The Path Kernel

Andrea Baisero, Florian T. Pokorny, Danica Kragic and Carl Henrik Ek
KTH Royal Institute of Technology
{baisero,fpokorny,danik,chek}@kth.se

Keywords: Kernel methods, Sequential modelling

Abstract: Kernel methods have been used very successfully to classify data in various application domains. Traditionally, kernels have been constructed mainly for vectorial data defined on a specific vector space. Much less work has been addressing the development of kernel functions for non-vectorial data. In this paper, we present a new kernel for encoding sequential data. We present our results comparing the proposed kernel to the state of the art, showing a significant improvement in classification and a much improved robustness and interpretability.

1 INTRODUCTION

Machine learning methods have had an enormous impact on a large range of fields such as computer vision, robotics and computational biology. These methods have allowed researchers to exploit evidence from data to learn models in a principled manner. One of the most important developments has been that of kernel methods (Cristianini and Shawe-Taylor, 2006) which embed the input data in a potentially high-dimensional vector space with the intention of achieving improved robustness of classification and regression techniques. The main benefit of kernel methods is that, rather than defining an explicit feature space that has the desired properties, the embedding is characterised implicitly through the choice of a kernel function which models the inner product in an induced space. This creates a very natural paradigm for recovering the desired characteristics of a representation. Kernel functions based on a stationary distances (usually an $\mathcal{L}_p - norm$) have been particularly successful in this context (Buhmann and Martin, 2003). However, for many application domains, the data does not naturally lend itself to a finite dimensional vectorial representation. Symbolic sequences and graphs, for example, pose a problem for such kernels.

For non-vectorial data, the techniques used for learning and inference are generally much less developed. A desirable approach is hence to first place the data in a vector space where the whole range of powerful machine learning algorithms can be applied. Simple approaches such as the Bag-of-Words model, which creates a vectorial representation based on occurrence counts of specific representative "words", have had a big impact on computer vision (Sivic and Zisserman, 2009). These methods incorporate the fact that a distance in the observed space of image features does not necessarily reflect a similarity. Another approach, where strings are transformed into a vectorial representation before a kernel method is applied, has been the development of string kernels (Lodhi et al., 2002; Saunders et al., 2002). Such kernels open up a whole range of powerful techniques for non-vectorial data and have been been applied successfully to robotics (Luo et al., 2011), computer vision (Li and Zhu, 2006) and biology (Leslie and Kuang, 2004). Other related works are based on convolution kernels (Haussler, 1999), which recover a vectorial representation that respects the structure of a graph. Another approach to define an inner product between sequences is to search for a space where similarity is reflected by "how well" sequences align (Watkins, 1999; Cuturi et al., 2007; Cuturi, 2010).

In this paper, we present a new kernel for representing sequences of symbols which extends and further develops the concept of sequence alignment. Our kernel is based on a *ground space* which encodes similarities between the symbols in a sequence. We show that our kernel is a significant improvement compared to the state of the art both in terms of computational complexity and in terms of its ability to represent the data.

The paper is organised as follows: In Sec. 2, we outline the problem scenario and provide a structure for the remaining discussion. Sec. 3 introduces our approach and Sec. 4 provides the experimental results. Finally, we conclude with Sec. 5.

2 KERNELS AND SEQUENCES

Before we proceed with describing previous work for creating kernel induced feature spaces for sequences, we will clarify our notation and our notion of kernels. When discussing kernels in the context of machine learning, we have to distinguish between several uses of the word kernel. In this paper, a kernel denotes any symmetric function $k : X \times X \to \mathbb{R}$, where X is a non-empty set (Haasdonk and Burkhardt, 2007). A positive semi-definite (psd) kernel is a kernel $k : X \times X \to \mathbb{R}$ such that $\sum_{i,j}^{n} c_i c_j k(x_i, x_j) \geqslant 0$ for any $\{x_1, \ldots, x_n\} \subset X$, $n \in \mathbb{N}$ and $c_1, \ldots c_n \in \mathbb{R}$. If the previous inequality is strict when $c_i \neq 0$ for at least one $i \in \{1, \ldots, n\}$, the kernel is called positive definite (pd). Further specialisations, such as negative definite (nd) kernels, exist and are of independent interest.

