Using String Kernels to Identify Famous Performers from their Playing Style

Craig Saunders, David R. Hardoon, John Shawe-Taylor, and Gerhard Widmer

[1] {cjs,drh,jst}@ecs.soton.ac.uk
School of Electronics & Computer Science, ISIS Research Group, Building 1, Highfield, University of Southampton, Southampton, SO17 1BJ, UK
[2] gerhard@ai.univie.ac.at
Department of Medical Cybernetics and Artificial Intelligence, Medical University of Vienna, Freyung 6/2, A-1010 Vienna, Austria and Austrian Research Institute for Artificial Intelligence Freyung 6/6/7, A-1010 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 characterstics 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.

1 Introduction

This paper focuses on the problem of identifying famous pianists using only minimal information obtained from audio recordings of their playing. A technique called the performance worm which 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 on this data has compared a variety of machine learning techniques whilst using as features statistical quantities obtained from 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. We show that using this representation delivers an improvement in performance over the current best results obtained using a feature-based approach. The ability of the string kernel to include non-contiguous features is shown to be key in the performance of the algorithm.

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 algorithm and string kernel function used to analyse the data 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.

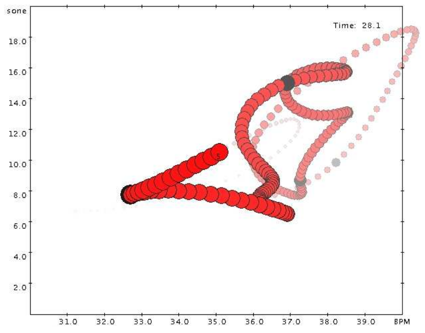
2 A Musical Representation

The data used in this paper, first described in [2], 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 both 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 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, 2, 4]. The performance worm extracts data from

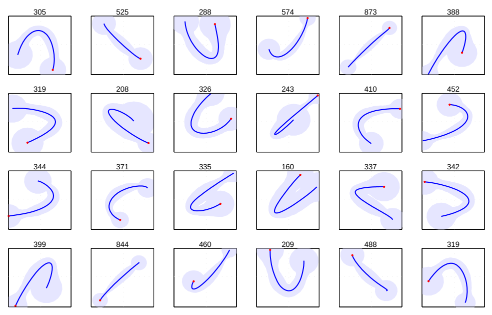
Table 1. Movements of Mozart piano sonatas selected for analysis.

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

audio recordings by examining tempo and general loudness of the audio when measured at the beat level. An interactive beat tracking program [5] 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(a) shows a screenshot of the worm in progress. Note that this is the only information used in the



(a)



(b)

Fig. 1. (a) The performance worm: A 2D representation of changes in beat-level tempo and loudness can be plotted in realtime from an audio recording. (b) The performance alphabet: A set of cluster prototypes extracted from the performance worm.

creation of the worm, more detailed information such as articulation, individual voicing or timing details of that 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 [4] 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 Figure 1(b)). 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 [2, 4]. 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 through 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

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

Table 2. List of pianists and recordings used

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. [6]). Partial least squares and its kernel-based variant KPLS has recenlty gained popularity within the machine learning community [7–9] and either can be used as a method for regression or classification, or as a method for dimension reduction. These methods however are still not very well known within the community and it is not always clear how to use these method to generate new input features for training and test data, so we shall briefly review the methods here. We start with a description of the original linear PLS algorithm, and then move on to the kernel variant.

3 Previous results

3.1 String kernels

The use of string kernels for analysing text documents was first studied by Lodhi et al. [1]. We briefly review the approach to creating a feature space and associated kernel.

