An Empirical Evaluation of Kernels for Time Series Classification

Mourtadha Badiane · Pádraig Cunningham

December 18, 2021

Abstract Time-series classification is a particular challenge for Machine Learning (ML) because the most powerful ML classification methods require data in a feature vector format. Whereas effective time-series classification depends on flexible similarity measures that can match sub-sections of signals. In this paper we address the challenge of incorporating five such similarity measures in Support Vector Machines (SVMs). This is a challenge because SVMs require kernel functions that are positive semi-definite (PSD). In order for a kernel to be PSD the underlying distance/similarity measure must be a metric and most time-series similarity measures do not meet this requirement. Of the five similarity measures we consider Time Warp Edit Distance (TWED) is the only one that does meet this requirement and it does deliver good SVM performance. Somewhat surprisingly, Dynamic Time Warping (DTW) which is not a proper metric slightly outperforms TWED when incorporated in a Gaussian SVM – an example where something works in practice but not in theory. In all our evaluations we use k Nearest Neighbour (kNN) as a baseline and it performs very well with all distance measures. Whereas some of the distance measures deliver poor SVM performance. We also discuss the reduction in computational cost achieved by using an SVM, finding that the negative kernel paired with the DTW distance produces the greatest reduction in computational cost. We find that a significant saving can be made when using an SVM on large datasets that is not attained by regular kNN.

Keywords Support Vector Machines · Dynamic Time Warping · Time Series Classification · Time Warp Edit Distance

Mourtahda Badiane
University College Dublin
E-mail: mourtadha.badiane@ucdconnect.com

Pádraig Cunnighamm
University College Dublin
E-mail: padraig.cunningham@ucd.com

1 Introduction

In this paper we aim to determine the best kernel for time series classification through experiments on a large collection of publicly available datasets[1]. Time series classfication can be challenging for kNN and SVMs since data is not in a feature or spatial vector format, however kNN and SVMs can adapt to the task if their undelying distance measures are also adapted for the task. This means that the distance measure used must take into account the time axis in a manner that is elastic. We re-farmiliarize the reader with five time series distance measures: DTW, Edit Distance on Real Sequences (EDR), Time Warp Edit Distance (TWED), Symbolic Aggregate Approximation (SAX), and Symbolic Fourier Approximation (SFA). Then we present the results of a kNN and SVM classifier paired with each distance measure and evaluated on each dataset. We find that the Gaussian SVM paired with DTW distance is the most accurate time series classifier. We also find that a significant reduction in computational cost can be found by using the negative SVM paired with the DTW distance. This is due to the fact that once the SVM model is compiled, all training samples not lieing on the SVM margin may be discarded. In static problems, where the dimensionality is low relative to the size of the training set, the reduction in the computational cost of computing a decision on test samples is significant, and we find later that a similar principle applies to time series data also.

Now, we will move on to a recapitulation of time series and the SVM. SVMs have a well deserved reputation as one of the best classifier methods available. SVMs became popular in the 1990s and many powerful implementations have come available in recent years (Boser et al., 1992; Cortes and Vapnik, 1995).

SVMs can classify data where the classes are not linearly separable by mapping the data into a higher dimension feature space where the classes are linearly separable. This mapping is achieved using a kernel function that is applied to each pair of data points in turn to produce a kernel matrix. If the kernel matrix is positive semi-definite (PSD) then training the SVM is a convex optimization problem and an optimum solution can be found. However, if this kernel matrix is not PSD then the optimality of the SVM training process is not guaranteed.

If the data follows a standard feature vector representation then there are standard kernels available that are PSD and will produce effective classifiers. For a kernel to truly be a kernel in the strictest sense it most correspond to an inner product in it's Hilbert reproducing space, this highly mathematical criterion simplifies to a condition known as positive semi-definiteness. This means that the Gaussian kernel must be paired with a distance measure that is a metric i.e. it must satisfy positive definiteness, symmetry, and the triangle inequality. It is known that the distance measure derived from the following three algorithms do not satisfy the triangle inequality:

– Dynamic Time Warping (DTW) (Keogh and Pazzani, 2001),

[1] http://www.timeseriesclassification.com

- Symbolic Aggregate Approximation (SAX) (with edit distance) (Lin et al., 2003)
- Symbolic Fourier Approximation (SFA) (with edit distance) (Schäfer and Högqvist, 2012b).

It is perhaps worth noting that it is trivial to recognise the latter two cannot be metrics since they are built upon edit distance which is not a metric.

DTW with 1 Nearest Neighbour (1NN) is considered an excellent baseline classifier for time-series (Bagnall et al., 2017), this paragraph acts as a brief digression to explain this before we continue with our analysis of kernel methods below. With SVM classification we must chose a kernel which should act like an inner product in a feature space. Distinct kernels which preserve order i.e. $k_1(x, y) \leq k_1(w, z) \iff k_2(x, y) \leq k_2(w, z)$ will lead to completely different SVM classifiers, the same cannot be said about kNN. For example the Gaussian kernel e^{-x} and the negative kernel $-x$ both reverse and then preserve the order. In other words if $e^{-a} \leq e^{-b} \iff -a \leq -b$. One possible reason why kNN performs better than SVMs in time series classification is that kNN has fewer parameters to tune, and every function of the distance will produce identical results in kNN so long as it is order preserving. This is not true for SVMs, if we have some distance ordering then we can form two distinct similarity measures: the negative kernel and the Guassian kernel. This added parameter: the choice of kernels means that kNN is simpler than SVMs and is perhaps one reason for its dominance in time series analysis.

