electronics

MDPI

Article

Improving the Performance of an Associative Classifier in the Context of Class-Imbalanced Classification

Carlos Alberto Rolón-González [1], Rodrigo Castañón-Méndez [1], Antonio Alarcón-Paredes [1,*], Itzamá López-Yáñez [2,*] and Cornelio Yáñez-Márquez [1,*]

[1] Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico City 07700, Mexico; carlosagrolon@gmail.com (C.A.R.-G.); rcastanonmendez@gmail.com (R.C.-M.)
[2] Centro de Innovación y Desarrollo Tecnológico en Cómputo, Instituto Politécnico Nacional, Mexico City 07738, Mexico
* Correspondence: aalarcon@cic.ipn.mx (A.A.-P.); ilopezy@ipn.mx (I.L.-Y.); coryanez@gmail.com (C.Y.-M.)

Abstract: Class imbalance remains an open problem in pattern recognition, machine learning, and related fields. Many of the state-of-the-art classification algorithms tend to classify all unbalanced dataset patterns by assigning them to a majority class, thus failing to correctly classify a minority class. Associative memories are models used for pattern recall; however, they can also be employed for pattern classification. In this paper, a novel method for improving the classification performance of a hybrid associative classifier with translation (better known by its acronym in Spanish, CHAT) is presented. The extreme center points (ECP) method modifies the CHAT algorithm by exploring alternative vectors in a hyperspace for translating the training data, which is an inherent step of the original algorithm. We demonstrate the importance of our proposal by applying it to imbalanced datasets and comparing the performance to well-known classifiers by means of the balanced accuracy. The proposed method not only enhances the performance of the original CHAT algorithm, but it also outperforms state-of-the-art classifiers in four of the twelve analyzed datasets, making it a suitable algorithm for classification in imbalanced class scenarios.

Keywords: associative memories; class-imbalanced datasets; pattern classifier; Lernmatrix; linear associator

check for updates

Citation: Rolón-González, C.A.; Castañón-Méndez, R.; Alarcón-Paredes, A.; López-Yáñez, I.; Yáñez-Márquez, C. Improving the Performance of an Associative Classifier in the Context of Class-Imbalanced Classification. *Electronics* **2021**, *10*, 1095. https://doi.org/10.3390/electronics10091095

Academic Editor: Jorge Igual

Received: 31 March 2021
Accepted: 3 May 2021
Published: 6 May 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Classification and recall are two important tasks that are performed in the context of the supervised paradigm of pattern recognition. It is a fact that few methods recall effectively (and typically recall associative memories) [1]. On the other hand, approaches and methods to classify patterns, such as Bayes, k-nearest neighbor (k-NN) classification, regression trees (CART), neural networks, support vector machines (SVM), and deep learning methods, among many others, have proliferated, and further improved classification algorithms can be commonly found in specialized literature [2].

Thus, the number of pattern recognition applications has grown significantly in recent years, and new areas of application continually appear. In this context, every machine learning researcher who designs and creates a new pattern classifier algorithm hopes that the number of errors is as low as possible, and ideally the number of errors is zero, i.e., 100% performance; however, the proof of the no free lunch theorem precludes the existence of an ideal classifier. This very important theorem governs the effectiveness of all pattern classification algorithms [3,4].

For this reason, machine learning researchers no longer try to design zero-error algorithms because they know that this search is useless. Now, under this reality generated by the no free lunch theorem, researchers aim for errors to tend to zero when classifying patterns in datasets belonging to the different application areas. This is done in several ways,

among which the treatment of data [5] and the search for novel and effective algorithms [6] stand out.

The original proposal of this paper is framed in terms of the second option, i.e., the search for a novel and effective algorithm. The effectiveness of the proposed algorithm is based on its novelty as models for both machine learning tasks mentioned above have been merged to achieve this, i.e., merging classification and recalling. Furthermore, unbalanced datasets have been selected for the experimental study. This is particularly relevant because it is known in the world of pattern recognition, machine learning, and related fields that most pattern classification algorithms suffer from a significant bias towards the majority class, and therefore higher misclassification of the minority class, which often corresponds to important events [7,8].

Ideally, datasets should contain the same number of observations in each of their classes; however, actual datasets rarely meet this condition. Instead, it is common that the most interesting and challenging datasets, such as those of medical diagnosis, fraud detection, etc., are unbalanced [1]. The imbalance ratio (IR) measures the imbalance for a given dataset by dividing the cardinality of the majority class by the cardinality of the minority one. If IR > 1.5, then the dataset is considered to be unbalanced.

One of the most common ways to measure algorithm performance is accuracy. This is calculated as the percentage of correctly classified patterns out of the total patterns in the testing set. Let N be the total number of observations in the testing set and C the number of well-classified patterns, where the accuracy is then obtained by dividing C by N and multiplying the result by 100, where $0 < C \leq N$ and, correspondingly, $0 < \text{accuracy} \leq 100$.

For example, let us consider an extremely unbalanced dataset with 100 observations of which 95 are positive and 5 are negative. To illustrate how class imbalance prevents algorithms from obtaining a reliable performance, think about a classification algorithm that, after fitting the data, assigns the "positive" class to each observation in the testing set regardless of the true class. The number of correctly classified patterns is 95 out of 100, and therefore the accuracy value is 95%. Although this can be considered quite good, we know that the classifier could not correctly identify any of the observations in the minority class. Surprisingly, many of the state-of-the-art classifiers follow the same behavior, i.e., many of the well-known classifiers override the minority class in unbalanced datasets.

Unlike what happens with the large number of pattern classification methods, the algorithms that perform a recall task are scarce, among which associative memories stand out [9]. The pioneering associative model is the Lernmatrix, which was created in Germany in 1961 by Karl Steinbuch [10]. Due to the nature of the patterns it works with, this model behaves like a pattern classifier; however, the original Lernmatrix model is not competitive as a pattern classifier.

