Transformation Based Ensembles for Time Series Classification

Anthony Bagnall
University of East Anglia, Norwich, UK.
ajb@uea.ac.uk

Luke Davis
Luke.Davis@uea.ac.uk

Jon Hills
J.Hills@uea.ac.uk

Jason Lines
J.Lines@uea.ac.uk

Abstract

Until recently, the vast majority of data mining time series classification (TSC) research has focused on alternative distance measures for 1-Nearest Neighbour (1-NN) classifiers based on either the raw data, or on compressions or smoothing of the raw data. Despite the extensive evidence in favour of 1-NN classifiers with Euclidean or Dynamic Time Warping distance, there has also been a flurry of recent research publications proposing classification algorithms for TSC. Generally, these classifiers describe different ways of incorporating summary measures in the time domain into more complex classifiers. Our hypothesis is that the easiest way to gain improvement on TSC problems is to simply transform into an alternative data space where the discriminatory features are more easily detected. To test our hypothesis, we perform a range of benchmarking experiments in the time domain, before evaluating nearest neighbour classifiers on data transformed into the power spectrum, the autocorrelation function, and the principal component space. We demonstrate that on some problems there is dramatic improvement in the accuracy of classifiers built on the transformed data over classifiers built in the time domain, but that there is also a wide variance in accuracy for a particular classifier built on different data transforms. To overcome this variability, we propose a simple transformation based ensemble, then demonstrate that it improves performance and reduces the variability of classifiers built in the time domain only. Our advice to a practitioner with a real world TSC problem is to try transforms before developing a complex classifier; it is the easiest way to get a potentially large increase in accuracy, and may provide further insights into the underlying relationships that characterise the problem.

1 Introduction

Time series classification (TSC) problems, where we consider time series as any ordered data set, have been addressed by researchers in a wide range of fields including, but not limited to, data mining, statistics, machine learning, signal processing, environmental sciences, computational biology, and chemometrics. For traditional classification problems, the order of the attributes is unimportant, and the interaction between variables is considered independent of their relative positions; for time series data, the ordering of the variables is often crucial in finding the best discriminating features. Until recently, the vast majority of data mining TSC research has focused on alternative distance measures for 1-Nearest Neighbour (1-NN) classifiers based on either the raw data, or on compressions or smoothing of the raw data (see [10] for a comprehensive summary). The experimental evidence suggests that 1-NN with an elastic measure such as Dynamic Time Warping (DTW) is the best approach for smaller data sets, but that as the number of series increases *"the accuracy of elastic measures converge with that of Euclidean distance"* [10]. This idea has been propagated through much current research. For example, Batista *et al.* state that *"there is a plethora of classification algorithms that can be applied to time series; however, all of the current empirical evidence suggests that simple nearest neighbor classification is very difficult to beat"* [3]. Despite the evidence in favour of 1-NN classifiers with Euclidean or DTW distance, there has been a flurry of recent research publications proposing alternative approaches. These include shapelets [18, 27], weighted dynamic time warping [14], support vector machines built on variable intervals [22], tree based ensembles constructed on summary statistics [9], and fusion of alternative distance measures [6]. These approaches combine deriving features and/or distance measures with classifiers of varying degrees of complexity, and the majority compare their algorithms to 1-NN with Euclidean or DTW distance. For example, Jeong *et al.* state that *"we use 1-nearest neighbor classifier because the 1-nearest neighbor classifier with DTW showed very competitive performance and has been widely used for time series classification"* [14]. Our initial aim is to extend the research presented in [10] to evaluate the relative merits of alternative classifiers, rather than combinations of representations and distance measures. We start by evaluating 1-NN with Euclidean and DTW against alternative families of classification algorithms: probabilistic classifiers; decision trees and ensembles thereof; and support vector machines. We conclude that, for the new problems we describe, 1-NN with either Euclidean or DTW, whilst competitive, is not

the best approach. We then assess alternative NN classifiers. There are a variety of standard ways to improve NN classifiers, including setting k through cross validation, filtering redundant attributes, and ensembling. We conduct an experimental comparison on 26 data sets (9 of which we contribute) of 1-NN classifiers against these alternative NN approaches. We conclude that the NN variants do not significantly improve the 1-NN classifier. We maintain that the difference in classifier performance cannot easily be overcome by further model selection, as it is at least in part caused by the fact that, for certain types of TSC problems, class similarity is detectable in phase independent and/or auto correlation related feature spaces. Though complex classifiers may be able to reconstruct this similarity through the internal non-linear mapping they employ to construct the classifier, a far simpler and more intuitive approach is to transform the data into an alternative space and use a basic classifier. There are now a massive variety of algorithms for classification, and a huge body of research describing a wealth of refinements and tweaks. Whilst there have been advances, we subscribe to the view expressed in [12] that *"the large gains in predictive accuracy in classification are won using relatively simple models at the start of the process"*, and that in the context of TSC these large gains are most easily achieved through transformation.

We demonstrate that classifiers constructed on the power spectrum (PS), autocorrelation function (ACF), or principal components (PCA), can achieve significant accuracy improvement over classifiers constructed in the time domain, and that there is a large variation between domains. The key decision is which data representation to use. We propose a very simple ensemble constructed on four data sets where voting is weighted based on cross validation on the training set. Whilst basic and clearly open to improvement, we show that using this ensemble improves the overall performance of several base classifiers including NN with Euclidean distance. Our broad conclusion is that when approaching a TSC problem, looking at the data in a different representation is likely to yield greater improvement than employing more complex classifiers, or using alternative representations in the time domain.

The contributions of this paper can be summarised as follows:

1. we provide several new data sets that are available online [19], and will be made available to the UCR TSC website. These are described in Section 3;
2. we perform a thorough comparison of classifiers in the time domain to test the commonly held belief that 1-NN with Euclidean distance or DTW distance is the best approach to TSC (Section 4.2);
3. we demonstrate the potential benefit of transformation for certain data sets through a comparison with single data set classifiers and a simple ensemble of classifiers (the ensemble is described in Section 2.2, the results in Section 4.2).

2 Background

We define time series classification as the problem of building a classifier from a collection of labelled training time series. We limit our attention to problems where each time series has the same number of observations. We define a time series $\mathbf{x_i}$ as a set of ordered observations

$$\mathbf{x_i} =< x_{i1}, \ldots, x_{im} >$$

and an associated class label y_i. The training set is a set of n labelled pairs $D = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$. This definition is purposefully consistent with the general classification problem. We assume that each $(\mathbf{x_i}, y_i)$ is independent, which means we are not looking at sliding based classifiers for a single time series, where alternative methods may be appropriate. The key characteristic that differentiates time series classification problems from the general classification task is that the ordering in the attributes is important; the characteristics of $\mathbf{x_i}$ that allow us to distinguish between classes may relate to the relative temporal location of attributes. For time series, similarity within a class can take one of three general forms:

Similarity in Time. Series from each class are observations of variation of an underlying common curve in the time dimension. Variation around this underlying common shape is caused by noise in observation, and also by possible noise in indexing which may cause a slight phase shift. A classic example of this type of similarity is the Cylinder-Bell-Funnel artificial data set, where there is noise around the underlying shape, but also noise in the index of where the underlying shape transitions. Intuitively, time domain NN classifiers would be ideal for this type of problem, where DTW can be employed to mitigate against any noise in the index.

Similarity in Shape. Series within a class are distinguished by some common sub-shape or shapes that are

phase independent. The key difference to similarity in time is that the defining feature may appear at any point in the series and be masked by a large degree of between class similarity. The less correlated in time the sub-shapes are within a class, the harder this type of problem will be for time domain NN classifiers. Phase independent similarity can be detected through the correct use of transforms or through the explicit search for discriminatory sub-series or shapelets [18, 27].

Similarity in Change. The least visually obvious type of similarity, similarity in change occurs when series have a similar form of autocorrelation. Thus, it is the relationship between observations over different time steps that differentiates classes rather than the actual values observed. A standard approach here is to fit a generative model, such as Autoregressive Moving Average [2] or Hidden Markov Models.

Hence, in proposing a solution for a TSC problem, there are two design choices: what data representation to use and what classifier to use. Our basic hypothesis is that the choice of transformation will yield greater improvement than the choice of classifier, and that by explicitly working in transformed space it is easier to build a comprehensible classifier that can yield insights into the problem domain. To test this hypothesis we propose a solution based on standard transformations that can capture all three types of similarity.