While strong theoretical results on the existence of embeddings corresponding to psd kernels exist (Berlinet and Thomas-Agnan, 2004, p.22), non-psd kernel functions can still be useful in applications. Examples of kernels that are known to be neither pd nor psd, but which are still successfully used in classification include (Haasdonk, 2005). On another note, there are also kernels which are conjectured to be psd, and which have been shown to be psd in experiments, but for which there currently is no proof for the corresponding positiveness (Bahlmann et al., 2002).

In this work, we consider finite sequences of symbols belonging to an alphabet set Σ, i.e. $s = (s_1, s_2, \ldots, s_{|s|})$ denotes such a sequence, with $s_i \in \Sigma$, and where $|s| \in \mathbb{N}_0$ denotes the length of the sequence. We denote a subsequence $(s_a, \ldots, s_b)$, for $1 \leqslant a < b \leqslant |s|$, by $s_{a:b}$. When the indices a or b are omitted, they implicitly refer to 1 or $|s|$ respectively. The *inverse* of a sequence s is defined by $inv(s)_i = s_{|s|-(i-1)}$.

In this work, we assume that we are given a psd kernel function $k_\Sigma : \Sigma \times \Sigma \to \mathbb{R}$ describing the similarity between elements of the alphabet Σ and will refer to k_Σ as the ground kernel.

Given k_Σ, we can now define the *path matrix*.

Definition 1 (Path Matrix)**.** *Given two finite sequences s,t with elements in an alphabet set Σ and a kernel $k_\Sigma : \Sigma \times \Sigma \to \mathbb{R}$, we define the* path matrix *$G(s,t) \in \mathbf{R}^{|s| \times |t|}$ by $[G(s,t)]_{ij} = k_\Sigma(s_i, t_j)$.*

We denote $\delta_{00} \overset{\text{def}}{=} (0,0)$, $\delta_{10} \overset{\text{def}}{=} (1,0)$, $\delta_{01} \overset{\text{def}}{=} (0,1)$, $\delta_{11} \overset{\text{def}}{=} (1,1)$ and $S \overset{\text{def}}{=} \{\delta_{10}, \delta_{01}, \delta_{11}\}$. S is called the set of admissible steps. A sequence of admissible steps starting from $(1,1)$ defines the notion of a path:

Definition 2 (Path)**.** *A* path *over a $m \times n$ path-matrix M is a map $\gamma : \{1, \ldots, |\gamma|\} \to \mathbf{Z}_{>0} \times \mathbf{Z}_{>0}$ such that*

$$\gamma(1) = (1,1), \tag{1}$$

$$\gamma(i+1) = \gamma(i) + \delta_i, \tag{2}$$

$$\textit{for } 1 \leqslant i < |\gamma|, \textit{ with } \delta_i \in S,$$

$$\gamma(|\gamma|) = (m,n). \tag{3}$$

$|\gamma|$ and δ_i denote the path's length *and i^{th}* step *respectively. Furthermore, we adopt the notation $\gamma(i) = (\gamma_X(i), \gamma_Y(i))$. A path defines* stretches*, or alignments, on the input sequences according to $s_{\gamma_X} = (s_{\gamma_X(1)}, \ldots, s_{\gamma_X(|\gamma|)})$ and $t_{\gamma_Y} = (t_{\gamma_Y(1)}, \ldots, t_{\gamma_Y(|\gamma|)})$.*

We denote the set of all paths on a $m \times n$ matrix as $\Gamma(m,n)$. Its cardinality is equal to the Delannoy number $\mathrm{Del}(m,n)$.

2.1 Sequence similarity measures

A popular similarity measure between time-series is Dynamic Time Warping (DTW) (Sakoe and Chiba, 1978; Gudmundsson et al., 2008), which determines the distance between two sequences s and t as the minimal score obtained by all paths, i.e.

$$d_{DTW}(s,t) = \min_{\gamma \in \Gamma} D_{s,t}(\gamma), \tag{4}$$

where $D_{s,t}$ represents the score of a path γ defined by

$$D_{s,t}(\gamma) = \sum_{i=1}^{|\gamma|} \varphi(s_{\gamma_X(i)}, t_{\gamma_Y(i)}), \tag{5}$$

where φ is some given similarity measure. However, DTW lacks a geometrical interpretation in the sense that it does not necessarily respect the triangle inequality (Cuturi et al., 2007). Furthermore, this similarity measure is not likely to be robust as it only uses information from the minimal cost alignment.