Another classic model for pattern recall is an associative memory model known as a linear associator, which dates back to 1972 and whose development is attributed to two scientists working simultaneously and independently, namely Kohonen in Finland [11] and Anderson in USA [12]. In relation to this model, studies have been carried out on its convergence [13] and performance [14]; however, it is pertinent to note that very few linear associator applications have been published [15] and that the reason for this is the low performance exhibited by this model in most datasets. This occurs because the linear associator enforces a very strong condition on the patterns in order for them to be recovered correctly. All patterns must be orthonormal, which is very difficult (if not impossible) to achieve in datasets generated in real-life applications.

The low performance exhibited by both models has motivated the scientific community to segregate them. Thus, these two pioneering models of associative memories have been forgotten for decades. Research on these two important classical associative models resumed in 2002. As a consequence of this research work, a postgraduate thesis was published in 2003 where a hybrid associative classifier with translation (better known by its acronym in Spanish, CHAT) was introduced, which merged both models. The new

hybrid model far outperformed the Lernmatrix and linear associator models when used separately [16].

Since then, work has been carried out to improve the performance of the CHAT. Current publications have presented successful case studies [17,18]. The original proposal presented in this paper is, in a certain way, a continuation of these works.

The rest of the paper is organized as follows. In Section 2, the algorithms that serve as the foundation for the proposal are explained individually. Section 3 is devoted to providing a detailed description of the main proposal of this communication, i.e., the extreme center points (ECP) method. The results are reported in Section 4, where it is discussed how the proposed method performs in an imbalanced class scenario. Finally, the conclusions and future work areas are presented.

2. Previous Works

According to what is disclosed by the no free lunch theorem, an ideal classifier does not exist. In consequence, machine learning researchers know that looking for algorithms that always produce zero errors is useless. For that reason, scientists are currently more focused on enhancing the performance of existing classifiers by reducing their production of classification errors. For instance, in [19], the authors proposed an enhancement to a k-NN algorithm by adding a cost-sensitive distance function with careful selection of parameter k. On the other hand, SVM models have been modified for improving their performance by introducing an advanced radial basis function kernel [20] and by using geometric transformations to achieve nonlinear structures for learning [21]. Furthermore, a refinement of the multilayer perceptron (MLP) in terms of optimizing data distribution in datasets as a method to find the most suitable number of hidden units was proposed in [22].

Correspondingly, associative memory (AM) models have also been modified to increase their performance. In the present work, a modification to the CHAT classification algorithm is proposed, which in turn represents an improvement in the associative algorithms by combining two of the first AM models. Due to its importance in this work, the main concepts of AM are detailed below, as well as those of the Lernmatrix, linear associator, and CHAT algorithms.

The main goal of AM is pattern recovery. An AM is an input–output system that is cleaved into two phases as follows: (1) the learning phase, in which input data are associated with the desired output, thus creating the associative memory, and the (2) recall phase, where a new input pattern is presented to the previously generated AM [23].

In this context, input patterns are denoted by column vectors $\mathbf{x}^{\mu}$, whereas their corresponding output patterns are denoted by column vectors $\mathbf{y}^{\mu}$, where $\mu = \{1,2, \ldots, p\}$, and p is the total amount of patterns in the training dataset. In the learning phase, the memory $\mathbf{M}$ is constructed by computing all the p associations $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$, and then an input pattern $\mathbf{x}^{\mu}$ is operated with the memory $\mathbf{M}$ to obtain their corresponding $\mathbf{y}^{\mu}$ output pattern in the recall phase.

If the whole set of input patterns is equal to the corresponding output pattern set, i.e., if each input pattern is associated with itself, then the memory is called an auto-associative memory. Conversely, if at least one of the input patterns differs from its associated output, i.e., it is not associated with itself, then the memory is referred to as a hetero-associative memory.

The pioneer models of associative memories emerged with the Lernmatrix by Steinbuch in the early 1960s [10]. This algorithm encouraged the generation of a range of associative memories that were subsequently proposed. This is the case for the linear associator proposed by Anderson and Kohonen in 1972 [11,12]. Both models had paramount importance in the origin of associative memories; however, the original models of the Lernmatrix and linear associator do not offer reasonable performance regarding current classification algorithms. Still, they have inspired the creation of a new set of competitive classification algorithms using their theory [16,17,24].

2.1. Lernmatrix

The Lernmatrix is a hetero-associative memory model that was proposed by Steinbuch in 1961 [10]. Due to its own nature, this model can work as a pattern classifier if it is provided with a proper set of output patterns. In this regard, when a binary pattern is presented to the memory in the recall phase, the memory will output a one-hot vector representing the class of the given pattern.

With the aim of clearly illustrating how the Lernmatrix behaves as a classifier, let us assume that $\mathbf{M}$ is a Lernmatrix and $\mathbf{x}^{\mu}$ represents the μ-*th* input pattern where $\mu = \{1, 2, \ldots, p\}$, with p being the number of patterns. Also, to represent the corresponding attribute of a given pattern, let $i = \{1, 2, \ldots, n\}$ where n is the dimension of the pattern. The *i-th* component of a given pattern $\mathbf{x}^{\mu}$ is denoted as x_i^{μ}. According to the latter, a given pattern of five dimensions representing the third pattern of a dataset would be denoted as follows:

$$\mathbf{x}^3 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \tag{1}$$

where the first component of this pattern equals one and is denoted as $x_1^3 = 1$.

