Financial credit analysis via a clustering weightless neural classifier

Douglas O. Cardoso[a], Danilo S. Carvalho[a], Daniel S. F. Alves[a], Diego F. P. Souza[a], Hugo C. C. Carneiro[a], Carlos E. Pedreira[a], Priscila M. V. Lima[b], Felipe M. G. França[a]

[a] *COPPE, Universidade Federal do Rio de Janeiro, BRAZIL*
[b] *NCE, Instituto Tércio Pacitti, Universidade Federal do Rio de Janeiro, BRAZIL*

Abstract

Credit analysis is a real-world classification problem where it is quite common to find datasets with a large amount of noisy data. State-of-the-art classifiers that employ error minimisation techniques, on the other hand, require a long time to converge, in order to achieve robustness. This paper explores ClusWiSARD, a clustering customisation of the WiSARD weightless neural network model, applied to two different credit analysis real-world problems. Experimental evidence shows that ClusWiSARD is very competitive with Support Vector Machine (SVM) w.r.t. accuracy, with the advantage of being capable of online learning. ClusWiSARD outperforms SVM in training time, by two orders of magnitude, and is slightly faster in test time.

Keywords: Bleaching, ClusWiSARD, clustering, concept drifting, credit assignment, online learning

1. Introduction

Credit analysis represents the complex tasks of deciding which credit applicants present a good probability of returning the granted credit and which do not. This task depends on many different factors, such as economic and cultural circumstances, and is often delegated to human experts. Human judgement, however, may not use explicit rules that can be referenced as basis for decision making. That could lead to conflicting analysis of the same problem instance from different experts. In some countries, this is considered illegal. This question would justify the design of a machine learning system

that is able to replace the decisions of experts, providing a single analysis standard.

Important pattern recognition challenges can be found in credit analysis. For example, data can be noisy or corrupted due to problems in data collection. Data could also embed temporal information, possibly useful to identify *concept drift*: movement of populations, changes in economy, natural catastrophes [1], general news [2] etc. These and other factors may affect the relations pertinent to credit assignment. Class imbalance is also expected, as credit applications labeled as "good" are more frequent than "bad" ones.

How observations were gathered and labeled is also noteworthy. Labelling could be done *a priori*, according to a risk appraisal system already in use. Alternatively, this could be performed after observing if payment of granted requests was duly realised. A system trained with data from the first case aims to reproduce the behaviour of the established classification system, instead of attempting to excel it. In the second case, training data is the product of a filtering process, implying in a reduction of information about the population.

Different machine learning techniques have been analysed in the context of this problem. As discriminated by Tsai [3], they may be classified in three smaller sets, which are: single classifiers, classifier ensemble and hybrid classifiers. The first one contains single supervised models, like Support Vector Machine (SVM) [4–8], Multilayer-Perceptron (MLP) [4, 9, 10], Decision Trees (DT) [11] and Genetic Algorithm/Programming (GA/GP) [12, 13]. Regarding classification accuracy over the UCI dataset, which was also used in this work, some results obtained were 77.34% by Ong, Huang and Tzeng [13] with the use of GP and 77.09%/76.59% by Tsai [4] with SVM and MLP, respectively. These models have achieved at most an accuracy of 77.34% working as single classifiers. However, they may achieve much better results when grouped together, forming classifier ensembles. For instance, Ghodselahi [14] has obtained 81.42% with the use of a SVMs ensemble, and Hoffmann [15] has reached 84.90% with a GA-based SVM. Some other approaches used GA-based MLP [16] and GA-based SVM [17] in other financial credit analysis datasets. The third category, called Hybrid Classifiers, contains approaches mixing two or more techniques. For instance, combining clustering and single classifiers. Previously a work [3] compared many different approaches and claimed to achieve up to 88.93% through the use of clustering and SVM. Since ClusWiSARD is an intrinsic clustering WiSARD classifier, it retains the single classifier characteristic. Therefore, its results are comparable with

other single classifier approaches. Besides, this model could later be further improved through the use of other techniques, like ensemble learning and GA.

Financial institutions may lend money to different types of entities (people, businesses, non-profits, among others) and under various sets of conditions. This makes the context of credit analysis very diverse, with specialized methods in constant development for each category of credit operation. This work focuses on the case of retail credit, i.e., when the credit is given directly to the consumer (a person) rather than an organization. Retail credit analysis is mostly done by means of *credit scoring* [18], which has been the standard practice for decision making in credit risk management for the past 35 years in the US, UK and some other countries [18].

The purpose of credit scoring revolves around the ability to rank a prospective customer in a given set of categories, in order to assess the probability of timely payment of the debt over a period of time. As a result, a "grade" is given to the customer, which will serve to classify him or her as being a probable good or bad payer and thus facilitating the decision of giving the credit or not. Such "grades" are dependent on the method used for scoring, usually being an number but "good" and "bad" categories are also often used. The scoring methods are usually divided into two categories: statistical and non-statistical. The first category includes techniques such as logistic regression and discriminant analysis, with uses dating back from the beginning of credit scoring activities. The second category includes a wide range of computer backed algorithms such as neural networks, linear programming and genetic algorithms. Advances in machine learning research and computational capabilities over the past decades have promoted significant increases in predictive accuracy for many non-statistical methods, boosting their adoption, although use of statistical methods has not been abandoned [19]. More sophisticated non-statistical methods, such as ensembles, have shown a boost in adoption after the 2008 financial crisis, which resulted in restrictions regarding retail credit provision [20]. The method presented in this paper is non-statistical and is compared to another method of the same category for illustration of its capabilities. The method chosen for comparison is the support vector machine (SVM) which is often used for the scoring task.

Having an automated learning and classification mechanism that could offer a more precise solution is an attractive idea. It must be able to analyse vast amounts of data on credit applications and consider subtle relations between the actual financial data and the borrower profile. However, such

mechanism would also need to be both efficient and robust in order to account for changes in the circumstances and sample biasing. Two classifying mechanisms which have potential to exhibit these characteristics are the WiSARD [21] weightless artificial neural network model and the Support Vector Machine (SVM) [22], which are introduced, respectively, in sections 2.1 and 2.2. This work proposes the application of WiSARD weightless neural network model for the credit analysis problem, both in its traditional form, targeting simpler scenarios, as well as in a clustering oriented architecture, called ClusWiSARD, in order to deal with more complex ones. Preprocessing methods for the data analysis are also discussed. For comparison purposes, the same data is classified by a Support Vector Machine.