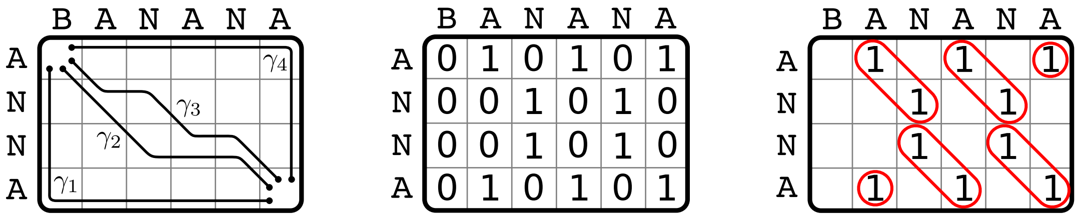
Taking the above into consideration, Cuturi suggests a kernel referred to as the *Global Alignment Kernel* (Cuturi et al., 2007). Instead of considering the minimum over all paths, the Global Alignment Kernel combines all possible path scores. The kernel makes use of an exponentiated soft-minimum of all scores, generating a more robust result which reflects the contents of all possible paths:

$$k_{GA}(s,t) = \sum_{\gamma \in \Gamma} e^{-D_{s,t}(\gamma)}. \tag{6}$$

By taking the ground kernel to be $k_\Sigma(\alpha, \beta) = e^{-\varphi(\alpha,\beta)}$, k_{GA} can be described using the path matrix as

$$k_{GA}(s,t) = \sum_{\gamma \in \Gamma} \prod_{i=1}^{|\gamma|} G(s,t)_{\gamma(i)}. \tag{7}$$

The leading principle in this approach is hence a combination of kernels on the level of symbols over all



(a) (b) (c)

Figure 1: *Illustration of the concept of paths and the contents of $G(s,t)$ for $s =$ "ANNA" and $t =$ "BANANA". On the left, we illustrate a small number of paths which traverse G. The path kernel makes use of these, together with all the other paths, to collect data from the matrix and to extract a final score. In the center, we display the contents of $G(s,t)$, assuming $k_\Sigma(\alpha,\beta) = \delta_{\alpha\beta}$, i.e. Kroenecker's delta function. On the right, we highlight the corresponding diagonals whose number, length and position relate to the similarity between the subsequences of s and t.*

paths along $G(s,t)$. Cuturi shows that incorporating all the elements of $G(s,t)$ into the final results can vastly improve classification compared to using only the minimal cost path. Furthermore, k_{GA} is proven to be psd under the condition that both k_Σ and $\frac{k_\Sigma}{1+k_\Sigma}$ are psd (Cuturi et al., 2007), giving foundation to its geometrical interpretation.

However, Cuturi's kernel makes use of products between ground kernel evaluations along a path. This implies that the score for a complete path will be very small if $\varphi(s_i,t_j)$ is sufficiently large, which leads to the problem of diagonally dominant kernel matrices (Gudmundsson et al., 2008; Cuturi, 2010) from which the global alignment kernel suffers. The issue is particularly troubling when occurring at positions near the top-left or bottom-right corners of the path matrix because it will affect many of the paths. Furthermore, paths contribute with equal weight to the value of the kernel. To reduce this effect it is suggested in (Cuturi et al., 2007) to rescale the kernel values and use its logarithm instead. We argue that paths which travel closest to the main diagonal of the path matrix should be considered as more important than others, as they minimise the distortion imposed on the input sequences, i.e. s_{γ_X} and t_{γ_Y} are then most similar to s and t. To rectify this and to include a preference towards diagonal paths, a generalisation called the *Triangular Global Alignment Kernel* was developed, which considers only a subset of the paths (Cuturi, 2010). This generalisation imposes a crude preference for paths which do not drift far away from the main diagonal.

In this paper, we develop a different approach by introducing a weighting of the paths in Γ based on the number of diagonal and off-diagonal steps taken. We manipulate the weights to encode a preference towards consecutive diagonal steps while at the same time accumulating information about all possible paths. Furthermore, by replacing the accumulation of symbol kernel responses along the path using a summation rather than a product, the kernel's evaluation reflects more gracefully the structure of the sequences and avoids abrupt changes in value.

3 THE PATH KERNEL

In this section, we will describe our proposed kernel which we will refer to as the *path kernel*.