In order to achieve proper classification with the Lernmatrix, it is required that output patterns would be denoted in such a way that they represent the belonging class of the patterns they are associated with. To do so, the use of a one-hot vector encoding system is favorable. If a given dataset is grouped into $k = 3$ different classes, we can associate the input patterns to an output pattern $\mathbf{y}^{\omega}$ where $\omega \in \{1, 2, 3\}$ represents the number of the associated class. In this case, the input patterns belonging to the first class would be associated with the corresponding $\mathbf{y}^1$ output pattern, represented as:

$$\mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{2}$$

The same would happen for the corresponding output patterns of the second and third classes, respectively:

$$\mathbf{y}^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{and } \mathbf{y}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{3}$$

Furthermore, as in the input pattern example, the same notation applies to describe a feature of a given output pattern, i.e., the i-th feature of a given output pattern $\mathbf{y}^{\omega}$ is denoted in the subscript of the pattern as: y_i^{ω}. For instance, in the previous example the second feature of the third output pattern would be denoted as: $y_2^3 = 0$.

Once understanding how the Lernmatrix could work as a classifier, we can proceed to explain the learning and recall phase of this associative memory per se.

2.1.1. Learning Phase

To build a corresponding Lernmatrix for a given dataset, it is necessary to first create a matrix $\mathbf{M} = \left[m_{ij}\right]_{kxn}$ where k is the number of classes in the dataset and n corresponds to the dimension of the given input patterns such that $m_{ij} = 0, \ \forall i, j$.

$$\mathbf{M} = \begin{pmatrix} m_{11} & \cdots & m_{1n} \\ \vdots & \ddots & \vdots \\ m_{k1} & \cdots & m_{kn} \end{pmatrix} \tag{4}$$

Then, the corresponding learning rule is determined according to the following:

$$m_{ij} = m_{ij} + \Delta m_{ij} \tag{5}$$

$$\Delta m_{ij} = \begin{cases} +\varepsilon & if\ y_i^\mu = 1 = x_j^\mu \\ -\varepsilon & if\ y_i^\mu = 1\ and = x_j^\mu = 0 \\ 0 & otherwise \end{cases} \tag{6}$$

where $\varepsilon > 0$.

2.1.2. Recalling Phase

Once the learning phase is performed, an input pattern $\mathbf{x}^\gamma$, whose class is unknown, is then presented to the previously generated memory $\mathbf{M}$. In order to obtain the corresponding class of such pattern, the next procedure is applied:

$$y_i^\omega = \begin{cases} 1 & if \sum_{j=1}^{n} m_{ij} x_j^\gamma = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} m_{hj} x_j^\gamma \right] \\ 0 & otherwise \end{cases} \tag{7}$$

where V represents the maximum operator.

After the unknown x^γ input pattern is operated with the memory, an output pattern y^ω is generated, represented by a one-hot vector. This one-hot vector represents the class of the pattern.

2.2. Linear Associator

The linear associator proposed by Anderson and Kohonen [11,12] is an AM that associates input patterns to their corresponding output for pattern recovery tasks. This model can appropriately recover the patterns of a training set if it meets the condition of featuring orthonormal vectors, which in practice is difficult to find. Analogous to the Lernmatrix, the linear associator also consists of the two phases explained below. In them, the training set consists of p patterns where $\mathbf{x}^\mu$ represents the input vectors with dimension n, and correspondingly $\mathbf{y}^\mu$ denotes the output vectors with dimension m.

2.2.1. Learning Phase of the Linear Associator

In this phase, the memory is obtained by operating the input and output patterns according to the following two steps.

1. For each input pattern $\mathbf{x}^\mu$, compute the matrix $\mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^t$ as detailed in Equations (8) and (9).

$$y^\mu \cdot (x^\mu)^t = \begin{pmatrix} y_1^\mu \\ y_2^\mu \\ \vdots \\ y_m^\mu \end{pmatrix} \begin{pmatrix} x_1^\mu, x_2^\mu, \ \dots \ , x_n^\mu \end{pmatrix} \tag{8}$$

$$y^\mu \cdot (x^\mu)^t = \begin{pmatrix} y_1^\mu x_1^\mu & y_1^\mu x_2^\mu & \dots & y_1^\mu x_j^\mu & \dots & y_1^\mu x_n^\mu \\ y_2^\mu x_1^\mu & y_2^\mu x_2^\mu & \dots & y_2^\mu x_j^\mu & \dots & y_2^\mu x_n^\mu \\ \vdots & \vdots & & \vdots & & \vdots \\ y_i^\mu x_1^\mu & y_i^\mu x_2^\mu & \dots & y_i^\mu j & \dots & y_i^\mu x_n^\mu \\ \vdots & \vdots & & \vdots & & \vdots \\ y_m^\mu x_1^\mu & y_m^\mu x_2^\mu & \dots & y_m^\mu x_j^\mu & \dots & y_m^\mu x_n^\mu \end{pmatrix} \tag{9}$$

where this produces resulting p matrices of dimension $m \times n$.

2. Sum the p matrices to get the following memory:

$$\mathbf{M} = \sum_{\mu=1}^{p} y^{\mu} \cdot \left(x^{\mu}\right)^{t} = \left[m_{ij}\right]_{mxn} \tag{10}$$

where the ij-th component of the memory is expressed as:

$$m_{ij} = \sum_{\mu=1}^{p} y_i^{\mu}\ x_j^{\mu} \tag{11}$$

2.2.2. Recalling Phase of Linear Associator

Considering **M**, the memory computed in the previous phase, and $\mathbf{x}^{\omega}$ being an input pattern, then the recalling phase consists in obtain the corresponding output vector $\mathbf{y}^{\omega}$ by performing the following operation:

$$\mathbf{y}^{\omega} = \mathbf{M} \cdot \mathbf{x}^{\omega} = \left[\sum_{\mu=1}^{p} \mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^{t}\right] \cdot \mathbf{x}^{\omega} \tag{12}$$

2.3. CHAT

The hybrid associative classifier with translation was proposed by [16] and is a combination of the two previously explained algorithms. More precisely, the CHAT algorithm implements the training phase of the linear associator and the recall phase of the Lernmatrix model. Furthermore, the CHAT is an enhancement of a hybrid associative classifier (CHA, by its acronym in Spanish) algorithm and differs by adding a translation of the coordinate axes.

