and FN are the counts of true positives, true negatives, false positives, and false negatives, respectively.

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

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

Table 2: Datasets for Experiment 4.

<table>
<thead>
<tr>
<th>Name</th>
<th>Training set</th>
<th>Test set</th>
<th>Number of inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splice</td>
<td>1000</td>
<td>2175</td>
<td>60</td>
</tr>
<tr>
<td>IJCNN1</td>
<td>49990</td>
<td>91701</td>
<td>22</td>
</tr>
<tr>
<td>Web Gaussian</td>
<td>49749</td>
<td>14951</td>
<td>300</td>
</tr>
<tr>
<td>Adult</td>
<td>22696</td>
<td>9865</td>
<td>123</td>
</tr>
</tbody>
</table>

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).
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.
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 ELM 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
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 $b = (\max_{x\in\mathcal{X}}|x|^2)/2$ [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].

<table>
<thead>
<tr>
<th>Key</th>
<th>AL Labels</th>
<th>Effective Labels</th>
<th>Ac</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRT</td>
<td>88.58 ± 5.97</td>
<td>88.58 ± 0.05</td>
<td>0.84 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>WBCO</td>
<td>39.10 ± 2.79</td>
<td>39.10 ± 0.00</td>
<td>0.98 ± 0.00</td>
<td></td>
</tr>
<tr>
<td>WBCD</td>
<td>80.07 ± 3.96</td>
<td>80.07 ± 0.01</td>
<td>0.97 ± 0.00</td>
<td></td>
</tr>
<tr>
<td>PIMA</td>
<td>188.16 ± 6.18</td>
<td>188.16 ± 0.76</td>
<td>0.72 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>SNR</td>
<td>103.89 ± 2.88</td>
<td>103.89 ± 0.71</td>
<td>0.72 ± 0.03</td>
<td></td>
</tr>
<tr>
<td>ION</td>
<td>105.30 ± 5.86</td>
<td>105.30 ± 0.89</td>
<td>0.87 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>AUST</td>
<td>137.92 ± 8.34</td>
<td>137.92 ± 0.86</td>
<td>0.86 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>LIV</td>
<td>163.54 ± 5.43</td>
<td>163.54 ± 0.60</td>
<td>0.60 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>GER</td>
<td>295.71 ± 8.26</td>
<td>295.71 ± 0.75</td>
<td>0.68 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>SPAM</td>
<td>515.48 ± 18.56</td>
<td>515.48 ± 0.92</td>
<td>0.91 ± 0.00</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Continued.

