Intelligent Data Analysis 12 (2008) 425–440
IOS Press

Using string kernels to identify famous performers from their playing style[1]

Craig Saunders[a], David R. Hardoon[b,*], John Shawe-Taylor[b] and Gerhard Widmer[c]
[a]*School of Electronics & Computer Science, ISIS Research Group, University of Southampton, Southampton, UK*
[b]*The Centre for Computational Statistics and Machine Learning, Department of Computer Science, University College London, London, UK*
[c]*Department of Medical Cybernetics and Artificial Intelligence, Medical University of Vienna, and Austrian Research Institute for Artificial Intelligence, Vienna, Austria*

Abstract. In this paper we show a novel application of string kernels: that is to the problem of recognising famous pianists from their style of playing. The characteristics of performers playing the same piece are obtained from changes in beat-level tempo and beat-level loudness, which over the time of the piece form a *performance worm*. From such worms, general performance alphabets can be derived, and pianists' performances can then be represented as strings. We show that when using the string kernel on this data, both kernel partial least squares and Support Vector Machines outperform the current best results. Furthermore we suggest a new method of obtaining feature directions from the Kernel Partial Least Squares algorithm and show that this can deliver better performance than methods previously used in the literature when used in conjunction with a Support Vector Machine.

Keywords: String kernel, partial least squares, support vector machine, music

1. Introduction

This paper focuses on the challenging problem of identifying famous pianists given a set of performances of the same piece using only minimal information obtained from audio recordings of their playing. Previous attempt at this challenge had been done by [4] where a system that generates expressive renditions using hierarchical hidden Markov models was trained on the stylistic variations employed by human performers was suggested. [4], while not concerned with artist classification, shows an example where their graphical model recognises one particular pianist (Cortot) among a large number of professional and non-professional pianists. Related work, such as [11] attempts at recognising student pianists using different performance features.

A technique called the performance worm plots a real-time trajectory over 2D space is used to analyse changes in tempo and loudness at the beat level, and extract features for learning. Previous work [14] on this data has compared a variety of machine learning techniques whilst using as features statistical

[1]This paper extends the work presented in a paper with a similar title that appeared at ECML 2004.
*Corresponding author. Tel: +44 20 7679 0425; Fax: +44 20 7387 1397; E-mail: D.Hardoon@cs.ucl.ac.uk.

quantities obtained from the performance worm. [14] addressed whether it is possible for a machine to learn to distinguish famous pianists using the features of the performance worm.

Using the performance worm it is possible however to obtain a set of cluster prototypes from the worm trajectory which capture certain characteristics over a small time frame, say of two beats. These cluster prototypes form a 'performance alphabet' and there is evidence that they capture some aspects of individual playing style. For example a performer may consistently produce loudness/tempo changes unique to themselves at specific points in a piece, e.g. at the loudest sections of a piece. Once a performance alphabet is obtained, the prototypes can each be assigned a symbol and the audio recordings can then be represented as strings constructed from this alphabet. This leads to a novel application of the string kernel, working on character-based representations of the performances.

The ability of the string kernel to include non-contiguous features is shown to be key in the performance of the algorithm. This is in contrast to results presented which used the string kernel on text, where the ability to handle non-contiguous substrings does not deliver significant performance over contiguous substrings [6]. Furthermore we examine the ability of dimension-reduction based methods such as Kernel Partial Least Squares (KPLS) [8] and Kernel PCA to extract significant directions in the feature space in order to possibly gain performance benefits. One issue with dimensionality reduction methods such as the above, however, is that it introduces yet another parameter (the number of feature directions to extract). We therefore employ a semi-definite programming solution defined in [5] which optimises over a combination of orthogonal kernel matrices, and thus removes the need to select a parameter. This is then compared to the more traditional cross-validation approach.

Our current work extends an earlier study with a similar title [9]. This paper addresses two limitations of the approach presented in the prequel, namely we addressed the issue of the KPLS scaling parameter and also employ a semi-definite programming technique which removes the need to set the parameter T which controls the number of feature directions chosen. In addition to this, we also consider a more thorough experimental set-up, which allows us to compare any significant differences in the approaches. Note that although we focus here on a specific, and difficult, application, the techniques developed are appropriate for a range of application domains. The combination of kernel PLS-style deflation to extract relevant features from structured kernels (where the mapping dimension is very high) in conjunction with an SVM we forsee to be a key combination.

The rest of this paper is laid out as follows. In the following section we provide background details on the performance worm representation used for the music data. Section 3 outlines the Partial Least Squares (PLS) algorithm and string kernel function used to analyse the data. Section 4 then presents the Kernel variant of PLS algorithm and gives a formulation for extracting features which are then used in conjunction with support vector machines (SVMs). We then present experimental results in Section 5 and end with some analysis and suggestions for future research.

2. A musical representation

The data used in this paper, first described in [17], was obtained from recordings of sonatas by W.A. Mozart played by six famous concert pianists. In total the performances of 6 pianists were analysed across 12 different movements of Mozart sonatas. The movements represent a cross section of playing keys, tempi and time signatures, see Table 1 for details.

In many cases the only data available for different performances are standard audio recordings (as opposed to for example MIDI format data from which more detailed analysis is possible), which poses

Table 1
Movements of Mozart piano sonatas selected for analysis

Sonata	Movement	Key	Time sig.
K.279	1st mvt.	C major	4/4
K.279	2nd mvt.	C major	3/4
K.279	3rd mvt.	C major	2/4
K.280	1st mvt.	F major	3/4
K.280	2nd mvt.	F major	6/8
K.280	3rd mvt.	F major	3/8
K.281	1st mvt.	Bb major	2/4
K.282	1st mvt.	Eb major	4/4
K.282	2nd mvt.	Eb major	3/4
K.282	3rd mvt.	Eb major	2/4
K.330	3rd mvt.	C major	2/4
K.332	2nd mvt.	F major	4/4

particular difficulties for the extraction of relevant performance information. A tool for analysing this type of data called the performance worm has recently been developed [3,13,17].

The performance worm extracts data from audio recordings by examining tempo and general loudness of the audio when measured at the beat level. An interactive beat tracking program [2] is used to find the beat from which changes in beat-level tempo and beat-level loudness can be calculated. These two types of changes can be integrated to form trajectories over tempo-loudness space that show the joint development of tempo and dynamics over time. As data is extracted from the audio the 2D plot of the performance curve can be constructed in real time to aid in visualisation of these dynamics, and this is called the performance worm. Figure 1 shows a screen-shot of the worm in progress.

Note that this is the only information used in the creation of the worm, more detailed information such as articulation, individual voicing or timing details below the level of a beat is not available.

2.1. A performance alphabet

From the performance worm, patterns can be observed which can help characterise the individual playing styles of some pianists. For example, in [13] a set of tempo-loudness shapes typical of the performer Mitsuko Uchida were found. These shapes represented a particular way of combining a crescendo-decrescendo with a slowing down during a loudness maximum. These patterns were often repeated in Mozart performances by Mitsuko Uchida, but were rarely found when analysing the recordings of other performers.