To perform corresponding pattern translation, the CHAT algorithm makes use of a translation vector which in turn is defined as the mean vector of all the given input patterns. After the translation vector has been obtained, this vector is subtracted from all the input patterns in order to translate them to a new coordinate axis system in which the translation vector is the new origin point. In doing so, a new set of input patterns is produced as detailed in Definition 1 and Definition 2. Finally, Algorithm 1 describes in detail how the CHAT works.

Definition 1. *Translation vector. In the CHAT algorithm, the translation vector is represented by the mean vector of the training input patterns by using the next equation*:

$$\bar{\mathbf{x}} = \frac{1}{p} \sum_{j=1}^{p} \mathbf{x}^{\mu} \tag{13}$$

Definition 2. *Pattern translation. After the translation vector* $\bar{x}$ is obtained, the whole set of input patterns is translated, having this point as the origin of the new coordinate axes.

$$\hat{\mathbf{x}^{\mu}} = \mathbf{x}^{\mu} - \bar{\mathbf{x}},\ \forall \mathbf{x}^{\mu} \text{ where } \mu\ \epsilon\ \{1, 2, \ldots, p\} \tag{14}$$

With the aim of explaining its functionality, pseudocode for the CHAT algorithm is presented below.

Algorithm 1. CHAT algorithm.

```
Input: dataset, input_pattern, p
Output: recovered_pattern
Initialize translation_vector = []
for pattern in dataset
    assign one-hot vector according to its class
end for
for pattern in dataset
    translation_vector = translation_vector + pattern
end for
translation_vector = translation_vector/p
pattern_translation = dataset—translation_vector
Learning phase from Linear associator ( )
Lernmatrix recalling phase ( )
Sub-routine 1: Learning phase from Linear associator
Input: dataset
Output: associative_memory
Initialize associative_memory = [[]]
for pattern in dataset
    matrix = one-hot_vector * transpose(pattern)
    associative_memory = associative_memory + matrixend for
Sub-routine 2: Lernmatrix recalling phase
Input: associative_memory, input_pattern
Output: recovered_pattern
Initialize recovered_pattern = []
pattern = multiply_matrix (input_pattern, associative_memory)
max_value = maximunValue(pattern)
for i in range (0, len(pattern)
    if pattern [i] == max_value
        recovered_pattern [i] = 1
    else
        recovered_pattern [i] = 0
    end if
end for
```

Although diverse improvements for state-of-the-art classification algorithms have been proposed, class imbalances still constitute an unresolved problem, since, as mentioned above, most pattern classification algorithms are highly biased towards the majority class, making it very difficult for them to correctly identify any observations in the minority class. The main purpose of this paper is to provide the CHAT algorithm with a mechanism for boosting performance in the context of class imbalance, which is demonstrated by a number of experiments over twelve unbalanced datasets.

3. Proposed Methodology

As mentioned before, the CHAT algorithm improves the performance of the CHA by means of a vector translation step. It is noteworthy that the translation vector in the original CHAT algorithm is represented by the mean point of the training dataset. In this regard, we hypothesize that the existence of a different translation point may lead to enhanced classification results. We conducted experiments in order to investigate this idea by proposing the ECP method, which in turn is a method for finding the best translation vector for a given dataset.

Extreme Center Points

This method creates an n-dimensional search hyperspace for selecting the translation vector that produces the best classification results in the CHAT algorithm. In this work, two alternatives for the ECP are presented: ECP (7) and ECP (9). Regardless, the election of

one of these two heuristics replaces steps 3 and 4 of the CHAT original algorithm and is detailed as follows:

1. Generate the extreme center points for each attribute of the training dataset. The generation of these points represents the construction of a mesh for exploring a wide range of values per attribute in order to select the translation vector that best fits the CHAT algorithm for pattern classification. The 7-point version of the ECP model, ECP (7), considers the following points:
   - Minimum value of attribute;
   - Minimum value of attribute + one standard deviation;
   - Mean value of attribute − one standard deviation;
   - Mean value of attribute;
   - Mean value of attribute + one standard deviation;
   - Maximum value of attribute − one standard deviation;
   - Maximum value of attribute.

   For the ECP (9), the attribute points for the search space are the following:
   - Minimum value of attribute − one standard deviation;
   - Minimum value of attribute;
   - Minimum value of attribute + one standard deviation;
   - Mean value of attribute − one standard deviation;
   - Mean value of attribute;
   - Mean value of attribute + one standard deviation;
   - Maximum value of attribute − one standard deviation;
   - Maximum value of attribute;
   - Minimum value of attribute + one standard deviation

   Note that the proposed methodology is capable of obtaining better results than the original CHAT model, or at least the same performance, as the mean vector, i.e., the original election of translation vector, is included in the search space of translation vector.
2. Generate all the possible combinations using the ECP over n attributes in the training dataset. Every combination represents a possible translation vector to be used in the original CHAT algorithm.
3. Test all possible solutions in the obtained search space, i.e., evaluate the CHAT algorithm using each of the points generated in the previous step, as the translation vector. Additionally, select the point that better improves the classification results. this point is called center point (CP).
4. With the aim of refining the values used to generate the translation vector, a neighborhood of the center point is then analyzed. A more fine-grained spatial search is performed around a neighborhood of ± 1 standard deviation from the CP. That is, for each attribute in the Center Point, n more points are equally distributed around it (for this work in particular, $n = 10$). These new set of points are called deep points (DP) and are distributed for each attribute as depicted in Figure 1.
5. Afterwards, a reevaluation of the CHAT using the DP as the translation vector is carried out. To this end, steps 2 and 3 of the proposed method are repeatedly performed with the recently obtained deep points. Finally, the best point is selected as the translation vector to be used for classification of unknown patterns using the CHAT algorithm.