<table>
<thead>
<tr>
<th>Key</th>
<th>AL Labels</th>
<th>Effective Labels</th>
<th>Ac</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRT</td>
<td>65.18 ± 1.43</td>
<td>146.18 ± 0.77</td>
<td>0.77 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>WBCO</td>
<td>63.46 ± 2.50</td>
<td>268.46 ± 0.97</td>
<td>0.97 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>WBCD7</td>
<td>36.01 ± 0.87</td>
<td>207.01 ± 0.93</td>
<td>0.92 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>PIMA</td>
<td>131.02 ± 3.86</td>
<td>361.02 ± 0.72</td>
<td>0.70 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>SNR</td>
<td>77.01 ± 1.85</td>
<td>139.01 ± 0.71</td>
<td>0.72 ± 0.03</td>
<td></td>
</tr>
<tr>
<td>ION</td>
<td>51.01 ± 1.77</td>
<td>156.01 ± 0.83</td>
<td>0.80 ± 0.03</td>
<td></td>
</tr>
<tr>
<td>AUST</td>
<td>145.56 ± 3.18</td>
<td>352.56 ± 0.83</td>
<td>0.83 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>LIV</td>
<td>142.22 ± 3.56</td>
<td>246.22 ± 0.62</td>
<td>0.61 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>GER</td>
<td>247.85 ± 5.07</td>
<td>547.85 ± 0.71</td>
<td>0.64 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>SPAM</td>
<td>314.13 ± 6.07</td>
<td>1694.13 ± 0.87</td>
<td>0.81 ± 0.01</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>AL Labels</th>
<th>Effective Labels</th>
<th>Ac</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRT</td>
<td>50.83 ± 2.10</td>
<td>131.83 ± 0.78</td>
<td>0.78 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>WBCO</td>
<td>182.32 ± 2.98</td>
<td>387.32 ± 0.97</td>
<td>0.97 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>WBCD</td>
<td>203.67 ± 2.79</td>
<td>374.67 ± 0.96</td>
<td>0.96 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>PIMA</td>
<td>193.01 ± 4.35</td>
<td>423.01 ± 0.71</td>
<td>0.69 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>SNR</td>
<td>93.28 ± 1.77</td>
<td>155.28 ± 0.71</td>
<td>0.70 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>ION</td>
<td>116.28 ± 3.33</td>
<td>221.28 ± 0.86</td>
<td>0.83 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>AUST</td>
<td>170.40 ± 2.75</td>
<td>377.40 ± 0.82</td>
<td>0.81 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>LIV</td>
<td>156.05 ± 2.65</td>
<td>260.05 ± 0.59</td>
<td>0.59 ± 0.03</td>
<td></td>
</tr>
<tr>
<td>GER</td>
<td>180.43 ± 3.37</td>
<td>480.43 ± 0.70</td>
<td>0.63 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>SPAM</td>
<td>1168.10 ± 12.43</td>
<td>2548.10 ± 0.89</td>
<td>0.89 ± 0.01</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>AL Labels</th>
<th>Effective Labels</th>
<th>Ac</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRT</td>
<td>168.47 ± 0.36</td>
<td>249.47 ± 0.77</td>
<td>0.76 ± 0.03</td>
<td></td>
</tr>
<tr>
<td>WBCO</td>
<td>424.06 ± 0.74</td>
<td>629.06 ± 0.97</td>
<td>0.96 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>WBCD</td>
<td>553.04 ± 0.54</td>
<td>524.04 ± 0.96</td>
<td>0.96 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>PIMA</td>
<td>480.08 ± 0.51</td>
<td>710.08 ± 0.69</td>
<td>0.67 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>SNR</td>
<td>130.46 ± 0.30</td>
<td>192.46 ± 0.71</td>
<td>0.70 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>ION</td>
<td>219.55 ± 0.33</td>
<td>324.55 ± 0.86</td>
<td>0.83 ± 0.03</td>
<td></td>
</tr>
<tr>
<td>AUST</td>
<td>430.49 ± 0.61</td>
<td>637.49 ± 0.79</td>
<td>0.79 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>LIV</td>
<td>215.38 ± 0.29</td>
<td>319.38 ± 0.59</td>
<td>0.59 ± 0.03</td>
<td></td>
</tr>
<tr>
<td>GER</td>
<td>624.64 ± 0.63</td>
<td>924.64 ± 0.70</td>
<td>0.64 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>SPAM</td>
<td>2860.56 ± 1.69</td>
<td>4240.56 ± 0.89</td>
<td>0.89 ± 0.01</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>AL Labels</th>
<th>Effective Labels</th>
<th>Ac</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRT</td>
<td>45.70 ± 4.24</td>
<td>126.70 ± 0.82</td>
<td>0.82 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>WBCO</td>
<td>18.70 ± 1.83</td>
<td>223.70 ± 0.97</td>
<td>0.97 ± 0.00</td>
<td></td>
</tr>
<tr>
<td>WBCD</td>
<td>40.00 ± 2.79</td>
<td>211.00 ± 0.98</td>
<td>0.97 ± 0.00</td>
<td></td>
</tr>
<tr>
<td>PIMA</td>
<td>138.60 ± 26.06</td>
<td>368.60 ± 0.75</td>
<td>0.72 ± 0.01</td>
<td></td>
</tr>
<tr>
<td>SNR</td>
<td>59.60 ± 9.71</td>
<td>121.60 ± 0.75</td>
<td>0.75 ± 0.02</td>
<td></td>
</tr>
<tr>
<td>ION</td>
<td>55.80 ± 5.77</td>
<td>160.80 ± 0.90</td>
<td>0.87 ± 0.01</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Key</th>
<th>Used Labels</th>
<th>Ac</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>EALM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Splice</td>
<td>422.80 ± 14.86 (42.28%)</td>
<td>0.83 ± 0.01</td>
<td>0.83 ± 0.01</td>
</tr>
<tr>
<td>IJCNN1</td>
<td>3244.70 ± 112.04 (6.49%)</td>
<td>0.96 ± 0.01</td>
<td>0.86 ± 0.02</td>
</tr>
<tr>
<td>Web Gaussian</td>
<td>2108.50 ± 82.46 (4.23%)</td>
<td>0.98 ± 0.01</td>
<td>0.77 ± 0.01</td>
</tr>
<tr>
<td>Adult</td>
<td>4285.70 ± 88.35 (18.88%)</td>
<td>0.84 ± 0.01</td>
<td>0.76 ± 0.01</td>
</tr>
<tr>
<td>SVM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Splice</td>
<td>1000 (100%)</td>
<td>0.85 ± 0.01</td>
<td>0.85 ± 0.01</td>
</tr>
<tr>
<td>IJCNN1</td>
<td>49990 (100%)</td>
<td>0.95 ± 0.01</td>
<td>0.79 ± 0.01</td>
</tr>
<tr>
<td>Web Gaussian</td>
<td>49749 (100%)</td>
<td>0.98 ± 0.01</td>
<td>0.79 ± 0.01</td>
</tr>
<tr>
<td>Adult</td>
<td>22696 (100%)</td>
<td>0.84 ± 0.00</td>
<td>0.74 ± 0.00</td>
</tr>
</tbody>
</table>