This article is organized as follows: Section 2 presents the methods and materials used for this research, including classifier models (2.1, 2.2) as well as data set handling (2.3); Section 3 details metrics and implementation of the experiments, shows their results and discusses some of the interesting findings; Section 4 concludes this work summarising the findings and present possible avenues for further work.

2. Methodology

2.1. WiSARD

A Weightless Artificial Neural Network (WANN) is a pattern recognition system whose main difference from other learning methodologies lies on the direct use of information storage in Random Access Memories (RAMs) [21]. No error minimisation technique is used. WANN operation uses the input to build a set of addresses to access RAM nodes contents.

This work adopts WiSARD (**Wi**lkie **S**tonham and **A**leksander **R**ecognition **D**evice) [21], a pioneering WANN architecture that is composed by distinct sets of RAM nodes called *discriminators*. Each discriminator is assigned to one of the classes of patterns to be recognized, i.e., the number of discriminators in the WiSARD network is the same as the number of classes. A discriminator consists of a single layer of RAM nodes, which are all initialised with the default value zero (0) in every addressable position. The network has also been extended with a tie breaking capability, called *bleaching* [23], in order to deal with inconclusive pattern classifications. With respect to this work, WiSARD speed was very useful: all its operations have polylogarithmic complexity on the number of input observations. Additionally, that only requires a small number of parameters to be set.

2.1.1. Input Encoding

WiSARD is, originally, a Boolean neural network, so any input given to the architecture must be converted into a binary string. However, the most common description of data is by numerical and/or categorical attributes. A data conversion process must be applied to bridge this gap. This process may not be straightforward, as the similarity between any two observations should be preserved in the new representation. The preferred binary encodings for numeric features are the ones with a *Hamming distance* related to the numeric distance. Encodings which do not have this characteristic, e.g., IEEE 754 [24], should be avoided.

This conversion can be tuned with respect to a number of factors, such as domain knowledge and classifier performance on tests. After the conversion is made, the input is shuffled according to a fixed pseudorandom mask (defined at the creation of the network) and split to generate input addresses of all RAM nodes. During the training phase, some memory locations at the RAM nodes in the discriminator corresponding to the trained class are accessed according to each input pattern. Each access increments by one the value stored in the respective location. During the classification phase, every discriminator retrieves the information addressed by the input pattern. Each RAM node accessed this way outputs one (1) if the memory position in question holds a value higher than the bleaching threshold, and zero (0) otherwise. A discriminator response is the sum of the outputs of each of its RAM nodes, as seen in Figure 1. In a WiSARD multi discriminator arrangement, the discriminator with the highest response is chosen for the classification, as depicted by Figure 2. If two or more discriminators share the highest response then the bleaching threshold must be incremented by one and a new classification iteration is performed. Training and classification can be interleaved during runtime. By doing so, WiSARD can be employed in continuous (online) learning tasks.

2.1.2. Comparison with other Learning Models

WiSARD may resemble other learning models in some aspects, while being intrinsically different in others. Bayesian classifiers [25] also learn by counting occurrences of events regarding attributes values, but without explicit use of binary features. Curve fitting is a principle shared between a variety of models: support vector machine [22], multilayer perceptron [26] and others. WiSARD does not employ this strategy, being identified as a weightless neural network.

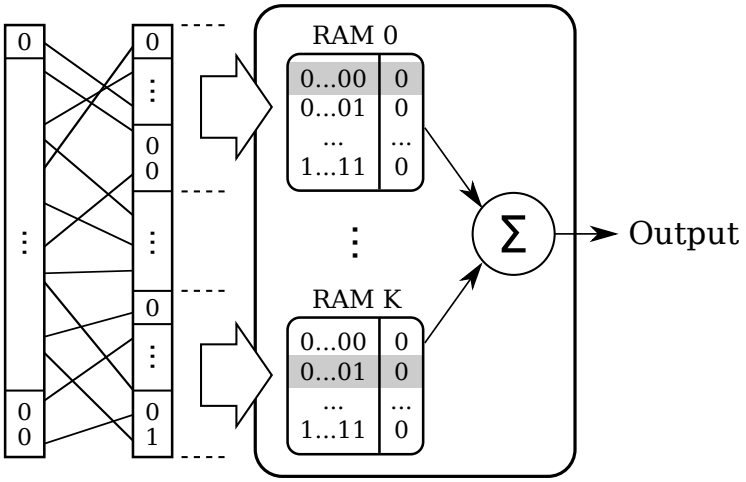


Figure 1: WiSARD discriminator (from left to right): binary input, pseudo random mapping, addressing RAM contents, summation and output.

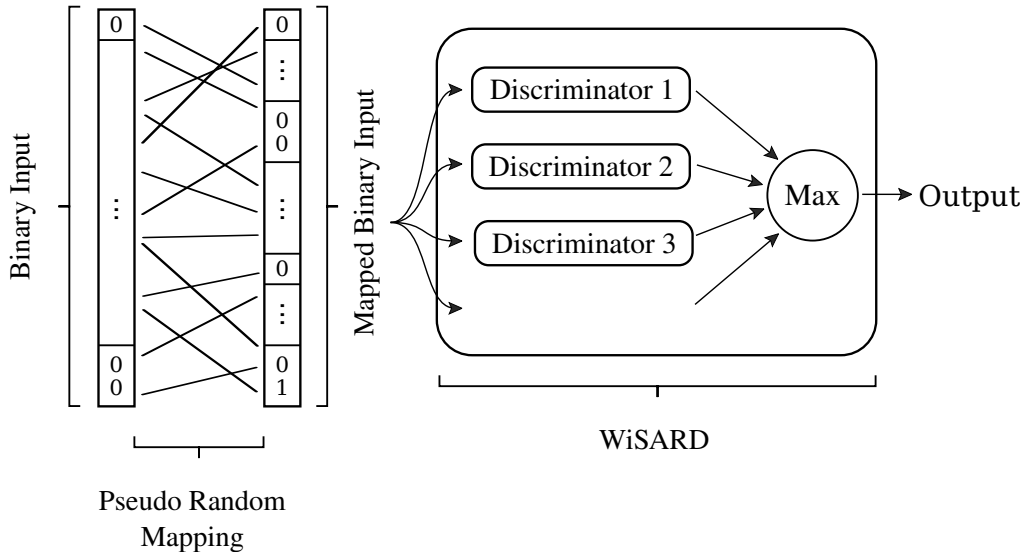


Figure 2: A WiSARD multidiscriminator system.