2.1 Data Transformations. A large amount of transforms have been proposed for time series data mining. Wavelets, adaptive linear approximations, clipped series, and Chebeyshev polynomials are essentially all compressions of the data in the time domain used for indexing. Fourier and cosine transforms represent the data in frequency domain, but have traditionally been used to approximate the Euclidean distance. In TSC, the research has focused primarily on alternative similarity measures. However, in [10] it was shown that there is little difference between the most cited measures in the literature. The majority of recent work proposing alternative classifiers [22, 9, 6] tends to embed any transforms within the classification process, thus making it hard to qualitatively or quantitatively assess which element of the proposed technique is the source of any improvement in accuracy. We separate the process of representation and the classifier in the hope of more transparency. We propose using three standard transformations that can each capture a different type of underlying similarity in the data. These transforms have all been used many times in time series data mining, primarily for indexing. However, their application in time series classification has to date been minimal, and where transformation has been employed it has tended to be either embedded within the classifier or used as an approximation to Euclidean distance in the time domain.

2.1.1 Spectral Approaches. Each series $\mathbf{x_i}$ is decomposed into a linear combination of sinusoidal functions with amplitudes p, q and phase w,

$$\mathbf{x_t} = \sum_{k=1}^{m} \left(p_k \cos(2\pi w_k t) + q_k \sin(2\pi w_k t)\right).$$

The fast Fourier transform (FFT) is usually expressed as a series of pairs. So, two series $\mathbf{x_t}$ and $\mathbf{y_t}$ become

$$\mathbf{x_f} =< (p_0, q_0), \ldots (p_{m-1}, q_{m-1}) >$$

and

$$\mathbf{y_f} =< (r_0, s_0), \ldots, (r_{m-1}, s_{m-1}) >$$

. In the majority of time series data mining research [11], the distance between two FFT series has been defined as the squared difference between the Fourier terms,

$$d(\mathbf{x_f}, \mathbf{y_f}) = \sum_{i=0}^{m-1} (p_i - r_i)^2 + (q_i - s_i)^2.$$

When used in this way, the distance between the Fourier series is phase dependent, i.e. shifts in the time axis will be detected in the difference between Fourier terms (see [13] for a more detailed discussion). The FFT is commonly truncated and the distance is used as an approximation for Euclidean distance, generally for query/indexing problems. However, for classification, if the goal is to detect similarity in time, it makes more sense simply to use Euclidean distance in the time domain (possibly after smoothing or downsampling). The purpose of taking the

FFT is to look for similarity in the *frequency* domain, i.e. for similarity in shape. This can be done by forming the periodogram, or **power spectrum (PS)**, of the Fourier terms by squaring and adding each Fourier coefficient; i.e. transforming to series $\mathbf{x_s} =< a_0, \ldots, a_{m-1} >$ and $\mathbf{y_s} =< b_0, \ldots, b_{m-1} >$ where $a_i = p_i^2 + q_i^2$ and $b_i = r_i^2 + s_i^2$. Distance between two power spectra can be defined as the Euclidean distance $d(\mathbf{x_s}, \mathbf{y_s}) = \sum_{i=0}^{m-1}(a_i - b_i)^2$. Alternatively, a likelihood ratio measure can be used [13], although this is only suitable for truncated spectra, as it extenuates the fluctuations in smaller coefficients. Since we are using the whole spectrum we use Euclidean distance. Speech and sound processing techniques make extensive use of the power spectrum and functions derived from the spectrum, such as Spectral envelopes, mel-frequency cepstrum coefficients and linear predictive coding, although these transforms have more relevance to streaming data than fixed length labelled time series. Speech processing may or may not be useful in TSC, but in our view it makes sense to start with the spectrum before investigating more complex transforms derived from the spectrum. Transforming to the spectrum should allow the detection of similarity in the frequency domain and hence some forms of similarity in phase independent shape.

2.1.2 Autocorrelation Function. The autocorrelation function (ACF) describes how values separated by a given lag value vary together. Formally, given a series $\mathbf{x_t} =< x_1, \ldots, x_m >$, the ACF is defined as $\mathbf{x}_\rho =< \rho_1, \rho_2, \ldots, \rho_{m-1} >$, where ρ_i measures the correlation between points i apart, i.e.

$$\rho_i = \frac{\sum_{j=1}^{m-i}(x_j - \bar{x}) \cdot (x_{j+i} - \bar{x})}{m \cdot s^2}$$

where $\bar{x}$ is the series mean and s^2 the series variance. High order ACF terms are based on very few data points and are therefore unreliable, so it is common to truncate the ACF. We ignore the final 10% of the ACF transform. Note that the ACF terms are the basis for fitting ARMA models [2] through calculation of the Partial ACF. Whilst these approaches may yield better classifiers, we again think it prudent to start with the simplest transform possible. The ACF is in fact the inverse Fourier transform of the power spectrum and offers the potential to discriminate between series with different underlying autocorrelation structures, a relationship not detectable in the time domain and hard to spot in the power spectrum domain.

2.1.3 Principal Components. Principal Component Analysis (PCA) is a method of transforming a set of attributes into an alternative set of uncorrelated variables called principal components. Each transformed variable is a linear combination of the original data determined so that the majority of the variance in the data is in the first component. Unlike the PS and ACF, PCA acts on a data set rather than a single series.

2.2 Classifiers. Recently, both decision trees [27, 21] and support vector machines (SVM) [22] have been used for TSC. In all of this work the classifier is operating on a specific data representation, and yet there has, to our knowledge, been no evaluation of the classifiers on the data in the time domain. Thus, we include a comparison of a variety of decision tree based classifiers and support vector machines on TSC problems. However, our primary focus is on instance based classifiers and ensembles of classifiers on different data transforms.

2.2.1 NN Variations. k-NN classifiers are enduringly popular and surprisingly effective in a wide range of problem domains, including TSC. 1-NN classifiers have the desirable theoretical property that the error rate tends to no worse than twice the Bayes error rate as the training set size increases. However, there are several commonly referred to problems with NN classifiers. Firstly, the discriminatory power of the distance calculation can be compromised by noisy and/or redundant features. Secondly, whilst training time is minimal, classifying new instances can be computationally expensive. Thirdly, the use of a single neighbour can cause over fitting and variance, whereas setting k too high causes bias; There are a range of refinements commonly employed to overcome these problems. The refinements include: setting the neighbourhood size, k, through cross validation on the training set; condensing the training set to reduce the number of training instances; filtering the attributes to remove redundant features; and combining many diversified NN classifiers into an ensemble.

There has been recent research on condensing data sets for TSC problems [26, 6], but very little evaluation of whether the other standard improvement techniques used in classification have any impact with TSC problems. In Section 4.2 we address this issue.

2.2.2 Ensembles. An ensemble of classifiers is a set of base classifiers whose individual decisions are combined through some process of fusion to classify new examples. One key concept in ensemble design is the requirement to inject diversity into the ensemble. Broadly speaking, diversity can be achieved in an ensemble by: employing different classification algorithms to train each base classifier to form a heterogeneous ensemble; changing the training data for each base classifier through a sampling scheme or by directed replication of instances (for example, Bagging); selecting different attributes to train each classifier, often randomly; or modifying each classifier internally, either through re-weighting the training data or through inherent randomization (for example, AdaBoost).

Ensembles have been used in time series data mining [6, 22, 9]. Deng and Runger [9] propose a random forest [4] built on summary statistics (mean, slope and variance) of subseries. They compare against a random forest built on the raw data, NN+Euclidean distance and NN+DTW. Their results suggest that the transforms they propose significantly improve the random forest classifier. Buza [6] proposes an ensemble that combines alternative elastic and inelastic distance measures and compares performance to DTW and stacking (an alternative form of ensemble) support vector machines. These approaches combine classifiers and transforms. Our approach is more incremental; we wish to assess the benefits of transformation and alternative classifiers in isolation.

2.3 A Transformation Based Ensemble for Time Series Classification. The approach we propose is the simplest way we could think of for combining classifiers using the data transforms described above. We perform the transforms, build a separate classifier on each transformed data set, and then combine predictions through a weighted voting scheme. This algorithm is more formally defined by Algorithm 1. The operations PowerSpectrum, ACF and PCA create a new data set by transforming each series by the methods described in Section 2.1.