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

$$e_k^2 = y_k^2 - 2y_k\hat{y}_k + \hat{y}_k^2. \quad (A.1)$$

Substituting $\hat{y}_k = (\sum_{i=1}^{N} y_i x_i)^T x_k$ in (A.1) results in

$$e_k^2 = y_k^2 - 2y_k \left( \sum_{i=1}^{N} y_i x_i \right)^T x_k + \left( \sum_{i=1}^{N} y_i x_i \right)^T x_k. \quad (A.2)$$

Removing the term $y_k x_k^T x_k$ due to $k_{th}$ sample from the summation results in

$$e_k^2 = y_k^2 - 2y_k \left( \sum_{i=1,j \neq k}^{N} y_i x_i \right)^T x_k - 2y_k \left( \sum_{i=1,j \neq k}^{N} y_i x_i \right)^T \left( \sum_{i=1,j \neq k}^{N} y_i x_i \right) x_k$$

$$+ \left( \sum_{i=1,j \neq k}^{N} y_i x_i \right)^T x_k + \left( \sum_{i=1,j \neq k}^{N} y_i x_i \right)^T x_k. \quad (A.3)$$

$$e_k^2 = y_k^2 - 2y_k \left( y_k x_k \right)^T x_k - 2y_k \left( \sum_{i=1,j \neq k}^{N} y_i x_i \right)^T x_k$$

$$+ \left( y_k x_k \right)^T x_k \left( \sum_{i=1,j \neq k}^{N} y_i x_i \right)^T x_k$$

$$+ 2 \left( y_k x_k \right)^T x_k \left( \sum_{i=1,j \neq k}^{N} y_i x_i \right)^T x_k$$

$$+ \left( \sum_{i=1,j \neq k}^{N} y_i x_i \right)^T x_k. \quad (A.4)$$
\[ \epsilon_k^2 = \left[ y_k - \left( y_k x_k^T x_k \right) \right]^2 + \left( y_k \left( \sum_{j=1, j \neq k}^N y_j x_j \right) x_k \right)^2 + 2 \left( y_k x_k^T x_k \left( \sum_{j=1, j \neq k}^N y_j x_j \right) x_k \right)^2. \] (A.5)

\[ \epsilon_k^2 = \left[ y_k - \left( y_k x_k^T x_k \right) \right]^2 + \left( y_k \left( \sum_{j=1, j \neq k}^N y_j x_j \right) x_k \right)^2 + \left( \sum_{i=1, i \neq k}^N y_i x_i \right)^2. \] (A.6)

Considering that the input data is normalized \((x_k)^T x_k = 1\) we have that

\[ \epsilon_k^2 = \left[ y_k - (y_k x_k^T x_k)^2 \right] + \left( y_k \left( \sum_{j=1, j \neq k}^N y_j x_j \right) x_k \right)^2 + \left( \sum_{i=1, i \neq k}^N y_i x_i \right)^2. \] (A.7)

\[ \epsilon_k^2 = [2 y_k - \tilde{y}_k]^2 - (y_k - \text{crosstalk})^2 + \text{crosstalk}^2, \]

\[ \epsilon_k^2 = [2 y_k - \tilde{y}_k]^2 - y_k^2 + 2 y_k \text{crosstalk} - \text{crosstalk}^2 + \text{crosstalk}^2, \]

\[ \epsilon_k^2 = [2 y_k - \tilde{y}_k]^2 + y_k \text{crosstalk} - y_k, \]

\[ \epsilon_k^2 = [2 y_k - \tilde{y}_k]^2 + y_k (2 \text{crosstalk} - y_k). \]

**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 \( t_{\text{max}} = (\beta + 2\theta)/\alpha^2 \), where \( \beta \) is the maximum square norm of the training patterns, \( \theta \) is the margin of the most distant pattern from the separating hyperplane, and \( \alpha \) is the margin of the closest pattern to the separating hyperplane.