Another possible reason which these authors intend to explore in later publications is that kNN is superior to SVM because all the time series studied are univariate. Upon careful consideration one can clearly imagine that this is most definitely the case, since kNN is always going to be superior in a one dimension space for datasets that are not composed of time series provided the noise is insignificant.

2 Time Series Classification with Support Vector Machines

2.1 Problem Setting

Time series classification involves training a classifier on time series and using that classifier to predict the target of a novel sample. Firstly, to allow for brevity in explanation we make the definition $[k] = \{1, ..., k\}$. We are given N training samples $\{x^{(i)}\}_{i=1}^{N}$ where each x_i is an ordered set of vectors, called a time series, of (perhaps varying) length m_i, i.e. $x^{(i)} = \{x_j^{(i)}\}_{j=1}^{m_i} = \{x_1^{(i)}, ..., x_{m_i}^{(i)}\}$ where $x_j^{(i)} \in \mathbb{R}^D \quad \forall i \in [N] \quad \forall j \in [m_i]$. D is the spatial dimensionality of the problem setting and is the same for all sample time series. Lastly, associated with each sample $x^{(i)}$ is a target class either $+1$ or -1. The problem is then: given the training data, a collection of time series $\{x^{(i)}\}_{i=1}^{N}$ and class labels $y_1, ..., y_N$ build a function which maps novel time series to ± 1 that best approximates the true hypothesis function which maps the data to

their correct classes. For multi-class classification we have an identical problem setting except that the class labels of the variables are not constrained to ± 1, they can take on any number of discrete class labels.

2.2 Support Vector Machines

The SVM is a state-of-the-art classifier that works by constructing a hyperplane that best fits the data. Without using a non-linear mapping, the algorithm works by solving a quadratic programming problem which maximizes the margin between the two classes where the margin is the space between the two class borders. Unfortunately this simple approach cannot be used to correctly classify data that is not linearly separable. To overcome this flaw SVMs make use of a map which sends the data to a higher dimensional feature space where the data is linearly separable. The separator is a hyperplane in the feature space but not necessarily in the input space. When using SVMs we do not actually need to know the feature map instead it is enough to know only the inner product between feature vectors. This inner product on feature vectors is called the kernel of an SVM. An SVM is only as good as the kernel it uses. The criterion for a function to be a kernel is that it must define an inner product in some feature space.

For a detailed and thorough explanation of training and prediction using SVMs written by the discoverer Vladimir Vapnik please see (Cortes and Vapnik, 1995).

2.3 Positive semi-definiteness

As stated earlier we require the function we are using as a kernel to behave like an inner product in the feature space. It has been shown that for a function to behave like an inner product in a feature space it is necessary and sufficient that it be positive semi-definite (PSD)(Badiane et al., 2018). A function which is not PSD is known as indefinite. A function $K \in \mathbb{R}^{D \times D}$ is PSD if $x \cdot Kx \geq 0 \quad \forall x \in \mathbb{R}^D$. It is trivial to show (and it's in the paper last cited) that a kernel matrix is PSD if all its eigenvalues are non-negative. It is known that the SVM optimization process is a convex cone problem with a global optimal solution when the kernel is PSD, however when the kernel is not PSD then the SVM optimization process may terminate at a local optimum distinct from the global optimum. This sub-optimal performance has led many to attempt to tackle the problem of indefinite kernels.

3 Distance Measures for Time Series

This section provides some detail on the five measures (DTW, TWED, EDR, SAX and SFA). It explains how they work and illustrates the problem that

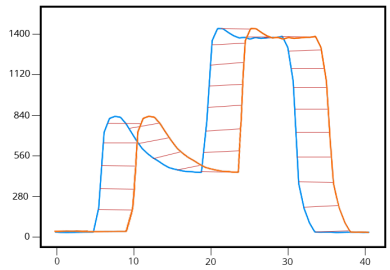

Fig. 1 Misalignment correction by DTW (see (Mahato et al., 2018))

they do not readily produce PSD kernels. Once we posses a distance measure d on time series x and y we can form the kernel in one of two obvious ways:

- The negative kernel: $K(x, y) = -d(x, y)$.
- The Gaussian kernel: $K(x, y) = e^{-\gamma d(x,y)}$.

γ is a parameter to be optimized.

3.1 Dynamic Time Warping

We will re-introduce the DTW algorithm presented by Shimodaira et al. (2002) as well as Cuturi (2011) and Cuturi et al. (2007). We begin this section by the definition of an alignment between two time series. We then define the DTW kernel in terms of those alignments. Both these definitions were made in (Shimodaira et al., 2002).

To compare two time series that do not line up on the x-axis we may use the DTW distance. As you can see in Figure 1 when comparing two time series X and Y that are similar in shape except that the crest of one is shifted along the x-axis, the DTW kernel will reflect this similarity by warping the time axis so that the two time series align. In contrast the Euclidean distance completely ignores the inherent similarity between the two series as a result of the misalignment. In summary DTW is elastic, and Euclidean distance is not.