Algorithm 1 buildTransformEnsemble$(D, base)$. A Transformation Ensemble Classifier. Input: Training DataSet $D = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$, Classifier *base*.

integer: $s = 4$
integer: $nosClasses = D$.numberClassValues()
DataSet list: $\mathbf{d} =< d_1, d_2, \ldots, d_s >$
Classifier list: $\mathbf{c} =< c_1, c_2, \ldots, c_s >$
real list: $\mathbf{w} =< w_1, w_2, \ldots, w_s >$
$d_1 = D$
$d_2 =$PowerSpectrum(D)
$d_3 =$ACF(D)
$d_4 =$PCA(D)
$\mathbf{w} =$findWeights$(d, base)$
for $i = 1 \rightarrow s$ **do**
    $c_i =$base.clone()
    c_i.buildClassifier(d_i)
end for

New cases are classified as the class with the maximum weighted prediction.

We have experimented with three separate very simple weighting schemes. The first, **Equal**, gives equal weight to the predictions of each classifier. Clearly the fastest to train, it nevertheless will be deceived on problems where the classifier is inaccurate on the transformed data. The other techniques are based on the cross validation accuracy on the training set (using a ten fold cross validation (CV)). **Best** assigns a weight of 1 to the transform with the highest CV accuracy and zero to the others. If the CV accuracy is a good estimate of the true accuracy, this will be the optimal scheme. However, with small data set sizes and complex underlying models, this approach may be brittle. Hence we also use a method **Proportional**, which assigns the weight as the CV accuracy, normalised over the number of transformations.

3 Data Sets

The main contribution of this work is to highlight that, for some time series classification problems, classifiers in the time domain will not work well, and the easiest way to gain a significant improvement is to transform the data into an alternative representation and build classifiers from there. In fact, for the majority of problems in

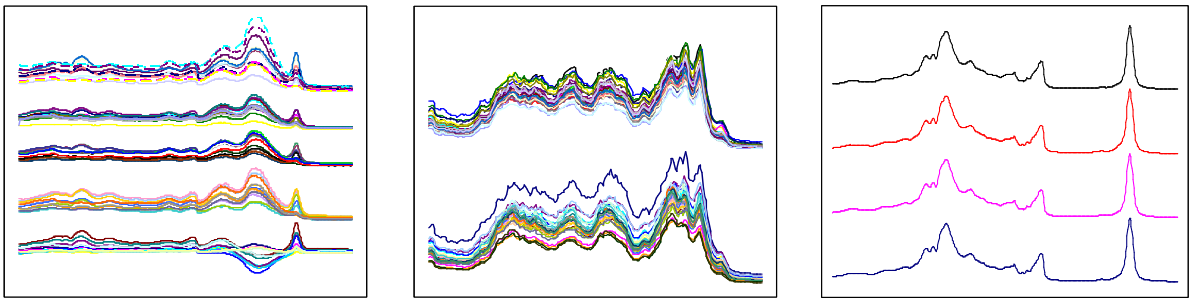
the UCR time series classification data sets [15], time domain classifiers work very well. Our work was motivated by real world problems we are attempting to solve for which time domain classifiers performed poorly. Thus, we include in our experiments the 17 UCR time series classification problems available at the time of writing (see Table 2 for a list), plus contribute 9 data sets, all of which have been, or will be, donated to the UCR repository.

Our new TSC problems, the characteristics of which are summarised in Table 1, are from a wide range of sources. We present the data with predetermined test/train splits in order to conform with the other UCR sets. The origins of these data sets are described below.

Data Set	Train Cases	Test Cases	Length	Classes
Olive Oil	30	30	570	4
Coffee	28	28	286	2
Beef	30	30	470	5
FordA	3571	1320	500	2
FordB	3601	810	500	2
Hand Outlines	1000	300	2709	2
Earthquakes	322	139	512	2
ElectricDevices	8953	7745	96	7
ARSim	2000	2000	500	2

Table 1: Time Series Classification Data Sets

3.1 Food Spectrograms: Beef, Olive Oil and Coffee. Food spectrographs are used in chemometrics to classify food types, a task that has obvious applications in food safety and quality assurance. Three spectrogram data sets for beef, coffee and olive oil are shown in Figure 1. The objective is to build classifiers so that food type can be identified from the spectrum alone. Each beef class represents a differing degree of contamination with offal [1]. The two classes of coffee are readings from Arabica and Robusta coffee variants [5]. For the olive oil data, each class is an extra virgin olive oil from alternative countries [23]. The beef spectrograms are from pure beef and beef adulterated with varying degrees of offal [1]. The data has not by default been normalised, although Figure 1 illustrates that this may be useful for beef and coffee at least.



(a) (b) (c)

Figure 1: Examples of food spectrogram data. (a) beef data, used in [1]. (b) coffee data, used in [5]. (c) olive oil data, used in [23]

3.2 WCCI 2008 Competition: Ford A and B. This data was originally used in a competition in the IEEE World Congress on Computational Intelligence, 2008 [24]. The classification problem is to diagnose whether a certain symptom exists or does not exist in an automotive subsystem. Each case consists of 500 measurements of engine noise and a classification. There are two separate problems:
FordA: Train and test data set were collected in typical operating conditions, with minimal noise contamination.
FordB: Training data were collected in typical operating conditions, but the test data samples were collected under noisy conditions.

This distinction with FordB makes it important to maintain the test and train sets separately (rather than combining and cross validating). Further details and the competition results can be found at [24]. Some example

series are shown in Figure 2. These graphs indicate that there is little temporal correlation within each class, and hence that Euclidean distance based classifiers may perform poorly.

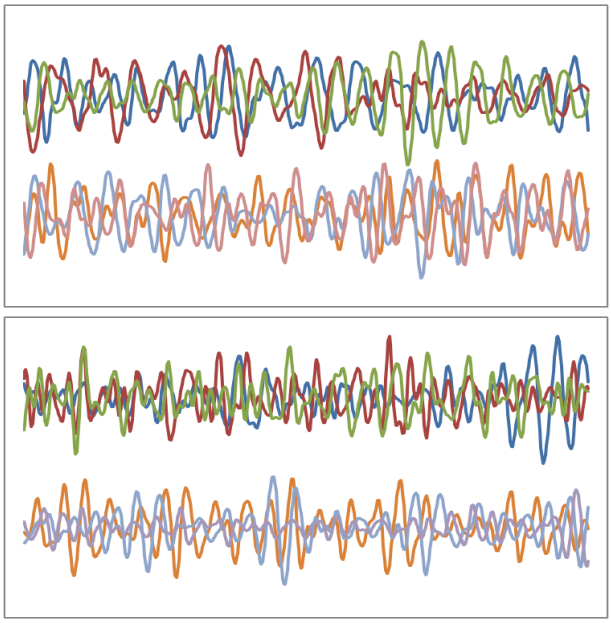


Figure 2: Three positive case and three negative case examples of the FordA (top) and FordB (bottom) motor engine data from WCCI 2008 Competition [24]

3.3 Hand Outlines. As part of a project to produce predictive models of bone age, we developed several methods for segmenting a hand image into hand/not hand. We constructed a training set of 1000 images and a test set of 370 images, taken from [20]. Each image was automatically segmented with an algorithm described in [7] and manually labelled as correct or incorrect by three human subjects. The outlines were then mapped onto a one-dimensional series based on the Euclidean distance of each pixel along the contour to the first point, and these series were smoothed using a median filter and z-normalised (across the series) to remove the scaling effect of age variation. Clearly, the length of outlines will vary. To simplify the classification, the outlines were resampled to ensure each was the same length as the shortest series (2709 attributes).

3.4 Electrical Device Usage. This data originates from a trial of electricity smart metering devices, which involved measuring the power consumption of 187 households for a variety of devices as identified by the participants (see [17]). We extracted data on the seven most commonly identified devices: kettle; immersion heater; washing machine; cold group (fridge, freezer and fridge/freezer); oven/cooker; screen group (computer and television); and dishwasher. The classification problem is to predict device type given the daily measurements of the specified device (96 attributes). Figure 4 gives an example of the demand profiles for each device type.

This problem has several confounding factors that make classification difficult. Firstly, the fact that measurements are summed over 15 minutes makes it harder to detect devices that peak over a short period. For example, a kettle will consume a large amount of power whilst on, but will only be on for two or three minutes; when summed over 15 minutes it will be harder to distinguish from a device such as a dishwasher or washing machine, which consume lower power but will be on for the whole period. Secondly, there will be a seasonal variation in the use of devices such as immersion heaters. Thirdly, we would also expect it to be hard to distinguish between similar devices such as a washing machine and a dishwasher and finally, we would expect considerable variation between different devices of the same class. The data have not been normalised.