Fig. 1 illustrates the contents of a path matrix in a simplified example, showing the emergence of diagonal patterns when the two sequences are in good correspondence. Table 1 shows the resulting alignments associated with the paths shown in Figure 1. We argue that the values, the length and the location of these diagonals positively reflect the relation between the inputs and should thus be considered in the formulation of a good kernel. High values imply a good match on the ground kernel level, while their length encodes the extent of the match. On the other hand, the position relative to the main diagonal reflects how much the input sequences had to be "stretched" in order for the match to be encountered. We wish to have a feature space where a smaller stretch implies a better correspondence between the sequences.

Let us now define a new kernel that incorporates different weightings depending on the steps used to travel along a path.

Definition 3 (Path Kernel)**.** *For any sequences s,t, we define*

$$k_{PATH}(s,t) \overset{\text{def}}{=} \begin{cases} k_\Sigma(s_1,t_1) & \\ \quad + C_{HV} k_{PATH}(s_{2:},t) & \\ \quad + C_{HV} k_{PATH}(s,t_{2:}) & |s| \geqslant 1 |t| \geqslant 1, \\ \quad + C_D k_{PATH}(s_{2:},t_{2:}) & \\ 0 & \textit{otherwise}, \end{cases} \tag{8}$$

where C_{HV} and C_D represent weights assigned to vertical or horizontal steps and diagonal steps respectively. By enforcing the constraints $C_{HV} > 0$ and $C_D > C_{HV}$, we aim to increase the relative importance of paths with many diagonal steps.

stretches			
γ_1	A N N $\underline{\text{A A A A A A}}$	γ_2	A N $\underline{\text{N N N}}$ A
	$\underline{\text{B B B B}}$ A N A N A		B A N A N A
γ_3	A $\underline{\text{N N}}$ $\underline{\text{N N}}$ A	γ_4	$\underline{\text{A A A A A A}}$ N N A
	B A N A N A		B A N A N $\underline{\text{A A A A}}$

Table 1: Stretches associated to the paths in Fig. 1a with the underlined substrings denoting a repeated symbol. Note that, even though γ_2 and γ_3 produce the same stretches, they traverse the matrix differently and should thus be considered separately.

The symmetry of the kernel is easily verifiable. On the other hand, positive definiteness remains to be proven, although all our experimental results have yielded psd kernel matrices in practice.

3.1 Efficient Computation

Kernel methods often require the computation of a kernel function on a large dataset, where the number of kernel evaluations will grow quadratic with the number of data-points. It is hence essential that the kernel evaluations themselves are efficiently computable.

Not only can the path kernel be evaluated using a Dynamic Programming algorithm which avoids the expensive recursion in (8) and which achieves a computational complexity comparable with DTW and k_{GA}, but it can also be computed very efficiently using the following alternative formulation:

$$k_{PATH}(s,t) = \sum_{ij} G(s,t)_{ij}\, \omega_{PATH\,ij}, \tag{9}$$

$$[\omega_{PATH}]_{ij} = \sum_{d=0}^{\min(i,j)-1} C_{HV}^{i+j-2-2d} C_D^d (d, i-1-d, j-1-d)!. \tag{10}$$

The usefulness of (9) comes from the fact that the contents of the weight matrix ω_{PATH} do not really depend on s,t since ω_{PATH} can in fact be pre-computed up to an adequate size[1] (see Figure 2). After this, the evaluation of the kernel for input of sizes m and n is achieved by simply selecting the sub-matrix ranging from $(1,1)$ to (m,n); the remaining matrix multiplication can then be efficiently carried out. By taking advantage of this, one can evaluate the kernel at speeds depending only on the speed of the evaluation of G and the speed of a simple matrix multiplication (with the initial overhead consisting of the pre-computation of ω_{PATH}). The weight matrix can also be computed through an efficient and very simple Dynamic Programming algorithm similar to the one which can be used to evaluate the kernel itself.

[1]For any specific dataset, that would be the length of the longest sequence.