The DTW distance is not a metric since it does not satisfy the triangle inequality[2] If the distance underlying a kernel fails this test, the kernel will fail to be PSD. A kernel that is not PSD is known as indefinite, and in theory the SVM optimization process involved when using an indefinite kernel need not be a convex quadratic programming problem, as a result the training algorithm is not guaranteed to terminate at a global optimum. This theoretical problem turns out to be of little consequence in experimentation, as we find in the experiments section that very often there is only a tiny margin in

[2] The triangle inequality which is a necessary condition for the distance measure d on a vector space χ to be a metric states: $d(u, v) \leq d(u, w) + d(w, u) \quad \forall u, v, w \in \chi$ i.e. the shortest distance between two points is the straight line (direct route) connecting them.

difference between the Gaussian and negative kernel even though the former nearly always produces PSD kernels while the latter does not. For example with DTW there is no (statistically significant) difference in mean accuracy between the Gaussian and the negative kernels.

To calculate the DTW distance we must first define what is meant by a good alignment. An alignment π is a pair of functions π^1 and π^2 which satisfy the following properties: ($[v] = \{1, ..., v\}$)

$$\pi^1 : [m] \rightarrow [l] \tag{1}$$

$$\pi^2 : [n] \rightarrow [l] \tag{2}$$

where l is known as the length of the alignment.

$$\pi_1^k = 1, \quad for \quad k \in [2] \tag{3}$$

and

$$\pi_l^1 = m \tag{4}$$

$$\pi_l^2 = n \tag{5}$$

$$\pi_i^k - \pi_{i-1}^k \in \{0, 1\} \quad \forall k \in [2] \ \forall i \in \{2, ..., l\} \tag{6}$$

$$\pi_i^a = \pi_{i-1}^a \implies \pi_i^b - \pi_{i-1}^b = 1, \ \forall a, b \in [2], a \neq b, \ \forall i \in \{2, ..., l\} \tag{7}$$

We may summarize the criteria as both π^1 and π^2 must be monotonic functions from $[m]$ and $[n]$ onto $[l]$ such that they contain no simultaneous repetitions (11).

Once we have the alignment π we may define the DTWdistance between two time series $\mathbf{x}$ of length m and $\mathbf{y}$ of length n.

$$d(\mathbf{x}, \mathbf{y}) = \min_{\pi \in \mathcal{A}(\mathbf{x}, \mathbf{y})} (\sum_{k=1}^{l} \|\mathbf{x}(t_{\pi_k^1}) - \mathbf{y}(t_{\pi_k^2})\|) \tag{8}$$

where $\mathcal{A}(\mathbf{x}, \mathbf{y})$ is the set of all possible alignments and $\|.\|$ is the regular Euclidean distance.

We may calculate the DTWdistance in $O(mn)$ via the recurrence relation:

$$M_{i,j} = \min(M_{i-1,j}, M_{i-1,j-1}, M_{i,j-1}) + \|\mathbf{x}_i - \mathbf{y}_j\| \tag{9}$$

The resultant $M_{m,n}$ is the DTWdistance between $\mathbf{x}$ and $\mathbf{y}$. Note it is often customary to use a warping window, this limits the maximum warping that may occur between the two time series. This is trivial to implement since if TOL is our tolerance (maximum warping) then we must simply ensure that when $|i - j| > TOL$: $M_{i,j} = \infty$. By doing this we are ensuring there is an upper bound on the warping. A warping window of 0 is equivalent to Euclidean distance and this means that by considering all warping windows we are also considering Euclidean distance.

DTW is an elastic distance measure in that it measures two time series as similar even when there is misalignment along the time axis.

3.2 Time Warp Edit Distance

Time Warp Edit Distance (TWED) is metric that operates on time series. It was first proposed by Marteau (2008) as an elastic time series measure which unlike DTW serves as a proper metric, satisfying all the axioms of a metric function. The algorithm is both similar to DTW and to EDR. The similarity between two time series is measured as the minimum cost sequence of "edit operations" needed to transform one time series into another. These edit operations are defined in a way which makes sense graphically. To define the "edit operations" they use the paradigm of a graphical editing process to end up with a dynamic programming algorithm that can efficiently compute TWED in roughly the same complexity as DTW.

3.3 Edit Distance on real sequences

The Edit Distance on Real sequences (EDR) distance was first published by Chen et al. (2005). The authors describe an algorithm that is an adaptation of edit on strings of letters to provide a distance measure that works on time series.

The idea is to count the number of edit operations (insert, delete, replace) that are necessary to transform one series into the other. Of course our time series are numerical functions and therefore will almost never likely match up exactly. Therefore we use a parameter ϵ, and if two time series at a particular point in time are within ϵ of each other, we count that as a match otherwise, we don't.

3.4 Symbolic Aggregate Approximation

The approach to time series analysis above is numerical. Here we introduce a symbolic approach to time series analysis: Symbolic Aggregate Approximation (SAX). One possible motivation for moving towards a symbolic approach is that we could then utilize the wealth of data-mining techniques pertaining to string representations, one example would be edit distance. Another source of motivation is that a symbolic approach may yield significant dimensionality reductions. For further explanation see (Lin et al., 2007).

