Support Vector Machines of Interval-based Features for Time Series Classification *

Juan José Rodríguez
Lenguajes y Sistemas Informáticos
Universidad de Burgos, Spain

Carlos J. Alonso
Grupo de Sistemas Inteligentes, Departamento de Informática
Departamento de Informática

Abstract

In previous works, a time series classification system has been presented. It is based on boosting very simple classifiers, formed only by one literal. The used literals are based on temporal intervals.

The obtained classifiers were simply a linear combination of literals, so it is natural to expect some improvements in the results if those literals were combined in more complex ways. In this work we explore the possibility of using the literals selected by the boosting algorithm as new features, and then using a SVM with these metafeatures. The experimental results show the validity of the proposed method.

1 Introduction

Normally, boosting [26] is used with well-known base classifiers, such as decision trees or neural networks. Hence, its main contribution is the capacity of improving the accuracy results of those methods. Nevertheless, boosting also allows to develop domain-specific learning methods in a rather simple way. It is only necessary to develop a modest (weak) base classifier, adequate for that domain. Then, using boosting, it is possible to obtain a strong learner for that domain. For time series classification, adequate base classifiers for boosting are interval-based literals [23].

The domain we consider is time series. Our weak classifiers, interval-based literals, consider what happens in a given interval, e.g. what is the average value. These classifiers are very simple, but are designed for this domain in particular. Using boosting with these classifiers it is possible to obtain good results [23].

The obtained classifiers with boosting are linear combinations of the base classifiers. Then, it is natural to question if it could be possible to obtain better results using more complex combinations of the base classifiers. Although

*This work has been supported by the Spanish MCyT project DPI2001-01809

boosting is not as resistant to overfitting as once was considered [20], it is still very useful the capacity for obtaining a combination of weak base classifier that are able to classify reasonably well. Hence, a natural idea is to obtain the base classifiers using boosting, but combine them using a most robust method. We explore in this paper the use of Support Vector Machines, SVM [6], for the combination of the obtained base classifiers. On the other hand, in [22] we considered the use of decision trees with features selected by boosting, with the objective of obtaining more comprehensible classifiers.

The rest of the paper is organized as follows. The classification method is described in Sect. 2. Section 3 presents experimental results. Some related works are mentioned in Sect. 4. Finally, we give some concluding remarks in Sect. 5.

2 The Classification Method

2.1 Interval-based Predicates

For comparison purposes we first introduce point-based predicates. They use only one point of the series:

- point$_{\leq}$(Example, Variable, Point, Threshold). It is true if, for the Example, the value of the Variable at the Point is less or equal than Threshold.

The Variable is included in the predicate because multivariate time series are also considered. This notion of variable is the one used in time series, not the one used in machine learning. The combination of a Variable (e.g., x) and a Point (e.g., 7) would be considered in machine learning as a feature (e.g., x_7).

Interval-based predicates consider what happens in a given interval. The ones used in the present paper are:

- average$_{\leq}$(Example, Variable, Begin, End, Threshold). It is true if, for the Example, the average value of the Variable in the interval given by Begin and End is less or equal than Threshold.

- deviation$_{\leq}$(Example, Variable, Begin, End, Threshold). It is true if, for the Example, the deviation of the values of the Variable in the interval given by Begin and End is less or equal than Threshold.

In [23], more interval-based predicates are described (e.g., always, sometime), and it is explained how to select them in an efficient way.

