Time Series Classification: Decision Forests and SVM on Interval and DTW Features

Juan J. Rodríguez
Escuela Politécnica Superior
University of Burgos
jjrodriguez@ubu.es

Ludmila I. Kuncheva
School of Computer Science
University of Wales, Bangor
l.i.kuncheva@bangor.ac.uk

ABSTRACT

This paper describes the methods used for our submission to the KDD 2007 Challenge on Time Series Classification. For each dataset we selected from a pool of methods (individual classifiers or classifier ensembles) using cross validation (CV). Three types of classifiers were considered: nearest neighbour (using DTW), support vector machines (linear and perceptron kernel) and decision forests (boosting, random forest, rotation forest and random oracles). SVM and decision forests used extracted features of two types: similarity-based and interval-based. Two feature selection approaches were applied: FCBF or SVM-RFE. Where the minimum CV errors of several classifiers tied, the labels were assigned through majority vote. We report results with both the practice and the contest data.

1. INTRODUCTION

The following two quotes from the challenge description [14] underpin our submission:

- "We do not expect a single algorithm will win on all datasets (but it is possible). We actually expect that the winning entry might come last on some datasets".
- "If you have two different algorithms, and you think either could win, you can just create a single meta-algorithm which tries both algorithms, and uses the predictions of the better algorithm on each problem".

Previous works have considered classifier ensemble approaches for time series classification [7, 9, 3]. In this submission we try for the first time a combination of our earlier time series classification methods [24, 25] with two recently proposed ensemble methods: Rotation Forest [27] and Random Oracles [16].

First, we tried a broad range of individual and ensemble classification methods, as detailed in section 2, using the practice data sets [15]. Feature construction and selection was carried out as explained in section 3. A subset of 36 methods was selected according to their performance on the practice data. The criteria for selecting the methods, as well as the other experimental details are provided in section 4. The most accurate methods for each contest dataset were selected by a 10-fold cross validation (CV) from the collection of 36 methods, within the competition time slot of 24 hours. The results are presented in section 5. Section 6 concludes the paper.

2. CLASSIFICATION METHODS

We considered two main approaches: (I) A nearest neighbour classifier with DTW as the dissimilarity function and (II) Construction and selection of features followed by a conventional classifier.

2.1 Dynamic Time Warping

DTW has been the technique of choice for a variety of time series data mining tasks [13, 21]. DTW had also achieved the smallest error rate on several of the practice data sets, as reported in [15], and so was placed among the most promising candidates in our experiments.

Two standard approaches to constrain the warp in DTW are the Ikatura parallelogram and the Saoke-Chiba band [21]. Both were considered in our study. Although it is possible to tune the band width for each data set, we chose a fixed band width of 5% of the series length.

In some data sets, the series of the same class are best characterised by their shapes rather than by their values. For instance, the existence of a local maximum at a given point may be more important than the value of its peak. Normalisation does not solve the problem because the values of the maxima can be very different even for normalised series. We address this question by using DTW with transformed series. A sliding window is run along the series, and the covariances between the series values in the window and the temporal axis are stored. The transformation consists in replacing the original series by these covariances. This approach is similar to Derivative DTW [12].

DTW has a quadratic time complexity, in the length of the series. For the data sets in [15] it is possible to run DTW in a sensible time. If the series were longer, however, it would not be possible to run the method in the required time. We tackle this problem by compressing the series. From each se-

ries, we generate 3 new, shorter series. The values of these series are obtained from the average, minimum and maximum of the values in an interval of the original series. The calculation of the distance between two series is as follows: first, for each series we generate the three shorter series. DTW is calculated 3 times, once for each type of series. The distance between series is the sum of these three distances.

2.2 Conventional Classifiers

A classification method is named "conventional" if it is not designed specifically for time series.

2.2.1 Support Vector Machines

Support Vector Machines [30] are among the most popular classification methods due to their accuracy and theoretical elegance. They have been applied to time series data, with kernels designed specifically for this task [28, 1, 29, 31]. Some SVM variants are based on DTW. We used the implementation in Weka [34]. It is based on the SMO algorithm [20, 10]. Two kernels were considered: the linear kernel $K(x, y) = \langle x, y \rangle$ and the perceptron kernel [18] $K(x, y) = ||x - y||_2$. The latter is less known than the Gaussian kernel but has the advantage of not requiring any parameters.

2.2.2 Classical ensemble methods

AdaBoost [6] is a well known ensemble design method which enjoys substantial theoretical and empirical support across many applications. The ensemble is built in steps where a single classifier (base classifier) is trained at each step. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ be the labelled dataset available for training the whole ensemble. A set of weights $W(i), i = 1, \ldots, N$, is maintained on the data set and used as a probability distribution for selecting the consecutive training sets. Starting from an initial uniform distribution ($W(i) = \frac{1}{N}$), a bootstrap sample S_1 is taken from $\mathcal{D}$ using the distribution W. A classifier is trained on S_1, and the weights of the mislabelled objects from $\mathcal{D}$ are increased, thereby increasing the probability that these objects will be selected in the next samples. The process of taking a sample, training a classifier and updating the weights is repeated for L steps, leading into and ensemble of L classifiers $C_1, \ldots, C_L$. The prediction of the class label for a new instance, $\mathbf{x}$, is done by weighted voting between ensemble members, where the classifiers that were found to be more accurate during the training stage participate with higher weights. AdaBoost has been found to reduce both the bias and the variance of the base classifier model which explains its almost universal success. However, other ensemble methods were found to be superior to AdaBoost when the data is noisy or contains outliers. Dozens of variants of AdaBoost have been proposed, one of them being MultiBoost [32]. In this method, after a chosen number of iterations the object weights are assigned randomly according to a Poisson distribution.