2.1.3. ClusWiSARD

The combination of very different input patterns of the same category in a WiSARD discriminator can enable the improper recognition of test patterns very dissimilar from those learned previously by this unit. This is a consequence of the multiplicative nature of WiSARD, that performs pattern classification based on the identification of previously seen patterns using stored collections (discriminators) of collections (RAM nodes) of features frequency logs (RAM contents). ClusWiSARD avoids this by creating clusters of input patterns, according to an acceptance threshold associated to each of them. This threshold is proportional to the number of elements in the cluster, i.e., the number of observations a discriminator has learnt. Dynamic thresholding works as a policy to prevent discriminator saturation, making harder the learning of new observations by a discriminator which possibly is capable of recognising any observation after numerous recordings.

Therefore, the main novelty of ClusWiSARD lies in its knowledge storage, which uses a group of discriminators per class. This allows for the distribution of training data into discriminators that better represent natural clusters, i.e., by capturing sub-patterns as "sub-classes". For each training input pattern, if any of the discriminators of this pattern class gives a recognition response higher than its acceptance threshold, the pattern is learned by such discriminator. If no discriminator accepts the new input pattern, a new discriminator is created to learn it. Classification with ClusWiSARD is similar to the original WiSARD: the input is tested with all discriminators; if there is a tie, a bleaching process occurs. The class of the discriminator with the highest response is chosen for the input. Figure 3 illustrates how ClusWiSARD works. Algorithm 1 describes ClusWiSARD training procedure.

It is worth noticing that $\text{score}(d, o)$ is nothing more than the recognition score a discriminator d provides to a given observation o. This operation is in its original form, having a computational cost of $O(|d|)$, where $|d|$ is the number of neurons in d. Therefore, in the worst case, each discriminator provides this score, resulting in a total training cost of $O(|d| \cdot |\mathcal{C}| \cdot |T|)$, where $|\mathcal{C}|$ is the number of discriminators the system keeps and $|T|$ is the number of data observations used during training.

Compared to WiSARD, ClusWiSARD introduces two new parameters: s, minimum score, and γ, threshold growth interval. s defines the minimum value of $\text{score}(d, o)$, which allows an observation o to be associated to a

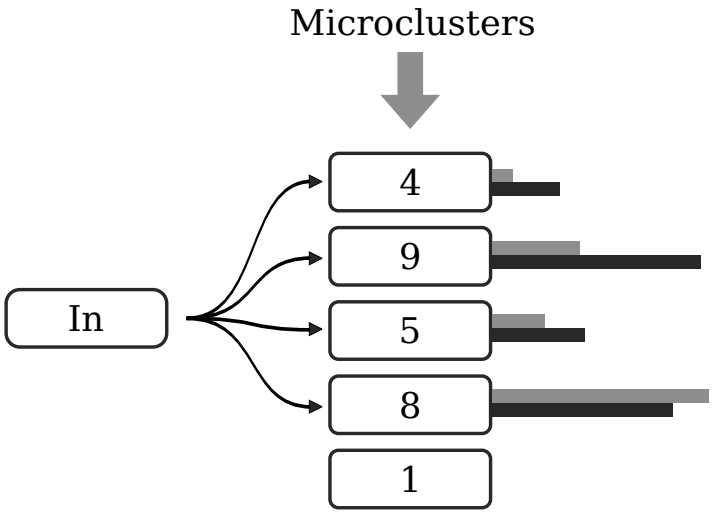


Figure 3: ClusWiSARD multidiscriminator system. An example of a pattern *In* being presented to the microclusters. Each microcluster contais a number corresponding to the quantity of samples stored in it. The darker bar on the right represents the cluster's threshold, while the gray bar corresponds to the pattern activation. Observe that the discriminator containing 8 samples is the one that will learn the new observation.

Algorithm 1 ClusWiSARD algorithm

Require: $s =$ `minimum score`
Require: $\gamma =$ `threshold growth interval`
Require: $T =$ `training data`
Ensure: $\mathcal{C}$ `is a trained ClusWiSARD classifier`

$\mathcal{C} \leftarrow$ `ClusWiSARD instance`
for each observation $o \in T$ **do**
    for each discriminator d currently in $\mathcal{C}$ **do** ▷ random order iteration
        if $\text{class}(o) = \text{class}(d)$ and $\text{score}(d, o) \geqslant \min(1, s + \text{size}(d)/\gamma)$ **then**
            Discriminator d learns observation o
            $\text{size}(d) \leftarrow \text{size}(d) + 1$
        end if
    end for
    if no discriminator learned observation o **then**
        A new discriminator d' is created
        d' is added to the collection of discriminators of $\mathcal{C}$
        $\text{class}(d') \leftarrow \text{class}(o)$
        Discriminator d' learns observation o
        $\text{size}(d') \leftarrow 1$
    end if
end for
Return $\mathcal{C}$

cluster d. In other words, it establishes the maximum distance between the boundaries of the cluster and an observation whose addition is being considered.

γ, the second parameter, is a consequence of the fact that as a discriminator learns more observations, the knowledge it keeps is probabilistically extended. From another point of view, the region of the feature space covered by the represented cluster has a chance to grow every time a new observation is included. This prompts an adjustment of the maximum "distance" mentioned in the previous paragraph. Therefore, the size of the clusters is closely related to the value of γ: as it gets lower, clusters reach their growth ceiling faster.

Setting the minimum score to zero and the threshold growth interval to infinity makes ClusWiSARD behave like the original WiSARD. In a good setup, these parameters avoid the definition of clusters with excessively dissimilar observations, as well as with few observations that do not characterize data patterns. In this work such parameters were set in an *ad hoc* manner.