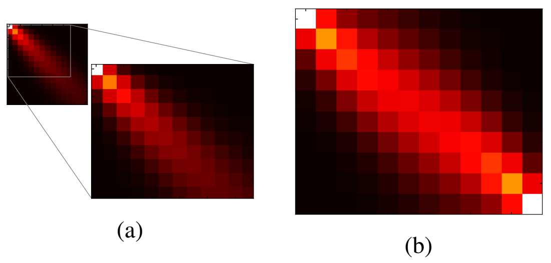


Figure 2: *On the left, a precomputed* 15×15 *weight matrix with* $C_{HV} = .3$ *and* $C_D = .34$ *is used to select a* 10×12 *weight matrix which can then be used to evaluate* $k_{PATH}(s,t)$ *for input sizes* $|s| = 10$ *and* $|t| = 12$. *On the right, the inversion invariant* $\tilde{\omega}_{PATH}$ *corresponding to* $\tilde{k}_{PATH}$ *for the same input sizes is displayed.*

We call a kernel satisfying $k(s,t) = k(inv(s), inv(t))$ inversion invariant. If a kernel k does not naturally have this property, it can be enforced by replacing k with

$$\tilde{k}(s,t) = \frac{k(s,t) + k(inv(s), inv(t))}{2}. \tag{11}$$

The path kernel is not originally inversion invariant, but invariance can be enforced without the need for a double computation of the kernel for each evaluation. This is done by modifying the selected sub-matrix of ω_{PATH} as follows: for any two inputs with lengths m and n, we replace the weight matrix ω_{PATH} by

$$[\tilde{\omega}_{PATH}]_{ij} = \frac{\omega_{PATH\,ij} + \omega_{PATH\,m-i+1,n-j+1}}{2}, \tag{12}$$

and then proceed using this weight matrix.

3.2 Ground Kernel Choice

The path kernel is based on a ground kernel which, apart from being a psd function, is not constrained in any other way. However, we show in this paragraph that an arbitrarily k_Σ may lead to undesirable results.

Assume an alphabet and a ground kernel such that $\alpha, \beta \in \Sigma$, $k_\Sigma(\alpha,\alpha) = k_\Sigma(\beta,\beta) = 1$ and $k_\Sigma(\alpha,\beta) = -1$. Given the input sequences $s = (\alpha, \beta, \ldots, \alpha, \beta)$ and $t = (\beta, \alpha, \ldots, \beta, \alpha)$, one may be inclined to say that s and t are very similar because each can be obtained from the other by cyclically shifting the symbols by one position. However, the contents of $G(s,t)$ show a collection of ones and negative ones organised in a chessboard-like disposition. This obviously leads to heavy fluctuations during the computation of $k_{PATH}(s,t)$ and to potentially very small values. Furthermore, the issue is present even in the computation of $k_{PATH}(s,s)$ and $k_{PATH}(t,t)$ which is not desirable under any circumstance.

This problem is however easily rectifiable by considering only ground kernels that yield non-negative results on elements of Σ.

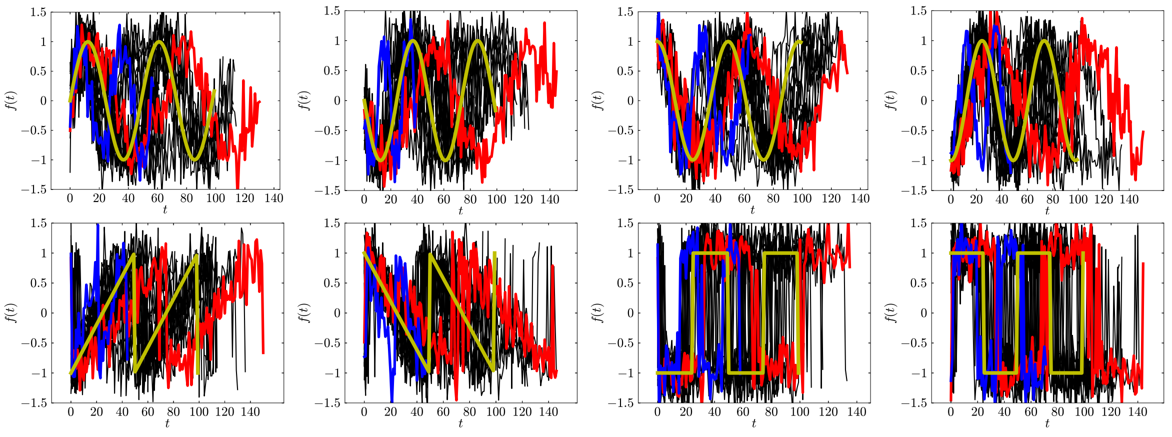


Figure 3: *The figure above shows the eight different waveforms used for the classification for a noise level corresponding to* $\sigma_{\{l,i,o\}} = 5$. *The golden curve depicts the base waveform without noise while the blue and red curves show the shortest and the longest noisy example respectively. The black curves display the remaining examples in the database.*