While the classifiers in AdaBoost need to be trained sequentially, other ensemble methods such as Random Subspaces and Random Forests, allow for independent training. To ensure that the classifiers in the ensemble are diverse, the Random Subspace method builds the ensemble members on randomly chosen subsets of features. Bootstrap samples may be taken instead of using always the whole $\mathcal{D}$. Random Forest is an ensemble of decision tree classifiers [2]. Bootstrap samples are taken independently and a tree is trained on each such sample. The tree induction differs slightly from the standard one: at each node, M features are taken randomly, and the best one among these features is elected to split the node. Compared to using all features to split each node, the random choice is shown to induce diversity in the ensemble without corrupting the individual accuracy. Owing to the massive number of experimental studies with classifier ensembles, the methods summarised here have acquired convincing records of stable and accurate behaviour, while no method is clearly dominated by another. This makes them all potential candidates for the classification task at hand.

2.2.3 Rotation Forests

Rotation Forest is a recently proposed classifier ensemble method [27, 17] with promising empirical results. Without loss of generality assume that the data lives in an n-dimensional feature space. The n features for building the ith tree in the ensemble are extracted from the data using principal component analysis (PCA). We first split the feature set into subsets of a specified size, M (to simplify explanation, assume that M is a factor of n). Then we take a random sample of classes, and a bootstrap sample from the instances labelled in these classes from $\mathcal{D}$. PCA is then applied on this set and all M components are retained (assuming that all M eigenvalues are non-zero). For each subset of features, M principal components are derived, using only these features. The $\frac{n}{M}$ groups, each of M extracted PCA features, are pooled to form the new feature set for classifier C_i.

We show the following example in order to clarify the feature extraction process. Let $F = \{f_1, \ldots, f_9\}$ be the feature set. Split F randomly into 3 groups of $M = 3$ features, e.g. (1,7,9), (2,3,5) and (4,6,8). Suppose that there are 6 classes in the problem. Take a random subset of classes, e.g., $\{\omega_3, \omega_6\}$. Sample from $\mathcal{D}$ using only the instances from these classes and only features (f_1, f_7, f_9). Run PCA on the sample and store the three principal components. Denote these principal components p_1, p_7, p_9, even though each component is a linear combination of the three features. Choose randomly another subset of classes and sample from $\mathcal{D}$ using only features (f_2, f_3, f_5). Run PCA and store p_2, p_3, p_5. Finally, derive in the same manner p_4, p_6, p_8. Pool all components p_j, $j = 1, \ldots, 9$. The data set for training classifier C_i is obtained by "rotating" $\mathcal{D}$. Consider the following set of principal components (arbitrary numbers)

$$
\begin{array}{llll}
p_1 = [2, -1, 4]^T & p_7 = [0, 3, 1]^T & p_9 = [-6, -9, 0]^T & (f_1, f_7, f_9) \\
p_2 = [-1, 4, 3]^T & p_3 = [1, 5, -7]^T & p_5 = [4, 0, 1]^T & (f_2, f_3, f_5) \\
p_4 = [5, 5, 1]^T & p_6 = [-2, 3, -1]^T & p_8 = [-1, 2, 2]^T & (f_4, f_6, f_8)
\end{array}
$$

Let $\mathbf{x}_1 = [6, 3, 1, -1, 2, 2, 0, -5, 1]^T$. The corresponding "rotated" instance, $\mathbf{x}_1^r$ will have values

$$
\begin{array}{rcll}
x^r_{1,1} & = & \mathbf{x}_1^T \cdot [2, 0, 0, 0, 0, 0, -1, 0, 4]^T = 12 + 0 + 4 = & 16 \\
x^r_{1,2} & = & \mathbf{x}_1^T \cdot [0, 0, 0, 0, 0, 0, 3, 0, 1]^T = 0 + 0 + 1 = & 1 \\
x^r_{1,3} & = & \mathbf{x}_1^T \cdot [-6, 0, 0, 0, 0, 0, -9, 0, 0]^T = -36 + 0 + 0 = & 36 \\
x^r_{1,4} & = & \mathbf{x}_1^T \cdot [0, -1, 4, 0, 3, 0, 0, 0, 0]^T = -3 + 4 + 6 = & 7 \\
\ldots & & & \\
x^r_{1,9} & = & \mathbf{x}_1^T \cdot [0, 0, 0, -1, 0, 2, 0, 2, 0]^T = 1 + 4 - 10 = & -5
\end{array}
$$

After all instances in the original training set $\mathcal{D}$ have been rotated, the whole rotated set is used to train a standard decision tree as the ith member of the ensemble.