The key idea behind the gap-weighted subsequences kernel is to compare strings by means of the subsequences they contain – the more subsequences in common, the more similar they are – rather than only considering contiguous n-grams, the degree of contiguity of the subsequence 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 subsequence $u = s(\mathbf{i})$ 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 subsequences kernel). *The feature space associated with the gap-weighted subsequences kernel of length p is indexed by $I = \Sigma^p$, with the embedding given by*

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

The associated kernel is defined as

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

Example 1. Consider the simple strings `"cat"`, `"car"`, `"bat"`, and `"bar"`. Fixing $p = 2$, the words are mapped as follows:

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

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

$$\kappa(\texttt{"cat"}, \texttt{"cat"}) = \kappa(\texttt{"car"}, \texttt{"car"}) = 2\lambda^4 + \lambda^6$$

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

We omit a description of the efficient dynamic programming algorithms for computing this kernel referring the reader to Lodhi et al. [1].

3.2 Partial Least Squares

Partial Least Squares (PLS) was developed by Herman Wold during the 1960's in the field of econometrics [10]. 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. [9]. The procedure for PLS feature extraction is shown in Algorithm 1. 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 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

$$\phi_{j+1}(\mathbf{x})' = \phi_j(\mathbf{x})' \left(\mathbf{I} - \mathbf{u}_j \mathbf{p}'_j\right),$$

Algorithm 1 Pseudocode for PLS feature extraction.

The PLS feature extraction algorithm is as follows:

```
input   | Data matrix $\mathbf{X} \in \mathbb{R}^{\ell \times N}$, dimension $k$, target vectors $\mathbf{Y} \in \mathbb{R}^{\ell \times m}$.
process | $\mathbf{X}_1 = \mathbf{X}$
        | for $j = 1, \dots, k$
        |    let $\mathbf{u}_j, \mathbf{v}_j, \sigma_j$ be the first singular vector/value of $\mathbf{X}_j'\mathbf{Y}$,
        |    $\mathbf{X}_{j+1} = \left(\mathbf{I} - \frac{\mathbf{X}_j\mathbf{u}_j\mathbf{u}_j'\mathbf{X}_j'}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}\right)\mathbf{X}_j = \mathbf{X}_j\left(\mathbf{I} - \frac{\mathbf{u}_j\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}\right)$
        | end
output  | Feature directions $\mathbf{u}_j$, $j = 1, \dots, k$.
```

where

$$\mathbf{p}_j = \frac{\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j},$$

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

$$\phi(\mathbf{x})' = \phi_{k+1}(\mathbf{x})' + \sum_{j=1}^{k} \phi_j(\mathbf{x})' \mathbf{u}_j \mathbf{p}_j'.$$

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

$$\hat{\phi}(\mathbf{x}) = \left(\phi_j(\mathbf{x})' \mathbf{u}_j\right)_{j=1}^{k},$$

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

$$\begin{aligned}\phi(\mathbf{x})'\mathbf{U} &= \phi_{k+1}(\mathbf{x})'\mathbf{U} + \sum_{j=1}^{k} \phi_j(\mathbf{x})' \mathbf{u}_j \mathbf{p}_j'\mathbf{U} \\ &= \phi_{k+1}(\mathbf{x})'\mathbf{U} + \hat{\phi}(\mathbf{x})'\mathbf{P}'\mathbf{U},\end{aligned}$$

where $\mathbf{P}$ is the matrix whose columns are $\mathbf{p}_j$, $j = 1, \dots, k$. Finally, it can be verified that

$$\mathbf{u}_i'\mathbf{p}_j = \delta_{ij} \quad \text{for } i \leq j. \tag{1}$$

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

$$\phi_{k+1}(\mathbf{x})'\mathbf{u}_j = \phi_j(\mathbf{x})' \prod_{i=j}^{k} (\mathbf{I} - \mathbf{u}_i\mathbf{p}_i')\mathbf{u}_j = 0, \text{ for } j = 1, \dots, k.$$

It follows that the new feature vector can be expressed as

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

Furthermore, the overall regression coefficients can be computed as

$$\mathbf{W} = \mathbf{U}\left(\mathbf{P}'\mathbf{U}\right)^{-1}\mathbf{C}', \tag{2}$$

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

$$\mathbf{c}_j = \frac{\mathbf{Y}'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}.$$

This appears to need a matrix inversion, but equation (1) implies that the matrix $\mathbf{P}'\mathbf{U}$ is upper triangular with constant diagonal 1 so that the computation of

$$\left(\mathbf{P}'\mathbf{U}\right)^{-1}\mathbf{C}'$$

only involves the solution of m sets of k linear equations in k unknowns with an upper triangular matrix.