3.5 California Earthquakes. The earthquake classification problem involves predicting whether a major event is about to occur based on the most recent readings in the surrounding area. The data is taken from Northern

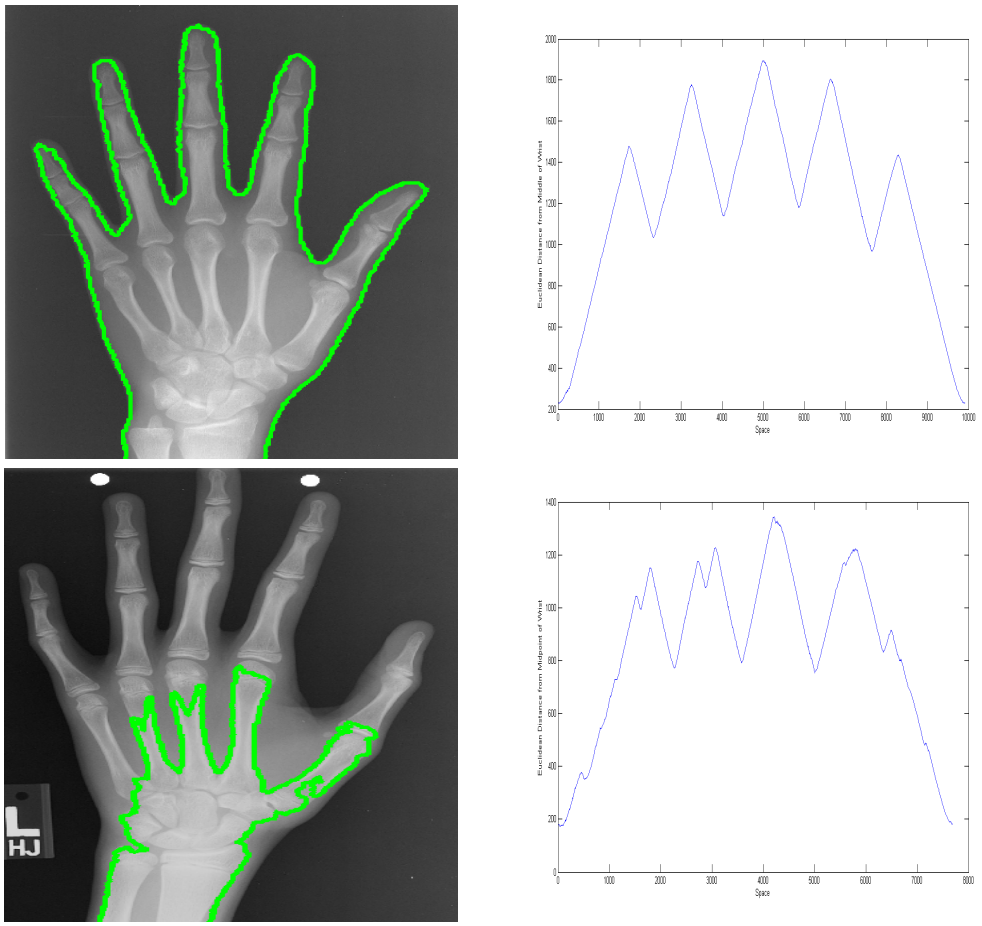


Figure 3: Examples of correct and incorrect hand outlines in two and one dimensions, as used in [7].

California Earthquake Data Center [16] and each data is an averaged reading for one hour, with the first reading taken on Dec 1st 1967, the last in 2003. We transform this single time series into a classification problem by first defining a major event as any reading of over 5 on the Rictor scale. Major events are often followed by aftershocks. The physics of these are well understood and their detection is not the objective of this exercise. Hence we consider a positive case to be one where a major event is not preceded by another major event for at least 512 hours. To construct a negative case, we consider instances where there is a reading below 4 (to avoid blurring of the boundaries between major and non major events) that is preceded by at least 20 readings in the previous 512 hours that are non-zero (to avoid trivial negative cases). None of the cases overlap in time (i.e. we perform a segmentation rather than use a sliding window). Of the 86,066 hourly readings, we produce 368 negative cases and 93 positive.

Figure 5 shows some examples of positive and negative cases. The data have not been normalised.

3.6 Simulated Autoregressive Data. This simulated data set is designed to present a time series classification problem for which it is hard to find a good classifier constructed in the time domain. The data in the two classes are generated from two different order 7 autoregressive models of the form

$$x_t = a_1 \cdot x_{t-1} + a_2 \cdot x_{t-2} + \cdots a_7 \cdot x_{t-7} + e_t$$

where e_t is an observation of a standard normally distributed random variable. Parameters for class 1 are $\{1.3532, 0.4188, -1.2153, 0.3091, 0.1877, -0.0876, 0.0075\}$ and for class 2
$\{1.0524, 0.9042, -1.2193, 0.0312, 0.263, -0.0567, -0.0019\}$.

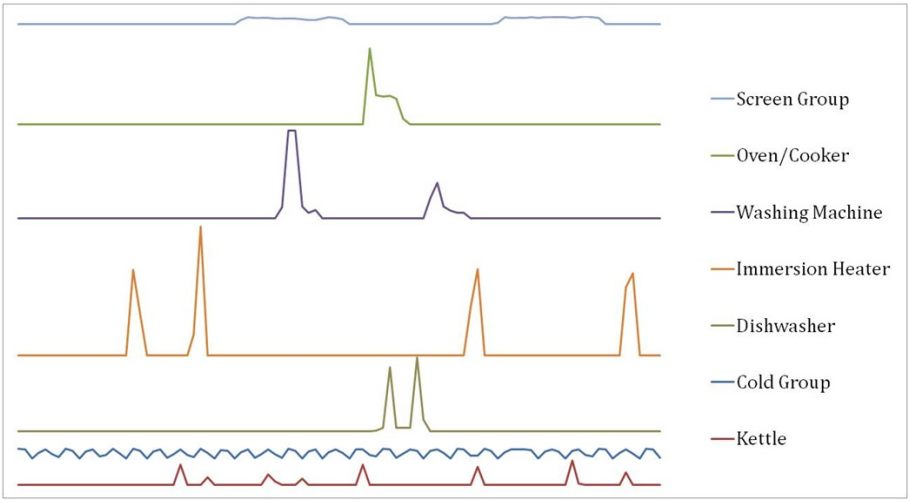


Figure 4: Examples of daily profiles for the seven types of device classification used in [17].

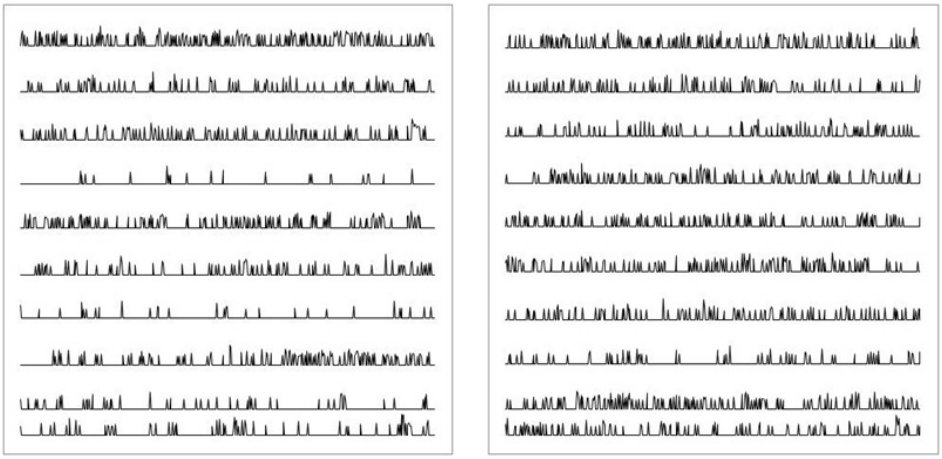


Figure 5: Examples of ten negative (left) and ten positive (right) cases in the earthquakes data set.

Both of these AR models are stationary, and were generated so that the differences between models is hard to detect, even in the ACF domain.