2.2.4 Random Oracles

Using a random oracle leads to a meta-ensemble strategy which combines classifier fusion and classifier selection into one simple scheme. The main idea is to let each ensemble member to run its own "mini-ensemble" by selecting from a set of subclassifiers. For a given input $\mathbf{x}$, each ensemble member nominates its representative and that representative becomes a member of the overall ensemble for $\mathbf{x}$. In the simplest case, two subclassifiers, A_i and B_i are trained for each ensemble member C_i. Each subclassifier is responsible for a part of the feature space where $\mathbf{x}$ falls. We have previously proposed random linear split of the feature space, so that each of A_i and B_i is trained on half space, using the data points from $\mathcal{D}$ falling on one side of the randomly chosen hyperplane [16]. A hyperplane is chosen randomly for each for C_i and then fixed as the oracle for that ensemble member. The training sets for the subclassifiers may be of very different sizes, and with different representations of the classes in the problem. This comes to no harm to the overall model, as it increases the diversity in the ensemble. At the same time, the accuracy is maintained by the fact that the bespoke classifier from the couple (A_i, B_i) is chosen for each $\mathbf{x}$. Random oracle can come in any shape. A spherical oracle is proposed in [26], where A_i is built from the data points from $\mathcal{D}$ falling inside a sphere centred at a randomly chosen instance and with a randomly chosen radius, and B_i is built using all examples outside the sphere.

Random oracles can be applied "on top" of another ensemble method. For example, in bagging, the data for building A_i and B_i can be the bootstrap sample taken from $\mathcal{D}$ for training ensemble member C_i. In other words, the ensemble heuristic is applied first, and then the data set designated to train C_i is split through the chosen random oracle. In fact, the oracle doesn't have to have only two outcomes; we can have subclassifiers A_1, A_2, ... A_M. Previous experiments showed that applying the oracle as a meta-ensemble heuristic always improved on the accuracy of the respective ensemble methods [16].

3. FEATURE CONSTRUCTION AND SELECTION

3.1 Feature Construction

The following types of features were used

R : raw	P : amplitude
A : average	F : frequency
D : deviation	T : dtw
C : covariance	$\mathsf{T_c}$: $\mathsf{dtw_c}$
M : minimum, maximum	

3.1.1 Interval Features

Interval features [24, 25] are functions of the values, s_k of a series, s, in an interval from i to j:

- The average of the values of the interval.

$$\mathsf{average}(s,i,j) = \frac{\sum_{k=i}^{j} s_k}{l}$$

where $l = j - i + 1$ is the length of the interval.

- The standard deviation.

$$\mathsf{deviation}(s,i,j) = \sqrt{\frac{\sum_{k=i}^{j} s_k^2}{l} - (\mathsf{average}(s,i,j))^2}$$

- The covariance with the temporal axis.

$$\mathsf{covariance}(s,i,j) = \frac{\sum_{k=i}^{j} s_k k}{l} - \mathsf{average}(s,i,j)\left(\frac{i+j}{2}\right)$$

- The minimum and maximum.

$$\begin{aligned} \mathsf{minimum}(s,i,j) &= \min(s_i,\ldots,s_j) \\ \mathsf{maximum}(s,i,j) &= \max(s_i,\ldots,s_j) \end{aligned}$$

- The amplitude.

$$\mathsf{amplitude}(s,i,j) = \mathsf{maximum}(s,i,j) - \mathsf{minimum}(s,i,j)$$

- The frequency of values in a given region r.

$$\mathsf{frequency}(s,i,j,r) = \#\{s_k | i \le k \le j \wedge s_k \in r\}$$

Interval Length. The number of intervals for a series of length n is $O(n^2)$. Using all possible intervals is unacceptable except for short series. Hence, some restrictions are necessary. We only consider intervals whose length is an exact power of 2, i.e., $j - i + 1 = l = 2^k$, with k a natural number. The only exception is that we always consider the complete series, i.e., the interval $(1, n)$. With this restriction the number of intervals is reduced to $O(n \log n)$.

Suppose that it is necessary to reduce further the number of intervals by a factor of f. An interval (i, j) will only be considered if its length is greater than or equal to f, and $(i \mod f) = 1$.

Feature Evaluation. For all intervals and feature types it is necessary to evaluate the function on an interval. The naïve approach is to do this individually for each interval. Timing can be improved by an iterative calculation. The feature value for interval $(i, i+l-1)$ is calculated after the values for intervals $(i, i+l/2-1)$ and $(i+l/2, i+l-1)$ have been calculated. For the majority of feature types (e.g., average, minimum) it is possible to obtain the value of the bigger interval from the values of the smaller intervals. For the rest of the features, it is only necessary to store intermediate values for each interval that will enable the iterative calculation. In this way, the computation time necessary for constructing the interval features from a series is proportional to their total number, $O(n \log n)$.

Regions. Feature frequency needs a region to be specified. A region is an interval on the values of s. A discretisation method could be used to obtain a set of regions. In this work, the series were discretised independently of one another. The discretisation method was based on selecting intervals with the same number of values. This approach is similar to the one used in SAX [11], the difference being that

SAX selects the thresholds assuming that the data follows a Gaussian distribution.

Given a set of thresholds, $t_0, \ldots, t_M$, usually the regions are defined by two consecutive thresholds, e.g.,

$$r_k : s \in [t_{k-1}, t_k), \quad k = 1, \ldots, M.$$

Instead, we follow the method of [5] where

$$r_k : s \leq t_k.$$

3.1.2 Dissimilarity-based Features

The idea is to use dissimilarity values between pairs of examples as the new features [19, 22, 23, 36, 35]. In our case, the dissimilarity value will be DTW (dtw). Given a series s and a set of series $t_1, t_2, \ldots, t_K$, the new features describing s will be

$$\mathsf{dtw}(s, t_1), \mathsf{dtw}(s, t_2), \ldots, \mathsf{dtw}(s, t_K).$$

We will use DTW with the series obtained using a sliding window and the covariance ($\mathsf{dtw_c}$). When using DTW as features, only the Ikatura parallelogram will be considered, although it would be possible to use the Saoke-Chiba band.

