Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial
ISSN: 1137-3601
revista@aepia.org
Asociación Española para la Inteligencia Artificial
España

Alonso González, Carlos J.; Rodríguez Diez, Juan J.
Time Series Classification by Boosting Interval Based Literals
Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, vol. 4, núm. 11, otoño, 2000, pp. 2-11
Asociación Española para la Inteligencia Artificial
Valencia, España

Time Series Classification by Boosting Interval Based Literals *

Carlos J. Alonso González **Juan J. Rodríguez Diez**

Grupo de Sistemas Inteligentes
Departamento de Informática
Universidad de Valladolid
{calonso,juanjo}@infor.uva.es

Abstract

A supervised classification method for temporal series, even multivariate, is presented. It is based on boosting very simple classifiers: clauses with one literal in the body. The background predicates are based on temporal intervals. Two types of predicates are used: i) relative predicates, such as "increases" and "stays", and ii) region predicates, such as "always" and "sometime", which operate over regions in the dominion of the variable. Experiments on different data sets, several of them obtained from the UCI repositories, show that the proposed method is highly competitive with previous approaches.

Keywords: time series classification, interval based literals, boosting, machine learning.

1 Introduction

Multivariate time series classification is useful in domains such as biomedical signals [10], continuous systems diagnosis [2] and data mining in temporal databases [5]. This problem can be tackled by extracting features of the series through some kind of preprocessing, and using some conventional machine learning method. However, this approach has several drawbacks [9]: the preprocessing techniques are usually *ad hoc* and domain specific, there are several heuristics applicable to temporal domains that are difficult to capture by a preprocess and the descriptions obtained using these features can be hard to understand. The design of specific machine learning methods for the induction of time series classifiers allows the construction of more comprehensible classifiers in a more efficient way.

When learning multivariate time series classifiers, the input consists of a set of training examples and associated class labels, where each example is formed by one or more time series. The series are often referred to as variables, since they vary over time. From a machine learning point of view, each point of each series is an attribute of the example.

The method for learning time series classifiers that we propose in this work is based on literals over temporal intervals (such as increases or always in region) and boosting (a method for the generation of ensembles of classifiers) [14].

The rest of the paper is organized as follows. Section 2 is a brief introduction to boosting, suited to our method. The base classifiers are described

*This work has been supported by the Spanish CYCIT project TAP 99–0344.

in section 3, including techniques for efficiently handling these special purpose predicates. Section 4 presents experimental results when using the new method. Finally, we give some concluding remarks in section 5.

2 Boosting

At present, an active research topic is the use of *ensembles* of classifiers. They are obtained by generating and combining base classifiers, constructed using other machine learning methods. The target of these ensembles is to increase the accuracy with respect to the base classifiers.

One of the most popular methods for creating ensembles is boosting [14], a family of methods, of which AdaBoost is the most prominent member. They work by assigning a weight to each example. Initially, all the examples have the same weight. In each iteration a base classifier is constructed, according to the distribution of weights. Afterwards, for each example it weight is readjusted, depending on the correctness of the class assigned to the example by the base classifier. The final result is obtained by weighted votes of the base classifiers.

Inspired by the good results of works using ensembles of very simple classifiers [14], sometimes named *stumps*, we have studied base classifiers consisting of clauses with only one literal in the body.

Multiclass problems There are several methods of extending AdaBoost to the multiclass case [14]. We have used AdaBoost.OC [13] since it can be used with any weak learner which can handle binary labeled data. It does not require that the weak learner can handle multilabeled data with high accuracy. The key idea is, in each round of the boosting algorithm, to select a subset of the set of labels, and train the binary weak learner with the examples labeled positive or negative depending if the original label of the example is or is not in the subset. In our concrete case, the base learner searches for a rule with the head:

class(Example, [class_1, ... class_k])

This predicate means that the Example is of one of the classes in the list. Figure 1 shows a fragment of one of this classifiers.

The classification of a new example is obtained from a weighted vote of the results of the weak classifiers. For each rule, if its antecedent is true the weights of all the labels in the list are increased by the weight of the rule, if it is false the weights of the labels out of the list are incremented. Finally, the label that has been given the highest weight is assigned to the example.