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. The vector

Algorithm 2 Pseudocode for kernel-PLS

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

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

  **for** $i = 1, \ldots, k$ **do**
    $\beta_i$ = first column of $\hat{Y}$
    normalise $\beta_i$
    **repeat**
      $\beta_i = YY'K_i\beta_i$
      normalise $\beta_i$
    **until** convergence
    $\tau_i = K_i\beta_i$
    $c_i = \hat{Y}'\tau_i$
    $\hat{Y} = \hat{Y} - \tau_i c_i'$
    $K_{i+1} = (I - \tau_i\tau_i')K_i(I - \tau_i\tau_i')$
  **end for**

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

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

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

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

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

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

Here $\operatorname{diag}(\tau_i'\tau_i)$ is the diagonal matrix with entries $\operatorname{diag}(\tau_i'\tau_i)_{ii} = \tau_i'\tau_i$, where $\tau_i = K_i\beta_i$. Finally, again using the orthogonality of $\mathbf{X}_j\mathbf{u}_j$ to τ_i, for $i < j$, we obtain

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

making

$$\mathbf{C} = \mathbf{Y}'\mathbf{T}\operatorname{diag}(\tau_i'\tau_i)^{-1}\operatorname{diag}(\mathbf{a}).$$

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

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

It is tempting to assume like [9] that a dual representation of the PLS features is then given by

$$\mathbf{B}(\mathbf{T}'\mathbf{K}\mathbf{B})^{-1},$$

but in fact

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

so that the dual representation is

$$\mathbf{B}(\mathbf{T}'\mathbf{K}\mathbf{B})^{-1}\operatorname{diag}(\mathbf{a})^{-1}\operatorname{diag}(\tau_i'\tau_i) = \mathbf{B}(\mathbf{T}'\mathbf{K}\mathbf{B})^{-1}\operatorname{diag}(\tau_i'\tau_i)\operatorname{diag}(\mathbf{a})^{-1}.$$

The missing diagonal matrices perform a rescaling of the features extracted, which skews the geometry of the space and affects the performance of for example an SVM. The quantities a_i are difficult to assess, though these should not vary significantly over similar adjacent features since they will be related to the corresponding singular values. In our experiments we have compared the results that can be obtained ignoring both diagonal matrices with those obtained including $\operatorname{diag}(\tau_i'\tau_i)$.

5 Experiments

In our experiments we followed the setup given in [2]. 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 [2]) 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 [11] 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 two options, either to use the features as described in [9] or to include the extra reweighting factors diag $(\tau_i'\tau_i)$ described above. 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) and the reweighted (REW) features. Table 3 shows the results obtained. There is a clear advantage shown for the reweighting scheme and so we adopted this method for the remaining experiments.

In the remaining experiments we chose one pair of composers (RB–DB) to use to select the various parameters. These 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. In each case the parameters that delivered the best performance on the RB–DB data were chosen.

Method/T	1	2	3	4	5	6	7	8	9	10
ORIG	287	265	248	251	253	250	256	252	246	238
REW	287	295	293	293	296	296	295	295	295	295

Table 3. Total number of correct predictions across all splits against number of feature directions used (T) for both the feature projection method described in this paper (REW) and that in previous work (ORIG). The parameters used in this were those optimal for the KP-SV combination ($k = 5$,$\lambda = 0.9$).

Once selected the settings of these parameters were fixed for the remaining test experiments for all of the results reported in Table 4 – note that the RB–DB row is deliberately left blank to emphasise this.

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, **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.

Table 4. Comparison of algorithms across each pairwise coupling of performers. Note that in all case the figures given are the number of movements correctly identified out of a maximum of 24. FB represents the previous best results using a feature-based representation rather than the 'performance alphabet' used for the other approaches.