Variable length series. There are several methods, more or less complicated, that allow to normalize the length of a set of series. These methods, which preprocess the data set, can be adequate for some domains. Nevertheless, the use of these methods is not a general solution. For instance, the series of two classes could have the same shape, but different lengths. Then, it is

Literal	Class 1	Class 2	Class 3
deviation$_{\leq}$(E, x, 63, 126, 1.813266)	-0.399084	-0.421169	1.037954
deviation$_{\leq}$(E, x, 48, 111, 1.889214)	-0.232020	-0.153754	0.606268
average$_{\leq}$(E, x, 38, 101, 0.770881)	0.096480	0.072766	0.715047
not average$_{\leq}$(E, x, 30, 33, 3.349725)	0.746371	-1.641331	0.664797
average$_{\leq}$(E, x, 55, 58, 4.280890)	-0.906306	0.352215	0.471577
deviation$_{\leq}$(E, x, 3, 34, 1.653625)	-0.334412	2.162192	-0.114016
not average$_{\leq}$(E, x, 49, 52, 5.508630)	0.743638	0.517645	-0.195655
deviation$_{\leq}$(E, x, 25, 56, 1.107998)	0.634717	0.540074	0.000214
not average$_{\leq}$(E, x, 44, 47, 3.720549)	0.484320	-0.932335	-0.155742
not deviation$_{\leq}$(E, x, 26, 33, 3.355703)	-0.149486	0.090624	0.657721

Table 1: Example of a classifier obtained with boosting, for a 3 class problem. For each literal, a weight is associated to every class.

important that the learning method can deal with variable length series. Of course, it can still be used with preprocessed data sets of uniform length.

If the series are of different lengths, there will be literals with intervals that are outside, partially or totally, for some of the series. For this cases, the result of the evaluation of the literal will not be true nor false, but an abstention.

2.2 Boosting Interval-based Literals

Clearly, in order to obtain accurate classifiers it is necessary to combine several literals. The approach used in this work is combining them by 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 [26], 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* (also named *weak*) classifier is constructed, according to the distribution of weights. Afterward, the weight of each example is readjusted, based 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.

AdaBoost is only for binary problems, but there are several methods for extending AdaBoost to the multiclass case. The one used in this work is AdaBoost.MH [27].

Table 1 shows an example classifier. It was obtained from a data set with three classes. This classifier is composed by 10 base classifiers. The first column shows the literal. For each class in the data set there is another column, with the weight associated to the literal for that class.

In order to classify a new example, a weight is calculated for each class, and then the example is assigned to the class with a greatest weight. Initially,

the weight of each class is 0. For each base classifier, the literal is evaluated. If it is true, for each class its weight is added with the weight of the class for the literal. If the literal is false, then the weight of the class for the literal is subtracted from the weight of the class.

For the basic boosting method the base classifiers return $+1$ or -1. Nevertheless, there are variants that use confidence-rated predictions [27]. In this case, the base learner returns a real value. The sign indicates the classification and the absolute value of the confidence in this prediction. There are three possible results for the evaluation of an interval based-literal: false, true, or abstention. They are assigned, respectively, the numeric values -1, 1 and 0.

2.3 SVM of Interval-based Features

Once that an ensemble of literals has been obtained, it is possible to use it as a classifier. Nevertheless, we are considering another alternative. Each literal can be considered as a new (meta)feature. Then, any other classification method can be used with these new features.

Although it would be possible to use the literals directly as boolean features, the ones considered here compare the value of a function over an interval (e.g., the average) with a threshold. If the classification method that is going to be used is able of dealing with numeric features, it is sensible to use the values of the functions instead of the values of the literals.

Boosting combines the base classifiers using a weighted voting. Our hypothesis was that perhaps the base classifiers could be combined in a better way. In these paper we test this hypothesis using Support Vector Machines.

3 Experimental Validation

3.1 Data Sets

Table 2 summarizes the characteristics of the data sets. Five are synthetic, six are real.

The *CBF* data set is an artificial problem, introduced in [25]. The learning task is to distinguish between three classes: cylinder, bell or funnel. Figure 1 shows some examples of this data set. The *CBF translated* data set is a modification of the previous one introduced in [9]. It emphasizes the shifting of patterns.

In the *Control Charts* data set there are six different classes of control charts, synthetically generated by the process in [1]. Figure 2 shows two examples of each class. The used data was obtained from the UCI KDD Archive [10].