4 Results

4.1 Experimental Procedure Our results present the testing error using the train/test splits provided in [15]. We adopt this approach rather than a cross validation, firstly so that our results are comparable to those previously published, and secondly because with some of the data sets (such as FordB and ElectricDevices), doing so would introduce bias. Any model selection performed is conducted purely on the training data. In order to assess the relative performance of the classifiers, we adopt the procedure described in [8], which is based on a two stage rank sum test. The first test, the non-parameteric equivalent to ANOVA, tests the null hypothesis that the average rank of k classifiers on n data sets is the same against the alternative that at least one classifier's mean rank is different. So, given an n by k matrix M of error rates, the first stage is to evaluate the n by k rank matrix R, where r_{ij} is the rank of of the j^{th} classifier on the i^{th} data set. The ranks of those classifiers that have equal error are averaged. The average rank of each classifier is then denoted $\bar{r}_j = \frac{\sum_{i=1}^{n} r_{ij}}{n}$. Under the null hypothesis of the ranks being equal for all classifiers, the Friedman statistic Q:

$$Q = \frac{12n}{k(k+1)} \cdot \left[\sum_{j=1}^{k} \bar{r}_j^2 - \frac{k(k+1)^2}{4} \right]$$

can be approximated by a a Chi-squared distribution with $(k-1)$ degrees of freedom. However, Demšar identifies that previous work has shown that this approximation to the distribution of Q is conservative, and proposed using statistic

$F = \frac{(n-1)Q}{n(k-1)-Q}$ which, under the null hypothesis, follows an F distribution with $(k-1)$ and $(k-1)(n-1)$ degrees of freedom. If the Friedman test results in a rejection of the null hypothesis (i.e. we reject the hypothesis that all the mean ranks are the same), Demšar recommends a *post-hoc* pairwise Nemenyi test to discover where the differences lie. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference $CD = q_a\sqrt{\frac{k(k+1)}{6n}}$, where q_a is based on the studentised range statistic. Alternatively, if one of the classifiers can be considered a control, it is more powerful to test for difference of mean rank between classifier i and j based on a Bonferonni adjustment. Under the null hypothesis of no difference in mean rank between classifier i and j, the statistic

$$z = \frac{(\bar{r}_i - \bar{r}_j)}{\sqrt{\frac{k(k+1)}{6n}}}$$

follows a standard normal distribution. If we are performing $(k-1)$ pairwise comparisons with our control classifier, a Bonferonni adjustment simply divides the critical value of the test α by the number of comparisons performed.

Data Set	NN	NNDTW	NB	C45	SVML	SVMQ	RandF30	RotF30
Lighting2	0.1967(2)	**0.1311(1)**	0.3279(7)	0.377(8)	0.2787(6)	0.2459(4.5)	0.2459(4.5)	0.2295(3)
Lighting7	0.3699(7)	0.2877(3)	0.3562(6)	0.4521(8)	0.2877(3)	0.2877(3)	0.3151(5)	**0.2603(1)**
ECG200	**0.11(1)**	0.12(2)	0.23(7)	0.28(8)	0.19(6)	0.14(3.5)	0.17(5)	0.14(3.5)
Adiac	0.4066(5)	0.3887(3)	0.4322(6)	0.468(7)	0.5601(8)	**0.2455(1)**	0.3964(4)	0.2634(2)
FaceFour	0.125(2)	0.1818(5)	0.1591(4)	0.2841(8)	**0.1136(1)**	0.1477(3)	0.2273(7)	0.2159(6)
Fiftywords	0.356(5)	0.3297(2)	0.4374(7)	0.5824(8)	0.3538(3.5)	**0.3099(1)**	0.3824(6)	0.3538(3.5)
CBF	0.15(7)	**0.0111(1)**	0.1033(2)	0.3267(8)	0.1233(4)	0.1244(5)	0.1222(3)	0.1456(6)
fish	0.2171(4.5)	0.2171(4.5)	0.3314(7)	0.4(8)	0.1486(2)	**0.1371(1)**	0.2629(6)	0.1543(3)
GunPoint	0.08(2.5)	0.0867(4)	0.2133(7)	0.2267(8)	0.2(6)	**0.06(1)**	0.1(5)	0.08(2.5)
OSULeaf	0.4545(3)	**0.3843(1)**	0.6281(7)	0.6322(8)	0.562(6)	0.4339(2)	0.5041(5)	0.5(4)
SwedishLeaf	0.2032(7)	0.1776(6)	0.1456(3)	0.344(8)	0.1584(4.5)	**0.136(1)**	0.1584(4.5)	0.1392(2)
syntheticControl	0.12(7)	**0.0233(1)**	0.04(2)	0.19(8)	0.0767(5)	0.0567(3)	0.0733(4)	0.0833(6)
Trace	0.18(4.5)	**0.01(1)**	0.2(6)	0.26(7)	0.27(8)	0.18(4.5)	0.16(3)	0.12(2)
TwoPatterns	0.094(2)	**0.0007(1)**	0.5432(8)	0.3487(7)	0.178(5)	0.1257(4)	0.1922(6)	0.1215(3)
wafer	0.006(3)	**0.0045(1)**	0.2917(8)	0.018(6)	0.0404(7)	0.0058(2)	0.0068(4)	0.0076(5)
yoga	0.167(2)	**0.1653(1)**	0.4577(8)	0.301(6)	0.3693(7)	0.214(5)	0.192(4)	0.1887(3)
FaceAll	0.3136(6)	**0.1923(1)**	0.3083(5)	0.4497(8)	0.2817(4)	0.2669(3)	0.3243(7)	0.2385(2)
HandOutlines	0.1405(6)	0.16(7)	0.1892(8)	0.1189(4)	0.1081(3)	0.1324(5)	0.1054(2)	**0.0838(1)**
Beef	0.3333(4.5)	0.3667(6.5)	0.3333(4.5)	0.4667(8)	0.1667(2)	**0.1333(1)**	0.3667(6.5)	0.2667(3)
Coffee	0.1429(6.5)	0.0714(4.5)	0.1429(6.5)	0.1786(8)	**0(1)**	0.0357(2.5)	0.0714(4.5)	0.0357(2.5)
OliveOil	0.1667(7.5)	0.1667(7.5)	0.1(2)	0.1333(5)	0.1333(5)	0.1333(5)	0.1(2)	0.1(2)
Earthquakes	0.2878(5.5)	0.2734(4)	0.3094(8)	0.3022(7)	0.2878(5.5)	**0.223(1)**	0.259(2.5)	0.259(2.5)
FordA	0.3182(6)	0.267(4)	0.4621(8)	0.2735(5)	0.4326(7)	0.203(3)	0.1583(2)	**0.1447(1)**
FordB	0.4086(7)	0.3812(4)	0.3864(5)	0.4012(6)	0.4395(8)	**0.2975(1)**	0.3728(3)	0.3247(2)
ElectricDevices	0.4013(5)	0.35(2)	0.5929(7)	0.3892(4)	0.5613(6)	0.7385(8)	0.356(3)	**0.3007(1)**
ARSim	0.486(7)	0.4426(5)	0.371(2)	0.4715(6)	0.506(8)	0.4225(4)	0.4015(3)	**0.3615(1)**
Mean Rank	4.8269	3.1923	5.8077	7	5.0577	3	4.2885	2.8269
Diff to NN		**0.008**	0.08	**0.0007**	0.367	**0.0036**	0.214	**0.0013**

Table 2: Test error (and rank) for the following classifiers: Nearest Neighbour with Euclidean distance (NN); Nearest Neighbour with Dynamic Time Warping, full warping window (NNDTW); Naive Bayes with discretized attributes (NB); C4.5 with Weka defaults (c45); Support Vector Machine with a linear (SVML) and quadratic (SVMQ) kernel; Random Forest with 30 base classifiers (RandF); and Rotation Forest with 30 base classifiers (RotF). These results can be generated by method SDMPaper.timeDomain("SingleClassifiers")

4.2 A Comparison of Classifiers in the Time Domain 1-NN classifier using Euclidean distance or DTW on the raw data has been shown to be a highly competitive classifier in relation to other distance measures [10, 26]. Our question is, how do 1-NN with Euclidean/DTW compare to some of the vast range of classifiers that have been proposed in the machine learning literature? The *no free lunch theorem* [25] convinces us there will not be

a single dominant classifier for all TSC problems. Our investigation is intended to discover whether there is any underlying structure causing variation in performance so that we can then develop the means for automatically compensating for this structural difference. Table 2 presents the test error and rank order of 8 classifiers on 26 data sets along with the average ranks. The classifiers consist of four commonly used simple classifiers (Naive Bayes, C4.5 Decision Tree, 1-Nearest Neighbour with Euclidean Distance and a Linear Support Vector Machine), two more complex classifiers, (1-Nearest Neighbour with DTW, a Quadratic Support Vector Machine), and two tree based ensemble methods (a Random Forest and a Rotation Forest). Classifiers were trained on the raw training set (after normalisation where appropriate) with no model selection (the parameters are described in Table 2).

