Feature Selection using Associative Memory Paradigm and Parallel Computing

Mario Aldape-Pérez[1,2], Cornelio Yáñez-Márquez[1], Oscar Camacho-Nieto[1], and Ángel Ferreira-Santiago[2]

[1] Instituto Politécnico Nacional (CIC), Distrito Federal, Mexico

[2] Instituto Politécnico Nacional (ESCOM), Distrito Federal, Mexico

www.aldape.mx, {cyanez, oscarc}@cic.ipn.mx, www.cornelio.org.mx

Abstract. Performance of most pattern classifiers is improved when redundant or irrelevant features are removed. Nevertheless, this is mainly achieved by highly demanding computational methods or successive classifiers' construction. This paper shows how the associative memory paradigm and parallel computing can be used to perform Feature Selection tasks. This approach uses associative memories in order to get a mask value which represents a subset of features which clearly identifies irrelevant or redundant information for classification purposes. The performance of the proposed associative memory algorithm is validated by comparing classification accuracy of the suggested model against the performance achieved by other well-known algorithms. Experimental results show that associative memories can be implemented in parallel computing infrastructure, reducing the computational costs needed to find an optimal subset of features which maximizes classification performance.

Keywords. Feature selection, associative memory, pattern classification.

Selección de características utilizando el paradigma de memoria asociativa y computación paralela

Resumen. El rendimiento en la mayoría de los clasificadores de patrones se mejora cuando las características redundantes o irrelevantes son eliminadas. Sin embargo, esto se logra a través de la construcción de clasificadores sucesivos o mediante algoritmos iterativos que implican altos costos computacionales. Este trabajo muestra la aplicación del paradigma de memoria asociativa y la computación paralela para realizar tareas de selección de características. Este enfoque utiliza las memorias asociativas para obtener el valor de una máscara que identifica claramente la información irrelevante o redundante para fines de clasificación. El desempeño del algoritmo propuesto es validado a través de la comparación de la precisión predictiva alcanzada por este modelo contra el desempeño alcanzado por otros algoritmos reconocidos en la literatura actual. Los resultados experimentales muestran que las memorias asociativas pueden ser implementadas en infraestructura de cómputo paralelo, reduciendo los costos computacionales necesarios para encontrar el subconjunto óptimo de características de maximiza el desempeño de clasificación.

Palabras clave. Selección de características, memorias asociativas, clasificación de patrones.

1 Introduction

Pattern recognition has existed for many years in a wide range of human activity. However, the general pattern recognition problem can be stated in the following form: given a collection of objects belonging to a predefined set of classes and a set of measurements on these objects, identify the membership class of each object by an appropriate analysis of the measurements (features).

Although features are functions of measurements performed on a class of objects, in most cases the initial set of features consists of a large number of potential attributes which constitute an obstacle not only to the accuracy but to the efficiency of algorithms.

In countless situations, it is a complicated task to find proper features for all patterns in a class; therefore, many machine learning algorithms have been used as tools to identify relevant information for classification purposes.

Support vector machines (SVMs) select a small number of critical boundary samples from each class and build a linear discriminant function which separates them as widely as possible. The main reason whereby SVMs are not commonly used for feature selection results from the fact that SVMs can perform badly in a situation of many irrelevant features [1].

Another approach is to apply different degrees of relevance to information; feature weighting schemes tend to be easier to implement when the obtained subset of “most relevant” features is fed into another algorithm capable of making additive changes to all weights. The main disadvantage of these schemes is that convergence in the learning phase is not guaranteed; moreover, weighting techniques may have difficulties when irrelevant features are present [2]. In order to overcome this limitation, multiclassifier approach arises [3, 4]; nonetheless, these methodologies lack criterions that help to ignore redundant information.

Although there is a large number of algorithms used to classify patterns, most of them cannot obtain an optimal solution to maximize classification accuracy, nor to reduce the dimensionality of the dataset [5]. There are algorithms capable to obtain an optimal subset of features which maximizes classification accuracy. They fulfill this task by evaluating classification accuracy achieved by the algorithm with each of the possible subsets of features. This accuracy assessment involves prohibitive computational costs for non-parallelizable algorithms.

Some remarkable things to mention about Associative Memories (AM) is that they are represented as matrices, and duration of the learning phase depends only on the number of patterns used to train the associative memory [6]; as a consequence, convergence in the learning phase is guaranteed [7, 9].

After carrying out the learning phase, the knowledge is stored in a numeric array which can be distributed and handled in parallel on multiple processing nodes.

Each of the available processing nodes in a computing cluster can evaluate the classification performance using different subsets of features. Thereby, the task of finding the best subset of features can be distributed among the available processors on each processing node in the computing cluster. As a result, the subset of features which maximizes classification accuracy can be found in a reasonable time.

In this paper, associative memories and parallel computing are used to identify the relevance of the information in some widely used datasets. By removing irrelevant information, dimensionality of the problem is reduced and classification performance is improved.

