Kernel Methods for Time Series Classification and Regression

Mourtadha Badiane, Martin O'Reilly and Pádraig Cunningham

University College Dublin

Abstract. The purpose of this paper is to collect together various SVM kernels presented in literature for time series classification and regression and to test them against each other to see if there is a clear winner. Support Vector Machines (SVMs) are a state-of-the-art set of algorithms which utilize a kernel function in their processing. This paper re-evaluates certain kernels that can be used for time series classification and regression paying particular attention to whether the kernels are positive semi-definite (PSD) and hence admissible. This paper explains the popular dynamic time warping kernel as well as variations. We test the LTW (linear time warping), DTW, and SAX kernels in various forms against each other to see if there is a clear winner on a time series classification and a time series regression problem. We find that a variation of the DTW kernel: the global alignment (GA) kernel wins when the time series are short in length (<100 temporal dimensions) but when the time series get too long the GA kernel values explode to infinity and as a result simpler kernels like LTW with a linear local kernel perform better. Furthermore we find that the original DTW kernel performs poorly as it does not produce PSD kernel matrices.

Keywords— Support Vector Machine, Time Series Regression, Time Series Classification, Dynamic Time Warping, Global Alignment Kernel, SAX

1 Introduction

Support vector machines were developed in Russia in the 60s by Vladimir Vapnik and Alexey Chervonenkis. The Support Vector Machine (SVM) was introduced in [1] and the training algorithm was presented in [3]. They were originally invented for classification but quickly became adapted for regression. Support Vector Machines have been successfully applied to pattern recognition and it is the hope of the authors to transfer that success to time series analysis. This paper serves as an introduction to time series analysis and aims to determine the most promising kernels for use in support vector machine classification and regression. This paper defines the concept of a positive semi-definite kernel in section 2, as well as the motivations for this paper. In section 3 the reader is introduced to perhaps the simplest solution to performing time series analysis on time series of differing lengths, the linear time warping (LTW) kernels (introduced in [7]). In section 4 we introduce the more sophisticated dynamic time warping (DTW) kernels as well as variations. In section 5 we briefly introduce a bag of features approach: Symbolic Aggregate Approximation (SAX) which is a symbolic approach to time series analysis. Section 6 details two experiments on time series classification and regression while section 7 concludes the paper.

2 Positive Semi-Definite (PSD) Kernels

A support vector machine is only as good as the kernel function it employs, in fact, it is helpful to think of a support vector machine as an algorithm which allows us to use a kernel function to make predictions. A kernel function is a measure of similarity between two vectors. A kernel function is meant to correspond to an inner (dot) product in a higher dimensional feature space. A sufficient condition to ensure that the kernel defines a dot-product in a feature space is that all the eigenvalues in its eigenfunction expansion are positive. To guarantee that these coefficients are positive, it is necessary and sufficient (Mercer's Theorem) that the condition (1) be satisfied ([3]). Therefore we have that for a kernel function to be admissible (i.e. possible to be utilized in an SVM) it must be positive semi-definite, if the kernel is not positive semi-definite then the support vector machine algorithm may not terminate at a global optimum, instead it may produce very suboptimal results. The Gram matrix is the matrix $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_i$ is the i^{th} training example and $K(.,.)$ is the kernel function. The kernel function K is positive semi-definite if and only if it satisfies Mercer's condition:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} K(\mathbf{u},\mathbf{v})g(\mathbf{u})g(\mathbf{v})d\mathbf{u}d\mathbf{v} \geq 0 \tag{1}$$

for all g such that:

$$\int_{-\infty}^{\infty} g^2(\mathbf{u})d\mathbf{u} < \infty. \tag{2}$$

In the discrete case we have that for any set of scalars $\{c_1, ..., c_n\}$, $c_i \in \mathbb{R}$ for $i = 1, ..., n$ the following criterion must be satisfied by the Gram (kernel) matrix $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$:

$$\sum_{i,j=1}^{n} c_i K_{ij} c_j \geq 0 \tag{3}$$

We may note that by definition a matrix A is positive semi-definite if and only if $x{\cdot}\, Ax \geq 0 \ \forall x \in \mathbb{R}^d$. Hence for an eigenvector x with corresponding eigenvalue λ

$$x{\cdot}\, Ax = x{\cdot}\, \lambda x = \lambda x{\cdot}\, x = \lambda \|x\|^2$$

We require the above expression to be non-negative which is true when the eigenvector λ is non-negative. So a positive semi-definite matrix must have no negative eigenvalues.