The fist step with SAX is to discretize the input space. First we set the alphabet size ($a > 2$) for the problem. Next for every time series $C = c_1, ..., c_n$ we must assign each c_i to a corresponding variable in the alphabet. So if $a = 3$ and our alphabet is $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ then for the time series $c_1, ..., c_n$ we must map each c_i to a letter in the alphabet. Our approach to discretization is to first transform the data into the Piecewise Aggregate Approximation (PAA) representation and then symbolize the PAA representation into a discrete string. The two main benefits of this process are the well documented dimensionality reduction of PAA (Keogh et al., 2001a; Yi and Faloutsos, 2000) and lower

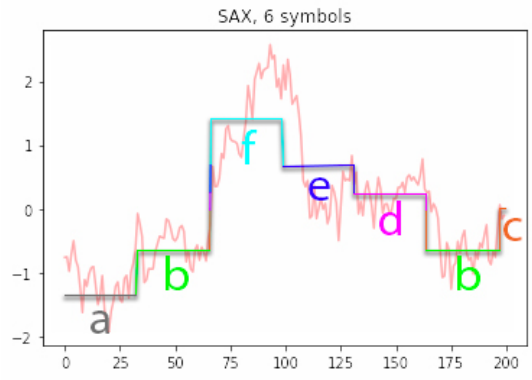

Fig. 2 SAX works on a version of the time-series that has been discretised via PAA (see (Mahato et al., 2018))

bounding: our distance measure between two symbolic string lower bounds the true distance between the original time series (Keogh et al., 2001b; Yi and Faloutsos, 2000).

We can represent a time series C of length n in a w-dimensional space by a vector $\bar{X} = \bar{x_1}, ..., \bar{x_n}$. As in (Lin et al., 2007), we can calculate $\bar{x_i}$ by

$$\bar{x_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} x_j \tag{10}$$

We have reduced the time series from n (temporal) dimensions to w dimensions by dividing the data into w equally sized frames and then $\bar{x_i}$ is simply the mean value of the time series for that frame. We may think of this process as attempting to approximate our time series with a linear combination of box functions. It is worth noting that it is important to z-normalise each time series. We then appropriately define breakpoints which determine the letter in our alphabet to which each $\bar{c_i}$ will be mapped. Usually we do this by analyzing the statistics of our time series and choosing breakpoints so that each letter in the alphabet is as likely to appear as each other letter. In other words we choose breakpoints to spread out the data evenly. For a more thorough explanation see (Lin et al., 2007). Once we have the breakpoints determined it is straightforward to map our PAA representation to a string consisting of letters from our alphabet. The PAA coefficient controls the proportion of examples that will be placed in each bin. Whereas the alphabet size regulates the discretization of the x-axis, the PAA coefficient regulates the discretization of the y-axis. When we have a large time series our approach is to first discretize the time series into a long string and then extract a bag of words. We determine a sliding window, usually found by parameter optimization, and this sliding window length becomes the length of each word in our bag of words. So we turn one long string of letters representing our original time series into a series of words, each word is the length of the sliding window. The first word starts

from the first index in the original long string, the second word starts from the second index in the original long string. We proceed until all the words have been extracted.

As a distance measure between the time series we could use Euclidean distance, however to make our time series analysis more elastic we use edit distance. Now time series that are similar but not aligned will be marked as similar by our distance measure.

3.5 Symbolic Fourier Approximation

Symbolic Fourier Approximation (SFA) was introduced by Schafer et al. in 2012 as an alternative method to SAX built upon the idea of dimensionality reduction by symbolic representation. Unlike SAX which works on the time domain, SFA works on the frequency domain. The algorithm is discussed and developed in (Schäfer and Högqvist, 2012a).

SFA uses the Discrete Fourier Transform (DFT) to represent a time-series as a linear combination of sines and cosines. Recall that $\{\sin(kx), \cos(kx)\}_{k=1}^{\infty}$ forms an orthogonal basis for real (and indeed complex) valued continuous functions. For each time series we perform what is known as an orthogonal projection onto basis functions. Let V be an inner product space over the field $\mathbb{F}$. Let $\mathbf{v} \in V \setminus \{0\}$. We want to decompose an arbitrary vector $\mathbf{y} \in V$ into the form:

$$\mathbf{y} = \alpha\mathbf{v} + \mathbf{z} \tag{11}$$

where $\mathbf{z} \in \{\mathbf{x} | \langle \mathbf{x}, \mathbf{v} \rangle = 0\}$ and $\alpha \in \mathbb{F}$. Since $\mathbf{z} \perp \mathbf{v}$ we have:

$$\langle \mathbf{v}, \mathbf{y} \rangle = \langle \mathbf{v}, \alpha\mathbf{v} + \mathbf{z} \rangle = \langle \mathbf{v}, \alpha\mathbf{v} \rangle + \langle \mathbf{v}, \mathbf{z} \rangle = \alpha \langle \mathbf{v}, \mathbf{v} \rangle \tag{12}$$

$\Longrightarrow$

$$\alpha = \frac{\langle \mathbf{v}, \mathbf{y} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \tag{13}$$

Whence we define the orthogonal projection of y onto v:

$$Proj_{\mathbf{v}}(\mathbf{y}) = \frac{\langle \mathbf{v}, \mathbf{y} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \mathbf{v} \tag{14}$$

.

In this case our inner product is defined as:

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x) g(x) dx \tag{15}$$

and our basis vectors are $\mu_k = \{e^{\frac{i2\pi kn}{N}} | n \in \{0, ..., N-1\}\}$. If k were infinity it would be possible to approximate any real value continuous function to arbitrary precision in much the same was as the Taylor series can be used. However for SFA we use the discrete transformation i.e. $k < \infty$.