The paper is organized as follows. In Section 2 a succinct description of Associative Memories fundamentals is presented. Section 3 introduces the topic of Feature Selection and presents the procedure used to reduce the dimensionality of the datasets. Section 4 outlines how to implement in parallel the proposed algorithm. Section 5 describes how the experimental phase was conducted. Section 6 presents classification accuracy results achieved by each one of the compared algorithms in six different pattern classification problems. Finally, some conclusions are discussed in Section 7.

2 Associative Memories

Early models of learning matrices appeared more than four decades ago [6, 9], and since then associative memories have attracted the attention of major research groups worldwide. From a connectionist model perspective, an associative memory can be considered a special case of the neural computing approach for pattern recognition [10].

An associative memory M is a system that relates input patterns and output patterns. Input patterns are represented by a column vector, denoted by x, while output patterns are represented by a column vector, denoted by y. Each input vector forms an association with its corresponding output vector. For each positive integer k, the corresponding association is denoted as $\left(x^k, y^k\right)$.

An associative memory M is represented by a matrix whose ij -th component is m_{ij}. Associative memory M is generated from an *a priori* finite set of known associations, called the fundamental set of associations. The fundamental set is represented as $\left\{\left(x^{\mu}, y^{\mu}\right) \mid \mu=1,2,...,p\right\}$ with p as the set cardinality.

Let n and m be the dimensions of the input patterns and output patterns, where $x^{\mu} \in A^{n}$, $y^{\mu} \in A^{m}$ with $A=\{0,1\}$. The patterns that form the fundamental set are called fundamental patterns. If it holds that $x^{\mu} = y^{\mu} \quad \forall \mu \in \{1,2,...,p\}$, M is autoassociative, otherwise it is heteroassociative; in this case, it is possible to establish that $\exists \mu \in \{1,2,...,p\}$ for which $x^{\mu} \neq y^{\mu}$.

If we consider the fundamental set of patterns $\left\{\left(x^{\mu}, y^{\mu}\right) \mid \mu=1,2,...,p\right\}$, where n and m are the dimensions of the input patterns and output patterns, respectively, it is said that $x^{\mu} \in A^{n}$ and $y^{\mu} \in A^{m}$, where $A=\{0,1\}$, then the j -th component of an input pattern $x^{\mu} \in A^{n}$ is $x_{j}^{\mu} \in A$.

Analogously, the i -th component of an output pattern $y^{\mu} \in A^{m}$ is represented as $y_{i}^{\mu} \in A$.

Therefore, the fundamental input and output patterns are represented as follows:

$$x^{\mu}=\begin{pmatrix} x_1^{\mu} \\ x_2^{\mu} \\ \vdots \\ x_n^{\mu} \end{pmatrix} \in A^{n} \qquad y^{\mu}=\begin{pmatrix} y_1^{\mu} \\ y_2^{\mu} \\ \vdots \\ y_m^{\mu} \end{pmatrix} \in A^{m}$$

A distorted version of a pattern x^{k} to be recalled is denoted as $\tilde{x}^{k}$. An unknown input pattern to be recalled is denoted as x^{ω}. If when an unknown input pattern x^{ω} with $\omega \in \{1,2,...,k,...,p\}$ is fed to an associative memory M , it happens that the output corresponds exactly to the associated pattern y^{ω}, it is said that recalling is correct.

3 Application to Feature Selection

The task of classification occurs in a wide range of human activity; however, satisfactory results depend on the amount of relevant information obtained when coherent features are selected.

Feature Selection is focused on finding a set of characteristics that best describes a hypothesis.

The number of features delimits the size of the hypothesis space containing all hypotheses that can be learned from data [11]. A hypothesis is a function that predicts classes based on given data. The linear increase of the number of features implies an exponential increase of the hypothesis space [12].

This section is divided into three parts; each one addresses one stage of the proposed algorithm. The first procedure describes associative memory training. The second describes the steps involved in the classification phase. The third explains the procedure of feature selection.

3.1 Learning Phase

The task of this phase is to find adequate operators and a way to generate an associative memory M which will store the p associations of the fundamental set.

An associative memory M is obtained by performing the following two steps:

1. Consider each one of the p associations $\left(x^{\mu}, y^{\mu}\right)$, so an m by n matrix is obtained according to the following expression:

$$y^{\mu} \cdot \left(x^{\mu}\right)^{t} = \begin{pmatrix} y_1^{\mu}x_1^{\mu} & \cdots & y_1^{\mu}x_j^{\mu} & \cdots & y_1^{\mu}x_n^{\mu} \\ \vdots & & \vdots & & \vdots \\ y_i^{\mu}x_1^{\mu} & \cdots & y_i^{\mu}x_j^{\mu} & \cdots & y_i^{\mu}x_n^{\mu} \\ \vdots & & \vdots & & \vdots \\ y_m^{\mu}x_1^{\mu} & \cdots & y_m^{\mu}x_j^{\mu} & \cdots & y_m^{\mu}x_n^{\mu} \end{pmatrix} \quad (1)$$

2. An associative memory M is obtained by adding all the p matrices:

$$M = \sum_{\mu=1}^{p} y^{\mu} \cdot \left(x^{\mu}\right)^{t} = \left[m_{ij}\right]_{mxn} \tag{2}$$

In this way the ij-th component of an associative memory M is expressed as follows:

$$m_{ij} = \sum_{\mu=1}^{p} y_i^{\mu} x_j^{\mu} \tag{3}$$

3.2 Classification Phase

This phase consists of finding the class to which an unknown input pattern $x^{\omega} \in A^n$ belongs. Finding the class means getting $y^{\omega} \in A^m$ which corresponds to x^{ω}.