We may go further and show that a symmetric matrix with only non-negative eigenvalues is positive semi-definite. If an $n \times n$ real matrix is symmetric it has n orthogonal eigenvectors $x_1, ..., x_n$ with corresponding eigenvalues $\lambda_1, ..., \lambda_n$. Then the eigenvectors form a basis for $\mathbb{R}^n$. That is any vector y can be written as a linear combination $c_1x_1 + ... + c_nx_n$ for suitable constants c_i. We then have:

$$y{\cdot}\, Ay = \sum_{i=1}^{n} c_i x_i {\cdot}\, A \sum_{i=1}^{n} c_i x_i = \sum_{i=1}^{n} c_i x_i {\cdot} \sum_{i=1}^{n} c_i A x_i$$

now since each x_i is an eigenvalue $Ax_i = \lambda_i x_i$, so we have

$$y \cdot Ay = \sum_{i=1}^{n} c_i x_i \cdot \sum_{i=1}^{n} c_i \lambda_i x_i = \sum_{i,j=1}^{n} \lambda_i c_i c_j x_i \cdot x_j$$

now since $x_1, ..., x_n$ are orthogonal vectors: $x_i \cdot x_j = 0$ if $i \neq j$. This yields:

$$y \cdot Ay = \sum_{i=1}^{n} c_i^2 \lambda_i x_i \cdot x_i = \sum_{i=1}^{n} c_i^2 \lambda_i \|x_i\|^2$$

which is strictly non-negative if all the eigenvalues $\lambda_1, ..., \lambda_n$ are non-negative.

We re-iterate a kernel is a similarity measure so we would like it to be big when two vectors are close and small when applied to vectors which are far. It is also worth noting that we do not directly require the training examples when training our SVM, all that is required is the Gram matrix K_{ij}. The following functions may be used as kernels (for static vectors) ([6]):

- Polynomial (homogeneous): $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$
- Polynomial inhomogeneous: $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$
- Gaussian radial basis function: $K(\mathbf{u}, \mathbf{v}) = e^{-\gamma ||\mathbf{u} - \mathbf{v}||^2}$
- Hyperbolic tangent: $K(\mathbf{u}, \mathbf{v}) = tanh(\kappa \mathbf{u} \cdot \mathbf{v} + c)$

One of the best performing kernels is the radial basis function (RBF) [4] which relies on computing a distance between two vectors $||\mathbf{u} - \mathbf{v}||$. Now these are kernels for static vectors, but our intuition is to extend the RBF kernel to time series data simply by formulating a distance measure which can be applied to time series, such as the dynamic time warping distance, which can then be plugged into the Gaussian RBF kernel exactly as in the static case. Unfortunately, before the reader gets their hopes up, it turns out that the popular dynamic time warping distance is not actually a metric, which means that the kernel induced is not positive semi-definite (PSD) and so is not an admissible kernel. Fortunately for some authors it has been used before successfully for SVM time series analysis [2].

3 Linear Time Warping

One of the first problems that confronts one attempting to perform regression on time series vectors is that the time series may not be of the same length. There is a simple and straightforward way around this problem: linear time warping (LTW), first introduced in [7]. We will simply warp each vector to the same length. In theory we could chose an arbitrary length to which we would map each time series, in reality to guarantee the derived LTW kernel is PSD, so long as the underlying kernel on static vectors is PSD, we should warp each time series in the data set to a vector of fixed length l. The maximum length of all the time series is a good candidate for l, the length to which all the time series will be warped. Let π^i be the warping function for example i, then for the i^{th} training example $\mathbf{x}_i^* = \{\mathbf{x}_{\pi_k^i}\}_{k=1}^{l}$, where in this case l is the length of the longest time series and

$$\pi^i(k) = \lceil \frac{km}{l} \rceil \tag{4}$$

where m is the length of $\mathbf{x}_i$ and $\lceil . \rceil$ denotes the ceiling function. We then can simply calculate the kernel of $\mathbf{x}$ and $\mathbf{y}$ as:

$$K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{l} k_L(\mathbf{x}_{\pi^1(i)}, \mathbf{y}_{\pi^2(i)})$$

where k_L is the local kernel (active on static vectors) for example: linear or quadratic; and π^1 is the ltw function for x ,while π^2 is the ltw function for y. We could also consider the linear time warping RBF kernel defined as:

$$K(\mathbf{x}, \mathbf{y}) = e^{\frac{\sum_{i=1}^{l} \|\mathbf{x}_{\pi^1(i)} - \mathbf{y}_{\pi^2(i)}\|}{\sigma^2}}$$

where σ^2 is the variance.