The *Two Patterns* data set was introduced in [9]. Each class is characterized by the presence of two patterns in a definite order. Figure 3 shows examples of this data set.

The data set *Trace* is introduced in [24]. It is proposed as a benchmark for classification systems of temporal patterns in the process industry. This data set was generated artificially. There are four variables, and each variable has

	Classes	Length	Variables	Examples	Training / Test
CBF	3	128	1	798	10-CV
CBF-tr	3	128	1	5000	1000 / 4000
Control Charts	6	60	1	600	10-CV
Two Patterns	4	128	1	5000	1000 / 4000
Trace	16	[268...394]	4	1600	800 / 800
Gun	2	150	2	200	10-CV
Pendigits	10	8	2	10992	7494 / 3498
Vowels	11	10	1	990	528 / 462
Japanese Vowels	9	[7...29]	12	640	270 / 370
Auslan/N	95	[17...149]	8	1900	5-CV
Auslan/F	95	[45...136]	22	2565	5-CV

Table 2: Characteristics of the data sets.

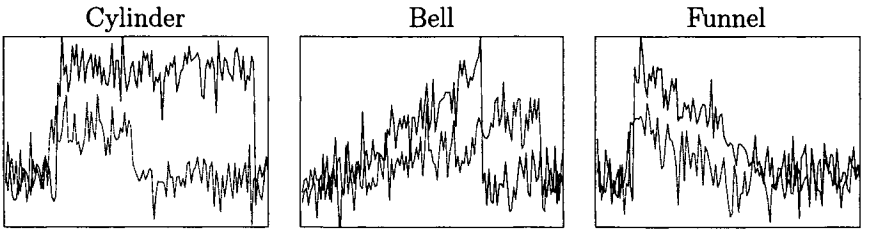

Figure 1: Examples of the *CBF* data set. Two examples of the same class are shown in each graph.

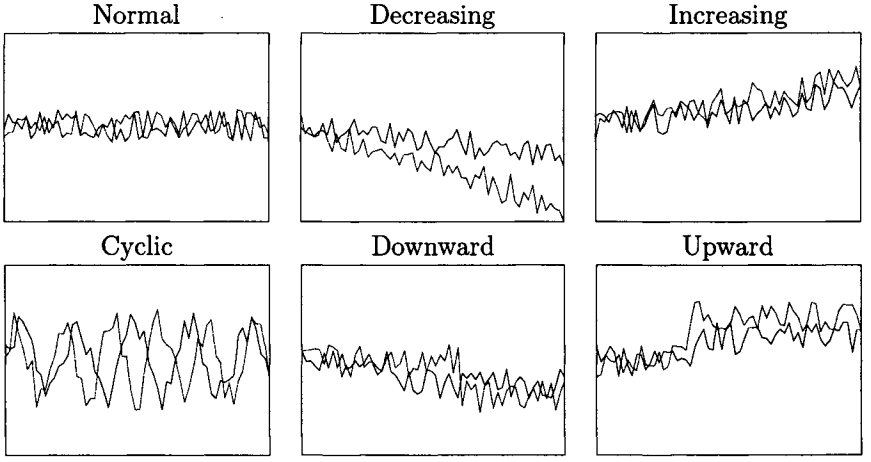

Figure 2: Examples of the *Control* data set. Two examples of the same class are shown in each graph.

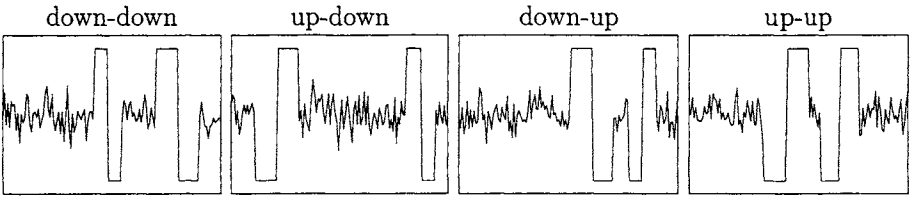

Figure 3: Examples of the *Two Patterns* data set.