2.2. SVM

The support vector machine (SVM) [22] constitutes a successful supervised learning model. Its main intuition lies on choosing a set of points to delimit a set of maximum-margin hyperplanes, capable of separating the input data in a class-nonclass fashion. This set of points is usually considerably small when compared to the training set and are called *support vectors*. A small set of support vectors implies in a small VC dimension, conferring a good generalisation capability to the system.

Credit analysis is a complex problem, whose data is not linearly separable. In such situations, SVM maps the input vectors x into a featural space Z through a nonlinear mapping chosen *a priori*. This mapping is also called the *kernel trick*. Some common kernel functions $k(\mathbf{x_i}, \mathbf{x_j})$ include: (*i*) linear: $k(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i}^\top \mathbf{x_j}$; (*ii*) polynomial: $k(\mathbf{x_i}, \mathbf{x_j}) = \left(\gamma \mathbf{x_i}^\top \mathbf{x_j} + c\right)^d$; (*iii*) gaussian radial basis function: $k(\mathbf{x_i}, \mathbf{x_j}) = e^{-\gamma \left\| \mathbf{x_i} - \mathbf{x_j} \right\|^2}$; and (*iv*) hyperbolic tangent: $k(\mathbf{x_i}, \mathbf{x_j}) = \tanh\left(\gamma \mathbf{x_i}^\top \mathbf{x_j} + c\right)$, where c, d and γ are parameters chosen *a priori*, and $\mathbf{x_i}$ and $\mathbf{x_j}$ are observations from the input sample.

Finally, some data are not separable due to noise. Using a higher-order kernel would overfit the model. Soft-margin SVM [22] is a variation of SVM capable of handling noise-affected data. In this variation of SVM, some data are not considered during the creation of the hyperplanes.

2.3. Datasets

This work uses two different data sets on credit analysis, with the purpose of evaluating different credit scoring scenarios:

2.3.1. UCI

The German Credit Data data set is part of the Statlog project, hosted at the UCI machine learning repository [27]. The data set consists of 1000 credit requests records, described by 20 attributes. These requests were classified as either of low (700 observations) or high (300 observations) risk by human experts. Given the lack of explicit rules to explain decisions, two different experts could classify the same applicant differently. This inconsistency could be considered a flaw of the financial institution operation. Therefore, a challenge presented by this data set is the reproduction of experts judgement by a computational learning system.

2.3.2. BRICS-CCI

This data set comes from the BRICS-CCI & CBIC Computer Intelligence Algorithm Competition [28]. It contains the data of credit applications from 2009 and 2010, labelled with the status of the approval. Temporal information is included in the dataset, enabling tracking of concept drift . Each entry corresponds to a credit request from a client of a private credit card retail chain, labeled as "good" or "bad" depending on the posterior payment behaviour. A client is considered "bad" if he or she defaults the debt for more than two consecutive months, and considered "good" otherwise. Attributes for each entry include age, gender, net income, among others, totalling 40 variables. However, some of the non-numerical original variables were converted into collections of simpler ones. Besides, new variables were generated after relating available data to publicly accessible external information. In the end, a total of 463 attributes was used.

2.4. Data Preprocessing

As in many applications, data preprocessing may improve the classifier performance. In this paper, three aspects of data analysis concerning preprocessing are covered: noisy data correction, attribute influence evaluation and optimal encoding. Next, a description of the data and the treatment proposed for each preprocessing aspect under consideration are presented.

Attributes could be divided into nominal, (categorical) ordinal and (interval) numerical ones. Nominal attributes values have no order and are

equidistant pairwise. The non-binary categorical features are encoded as a set of binary variables. If a non-numerical feature f may have n different values, then it is transformed into n binary features. A value of $f = i$ implies that the value of the i-th corresponding binary feature is 1 while the others are -1. A collection of values of an ordinal attribute can be ranked, and the similarity between two values is inversely proportional to their rank distance. Numerical values can be compared directly according to their difference.

As already stated in Subsection 2.1, the WiSARD model accepts a bit string as input. For this reason, given any input observation, the values of each of its attributes are transformed into bit strings by a procedure called *encoding*. The concatenation of these strings is then input to ClusWiSARD. A specific encoding procedure is applied to each attribute according to its type.

2.4.1. Numerical and Ordinal Attributes

Encoding of numerical attributes is performed as follows: values were first scaled to a $[0, x]$ interval and then rounded to nearest integer, where x is manually defined. Next, this integer was represented in the unary numeral system (base-1, thermometer). This unary representation with padding zeros on the left is the result of this conversion procedure. Ordinal attributes are encoded like numerical ones, after the substitution of the original values by their respective positions in the rank of all values of each attribute. Figure 4 illustrates the conversion to the unary system.

Such representation, and the scaling and rounding required to obtain it, means that the original attribute values were divided in x ordered intervals. Each interval is thus represented by a unique bit string. As an intended consequence of this conversion, the Hamming distance [29] between resulting binary representations of two values is proportional to their numerical difference.

The value of x is the actual number of bits that represent each numerical feature. It is empirically chosen by balancing the trade-off between information given to the classifier and complexity of the system. In other words, by determining the number of operations needed to be performed during each training or classification given the number of the RAM neurons in each discriminator. If a very small amount of bits is used to represent each feature, the classifying capability of ClusWiSARD may get impaired, i.e., its accuracy may decrease. On the other hand, if a very large number of bits is used, ClusWiSARD classifying capability will not be impaired, but time

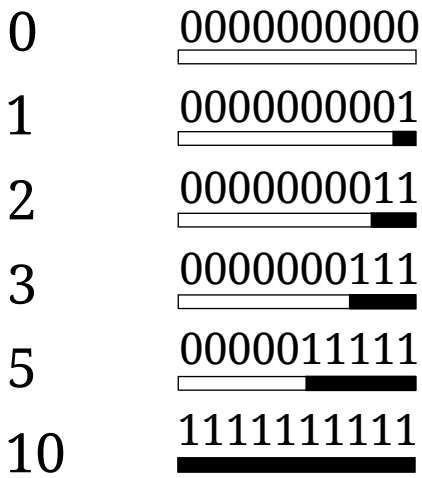


Figure 4: Example of unary (thermometer) encoding for a 10-bit bit string. Natural distance proportionality is preserved in the Hamming distance between the encoded values. Larger distances imply a greater contrast between values.

spent in the training phase may grow considerably. Since the number of RAM neurons grows linearly with the number of inputs (the addresses built from the inputs have a fixed length; see Section 2.1.1), the time spent in each operation will also increase proportionally.