3.2 Feature Selection

The number of similarity features depends on the number of training examples while number of interval features depends on the length of the series. Some classification methods (e.g., SVM and Random Forest) can work directly with high-dimensional data sets such as time series. For other methods, however, the computational cost using all interval features will be prohibitive. Hence, it is necessary to put in place a procedure for selecting subsets of the features.

Two feature selection methods are considered. The first method is FCBF (Fast Correlation-Based Filter) [37]. The number of selected features is determined automatically within the method. One problem detected with this method is that for some data sets, it finishes with an inadequately small number of features. For instance, for the CBF data set from [15] FCBF selected only one among all the interval features.

The second feature selection method is SVM-RFE (Recursive Feature Elimination using SVM) [8]. An SVM (with linear kernel) is built and a pre-selected number of features with the smallest coefficients in the classifier are discarded. This process is repeated until a desired number of features is reached. This method is slower than FCBF but does not share its drawback of drastically overcutting the feature set in some cases. Our preliminary experiments did not pick a clear winner between the two feature selection methods, so we decided to use both.

4. EXPERIMENTAL SETTINGS

4.1 Accuracy estimation protocol

Ten-fold stratified cross validation was used. Due to efficiency reasons, feature selection was not done for each fold, but only once using all the available data. This can bias optimistically the error estimations obtained with the CV. We hope that the bias will be insignificant, and the CV results will be an accurate gauge of the quality of the method for the respective dataset. Anyway, even though we take a risk with this assumption, the experimental protocol is still fair because the results will be finally evaluated on unseen testing data.

4.2 Feature extraction/selection details

The number of thresholds (and therefore regions) for feature frequency was 6. This means that for each considered (time) interval, there are 6 features (one for each region). To cope with the feature explosion, the number of intervals was reduced by requiring that the length of the interval has to be an exact power of 2. A further reduction by factor of $f = 4$ was applied. This amounts to taking intervals with length 4, 16, 32, etc., starting at times 1, 5, 9, 13. . . .

For all the interval features, the number of intervals depends on the length of the series. For time series longer than the practice data sets we found it necessary to reduce the number of intervals even further. A reduction factor $f = 16$ was used for contest data sets 01, 05, 12 and 18.

Interval features were grouped in three groups: ADC (average, deviation and covariance), MP (minimum, maximum and amplitude), and F (frequency). This means that the classifiers and the feature selection methods were used with one of these groups. The raw data is included in group ADC, because $\mathsf{average}(s, i, i) = s_i$. If a reduction factor was used, the raw data is not seen by the classifier.

For the DTW method using the covariance values, the size of the sliding window could be optimized for each data set. As a rule of thumb, this value was set to 4 for all the data sets.

For contest data sets 01, 05, 12 and 18, dynamic time warping was used with reduced series, as described in section 2.1. The length of each of the three reduced series was 10% of the original series length.

For the feature selection methods, FCBF determines the number of features automatically. For SVM-RFE, the number of selected features was fixed at 100. In each iteration, at most, 20% of the features were discarded. This means that the SVM classifier was run up to 5 times.

Feature selection was done independently for each group of features. Further to that, we formed new data sets by joining selected subsets of features. In the display of the results we use a slash (/) to indicate the joined groups of features. For instance, ADC/MP means that feature selection was done for ADC (average, deviation and covariance) and for MP (minimum, maximum and amplitude) and then the selected features from the two groups were concatenated to create a new data set.

The number of trees in the decision forest ensembles was 100.

4.3 Selection of Classification Methods

A *method* is described completely by

- a set of feature types,
- an optional feature selection method, and

□ a classification method.

Classification methods themselves have different options and parameters. If the classification method is an ensemble, it can be used with different base classifiers. The combinatorial explosion of choices requires a pre-selection of classifiers that can be trained, tested and applied to the unlabelled data within the 24 hours time allocated for the competition. To form such a collection of good classification methods we used the testing [15] results for the practice data sets. The following criteria were deemed relevant.

- Select the 5 best methods according to their average ranks [4].
- Select the 5 best methods according to the evaluation metric used in the challenge. This criterion is similar to the previous one. The difference is that, for each data set, only the top 10 methods receive points. Therefore methods that were uniformly good but were outranked by bespoke methods for the various data sets will score low on this metric.
- Each data set in the practice suite was allowed to "vote" for one or more methods with minimum error on that dataset. All methods that score any vote were added to the collection.
- For each group of interval feature types, select SVM with linear kernel. These classifiers will be available as a by-product after applying SVM-RFE feature selection [8].
- For each group of features and for each feature selection method, select the Random Forest method. The motivation is that Random Forests was found to offer a good compromise between speed and accuracy.

The collection of 36 classifiers, formed as the union of the methods chosen through these criteria, appear in tables 2 and 4. The methods are ordered differently, according to their success in the practice and in the contest data, respectively. The notation used is

{feature types}-{selection method}-{classifier}.

In some cases all the features are used; then no selection method is shown.

DTW_c is DTW using the series obtained from the covariance. DTW_b is DTW using a 5%band. By default, the parallelogram is used.

In decision forests, "Linear" and "Sphere" after the classifier name indicate that the corresponding random oracle was used. The selection process picked Boosting with pruned trees and Random Subspaces and Rotation Forest with unpruned trees.