3 Base Classifiers

3.1 Temporal Predicates

Figure 2 shows a classification of the predicates. Point based predicates use only one point of the series:

- point_region(Example, Variable, Region, Point) it is true if, for the Example, the value of the Variable at the Point is in the Region.

Note that a learner which only uses this predicate is equivalent to an attribute-value learning algorithm. This predicate is introduced to test the results obtained with boosting without using interval based predicates.

Two kinds of interval predicates are used: relative and region based. Relative predicates consider the differences between the values in the interval. Region based predicates are based on the presence of the values of a variable in a region during an interval.

3.1.1 Relative Predicates

A natural way of describing series is to indicate when they increase, decrease or stay. These predicates deal with these concepts:

- increases(Example, Variable, Beginning, End, Value). It is true, for the Example, if the difference between the values of the Variable for End and Beginning is greater or equal than Value.

```
class( E, [ thank, maybe, name, man, science ] ) :- true_percentage( E, z, 1_4, 12, 16, 70 ). % 79, 30. 0.312225
class( E, [ thank, science, right, maybe, read ] ) :- true_percentage( E, roll, 1_4, 2, 10, 50 ). % 77, 8. 0.461474
class( E, [ maybe, right, man, come, thank ] ) :- true_percentage( E, z, 1_3, 2, 18, 50 ). % 70, 18. 0.339737
class( E, [ thank, come, man, maybe, mine ] ) :- true_percentage( E, roll, 5, 6, 14, 50 ). % 64, 27. 0.257332
class( E, [ girl, name, mine, right, man ] ) :- not true_percentage( E, z, 1_2, 6, 14, 5 ). % 78, 22. 0.361589
```

Figure 1: Initial fragment of an ensemble of classifiers, obtained with AdaBoost.OC, for the dataset *Auslan* (Sect. 4.1). At the right of each clause the number of positive and negative covered examples and the weight of the classifier.

- decreases(Example, Variable, Beginning, End, Value).
- stays(Example, Variable, Beginning, End, Value). It is true, for the Example, if the range of values of the Variable in the interval is less or equal than Value.

Frequently, series are noisy and, hence, a strict definition of increases and decreases in an interval, i.e., the relation holds for all the points in the interval, is not useful. It is possible to filter the series prior to the learning process, but we believe that a system for time series classification must not rely on the assumption that the data is clean. For these two predicates we have opted for consider what happens only in the extremes of the interval. The parameter value is necessary for indicating the amount of change.

For the predicate stays neither is useful to use a strict definition. In this case all the points in the interval are considered. The parameter Value is used to indicate the maximum allowed difference between the values in the interval.

3.1.2 Region Based Predicates

The selection and definition of these predicates is based in the ones used in a visual rule language for dynamic systems [2] and it is introduced in [11]. These predicates are:

- always(Example, Variable, Region, Beginning, End). It is true, for the Example, if the Variable is always in this Region in the interval between Beginning and End.
- sometime(Example, Variable, Region, Beginning, End).
- true_percentage(Example, Variable, Region, Beginning, End, Percentage). It is true, for the Example, if the percentage of the time between Beginning and End where the variable is in Region is greater or equal to Percentage.

Once that it is decided to work with temporal intervals, the use and definition of the predicates always and sometime is natural, due to the fact that they are the extension of the conjunction and disjunction to intervals. Since one appears too demanding and the other too flexible, a third one has been introduced, true_percentage. It is a "relaxed always" (or a "restricted sometime"). The additional parameter indicates the degree of flexibility (or restriction).

Regions. The regions that appear in the previous predicates are intervals in the domain of values of the variable. In some cases the definitions of these regions can be obtained from an expert, as background knowledge. Otherwise, they can be obtained with a discretization preprocess, which obtains r disjoint, consecutive intervals. The regions considered are these r intervals (equality tests) and others formed by the union of the intervals $1 \dots i$ (less or equal tests).

The reasons for fixing the regions before the classifier induction, instead of obtaining them while inducing, are efficiency and comprehensibility. The literals are easier to understand if the regions are few, fixed and not overlapped.