Classification phase is done by operating an associative memory M with an unknown input pattern x^{ω}, where $\omega \in \{1, 2, ..., k, ..., p\}$. $M \cdot x^{\omega}$ is operated as follows:

$$M \cdot x^{\omega} = \left[\sum_{\mu=1}^{p} y^{\mu} \cdot \left(x^{\mu}\right)^{t}\right] \cdot x^{\omega} \quad . \tag{4}$$

Let's expand Expression 4 this way:

$$M \cdot x^{\omega} = y^{\omega} \cdot \left[\left(x^{\omega}\right)^{t} \cdot x^{\omega}\right] + \sum_{\mu \neq \omega} y^{\mu} \cdot \left[\left(x^{\mu}\right)^{t} \cdot x^{\omega}\right] \quad . \tag{5}$$

Expression 5 lets us know which restrictions have to be observed, thus correct recalling is achieved. This is expressed as:

$$\left(x^{\mu}\right)^{t} \cdot x^{\omega} = \begin{cases} 1 & if \quad \mu = \omega \\ 0 & if \quad \mu \neq \omega \end{cases} \quad . \tag{6}$$

If the condition in Expression 6 is met, then a correct recalling is expected. Therefore, Expression 5 is given as

$$M \cdot x^{\omega} = y^{\omega}. \tag{7}$$

Once it is already known under which conditions it is possible to successfully recover a pattern, it is likely to state the classification rule, where $\vee$ is the maximum operator:

$$y_i^{\omega} = \begin{cases} 1 & if \quad \sum_{j=1}^{n} m_{ij} \cdot x_j^{\omega} = \vee_{h=1}^{m} \left[\sum_{j=1}^{n} m_{hj} \cdot x_j^{\omega}\right] \\ 0 & otherwise \end{cases} \tag{8}$$

Expression 8 allows obtaining each of the components of the vector representing the class label.

Performance of the classification phase is measured in terms of error rate; so, classifier accuracy represents the correct classification rate when unseen patterns are presented.

3.3 Selection of Relevant Features

This section introduces the procedure that implements an associative memory to identify and preserve relevant features for classification purposes.

Definition 3.1: the r-th masking vector. Let $A = \{0,1\}$, let f be the number of features in the original set of data, and let r be a positive integer such that $r \in \{1, 2, ..., (2^f - 1)\}$. The r-th masking vector of size n is then defined to be

$$e^r = \begin{pmatrix} e_1^r \\ e_2^r \\ \vdots \\ e_n^r \end{pmatrix} \in A^n \quad . \tag{9}$$

Definition 3.2: *IntegerToVector* operator. Let $A = \{0,1\}$, let n be the dimension of an input pattern and let r be a positive integer. The *IntegerToVector* operator, takes r as input and returns a column vector e^r with r value expressed in its binary representation. Note that e_1^r is the Most Significant Bit (MSB), while e_n^r is the Least Significant Bit (LSB).

Example 3.1. Let $A = \{0,1\}$, let $n = 4$ and let $r = 11$. The *IntegerToVector* operator is applied to obtain the r-th masking vector as stated in Definition 3.2.

To convert an integer to its binary representation, we divide r by two repeatedly, until the final remainder is zero. If we take only the remainder of each division, then $r = 11$ can be expressed as

$$r = 11 = \left[(\boxed{1} \text{x} 2^3) + (\boxed{0} \text{x} 2^2) + (\boxed{1} \text{x} 2^1) + (\boxed{1} \text{x} 2^0) \right]$$

If we apply the *IntegerToVector* operator on r, we obtain the r-th masking vector e^r.

$$e^r = IntegerToVector(r) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

In summary, the *IntegerToVector* operator, helps us to obtain a column vector e^r with r expressed in its binary representation.

Let's take classification rule presented in Expression 8 and incorporate the r-th masking vector presented in Expression 9. The resulting expression is as follows:

$$y_i^{\omega} = \begin{cases} 1 & if \quad \sum_{j=1}^{n} m_{ij} \cdot \left(x_j^{\omega} \cdot e^r \right) = \theta \\ 0 & otherwise \end{cases}, \tag{10}$$

where θ represents the maximum threshold value and $\vee$ is the maximum operator.

$$\theta = \vee_{h=1}^{m} \left[\sum_{j=1}^{n} m_{hj} \cdot \left(x_j^{\omega} \cdot e^r \right) \right]. \tag{11}$$

Expression 10 classifies an unknown input pattern using the r-th masking vector. This expression allows us to estimate classification performance of the associative memory, using different sets of features, without having to perform the training phase repeatedly. By not having to perform the training phase repeatedly, the computational costs are reduced, and it is possible to search for an optimal subset of features that maximizes classification performance.