The time series is warped since every $\mathbf{x}_i^*$ is of length l, simply by repeating certain time points again in the series so that they are all the same length. Linear time warping is attractive since it nearly always produces kernels which are PSD and is a trivial way of comparing time series of different length. The equations above are a summation of those presented in [7].

4 Dynamic Time Warping

So that the paper may retain its self-contained nature, i.e. its completeness we will re-introduce the DTW algorithm presented in [7] as well as [8] and [9] . 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 [7]. The rest of this section deals with variations of the original DTW algorithm described in literature.

4.1 The original algorithm

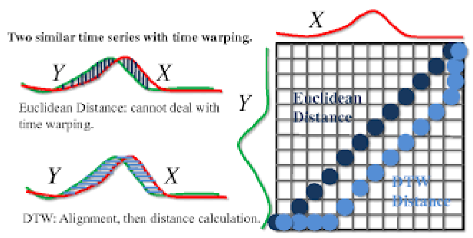
Two similar time series with time warping.
Y
X
Euclidean Distance: cannot deal with time warping.
Y
X
DTW: Alignment, then distance calculation.
X
Y
Euclidean Distance
DTW Distance

Fig. 1. Misalignment correction by DTW

To compare two time series that do not line up on the x-axis we may use the dynamic time warping 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 dynamic time warping distance is not a metric since it does not satisfy the triangle inequality [1]. However on some problems the kernel still performs well. It is usually a good idea to plot the eigenvalues of the Gram (kernel) matrix, many negative eigenvalues usually implies poor performance.

To calculate the dynamic time warping 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: ($[v] = \{1, ..., v\}$)

$$\pi^1 : [m] \to [l] \tag{5}$$

$$\pi^2 : [n] \to [l] \tag{6}$$

where l is known as the length of the alignment.

$$\pi_1^k = 1, \quad for \quad k \in [2] \tag{7}$$

and

$$\pi_l^1 = m \tag{8}$$

$$\pi_l^2 = n \tag{9}$$

$$\pi_i^k - \pi_{i-1}^k \in \{0, 1\} \quad \forall k \in [2] \ \forall i \in \{2, ..., l\} \tag{10}$$

$$\pi_i^a = \pi_{i-1}^a \implies \pi_i^b - \pi_{i-1}^b = 1, \ \forall a, b \in [2], a \neq b, \ \forall i \in \{2, ..., l\} \tag{11}$$

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 dynamic time warping distance between two time series $\mathbf{x}$ of length m and $\mathbf{y}$ of length n.

$$d(\mathbf{x}, \mathbf{y}) = \min_{\pi \in \mathcal{A}(\mathbf{x}, \mathbf{y})} (\sum_{k=1}^{l} \|\mathbf{x}(t_{\pi_k^1}) - \mathbf{y}(t_{\pi_k^2})\|) \tag{12}$$

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

We may calculate the dynamic time warping distance in $O(mn)$ via the recurrence relation:

$$M_{i,j} = min(M_{i-1,j}, M_{i-1,j-1}, M_{i,j-1}) + \|\mathbf{x}_i - \mathbf{y}_j\| \tag{13}$$

The resultant $M_{m,n}$ is the dynamic time warping distance 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.

We may form the kernel in the usual way:

$$K_{GDTW}(\mathbf{x}, \mathbf{y}) = e^{-d(\mathbf{x}, \mathbf{y})^2/\sigma^2} \tag{14}$$

[1] the triangle inequality which is a necessary condition for the distance measure d on a vector space χ to be a metric states: $d(u, v) \leq d(u, w) + d(w, u) \quad \forall u, v, w \in \chi$. If the distance underlying a kernel fails this test, the kernel will fail to be PSD.

where σ is the standard deviation of the distances. This is the Gaussian dynamic time warping kernel: K_{GDTW}. The negative dynamic time warping kernel is defined as:

$$K_{NDTW}(\mathbf{x}, \mathbf{y}) = -d(\mathbf{x}, \mathbf{y}) \tag{15}$$

it is simply the negated DTW distance.