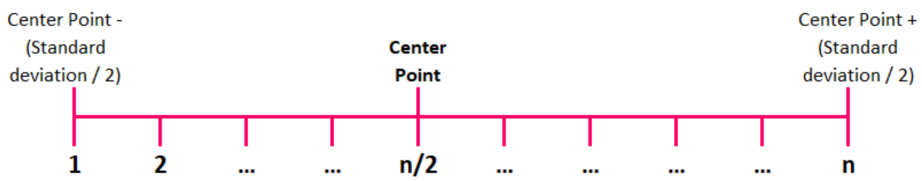

Figure 1. Diagram of the spatial distribution of deep points around the center point.

4. Results and Discussion

This section presents a detailed report about the experiments conducted using our proposal in comparison to well-known classifiers in the state of the art.

In preliminary tests using CHAT-ECP, we observed that the election of different translation vectors is particularly useful in imbalanced class scenarios. In accordance with this premise, the datasets used in this paper were balanced as described below.

4.1. Datasets

In general, the selected datasets had an imbalance ratio (IR) of more than 5, except for three of them, with IR values of of 2.78, 2, and 1.7, respectively. All datasets used here contained only numerical attributes and are available from the KEEL data repository (https://sci2s.ugr.es/keel/index.php). A general picture of these datasets can be seen in Table 1.

Table 1. Dataset information.

Dataset	Attributes	Patterns	IR*
Haberman	3	306	2.78
New-thyroid1	5	215	5.14
Iris0	4	150	2
E. coli (imbalanced: 0–4–6 vs. 5)	6	203	9.15
E. coli (imbalanced: 0–1 vs. 5)	6	240	11
E. coli (imbalanced: 0–6–7 vs. 5)	6	220	10
E. coli (imbalanced: 0–1–4–7 vs. 5–6)	6	332	12.28
E. coli (imbalanced: 0–1–4–6 vs. 5)	6	280	13
E. coli (imbalanced: 2–6 vs. 0–1–3–7)	7	281	39.14
LED display domain (imbalanced: 0–2–4–5–6–7–8–9 vs. 1)	7	443	10.97
Hayes-Roth	5	160	1.7
Balance scale	4	625	5.88

* IR: Imbalanced Ratio.

Short descriptions for each selected dataset are presented below.

Haberman: This dataset comes from a study conducted by University of Chicago's Billings Hospital between 1958 and 1970. It was recovered from the UCI repository dataset at https://archive.ics.uci.edu/ml/datasets/haberman%27s+survival, donated in 1999. It was a study regarding the survival rate from patients that went through a breast cancer procedure. All the dataset's attributes are integers. It is a binary class dataset where the possible classes are patient survival after 5 years or longer (class 1) or patient death within 5 years (class 2).

New-thyroid1: This was a modification from the original dataset that can be accessed through the UCI machine learning repository: https://archive.ics.uci.edu/ml/datasets/thyroid+disease and was donated by Ross Quinlan from the Garavan Institute. Nevertheless, the new-thyroid1 dataset is an imbalanced version of the aforementioned dataset. It can be obtained from the KEEL repository and the classes are divided in two. The examples which have hyperthyroidism represent the positive class and the rest of the classes represent the negative class.

Iris0: The iris0 dataset is an adaptation of the well-known flower classification iris dataset. As the original iris dataset is completely balanced, iris0 was modified to feature imbalanced classes. For this purpose, the original iris-versicolor and iris-virginica classes were grouped together into one single class (negative) and the remaining iris-setosa class was label as positive.

E. coli datasets collect different measurements of the cell to predict the locations of proteins. The datasets featured the following classes: cytoplasm (cp), inner membrane (im), perisplasm (pp), outer membrane (om), outer membrane lipoprotein (omL), inner

membrane uncleavable signal sequence (imU), inner membrane lipoprotein (imL), and inner membrane cleavable signal sequence (imS).

KEEL created different imbalanced versions of E. coli that separated the patterns into two classes (positive and negative). The versions of E. coli used in this experiment were the following:

E. coli (**imbalanced: 0–4–6 vs. 5**): A version of E. coli where the positive class is comprised of patterns of cp, imU, and omL and the negative class is the class om.

E. coli (**imbalanced: 0–1 vs. 5**): A version of E. coli where the positive class is comprised of patterns of cp and im and the negative class is the class om.

E. coli (**imbalanced: 0–6–7 vs. 5**): A version of E. coli where the positive class is comprised of patterns of cp, omL, and pp, whereas the negative class is the class om.

E. coli (**imbalanced: 0–1–4–7 vs. 5–6**): A version of E. coli where the positive class groups the patterns of the cp, im, imU, and pp classes, while the negative class is comprised of the patterns of om and omL.

E. coli (**imbalanced: 0–1–4–6 vs. 5**): A version of E. coli where the positive class is comprised of the patterns of cp, im, imU, and omL and the negative class is the class om.

E. coli (**imbalanced: 2–6 vs. 0–1–3–7**): A version of E. coli in which the positive class is represented by the patterns of pp and imL and the negative class groups the patterns of cp, im, imU, and imS.

LED display domain: Similar to previous datasets, this was obtained from the KEEL repository, but the original data can be recovered from the UCI repository dataset. Each pattern describes the recording of a LED display by seven light-emitting diodes represented by binary values: one if the LED is on or zero if not. Also, as mentioned in the dataset information, each of these attributes has a 10% probability of being inverted, hence introducing noise which represents a theoretical misclassification rate of 26%. The class is an integer value from zero to nine, representing the digit shown on the display.

Hayes-Roth: This dataset was obtained from the KEEL repository and constitutes a modified version of the original UCI dataset created by Barbara and Frederick Hayes-Roth and was donated by David W. Aha. It is an artificial dataset created with the purpose of having a baseline to compare the behavior of distinct classification algorithms. It is comprised of three attributes (age, educational level, and marital status) ranging from 1–4 and one attribute (hobby) generated at random in a range of 1–3 with the aim of adding noise to the data.