3.2 Searching Literals

The base learner receives a set of examples, labeled as positive or negative. Its mission is to find the best literal for discriminating positive from negative examples. Then it is necessary to search over the space of literals. For each literal considered it is necessary to know which positive and negative examples are true.

The possible number of intervals, if each series

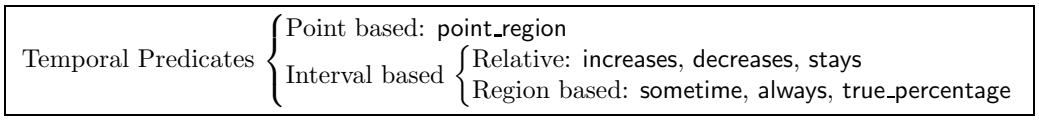

Figure 2: Classification of the predicates.

has n points, is $(n^2 - n)/2$. With the objective of reducing the search space, not all the windows are explored. Only those that are of size power of 2 are considered. The number of these windows is $\sum_{i=1}^{k}(n - 2^{i-1}) = kn - 2^k - 1$ where $k = \lfloor \lg n \rfloor$.

Given an interval and an example the predicates increases and decreases can be evaluated in $O(1)$, because only the extremes of the interval are considered. For the other interval predicates this time is $O(w)$, where w is the number of points in the interval.

Given a predicate, for searching the best literal it is necessary, for each considered interval, to calculate how many examples of each class are true. A simple method for this would be to consider all the intervals and for each interval and example to evaluate the literal.

A better method is possible, if the relationships among intervals are considered. When a literal is evaluated, some information is saved for posterior use. Then, the evaluation of a literal with an interval of width $2w$ is obtained from the previous evaluation of two literals whose intervals are consecutive and with widths w.

For example, consider the predicate true_percentage. For evaluating it, two values are necessary: l the length of the interval and s the sum of the lengths of the sub-intervals in the interval where the value is in the region. If there are two consecutive intervals with lengths l_1 and l_2 and sums s_1 and s_2, the union interval has length $l_1 + l_2$ and sum $s_1 + s_2$. In this way, for each example and interval only a time of $O(1)$ is necessary to calculate the percentage associated. For each interval the best threshold (from the allowed values) for the parameter Percentage of the predicate is selected.

The selection of the best literal is linear in the number of examples, the number of variables, the number of regions (for region based predicates) and the number of intervals (which is $O(n \lg n)$). There are also additional costs for selecting the best additional parameters for some predicates (e.g. the parameter Percentage of true_percentage) but they are linear in the number of values allowed.

4 Experimental Validation

4.1 Datasets

The characteristics of the datasets are summarized in table 1. Datasets for classification of time series are not easy to find [9]. For this reason we have used four artificial datasets and only one "real world" dataset:

Cylinder, Bell and Funnel (CBF). This is an artificial problem, introduced in [12]. The learning task is to distinguish between three classes: cylinder (c), bell (b) or funnel (f). Examples are generated using the following functions:

$$c(t) = (6 + \eta) \cdot \chi_{[a,b]}(t) + \epsilon(t)$$
$$b(t) = (6 + \eta) \cdot \chi_{[a,b]}(t) \cdot (t - a)/(b - a) + \epsilon(t)$$
$$f(t) = (6 + \eta) \cdot \chi_{[a,b]}(t) \cdot (b - t)/(b - a) + \epsilon(t)$$

where

$$\chi_{[a,b]} = \begin{cases} 0 & \text{if} \quad t < a \vee t > b \\ 1 & \text{if} \quad a \leq t \leq b \end{cases}$$

	Classes	Examples	Points	Variables
CBF	3	798	128	1
Control charts	6	600	60	1
Waveform	3	900	21	1
Wave + noise	3	900	40	1
Auslan	10	200	20	8