In order to try and capture more of these types of characterisations a ‘Mozart Performance Alphabet’ can be constructed in the following way. The trajectories of the performance worm are cut into short segments of a fixed length (e.g. 2 beats) and clustered into groups of similar patterns to form a series of prototypes (see Fig. 2). Recordings of a performance can then be transcribed in terms of this alphabet which can then be compared using string matching techniques. The list of pianists and the recordings used to obtain the data can be found in Table 2. For more detailed information on the performance worm and constructing a performance alphabet of cluster prototypes, please refer to [13,17].

Note that the only input to our algorithm will be the string representation of the piece, which is simply a sequence of curves from the performance alphabet. We are therefore using very little information (only loudness/temp at the beat level), which is known to be noisy (there may be some error in the clustering process, and the string of cluster prototypes certainly does not reproduce the original worm).

Table 2
List of pianists and recordings used

ID	Name	Recording
DB	Daniel Barenboim	EMI Classics CDZ 7 67295 2, 1984
RB	Roland Batik	Gramola 98701-705, 1990
GG	Glenn Gould	Sony Classical SM4K 52627, 1967
MP	Maria João Pires	DGG 431 761-2, 1991
AS	András Schiff	ADD (Decca) 443 720-2, 1980
MU	Mitsuko Uchida	Philips Classics 464 856-2, 1987

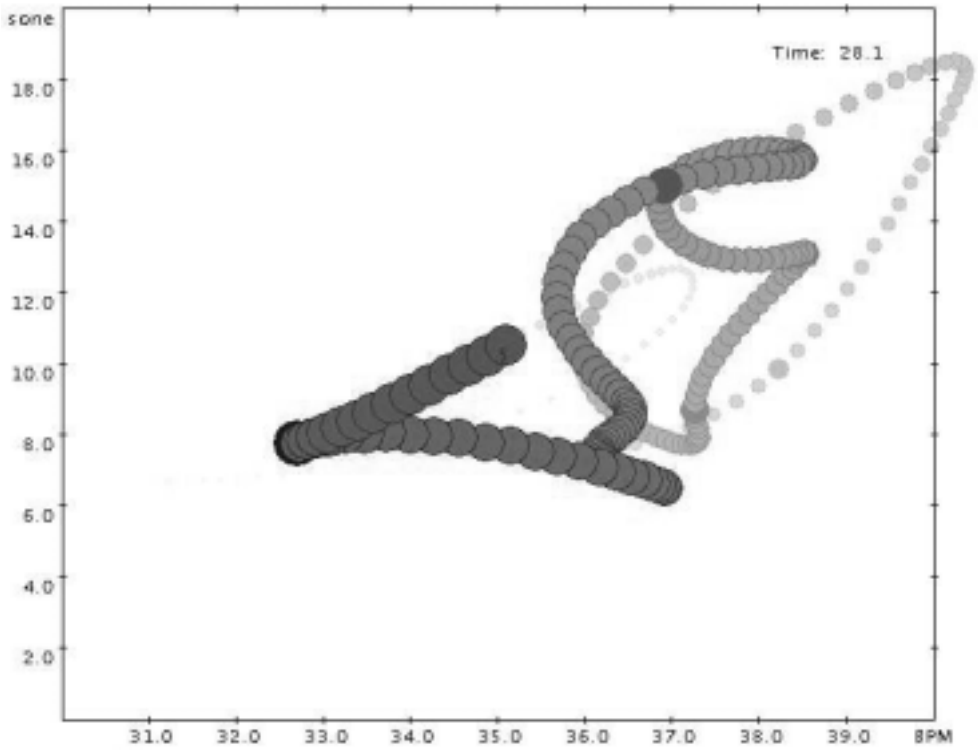

Fig. 1. The performance worm: A 2D representation of changes in beat-level tempo and loudness can be plotted in realtime from an audio recording.

The task addressed in this paper is to learn to recognise pianists solely from characteristics of their performance strings. The ability of kernel methods to operate over string-like structures using kernels such as the n-gram kernel and the string kernel will be evaluated on this task. In addition to simply applying an SVM to the data however, we will also examine the ability of dimension reduction methods such as Kernel PCA and Kernel Partial Least Squares (KPLS) to extract relevant features from the data before applying an SVM, which will hopefully lead to improved classification performance. KPCA is well known method and has often been used to extract features from data (see e.g. [10]). Partial least squares and its kernel-based variant KPLS has recently gained popularity within the machine learning community [1,7,8] and either can be used as a method for regression or classification, or as a method for dimension reduction. It is not always clear however, how to use the PLS-based methods to generate new input features for training and test data, so we shall briefly review the methods here.

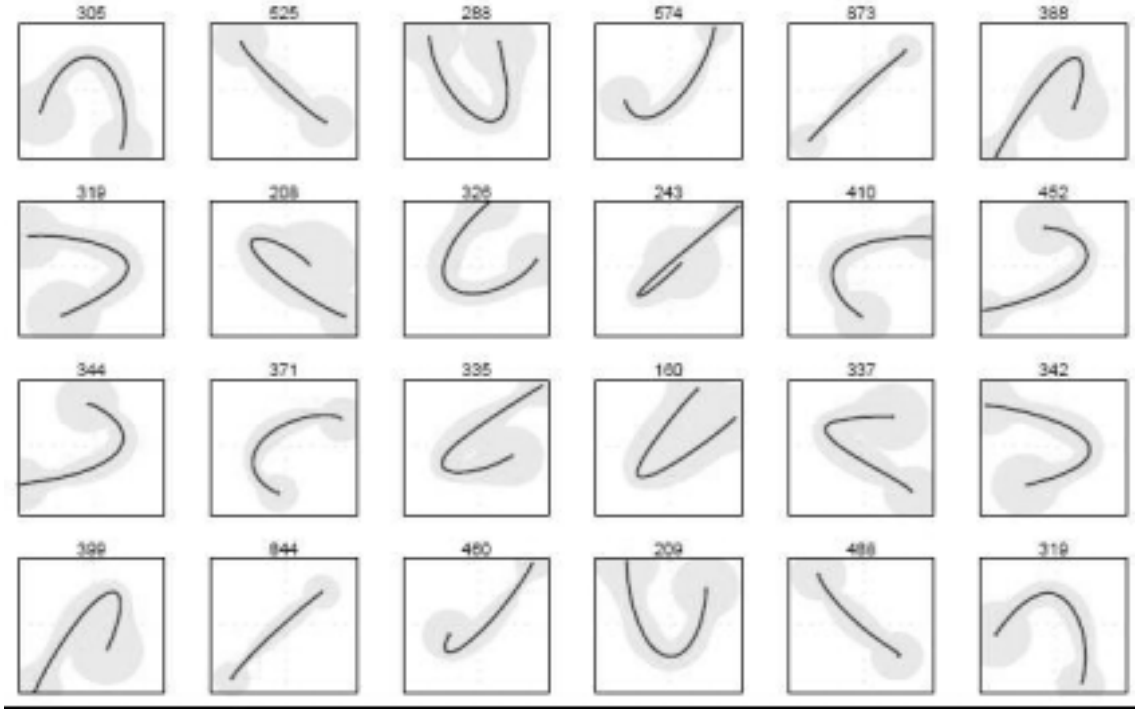

Fig. 2. The performance alphabet: A set of cluster prototypes extracted from the performance worm.