The DFT Approximation is a part of the preprocessing step of the SFA algorithm, where all time series data are approximated by computing DFT coefficients. When all these DFT coefficients are calculated, Multiple Coefficient Binning (MCB) is used to turn the approximated time series represented as a series of coefficients into a string representation. Next we use the very simple sliding window technique to extract a bag of features. Our first feature will be the string cut from the first letter to the length of the sliding window. Then we simply slide the window along one letter, starting from the second letter instead of the first this time. Each time we extract a feature from the string and then slide up the window by one and extract the next feature until we have exhausted the string. Once we have mapped each time series to a list of strings of length sliding window, we use edit distance to compare the string representations.

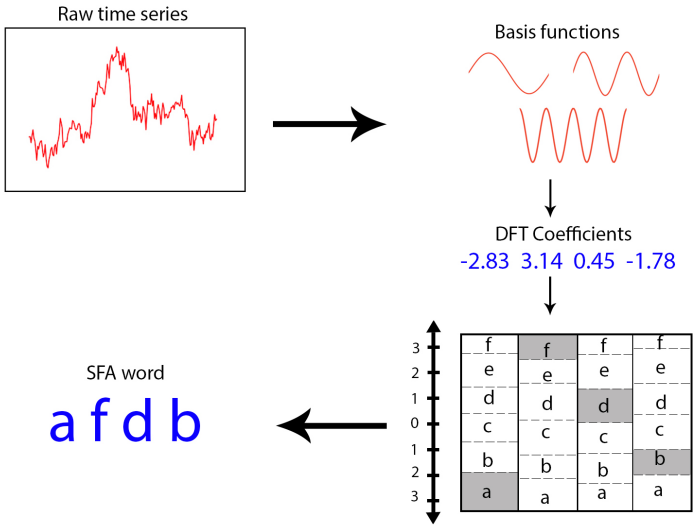

Fig. 3 SFA uses DFT to discretize the time-series (see (Mahato et al., 2018))

4 Experiments

Thirty seven datasets were selected from the UEA & UCR time series repository. We will henceforth present the results of this extensive evaluation: every distance measure presented in this paper (DTW, TWED, SAX, SFA, and EDR) paired with the negative SVM classifier, the Gaussian SVM classifier, and the well-known kNN classifier.

Each dataset comes with a prescribed train:test split. A four-fold cross validation strategy was used and the validation set was used to optimize the parameters of the the 15 classifier-distance models via grid search. In order

to ensure fairness in the parameter selection process each of the 15 classifier-distance model combinations was restricted to a grid search *budget* of 50 states. For each combination 50, distance matrices were constructed to represent 50 parameter combinations. In the case where the parameter space exceeds 50, the parameter combinations where selected from a uniform distribution. Once the optimal parameters where found, the model was built using those parameters together with the full training set and then evaluated on the held-out test set. Accuracy was calculated as expected, the ratio of correct predictions to test set size.

We have three tables (Tables 4, 5 and 6) detailing the results of the three classifiers, respectively kNN, Gaussian SVM and negative SVM. Table 1 summarises the results in these three tables. The main findings are as follows:

- Across all classifier methods DTW is the most effective similarity measure with TWED a close second.
- The best classifier - similarity measure combination is Gaussian DTW.
- It is worth noting that kNN performs well across all similarity measures.
- In contrast, EDR and SFA do no produce good kernels.

Classifier	DTW	TWED	EDR	SAX	SFA	Average
kNN	**0.76**	**0.76**	0.68	0.62	0.68	**0.70**
Gaussian SVM	**0.78**	0.76	0.41	0.6	0.44	0.60
Negative SVM	**0.77**	0.76	0.39	0.55	0.44	0.59
Average	**0.77**	0.76	0.50	0.59	0.52	

Table 1: Summary Table

It is clear that DTW and TWED are the best distance measures across all datasets and across all classifiers. This is perhaps a surprising result as TWED tends to be underrated. This study finds otherwise: TWED is a very good distance measure, of almost equal efficacy to that of DTW. It is perhaps worth noting that the two distance measures are very similar.

Next, we ask which is the most successful classifier-distance pairing (model). There are two ways of approaching this question, first we could construct a league table, where for each dataset we would award 1 point for the least accurate model and full points (at most 15) to the best performing model. This league table is shown in Table 2. We could also simply compute the mean accuracy of each model across all the datasets – this is shown in Table 3.

Model	Points
Gaussian-DTW	372
Negative-DTW	361
kNN-DTW	357
kNN-TWED	353
Negative-TWED	344
Gaussian-TWED	330
kNN-SFA	263
kNN-EDR	257
kNN-SAX	199
Gaussian-SAX	197
Negative-SAX	168
Gaussian-SFA	84
Negative-SFA	83
Gaussian-EDR	62
Negative-EDR	49

Table 2: Classifier-distance pair league table

Classifier	Distance	Mean Accuracy
Gaussian	DTW	0.78
Negative	DTW	0.77
Negative	TWED	0.77
kNN	DTW	0.76
kNN	TWED	0.76
Gaussian	TWED	0.76
kNN	EDR	0.68
kNN	SFA	0.68
kNN	SAX	0.62
Gaussian	SAX	0.60
Negative	SAX	0.55
Negative	SFA	0.44
Gaussian	SFA	0.44
Gaussian	EDR	0.41
Negative	EDR	0.39

Table 3: Mean Accuracy of all models

Both tables agree that the best model for time series classification is the Gaussian-SVM paired with the DTW distance measure. You can think of the latter table as a goal difference score, whereas the former table is the league points, two different measures of performance which both agree that Gaussian-DTW is the superior model amongst all here studied. Finally, we have three tables (Tables 4, 5 and 6) detailing the results of the three classifiers, respectively kNN, Gaussian SVM and negative SVM.