4.4 Selection of the best method for a dataset

The collection of classifiers was applied to the contest data sets, with the aim to select one most accurate classifier for each data set. The following protocol was adopted: A 10-fold cross-validation was run on the labelled data for each of the 36 selected methods. The method with the minimum error was used to classify the unlabelled data. If several methods had the same minimum CV error, majority vote between them was used. If there was a tie in the majority vote, the classifier (or tier of classifiers) with the next smallest error was consulted. In case the tie was still not resolved, a random class label was picked from the tied classes. Ties in the accuracy were not be unlikely because some of the contest data sets had small number of labelled instances, and the classification accuracy was practically a discrete variable.

5. RESULTS

5.1 On the Practice Data Sets

This section describes the results obtained with the data sets provided in the UCR Time Series Classification/Clustering Page [15]. For all these data sets, training and testing sets were specified so that classification accuracies can be directly compared against accuracies of strawmen classifiers, kindly supplied by the contest organisers. We trained the 36 classifiers on the training data and tested them on the testing data. Table 1 shows some statistics for the error percentages on the testing data. For example, the best classifier for dataset "adiac" was T_c-SVMRFE-RotationForest, giving an error of 16.88%, while the worst classifier was F-FCBF-RandomForest with 85.93%. To test the classifier selection protocol devised for the contest data, we give the following results in the last column of the table, labelled "CV". The number is the error obtained following our protocol: pick the classifier with the smallest CV error and label the testing set with it. Note that the classifier associated with the minimum testing error, e.g., classifier T_c-SVMRFE-RotationForest for dataset "adiac", is not necessarily the classifier selected by the protocol. Althought it is unlikely that the CV selection will match the minimum achievable error (from all 36 classifiers), the expectation is that the error in column CV will be smaller than the mean or the median. While this is the case in general, we can point at a couple of anomalies in the table: data sets lighting-7 and yoga, where the error of the method selected with CV is greater than the average and median errors.

Table 2 shows the average ranks of the methods [4]. For each data set, the methods are sorted according to their performance. The best method has rank 1, the second best has rank 2, and so on. If several methods have the same performance, they receive an average rank. For instance, if 4 methods had the minimum error, they would receive a rank of 2.5 each. The ranks shown in Table 2 are averaged across all data sets.

Together with the 36 methods in the collection, this table also contains the results from our experimental protocol. A perfect selection would have an average rank of 1. Our strategy reached only 7.60. Nevertheless, there is an important gap between that and the second best method in the table.

From these average ranks we can conclude that combining several types of features is generally a good idea and that best results are obtained by Rotation Forest methods followed by SVM using the perceptron kernel. It has to be noted that this table ranks the methods with respect to all data sets put together. The rankings for an individual data set can be very different.

Table 1: Error percentages for the practice data sets.

dataset	minimum	maximum	average	median	CV
50words	19.12	39.56	26.21	25.16	19.12
adiac	16.88	85.93	31.20	23.79	17.65
beef	3.33	60.00	29.44	25.00	13.33
cbf	0.00	41.22	6.35	3.06	1.22
coffee	0.00	35.71	9.82	7.14	0.00
ecg-200	0.00	29.00	10.28	9.50	0.00
face-all	6.39	29.11	20.82	22.63	13.49
face-four	0.00	31.82	11.08	9.09	2.27
fish	2.86	24.00	11.70	10.86	4.57
gun-point	0.00	18.00	5.72	6.00	0.00
lighting-2	9.84	45.90	27.14	26.23	22.95
lighting-7	19.18	47.95	27.82	26.03	34.25
olive-oil	10.00	50.00	17.78	13.33	13.33
osu-leaf	10.74	58.26	33.47	33.06	10.74
swedish-leaf	4.48	23.52	9.54	7.60	4.48
synthetic-control	0.00	53.67	5.88	2.17	0.33
trace	0.00	44.00	4.44	0.00	0.00
two-patterns	0.00	58.53	5.38	0.30	0.02
wafer	0.02	3.91	1.00	0.53	0.13
yoga	4.23	24.97	13.77	13.83	16.03

5.2 On the Contest Data Sets

Table 3 shows some statistics for the cross validation estimates of the error rates across the 36 classification methods. There are at least two reasons why these estimates may not be completely accurate. First, as stated previously, feature selection was only done once for each data set; it was not done for each run of cross validation. Second, for several data sets, the labelled sets were very small.

Table 4 shows the average ranks of the methods. For five contest data sets (03, 06, 10, 17 and 18), the predictions were based solely on either (or both) top methods. These two methods also tied for other data sets, and in those cases majority vote was used. In summary, many of the predictions for the contest data relied heavily on the top two methods. One cause for concern is whether the good results for the top two methods are genuine or an artifact coming from our experimental protocol. The two caveats we note here are as follows. (i) Both the selection of features and the classification method are based on SVM. (ii) Feature selection is only done once for each data set. However, we do not expect this to be a major problem because:

- SVM behaves differently with linear kernels (used for feature selection) and perceptron kernels (used for classification).
- In the classification, we use a concatenation of features of different types. Thus one single run of SVM-RFE selection is not fully responsible for the entire set of features that is subsequently processed by the SVM classifier.

In retrospect, using the practice data, for each of the 36 methods, we should have analysed the differences between the CV estimates of the error and the true testing error. Methods for which the discrepancy was found to be large should have been approached more carefully, or perhaps left aside for the particularly small datasets.