2.4.2. Nominal Attributes

For a nominal attribute, the binary representations of its values are defined considering the following goal: to maximize the minimum Hamming distance between any pair of representations. Nominal values are as far as possible from each other, and that should be true regardless of how they are represented. The resulting bit strings must have a fixed length, and larger distances imply a greater contrast between bit strings, making them more distinguishable. The encoding procedure should account for that. Figure 5 illustrates a set of equidistant bit strings which are also as far as possible from each other.

As nominal attributes assume a small number of possible values in the credit analysis problem, a simple brute-force algorithm (Algorithm 2) was used to generate a set of tentatively equidistant bit strings of fixed length. Although the algorithm does not guarantee that the distance between all pairs are equal, it generates a set where the minimum distance between any pair of elements is as high as possible: through the algorithm iterations, a

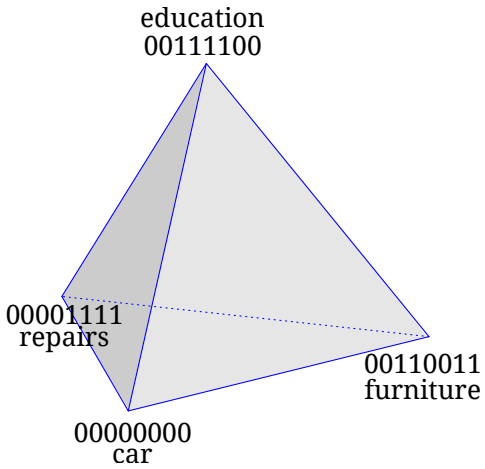


Figure 5: Example of equidistant set of 8-bit bit strings for a subset of the values of the attribute "credit purpose", with Hamming distance 4. The absence of natural order or distance between values is preserved by making the contrast between the encoded values equivalent pairwise.

constraint regarding the targeted minimum distance is relaxed as needed. This process can be repeated until the distance cannot be decreased or all the values are converted.

The encoding chosen for representing m possible values determines the minimum number of bits required for representing them. Regardless of the encoding, at least $n = \lceil log_2 m \rceil$ bits are necessary to assign a different code to each of the m values. To assure that the Hamming distance between any two codes is equal or greater than $2d$, $n = md$ is a trivial upper bound of the minimum value of n which allows such condition. However, it is possible to have lower values of n such that this condition is still reachable. The applicability of Algorithm 2 is directly related to such possibility.

2.4.3. Encoding Tunning

For both numerical and nominal features, an optimal length of their binary representations was chosen by repeating the experiments with increasingly larger values until classifying accuracy could no longer be improved. This was possible due to the low training time characteristic of ClusWiSARD, which was also exploited for feature selection on the UCI dataset experiment (Section 3.1).

Algorithm 2 Algorithm for encoding nominal attribute values

Require: n, `number of nominal values to encode`
Require: x, `length of the bit strings`
Ensure: s, `a set of bit strings to represent nominal values`

$t \leftarrow '000...0'$ ▷ t is a string of n zeroes
$s \leftarrow \{t\}$ ▷ s is a unitary set containing t
$n \leftarrow n - 1$
if $n > 0$ **then**
    $t \leftarrow '111...1'$ ▷ t is a string of n ones
    $s \leftarrow s \cup \{t\}$
    $n \leftarrow n - 1$
end if
$d \leftarrow \lfloor x/2 \rfloor$ ▷ Target minimum distance between elements of s
while $n > 0$ **do**
    for $v = 1$ `to` $2^n - 2$ **do**
        $w \leftarrow \texttt{bin}(v)$ ▷ w is the zero-padded binary representation of v
        $d' \leftarrow \min_{x \in s} \{\texttt{hamming\_distance}(w, x)\}$
        if $d' = d$ **then**
            $s \leftarrow s \cup \{w\}$
            $n \leftarrow n - 1$
            if $n = 0$ **then**
                return ▷ Done
            end if
        end if
    end for
    $d \leftarrow d - 1$ ▷ Relaxing target minimum distance
end while

It is also reasonable to expect that the attributes contribute differently to the classification task and that the dataset contains a large number of attributes with small or no contribution. To address this and possibly reduce the number of attributes, a study was conducted on the degree of influence of each attribute over payment behaviour. The information gain ratio [30] was used to adjust input encoding to ClusWiSARD, varying the number of bits to represent each attribute according to the calculated ratio. The same procedure was used to filter which attributes should be used to describe data which was input to SVM, whenever using using all data is unfeasible. Figure 6 shows an example of this filtering, applied to the BRICS-CCI dataset.

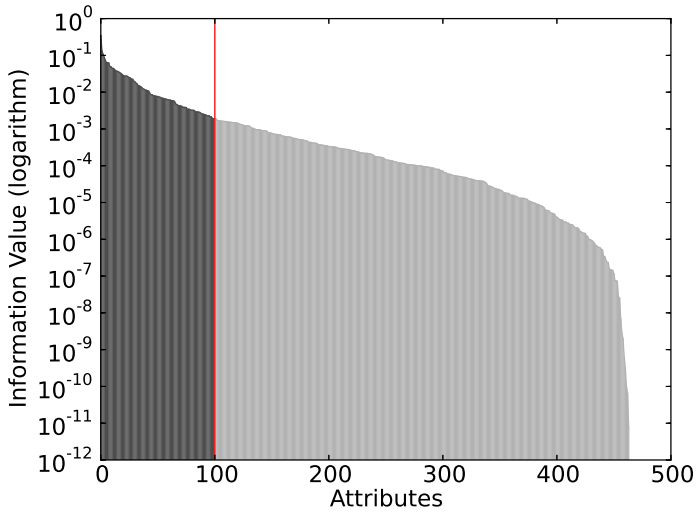


Figure 6: Information gain ratio of the attributes of the BRICS-CCI dataset. The first one hundred attributes having the highest information values were chosen.

3. Experimental evaluation

On both data sets SVM and ClusWiSARD were compared with respect to binary classification performance and speed. The following SVM parameters were adjusted: degree, C and kernel function. The first two were adjusted after testing different values in a range, while the kernel function was manually chosen. The address length of the RAM nodes of ClusWiSARD was also chosen after testing a set of options in a range.