Table 1: Characteristics of the datasets

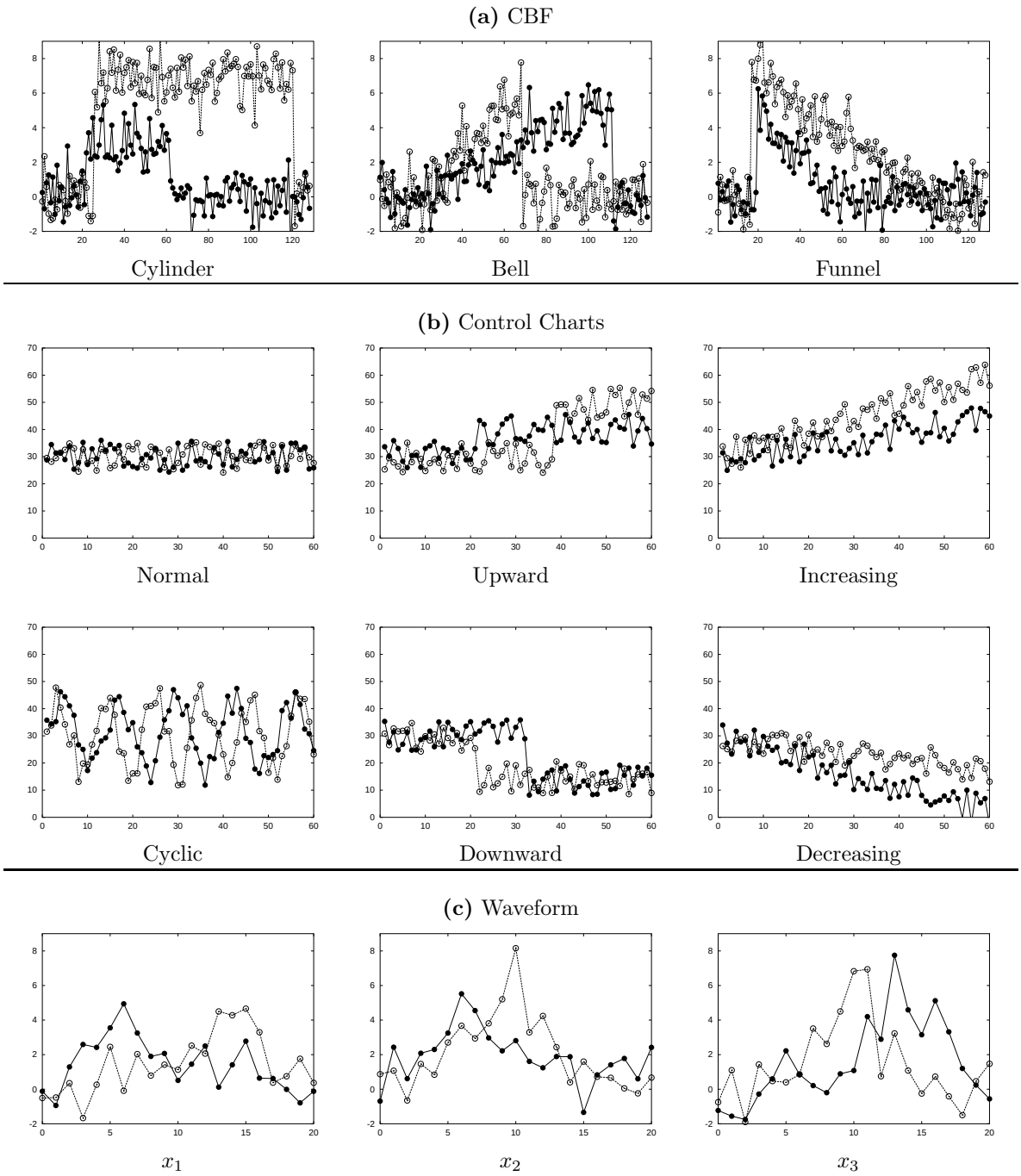

Figure 3: Some examples of the datasets. Two examples of the same class are shown in each graph.

and η and $\epsilon(t)$ are obtained from a standard normal distribution $N(0,1)$, a is an integer obtained from a uniform distribution in $[16,32]$ and $b-a$ is another integer obtained from another uniform distribution in $[32,96]$. The examples are generated evaluating those functions for $1,2\ldots128$. Figure 3.a shows two examples of each class.