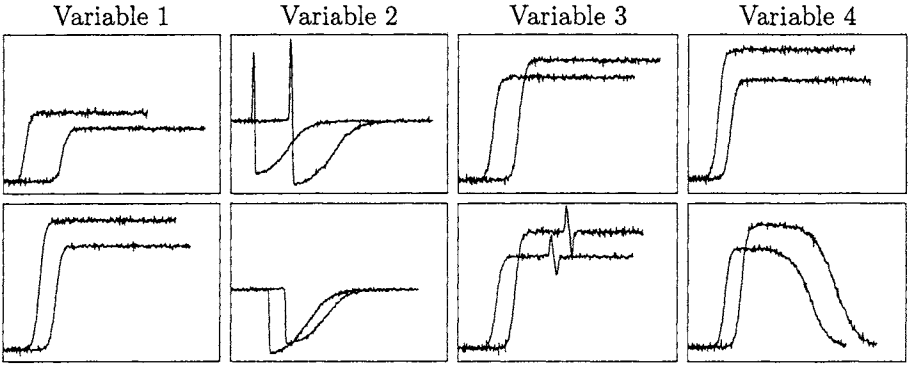

Figure 4: *Trace* data set. Each example is composed by 4 variables, and each variable has two possible behaviors. In the graphs, two examples of each behavior are shown.

two behaviors, as shown in figure 4. The combination of the behaviors of the variables produces 16 different classes. 1600 examples were generated, 100 of each class. Half of the examples are for training and the other half for testing.

The data set *Gun* was introduced in [19]. It comes from the video surveillance domain. There are two classes "gun-draw" and "point". In the first case the actor takes a replicate gun and in the second one he points with their index fingers. The data is obtained from tracking the centroid of the right hand in the X-axis. Figure 5 shows two examples of each class.

The data set *Pendigits* is introduced in [2]. It was obtained by collecting

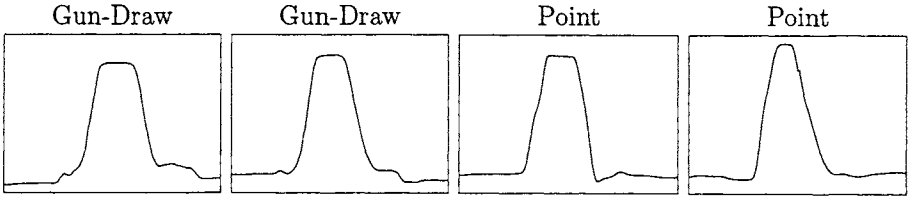

Figure 5: Examples of the *Gun* data set.

250 samples from 44 writers, using a tablet with a cordless stylus. The training data are the digits from 30 writers and the digits from the other 14 writers are the testing data.

The data set *Vowels* is introduced in [7]. It is a problem of speaker independent recognition of the eleven steady state vowels of British English.

The data set *Japanese Vowels* is introduced in [15]. It is a speaker recognition problem. Nine male speakers uttered two Japanese vowels /ae/ successively. For each utterance, it was applied 12-degree linear prediction analysis to it to obtain a discrete-time series with 12 LPC cepstrum coefficients. This means that one utterance by a speaker forms a time series whose length is in the range 7–29 and each point of a time series is of 12 features (12 coefficients).

Auslan is the Australian sign language, the language of the Australian deaf community. Instances of the signs were collected using an instrumented glove [12]. There are two versions of this data set, obtained with different equipments. In the first one the manufacturer is Nintendo, and in the second one it is Flock. According to [12], in terms of the quality of the data, the Flock system was far superior to the Nintendo system.

3.2 Results

For those data sets with an specified partition, the experiments were repeated five times, because the learning method is not deterministic, some decisions are made randomly. For the rest of data sets, 10 fold cross validation was used. The exception are the Auslan data sets because for them it is the norm to use 5 fold cross validation [12]. Boosting was used to select 100 features.