Accuracy and F_1 score were adopted as metrics. The F_1 is intended to tackle the imbalance between the two classes: accuracy might allow a higher score if the classifier favours the class with more elements. Two F_1 scores were used, since the F_1 score depends on a single class defined as the gold standard and because this standard depends on the reason of the classification process. These score are F_1 (good) and F_1 (bad), which are the F_1 score values when the good and the bad payers respectively are the gold standard. Additionally, a weighted F_1 score is used as a criterion for a gold-standard-independent feature selection. Considering Figure 7 values, these are the mathematical definitions of the metrics:

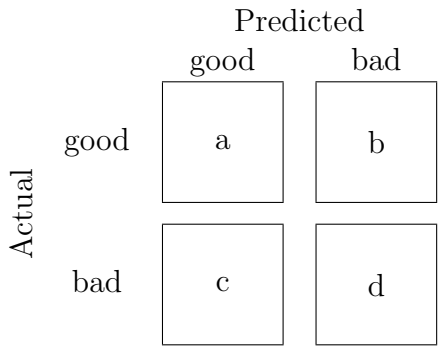


Figure 7: Confusion matrix structure

$$\texttt{Accuracy} = \frac{a+d}{a+b+c+d}$$

$$\texttt{F}_1\ \texttt{(good)} = G = \frac{2a}{2a+b+c}$$

$$\texttt{F}_1\ \texttt{(bad)} = B = \frac{2d}{2d+b+c}$$

$$\texttt{Bad ratio} = \beta = \frac{a+b}{a+b+c+d}$$

$$\texttt{Good ratio} = \gamma = \frac{c+d}{a+b+c+d}$$

$$\texttt{Weighted F}_1\ \texttt{score} = \gamma G + \beta B$$

3.1. UCI

Compared to the BRICS-CCI dataset, this data set has a small number of variables and observations. This pair of conditions enabled brute-force feature selection. Table 1 compares the results found by the ClusWiSARD and the SVM classifier using the five best combinations of attributes found exhaustively; this was only possible due to ClusWiSARD's agility. The classifiers performance when using all attributes is shown in the last line of Table 1. While both models presented higher accuracies and F_1 scores when using the best selected attributes, it was the ClusWiSARD model who benefited most in both metrics. Values presented in Table 1 are the average results of ten thousand random 10-fold cross validation runs. Table 2 presents the confusion matrix for each model using the best selection of attributes that produced the best results.

Rank	ClusWiSARD			SVM		
	Accuracy	F_1 (Good)	F_1 (Bad)	Accuracy	F_1 (Good)	F_1 (Bad)
1	0.767	0.841	0.563	0.765	0.843	0.540
2	0.766	0.841	0.554	0.767	0.844	0.542
3	0.763	0.839	0.556	0.763	0.845	0.494
4	0.765	0.841	0.552	0.761	0.840	0.533
5	0.759	0.836	0.544	0.722	0.848	0.547
All	0.754	0.841	0.458	0.770	0.849	0.522

Table 1: UCI dataset: comparison between ClusWiSARD and SVM. Five best combinations of attributes followed by all attributes accuracies and F_1 scores for each class.

		Predicted			
		ClusWiSARD		SVM	
		Good	Bad	Good	Bad
Actual	Good	620	80	625	75
	Bad	155	154	164	136

Table 2: Confusion matrix of UCI dataset for the best combination of attributes.

3.2. BRICS-CCI

Credit data is influenced by macroeconomical factors and events, which could be seasonal or not. What characterizes a credit request as good or bad is related to when it was submitted, this is called *concept drift*. In order to deal with this factor, the performance of the classifiers was evaluated using the observations of each month in a two-year span; training of monthly observations is performed using data from the previous three months. Therefore, the first three months of 2009 were not used as test targets.

In order to determine the best SVM parameters for this specific problem, a second dataset composed of 12000 observations was created. This dataset was randomly generated from the original one with five hundred observations for each month. Once this dataset was ready, multiple instances of SVM networks were trained varying its parameters. These consisted of the kernel function used (Polynomial, RBF and Sigmoid) along with their configuration parameters: γ, $coef0$ and $degree$ for Polynomial; γ for RBF; and γ and $coef0$ for the Sigmoid kernel. In the end, the polynomial kernel function presented the best results. The configuration had a $degree$ of 3, $coef0$ equal to zero and the value for γ was approximately 0.01509, according to the equation:

$$k(u, v) = (\gamma * u^T * v + coef0)^{degree} \tag{1}$$

The parameters u and v from the kernel function are two samples from the training set passed to the kernel. An extended description about the meaning of the kernel function and how they are used can be found in [22]. Both classifiers were optimised according to the average per-class accuracy. This was preferred over the general accuracy because of the great difference in the dataset share between classes, which, however, does not represent a greater significance of one class compared to the other. Notice that, predicting all observations as being of the biggest ("good") class would result in a good general accuracy. On the other hand, a high average per-class accuracy could also be a sign of overfitting.

As seen in Figure 8a, SVM performs very competitively with ClusWiSARD in the BRICS-CCI dataset when considering the final accuracy, aside from the first four months. During these first months, the latter performed considerably better. However, despite losing in general accuracy SVM shows consistently better discrimination capabilities for bad payers, as seen in Figure 8c, which shows the F_1 scores obtained when the bad payers are the gold standard. On the other hand, ClusWiSARD shows a better discrimination

capability for good payers, especially on the first four and on the last seven months, as depicted in Figure 8b. Table 3 shows a more detailed representation of the values presented in Figure 8 for reference purposes. Table 4 presents the confusion matrix for each model.

Additionally, Figure 9 shows that ClusWiSARD performed clearly faster than SVM in both training and classification times, especially during training step. Figure 9a shows that the training time of ClusWiSARD grows linearly with the number of observations and is 3 orders of magnitude lower than the one that of SVM, which grows exponentially. Nonetheless, it is important to note that SVM internally adjusts the relative contribution of each attribute to the classification function, while WiSARD and ClusWiSARD use the previously calculated information gain ratio for this end, as explained in Subsection 2.4.