Control Charts. In this dataset there are six different classes of control charts, synthetically generated by the process in [1]. Each time series is of length n, and is defined by $y(t)$, with $1 \leq t \leq n$:

1. Normal: $y(t) = m + rs$. Where $m = 30$, $s = 2$ and r is a random number in $[-3,3]$.
2. Cyclic: $y(t) = m + rs + a\sin(2\pi t/T)$. a and T are in $[10,15]$.
3. Increasing: $y(t) = m + rs + gt$. g is in $[0.2,0.5]$.
4. Decreasing: $y(t) = m + rs - gt$.
5. Upward: $y(t) = m + rs + kx$. x is in $[7.5,20]$ and $k = 0$ before time t_3 and 1 after this time. t_3 is in $[n/3, 2n/3]$.
6. Downward: $y(t) = m + rs - kx$.

Figure 3.b shows two examples of three of the classes. The data used was obtained from the UCI KDD Archive [4].

Waveform. This dataset was introduced by [7]. The purpouse is to distinguish between three classes, defined by the evaluation in $1,2\ldots21$, of the following functions:

$$\begin{aligned} x_1(i) &= uh_1(i) + (1-u)h_2(i) + \epsilon(i) \\ x_2(i) &= uh_1(i) + (1-u)h_3(i) + \epsilon(i) \\ x_3(i) &= uh_2(i) + (1-u)h_3(i) + \epsilon(i) \end{aligned}$$

where $h_1(i) = \max(6-|i-7|, 0)$, $h_2(i) = h_1(i-8)$, $h_3(i) = h_1(i-4)$, u is a uniform aleatory variable in $(0,1)$ and $\epsilon(t)$ follows a standard normal distribution.

Figure 3.a shows two examples of each class. We used the version from the UCI ML Repository [6].

	1	2	3	4	5
point_region	•		○	○	○
increases		•			•
decreases		•			•
stays		•			•
always			•	○	○
sometime			•	○	○
true_percentage				•	•

Table 2: Predicates used in each experimental setting. The symbol '•' indicates that the predicate is used in the experiment, and '○' indicates that the predicate is not used but there is another one that includes it.

Wave + Noise. This dataset is generated in the same way than the previous one, but 19 random points are added at the end of each example, with mean 0 and variance 1. Again, we used the dataset from the UCI ML Repository.

Auslan. Auslan is the Australian sign language, the language of the Australian deaf community. Instances of the signs were collected using an instrumented glove [9]. Each example is composed by 8 series: x, y and z position, wrist roll, thumb, fore, middle and ring finger bend. There are 10 classes and 20 examples of each class. The number of points in each example is variable and currently the system does not support variable length series, so they were resampled to 20 points.

4.2 Results

The results for each dataset were obtained using five ten-fold stratified cross-validation. The number of regions for each variable was 6. The values considered for the parameter value of relative predicates were multiples of the range of the variable divided by 20. The allowed values for the parameter percentage where 5, 15, 30, 50, 70, 85 and 95. Table 2 shows which predicates are used in the different experimental settings. Table 3 and figure 4 resume the results.

Globally, we can highlight the good evolution of the maximum error for each dataset with the number of iterations in the boosting process. Many times, in Table 3, the best results are obtained in the last iteration. For these datasets, and until the number of iterations considered, we can say that the method is rather robust to over-