3.3.1 Feature Selection Procedure

1. Let n be the dimension of each input pattern in the fundamental set, grouped in m different classes.
2. Each one of the input patterns belongs to a k class, $k \in \{1, 2, \ldots, m\}$, represented by a column vector whose components are assigned by $y_k^{\mu} = 1$, so $y_j^{\mu} = 0$ for $j = 1, 2, \ldots, k-1, k+1, \ldots, m$.
3. Create a classifier using Expressions 1, 2 and 3.
4. Use the *IntegerToVector* operator to obtain the r-th masking vector, as stated in Definition 3.2.
5. The classification phase is carried out according to Expression 10 so an r-th classification accuracy parameter is obtained.
6. Store the r-th classification accuracy parameter and the r-th masking vector.
7. Compare the r-th classification accuracy parameter with the $(r-1)$-th classification accuracy parameter. The best classification accuracy value is stored.
8. Finalize when classification performance has been estimated using all possible masking vectors.

3.3.2 Time Complexity Analysis

It is generally accepted that an algorithm provides a satisfactory solution when it produces a correct answer efficiently. The efficiency of an algorithm can be estimated by measuring the time required by the computer to solve a problem using a given algorithm.

The worst-case time complexity of an algorithm is defined as a function of the size of the input. For a given input size, the worst-case time complexity is the maximal number of execution steps needed for executing the program on arbitrary input of that size.

Operations used to measure time complexity can be single-precision floating point comparison, single-precision floating point addition, single-precision floating point division, variable assignation, logical comparison, or any other elemental operation. Listing 1 shows the implementation of the feature selection process.

In order to analyze the time complexity of the feature selection process, the following is defined:

- EO: elemental operation.
- n: dimension of input patterns.
- p: cardinality of the fundamental set.

Listing 1. Feature Selection Procedure

```
1:  r_max=(2^(n));
2:  for r=1:r_max-1
3:   class_hit=0;
4:   class_miss=0;
5:   e_r = IntegerToVector(r);
6:   for i=1:p
7:    y_mu_1=sum(x_mu(i) .* e_r .* M(1));
8:    y_mu_2=sum(x_mu(i) .* e_r .* M(2));
9:    if y_mu_1>y_mu_2
10:       class_label=class_1;
11:   else
12:       class_label=class_2;
13:   end
14:   if class_label==x_mu(i,n)
15:       class_hit=class_hit+1;
16:   else
17:       class_miss=class_miss+1;
18:   end
19:   end
20:  end
```

Table 1 shows the number of elemental operations required to execute each of the lines of code shown in Listing 1. The code segment that represents the largest number of operations is in line 7 and 8. Executing these two lines of code, the class label is recovered. Line 7 retrieves the first component of the class label y_1^{ω}, while line 8 retrieves the second component y_2^{ω}.

The total number of Elemental Operations is as follows:

$$Total\_EOs = 1 + 3(-1+(2^n)) + (-1+(2^n))n + 7(-1+(2^n))p + 6(-1+(2^n))np$$

By grouping some terms, we obtain the formula:

$$Total\_EOs = -2 + 3(2^n) + 7(-1+(2^n))p + (-1+(2^n))n(1+6p)$$

Table 1. Time Complexity Analysis

1:	1 EO, assignation
2:	max_iter EO, comparison
3:	max_iter EO, assignation
4:	max_iter EO, assignation
5:	max_iter*n EO, comparison
6:	max_iter*p EO, comparison
7a:	max_iter*n*p EO, multiplication
7b:	max_iter*n*p EO, multiplication
7c:	max_iter*n*p EO, addition
7d:	max_iter*p EO, assignation
8a:	max_iter*n*p EO, multiplication
8b:	max_iter*n*p EO, multiplication
8c:	max_iter*n*p EO, addition
8d:	max_iter*p EO, assignation
9:	max_iter*p EO, comparison
10:	max_iter*p EO, assignation
14:	max_iter*p EO, comparison
15:	max_iter*p EO, assignation

If we factor some terms, we get the following:

$$Total\_EOs = -2 + 3(2^n) - n + (2^n)n - 7p + 7(2^n)p - 6np + 3(2^{n+1})np$$

Finally, the equation of the total number of Elemental Operations can be written as

$$Total\_EOs = -2 - 7p + (-1+(2^n))n(1+6p) + (2^n)(3+7p)$$

The growth of time and space complexity with increasing input size n is a suitable measure of the efficiency of the algorithm. To obtain an estimate of the complexity of the algorithm when it is applied to a known test set, we chose the dataset with the largest number of features, which is the Hepatitis disease dataset. As it is shown in Table 2, each of the 155 instances has 19 features and a class label. The number of fundamental input patterns is $p = 155$.