Table 5 shows the methods that had minimum CV error for each one of the 20 contest data set. These methods were used for the submission. A summary is given below, where # stands for "Number of data sets"

#:	Classifier method(s)
2:	Nearest nighbour (NN)
3:	Decision forests
5:	SVM
6:	MAJORITY VOTE: Decision forests and SVM
4:	MAJORITY VOTE: Decision forests, SVM and NN

6. CONCLUSION (THE AFTERMATH)

In the contest suite, the imbalance between the sizes of labelled and unlabelled data in favour of the unlabelled data was unexpected. Practice sets did not suggest that we may be faced with very small training samples, so we did not look into alternative classification strategies suitable for this scenario. Had we known that large unlabelled data would be made available, we would have explored transductive or semi-supervised learning [33].

Table 3 also includes the final results and scores, courtesy of the Contest organisers. The scores in the last column are based on the ranking of the test error (among 12 submissions). For each data set, the best method received 10 points, the second best 9 points, and so on.

The protocol we followed requires that we take the classifier with the minimum estimated CV error for the respective data set. Therefore, the predicted error according to our strategy is the one given in column "minimum" in the CV section of Table 3. Comparing this error with the true test error shown in the penultimate column, we were quite surprised by the large discrepancy as well as the lack of

Table 3: Characteristics and error percentages for the contest data sets.

	—characteristics—				—cross validation error—				test	
dataset	classes	train	test	lenth	minimum	maximum	average	median	error	score
01	8	55	2345	1024	1.82	20.00	7.07	5.45	23.98	10
02	2	67	1029	24	1.49	10.45	4.35	4.48	12.10	3
03	2	367	1620	512	8.72	55.59	41.45	45.50	66.00	6
04	2	178	1085	512	0.00	43.82	6.74	1.40	5.98	8
05	4	40	1380	1639	0.00	55.00	17.99	10.00	3.37	9
06	5	155	308	1092	22.58	64.52	45.02	41.94	14.35	9
07	6	25	995	398	0.00	52.00	9.00	4.00	14.62	6
08	10	381	760	99	18.37	46.19	26.76	26.64	6.86	4
09	2	20	601	70	0.00	30.00	14.03	15.00	1.95	10
10	2	27	953	65	3.70	29.63	13.37	14.81	2.37	8
11	2	23	1139	82	0.00	26.09	6.76	4.35	17.31	5
12	3	1000	8236	1024	1.70	17.20	6.15	4.70	19.47	9
13	4	16	306	345	6.25	18.75	9.20	6.25	20.66	10
14	2	20	1252	84	0.00	30.00	12.50	10.00	4.12	9
15	3	467	3840	166	15.42	48.18	29.47	22.91	52.92	10
16	2	23	861	136	0.00	56.52	14.61	8.70	2.10	10
17	2	73	936	405	1.37	24.66	7.72	5.48	2.12	10
18	7	100	550	1882	56.00	79.00	65.47	65.50	52.53	0
19	14	200	2050	131	3.50	19.50	8.21	8.00	5.44	5
20	25	267	638	270	21.35	52.81	35.04	33.90	4.22	10

consistency of the prediction.

First, our concerns about the two top ranked methods in Table 4 were neither confirmed nor eliminated. For 5 data sets (03, 06, 10, 17 and 18) our submission was based only on these two methods. Our respective scores are 6, 9, 8, 10 and 0.

Second, for 4 data sets with the longest series (01, 05, 12 and 18) we had to use DTW with reduced series and fewer intervals. The scores for these data sets were 10, 9, 9 and 0, suggesting that the reduction did not have detrimental effect on the accuracy. Further analysis needs to be carried out in order to find out whether the 0-score for data set 18 is caused by the series reduction or other factors.

What was even more surprising though was that in spite of the mismatch between our prediction of the error and the true error rate, the scores were as good. The reason could be that the collection of 36 methods was formed out of a large and diverse initial pool, so every single member was a capable classifier. Thus, even if the CV error might have represented a biased and noisy estimate of the true error, the chosen of method still did a reasonable job on the intended data set. There is a lot of room for improvement, particularly in devising a more reliable error estimation strategy. It will have to be flexible enough to accommodate cases of very small labelled data sets, as well as make use of any available unlabelled data.

7. ACKNOWLEDGEMENTS

Thanks to the donors of the data sets and the organisers of the challenge. For the experiments, Weka [34] was used; thanks to its developers and contributors. We thank Dr. Matthew Williams for his assistance in running the experiments in the computer laboratory of the School of Computer Science, University of Wales Bangor.

Some of the used techniques were developed by the first author when he was doing his PhD under the supervision of Dr. Carlos Alonso. This supervision is acknowledged.

This work has been supported in part by the Spanish Ministry of Education and Science under grant PR20060503, "Junta de Castilla y León" project VA088A05 and EPSRC grant #EP/D04040X/1.

8. REFERENCES

[1] C. Bahlmann, B. Haasdonk, and H. 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.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] L. Chen and M. S. Kamel. Design of multiple classifier systems for time series data. In *Multiple Classifier Systems, 6th International Workshop, MCS 2005*, volume 3541 of *Lecture Notes in Computer Science*, pages 216–225. Springer, 2005.

[4] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[5] E. Frank and I. H. Witten. Making better use of global discretization. In *16th International Conference on Machine Learning (ICML'99)*, pages 115–123, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[6] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156, San Francisco, 1996. Morgan Kaufmann.

[7] P. Geurts and L. Wehenkel. Segment and combine approach for non-parametric time-series classification.

Table 5: Methods used in the submission for each data set.