3. Previous results

3.1. String kernels

The use of string kernels for analysing text documents was first studied by Lodhi et al. [6]. We briefly review the approach to creating a feature space and associated kernel.

The key idea behind the gap-weighted sub-sequences kernel is to compare strings by means of the sub-sequences they contain. The more sub-sequences in common, the more similar they are. Rather than only considering contiguous n-grams, the degree of contiguity of the sub-sequence in the input string s determines how much it will contribute to the comparison.

In order to deal with non-contiguous substrings, it is necessary to introduce a decay factor $\lambda \in (0, 1)$ that can be used to weight the presence of a certain feature in a string. For an index sequence $\mathbf{i} = (i_1, \ldots, i_k)$ identifying the occurrence of a sub-sequence $u = s(\mathbf{i})$ in a string s, we use $l(\mathbf{i}) = i_k - i_1 + 1$ to denote the length of the string in s. In the gap-weighted kernel, we weight the occurrence of u with the exponentially decaying weight $\lambda^{l(\mathbf{i})}$.

Definition 1 Gap-weighted sub-sequences kernel. *The feature space associated with the gap-weighted sub-sequences kernel of length p is indexed by $I = \Sigma^p$ (i.e. sub-sequences of length p from some alphabet Σ), with the embedding given by*

$$\phi_u^p(s) = \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{l(\mathbf{i})}, \mathrm{u} \in \Sigma^{\mathrm{p}}. \tag{1}$$

Table 3
Features and weights for the string kernel with $p = 2$ for the words ''cat'', ''car'', ''bat'', and ''bar''

ϕ	ca	ct	at	ba	bt	cr	ar	br
cat	λ^2	λ^3	λ^2	0	0	0	0	0
car	λ^2	0	0	0	0	λ^3	λ^2	0
bat	0	0	λ^2	λ^2	λ^3	0	0	0
bar	0	0	0	λ^2	0	0	λ^2	λ^3

The associated kernel is defined as

$$\kappa_p(s,t) = \langle \phi^p(s), \phi^p(t) \rangle = \sum_{u \in \Sigma^p} \phi_u^p(s)\, \phi_u^p(t). \tag{2}$$

Consider the simple strings ''cat'', ''car'', ''bat'', and ''bar''. Fixing $p = 2$, the words are mapped as shown in Table 3.

So the unnormalised kernel between ''cat'' and ''car'' is $\kappa(\text{''cat''},\text{''car''}) = \lambda^4$, while the normalised version is obtained using

$$\kappa(\text{''cat''},\text{''cat''}) = \kappa(\text{''car''},\text{''car''}) = 2\lambda^4 + \lambda^6 \tag{3}$$

as $\hat{\kappa}(\text{''cat''},\text{''car''}) = \lambda^4/(2\lambda^4 + \lambda^6) = (2 + \lambda^2)^{-1}$.

We omit a description of the efficient dynamic programming algorithms for computing this kernel referring the reader to Lodhi et al. [6].

3.2. Partial Least Squares

Partial Least Squares (PLS) was developed by Herman Wold during the 1960's in the field of econometrics [16]. It offers an effective approach to solving problems with training data that has few points but high dimensionality, by first projecting the data into a lower-dimensional space and then utilising a Least Squares (LS) regression model. This problem is common in the field of Chemometrics where PLS is regularly used. PLS is a flexible algorithm that was designed for regression problems, though it can be used for classification by treating the labels $\{+1, -1\}$ as real outputs. Alternatively it can also be stopped after constructing the low-dimensional projection. The resulting features can then be used in a different classification or regression algorithm. We will also adopt this approach by applying an SVM in this feature space, an approach pioneered by Rosipal et al. [8]. Let $\mathbf{X}$ contain as rows the feature vectors of the samples and $\mathbf{y}$ contain the outputs, where $\mathbf{Y}$ is the matrix containing as rows a set of output variables.

The procedure for PLS feature extraction is shown in Algorithm 1. The algorithmic procedure iteratively takes the first singular vector $\mathbf{u}_i$ of the matrix $\mathbf{X}_i'\mathbf{Y}$, and then deflates the matrix $\mathbf{X}_i$ to obtain $\mathbf{X}_{i+1}$, where $i = 1 \ldots T$. The deflation is done by projecting the columns of $\mathbf{X}_i$ into the space orthogonal to $\mathbf{X}_i\mathbf{u}_i$. The difficulty with this simple description is that the feature directions $\mathbf{u}_j$ are defined relative to the deflated matrix. We would like to be able to compute the PLS features directly from the original feature vector.

If we now consider a test point with feature vector $\phi(\mathbf{x})$ the transformations that we perform at each step should also be applied to $\phi_1(\mathbf{x}) = \phi(\mathbf{x})$ to create a series of feature vectors

$$\phi_{i+1}(\mathbf{x})' = \phi_i(\mathbf{x})' \left(\mathbf{I} - \mathbf{u}_i \mathbf{p}_i'\right), \tag{4}$$

Algorithm 1. The PLS feature extraction algorithm

The PLS feature extraction algorithm is as follows:

input	given a selection of T feature directions to extract
process	$\mathbf{X}_1 = \mathbf{X}$
	for $i = 1, \ldots, T$
	let $\mathbf{u}_i$ be the first singular vector of $\mathbf{X}_i'\mathbf{Y}$,
	$\mathbf{X}_{i+1} = \left(\mathbf{I} - \frac{\mathbf{X}_i\mathbf{u}_i\mathbf{u}_i'\mathbf{X}_i'}{\mathbf{u}_i'\mathbf{X}_i'\mathbf{X}_i\mathbf{u}_i}\right)\mathbf{X}_i = \mathbf{X}_i\left(\mathbf{I} - \frac{\mathbf{u}_i\mathbf{u}_i'\mathbf{X}_i'\mathbf{X}_i}{\mathbf{u}_i'\mathbf{X}_i'\mathbf{X}_i\mathbf{u}_i}\right)$
	end
output	Feature directions $\mathbf{u}_i$, $i = 1, \ldots, T$.

where

$$\mathbf{p}_i = \frac{\mathbf{X}_i'\mathbf{X}_i\mathbf{u}_i}{\mathbf{u}_i'\mathbf{X}_i'\mathbf{X}_i\mathbf{u}_i}, \tag{5}$$

This is the same operation that is performed on the rows of $\mathbf{X}_i$ in Algorithm 1. We can now write

$$\phi(\mathbf{x})' = \phi_{T+1}(\mathbf{x})' + \sum_{i=1}^{T} \phi_i(\mathbf{x})'\mathbf{u}_i\mathbf{p}_i'. \tag{6}$$