Balance scale: This dataset was recovered from the KEEL repository, but it is not a native dataset from the KEEL project. The original dataset comes from the UCI machine learning repository and has the purpose of modeling psychological experimental results by classifying four attributes (left weight, left distance, right weight, and right distance) with integer values ranging between 1–5 into one of three classes: tip to the right (R), tip to the left (L), or balanced (B).

4.2. Classifiers

In this section, a brief description of each classification algorithm used in the experimental stage is provided. All algorithms employed in the experiments are included in the WEKA [25] data mining software, which was made with Java. Although WEKA encompass a wide range of algorithms, for comparative purposes, only those with the best results were included. All algorithms were executed using their default parameters.

K-nearest neighbor (**KNN**): The KNN algorithm is a simple and robust supervised classification algorithm without a specific learning stage [26]. K-nearest neighbor methods use a distance function in order to assign the most frequent class of the K-closest neighbors to the pattern.

Sequential minimal optimization (**SMO**): SMO uses quadratic programming and sequential minimal optimization to represent hyperplanes or decision boundaries to separate subsets of patterns. This method performs a high-dimensional mapping of the data and looks for boundaries between classes or regions [27].

Multilayer perceptron (**MLP**): A MLP is a type of a backward propagation neural network algorithm [28]. Multilayer perceptrons are networks composed of a multitude of units (called neurons) that are interconnected with each other in multiple layers. Neurons provide an output based on their inputs. The outputs are obtained applying a predefined function; it is generally simple but becomes more complex as we add more layers or neurons.

JRip: The JRip classifier implements a set of proportional learning rules known collectively as repeated incremental pruning to produce error reduction (RIPPER) [29] and was proposed by Cohen W. William as an optimized version of IREP.

Naïve Bayes: Naïve Bayes methods are based on the use of probability and statistics using the principles of the Bayes theorem [30]. Specifically, it uses the Bayes theorem by supposing a conditional naïve independent probability between each pair of characteristics, thus producing the value of the class of a given pattern.

J48: Also known as C4.5 [31], the J48 algorithm generates a decision tree as an extension of the ID3 algorithm. It implements a process of trimming one single step to mitigate overfitting. It can handle discrete or continuous data and it is also capable of handling missing values.

Random Forest: The random forest algorithm proposed by [32] is a combination of other proposed models. The main idea behind the algorithm is to create a collection of decision trees using a random dependent vector. It also implements a random selection of features in combination with a bootstrap aggregation algorithm in order to generate more controlled variance decision trees.

4.3. Validation Method

A validation method must be employed in order to estimate the behavior of the classification models when applied to unknown input patterns. Here, a leave-one-out cross-validation (LOOCV) method was implemented to separate datasets into training and testing subsets.

LOOCV is an iterative method that selects a single instance at each iteration for the validation set and the remaining examples for training the classifier. This process is repeated for each pattern in the dataset. The main advantage of employing LOOCV is that it follows a deterministic process, i.e., that there is no random separation of the data and consequently the results are fully reproducible.

4.4. Performance Evaluation Metrics

An evaluation metric should be adopted in order to evaluate the performance of the classification methods. Considering that the datasets included in this study featured imbalanced classes, a quite suitable metric for measuring classification results is the balanced accuracy metric. To calculate the balanced accuracy, the following equations were computed:

$$Balanced\ Accuracy = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right) \tag{15}$$

$$Overall\ Accuracy = \frac{(TP + TN)}{(P + N)} \tag{16}$$

where TP (true positive) and TN (true negative) represent the number of the positive and negative instances that are correctly classified, respectively. On the other hand, P and N represent the number of positive (P) and negative (N) instances in the dataset.

4.5. Classification Results

The experimental results for the balanced accuracy can be seen in Table 2. Each column represents the tested classifier, whereas each row corresponds to the selected dataset. For each dataset, the best results for a particular dataset are shown in bold.

Table 2. Dataset classification results.

Dataset	CHAT ECP (9)	CHAT ECP (7)	CHAT (Original)	IB1	IB3	IB5	JRip	Random Forest	SMO	Naive Bayes	MLP	J48 (C4.5)
Haberman	**0.635**	**0.635**	0.632	0.557	0.556	0.529	0.598	0.541	0.498	0.588	0.595	**0.635**
New-thyroid1	0.886	0.886	0.746	0.98	0.937	0.937	0.926	0.929	0.786	**0.989**	0.966	0.98
Iris0	0.98	0.99	0.96	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0.99
E. coli (imbalanced: 0–4–6 vs. 5)	0.897	0.897	0.809	0.872	**0.922**	**0.922**	0.839	0.87	0.847	0.881	0.892	0.756
E. coli (imbalanced: 0–1 vs. 5)	**0.923**	**0.923**	0.775	0.87	**0.923**	0.898	0.832	0.868	0.85	**0.923**	0.895	0.782
E. coli (imbalanced: 0–6–7 vs. 5)	**0.89**	**0.89**	0.798	0.84	0.82	0.848	0.868	0.873	0.8	0.878	0.87	0.843
E. coli (imbalanced: 0–1–4–7 vs. 5–6)	**0.918**	**0.918**	0.792	0.87	0.873	0.915	0.789	0.817	0.778	0.853	0.897	0.917
E. coli (imbalanced: 0–1–4–6 vs. 5)	**0.925**	**0.925**	0.777	0.873	0.923	0.898	0.763	0.844	0.798	0.888	0.84	0.752
E. coli (imbalanced: 2–6 vs. 0–1–3–7)	**0.857**	**0.857**	0.772	0.848	0.852	0.853	0.852	0.784	0.852	0.853	0.855	0.855
LED display domain (imbalanced: 0–2–4–5–6–7–8–9 vs. 1)	0.87	0.87	0.823	0.909	**0.91**	**0.91**	0.893	0.896	0.88	0.849	0.863	0.88
Hayes-Roth	0.556	0.556	0.516	0.744	0.373	0.288	0.784	**0.817**	0.567	0.732	0.751	0.784
Balance scale	0.618	0.618	0.618	0.63	0.63	0.643	0.582	0.58	0.639	0.657	**0.828**	0.563

Best results are denoted in bold.