The growth of functions is usually described using the big-O notation [13].

Definition 3.3. Let f and g be functions from the integers or the real numbers to the real numbers. We say that $f(n)$ is $O(g(n))$ if there are constants C and k such that $|f(n)| \leq C|g(n)|$ whenever $n > k$.

The total number of Elemental Operations can be computed as

$Total\_EOs = 1 + 1088(-1 + (2^n)) + 931(-1 + (2^n))n$.

A function $g(n)$ and constants C and k must be found, such that the inequality holds. We propose the following $g(n)$:

$1(2^n) - 1088(2^n) + 1088(2^n) - 931n(2^n) + 931n(2^n)$

Then, if $g(n) = 2^n, C = 20000$ and $k = 1$, we have that $|f(n)| \leq 20000|g(n)|$ whenever $n > 1$.

Therefore, $f(n)$ is $O(2^n)$.

4 Parallel Implementation

Once we know the procedure that must be carried out to find an optimal subset of features, we can reduce computational costs by distributing this procedure in parallel computing infrastructure.

To accomplish this, we need to assign each node a search range and a copy of the associative memory which results from the learning phase. After completing a search range, each node delivers the masking vector with which the associative memory achieved the best classification rate in such search range. After all nodes have completed the search task, the master node retains the masking vector which maximizes classification accuracy. In the event that several masking vectors produce the best classification performance, the masking vector that represents the smallest subset is retained.

5 Experimental Phase

Throughout the experimental phase, six datasets were used as the test set to estimate classification performance of each one of the compared algorithms. These datasets were taken from the UCI machine learning repository from which full documentation for all datasets can be obtained. The main characteristics of these datasets are shown in Table 2.

The performance of the proposed algorithm was compared against the performance achieved by the twenty best-performing algorithms of the seventy six algorithms available in WEKA 3 Data Mining Software in Java [14]. WEKA is an open source software issued under the GNU General Public License: Further information on each of the algorithms used during the experimental phase can be found in [15]. In order to carry out such a comparison, we applied the same conditions and validation schemes for each experiment. Classification accuracy of each one of the compared algorithms was calculated using 10-fold cross-validation. The proposed algorithm was parallelized using an MPI implementation for the Java programming language [16] and tested on an 8-machine cluster. Each machine has a dual-core Intel Xeon CPU running at 2 GHz.

Table 2. Characteristics of datasets

Dataset	Instances	Attributes
1. Haberman	306	3
2. Liver	345	6
3. Inflammation	120	6
4. Breast	699	9
5. Heart	270	13
6. Hepatitis	155	19

6 Results and Discussion

Figures 1-6 show the running time of the feature selection process using 10-fold cross-validation. The proposed algorithm was tested in configurations ranging from two to eight simultaneous processors.

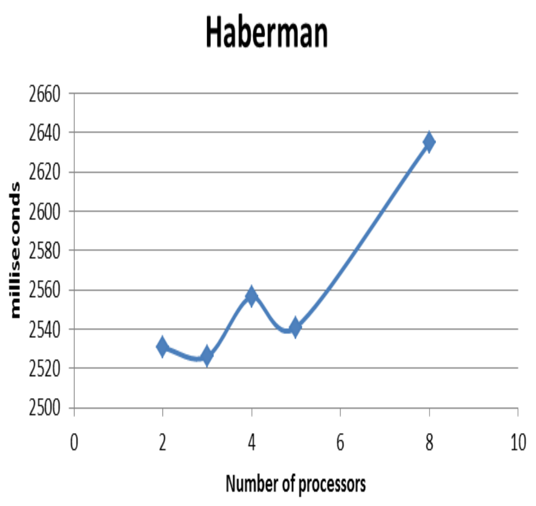

Fig. 1. Running time using 10-fold cross-validation

It can be seen in Figure 1 that when the search process is distributed over a larger number of processors, the execution time increases. This result is contrary to the expected, since if the search task is distributed among a larger number of processors, the execution time is expected to decrease. This unwanted behavior occurs because the number of features of the Haberman dataset is too small to be explored by eight processors simultaneously. The increase in running time (108.63 milliseconds) is due to redundant message passing between the master node and the computing nodes.

Figures 2-6 show the running time of the feature selection process for the remaining datasets. It can be seen that if the search task is distributed among a larger number of processors, the execution time decreases. Table 3 shows the running time performance of the proposed algorithm using all data sets. The running time performance observed in Table 3 corresponds to the expected behavior. The larger the number of processors involved in the feature selection process is, the shorter is the time to find the optimal subset of features.

Table 4 shows classification accuracy results achieved by each one of the compared algorithms in six different pattern classification problems, using 10-fold cross-validation.