Method	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
DTW-1NN	×						×						×							
DTW-3NN							×													
DTW-3NN																				
DTW$_b$-1NN													×							
DTW$_c$-1NN							×				×		×							×
DTW$_c$-3NN											×									
DTW$_c$-5NN		×									×									
ADC-FCBF-RandomForest											×		×							
MP-FCBF-RandomForest													×							
F-FCBF-RandomForest																				
T-FCBF-RandomForest																				
T$_c$-FCBF-RandomForest							×				×		×							
ADC/MP/T/T$_c$-FCBF-RandomForest				×							×		×							
ADC/MP/F/T/T$_c$-FCBF-RandomForest				×							×		×							
ADC-SVMRFE-RandomForest	×						×				×		×							
MP-SVMRFE-RandomForest							×						×							
F-SVMRFE-RandomForest							×						×							
T-SVMRFE-RandomForest							×						×							
T$_c$-SVMRFE-RandomForest	×						×						×							
ADC/MP/T/T$_c$-SVMRFE-RandomForest											×		×							
ADC/MP/T/T$_c$-FCBF-SubspacesSphere					×											×				
ADC/MP/F/T/T$_c$-FCBF-SubspacesSphere											×					×				
ADC-SVMRFE-AdaBoostLinear															×	×				
ADC/MP/F/T/T$_c$-FCBF-MultiBoostSphere								×											×	
ADC/MP/T/T$_c$-FCBF-RotatedRandomForest									×				×	×						
ADC/MP/F/T/T$_c$-FCBF-RotatedRandomForest	×										×		×							
ADC/MP/T/T$_c$-FCBF-RotationForest				×					×		×		×							
ADC/MP/F/T/T$_c$-FCBF-RotationForest				×									×			×				
T$_c$-SVMRFE-RotationForest											×	×								
ADC-SVMLinear													×							
F-SVMLinear													×							
MP-SVMLinear													×							
MP-SVMPerceptron													×							
ADC/MP/T/T$_c$-FCBF-SVMPerceptron				×					×		×		×							
ADC/MP/T/T$_c$-SVMRFE-SVMPerceptron	×		×	×		×	×		×	×	×	×	×			×	×	×		
ADC/MP/F/T/T$_c$-SVMRFE-SVMPerceptron				×			×		×		×		×			×	×	×		

Table 2: Average ranks from the results for the practice data sets.

Rank	Method
7.60	Selected by CV (proposed protocol)
10.93	ADC/MP/F/T/T_c-FCBF-RotationForest
11.45	ADC/MP/T/T_c-FCBF-RotationForest
11.93	ADC/MP/T/T_c-FCBF-RotatedRandomForest
12.88	ADC/MP/F/T/T_c-FCBF-RotatedRandomForest
12.83	ADC/MP/T/T_c-SVMRFE-SVMPerceptron
12.90	ADC/MP/T/T_c-FCBF-SVMPerceptron
13.08	ADC/MP/F/T/T_c-SVMRFE-SVMPerceptron
13.10	ADC/MP/F/T/T_c-FCBF-RandomForest
13.18	ADC/MP/T/T_c-FCBF-RandomForest
13.55	ADC/MP/T/T_c-SVMRFE-RandomForest
14.25	ADC/MP/F/T/T_c-FCBF-SubspacesSphere
15.25	ADC/MP/T/T_c-FCBF-SubspacesSphere
16.53	T_c-SVMRFE-RotationForest
17.90	ADC/MP/F/T/T_c-FCBF-MultiBoostSphere
18.03	MP-SVMPerceptron
18.78	DTW_c-1NN
19.25	DTW-1NN
19.15	MP-SVMLinear
19.45	ADC-SVMRFE-AdaBoostLinear
20.83	DTW-3NN
19.65	DTW_c-3NN
19.53	ADC-SVMRFE-RandomForest
19.80	ADC-SVMLinear
19.83	ADC-FCBF-RandomForest
20.03	MP-SVMRFE-RandomForest
20.88	DTW_b-1NN
22.20	MP-FCBF-RandomForest
22.73	DTW_c-5NN
24.93	DTW-3NN
23.60	T_c-SVMRFE-RandomForest
25.83	T-SVMRFE-RandomForest
28.60	F-SVMLinear
29.20	F-SVMRFE-RandomForest
30.68	T-FCBF-RandomForest
30.53	T_c-FCBF-RandomForest
32.25	F-FCBF-RandomForest

Table 4: Average ranks from the cross validation results for the contest data sets.

Rank	Method
5.18	ADC/MP/T/T_c-SVMRFE-SVMPerceptron
5.45	ADC/MP/F/T/T_c-SVMRFE-SVMPerceptron
10.13	ADC/MP/F/T/T_c-FCBF-RotationForest
10.45	ADC/MP/T/T_c-FCBF-RotationForest
10.88	ADC/MP/F/T/T_c-FCBF-RandomForest
11.20	ADC/MP/T/T_c-FCBF-RotatedRandomForest
11.55	ADC/MP/T/T_c-FCBF-SubspacesSphere
11.58	ADC/MP/F/T/T_c-FCBF-RotatedRandomForest
12.13	ADC/MP/T/T_c-FCBF-RandomForest
12.35	ADC-SVMRFE-RandomForest
12.35	ADC/MP/F/T/T_c-FCBF-SubspacesSphere
12.75	ADC/MP/T/T_c-SVMRFE-RandomForest
14.05	ADC/MP/T/T_c-FCBF-SVMPerceptron
16.45	ADC-FCBF-RandomForest
16.50	ADC/MP/F/T/T_c-FCBF-MultiBoostSphere
17.13	MP-SVMRFE-RandomForest
19.60	ADC-SVMRFE-AdaBoostLinear
20.13	DTW-1NN
20.25	MP-FCBF-RandomForest
20.73	T_c-SVMRFE-RotationForest
22.03	MP-SVMPerceptron
22.20	DTW_c-1NN
22.25	F-SVMRFE-RandomForest
22.33	ADC-SVMLinear
22.58	MP-SVMLinear
23.25	T_c-SVMRFE-RandomForest
24.18	T-SVMRFE-RandomForest
24.33	DTW_b-1NN
24.43	DTW-3NN
25.65	DTW_c-3NN
25.95	T_c-FCBF-RandomForest
26.13	F-SVMLinear
27.08	DTW_c-5NN
27.13	DTW-3NN
27.15	F-FCBF-RandomForest
28.60	T-FCBF-RandomForest