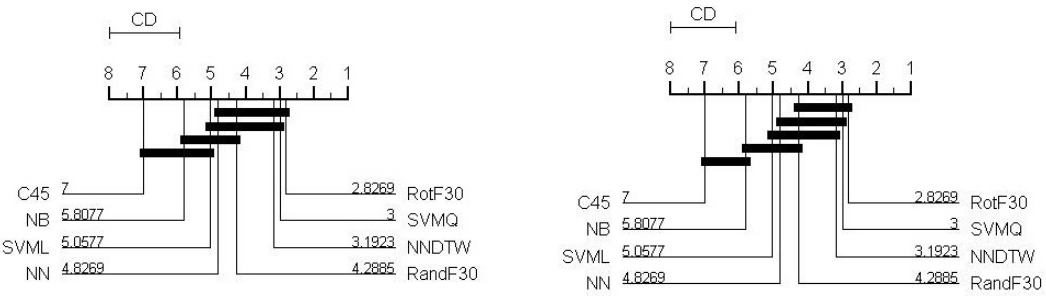


Figure 6: Critical difference plot for eight classifiers built on the raw data of 26 data sets, (left) $\alpha = 0.05$, CD=2.06 and (right) $\alpha = 0.1$, CD=1.889

We can reject the null hypothesis of no difference in mean rank (the F statistic is 15.45 which gives a negligible p value). The best performing algorithm is rotation forest (average rank 2.8), closely followed by SVM with a quadratic kernel (3.04) and NN with DTW (average rank 3.19). The last row of Table 2 shows the p-values of a two-sided pairwise test of our control classifier (1-NN) with the other classifiers. Significant differences are in bold. Note that after a Bonferroni adjustment, the critical level of a two-sided test with $\alpha = 0.05$ is 0.003571. 1-NN DTW and Rotation Forest are significantly better than 1-NN Euclid, whereas C4.5 is significantly worse.

The results of a *post-hoc* Nemenyi test are shown in the critical difference diagrams (as introduced in [8]) shown in figure 4. These graphs show the mean rank order of each algorithm on a linear scale. The left hand graph shows the critical difference for the post hoc test with a critical level of $\alpha = 0.05$ (CD=2.05) and the right hand graph with a level of $\alpha = 0.1$ (CD=1.88). The bars group algorithms into *cliques*, within which there is no significant difference in rank. The first comment to make is that even with 26 data sets there must be a large difference in rank to claim significance. The fact that rotation forest is the highest rank on these data with this particular experimental set up may be of interest, but we cannot make any claim about it being better at TSC in general. We can observe that the top clique excludes the three simple classifiers (Naive Bayes, C4.5, and linear SVM) at both 5% and 10%, and we can conclude that with the raw data these classifiers are clearly inferior. The top clique at 10% excludes 1-NN, but not at 5%. Hence 1-NN is marginal. There is also no correlation between training set size and the error rank of 1-NN (correlation 0.16, p value 0.43), or the training set ranked size (correlation 0.16, p value 0.13), indicating that it is not just the lack of data that is the cause of 1-NN performing poorly.

Overall, the results suggest that 1-NN-DTW is competitive and that 1-NN-Euclid performs better than other simple classifiers, but is significantly outperformed by more complex classifiers. So should we just employ one of the more complex classifiers for TSC? Two factors stop us drawing this simple conclusion. Firstly, even though rotation forest and SVMQ performed well relative to the other apporaches tested here, the actual error rates achieved are not as good as reported results. For example, Rotation Forest performs best on Electric Devices with an error of 0.3, but an error of 0.2 was reported in [17] through performing a simple transform on the data. Similarly, SVMQ is best on FordB with an error of 0.2975, but the winning entry achieved an error of 0.14 with a classifier constructed on the autocorrelations, and the second placed entry (error 16%) proposed a classifier constructed on the power spectrum [24]. SVMQ would have been placed 10th out of twenty entries. This indicates that there may be greater improvement achievable through transformation. Secondly, the results

aggregated over all 26 data sets do not tell the whole story. Figure 7 shows the critical difference diagrams when the results are broken down into the original UCR data sets and those provided by us.

Figure 7: Critical difference plots derived from the results in Table 2 separated into (left) 17 UCR data sets and (right) 9 data sets contributed by the authors.

The number of data sets is now smaller, so it is harder to detect significant differences. However, the ordering of the algorithms is now very different. NN-DTW is the top performer on the UCR sets, and NN-Euclid now outperforms Random Forest and is closer to Rotation Forest. The story on our data sets is reversed. NN-DTW is now the forth ranked algorithm and NN-Euclid the worst performer. The question we wish to address is, why is there this difference? Is there some inherent property of our datasets that makes this difference in performance explainable, or is it due to chance? Our prior belief was that the differences were the result of the discriminating factors being hard to detect in the time domain. However, prior to investigating time series specific ways of improving NN classifiers, it is worthwhile investigating whether techniques commonly used in traditional classification problems help with TSC.

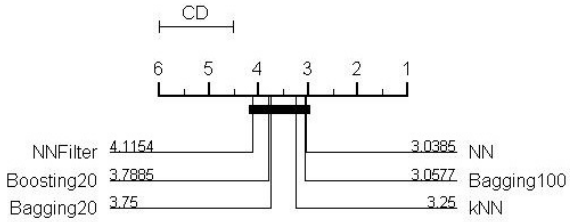


Figure 8: Critical difference plots for Nearest Neighbour (with Euclidean distance) variants. 1-NN (NN), 1-NN with 50% of the attributes, filtered with Information Gain (NNFilter), k-NN, with k set through cross validation (kNN), Bagging 1-NN with 50% of the data, 20 base classifiers (Bagging20) and 100 base classifiers (Bagging100), adaboost with 20 base classifiers (Boosting20). Data generated by method SDMPaper.timeDomain("NNEuclidClassifiers"), full results available in Table 3.

Figure 8 shows the critical difference diagram for six Nearest Neighbour variants as described in Section 2.2. There is no significant difference between them and indeed, 1-NN is ranked top. We conclude from this that improving NN classifiers for TSC problems requires more than the standard model selection techniques. To improve performance in the time domain we have several options specific to time series. We can downsample/smooth the data with one of the variety of approximations to Euclidean distance proposed in the literature, such as wavelets, adaptive splines, or discretisations. This may mitigate against the presence of noise, but it will not help detect similarity in change. The easiest way to detect this form of self similarity is through transformation.

4.3 Transformation Based Classification To demonstrate how transformation can improve a classifier, we compare 1-NN results in the time domain against the same classifier in the PS, ACF, and PCA domains, and against ensembles using the three weighting schemes described in Section 2.2. We also include 1-NN with DTW in the time domain for benchmarking. Table 4 shows the detailed results, and Figure 9 the CD diagram.

Firstly, there are clearly large variation in individual data representation performance over the data sets. For example, 1-NN with DTW has an error of almost zero on the synthetic control and two patterns data in the time