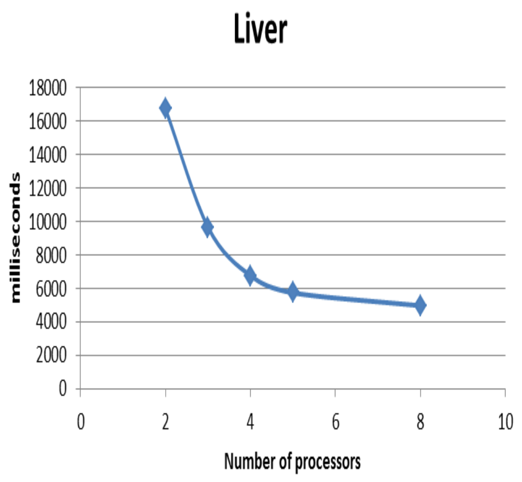

Fig. 2. Running time using 10-fold cross-validation

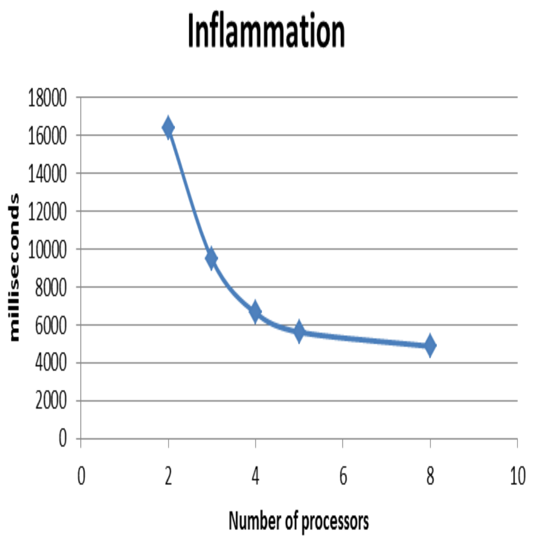

Fig. 3. Running time using 10-fold cross-validation

Although WEKA 3 Data Mining Software in Java [15] has more than seventy well known algorithms implemented, only the twenty best-performing algorithms were considered for comparison purposes. According to the type of learning scheme, each of these can be grouped in one of the following types of classifiers:

Table 3. Running time of the proposed algorithm using 10-fold cross-validation

Number of processors	Haberman (milliseconds)	Liver (milliseconds)	Inflammation (milliseconds)	Breast (seconds)	Heart (seconds)	Hepatitis (minutes)
2	2530.59	16730.27	16395.66	1184.71	1773.29	404.20
3	2526.19	9654.51	9461.41	590.66	907.27	205.86
4	2556.62	6763.20	6627.93	412.13	631.25	145.73
5	2541.72	5736.41	5621.68	324.29	510.61	115.70
8	2634.82	4965.90	4866.58	237.26	393.05	92.79

Table 4. Classification accuracy using 10-fold cross-validation

Algorithm	Haberman	Liver	Inflammation	Breast	Heart	Hepatitis
1. AdaBoostM1	73.20	66.66	100.00	95.60	82.22	67.09
2. Bagging	73.20	73.04	100.00	96.19	83.70	69.67
3. BayesNet	72.54	56.81	100.00	97.21	82.22	69.03
4. Dagging	73.52	57.97	100.00	96.77	82.22	66.45
5. DecisionTable	72.54	57.97	100.00	95.75	83.33	72.25
6. DTNB	72.54	57.97	100.00	97.51	82.59	68.38
7. FT	72.87	70.43	100.00	96.92	82.22	69.03
8. LMT	73.85	69.85	100.00	96.48	82.22	67.09
9. Logistic	74.50	68.69	100.00	96.63	83.70	68.38
10. MultiClassClassifier	74.50	68.69	100.00	96.63	83.70	68.38
11. NaiveBayes	74.50	54.20	95.83	96.19	83.33	71.61
12. NaiveBayesSimple	73.85	55.07	95.83	96.33	82.96	70.96
13. NveBayesUpdateable	74.50	54.20	95.83	96.19	83.33	71.61
14. RandomCommittee	64.37	68.11	100.00	96.48	82.22	63.22
15. RandomForest	67.97	70.72	100.00	97.07	83.70	65.16
16. RandomSubSpace	72.22	64.05	100.00	95.54	82.22	67.74
17. RBFNetwork	72.87	66.08	100.00	95.90	84.07	69.67
18. RotationForest	73.20	73.04	100.00	97.21	82.59	66.45
19. SimpleLogistic	73.85	71.01	100.00	96.63	82.22	66.45
20. SMO	73.52	57.97	100.00	96.92	83.33	72.25
* Our proposal	76.33	65.50	100.00	97.80	83.70	85.16