In *9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, October 2005.

[8] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[9] M. W. Kadous and C. Sammut. Classification of multivariate time series and structured data using constructive induction. *Machine Learning Journal*, 58:179–216, 2005.

[10] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13(3):637–649, 2001.

[11] E. Keogh, J. Lin, and A. Fu. HOT SAX: Efficiently finding the most unusual time series subsequence. In *5th IEEE International Conference on Data Mining (ICDM 2005)*, pages 226–233, 2005.

[12] E. Keogh and M. Pazzani. Dynamic time warping with higher order features. In *First SIAM International Conference on Data Mining, SDM'2001*, 2001.

[13] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

[14] E. Keogh, C. Shelton, and F. Moerchen. Workshop and challenge on time series classification, 2007. `http://www.cs.ucr.edu/~eamonn/KDD_Challenge.ppt`.

[15] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR time series classification/clustering homepage: `www.cs.ucr.edu/~eamonn/time_series_data`, 2006.

[16] L. I. Kuncheva and J. J. Rodríguez. Classifier ensembles with a random linear oracle. *IEEE Transactions on Knowledge and Data Engineering*, 19(4), 2007.

[17] L. I. Kuncheva and J. J. Rodríguez. An experimental study on rotation forest ensembles. In *7th International Workshop on Multiple Classifier Systems, MCS 2007*, volume 4472 of *LNCS*, pages 459–468. Springer, 2007.

[18] H.-T. Lin and L. Li. Novel distance-based SVM

kernels for infinite ensemble learning. In *12th International Conference on Neural Information Processing, ICONIP'05*, pages 761–766, 2005.

[19] E. Pękalska and R. P. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific, Singapore, 2005.

[20] J. Platt. Machines using sequential minimal optimization. In B. Schoelkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[21] C. A. Ratanamahatana and K. E. Three myths about dynamic time warping. In *SIAM International Conference on Data Mining (SDM '05)*, pages 506–510, 2005.

[22] J. J. Rodríguez and C. J. Alonso. Applying boosting to similarity literals for time series classification. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems: first international workshop, MCS 2000*, volume 1857 of *Lecture Notes in Computer Science*, pages 210–219. Springer-Verlag, 2000.

[23] J. J. Rodríguez and C. J. Alonso. Interval and dynamic time warping-based decision trees. In *19th Annual ACM Symposium on Applied Computing, Special Track on Data Mining*, volume 1, pages 548–552, 2004.

[24] J. J. Rodríguez, C. J. Alonso, and H. Boström. Boosting interval based literals. *Intelligent Data Analysis*, 5(3):245–262, 2001.

[25] J. J. Rodríguez, C. J. Alonso, and J. A. Maestro. Support vector machines of interval-based features for time series classification. *Knowledge-Based Systems*, 18(4–5):171–178, 2005.

[26] J. J. Rodríguez and L. I. Kuncheva. Naïve bayes ensembles with a random oracle. In *7th International Workshop on Multiple Classifier Systems, MCS 2007*, volume 4472 of *LNCS*, pages 450–458. Springer, 2007.

[27] J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, Oct 2006.

[28] H. Shimodaira, K. ichi Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *NIPS 2001*, 2001.

[29] K. Sivaramakrishnan and C. Bhattacharyya. Time series classification for online tamil handwritten character recognition — a kernel based approach. In *Neural Information Processing: 11th International Conference, ICONIP 2004*, volume 3316 of *Lecture Notes in Computer Science*, pages 800–805. Springer, 2004.

[30] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[31] V. Wan and J. Carmichael. Polynomial dynamic time warping kernal support vector machines for dysarthric speech recognition with sparse training data. In *9th European Conference on Speech Communication and Technology*, pages 3321–3324, 2005.

[32] G. I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2), 2000.

[33] L. Wei and E. Keogh. Semi-supervised time series classification. In *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD 2006*, 2006.

[34] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

[35] Y. Yamada, E. Suzuki, H. Yoko, and K. Takabayashi. Experimental evaluation of time-series decision tree. In *Active Mining: Second International Workshop, AM 2003. Revised Selected Papers*, pages 190–209. Springer, 2005.

[36] Y. Yamada, E. Suzuki, H. Yokoi, and K. Takabayashi. Decision-tree induction from time-series data based on a standard-example split test. In *The Twentieth International Conference on Machine Learning (ICML-2003)*, 2003. `http://www.hpl.hp.com/conferences/icml2003/papers/145.pdf`.

[37] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *20th International Conference on Machine Learning*, pages 856–863. AAAI Press, 2003.