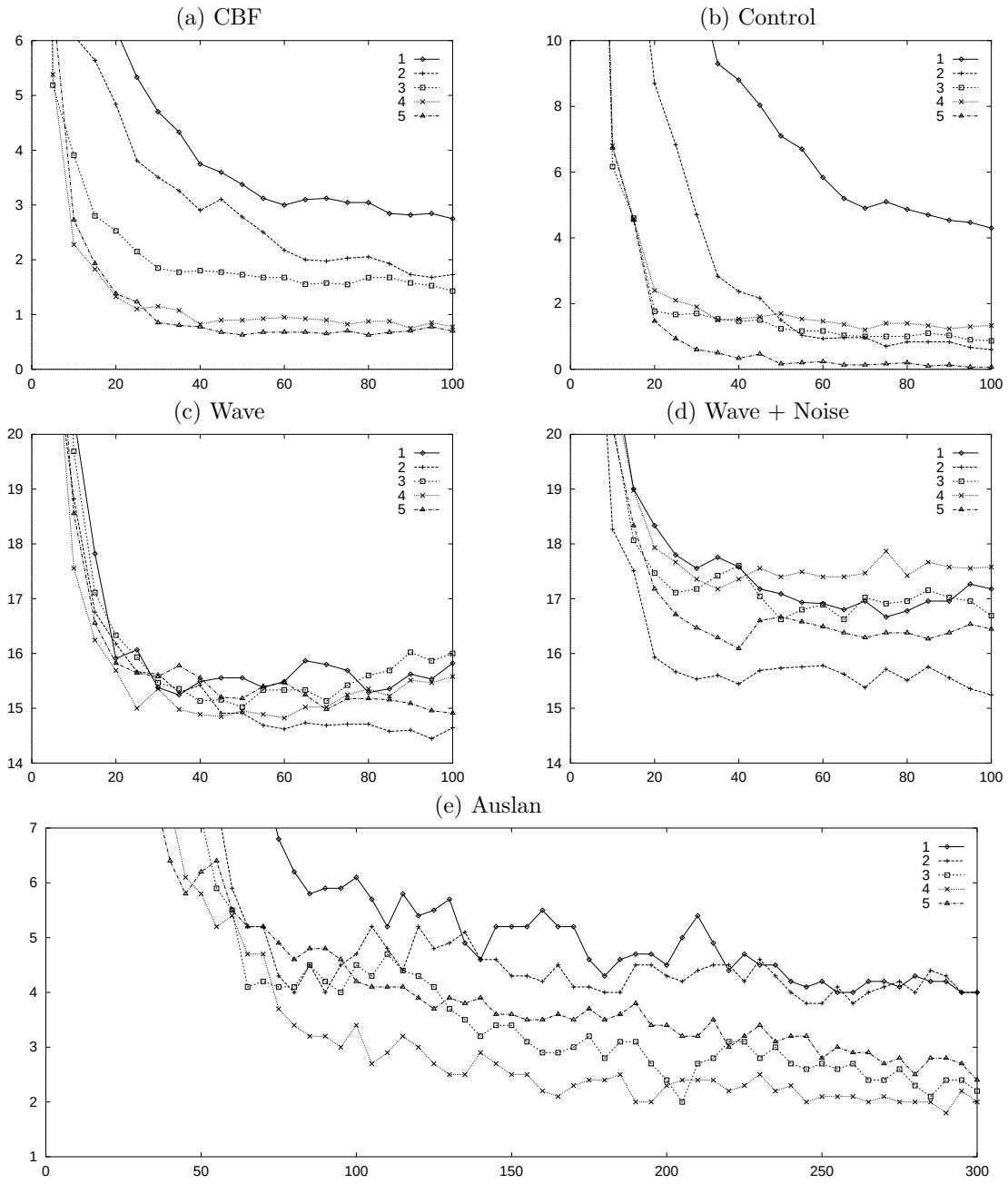

Figure 4: Graphs of the results for the different datasets.