The feature vector that we need for the regression $\hat{\phi}(\mathbf{x})$ has components

$$\hat{\phi}(\mathbf{x}) = \left(\phi_i(\mathbf{x})'\mathbf{u}_i\right)_{i=1}^{T}, \tag{7}$$

since these are the projections of the residual vector at stage i onto the next feature vector $\mathbf{u}_i$. Rather than compute $\phi_i(\mathbf{x})'$ iteratively, consider using the inner products between the original $\phi(\mathbf{x})'$ and the feature vectors $\mathbf{u}_i$ stored as the columns of the matrix $\mathbf{U}$:

$$\begin{aligned}\phi(\mathbf{x})'\mathbf{U} &= \phi_{T+1}(\mathbf{x})'\mathbf{U} + \sum_{i=1}^{k} \phi_i(\mathbf{x})'\mathbf{u}_i\mathbf{p}_i'\mathbf{U} \\ &= \phi_{T+1}(\mathbf{x})'\mathbf{U} + \hat{\phi}(\mathbf{x})'\mathbf{P}'\mathbf{U},\end{aligned}$$

where $\mathbf{P}$ is the matrix whose columns are $\mathbf{p}_i$. Finally, it can be verified that

$$\mathbf{u}_j'\mathbf{p}_i = \delta_{ji} \quad \text{for } j \leqslant i. \tag{8}$$

Hence, for $s > i$, $(\mathbf{I} - \mathbf{u}_s\mathbf{p}_s')\mathbf{u}_i = \mathbf{u}_i$, while $(\mathbf{I} - \mathbf{u}_i\mathbf{p}_i')\mathbf{u}_i = 0$, so we can write

$$\phi_{T+1}(\mathbf{x})'\mathbf{u}_i = \phi_i(\mathbf{x})'\prod_{j=i}^{T}\left(\mathbf{I} - \mathbf{u}_j\mathbf{p}_j'\right)\mathbf{u}_i = 0, \text{ for } i = 1, \ldots, T. \tag{9}$$

It follows that the new feature vector can be expressed as

$$\hat{\phi}(\mathbf{x})' = \phi(\mathbf{x})'\mathbf{U}\left(\mathbf{P}'\mathbf{U}\right)^{-1}. \tag{10}$$

If we consider the vectors of feature values across the training set $\mathbf{X}_j\mathbf{u}_j$, these are orthogonal since they are a linear combination of the columns of $\mathbf{X}_j$ that have been repeatedly projected into the orthogonal

complement of previous $\mathbf{X}_i\mathbf{u}_i$, for $i < j$. By the analysis above these vectors can be written as $\mathbf{XU}(\mathbf{P}'\mathbf{U})^{-\mathbf{1}}$, from which it follows that

$$(\mathbf{U}'\mathbf{P})^{-\mathbf{1}}\mathbf{U}'\mathbf{X}'\mathbf{X}\mathbf{U}(\mathbf{P}'\mathbf{U})^{-\mathbf{1}}$$

is a diagonal matrix. Since $(\mathbf{U}'\mathbf{P})$ is upper triangular it follows that the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ are conjugate with respect to $\mathbf{X}'\mathbf{X}$.

These feature vectors can now be used in conjunction with a learning algorithm. If one wishes to calculate the overall regression coefficients as in the full PLS algorithm, these can be computed as:

$$\mathbf{W} = \mathbf{U}\left(\mathbf{P}'\mathbf{U}\right)^{-1}\mathbf{C}', \tag{11}$$

where $\mathbf{C}$ is the matrix with columns

$$\mathbf{c}_i = \frac{\mathbf{Y}'\mathbf{X}_i\mathbf{u}_i}{\mathbf{u}_i'\mathbf{X}_i'\mathbf{X}_i\mathbf{u}_i}. \tag{12}$$

4. Kernel PLS

In this section we set out the kernel PLS algorithm and describe its feature extraction stage. The kernel PLS algorithm is given in Algorithm 2.

Algorithm 2. Pseudocode for kernel-PLS

Input: Data $S = x_1, \ldots, x_l$; dimension T; target outputs $Y \in \mathbb{R}^{l\times m}$

$K_{ij} = \kappa(x_i, x_j)$
$K_1 = K$
$\hat{\mathbf{Y}} = \mathbf{Y}$

for $i = 1, \ldots, T$ **do**
  $\beta_i =$ first column of $\hat{\mathbf{Y}}$
  normalise β_i
  repeat
    $\beta_i = \mathbf{Y}\mathbf{Y}'K_i\beta_i$
    normalise β_i
  until convergence
  $\tau_i = K_i\beta_i$
  $c_i = \hat{\mathbf{Y}}'\tau_i/||\tau_i||^2$
  $\hat{\mathbf{Y}} = \hat{\mathbf{Y}} - \tau_i c_i'$
  $K_{i+1} = \left(I - \tau_i\tau_i'/||\tau_i||^2\right)K_i\left(I - \tau_i\tau_i'/||\tau_i||^2\right)$
end for

$B = [\beta_i, \ldots, \beta_k]$
$\mathbf{T} = [\tau_i, \ldots, \tau_k]$
$\alpha = B(\mathbf{T}KB)^{-1}\mathbf{T}'\mathbf{Y}$

Output: Training outputs $Y - \hat{Y}$ and dual regression coefficients α

The vector β_i is a rescaled dual representation of the primal vectors $\mathbf{u}_i$:

$$a_i\mathbf{u}_i = \mathbf{X}_i'\beta_i, \tag{13}$$

the re-scaling arising because of the different point at which the re-normalising is performed in the dual. We can now express the primal matrix $\mathbf{P}'\mathbf{U}$ in terms of the dual variables as

$$\begin{aligned}\mathbf{P}'\mathbf{U} &= \text{diag}\,(\mathbf{a})\,\text{diag}\left(\tau_i'\tau_i\right)^{-1}\mathbf{T}'\mathbf{X}\mathbf{X}'\mathbf{B}\,\text{diag}\,(\mathbf{a})^{-1}\\ &= \text{diag}\,(\mathbf{a})\,\text{diag}\left(\tau_i'\tau_i\right)^{-1}\mathbf{T}'\mathbf{K}\mathbf{B}\,\text{diag}\,(\mathbf{a})^{-1}.\end{aligned}$$

Here $\text{diag}\,(\tau_i'\tau_i)$ is the diagonal matrix with entries $\text{diag}\,(\tau_i'\tau_i)_{ii} = \tau_i'\tau_i$, where $\tau_i = K_i\beta_i$ and $\mathbf{T}$ is the matrix containing τ_i as rows . Finally, again using the orthogonality of $\mathbf{X}_j\mathbf{u}_j$ to τ_i, for $i < j$, we obtain

$$\mathbf{c}_j = \frac{\mathbf{Y}_j'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j} = \frac{\mathbf{Y}'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j} = a_j\frac{\mathbf{Y}'\tau_j}{\tau_j'\tau_j}, \tag{14}$$

making

$$\mathbf{C} = \mathbf{Y}'\mathbf{T}\,\text{diag}\left(\tau_i'\tau_i\right)^{-1}\text{diag}\,(\mathbf{a}). \tag{15}$$