Dataset	DTW	TWED	EDR	SAX	SFA
BirdChicken	0.6	0.6	0.7	0.45	**0.8**
BeetleFly	0.65	0.75	0.6	0.6	**0.85**
Coffee	**0.96**	0.93	**0.96**	0.89	0.75
Beef	**0.63**	0.4	0.4	0.27	0.53
OliveOil	0.77	0.8	0.17	0.8	**0.9**
Wine	**0.65**	0.59	0.43	**0.65**	0.5
FaceFour	0.9	**0.94**	0.91	0.3	0.59
Meat	**0.93**	0.92	0.67	0.83	0.72
Car	0.72	**0.73**	0.72	0.63	0.48
Lighting2	**0.8**	0.75	0.74	0.61	0.7
Herring	0.59	0.53	0.5	**0.62**	0.59
Lighting7	0.71	**0.74**	0.64	0.4	0.44
ToeSegmentation2	0.85	0.76	**0.87**	0.73	0.41
Trace	**0.99**	0.93	0.77	0.65	0.96
ECG200	0.89	**0.9**	0.89	0.8	0.82
ShapeletSim	0.71	**0.82**	0.48	0.46	0.46
Gun Point	0.96	**0.99**	0.97	0.97	0.85
Plane	**1.0**	0.99	0.98	0.99	0.94
ArrowHead	**0.79**	0.72	0.66	0.43	0.75
Ham	**0.67**	0.65	0.55	0.56	0.64
WormsTwoClass	**0.68**	0.63	0.61	0.44	0.62
Worms	**0.5**	0.47	0.36	0.4	0.46
ToeSegmentation1	0.79	**0.81**	0.59	0.55	0.62
DiatomSizeReduction	**0.93**	**0.93**	0.92	0.79	0.73
FISH	0.85	**0.93**	0.76	0.58	0.65
OSULeaf	0.58	**0.72**	0.48	0.37	0.6
Earthquakes	**0.82**	**0.82**	0.8	0.81	0.79
Haptics	**0.41**	**0.41**	**0.41**	0.24	0.3
Computers	0.62	**0.72**	0.68	0.64	0.68
DistalPhalanxOutlineAgeGroup	**0.84**	**0.84**	0.82	0.75	0.77
DistalPhalanxTW	**0.79**	0.74	0.74	0.68	0.76
MiddlePhalanxTW	0.61	**0.63**	0.61	0.58	0.61
MiddlePhalanxOutlineAgeGroup	**0.79**	**0.79**	**0.79**	0.66	0.71
Synthetic Control	**0.97**	0.96	0.83	0.68	0.79
ProximalPhalanxTW	**0.81**	0.8	0.8	0.76	0.78
ProximalPhalanxOutlineAgeGroup	0.84	0.84	0.83	**0.85**	0.81
SonyAIBORobotSurface	0.69	0.69	0.69	0.58	**0.75**
Mean	**0.76**	**0.76**	0.68	0.62	0.68

Table 4: The kNN classifier evaluated on all 37 datasets. For kNN, the most accurate distance measures are DTW, and TWED.

Dataset	DTW	TWED	EDR	SAX	SFA
BirdChicken	0.65	0.7	0.5	**0.85**	0.5
BeetleFly	**0.7**	0.65	0.5	0.45	0.45
Coffee	**1.0**	**1.0**	0.54	0.61	0.54
Beef	**0.67**	0.63	0.2	0.6	0.2
OliveOil	**0.87**	**0.87**	0.4	**0.87**	0.4
Wine	0.78	0.74	0.5	**0.81**	0.5
FaceFour	**0.89**	0.86	0.16	0.36	0.48
Meat	**0.97**	0.93	0.33	0.83	0.33
Car	0.63	**0.68**	0.22	0.38	0.22
Lighting2	0.72	**0.75**	0.54	0.56	0.54
Herring	0.53	**0.59**	**0.59**	**0.59**	**0.59**
Lighting7	**0.84**	0.73	0.26	0.26	0.26
ToeSegmentation2	0.81	**0.85**	0.82	0.53	0.82
Trace	**0.99**	0.89	0.19	0.62	0.19
ECG200	0.75	**0.93**	0.64	0.71	0.85
ShapeletSim	**0.79**	0.46	0.5	0.5	0.5
Gun Point	0.92	**0.98**	0.49	0.89	0.49
Plane	**1.0**	0.97	0.1	0.99	0.1
ArrowHead	0.74	**0.76**	0.39	0.46	0.39
Ham	**0.71**	**0.71**	0.51	0.63	0.51
WormsTwoClass	**0.61**	0.58	0.58	0.58	0.58
Worms	**0.46**	0.42	0.42	0.42	0.42
ToeSegmentation1	**0.86**	0.76	0.53	0.64	0.55
DiatomSizeReduction	**0.93**	0.92	0.3	0.3	0.3
FISH	**0.87**	0.83	0.13	0.55	0.13
OSULeaf	**0.59**	0.52	0.18	0.23	0.18
Earthquakes	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**
Haptics	0.43	**0.47**	0.21	0.39	0.21
Computers	**0.68**	**0.68**	0.5	0.59	0.6
DistalPhalanxOutlineAgeGroup	**0.86**	0.81	0.64	0.81	0.64
DistalPhalanxTW	**0.79**	0.77	0.53	0.74	0.53
MiddlePhalanxTW	0.61	**0.62**	0.21	0.6	0.21
MiddlePhalanxOutlineAgeGroup	0.78	**0.79**	0.27	0.6	0.27
Synthetic Control	**0.98**	0.97	0.17	0.27	0.17
ProximalPhalanxTW	**0.81**	**0.81**	0.45	0.78	0.79
ProximalPhalanxOutlineAgeGroup	**0.86**	0.85	0.49	0.84	0.49
SonyAIBORobotSurface	**0.79**	0.74	0.43	0.63	0.43
Mean	**0.78**	0.76	0.41	0.6	0.44