4.2 Variations

We may also utilize DTW with a linear local kernel. This kernel would be defined similar to K_{GDTW}.

$$K_{LDTW} = \max_{\pi \in \mathcal{A}(x,y)} (\sum_{k=1}^{|\pi|} \mathbf{x}_{\pi^1(k)} \cdot \mathbf{y}_{\pi^2(k)}) \tag{16}$$

We may calculate K_{LDTW} in a similar way to K_{GDTW} in $O(mn)$ via the recurrence relation :

$$M_{i,j} = \max(M_{i-1,j} + M_{i-1,j-1} + M_{i,j-1}) + \mathbf{x}_i \cdot \mathbf{y}_j \tag{17}$$

the resulting $M_{m,n}$ is the dynamic time warping method with a linear local kernel.

We may extend the dynamic time algorithm to any non-linear local kernel k_L by modifying equation (16) to:

$$K = \max_{\pi \in \mathcal{A}(\mathbf{x},\mathbf{y})} (\sum_{i=1}^{|\pi|} k_L(\mathbf{x}_{\pi^1(i)}, \mathbf{y}_{\pi^2(i)})) \tag{18}$$

and hence equation (17) would be modified to:

$$M_{i,j} = \max(M_{i-1,j} + M_{i-1,j-1} + M_{i,j-1}) + k_L(\mathbf{x}_i, \mathbf{y}_j) \tag{19}$$

which defines the DTW kernel induced from a local kernel k_L. Some choices for k_L include $x \cdot y$, $(x \cdot y)^n$, and $tanh(x \cdot y)$.

In regular DTW we only consider the path through the cost matrix M which minimizes the total cost and therefore the path which minimizes the cumulative distance between the two time series. The Global Alignment kernel instead considers every single admissible alignment $\pi \in \mathcal{A}(\mathbf{x}, \mathbf{y})$. Fist we define the cumulative cost S of an alignment:

$$S(\pi) = \sum_{i=1}^{|\pi|} \|\mathbf{x}_{\pi^1(i)} - \mathbf{y}_{\pi^2(i)}\| \tag{20}$$

and then the Global Alignment (GA) kernel is defined as

$$K(\mathbf{x}, \mathbf{y}) = \sum_{\pi \in \mathcal{A}(\mathbf{x},\mathbf{y})} e^{S(\pi)} \tag{21}$$

The Global Alignment (GA) kernel is discussed in detail in [8] and [9] where it is shown to produce PSD kernels. We can simply calculate the GA kernel in $O(mn)$ time via the recurrence relation:

$$M_{ij} = (M_{i-1,j} + M_{i,j-1} + M_{i-1,j-1}) e^{\|\mathbf{x}_i - \mathbf{y}_j\|} \tag{22}$$

as you can see the $min()$ in equation (13) is replaced by a summation here, so this kernel is sometimes called the soft-minimum kernel.

5 SAX

The approaches to time series analysis above are 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 datamining 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 [10].

The fist step to 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 ([11], [12]) and lower bounding: our distance measure between two symbolic string lower bounds the true distance between the original time series ([13], [12]).

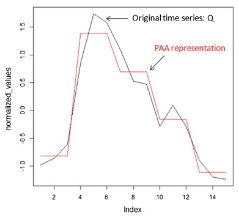

Fig. 2. Discretization via PAA

We can represent a time series C of length n in a w-dimensional space by a vector $\bar{C} = \bar{c_1}, ..., \bar{c_n}$. As in [10], we can calculate $\bar{c_i}$ by

$$\bar{c_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j \tag{23}$$

We have reduced the time series from n (temporal) dimensions to w dimensions by dividing the data into w equally sized frames and then $\bar{c_i}$ 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 [10]. 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 more elastic we use edit distance. Now time series that are similar but not aligned will be marked as similar by our distance metric: edit distance, since it is elastic. It is also worth noting that edit distance is not a metric as it does not satisfy the triangle inequality, however in the experiments reported here, time series discretized with SAX combined with edit distance and a Gaussian RBF-like kernel produces PSD Gram (kernel) matrices.