4 EXPERIMENTS

In this section, we present the results of experiments performed with the proposed kernel. In particular, we perform two separate quantitative experiments that, in addition to our qualitative results, shed some light on the behaviour of the proposed method in comparison to previous work. In the first experiment, we generate eight different classes of univariate sequences. Each class consists of a periodic waveform namely $\pm$*sine*, $\pm$*cosine*, $\pm$*sawtooth* and $\pm$*square*. From these, we generate noisy versions by performing three different operations:

- The length of the sequence is generated by sampling from a normal distribution $\mathcal{N}(100, \sigma_l^2)$, rounding the result and rejecting non-positive lengths.
- We obtain an *input* sequence as $|s|$ equidistant numbers spanning 2 periods of the wave; we then add to each element an *input* noise which follows a normal distribution $\mathcal{N}(0, \sigma_i^2)$.
- We feed the noisy input sequence to the generating waveform and get an *output* sequence, to which we add *output* noise which follows a normal distribution $\mathcal{N}(0, \sigma_o^2)$,

Figure 3 shows the sequences for the parameter setting $\sigma_{\{l,i,o\}} = 5$. This corresponds to the setting which we will use to present our main results.

We will compare our approach to k_{GA} as well as the non-psd kernel obtained by using the negative exponential of the DTW distance,

$$k_{DTW}(s,t) = e^{-d_{DTW}(s,t)}. \qquad (13)$$

The path kernel has two different sets of parameters: the ground kernel and the weights associated with steps in the path matrix. In our experiments, we use a simple zero mean Gaussian kernel with standard deviation 0.1 as ground kernel. The step weights C_{HV} and C_D are set to 0.3 and 0.34 respectively. We use the same setting throughout the experiments. The behaviour of the kernel will change with the value of these parameters. A complete analysis of this is however beyond the scope of this paper. Here, we focus on the general characteristics of our kernel which summarises all possible paths using step weights satisfying $C_{HV} < C_D$ – implying a preference for diagonal paths.

In order to get an understanding of the geometric configuration of the data that our kernel matrices corresponds to, we project the three different kernels onto their first two principal directions in Figure 4. It is important to note that, as the DTW kernel has negative eigenvalues, it does not imply a geometrically valid configuration of datapoints in a feature space. From Figure 4, we get a qualitative understanding of how the induced feature spaces looks like. However, a representation is simply the means to an end and to be able to make a valuable assessment of its useability, we need to use it to achieve a task. We do so through two different experimental setups: The first is meant to test the discriminative capabilities of the representation; the second evaluates how well the representation is suited for generalisation.

In order to test the discriminability of the feature space generated by the path kernel, we perform a classification experiment using the same data as explained above, and where the task is to predict the generating class of a waveform. We feed the kernel matrix into an SVM classifier (Chang and Lin, 2001), use a 2-fold cross-validation, and report the average over