The series are of variable length, and if we want to use SVM (with conventional kernels) with these data sets it is necessary to express them using a fixed number of attributes. The used solution is to have as many attributes as the longest series (multiplied by the number of variables). The attributes corresponding to points after the end of a series are given the value "missing". This is done for implementation convenience, because the used SVM tool allows to give the value missing for an attribute. It substitutes missing attributes with the average value of the attributes. The same approach is used for interval predicates, if the interval is after the end of a series, the value of the feature will be "missing".

For linear kernels, the used implementation of SVM was the one available in WEKA [29]. It is based on the sequential minimal optimization (SMO) algorithm [18, 13]. The parameters were not adjusted in any way, the default values were used. The used features were normalized. For multiclass problems, the used tool builds a classifier for each pair of classes.

For gaussian kernels, LIBSVM was used [11, 5], because it includes a tool for selecting the parameters of this kind of kernel. It considers different values for two parameters (γ and C) and uses 5-fold cross validation for evaluating each pair of values.

The results are shown in Table 3. It shows the results obtained with boosting and with SVM, using point- and interval-based features and linear and gaussian

	Boosting		SVM (linear)			SVM (gaussian)		
Data set	points	intervals	original	points	intervals	original	points	intervals
CBF	3.51	1.13	4.88	5.37	1.25	1.00	1.00	**0.75**
CBF-tr	22.54	6.64	31.31	30.39	6.39	4.25	4.72	**0.86**
Control	4.00	0.83	1.00	1.17	**0.17**	0.50	1.67	**0.17**
Two Pat.	46.09	20.59	17.90	21.76	9.34	8.42	13.07	**2.56**
Trace	73.92	10.90	64.88	66.95	0.15	63.37	32.40	**0.12**
Gun	2.00	**0.50**	6.50	11.00	5.50	2.50	2.50	3.00
Pendigits	10.30	6.39	5.06	5.06	2.40	1.86	1.77	**1.59**
Vowels	56.10	47.10	53.03	53.38	40.17	38.10	**37.45**	39.18
J. Vowels	6.32	4.86	3.24	2.81	1.51	2.86	6.43	**1.41**
Auslan/N	40.05	34.11	42.34	34.11	22.84	43.53	39.53	**21.58**
Auslan/F	9.49	10.17	7.27	3.35	1.42	7.46	3.28	**1.28**

Table 3: Experimental results. Error rates (as percentages). The best result for each data set is marked in boldface.

kernels. It also shows the results for SVM with the original data.

For nine of the eleven data sets the best result is obtained when using interval features and gaussian kernels. The exceptions are *Gun* and *Vowels*. For the first one the best result is obtained with boosting and interval features. For this data set SVM has not any advantage for using "one vs one", because there are only two classes. For *Vowels*, the best result is for point features and gaussian kernels. In this data set the series are very short, only 10 points.

The use of boosting with the point-based feature is in fact a kind of feature selection method, the features selected are attributes that were already present in the data set. The reduction of the number of features is rather important, e.g., for the *Auslan-F* data set the number of attributes of the original data is 2992, while the selected features are only 100.

An important question when using gaussian kernels is the number of Support Vectors, because it it necessary to store them with the classifier. For the linear kernel they are not necessary because the obtained classifiers are linear combinations of the inputs. Table 4 shows the average number for the different data sets. The size of the support vectors is not the same, when using the original data set it is its dimensionality, when using selected features is the number of selected features (100 or less). For some of them (*Two Patterns*, *Trace*, *Auslan*) nearly all the training examples are considered as support vectors. If the size of the classifier is important, it could be more adequate to use linear kernels.