6 Experiments

For both experiments we can summarise our methods into four categories:

1. Linear time warping (LTW) described previously and paired with various local kernels: ltw - linear, quadratic, quadratic inhomogeneous, cubic, cubic inhomogeneous, tanh, and rbf.
2. Dynamic time warping (DTW) paired with various local kernels and also the variation: the global alignment kernel. The kernels are: GDTW; NDTW; DTW - linear, quadratic, quadratic inhomogeneous, cubic, cubic inhomogeneous, tanh; and the GA kernel.
3. SAX with edit distance. The two kernels are SAX - rbf and SAX - negative.
4. Gaussian Euclidean rbf kernel.

SAX RBF is the edit distance calculated as above (section 5) paired with a Gaussian rbf kernel. SAX - negative is simply the negated edit calculated as above (section 5) used as the kernel. NDTW stands for the negative DTW kernel and is simply the negative of the DTW distance used as the kernel. GDTW stands for Gaussian DTW and is simply the DTW distance plugged into the Gaussian RBF kernel (thus replacing the Euclidean distance). GA stands for the global alignment kernel discussed with the other variations of DTW above.

6.1 Experiment I - Time Series Regression

This experiment consisted of training and testing numerous time series regression models on the Proximal Phalanx dataset available from the UCR archive [14]. The dataset consists of examples which are univariate time series representing bone measurements in an attempt to automate the job of a radiographer in estimating the age of a patient given a radiograph of the non-dominant hand. The dataset consists of 399 training samples and 204 test samples. The target variable is the bone age and there are 6 classes (3-8) however for the purposes of this paper we are treating this as a regression exercise. Every time series is of length 80. The parameters used for the different models are summarized in the Appendix.

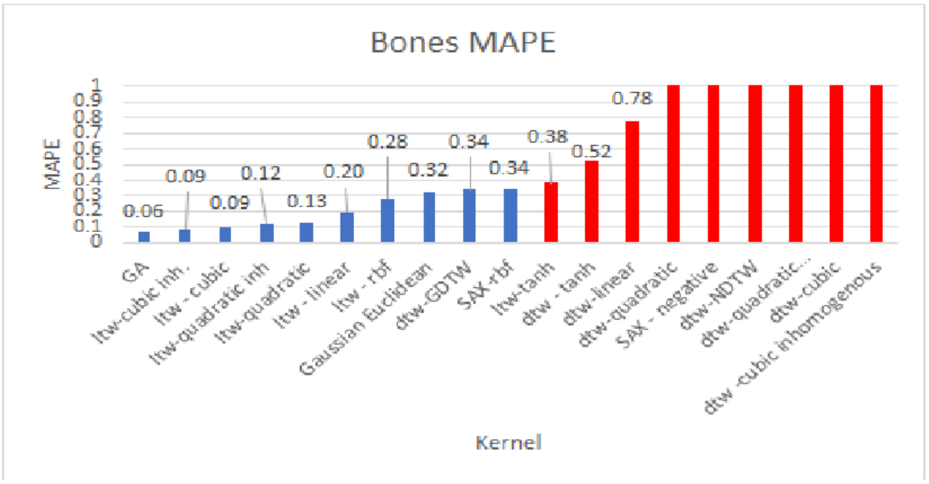

Fig. 3. Experiment I - Proximal Phalanx (Bones)

The results of this experiment are presented in Figure 2. Red bars indicate that the method is not PSD. Here the global alignment kernel (GA) kernel produces a PSD kernel matrix. When that happens the GA usually performs outstandingly well as is the case here. It is an easy winner.

6.2 Experiment II - Time Series Classification

This experiment consisted of training and testing numerous time series classification models on the Jumping dataset [15]. This dataset consists of a univariate time series created by recording with a Shimmer sensor people performing a jumping exercise. It is a classification task with one 'correct' class and two 'incorrect' classes. There are 419 training samples and 179 test samples. The original time series vary in length from 1231 to 6710.

The results of this experiment are shown in Figure 3. Again red corresponds to having at least one negative eigenvalue. We observe again that better performance usually means having a PSD kernel matrix. The simple techniques perform really well here with LTW besting DTW as in the first experiment.