Furthermore, Figure 10 shows a comparison using different fractions of the BRICS-CCI dataset from its original size to 1/1024 of it. Figure 10a illustrates time spent during training, while Figure 10b shows the same information for classification. In both situations the plots show that the time spent by SVM increases faster.

4. Conclusion

Credit analysis is an interesting problem since many important challenges such as large datasets, concept drift, class imbalance, and noisy data, are present. Data preprocessing was a very important step in the classification job performed both by ClusWiSARD and by SVM. The transformation of input data into binary patterns, a step required for weightless neural models, like ClusWiSARD, brings another interesting dimension to the target problem.

ClusWiSARD presented accurate results that were close to SVM, a natural choice as a classifier standard. Besides, ClusWiSARD displayed the ability of dealing with class imbalance as well. The main contribution of this work is to show that ClusWiSARD is able to achieve online training agility whilst keeping SVM-like performance.

5. Acknowledgement

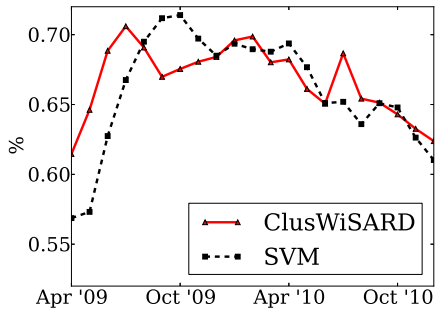
This work was partially supported by Inovax, GE, and CAPES, CNPq, FAPERJ and FINEP Brazilian research agencies.

Month &	ClusWiSARD			SVM		
Year	Accuracy	F_1 (Good)	F_1 (Bad)	Accuracy	F_1 (Good)	F_1 (Bad)
Apr '09	0.613	0.713	0.406	0.626	0.538	0.596
May '09	0.643	0.749	0.383	0.647	0.604	0.537
Jun '09	0.693	0.799	0.351	0.694	0.708	0.484
Jul '09	0.705	0.811	0.329	0.704	0.757	0.472
Aug '09	0.690	0.793	0.380	0.685	0.783	0.487
Sep '09	0.667	0.771	0.392	0.655	0.799	0.491
Out '09	0.671	0.778	0.366	0.660	0.803	0.476
Nov '09	0.682	0.785	0.387	0.676	0.785	0.490
Dez '09	0.683	0.788	0.373	0.684	0.774	0.479
Jan '10	0.697	0.801	0.367	0.688	0.786	0.458
Feb '10	0.702	0.806	0.365	0.696	0.783	0.455
Mar '10	0.682	0.787	0.374	0.675	0.779	0.469
Apr '10	0.682	0.786	0.381	0.680	0.784	0.475
May '10	0.659	0.759	0.417	0.656	0.756	0.520
Jun '10	0.649	0.752	0.398	0.641	0.730	0.506
Jul '10	0.688	0.792	0.371	0.683	0.737	0.484
Aug '10	0.656	0.758	0.404	0.650	0.712	0.505
Sep '10	0.653	0.756	0.402	0.644	0.732	0.500
Out '10	0.643	0.745	0.403	0.641	0.721	0.523
Nov '10	0.631	0.730	0.420	0.631	0.680	0.551
Dez '10	0.622	0.715	0.441	0.618	0.657	0.549

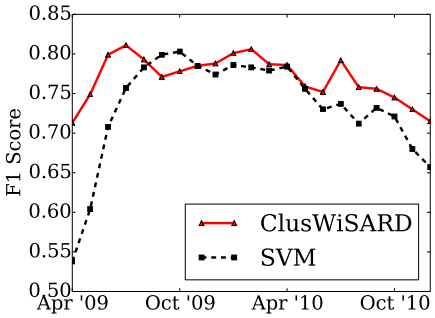
Table 3: BRICS-CCI dataset: comparison between ClusWiSARD and SVM.

		Predicted			
		ClusWiSARD		SVM	
		Good	Bad	Good	Bad
Actual	Good	394126	126150	341475	178801
	Bad	112595	77485	64134	125946

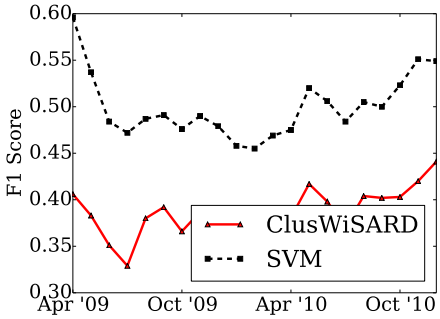
Table 4: Confusion matrix of BRICS-CCI dataset.



(a) General accuracy

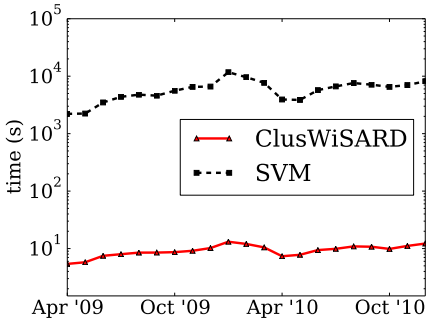


(b) F_1 score for good payers

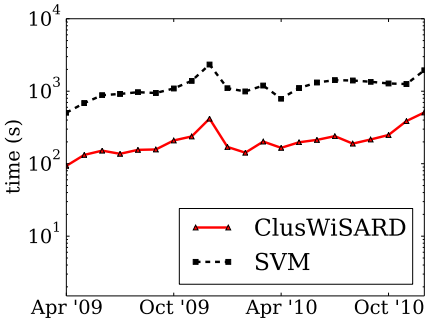


(c) F_1 score for bad payers

Figure 8: Classification performance measures: SVM and ClusWiSARD over the BRICS-CCI dataset.