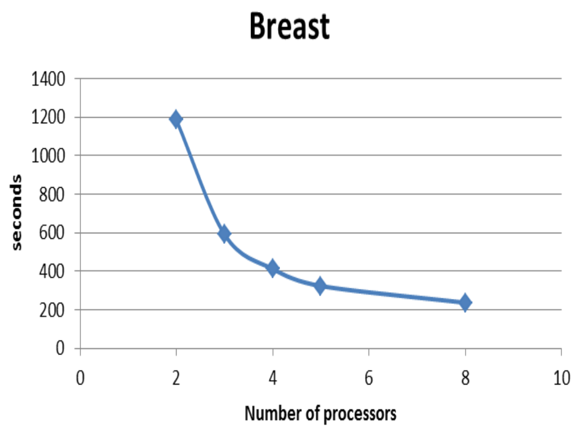

Fig. 4. Running time using 10-fold cross-validation

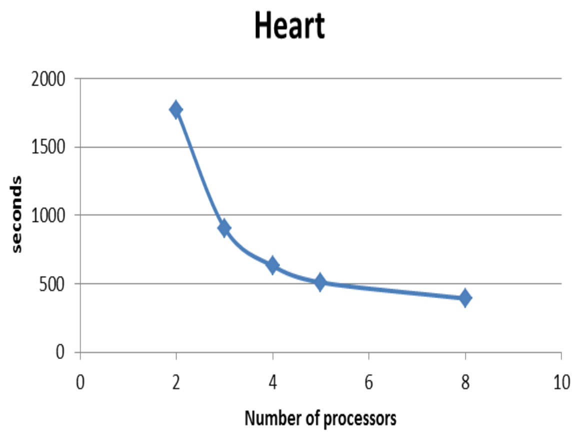

Fig. 5. Running time using 10-fold cross-validation

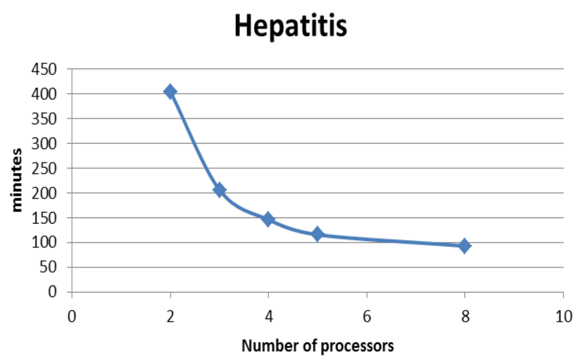

Fig. 6. Running time using 10-fold cross-validation

Bayesian classifiers, Function-based classifiers, Meta classifiers, Rule-based classifiers and Decision Tree classifiers. The twenty best-performing algorithms are as follows:

- Four algorithms based on the Bayesian approach (BayesNet, NaiveBayes, NaiveBayesSimple and Naive-BayesUpdateable).
- Four function-based classifiers (Logistic, RBFNetwork, SimpleLogistic and SMO).
- Seven metaclassifiers (AdaBoostM1, Bagging, Dagging, MultiClassClassifier, RandomCommittee, RandomSubSpace, RotationForest).
- Two rule-based classifiers (DecisionTable and DTNB).
- Three decision tree classifiers (FT, LMT, Random-Forest).

Classification results are as follows: two of the seven metaclassifiers (Bagging and RotationForest]) achieved the best performance in two of the six pattern classification problems. Three decision trees classifiers (FT, LMT and Random-Forest) achieved the best performance in one of the six datasets. Likewise, one of the four algorithms based on the Bayesian approach (BayesNet) achieved the best performance in one of the six datasets. Two rule-based classifiers (DecisionTable and DTNB) achieved the best performance in one of the six datasets. Similarly, one of the four function-based classifiers (RBFNetwork) achieved the best performance in two of the six datasets.

It is worth noting that our proposal achieved the best performance in four of the six pattern classification problems. As it is shown in Table 4, there is no particular method which surpasses all the other algorithms in all sorts of problems. This should not be surprising since Wolpert and Macready [17] demonstrated that what an algorithm gains in performance on one class of problems is necessarily offset by its performance on the remaining problems.

Table 5 shows the classification performance achieved by the associative memory using the full set of features against the classification performance achieved by the associative memory using the optimal subset of features.

Table 5. Classification accuracy using the full set of features and the Feature Selection procedure

Dataset	Full Set	Feature Selection
1. Haberman	66.34	**76.33**
2. Liver	55.36	65.50
3. Inflammation	97.50	**100.00**
4. Breast	97.51	**97.80**
5. Heart	64.07	83.70
6. Hepatitis	66.45	**85.16**

In all data sets, the feature selection procedure found an optimal subset of features which increases classification performance of the associative memory.

7 Conclusions

In this paper, a novel approach to perform Feature Selection tasks using Associative Memory Paradigm and Parallel Computing is presented. Throughout the experimental phase, six datasets were used as test set to estimate classification performance of each one of the compared algorithms. Experimental results show that associative memories can be implemented in parallel computing infrastructure, reducing the computational costs needed to find an optimal subset of features which maximizes classification performance.

As a result of analyzing the execution times used by the proposed algorithm to find an optimal subset of features, we conclude that the feature selection procedure in an associative memory is fully parallelizable. Consequently, this permits to perform feature selection tasks in larger datasets using the proposed algorithm.