Putting the pieces together we can compute the dual regression variables as

$$\alpha = \mathbf{B}\left(\mathbf{T}'\mathbf{K}\mathbf{B}\right)^{-1}\mathbf{T}'\mathbf{Y}. \tag{16}$$

In [8] it was assumed that a dual representation of the PLS features is then given by

$$\mathbf{B}\left(\mathbf{T}'\mathbf{K}\mathbf{B}\right)^{-1}, \tag{17}$$

but in fact

$$\mathbf{U}\left(\mathbf{P}'\mathbf{U}\right)^{-1} = \mathbf{X}'\mathbf{B}\,\text{diag}\,(\mathbf{a})^{-1}\left(\text{diag}\,(\mathbf{a})\,\text{diag}\left(\tau_i'\tau_i\right)^{-1}\mathbf{T}'\mathbf{K}\mathbf{B}\,\text{diag}\,(\mathbf{a})^{-1}\right)^{-1}$$

so that the dual representation is

$$\mathbf{B}\left(\mathbf{T}'\mathbf{K}\mathbf{B}\right)^{-1}\text{diag}\,(\mathbf{a})^{-1}\,\text{diag}\left(\tau_i'\tau_i\right) = \mathbf{B}\left(\mathbf{T}'\mathbf{K}\mathbf{B}\right)^{-1}\text{diag}\left(\tau_i'\tau_i\right)\text{diag}\,(\mathbf{a})^{-1}. \tag{18}$$

The missing diagonal matrices perform a re-scaling of the features extracted, which skews the geometry of the space and affects the performance of for example an SVM.

At first sight it seems as though the a_i are difficult to assess, and there is an argument that the values of a_i should not vary significantly over similar adjacent features since they will be related to the corresponding singular values. Recalling Eq. (13) however, we have the following

$$(a_i\mathbf{u}_i)^2 = (\mathbf{X}_i'\beta_i)^2 = \beta_i'\mathbf{K}\beta_i = a_i^2.$$

In our experiments we have compared the results that can be obtained ignoring both diagonal matrices with those obtained including the tau rescaling $\text{diag}\,(\tau_i'\tau_i)$ and the complete rescaling $\text{diag}\,(\tau_i'\tau_i)\,\text{diag}\,(\mathbf{a})^{-1}$.

4.1. Using semi-definite programming

One disadvantage of using these subspace approaches is that we have introduced yet another parameter: the number of feature directions to use T. Also, we are forced into using just one particular value for T, rather than exploring combinations of directions which may yield better results (in practice of course we could try combinations, but this starts to explode the size of our parameter space even more).

An ideal scenario would be to generate several rank-one kernel matrices from the projections on to the directions $\mathbf{u}_1, \ldots, \mathbf{u}_T$, and then optimise over the combination of these matrices in one step. This would both remove the need to choose a parameter T (apart from setting a limit for T in advance) and also allow combinations of features to be used. An approach outlined in [5] shows that it is possible to use semi-definite programming techniques to optimise over a combination over kernel matrices whilst simultaneously solving the SVM maximisation problem. In this section we briefly review the method in [5] and discuss how it can be applied to the problem of selecting the number of feature directions to use.

In the paper it was shown that a if $\hat{K}$ is a linear combination of kernel matrices $\hat{K} = \sum_{i=1}^{k} \mu_i K_i$ then it is possible to optimise the standard SVM formulation and the coefficients μ_i simultaneously using a semi-definite program. This is due to the problem being convex in $\hat{K}$. Furthermore, a generalisation bound is given which shows that the generalisation error, at least in part, depends on the trace of the matrix $\hat{K}$; for further details we refer the reader to [5]. In essence however, the paper considers several settings of optimising combinations of kernel matrices, whilst solving the standard hard and soft margin SVM optimisation problems. One special instance that is considered is when the μ_i components of the combination are non-negative. In this case, the problem reduces to a Quadractically Constrained Quadratic Problem (QOQP), which is a special case of a Second Order Cone Pogramming problem (SOCP), which in turn is a special instance of a semi-definite programming problem (SDP). For the application considered here, this has two important consequences. Firstly, SOCP problems can be efficiently solved by publicly available solvers such as SeDuMi [12]. Secondly, in the general case, the restriction of $\mu_i \geqslant 0$ may lead to sub-optimal combinations of the base kernel matrices, as allowing μ to have negative coefficients may still lead to a positive semi-definite matrix $\hat{K}$. Due to the orthogonality of $\mathbf{v}_i$ (as discussed in first part of Section 4), however, negative components μ_i would lead to negative eigenvalues; as eigenvalues of $\hat{K}$ are equal to $\mu_i ||\mathbf{v}_i||$ and therefore $\hat{K}$ would not be positive definite.

In summary, the optimisation problem becomes

$$\min_{\hat{K}} \max_{\boldsymbol{\alpha}} 2\boldsymbol{\alpha}'\mathbf{e} - \boldsymbol{\alpha}'(G(\hat{K}) + \tau I)\boldsymbol{\alpha}$$

subject to

$$0 \leqslant \boldsymbol{\alpha} \leqslant C, \boldsymbol{\alpha}'\mathbf{y}, \tau > 0, \text{trace}(\hat{K}) = c,$$

where $G_{ij}(K) = \kappa(\mathbf{x}_i, \mathbf{x}_j) y_i y_j$. For the case where we restrict $\mu_i \geqslant 0$ and consider the 1-norm SVM margin, the optimisation problem becomes:

$$\begin{aligned} \max_{\boldsymbol{\alpha},t} \quad & 2\boldsymbol{\alpha}'\mathbf{e} - ct \qquad (19) \\ \text{subject to } t \geqslant \tfrac{1}{r_i}\boldsymbol{\alpha}' G(K_i)\boldsymbol{\alpha}, \; & i = 1, \ldots, k, \\ & \boldsymbol{\alpha}'\mathbf{y} = 0 \\ & 0 \leqslant \boldsymbol{\alpha} \leqslant C \end{aligned}$$

where $r_i = \text{trace}(K_i)$.

In this paper we are considering the problem of choosing the number of feature projections to use when considering low-dimensional projections of feature-space data. Therefore, the kernel matrices K_i are rank one matrices formed by the outer product of feature projections $\mathbf{v}_i \mathbf{v}_i'$, where $\mathbf{v}_i$ is the dual

representation of the data projected onto the ith extracted feature (e.g. the columns of the matrix given by Eq. (18)). Note that most general SDP solvers use primal dual methods to solve problems such as Eq. (19). The primal variables μ_i are therefore easily covered once a solution is found.