Data Set	NN	kNN	NNFilter	Bagging20	Bagging100	Boosting20
Lighting2	**0.1967(1.5)**	0.2787(6)	0.2131(3.5)	0.2623(5)	0.2131(3.5)	**0.1967(1.5)**
Lighting7	0.3699(4.5)	0.3699(4.5)	0.3562(2.5)	**0.3425(1)**	0.3562(2.5)	0.3836(6)
ECG200	0.11(3)	0.13(6)	0.11(3)	**0.09(1)**	0.11(3)	0.12(5)
Adiac	0.4066(4.5)	0.4066(4.5)	**0.3939(1)**	0.4118(6)	0.3964(2)	0.4015(3)
FaceFour	0.125(4.5)	0.125(4.5)	**0.0909(1)**	0.125(4.5)	0.125(4.5)	0.1023(2)
Fiftywords	0.356(2.5)	0.356(2.5)	0.3912(6)	0.3604(4)	**0.3516(1)**	0.378(5)
CBF	0.15(4.5)	0.15(4.5)	0.1756(6)	**0.13(1)**	0.1367(2)	0.1411(3)
fish	0.2171(3)	0.2229(4)	0.2571(6)	0.2(2)	**0.1943(1)**	0.2343(5)
GunPoint	0.08(2)	0.08(2)	0.1(6)	0.08(2)	0.0867(4.5)	0.0867(4.5)
OSULeaf	**0.4545(1)**	0.5124(6)	0.4793(4)	0.5041(5)	0.4711(3)	0.4587(2)
SwedishLeaf	**0.2032(1.5)**	**0.2032(1.5)**	0.2304(5)	0.232(6)	0.2128(3)	0.2224(4)
syntheticControl	0.12(3)	0.12(3)	0.1633(6)	0.13(5)	**0.1167(1)**	0.12(3)
Trace	0.18(2.5)	0.18(2.5)	0.34(6)	0.24(5)	0.2(4)	**0.17(1)**
TwoPatterns	**0.094(1.5)**	**0.094(1.5)**	0.3675(6)	0.115(5)	0.0978(3)	0.1098(4)
wafer	**0.006(1.5)**	**0.006(1.5)**	0.0131(6)	0.007(4)	0.0067(3)	0.0079(5)
yoga	**0.167(1.5)**	**0.167(1.5)**	0.2253(6)	0.185(5)	0.1707(3)	0.1817(4)
FaceAll	0.3136(2.5)	0.3136(2.5)	**0.3041(1)**	0.3213(5)	0.3142(4)	0.3284(6)
HandOutlines	0.1405(4)	0.1189(2)	0.1486(6)	**0.1162(1)**	0.127(3)	0.1432(5)
Beef	0.3333(2.5)	0.3333(2.5)	0.4(6)	0.3667(4.5)	0.3667(4.5)	**0.3(1)**
Coffee	0.1429(3.5)	**0.1071(1.5)**	0.1429(3.5)	0.1786(5)	**0.1071(1.5)**	0.2143(6)
OliveOil	0.1667(3.5)	0.1667(3.5)	0.1667(3.5)	0.1667(3.5)	0.1667(3.5)	0.1667(3.5)
Earthquakes	0.2878(5)	0.2518(2)	0.2518(2)	0.2518(2)	0.2662(4)	0.3453(6)
FordA	0.3182(3.5)	0.3182(3.5)	0.3167(2)	0.3333(6)	0.3189(5)	**0.2636(1)**
FordB	0.4086(5.5)	0.4086(5.5)	0.379(2)	0.3963(3)	0.3975(4)	**0.363(1)**
ElectricDevices	0.4013(3.5)	0.4013(3.5)	0.4642(6)	**0.3744(1)**	0.3897(2)	0.4114(5)
ARSim	0.486(3)	0.4795(2)	**0.4725(1)**	0.4955(5)	0.4945(4)	0.498(6)
Mean Rank	3.0385	3.25	4.1154	3.75	3.0577	3.7885
	0	-0.4077	-2.0755	-1.3713	-0.0371	-1.4454

Table 3: Comparison of different NN approaches to time series classification

Data Set	Time	TimeDTW	PS	ACF	PCA	Equal	Best	Prop
Lighting2	0.1967(3.5)	**0.1311(1)**	0.2623(8)	0.2295(5.5)	0.2459(7)	0.1475(2)	0.1967(3.5)	0.2295(5.5)
Lighting7	0.3699(4)	**0.2877(1)**	0.4247(7)	0.411(6)	0.5068(8)	0.3699(4)	0.3699(4)	0.3014(2)
ECG200	**0.11(1.5)**	0.12(3.5)	0.14(6)	0.18(7.5)	0.13(5)	0.12(3.5)	0.18(7.5)	**0.11(1.5)**
Adiac	0.4066(5.5)	0.3887(4)	**0.312(1)**	0.4757(8)	0.422(7)	0.3299(2)	0.4066(5.5)	0.3555(3)
FaceFour	**0.125(1.5)**	0.1818(5.5)	0.4659(8)	0.1818(5.5)	0.2614(7)	0.1705(4)	**0.125(1.5)**	0.1364(3)
fiftywords	0.356(3.5)	**0.3297(1)**	0.4484(7)	0.4747(8)	0.3802(6)	0.3582(5)	0.356(3.5)	0.3516(2)
CBF	0.15(2.5)	**0.0111(1)**	0.5378(8)	0.2789(5)	0.3456(7)	0.2822(6)	0.15(2.5)	0.1722(4)
fish	0.2171(2.5)	0.2171(2.5)	0.4229(8)	0.2914(7)	0.2457(5.5)	0.2171(2.5)	0.2457(5.5)	0.2171(2.5)
GunPoint	0.08(4.5)	0.0867(6)	0.0533(3)	0.1467(8)	0.1133(7)	0.08(4.5)	**0.0467(1.5)**	**0.0467(1.5)**
OSULeaf	0.4545(6)	**0.3843(1)**	0.562(8)	0.3884(2.5)	0.5041(7)	0.3884(2.5)	0.3926(4)	0.4215(5)
SwedishLeaf	0.2032(7)	0.1776(6)	0.1616(5)	**0.1344(1)**	0.2128(8)	0.1376(2)	0.1408(3)	0.152(4)
SyntheticControl	0.12(4.5)	**0.0233(1)**	0.43(8)	0.35(7)	0.22(6)	0.0867(2)	0.12(4.5)	0.09(3)
Trace	0.18(3)	**0.01(1)**	0.19(5.5)	0.18(3)	0.24(8)	0.2(7)	0.18(3)	0.19(5.5)
TwoPatterns	0.094(2.5)	**0.0007(1)**	0.5242(8)	0.505(7)	0.394(6)	0.246(5)	0.094(2.5)	0.1(4)
wafer	0.006(7.5)	0.0045(5)	**0.0036(1)**	0.0044(3.5)	0.0057(6)	0.0037(2)	0.006(7.5)	0.0044(3.5)
yoga	0.167(3)	0.1653(2)	0.2793(8)	0.1877(7)	0.178(5.5)	0.1673(4)	0.178(5.5)	**0.1633(1)**
FaceAll	0.3136(6)	**0.1923(1)**	0.3213(8)	0.3018(4)	0.3148(7)	0.2917(2)	0.303(5)	0.2941(3)
Beef	0.4(4.5)	0.3667(3)	0.5667(7)	**0.3333(1.5)**	0.6(8)	0.4667(6)	**0.3333(1.5)**	0.4(4.5)
Coffee	0.25(6)	0.0714(2)	0.4286(8)	0.0714(2)	0.3929(7)	0.2143(5)	0.0714(2)	0.1786(4)
OliveOil	0.2333(7)	0.1667(3)	0.5(8)	**0.1333(1.5)**	0.2(5)	0.2(5)	**0.1333(1.5)**	0.2(5)
Earthquakes	0.2878(6.5)	0.2734(4.5)	**0.2446(1.5)**	0.295(8)	0.2734(4.5)	**0.2446(1.5)**	0.2878(6.5)	0.2662(3)
HandOutlines	0.1405(4.5)	0.16(6)	0.2838(8)	0.1811(7)	0.1243(2.5)	**0.1108(1)**	0.1405(4.5)	0.1243(2.5)
FordA	0.3197(8)	0.267(7)	0.1795(2)	0.1977(4.5)	0.1962(3)	0.2295(6)	0.1977(4.5)	**0.1515(1)**
FordB	0.4099(8)	0.3812(7)	**0.2593(1)**	0.3543(6)	0.2827(4)	0.316(5)	0.2691(3)	0.2654(2)
ElectricDevices	0.4013(5.5)	**0.35(1)**	0.5269(8)	0.446(7)	0.4012(4)	0.3725(2)	0.4013(5.5)	0.3779(3)
ARSim	0.486(8)	0.4426(4)	0.465(5)	0.468(6)	0.4705(7)	0.347(3)	**0.252(1)**	0.3215(2)
Mean Rank	4.8654	3.1154	6	5.3462	6.0769	3.6346	3.8462	3.1154
Diff to 1-NN		**0.0064**	0.045	0.22	0.035	0.035	0.07	**0.0064**

Table 4: Results for 1-NN with Euclidean distance in the Time domain (Euclidean and DTW), power spectrum , ACF, PCA, Ensemble with equal weights, Ensemble with best CV accuracy, Ensemble with vote weighted by CV accuracy. Data generated by method sdmPaper.ensembleTransforms("1NN")

domain, whereas the error in the spectrum and ACF are not much better than random guessing. Conversely, the NN classifiers in the time domain have over 10% greater error than an alternative representation on FordA, FordB and ARSim, even with DTW. Secondly, the test of overall difference in rank yields an F statistic of 4.87, which is significant. Figure 9 indicates this difference is caused primarily by the relatively poor performance of 1-NN on the single domain data. Figure 9 also demonstrates that there is no significant difference in the single classifiers applied to the Time, PS, ACF, and PCA domains. The top clique groups 1-NN DTW with our three ensembles, and the actual average rank of 1-NN DTW is the same as our proportionally weighted ensemble. So,

in contrast to the ineffective model selection criteria in the time domain (see Figure 8), we have improved the performance of 1-NN with Euclidean distance to that of 1-NN with DTW simply by transforming the data into four separate representations. Whilst it is true that we are performing a cross validation on each training set, just weighting each transform equally still significantly improves 1-NN. It is also worth noting that weighting equally actually outperformed choosing the best transform based on cross validation accuracy. This indicates the dangers of making binary modelling decisions on statistics derived on relatively small training samples.