Acknowledgements

The authors wish to thank the following institutions for their support to develop the present work: Science and Technology National Council of Mexico (CONACyT), National System of Researchers of Mexico (SNI), National Polytechnic Institute of Mexico (IPN, Project No. SIP-IPN 20121556) and the Institute of Science and Technology of the Federal District (ICyT DF, Project No. PIUTE10-77).

References

1. **Maldonado, S., Weber, R., & Basak, J. (2011).** Simultaneous feature selection and classification using kernel-penalized support vector machines. *Information Sciences: An International Journal*, 181(1), 115–128.
2. **Youn, E., Koenig, L., Jeong, M.K., & Baek, S.H. (2010).** Support vector-based feature selection using Fisher's linear discriminant and Support Vector Machine. *Expert Systems with Applications: An International Journal*, 37(9), 6148–6156.
3. **Lan, Y., Soh, Y.C., & Huang, G.B. (2009).** Ensemble of online sequential extreme learning machine. *Neurocomputing*, 72(13-15), 3391–3395.
4. **Kittler, J., Hatef, M., Duin, R.P.W., & Matas, J. (1998).** On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3), 226–239.
5. **Ozcift, A. & Gulten, A. (2011).** Classifier ensemble construction with rotation forest to improve medical diagnosis performance of machine learning algorithms. *Computer Methods and Programs in Biomedicine*, 104(3), 443–451.
6. **Steinbuch, K. (1961).** Die lernmatrix. *Kybernetik*, 1(1), 36–45.
7. **Steinbuch, K. (1964).** Adaptive networks using learning matrices. *Kybernetik*, 2(4), 148–152.
8. **Kohonen, T. (1972).** Correlation Matrix Memories. *IEEE Transactions on Computers*, C-21(4), 353–359.
9. **Acevedo-Mosqueda, M.E., Yáñez-Márquez, C., & López-Yáñez, I. (2007).** Alpha-beta bidirectional associative memories: theory and applications. *Neural Processing Letters*, 26(1), 1–40.
10. **Sacramento, J. & Wichert, A. (2011).** Tree-like hierarchical associative memory structures. *Neural Networks*, 24(2), 143–147.
11. **Kohavi, R. & John, G. H. (1997).** Wrappers for Feature Subset Selection. *Artificial Intelligence - Special issue on relevance*, 97(1-2), 273–324.
12. **Dornaika, F., Lazkano, E., & Sierra, B. (2011).** Improving dynamic facial expression recognition

with feature subset selection. *Pattern Recognition Letters*, 32(5), 740–748.

13. **Rosen, K. H. (2007).** *Discrete Mathematics and Its Applications* (6th ed.), Boston: McGraw-Hill Higher Education.
14. **Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I.H. (2009).** The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1), 10–18.
15. **Witten, I.H. & Frank, E. (2005).** *Data Mining: Practical Machine Learning Tools and Techniques* (2nd ed.), Amsterdam; Boston, MA: Morgan Kaufmann.
16. **Shafi, A., Carpenter, B., & Baker, M. (2009).** Nested parallelism for multi-core HPC systems using Java. *Journal of Parallel and Distributed Computing*, 69(6), 532–545.
17. **Wolpert, D.H. & Macready, W.G. (1997).** No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.

Mario Aldape-Pérez received his Ph.D. in Computer Science from the National Polytechnic Institute of Mexico, in 2011. He is a Professor of Computer Architecture at the Superior School of Computing of the National Polytechnic Institute of Mexico. His current research interests include Associative Memories, Soft Computing and FPGA Implementation of High Performance Pattern Classification algorithms. Dr. Aldape-Pérez is a member of the IEEE Computer Society and the ACM. He is currently the President of the Academy of Digital Systems at the School of Computing of the National Polytechnic Institute of Mexico.

Cornelio Yáñez-Márquez received his Ph.D. degree in Computer Science from the National Polytechnic Institute of Mexico in 2002. Since then, he has been with the Center for Computing Research (CIC-IPN), Mexico, where he is currently the Head of the Laboratory of Unconventional Computing and Neural Networks. His research interests cover Neural Networks and Unconventional Computing, Associative Memory, Pattern Recognition, Mathematical Morphology and Soft Computing, with over 100 technical publications. Dr. Yáñez-Márquez is the leader and founder of Alpha-Beta Research.

Oscar Camacho-Nieto received the M.Sc. degree in Computer Engineering and the Ph.D. degree in Computer Science from the National Polytechnic Institute of Mexico, in 1995 and 2003, respectively. He had been a Senior Lecturer at the Center for Computing Research (CIC-IPN), Mexico, for several years. He is currently the Director of the Center for Innovation and Technology Development in Computing (CIDETEC-IPN), Mexico.

Angel Ferreira-Santiago is a Computer Systems Engineer from the National Polytechnic Institute. Since 2012 he has been working in the parallel implementation of supervised learning algorithms. His research interests cover Associative Memories, Feature Selection and Space-Time Analysis of Video.

Article received on 12/10/2012; accepted on 18/12/2012.