**Proof.** Suppose that we wish to classify two linearly separable classes \( \zeta_1 \) and \( \zeta_2 \) belonging to dataset \( \zeta \) and that the initial weight \( w_0 = 0 \). For patterns \( x_1, x_2, \ldots, x_t \) with labels \( y_1, y_2, \ldots, y_t \in \{-1, +1\} \) the updated weight \( w_{t+1} \) is given by

\[ w_{t+1} = w_t + y_t x_t. \] (B.1)

Since classes \( \zeta_1 \) and \( \zeta_2 \) are linearly separable, there is a solution \( \omega \) that correctly classifies all patterns. Thus, for a fixed solution \( \omega \) we can define a positive number \( \alpha \) according to

\[ \alpha = \min_{x \in \zeta} \| \omega^T x \|. \] (B.2)

Multiplying both sides of (B.1) by \( \omega^T \), we have

\[ \omega^T w_{t+1} = \omega^T x_1 y_1 + \omega^T x_2 y_2 + \cdots + \omega^T x_t y_t. \] (B.3)

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

\[ \omega^T w_{t+1} \geq t \alpha. \] (B.4)

Using now Cauchy-Schwarz inequality [43] we reach

\[ \| \omega^T \|^2 \| w_{t+1} \|^2 \geq \| \omega^T w_{t+1} \|^2. \] (B.5)

Observing (B.4) we can conclude that \( \| \omega^T w_{t+1} \|^2 \geq t^2 \alpha^2 \), or equivalently

\[ \| w_{t+1} \|^2 \geq \frac{t^2 \alpha^2}{\| \omega^T \|^2}. \] (B.6)

We can also write (B.1) as \( w_{k+1} = w_k + y_k x_k \), where \( k = 1, \ldots, t \). Taking the square of the Euclidean norm on both sides of this equation we obtain

\[ \| w_{k+1} \|^2 = \| w_k \|^2 + \| y_k x_k \|^2 + 2 \text{Re} \langle w_k, y_k x_k \rangle. \] (B.7)

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

\[ \| w_{k+1} \|^2 = \| w_k \|^2 + \| y_k x_k \|^2 + 2 \text{Re} \langle w_k, y_k x_k \rangle. \] (B.8)

Summing up the terms \( k = 1, \ldots, t \) of (B.8) and considering the initial condition \( w_0 = 0 \) we have

\[ \| w_{t+1} \|^2 = \sum_{k=1}^{t} \| y_k x_k \|^2 + \sum_{k=1}^{t} \| w_k \|^2. \] (B.9)

Since \( \| y_k x_k \|^2 = \| x_k \|^2 \), regardless of the value of \( y_k \), we can define

\[ \beta = \max_{x \in \zeta} \| x \|, \]

\[ \theta = \max_{x \in \zeta} \| w^T x \| = \max_{x \in \zeta} \| w^T x \|. \] (B.10)
Using the definitions of (B.10), we can ensure that the following inequality is true:
\[ \|w_{t+1}\|^2 \leq t\beta + 2t\theta. \] (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 \( t_{\text{max}} \) where (B.6) and (B.11) are equal:
\[ \frac{t_{\text{max}}^2\alpha^2}{\|w^*\|^2} = \frac{\beta + 2\theta}{\alpha^2}. \] (B.12)

Thus the value of \( t_{\text{max}} \) can be obtained by
\[ t_{\text{max}} = \frac{\beta + 2\theta}{\alpha^2}. \] (B.13)

For normalized weights [23] we have \( \|w^*\| = 1 \) and (B.13) is rewritten as
\[ t_{\text{max}} = \frac{\beta + 2\theta}{\alpha^2}. \] (B.14)

\[ \Box \]

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


Submit your manuscripts at http://www.hindawi.com