SVM with linear kernels and interval features has better results than boosting with interval features for nine of the eleven data sets. In some cases the differences are important (e.g., the 5 last data sets in the table). On the other hand, SVM with linear kernels and interval features has better results than SVM with linear kernels on the original data sets for the eleven data sets.

	original	points	intervals
CBF	326.0	287.1	191.7
CBF-tr	732.0	724.6	543.0
Control	351.4	336.6	222.7
Two Patterns	924.0	906.0	799.6
Trace	800.0	800.0	776.2
Gun	116.6	109.1	105.7
Pendigits	1135.0	1656.0	932.6
Vowels	522.0	522.0	516.0
Jap. Vowels	234.0	226.2	235.6
Auslan/F	2043.2	2008.8	2019.6
Auslan/N	1520.0	1519.8	1519.4

Table 4: Average number of Support Vectors.

3.2.1 Previous results.

The results that we know for these data sets from other authors are:

- *CBF*: 0.0% [12] using nearest neighbor and using Hidden Markov Models. Their data set is not the same that ours, although they were generated using the same method.
- *CBF-tr*: 3.22% [9] using decision trees of extracted patterns. Our error with gaussian SVM and interval features is 0.86%.
- *Control*: 1.50% [9] with boosting of decision trees. Our error with SVM of interval features is 0.17%.
- *Two Patterns*: 4.95% [9] using decision trees of extracted patterns. Our error with gaussian SVM and interval features is 2.56%.
- *Trace*: 1.4% [24], using recurrent neural networks and wavelets, but 4.5% of the examples are not assigned to any class. We obtain 0.15% with linear SVM and interval features.
- *Gun*: 0.5%[19] using a variant of dynamic time warping. We obtain the same result using boosting with interval features.
- *Pendigits*: 2.2% [2] using 3-nearest neighbors. 2.97% [3] using multilayer perceptrons with the series and an image of 8×8 pixels as input. Using SVM with gaussian kernels our results are smaller than 2%.
- *Vowels*: 44% [21] using the nearest neighbour classifier, the results using different neural networks are not better. Some authors use this data set with cross validation, but in that case the problem is a lot easier. In the specified partition the speakers for the training and test data sets are different.

	Boosting		SVM (linear)			SVM (gauss)		
Data set	points	intervals	original	points	intervals	original	points	intervals
Trace	69.50	13.72	44.00	59.93	**0.10**	44.75	56.63	0.13
J. Vowels	4.92	6.00	2.97	2.65	**2.00**	2.16	2.49	2.32
Auslan/N	33.89	31.74	32.79	30.53	21.47	31.37	27.42	**20.05**
Auslan/F	10.24	10.81	7.67	3.14	1.40	7.56	3.05	**1.35**

Table 5: Experimental results for the extended data sets. Error rates (as percentages).

- *Japanese Vowels*: 5.9% [15] using their method and 3.8% using a 5-state continuous Hidden Markov Model. Our results using SVM with the original data set or with interval features are better.
- *Auslan/N*: 28.80% [12], for Hidden Markov Models. Our result with linear SVM and interval features is 22.84%.
- *Auslan/F*: slight smaller than 2% [12]. This result was obtained by voting several classifiers (9 for the best result). Each one of these classifiers was obtained using feature extraction, and then boosting decision trees of that features. Our results for linear SVM and interval features are close to 1.5%.

3.2.2 Normalized data sets.

It could be argued that the used method for dealing with variable length series, to consider that the values after the end of the series have the value "missing", is not adequate for using SVM with the original data. Hence, we consider another alternative. For each data set, the series are extended to the length of the longest series in the data set. To extend a series, some of the values are duplicated. For instance, if we want to extend a series of length 100 to a length of 150, for each pair of values one would be duplicated. In this way all the values present in the original series are in the extended series and the last one has not values that are not present in the original one.

The results for the extended data sets are shown in table 5. The best results for each data set are again obtained using interval features with SVM.

4 Related Work

There are some works that also propose to select features with boosting and using these features with other classifiers. For instance, [16] considers a problem of facial expression classification and [14] a speech recognition problem.