The obtained results show that CHAT-ECP (9) and CHAT-ECP (7) achieved balanced accuracy values greater than the rest of the algorithms for four of the twelve datasets, namely, E. coli (imbalanced: 0–6–7 vs. 5) with balanced accuracies of 0.89 and 0.89 respectively, along with E. coli (imbalanced: 0–1–4–7 vs. 5–6), E. coli (imbalanced: 0–1–4–6 vs. 5), and E. coli (imbalanced: 2–6 vs. 0–1–3–7) with values of 0.918, 0.925, and 0.857, respectively, for both proposed methods.

Moreover, the proposed methodology had the best performance in two more datasets, namely the Haberman dataset, which tied with J48, obtaining a result of 0.635. For the E. coli (imbalanced: 0–1 vs. 5) dataset, our proposal achieved the best results jointly with the IB3 and the Naïve Bayes algorithms with a balanced accuracy of the 0.923.

As can be seen in Table 2, the proposed methodology did not obtain a value of one as a balanced accuracy value for any dataset. In difference, eight classifiers obtained a performance with a value of one, but only in one of the datasets for all classifiers, namely the one with the lowest IR (iris0), for which our method achieved a competitive result of 0.98.

Besides, the results for both CHAT-ECP (9) and CHAT-ECP (7) were practically the same, being indicative of the consistency of the method even when the number of center points (on which the algorithm is built) differed. In addition, we obtained a balanced accuracy greater than 0.85 in 9 of the 12 datasets, while the competing algorithms obtained similar values on average for 6 datasets.

It is important to mention that our proposal achieved the best score for datasets with IR values higher than 10, except for the LED display domain.

Do not forget that the initial intention of this work was to overcome the limitations of the CHAT algorithm by adding the ECP method. In this regard, it is clear that ECP (9) and ECP (7) improved the results of the original version for all datasets as seen in Table 2.

Statistical significance tests consist of rejecting or accepting null hypothesis H0, i.e., that there are no significant differences among a group of data. To this regard, we used the Friedman test [33] in order to identify statistical differences in the performance results of the classification algorithms.

When only looking to Table 2, the performances of the different classifiers show similar results. Nevertheless, after running the Friedman statistical test, the null hypothesis was rejected with a confidence of 95% with a p-value of 0.001, which provides evidence of statistically significant differences among classifiers. Furthermore, the proposed CHAT-ECP (7) method was ranked best according to the Friedman mean ranks for comparative methods, whereas the original CHAT algorithm stayed in last place, as depicted in Table 3.

Table 3. Friedman mean ranks.

Classifier	Mean Ranks [1]
CHAT-ECP (7)	4.833
CHAT-ECP (9)	4.958
Naive Bayes	5.125
MLP	5.167
IB5	5.375
IB3	5.667
IB1	6.208
Random Forest	7.042
J48 (C4.5)	7.042
JRip	7.583
SMO	8.583
CHAT (original)	10.417

[1] Ordered from best to worst.

5. Conclusions

In this paper, a novel methodology for obtaining the translation vector for the CHAT classification algorithm was presented. This proposal has two alternatives, either using ECP (7) or ECP (9), whose points are defined by the proposed heuristic.

The results show that the proposed method is competitive with other classification algorithms shown in the specialized literature.

In particular, one of the benefits of exploring the space of possible solutions to choose the best translation vector for each dataset is that when the findings are applied to imbalanced class scenarios, they translate the dataset in such a way that the chat algorithm predictably carries out the classification task accurately.

For future research, we suggest considering an approach that allows the exploration a wider range in the search space without the need to define specific points. In this regard we believe that using metaheuristic algorithms may be of great benefit for achieving this goal.

It has been shown that an appropriate election of the translation vector is helpful to obtain an improved performance. From there, we conclude that the application of the new ECP method, results in a meaningful improvement in the classification performance of the CHAT algorithm.

Author Contributions: Conceptualization, C.Y.-M. and A.A.-P.; methodology, A.A.-P., I.L.-Y. and C.Y.-M.; software, C.A.R.-G. and R.C.-M.; validation, I.L.-Y. and A.A.-P.; formal analysis, C.Y.-M. and I.L.-Y.; investigation, C.A.R.-G. and R.C.-M.; writing—original draft preparation, A.A.-P. and C.Y.-M.; writing—review and editing, R.C.-M., I.L.-Y., C.A.R.-G., A.A.-P. and C.Y.-M.; visualization, R.C.-M. and C.A.R.-G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors want to thank the Instituto Politécnico Nacional of Mexico (Secretaría Académica, CIC, SIP and CIDETEC), the CONACyT, and the SNI for their support to develop this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Burkart, N.; Huber, M.F. A Survey on the Explainability of Supervised Machine Learning. *J. Artif. Intell. Res.* **2021**, *70*, 245–317. [CrossRef]
2. Duda, R.O.; Hart, P.E.; Stork, D.G. *Pattern Classification*, 2nd ed.; John Wiley & Sons: New York, NY, USA, 2001; pp. 20–450.
3. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
4. Adam, S.P.; Alexandropoulos, S.-A.N.; Pardalos, P.M.; Vrahatis, M.N. No Free Lunch Theorem: A Review. In *Dynamics of Disasters*; Springer Science and Business Media LLC: Berlin/Heidelberg, Germany, 2019; Volume 145, pp. 57–82.
5. Ruan, S.; Li, H.; Li, C.; Song, K. Class-Specific Dee: Feature Weighting for Naïve Bayes Text Classifiers. *IEEE Access* **2020**, *8*, 20151–20159. [CrossRef]