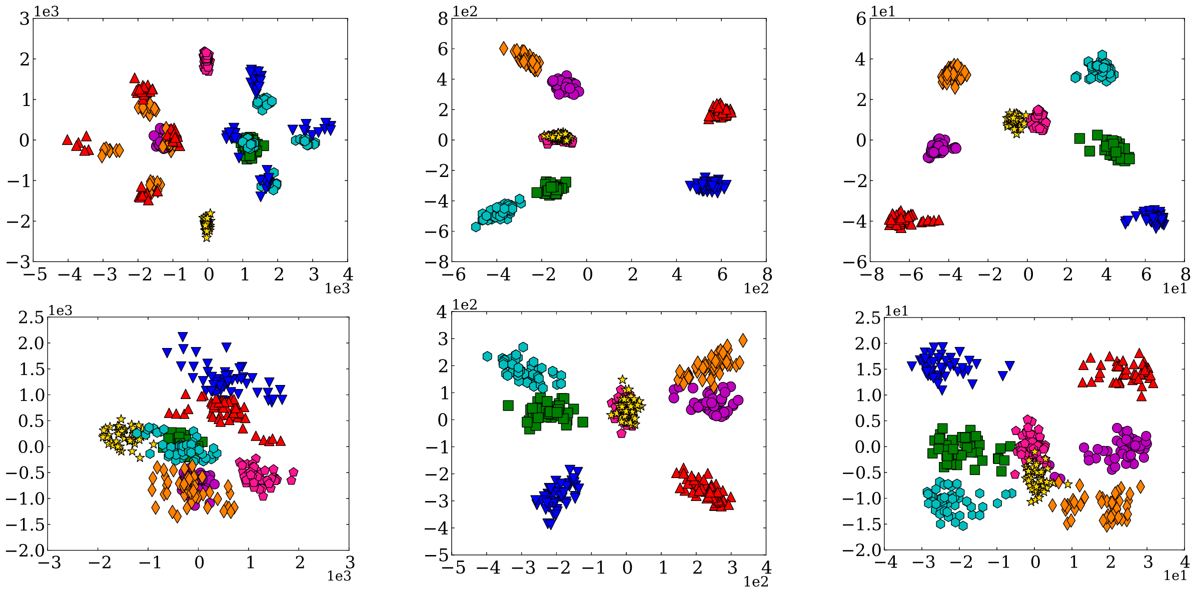


Figure 4: *The above figure displays the two dimensional principal subspace for the Global Alignment Kernel (left), Dynamic Time Warping (middle) and the Path Kernel (right). The first row has a generating noise with* $\sigma_{\{l,i,o\}} = 2$*, while, in the bottom row, the noise is increased to* $\sigma_{\{l,i,o\}} = 5$*, corresponding to the waveforms in Figure 3. The different waveforms are displayed as follows: sine and -sine as a magenta circle and a green square, cosine and -cosine as a pink pentagon and a yellow star, sawtooth and -sawtooth as a light-blue hexagon and an orange diamond and square and -square as a blue and a red triangle respectively.*

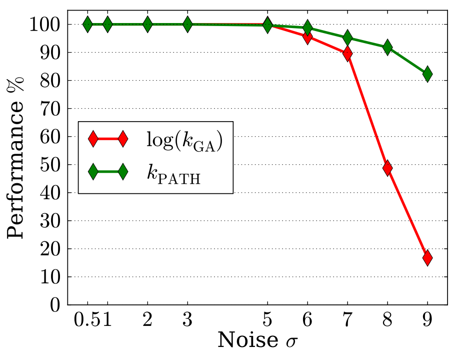


Figure 5: *We display the classification rate for predicting the waveform type using an SVM classifier in the feature space defined by the global alignment kernel (blue) and the path kernel (green). The x-axis depicts the noise level parametrized by* $\sigma_{\{l,i,o\}}$.

50 runs. Due to the negative eigenvalues, the classification fails for the DTW kernel. For this reason, we only present results for the remaining two kernels. In Figure 5, the results for the classification with increasing noise levels are shown. For moderate noise-levels (up to $\sigma_{\{l,i,o\}} = 5$), the global alignment and the path kernel are comparable in performance, while – at a higher noise level – the performance of the global alignment rapidly deteriorates and at $\sigma_{\{l,i,o\}} = 9$ its performance is about chance, while the path kernel still achieves a classification rate of over 80%.

The classification experiment shows that the path kernel significantly outperformed the global alignment kernel when noise in the sequence became significant. Looking at the feature space, depicted in Figure 4, we see that the path kernel encodes a feature space having more clearly defined clusters corresponding to the different waveforms. Additionally, the clusters also have a simpler structure. This signifies that the path kernel should be better suited for generalisation purposes, where it is beneficial to have a large continuous region of support which gracefully describes the variations in the data – rather than working in a space that barely separates the classes. We now generate a new dataset consisting of 100 noisy sine-waves ($\sigma_{\{l,i,o\}} = 5$) shifted in phase between 0 and π. The data is split uniformly into two halves and the first is used for training and the second for testing. We want to evaluate how well the kernel is capable of generalising over the training data. To that end we regress from the proportion of the training data to the test data and evaluate how the prediction error changes by altering this proportion. The prediction is performed using simple least-square regression over the kernel induced feature space. In Figure 6, the results are shown using different sizes of the training data. As shown, the path kernel performs significantly better compared to the global alignment kernel and the results improve with the size of the train-