Iter.:		10	20	30	40	50	60	70	80	90	100
CBF	1	9.26	6.20	4.70	3.74	3.37	3.00	3.12	3.04	2.82	**2.75**
	2	6.09	4.84	3.51	2.90	2.78	2.17	1.98	2.05	**1.73**	**1.73**
	3	3.91	2.53	1.85	1.80	1.73	1.68	1.58	1.67	1.58	**1.43**
	4	2.28	1.33	1.15	0.82	0.90	0.95	0.90	0.87	**0.75**	0.78
	5	2.73	1.38	0.85	0.77	**0.63**	0.68	0.65	**0.63**	0.70	0.70
Control	1	27.03	14.83	12.00	8.80	7.10	5.83	4.90	4.87	4.53	**4.30**
	2	27.40	8.70	4.70	2.37	1.50	0.93	0.97	0.83	0.83	**0.60**
	3	6.17	1.77	1.70	1.47	1.23	1.17	1.00	1.00	1.03	**0.87**
	4	6.80	2.40	1.90	1.53	1.70	1.47	**1.20**	1.40	1.23	1.33
	5	6.73	1.47	0.60	0.33	0.17	0.23	0.13	0.20	0.13	**0.07**
Wave	1	20.33	15.91	15.38	15.49	15.56	15.49	15.80	**15.29**	15.62	15.82
	2	18.82	16.18	15.62	15.42	14.91	14.62	14.69	14.71	**14.60**	14.64
	3	19.69	16.33	15.47	15.13	15.02	15.33	**15.13**	15.60	16.02	16.00
	4	17.56	15.69	15.36	14.89	14.96	**14.82**	15.02	15.36	15.51	15.58
	5	18.56	15.82	15.58	15.56	15.18	15.47	14.98	15.18	15.09	**14.91**
Wave+Noise	1	21.27	18.33	17.56	17.58	17.09	**16.91**	16.96	16.78	16.96	17.18
	2	18.27	15.93	15.53	15.44	15.73	15.78	15.38	15.51	15.56	**15.24**
	3	20.24	17.47	17.18	17.60	**16.62**	16.89	17.02	16.96	17.02	16.69
	4	20.93	17.93	**17.36**	**17.36**	17.40	17.40	17.47	17.42	17.58	17.58
	5	20.11	17.18	16.47	**16.09**	16.67	16.49	16.29	16.38	16.38	16.44

Iter.:		30	60	90	120	150	180	210	240	270	300
Auslan	1	16.00	8.41	5.96	5.40	5.20	4.30	5.40	4.20	4.20	**4.00**
	2	10.90	5.90	**4.00**	5.20	4.30	**4.00**	4.40	**4.00**	4.10	**4.00**
	3	11.60	5.50	4.20	4.30	3.40	2.80	2.70	2.70	2.40	**2.20**
	4	9.30	5.40	3.20	3.00	2.50	2.40	2.40	2.30	2.10	**2.00**
	5	8.80	5.50	4.80	3.90	3.60	3.50	3.20	3.20	2.70	**2.40**

Table 3: Experimental results. Each row shows the results for one experimental setting. Columns show the results for different number of boosting iterations. In boldface, the best results of each row.

fitting.

The advantages of using interval predicates are self-evident. The unique possible exception is the two wave datasets, where the point based predicates do not obtain the worst results. This situation is reasonable, because the *CBF* and *Control* datasets were designed specifically to test time series classifers, and hence involve situations like shifts, compressions and expansions. These situations are not present in the *Wave* datasets.

From the results it is also clear that none of the predicates is the best. There are datasets where relative predicates are more adequate and datasets where region based are more adequate. Moreover, there are some examples where the additional complexity of using true_percentage instead of always and sometime is not justified.

Cylinder, bell and funnel. The best previously published result, according to our knowledge, with this dataset is an error of 1.9 [9]. For all the experimental settings, with the exception of the predicate point_region, it is possible to obtain better results than this value. Moreover, for region based predicates, this value is reached with less than 30 iterations, and for settings 4–5 with only 20.

Control charts. The only previous result we are aware of regarding this dataset is for similarity queries [1], and not for supervised classification. To check if this dataset was trivial, we tested it with C4.5, over the raw data, and obtained an average error of 8.6 (also using five ten-fold cross validation). At the end, the result for the relative predicates (setting 2) is better than the results for region predicates (settings 3–4), but the results for region based predicates are much better using few iterations. The most remarkable point of the results for this dataset is the great improvement obtained combining different predicates (experimental setting 5) with

respect to the results obtained using them independently.

Waveform. The error of a Bayes optimal classifier on this dataset is approximately 14 [7]. This dataset is frequently used for testing classifiers. It has also been tested with boosting (and other methods of combining classifiers), over the raw data, in different works. The best previous result we are aware of for this dataset is an error of 15.21 [8]. That result was obtained using boosting, with decision trees as base classifiers, which are much more complex than our base classifiers (clauses with one literal in the body). For all the experimental settings with interval based predicates a better result than 15.21 is obtained. Nevertheless, the results for region based predicates with 100 iterations are worst; from all the experimental results this is the case where overfitting is more evident.