6. Paranjape, P.; Dhabu, M.; Deshpande, P. A novel classifier for multivariate instance using graph class signatures. *Front. Comput. Sci.* **2020**, *14*, 144307. [CrossRef]
7. Fernández, A.; López, V.; Galar, M.; del Jesus, M.J.; Herrera, F. Analysing the classification of unbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowl.-Based Syst.* **2013**, *42*, 97–110. [CrossRef]
8. Mullick, S.S.; Datta, S.; Dhekane, S.G.; Das, S. Appropriateness of performance indices for imbalanced data classification: An analysis. *Pattern Recognit.* **2020**, *102*, 107197. [CrossRef]
9. Karpov, Y.L.; Karpov, L.E.; Smetanin, Y.G. Some Aspects of Associative Memory Construction Based on a Hopfield Net-work. *Program. Comput. Softw.* **2020**, *46*, 305–311. [CrossRef]
10. Steinbuch, K. Die Lernmatrix. *Biol. Cybern.* **1961**, *1*, 36–45. [CrossRef]
11. Kohonen, T. Correlation Matrix Memories. *IEEE Trans. Comput.* **1972**, *21*, 353–359. [CrossRef]
12. Anderson, J.A. A simple neural network generating an interactive memory. *Math. Biosci.* **1972**, *14*, 197–220. [CrossRef]
13. Reid, R.; Frame, J. Convergence in Iteratively Formed Correlation Matrix Memories. *IEEE Trans. Comput.* **1975**, *C-24*, 827–830. [CrossRef]
14. Turner, M.; Austin, J. Matching performance of binary correlation matrix memories. *Neural Netw.* **1997**, *10*, 1637–1648. [CrossRef]
15. Austin, J.; Lees, K. A search engine based on neural correlation matrix memories. *Neurocomputing* **2000**, *35*, 55–72. [CrossRef]
16. Santiago-Montero, R. Clasificador Híbrido de Patrones Basado en la Lernmatrix de Steinbuch y en el Linear Associator de Anderson-Kohonen. Master Thesis, Centro de Investigación en Computación del Instituto Politécnico Nacional, Ciudad de México, México, 2003.
17. Uriarte-Arcia, A.V.; López-Yáñez, I.; Yáñez-Márquez, C. One-hot vector hybrid associative classifier for medical data classification. *PLoS ONE* **2014**, *9*, e95715.
18. Cleofas-Sánchez, L.; Sánchez, J.S.; García, V.; Valdovinos, R.M. Associative learning on imbalanced environments: An empirical study. *Expert Syst. Appl.* **2016**, *54*, 387–397. [CrossRef]
19. Zhang, S. Cost-sensitive KNN classification. *Neurocomputing* **2020**, *391*, 234–242. [CrossRef]
20. Gopi, A.P.; Jyothi, R.N.S.; Narayana, V.L.; Sandeep, K.S. Classification of tweets data based on polarity using improved RBF kernel of SVM. *Int. J. Inf. Technol.* **2020**, 1–16. [CrossRef]
21. Shi, B.; Liu, J. Nonlinear metric learning for kNN and SVMs through geometric transformations. *Neurocomputing* **2018**, *318*, 18–29. [CrossRef]
22. Zhao, Z.; Xu, S.; Kang, B.H.; Kabir, M.M.J.; Liu, Y.; Wasinger, R. Investigation and improvement of multi-layer perceptron neural networks for credit scoring. *Expert Syst. Appl.* **2015**, *42*, 3508–3516. [CrossRef]
23. Hassoun, M.H. *Associative Neural Memories*, 1st ed.; Oxford University Press, Inc: Ann Arbor, MI, USA, 1993.
24. Velázquez-Rodríguez, J.-L.; Villuendas-Rey, Y.; Camacho-Nieto, O.; Yáñez-Márquez, C. A Novel and Simple Mathematical Transform Improves the Perfomance of Lernmatrix in Pattern Classification. *Mathematics* **2020**, *8*, 732. [CrossRef]
25. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA data mining software: An update. *ACM SIGKDD Explor. Newsl.* **2009**, *11*, 10–18. [CrossRef]
26. Tsalera, E.; Papadakis, A.; Samarakou, M. Monitoring, profiling and classification of urban environmental noise using sound characteristics and the KNN algorithm. *Energy Rep.* **2020**, *6*, 223–230. [CrossRef]
27. Luo, Y.; Xiong, Z.; Xia, S.; Tan, H.; Gou, J. Classification noise detection based SMO algorithm. *Optik* **2016**, *127*, 7021–7029. [CrossRef]
28. Hoffmann, L.F.S.; Bizarria, F.C.P.; Bizarria, J.W.P. Detection of liner surface defects in solid rocket motors using multi-layer perceptron neural networks. *Polym. Test.* **2020**, *88*, 106559. [CrossRef]
29. Toneva, D.H.; Nikolova, S.Y.; Agre, G.P.; Zlatareva, D.K.; Hadjidekov, V.G.; Lazarov, N.E. Data mining for sex estima-tion based on cranial measurements. *Forensic Sci. Int.* **2020**, *315*, 110441. [CrossRef] [PubMed]
30. Andrejiova, M.; Grincova, A. Classification of impact damage on a rubber-textile conveyor belt using Naïve-Bayes method-ology. *Wear* **2018**, *414–415*, 59–67. [CrossRef]
31. Mohanty, M.; Sahoo, S.; Biswal, P.; Sabut, S. Efficient classification of ventricular arrhythmias using feature selection and C4.5 classifier. *Biomed. Signal Process. Control.* **2018**, *44*, 200–208. [CrossRef]
32. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
33. Friedman, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J. Am. Stat. Assoc.* **1937**, *32*, 675–701. [CrossRef]