Table 5: The Gaussian SVM evaluated on 37 datasets. For the Gaussian kernel, DTW and TWED are the most accurate distance measures.

Dataset	DTW	TWED	EDR	SAX	SFA
BirdChicken	**0.8**	0.7	0.5	0.6	0.5
BeetleFly	0.75	**0.85**	0.5	0.45	0.75
Coffee	**1.0**	**1.0**	0.46	0.43	0.46
Beef	0.67	**0.73**	0.2	0.67	0.2
OliveOil	**0.87**	**0.87**	0.4	0.4	0.4
Wine	0.76	**0.87**	0.5	0.57	0.5
FaceFour	**0.89**	0.86	0.3	0.32	0.41
Meat	**0.97**	0.93	0.33	0.8	0.33
Car	0.58	**0.8**	0.22	0.22	0.22
Lighting2	0.66	**0.75**	0.54	0.54	0.54
Herring	**0.59**	**0.59**	**0.59**	**0.59**	**0.59**
Lighting7	**0.89**	0.74	0.26	0.33	0.38
ToeSegmentation2	**0.81**	0.75	0.18	0.73	0.54
Trace	**1.0**	0.9	0.19	0.64	0.19
ECG200	0.72	**0.92**	0.64	0.64	0.82
ShapeletSim	**0.82**	0.48	0.5	0.49	0.5
Gun Point	0.85	**0.97**	0.49	0.87	0.49
Plane	**1.0**	0.98	0.1	0.89	0.1
ArrowHead	0.74	**0.77**	0.3	0.34	0.53
Ham	0.71	**0.72**	0.51	0.46	0.51
WormsTwoClass	0.61	0.58	0.58	**0.64**	0.58
Worms	0.44	0.42	0.42	**0.45**	0.42
ToeSegmentation1	**0.84**	0.6	0.47	0.55	0.47
DiatomSizeReduction	**0.96**	0.93	0.3	0.73	0.59
FISH	**0.86**	0.81	0.13	0.24	0.13
OSULeaf	**0.51**	0.5	0.18	0.31	0.18
Earthquakes	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**
Haptics	0.29	**0.45**	0.21	0.33	0.21
Computers	0.57	**0.68**	0.5	0.61	0.54
DistalPhalanxOutlineAgeGroup	**0.86**	0.81	0.64	0.81	0.74
DistalPhalanxTW	**0.79**	0.77	0.53	0.74	0.53
MiddlePhalanxTW	0.6	**0.64**	0.21	0.61	0.21
MiddlePhalanxOutlineAgeGroup	**0.8**	0.77	0.27	0.27	0.27
Synthetic Control	**0.98**	0.96	0.17	0.19	0.17
ProximalPhalanxTW	0.81	**0.82**	0.45	0.78	0.45
ProximalPhalanxOutlineAgeGroup	0.86	0.85	0.49	**0.87**	0.49
SonyAIBORobotSurface	**0.79**	0.71	0.43	0.43	0.43
Mean	**0.77**	**0.77**	0.39	0.55	0.44

Table 6: The negative SVM evaluated on all 37 datasets. For the negative kernel, DTW and TWED are effectively tied as the best distance measure.

4.1 Computational Cost Reduction

When using an SVM, all training samples that are not support vectors may be discarded. This produces a reduction in computational cost. We investigated the mean support vector density (ratio of support vectors to training samples) and display the results in the following table.

As you can see, the Negative-DTW model produces the greatest saving in computational cost on average. However, upon further investigation we found that for this model the saving tends to increase as samples are added and decreases as time series length is decreased, a completely intuitive result. The

SVM Kernel	Distance	mean SV density	standard deviation
Negative	DTW	0.77	0.16
Gaussian	DTW	0.83	0.16
Negative	TWED	0.86	0.13
Gaussian	TWED	0.89	0.12
Gaussian	EDR	0.91	0.13
Negative	EDR	0.91	0.13
Gaussian	SAX	0.91	0.14
Gaussian	SFA	0.91	0.13
Negative	SAX	0.92	0.14
Negative	SFA	0.93	0.11