There are several proposals for selecting features from time series. For instance, [12] proposes to extract features from each series. Then these features are grouped, and each group is considered as an attribute. [17] approximates each series with several functions, each function for an interval of the series. The

parameters of the functions and the positions of the intervals are considered as attributes. [8] proposes to use genetic programming for feature selection. The selected features are trees. The nodes in the trees are signal processing and statistical methods. The selected features are used with SVM.

With respect to SVM, one of the most related works is [4]. They use *dynamic time warping* (DTW) as a kernel. They state that although DTW it is not a metric, because triangular inequality does not hold, and that this kernel is not positive definite, they obtain good recognition rates. On the other hand, the "Dynamic Time-Alignment Kernel" [28] is a kernel based on DTW.

5 Conclusions and Further Work

A novel approach has been presented for multivariate time series classification. It is based on the application of SVM to obtained features. The obtaining of these features could be considered as a preprocessing step. Nevertheless, these features are obtained using another classification method, boosting, using as base classifiers algorithms that select the best feature. Although it is possible to use directly the boosted classifiers, the experimental results demonstrate that the accuracy can be improved substantially if SVM is applied to the obtained features. The experimental validation shows that this method is a strong alternative to state ot the art time series classification methods.

One sensible question is why boosting is used. It would be easier to use SVM with the defined meta features. This approach is not considered because there are a lot of possible metafeatures. If the length of the series is n, the number of possible intervals is $O(n^2)$. With the exception of small data sets, it is not an option to multiply the size of the data set. Hence, it is necessary some process that selects interesting metafeatures. The process considered here is boosting.

Among the objectives and conclusions of this work is not any statement about any kind of superiority of SVM against boosting. First, the complexity of the classifiers obtained with boosting and SVM are not comparable. Boosting and linear SVM produce linear combinations, but in the case of boosting there is a linear combination of features per class, but in SVM there is a linear combination per each *pair* of classes. This is because the method used for dealing with multiclass problems in the SVM case was "one vs one".

Probably, the results obtained with boosting could be improved. Using only 100, rather simple, base classifiers for a 95 class problem is not comparable with the settings commonly reported for boosting, where it is not strange to use ensembles of 100 decision trees. The fact is that obtaining and using the base classifiers has a cost, and improving the accuracy adding more classifiers to the ensemble is not always an option. The "one vs one" method could be used also for boosting, but in this case an ensemble of literals would be obtained for each pair of classes.

On the other hand, it would be possible to apply the same scheme (that is, obtaining metafeatures with boosting, using other classification method for

combining them) with any other classification method instead of SVM. In particular, it would be possible to use again boosting with, for instance, decision trees of interval literals. In this way we believe it could be possible to obtain similar results to the ones obtained with SVM.

Acknowledgements. To the maintainers of the UCI KDD Repository [10]. To the donors of the different data sets [1, 2, 7, 9, 12, 15, 19, 24]. To the developers of the WEKA library [29] and of LIBSVM [11, 5].

References

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

[2] Fevzi Alimoglu. Combining multiple classifiers for pen-based handwritten digit recognition. Master's thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University, 1996. `http://www.cmpe.boun.edu.tr/~alimoglu/alimoglu.ps.gz`.

[3] Fevzi Alimoglu and Ethem Alpaydin. Combining multiple representations for pen-based handwritten digit recognition. *ELEKTRIK: Turkish Journal of Electrical Engineering and Computer Sciences*, 9(1):1–12, 2001.

[4] Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. On-line handwriting recognition with support vector machines: A kernel approach. In *8th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 49–54, 2002.

[5] Chih-Chung Chan and Chih-Jen Lin. Training nu-support vector classifiers: Theory and algorithms. *Neural Computation*, 13(9):2119–2147, 2001. `http://www.csie.ntu.edu.tw/~cjlin/papers/newsvm.ps.gz`.