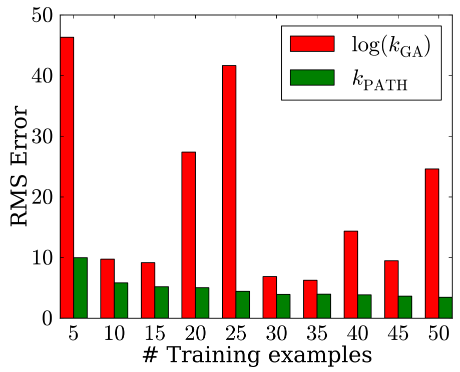


Figure 6: *The above figure depicts the RMS error when predicting the phase shift from a noisy sine waveform by a regression over the feature space induced by the kernels. The red bars correspond to the global alignment kernel and the green bars to the path kernel. The y-axis shows the error in percentage of phase, while the x-axis indicates the size of the training dataset.*

ing dataset. Interestingly, the global alignment kernel produces very different results dependent on the size of the training dataset indicating that it is severely over-fitting the data.

5 CONCLUSIONS

In this paper, we have presented a novel kernel for encoding sequences. Our kernel reflects and encodes all possible alignments between two sequences by associating a cost to each. This cost encodes a preference towards specific paths. The kernel is applicable to any kind of symbolic or numerical data as it requires only the existence of a kernel between symbols. We have presented both qualitative and quantitative experiments exemplifying the benefits of the path kernel compared to competing methods. We show that the proposed method significantly improves results both with respect to discrimination and generalisation especially in noisy scenarios. The computational cost associated with the kernel is considerably lower than competing methods, making it applicable to data-sets that could previously not be investigated using kernels.

Our experimental results indicate that the kernel we propose is positive semi-definite. In future we intend to investigate proving this property. Furthermore, in this paper, we have chosen a very simplistic dataset in order to evaluate our kernel. Given our encouraging results, we are currently working on applying our kernel to more challenging real-world datasets.

REFERENCES

Bahlmann, C., Haasdonk, B., and Burkhardt, H. (2002). Online handwriting recognition with support vector machines - a kernel approach. In *8th International Workshop on Frontiers in Handwriting Recognition*.

Berlinet, A. and Thomas-Agnan, C. (2004). *Reproducing kernel Hilbert spaces in probability and*.

Buhmann, M. D. and Martin, D. (2003). *Radial basis functions: theory and implementations*.

Chang, C. C. and Lin, C. J. (2001). LIBSVM: a library for support vector machines.

Cristianini, N. and Shawe-Taylor, J. (2006). *An introduction to support Vector Machines: and other kernel-based learning methods*.

Cuturi, M. (2010). Fast Global Alignment Kernels. In *International Conference on Machine Learning*.

Cuturi, M., Vert, J.-P., Birkenes, O., and Matsui, T. (2007). A Kernel for Time Series Based on Global Alighments. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 413–416.

Gudmundsson, S., Runarsson, T. P., and Sigurdsson, S. (2008). Support vector machines and dynamic time warping for time series. *IEEE International Joint Conference on Neural Networks*, pages 2772–2776.

Haasdonk, B. (2005). Feature space interpretation of SVMs with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–492.

Haasdonk, B. and Burkhardt, H. (2007). Invariant kernel functions for pattern analysis and machine learning. *Machine learning*, 68(1):35–61.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical report.

Leslie, C. and Kuang, R. (2004). Fast String Kernels using Inexact Matching for Protein Sequences. *The Journal of Machine Learning Research*, 5:1435–1455.

Li, M. and Zhu, Y. (2006). Image classification via LZ78 based string kernel: a comparative study. *Advances in knowledge discovery and data mining*.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444.

Luo, G., Bergström, N., Ek, C. H., and Kragic, D. (2011). Representing actions with Kernels. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2028–2035.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49.

Saunders, C., Tschach, H., and Shawe-Taylor, J. (2002). Syllables and other string kernel extensions. *Proceedings of the Nineteenth International Conference on Machine Learning (ICML'02)*.

Sivic, J. and Zisserman, A. (2009). Efficient Visual Search of Videos Cast as Text Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):591–606.

Watkins, C. (1999). Dynamic alignment kernels. In *Advances in Neural Information Processing Systems*.