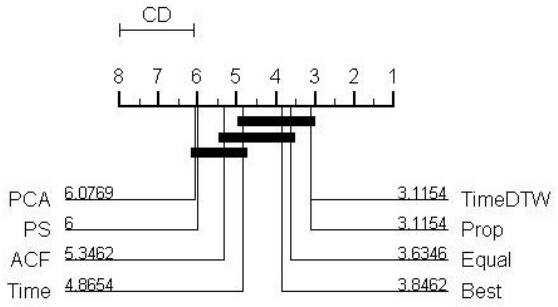
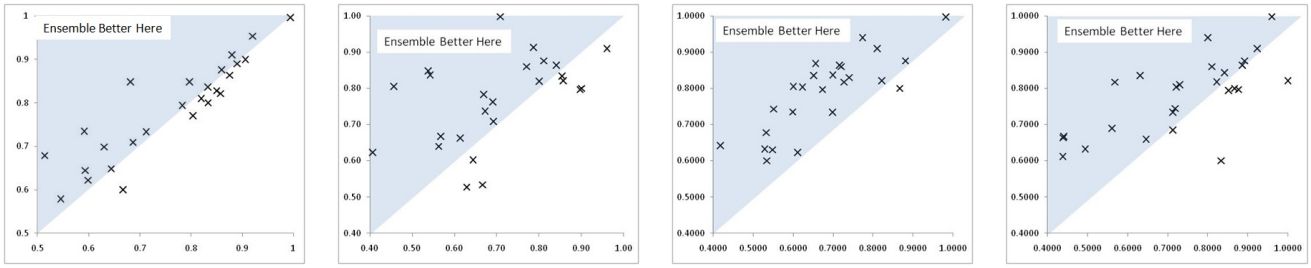


Figure 9: Critical difference plots for Nearest Neighbour classifiers in the Time domain, derived from the data in Table 4.

In fact, our simple ensemble will improve all our simple classifiers. To demonstrate this, we employ a form of graph used in [10] to compare classifiers. Figure 10 shows the accuracy of four different base classifiers against of the accuracy of an ensemble of the same base classifier constructed on the four transformed data sets.



Nearest Neighbour | naive Bayes | C4.5 | linear SVM

Figure 10: Accuracy comparison of four classifiers in the time domain (x-axis) vs proportion ensemble (y-axis).

5 Conclusions

The main objective of this paper is to highlight the importance of data transformation in time series classification. To this end, we have demonstrated that, firstly, on problems where the discriminatory features are not in the time domain, operating in a different data space produces a better performance improvement than designing a more complex classifier. Secondly, a basic ensemble on four transformed data sets can significantly improve simple classifiers. We are not suggesting that our ensemble is the best way to classify time series. We could have easily developed a more complex ensemble using standard diversification techniques, such as filtering, resampling or boosting. Equally, we could have processed our transformed data using commonly employed methods to characterise these feature spaces such as spectral envelopes or partial autocorrelations. Indeed, for our solution to the Ford challenge, we used a NN classifier with DTW on a subsection of the spectrum, selected through model selection via cross validation. Extending such approaches to the general problem of TSC form part of our future work, and may provide improved classification algorithms. However, we believe making our ensemble more complex at this point would have masked our basic message: transforming the data is the simplest way of achieving improvement in problems where the discriminating features are based on similarity in change and similarity in shape. Transformation also allows for the application of standard preprocessing methods such as

attribute filters and offers the potential for discovering other underlying characteristics of the data that may guide the exploratory analysis. So, for example, C4.5 has an error rate of 28% on the ECG200 data set in the time domain, but in the Power Spectrum domain it achieves zero error. When evaluated with an Information Gain filter we see that perfect classification can be achieved using just the first term of the power spectrum. This term represents the sum of squares of the series, which, since the data has been normalised, should be zero. Further enquiry with the data owners revealed this to be a preprocessing problem, and this classification problem is in no way representative of ECG classification generally. Other than recommending this problem be replaced in the UCR repository, our point is that this is easy to detect in the Power Spectrum but less obvious in the time domain, and highly complex time domain classifiers may just be reconstructing this feature and claiming greater representational power than is actually the case. The data sets and experimental code (embedded in the WEKA framework) are both available from the website [19] and we hope that this will serve to add to the ongoing process of making TSC research more transparent and reproducible.

References

[1] O. AlJowder, E. K. Kemsley, and R. H. Wilson, *Detection of adulteration in cooked meat products by mid-infrared spectroscopy*, J. Agric. Food Chem. **50** (2003).

[2] A. J. Bagnall and G. J. Janacek, *Clustering time series from ARMA models with clipped data*, proc. 10th ACM SIGKDD, 2004, pp. 49–58.

[3] G. Batista, X. Wang, and E. Keogh, *A complexity-invariant distance measure for time series*, Proc. 11th SDM, 2011.

[4] L. Breiman, *Random forests*, Machine Learning **45** (2001), no. 1, 5–32.

[5] R. Briandet, E. K. Kemsley, and R. H. Wilson, *Discrimination of arabica and robusta in instant coffee by fourier transform infrared spectroscopy and chemometrics*, J. Agric. Food Chem. **44** (1996), no. 1.

[6] K. Buza, *Fusion methods for time-series classification*, Ph.D. thesis, University of Hildesheim, Germany, 2011.

[7] L. M. Davis, B. J. Theobald, A. Toms, and A. J. Bagnall, *On the extraction and classification of hand outlines*, Proc. of the 12th IDEAL, 2011.

[8] J. Demšar, *Statistical comparisons of classifiers over multiple data sets*, JMLR **7** (2006), 1–30.

[9] H. Deng, G Runger, E. Tuv, and M. Vladimir, *A time series forest for classification and feature extraction*, Tech. report, Arizona State University, 2011.

[10] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, *Querying and mining of time series data: Experimental comparison of representations and distance measures*, Proc. 34th VLDB, 2008.

[11] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, *Fast subsequence matching in time-series databases*, Proc. ACM SIGMOD, 1994.

[12] D. Hand, *Classifier technology and the illusion of progress*, Statistical Science **21** (2006), no. 1, 1–15.

[13] G. J. Janacek, A. J. Bagnall, and M. Powell, *A likelihood ratio distance measure for the similarity between the fourier transform of time series*, Proc. 9th PAKDD, 2005.

[14] Y. Jeong, M. Jeong, and O. Omitaomu, *Weighted dynamic time warping for time series classification*, Pattern Recognition **44** (2010), 2231–2240.

[15] E. Keogh and T. Folias, *The UCR time series data mining archive*, `http://www.cs.ucr.edu/ eamonn/TSDMA/`.

[16] Berkeley Seismological Laboratory, *The northern california earthquake data center*, `http://www.ncedc.org/`.

[17] J. A. Lines, A. J. Bagnall, P. Caiger-Smith, and S. Anderson, *Classification of household devices by electricity usage profiles*, Proc. of the 12th IDEAL, 2011.

[18] A. Mueen, E. Keogh, and N. Young, *Logical-shapelets: An expressive primitive for time series classification*, Proc. 17th ACM SIGKDD, 2011.

[19] paper authors, *Accompanying information for this paper*, `https://sites.google.com/site/tscensembles/`.

[20] The Image Processing and University of Southern California Informatics Lab, *The digital hand atlas database system*, `http://www.ipilab.org/BAAweb/`.

[21] J. Rodriguez and C. Alonso, *Interval and dynamic time warping-based decision trees*, Proc. 19th ACM SAC, 2004.

[22] ———, *Support vector machines of interval-based features for time series classification*, Knowledge-Based Systems **18** (2005).

[23] H. S. Tapp, M. Defernez, and E. K. Kemsley, *FTIR spectroscopy and multivariate analysis can distinguish the geographic origin of extra virgin olive oils*, J. Agric. Food Chem. **51** (2004), no. 21.

[24] WCCI, *Ford classification challenge*, `http://home.comcast.net/ nn_classification/`.

[25] D. H. Wolpert and W. G. Macready, *No free lunch theorems for optimization*, IEEE Trans. on Evo. Comp. **1** (1997), no. 1, 67–82.

[26] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. Ratanamahatana, *Fast time series classification using numerosity reduction*, Proc. 23rd ICML, 2006.

[27] L. Ye and E. Keogh, *Time series shapelets: A new primitive for data mining*, Proc. 15th ACM SIGKDD, 2009.