[6] Nello Cristianini and John Shawe-Taylor. *An Introductio to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[7] David H. Deterding. *Speaker Normalisation for Automatic Speech Recognition.* PhD thesis, Department of Engineering, University of Cambridge, 1989.

[8] Damian Eads, Daniel Hill, Sean Davis, Simon Perkins, Junshui Ma, Redi Porter, and James Theiler. Genetic algorithms and support vector machines for time series classification. In *5th Conference on the Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation. Symposium on Optical Science and Technology of the 2002 SPIE Annual Meeting*, 2002. `http://www.cs.rit.edu/~dre9227/papers/eadsSPIE4787.pdf`.

[9] Pierre Geurts. *Contributions to decision tree induction: bias/variance tradeoff and time series classification.* PhD thesis, Department of Electrical Engineering and Computer Science, University of Lige, Belgium, 2002.

[10] S. Hettich and S. D. Bay. The UCI KDD archive [`http://kdd.ics.uci.edu`], 1999. Irvine, CA: University of California, Department of Information and Computer Science.

[11] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

[12] Mohammed Waleed Kadous. *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series.* PhD thesis, The University of New South Wales, School of Computer Science and Engineering, 2002.

[13] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. Improvements to platt's smo algorithm for svm classifier design. Technical Report CD-99-14, Control Division, Dept. of Mechanical and Production Engineering, National University of Singapore, 1999.

[14] Aldebaro Klautau. Mining speech: automatic selection of heterogeneous features using boosting. In *ICASSP 2003*, 2003. `http://www.laps.ufpa.br/aldebaro/papers/klautau-icassp03.zip`.

[15] Mineichi Kudo, Jun Toyama, and Masaru Shimbo. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20(11–13):1103–1111, 1999.

[16] Gwen Littlewort, Marian S. Bartlett, Ian Fasel, Joel Chenu, Takayuki Kanda, Hiroshi Ishiguro, and Javier R. Movellan. Towards social robots: Automatic evaluation of human-robot interaction by facial expression classification. In *NIPS 2003 Conference Proceedings, Advances in Neural Information Processing Systems 16*, 2003.

[17] Robert T. Olszewski. *Generalized Feature Extraction for Structural Pattern Recognition in Time-Series Data.* PhD thesis, Computer Science Department, Carnegie Mellon University, 2001. `http://reports-archive.adm.cs.cmu.edu/anon/2001/abstracts/01-108.html`.

[18] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning.* MIT Press, 1998.

[19] Chotirat Ann Ratanamahatana and Eamonn Keogh. Making time-series classification more accurate using learned constraints. In *SIAM International Conference on Data Mining (SDM'04)*, 2004.

[20] Gunnar Rätsch, Takashi Onoda, and Klaus R. Müller. Regularizing adaboost, 1999. `http://ida.first.gmd.de/~raetsch/ps/RaeOnoMue98d.pdf`.

[21] Anthony J. Robinson. *Dynamic Error Propagation Networks.* PhD thesis, Cambridge University Engineering Department, 1989.

[22] Juan J. Rodríguez and Carlos J. Alonso. Interval and dynamic time warping-based decision trees. In *19th Annual ACM Symposium on Applied Computing, Special Track on Data Mining*, 2004.

[23] Juan J. Rodríguez, Carlos J. Alonso, and Henrik Boström. Boosting interval based literals. *Intelligent Data Analysis*, 5(3):245–262, 2001.

[24] Davide Roverso. Multivariate temporal classification by windowed wavelet decomposition and recurrent neural networks. In *3rd ANS International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface*, 2000.

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

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

[27] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *11th Annual Conference on Computational Learning Theory (COLT 1998)*, pages 80–91. ACM, 1998.

[28] Hiroshi Shimodaira, Ken ichi Noma, Mitsuru Nakai, and Shigeki Sagayama. Dynamic time-alignment kernel in support vector machine. In *NIPS 2001*, 2001.

[29] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.