In Figure 5 we can see the eigenvalues of the GTDW method. The largest eigenvalue is 343.76 and has been removed. Negative eigenvalues are coloured red whilst non-negative eigenvalues are coloured blue. We see that GDTW does better than average despite the fact it is not PSD, however the method is far outperformed by the PSD linear time warping with linear local kernel method. The smallest eigenvalue of the GDTW kernel matrix is -0.9178 which is not that negative. This explains why the method does not perform that badly, however those negative eigenvalues are always going to hamper performance. We also note that dtw-GDTW is PSD in the first experiment but not in the second. The method is not in general PSD in that for some datasets it may PSD and for others it may not, exactly as we observe here.

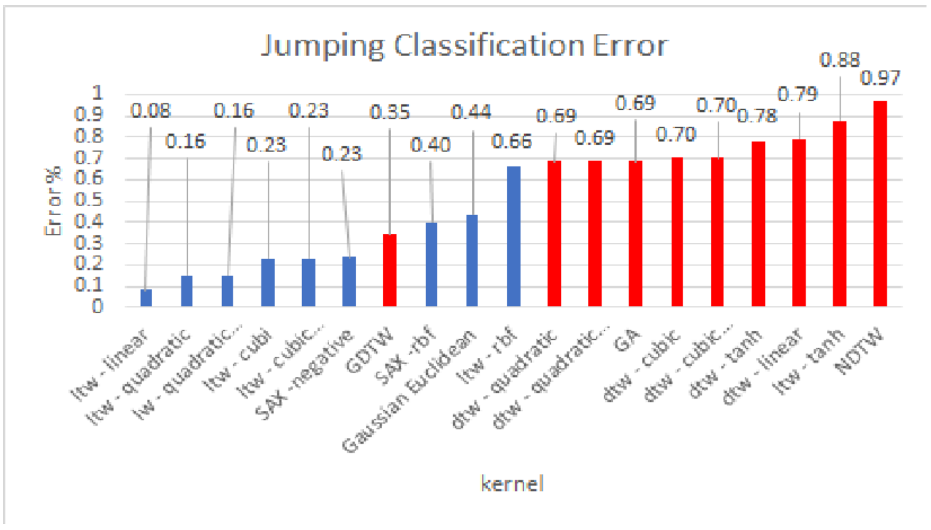

Fig. 4. Experiment II - Jumping

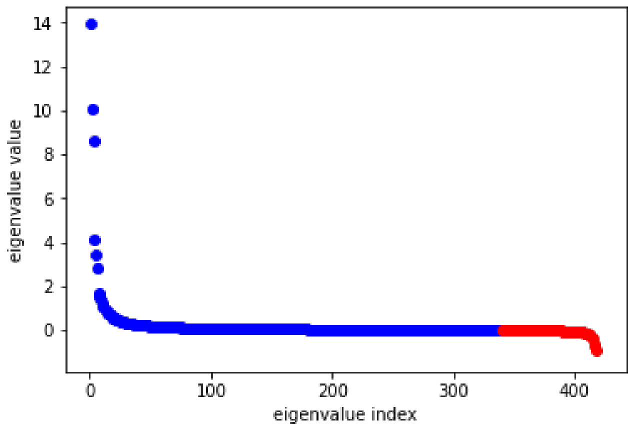

Fig. 5. Eigenvalues of the GDTW kernel matrix (largest eigenvalue 343.76 excluded)

7 Conclusion

It is clear from the results above that usually PSD kernels outperform non-PSD kernels, in the like manner the orignal DTW algorithm fails to produce PSD kernel matrices and therefore performs particularly poorly in both investigations. Linear time warping with a linear local kernel is perhaps the easiest possible way to compare time series of differing lengths, yet despite its simplicity it performs remarkably well. Other techniques though they are significantly more sophisticated struggle to beat linear time warping. Furthermore, the computation of ltw kernels requires only linear time: $O(l)$ where l is the length of the longest time series. This is much more time efficient than dynamic time warping . It is interesting to note that SAX paired with a RBF kernel performed reasonably well on both datasets, moreover it produces PSD kernel matrices. Although it is known that edit distance is not a metric it would be interesting to investigate SAX paired with the RBF kernel further and see will it always produce PSD kernels.