Table 7: Support Vector Densities

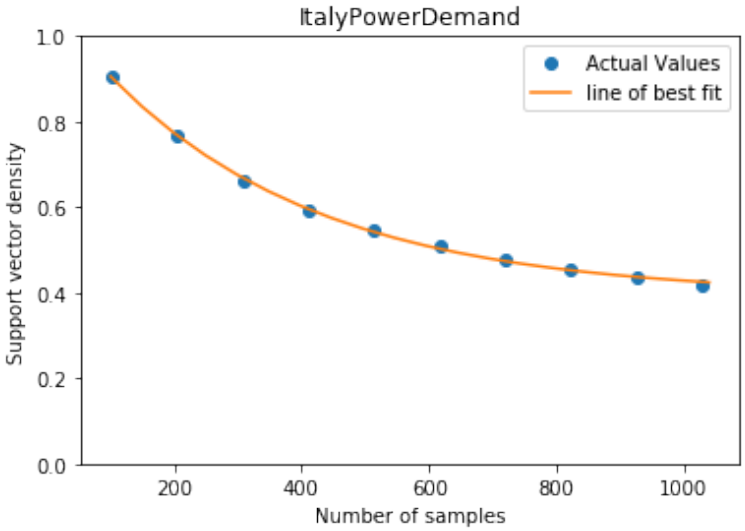

Fig. 4 Support Vector density against subset size for the ItalyPowerDemand dataset..

correlation coefficients between training set size and support vector density; and training set size and time series length where -0.27 and 0.11 respectively, confirming that deduction. Finally, to complete our reduction investigation and to confirm this intuitive result, we computed a Negative-DTW kernel matrix for a much larger dataset the "ItalyPowerDemand" dataset which has 1029 training samples of length 24. The support vector density was very low at only 42%. This means that only 430 of the 1029 training samples would be needed in model prediction. A massive saving in computational cost seeming to confirm the hypothesis that given many samples of relatively short length, the reduction in computational cost derived from using a Support Vector Machine will be substantial. To validate this theory, we selected random subsets of the training set and plotted the support vector density of that subset against the size of the subset in Figure 4.

We must state that techniques exist for kNN-DTW models to reduce computational cost too, by bounding techniques. However, these work best on long time series, and have little or no effect at all on shorter time series. On the other hand, the SVM does the opposite working well for shorter time series, especially when there is a large training set size relative to the length of the time series.

We also ran various weighted league tables, using time series length or dataset size as weights, the results remained unchanged. Gaussian-DTW remains the best model even after weighting.

5 Conclusion

In this paper we have presented experiments with the hopes of identifying time series distance-kernel pairs which provide the highest accuracy. We have found two superior distance-kernel pairs: Gaussian-DTW and Negative-DTW, with Gaussian-DTW having the slight edge over its primary rival. kNN is found to be a robust classifier measure compatible with all distances studies, while DTW and TWED are found to be robust distance measures compatible with all classifiers studied. However, this paper identifies Gaussian-DTW as the best model for time series classfication.

There is an advantage of using an SVM over using kNN and that is the saving that results from discarding training samples. Every training sample that is not a support vector is not used for model prediction. We have found that the Negative-DTW on average retains 77% of samples whereas Gaussian-DTW retains 83% of samples. This saving is marginal, however upon analysis we identified that the support vector density tends to decrease as the size of the training set increases and tends to increase as the length of the time series in the dataset increases. The correlation coefficient in both cases easily lead us to this deduction, and so there is great merit in using an SVM.

While Gaussian DTW provides best accuracy on average over the 37 datasets, Gaussian TWED also performs very well. Given that TWED is a proper metric, there is a case for selecting Gaussian TWED for novel time-series classification applications given that it has a sound theoretical foundation.

Finally, we reflect on the distance measures that do not perform well. While EDR, SAX and SFA sometimes perform well with kNN, they rarely provide good accuracy when paired with SVM. We can conclude that EDR is simply poorly adjusted to time series classification. For SAX and SFA our parameter tuning policy may have had an impact. Their parameter space is massive so the restriction to fifty randomly selected parameter combination may significantly interfere with their performance, and a relaxation of this restriction would most likely improve their performance. However, the restriction is reasonable in a fair competition.

Lastly in summation and conclusion, we have found that SVMs are a state-of-the-art classifier for time series and are indeed very useful in the domain of time series analysis. The Gaussian-DTW SVM outperforms kNN-DTW, the

assumed baseline for time series, and is a better classifier. Not only that, but using an SVM instead of a kNN model results in a significant reduction in computational cost when there are many training samples relative to time series length. We can therefore recommend the Gaussian-DTW SVM for time series classification.

Acknowledgements This publication has resulted from research supported in part by a grant from Science Foundation Ireland (SFI) under Grant Number 16/RC/3872 and is co-funded under the European Regional Development Fund.

References

M. Badiane, M. O'Reilly, and P. Cunningham. Kernel methods for time series classification and regression. In *Proceedings for the 26th AIAI Irish Conference on Artificial Intelligence and Cognitive Science Trinity College Dublin, Dublin, Ireland, December 6-7th, 2018.*, pages 54–65, 2018. URL `http://ceur-ws.org/Vol-2259/aics_7.pdf`.

A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502, 2005.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

M. Cuturi. Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 929–936, 2011.

M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 2, pages II–413. IEEE, 2007.

E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Sigmod Record*, 30(2):151–162, 2001a.

E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001b.

E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–11. SIAM, 2001.

J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD 03*, 2003. doi: 10.1145/882082.882086.

J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2): 107–144, 2007.

V. Mahato, M. O'Reilly, and P. Cunningham. A comparison of k-nn methods for time series classification and regression. In *AICS*, pages 102–113, 2018.

P.-F. Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):306–318, 2008.

P. Schäfer and M. Högqvist. Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 516–527. ACM, 2012a.

P. Schäfer and M. Högqvist. SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 516–527. ACM, 2012b.

H. Shimodaira, K.-i. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *Advances in neural information processing systems*, pages 921–928, 2002.

B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB*, volume 385, page 99. Citeseer, 2000.