5. Experiments

In our experiments we follow the setup given by [14,17] where the original n-class problem (where n is the number of pianists) was converted to $n(n-1)/2$ two-class discrimination problem, one for each possible pair of pianists. This set-up gives more insight into the discriminability of various pianists, and is easier for a classifier than the n-class problem.

For each pair of performers a leave-one-out procedure was followed where on each iteration one movement played by each of a pair of performers was used for testing and the rest of the data was used for training. That is, for a given pair of performers, say Mitsuko Uchida and Daniel Barenboim (MU-DB), a total of 12 runs of an algorithm were performed (there are 12 movements and each time one movement by both performers was left out of the training set and tested upon). This was repeated for each of the possible 15 pairings of performers. Note that in all results the number reported is the number of *correct* classifications made by the algorithm.

5.1. Previous results

Previous results on the data (as described in [14,17]) used a feature-based representation and considered a range of machine learning techniques by using the well-known Waikato Environment for Knowledge Analysis (WEKA) software package [15] to compare bayesian, rule-based, tree-based and nearest-neighbour methods. The best results obtained previously on the data are for a classification via regression meta-learner. These results are reported as FB (feature-based) in the results table. The feature-based representation used in the experiments included the raw measures of tempo and loudness along with various statistics regarding the variance and standard deviation of these and additional information extracted from the worm such as the correlation of tempo and loudness values.

5.2. Results

Experiments were conducted using both the standard string kernel and the n-gram kernel and several algorithms. In both cases experiments were conducted using a standard SVM on the relevant kernel matrix. Kernel Partial Least Squares and Kernel Principal Component Regression were also used for comparison. Finally, an SVM was used in conjunction with the projected features obtained from the iterative KPLS deflation steps. For these features there were three options; to use the features as described in [8], to include the extra reweighting factors diag $(\tau_i'\tau_i)$ or to also include the scaling produced by the a_i values as described above[2]. We first performed a comparison of these two options by counting the total number of correct predictions across all splits for different feature dimensions (T) for the original weighting (ORIG), the tau-reweighted (τ-R), and the features with both tau and a rescaling (τa-R). Table 4 shows the results obtained. There is a clear advantage shown for the re-weighting schemes,

[2]We find that the re-scaling does not add to the total number of correct predictions and therefore only compare with the original two options.

Table 4
Total number of correct predictions across all splits against number of feature directions used (T) for both the feature projection methods described in this paper (τ-R) and (τa-R) and that in previous work (ORIG). The parameters used in this were those optimal for the KPLS combination ($k = 5$, $\lambda = 0.9$)

Method/T	1	2	3	4	5	6	7	8	9	10
ORIG	284	246	230	236	249	240	235	239	244	240
τ-R	288	289	290	290	290	290	290	290	290	290
τa-R	286	289	290	287	287	290	289	290	291	289

however the introduction of the diag $(\mathbf{a})^{-1}$ matrix seems to have little effect as expected. In this situation using the full reweighting actually suffered a slight performance loss against using τ only, however for completeness we will use the full reweighting in all future experiments. Note that this was an illustration to show the rescaling difference only, and results here cannot be directly compared to the tables below, where a more rigorous experimental set-up was used for comparison purposes.

In the remaining experiments we used the following set up. There are 15 pairs of performers, which were split up into three groups of five. We used one group as a hold-out set and used the leave-one-out error on the 24 movements for each pair (each time leaving out one of the 12 movements for both performers) as an accuracy measure. Using this measure we were able to select the parameters which performed best across this group. Using these parameters we then obtained an accuracy measure on the remaining 12 pairs of performers using the same leave-one-movement-out procedure as described above. We repeated this for each group, and therefore obtain four accuracy readings for each pair. The parameters optimised included the number of characters used by the string kernel, the decay parameter and the number of PLS features extracted where appropriate. Substring lengths of $k = 1, \ldots, 10$ were tried for both the n-gram and string kernels, $\lambda = \{0.2, 0.5, 0.9\}$ decay parameters were used for the string kernel and for both KPLS and KPCR methods the number of feature directions (T) ranged from 1 to 10. All kernel matrices were normalised and whenever an SVM was used, the parameter C was set to one.

Tables 5 and 6 show results using the n-gram and string kernels respectively. Mean and standard deviations of the accuracy (maximum of 24 in each case) for each pair are predicted. Significance results were obtained by using a one-tailed t-test. Note that for the feature-based results, the results reported are the best possible that can be achieved – parameters were selected according to the classification performance of the movement that was left out (see [11,14] for details). Therefore, we are competing against a gold standard, and it is likely higher performance for the methods in this paper could be achieved if we didn't follow the cross-validation procedure outlined above. In some cases, when optimising parameters using the hold-out group, several different choices lead to the same error value. In order to be consistent we simply chose the lowest substring length/λ-value/T-value that gave this result.

The results obtained from using these methods and kernels show an improvement over the previous best results using statistical features extracted from the performance worm. We use the following shorthand to refer to the relevant algorithm/kernel combinations; **FB**: Previous best method using statistical features [14], **KPLS**: Kernel Partial Least Squares, **SVM**: Support Vector Machine, **KP-SV**: SVM using KPLS features, **KPCR**: Kernel Principal Components regression. If an n-gram kernel is used rather than a string kernel we append '**-n**' to the method name. We report both the leave-one-movement-out accuracy for each pairing, and also statistics for the total accuracy for each fold of the data. Therefore the maximum total value is $12 \times 24 = 288$. For the feature based approach we can simulate this by simply summing accuracies over those test-pairings in each fold, and therefore obtain a value for each of the 5 runs.

Table 5
Comparison of algorithms across each pair-wise coupling of performers for the n-gram kernel. Mean and standard deviation rates for accuracy are given for each pair (the maximum is 24). FB represents the previous best results using a feature-based representation rather than the 'performance alphabet' used for the other approaches. *Indicates significant difference from KPCR at 95% significance level, **denotes same at 99% significance. Similarly, significance from the SVM is denoted by $^{+}$ and $^{++}$, and $^{\$}$ is used for comparison to the feature-based representation (fold totals only). Totals are for predicting remaining 12 performer pairs (see text)