One noteworthy point is that mapping input vectors to a higher dimensional space becomes less effective the higher the input dimensions are ([5]). Basically, as the number of input dimensions increase the linear svm performance improves greatly in comparison to other kernels which involve mapping to a higher dimensional feature space such as polynomial or RBF kernels. This is perhaps why the LTW with a linear kernel performs best on experiment 2 where the input vectors' (temporal) dimensions varies from 1231 to 6710.

It is also worth noting that the global alignment kernel should always produce good results and a PSD kernel. However in experiment 2 above GA failed to produce a PSD kernel and failed to perform well, this was most likely due to the fact that the total number of possible alignments grew so large that the value stored in the computer started to overflow (towards e^{-100}). The GA kernel should always produce PSD matrices. However because values stored in a computer can collapse when they become too large, the GA kernel can break down when the time series are too long. We recommend GA as the superior kernel so long as the time series are not too long or the computer can handle the large magnitude of the numbers being stored. In that case the GA needs no adjusting, it will produce PSD kernels which outperforms both the regular DTW and the simplistic LTW. However when the time series are too long or the computer cannot handle the magnitude of the numbers being stored, the algorithm needs to be adjusted to work on the machine. Unfortunately this adjustment causes the kernel to be no longer PSD and therefore it performs poorly. To understand this remember the global alignment kernel is defined as:

$$K(\mathbf{x}, \mathbf{y}) = \sum_{\pi \in \mathcal{A}(\mathbf{x},\mathbf{y})} e^{S(\pi)} \tag{24}$$

Now when x and y are time series of length 40,000 (as in the Jumping dataset) then $|\mathcal{A}(\mathbf{x}, \mathbf{y})|$ is the Dellanoy number $D_{40000,40000}$ which is well over $2^{40000} = 10^{12041}$ a massive massive number. This number is (the index of) what we are summing over to calculate the kernel on x and y.

In summation, we recommend to use the simple LTW technique as DTW is poorly suited for SVMs since it does not produce PSD kernels and therefore it usually performs poorly. For a sophisticated technique to beat LTW, one should try the global alignment kernel (GA) so long as the time series are not too long that the numbers would explode. GA like DTW takes into consideration misalignment on the x-axis (warping) but unlike DTW should produce PSD kernel matrices.

Acknowledgments

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

1. Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. *A Training Algorithm for Optimal Margin Classifiers.*
2. Gudmundsson, Steinn, Thomas Philip Runarsson, and Sven Sigurdsson. *Support vector machines and dynamic time warping for time series.*
3. Corinna Cortes, and Vladimir N. Vapnik. *Support-vector networks.*
4. Justino, Edson JR, Flavio Bortolozzi, and Robert Sabourin. *A comparison of SVM and HMM classifiers in the off-line signature verification.*
5. Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. *A practical guide to support vector classification.*
6. $https://en.wikipedia.org/wiki/Support_vector_machine$
7. Shimodaira, H., Noma, K. I., Nakai, M., Sagayama, S. *Dynamic time-alignment kernel in support vector machine.*
8. Cuturi, Marco. *Fast global alignment kernels.*
9. Cuturi, Marco, et al. *A kernel for time series based on global alignments.*
10. Lin, J., Keogh, E., Wei, L., Lonardi, S. (2007). *Experiencing SAX: a novel symbolic representation of time series.*
11. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S. (2001) *Locally adaptive dimensionality reduction for indexing large time series databases*
12. Yi, B. K., Faloutsos, C. (2000, September). *Fast time sequence indexing for arbitrary Lp norms.*
13. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S. (2001). *Dimensionality reduction for fast similarity search in large time series databases.*
14. Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista (2015), *The UCR Time Series Classification Archive*, www.cs.ucr.edu/~eamonn/time_series_data/
15. V. Mahato, M. O'Reilly, P. Cunningham (2018) *A Survey of k-NN Methods for Time Series Classification and Regression* under reviews for AICS 2018.

Appendix

Parameters for Experiment 1

- GDTW & NDTW - warping window: 2.
- SAX (negative & Gaussian) - sliding window: 15; PAA coefficient: 10; alphabet size: 5; numurosity reduction: none; Z threshold: 0.01

Parameters for Experiment 2

- GDTW & NDTW - warping window: 26
- SAX (negative & Gaussian) - sliding window: 10; PAA coefficient: 10; alphabet size: 15; numurosity reduction: none; Z threshold: 0.1.