Wave + Noise. The best results are obtained with relative predicates: 15.25 with 100 iterations. Again, the error of an optimal Bayes classifier on this dataset is 14. This dataset was tested with bagging, boosting and variants over decision trees [3]. Although their results are given in graphs, their best error is apparently approximately 17.5.

Auslan. This is the dataset with the highest number of classes (10) and also is the only one with more than one variable (8). Hence, we incremented the number of iterations for this dataset, until 300. The best result previously published is an error of 2.50 [9], which is greater than the results obtained using region based predicates as shown in Table 3.

5 Conclusions

A temporal series classification system has been developed. It is based on boosting very simple classifiers. The individual classifiers are formed by clauses with only one literal in the body. The predicates used are based on intervals. Two kinds of interval predicates are used: relative and region based. Relative predicates consider the differences between the values in the interval. Region based predicates consider the presence of the values of a variable in a region during an interval.

Experiments on several different datasets show that the proposed method is highly competitive with previous approaches. On all data sets, the proposed method achieves better than all previously reported results we are aware of. Moreover, although the strength of the method is based on boosting, the experimental results using point based predicates shows that the incorporation of interval predicates improves significantly the obtained classifiers.

Another interesting feature of the method is its simplicity. From an user point of view, the method has only one free parameter, the number of iterations. Moreover, the classifiers obtained with a number of iterations are included in the ones obtained with more iterations. Hence, it is possible i) to select only an initial fragment of the obtained classifier and ii) to continue adding base classifiers to a previously obtained classifier. Although less important, from the programmer point of view the method is also rather simple. The implementation of boosting of stumps is one of the easiest among classification methods.

Acknowledgements

To the maintainers of the ML [6] and KDD [4] UCI Repositories. To Mohammed Waleed Kadous, David Aha and Robert J. Alcock for, respectively, donating the *Auslan*, *Wave* and *Control Charts* datasets.

References

[1] Robert J. Alcock and Yannis Manolopoulos. Time-series similarity queries employing a feature-based approach. In 7^{th} *Hellenic Conference on Informatics*, Ioannina, Greece, 1999.

[2] Carlos J. Alonso González and Juan J. Rodríguez Diez. A graphical rule language for continuous dynamic systems. In Masoud Mohammadian, editor, *Computational Intelligence for Modelling, Control and Automation.*, volume 55 of *Concurrent Systems Engineering Series*, pages 482–487, Amsterdam, Netherlands, 1999. IOS Press.

[3] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36(1/2):105–139, 1999.

[4] Stephen D. Bay. The UCI KDD archive, 1999. `http://kdd.ics.uci.edu/`.

[5] D.J. Berndt and J. Clifford. Finding patterns in time series: a dynamic programming approach. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 229–248. AAAI Press /MIT Press, 1996.

[6] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[7] L. Breiman, J.H. Friedman, A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1993. Previously published by Wadsworth & Brooks/Cole in 1984.

[8] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 1999.

[9] Mohammed Waleed Kadous. Learning comprehensible descriptions of multivariate time series. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of the* 16^{th} *International Conference of Machine Learning (ICML-99)*. Morgan Kaufmann, 1999.

[10] M. Kubat, I. Koprinska, and G. Pfurtscheller. Learning to classify biomedical signals. In R.S. Michalski, I. Bratko, and M. Kubat, editors, *Machine Learning and Data Mining*, pages 409–428. John Wiley & Sons, 1998.

[11] Juan J. Rodríguez Diez and Carlos J. Alonso González. Clasificación de series temporales mediante clausulas restringidas a literales sobre intervalos. In *VIII Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA'99*, Murcia, Spain, November 1999.

[12] Naoki Saito. *Local Feature Extraction and Its Applications Using a Library of Bases*. PhD thesis, Department of Mathematics, Yale University, 1994.

[13] Robert E. Schapire. Using output codes to boost multiclass learning problems. In *14*th *International Conference on Machine Learning (ICML-97)*, pages 313–321, 1997.

[14] Robert E. Schapire. A brief introduction to boosting. In Thomas Dean, editor, *16*th *International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1401–1406. Morgan Kaufmann, 1999.