Pairing	FB	KPLS-n	SVM-n	KP-SV-n	KPCR-n
RB – DB	19	17.8 (0.5)	17.5 (1.0)	17.3 (0.5)	16.5 (1.0)
GG – DB	18	21.3 (0.5)**	22.0 (0.0)**	21.0 (0.0)**	13.3 (0.5)
GG – RB	17	21.3 (1.5)*	21.5 (1.0) *	22.0 (0.0)**	12.0 (2.0)
MP – DB	15	20.5 (1.7)	19.0 (0.0)	18.3 (0.5)	15.8 (1.5)
MP – RB	20	16.5 (1.0)	15.0 (0.0)	18.8 (1.3)$^{*+}$	15.8 (0.5)
MP – GG	17	22.5 (0.6)**	22.0 (0.0)**	22.8 (0.5)**	13.0 (0.0)
AS – DB	16	19.8 (0.5)*	18.5 (1.0)*	19.5 (3.0)	13.5 (1.9)
AS – RB	17	20.0 (0.0)**	20.8 (0.5)**	19.5 (0.6)**	13.3 (0.5)
AS – GG	17	14.5 (1.0)	13.3 (0.5)	18.5 (0.6) $^{++}$	16.5 (3.3)
AS – MP	16	20.8 (0.5)*	19.8 (0.5)**	22.0 (0.0)$^{**+}$	13.5 (1.7)
MU – DB	15	15.0 (1.6)	17.0 (0.0)	18.3 (2.9)	13.3 (3.2)
MU – RB	17	16.5 (2.0)	14.3 (0.5)	16.8 (1.0)$^{*++}$	12.8 (1.3)
MU – GG	18	20.5 (1.0)	21.3 (1.5)*	21.3 (1.0)	17.0 (2.4)
MU – MP	13	13.3 (1.0)	12.8 (0.5)	16.5 (3.1)	15.0 (2.4)
MU – AS	16	17.5 (0.6)	17.3 (1.5)	18.3 (1.7)	16.0 (3.8)
Fold Total	200.8 (2.9)	222.0 (8.5)$^{\$*}$	217.4 (6.1)$^{\$**}$	232.4 (7.3)$^{\$**\$+}$	173.6 (13.3)

Table 6
Comparison of algorithms across each pairwise coupling of performers for the string kernel

Pairing	FB	KPLS	SVM	KP-SV	KPCR
RB – DB	19	16.5 (0.6)	17.0 (1.2)	16.3 (2.1)	14.5 (3.5)
GG – DB	18	19.0 (1.4)	20.3 (0.5)*	20.3 (1.0)*	12.0 (2.2)
GG – RB	17	22.5 (0.6)*	22.8 (0.5)*	23.0 (0.0)*	11.3 (3.3)
MP – DB	15	22.5 (0.6)**	22.8 (0.5)**	22.0 (1.2)*	10.8 (2.2)
MP – RB	20	22.0 (1.4)	21.0 (0.0)	20.3 (1.0)	17.5 (2.4)
MP – GG	17	24.0 (0.0)**	24.0 (0.0)**	24.0 (0.0)**	14.3 (1.9)
AS – DB	16	18.3 (1.0)**	18.5 (0.6)*	19.0 (0.8)*	11.3 (1.7)
AS – RB	17	22.3 (1.3)*	23.0 (1.2)*	22.0 (0.0)*	15.5 (1.7)
AS – GG	17	16.5 (0.6)*	17.0 (0.0)*	16.0 (0.0)	12.3 (1.9)
AS – MP	16	24.0 (0.0)**	24.0 (0.0)**	24.0 (0.0)**	12.5 (2.1)
MU – DB	15	15.0 (0.8)	14.8 (1.3)	14.3 (1.9)	12.5 (2.4)
MU – RB	17	15.3 (0.5)	13.8 (1.3)	17.0 (1.2)	14.8 (2.9)
MU – GG	18	18.8 (0.5)*	20.0 (0.0)**	20.8 (0.5)**	12.5 (1.3)
MU – MP	13	16.3 (1.3)	16.5 (1.0)	17.5 (0.6)	15.0 (3.8)
MU – AS	16	18.8 (1.3)*	18.3 (0.5)**	19.8 (0.5)$^{**+}$	13.8 (0.5)
Fold Total	200.8 (2.9)	233.2 (5.2)$^{\$\$**}$	234.8 (6.4)$^{\$\$**}$	236.8 (8.0)$^{\$\$**}$	160.2 (8.8)

The use of the methods in this paper in conjunction with the n-gram kernel offer a clear performance advantage over the feature-based approach. Interestingly, KPLS outperforms an SVM when using this kernel. This may suggest that for this kernel, projecting into a lower subspace is beneficial. Indeed, the performance of KPCR is also close to the SVM. The ability of KPLS however to correlate the feature directions it selects with the output variable gives it a clear advantage over KPCR and as expected from previous results on other data [1,7], a performance gain is achieved. When using the SVM in conjunction

Table 7
Performance of SDP using KPLS features in conjunction with the n-gram and string kernels. The KP-SV-n and KP-SV columns show the best performance achieved using n-gram and string kernels respectively with the SVM in conjunction with KPLS features when T was selected by cross-validation

Pairing	KP-SV-n	SDP-n	KP-SV	SDP
RB – DB	17.3 (0.5)	17.8 (1.5)	16.3 (2.1)	15.8 (0.5)
GG – DB	21.0 (0.0)	21.0 (0.0)	20.3 (1.0)	19.8 (0.5)
GG – RB	22.0 (0.0)	22.3 (1.5)	23.0 (0.0)	23.0 (0.0)
MP – DB	18.3 (0.5)	19.0 (0.8)	22.0 (1.2)	21.5 (0.6)
MP – RB	18.8 (1.3)	18.5 (1.7)	20.3 (1.0)	21.3 (0.5)
MP – GG	22.8 (0.5)	22.5 (0.6)	24.0 (0.0)	23.8 (0.5)
AS – DB	19.5 (3.0)	19.5 (2.4)	19.0 (0.8)	18.0 (1.4)
AS – RB	19.5 (0.6)	19.3 (0.5)	22.0 (0.0)	21.3 (1.3)
AS – GG	18.5 (0.6)	17.0 (3.4)	16.0 (0.0)	16.8 (0.5)
AS – MP	22.0 (0.0)	22.0 (0.0)	24.0 (0.0)	23.5 (0.6)
MU – DB	18.3 (2.9)	17.8 (2.9)	14.3 (1.9)	15.8 (0.5)
MU – RB	16.8 (1.0)	16.0 (2.0)	17.0 (1.2)	16.5 (1.3)
MU – GG	21.3 (1.0)	21.3 (0.5)	20.8 (0.5)	19.3 (1.0)
MU – MP	16.5 (3.1)	14.0 (2.0)	17.5 (0.6)	17.8 (1.0)
MU – AS	18.3 (1.7)	19.0 (0.0)	19.8 (0.5)	18.5 (1.3)
Fold Total	232.4 (7.3)	229.4 (9.7)	236.8 (8.0)	233.8 (8.0)

with the features obtained from the KPLS deflation steps, the performance improves further which has also been the case on other data sets [8].

In all cases short substrings (with substring lengths of only 1 or 2 characters) achieved the best performance, which would perhaps indicate that complex features are not used. Indeed when running the n-gram kernel, any string length above 5 has very few non-zero off-diagonal entries and therefore contains almost no information. It is interesting to note that in the experiments KPCR requires more feature directions to achieve good performance, whereas KPLS consistently requires fewer directions to perform well.

The string kernel operating over the performance alphabet (results shown in Table 6) provides significantly better classification performance than the feature-based method and in most cases (the exception being PCR) also outperforms the n-gram kernel. This indicates that the ability of the string kernel to allow gaps in matching sub-sequences is a key benefit for this data, and that complex features are indeed needed to obtain good performance. This is in contrast to results reported using the string kernel for text, where the classification rate of n-gram kernels using contiguous sequences is equal to that of the string kernel if not superior [6]. For the n-gram kernel, using KPLS features improves performance over the Support Vector Machine (significantly for some pairings). For the string kernel however, this was not the case and only slightly improved the mean, at the cost of additional variance. It is therefore not clear in which situations the use of KPLS features in conjunction with an SVM will produce a performance gain.