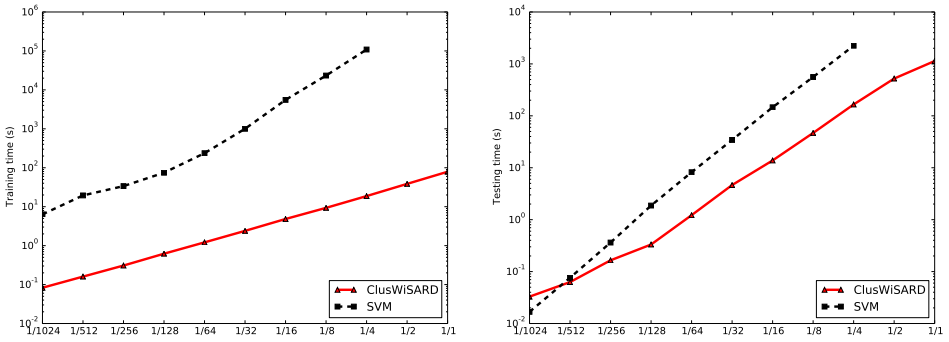


(a) Training time



(b) Classification time

Figure 9: Time spent: SVM and ClusWiSARD over the BRICS-CCI dataset.



(a) Training time (b) Classification time

Figure 10: Time spent: SVM and ClusWiSARD over reduced BRICS-CCI dataset.

References

[1] T. Joro, P. Na, Derivatives and credit risk: Credit risk modeling for catastrophic events, WSC '02, Winter Simulation Conference, 2002. URL http://dl.acm.org/citation.cfm?id=1030453.1030674

[2] H.-M. Lu, F.-T. Tsai, H. Chen, M.-W. Hung, S.-H. Li, Credit rating change modeling using news and financial ratios, ACM Trans. Manage. Inf. Syst. 3 (3) (2012) 14:1–14:30. doi:10.1145/2361256.2361259. URL http://doi.acm.org/10.1145/2361256.2361259

[3] C.-F. Tsai, Combining cluster analysis with classifier ensembles to predict financial distress, Information Fusion 16 (2014) 46–58.

[4] C.-F. Tsai, Financial decision support using neural networks and support vector machines, Expert Systems 25 (4) (2008) 380–393.

[5] D. Martens, B. Baesens, T. Van Gestel, J. Vanthienen, Comprehensible credit scoring models using rule extraction from support vector machines, European journal of operational research 183 (3) (2007) 1466–1476.

[6] Y. Yang, Adaptive credit scoring with kernel learning methods, European Journal of Operational Research 183 (3) (2007) 1521–1536.

[7] K.-S. Shin, T. S. Lee, H.-j. Kim, An application of support vector machines in bankruptcy prediction model, Expert Systems with Applications 28 (1) (2005) 127–135.

[8] J. H. Min, Y.-C. Lee, Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters, Expert systems with applications 28 (4) (2005) 603–614.

[9] A. Khashman, A neural network model for credit risk evaluation, International Journal of Neural Systems 19 (04) (2009) 285–294.

[10] K. Lee, D. Booth, P. Alam, A comparison of supervised and unsupervised neural networks in predicting bankruptcy of korean firms, Expert Systems with Applications 29 (1) (2005) 1–16.

[11] T.-S. Lee, C.-C. Chiu, Y.-C. Chou, C.-J. Lu, Mining the customer credit using classification and regression tree and multivariate adaptive regression splines, Computational Statistics & Data Analysis 50 (4) (2006) 1113–1130.

[12] T. Lensberg, A. Eilifsen, T. E. McKee, Bankruptcy theory development and classification via genetic programming, European Journal of Operational Research 169 (2) (2006) 677–697.

[13] C.-S. Ong, J.-J. Huang, G.-H. Tzeng, Building credit scoring models using genetic programming, Expert Systems with Applications 29 (1) (2005) 41–47.

[14] A. Ghodselahi, A hybrid support vector machine ensemble model for credit scoring, International Journal of Computer Applications 17 (5) (2011) 1–5.

[15] F. Hoffmann, B. Baesens, C. Mues, T. Van Gestel, J. Vanthienen, Inferring descriptive and approximate fuzzy rules for credit scoring using evolutionary algorithms, European Journal of Operational Research 177 (1) (2007) 540–555.

[16] A. Tsakonas, G. Dounias, M. Doumpos, C. Zopounidis, Bankruptcy prediction with neural logic networks by means of grammar-guided genetic programming, Expert Systems with Applications 30 (3) (2006) 449–461.

[17] C.-H. Wu, G.-H. Tzeng, Y.-J. Goo, W.-C. Fang, A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy, Expert systems with applications 32 (2) (2007) 397–408.

[18] L. C. Thomas, J. Crook, D. Edelman, Credit Scoring and Its Applications, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[19] M. Vojtek, E. Kočenda, Credit-scoring methods (in english), Czech Journal of Economics and Finance (Finance a uver) 56 (3–4) (2006) 152–167.

[20] H. Kimura, L. F. C. Basso, E. K. Kayo, Decision models in credit risk management, in: P. Guarnieri (Ed.), Decision Models in Engineering and Management, Decision Engineering, Springer International Publishing, 2015, p. 57–73.

[21] I. Aleksander, W. V. Thomas, P. A. Bowden, WISARD: A radical step foward in image recognition, Sensor Review 4 (1984) 120–124.

[22] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297. doi:10.1007/BF00994018.
URL http://dx.doi.org/10.1007/BF00994018

[23] B. P. A. Grieco, P. M. V. Lima, M. D. Gregorio, F. M. G. França, Producing pattern examples from "mental" images, Neurocomputing 73 (7-9) (2010) 1057–1064.

[24] Ieee standard for floating-point arithmetic, IEEE Std 754-2008 (2008) 1–70doi:10.1109/IEEESTD.2008.4610935.

[25] P. Langley, W. Iba, and, K. Thompson, An analysis of bayesian classifiers, in: Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI'92, AAAI Press, 1992, pp. 223–228.
URL http://dl.acm.org/citation.cfm?id=1867135.1867170

[26] A. Lapedes, R. Farber, How neural nets work, in: D. Z. Anderson (Ed.), Neural information processing systems, AIP, New York, 1987, p. 442.

[27] K. Bache, M. Lichman, UCI machine learning repository (2013).
URL http://archive.ics.uci.edu/ml

[28] NeuroTech S.A. (Brazil), CI algorithms competition dataset, http://brics-cci.org/ci-algorithms-competition-ciac/ (2013).

[29] R. W. Hamming, Error detecting and error correcting codes, Bell System Technical Journal 26 (2) (1950) 147–160.

[30] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003) 1157–1182.
URL http://dl.acm.org/citation.cfm?id=944919.944968