		String Kernel				n-gram Kernel			
Pairing	FB	KPLS	SVM	KP-SV	KPCR	KPLS-n	SVM-n	KP-SV-n	KPCR-n
RB - DB	–	–	–	–	–	–	–	–	–
GG - DB	15	19	21	22	21	18	18	22	14
GG - RB	17	24	22	23	24	21	21	20	11
MP - DB	17	18	18	20	17	16	16	18	17
MP - RB	21	22	22	23	19	18	15	17	12
MP - GG	17	23	23	23	23	20	20	22	18
AS - DB	15	19	19	19	18	15	16	16	10
AS - RB	16	23	23	21	16	17	17	20	14
AS - GG	17	17	18	18	17	18	15	18	13
AS - MP	20	23	23	22	22	20	17	22	14
MU - DB	17	15	15	15	13	13	13	14	12
MU - RB	16	17	17	14	11	18	16	17	12
MU - GG	16	19	19	19	20	16	16	20	14
MU - MP	15	18	19	17	17	17	17	21	16
MU - AS	17	16	16	18	16	16	15	16	17
Total	236	273	275	274	254	243	232	263	194
Average (%)	70.2	81.3	81.9	81.6	75.6	72.3	69.5	78.2	57.7

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 suggests 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 ([7, 8]), a performance gain is achieved. When using the SVM in conjunction with the features obtained from the KPLS deflation steps, the performance improves further which has also been the case on other data sets [9]. In all cases short substrings achieved the best performance (with substring lengths of only 1 or 2 characters, which would perhaps indicate that complex features are not used). 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 provides significantly better classification performance than the feature-based method and in every case also outperforms the n-gram kernel. This would indicate that the ability of the string kernel to allow gaps in matching subsequences 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 [1]. For the string kernel however, the use of KPLS features did not improve the performance of the support vector machine (in fact over all of the data it made 1 more misclassification). It is therefore not clear in which situations the use of KPLS features in conjunction with an SVM will produce a performance gain.

6 Conclusions

In this paper we have presented a novel application of the string kernel: to classify pianists by examining their playing style. 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. It is an open problem to determine in what circumstances using KPLS to obtain features will result in an improvement in generalisation performance. Also, currently the number of dimensions has to be chosen via cross-validation or some other method, therefore an automatic selection method for this parameter would also be beneficial. These two problems will be addressed in future research.

7 Acknowledgements

This work was supported by EPSRC grant no GR/S22301/01 ("Development and Application of String-Type Kernels") 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.

References

1. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. Journal of Machine Learning Research (2002) 419–444
2. Zanon, P., Widmer, G.: Learning to recognise famous pianists with machine learning techniques. In: Proceedings of the Stockholm Music Acoustics Conference (SMAC '03). (2003)
3. Dixon, S., Goebl, W., Widmer, G.: The performance worm: Real time visualisation of expression based on langner's tempo-loudness animation. In: Proceedings of the International Computer Music Conference (ICMC 2002). (2002)
4. Widmer, G., Dixon, S., Goebel, W., Pampalk, E., Tobudic, A.: In search of the horowitz factor. AI Magazine **3** (2003) 111–130
5. Dixon, S.: Automatic extraction of tempo and beat from expressive performances. Journal of New Music Research **30** (2001) 39–58
6. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation (1998)
7. Rosipal, R., Trejo, L.: Kernel partial least squares regression in reproducing kernel hilbert space. In: Journal of Machine Learning Research 2. (2001) 97–123
8. Bennett, K.P., Embrechts, M.J.: 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
9. Rosipal, R., Trejo, L., Matthews, B.: Kernel pls-svc for linear and nonlinear classification. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003). (2003)
10. Wold, H.: Estimation of principal components and related models by iterative least squares. Multivariate Analysis (1966) 391–420
11. Witten, I., Frank, E.: Data Mining. Morgan Kaufmann, San Francisco, CA (1999)