When using the semi-definite programming approach outlined in Section 4.1, we no longer have to use cross-validation to select the parameter T; however the sub-sequence length k and for the string kernel the gap penalty λ must still be chosen. We therefore used the same experimental set-up as before, however simply let the SDP solver choose the combination of feature directions. The results are shown in Table 7.

In both cases the SDP solution performed well, and although achieved a slightly lower mean in both cases, the differences are minor. Interestingly in nearly all cases for all parameter settings, the SDP

optimisation problem chose to weight the features beyond $T = 5$ with very little or no weight. This result is in keeping with observations from running the full experiments with all parameter settings; choosing $T > 5$ tends to lead to poor performance, hence most information is contained within the first few projections.

Overall the techniques used in this paper offer a clear performance advantage over the feature based method which delivered the previous state of the art performance on this data set. As expected KPCR does not perform as well as KPLS, the ability to use label information when selecting feature directions gives a clear performance benefit. The combination of symbolic information and kernel methods provides a significant advantage over the previous feature-based approach. Using KPLS features with an SVM improved performance greatly using the n-gram kernel, although this was not the case for the string kernel. This suggests that the feature space induced by the string kernel is able to capture more informative structure in the training examples, and is therefore particularly suited to this type of application.

6. Conclusions

In this paper we have presented a novel application of the string kernel: to classify pianists by examining their playing style given the performance on the same piece. This is an extremely complex task and has previously been attempted by analysing statistical features obtained from audio recordings. Here we have taken a different approach and have examined using feature-projection methods in conjunction with kernels which operate on text. These can be applied to the performer recognition problem by representing the performance as a string of characteristic tempo-loudness curves, which are obtained by analysing a performance worm. We have reviewed the Kernel Partial Least Squares method and shown how this can be successfully used to generate new features which can then be used in conjunction with learning methods such as an SVM. We have also shown a reweighting scheme for obtaining feature directions from KPLS that peforms better than the technique used in current literature. All algorithms tested in this paper provided higher performance than the previous state of the art results on the data. We have also shown that the ability of the string kernel to consider and match non-contiguous substrings of input sequence has a real performance benefit over only considering contiguous substrings. This is in contrast to many applications of the string kernel to text, where the relative performance of the string kernel to the n-gram kernel tends to be very close or even slightly worse. We have also shown that it is feasible to use SemiDefinite programming approaches in practice in order to remove the need to select the number of feature directions via cross-validation. It is an open problem to determine in what circumstances using KPLS to obtain features will result in an improvement in generalisation performance and this will be researched in future. Specifically however, one might expect the deflation process highlighted in this paper to be particularly useful with structure based kernels in general (e.g. string and graph kernels). In these cases the feature space mapping tends to be very high, with many irrelevant features, thus the combination of KPLS extraction and an SVM is very promising. Therefore one direction of research which we wish to pursue is the use of these techniques on enzyme function prediction tasks (using string kernels across protein sequences to predict reaction types) and drug discovery applications (using graph kernels on molecules to predict toxicity/activity of chemical compounds). One other aspect that is perhaps unsatisfactory with these techniques is that these projections can still be quite time-consuming and are not sparse, therefore we would like to investigate the use of greedy methods within the KPLS feature extraction framework.

Acknowledgements

This work was supported in part by EPSRC grant no GR/S22301/01 ("Development and Application of String-Type Kernels"), the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778 and by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under grant Y99-INF. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Education, Science, and Culture, and by the Austrian Federal Ministry for Transport, Innovation, and Technology. We would also like to thank Tjel de Bie for useful discussions.

References

[1] K.P. Bennett and M.J. Embrechts, An optimization perspective on kernel partial least squares regression, *Advances in Learning Theory: Methods, Models and Applications. NATO Science Series III: Computer & Systems Science* **190** (2003), 227–250.
[2] S. Dixon, Automatic extraction of tempo and beat from expressive performances, *Journal of New Music Research* **30**(1) (2001), 39–58.
[3] S. Dixon, W. Goebl and G. Widmer, The performance worm: Real time visualisation of expression based on langnerfls tempo-loudness animation. In *Proceedings of the international computer music conference* (*icmc 2002*), 2002.
[4] G. Grindlay and D. Helmbold, Modeling, analyzing, and synthesizing expressive piano performance with graphical models, *Machine Learning* **65**(2–3) (2006), 361–387.
[5] G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui and M.I. Jordan, Learning the kernel matrix with semidefinite programming, *Journal of Machine Learning Research* **5** (Jan 2004), 27–72.
[6] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini and C. Watkins, Text classification using string kernels, *Journal of Machine Learning Research* **2** (2002), 419–444.
[7] R. Rosipal and L. Trejo, Kernel partial least squares regression in reproducing kernel hilbert space, *Journal of Machine Learning Research* **2** (2001), 97–123.
[8] R. Rosipal, L. Trejo and B. Matthews, Kernel pls-svc for linear and nonlinear classification. In *Proceedings of the twentieth international conference on machine learning* (*icml-2003*), 2003.
[9] C. Saunders, D. Hardoon, J. Shawe-Taylor and G. Widmer, Using string kernels to identify famous performers from their playing style, in: *Proceedings of the 15th european conference on machine learning* (*ecml*) *and the 8th european conference on principles and practice of knowledge discovery in databases* (*pkdd*), (Vol. 3201), J.-F. Boulicaut, F. Esposito, F. Giannotti and D. Pedreschi, eds, Springer-Verlag Heidelberg, September 2004, pp. 384–399.
[10] B. Schölkopf, A. Smola and K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, *Neural Computation* (1998).
[11] E. Stamatatos and G. Widmer, Automatic Identification of Music Performers with Learning Ensembles, *Artificial Intelligence* **165**(1) (2005), 37–56.
[12] J. Sturm, Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones, *Optimization Methods and Software* **11**(12) (1999), 625–653. (Special issue on Interior Point Methods).
[13] G. Widmer, S. Dixon, W. Goebel, E. Pampalk and A. Tobudic, In search of the horowitz factor, *AI Magazine* **3**(24) (2003), 111–130.
[14] G. Widmer and P. Zanon, *Automatic Recognition of Famous Artists by Machine*, 2004, 1109–1110.
[15] I. Witten and E. Frank, *Data Mining*, San Francisco, CA: Morgan Kaufmann, 1999.
[16] H. Wold, Estimation of principal components and related models by iterative least squares, *Multivariate Analysis* (1966), 391–420.
[17] P. Zanon and G. Widmer, Learning to recognise famous pianists with machine learning techniques. In *Proceedings of the stockholm music acoustics conference* (*smac '03*), August 2003.