
Can Style be Learned? A Machine Learning Approach Towards ‘Performing’ as Famous Pianists

Louis Dorard
louis@dorard.info

David R. Hardoon
D.Hardoon@cs.ucl.ac.uk

John Shawe-Taylor
jst@cs.ucl.ac.uk

Dept. of Computer Science
University College London
London WC1E 6BT, U.K

Abstract

In this paper a novel method for performing music in the style of famous pianists is presented. We use Kernel Canonical Correlation Analysis (KCCA), a method which looks for a common semantic representation between two views, to learn the correlation between a representation of a musical score and a representation of an artist’s performance of that score. We use the performance representation based on the variations of beat level global loudness and tempo through time, as suggested by [3]. Therefore, the crux of the matter is the representation of the musical scores and by implication a similarity measure between relevant features that capture our prior knowledge of music. We therefore proceed to propose a novel kernel for musical scores, which is a Gaussian kernel adaptation to the distances between rhythm patterns, melodic contours and chords progressions.

1 Introduction

These days computers are able to reproduce accurately an artist’s performance of a piece of music for piano with MIDI synthesisers that can take into account information on the keys that are pressed, the timing, and the way they are pressed (hammer velocity on the strings, release of the key, setting of the damper). All these parameters are set from the recording of a human pianist on systems such as the Computer Grand Piano by Bösendorfer.

A computer would not be able to set the parameters’ values by itself in a way that would produce a musically interesting performance. We could only provide the computer with a computer representation of the musical score (such as the note matrix, as defined by [4]¹)

¹To each row of the note matrix corresponds a musical event. The first column gives the onset time of the event and the fourth column gives the pitch.

and then the machine would just know which notes to play and at which times. The rest of the parameters of the performance would have to be initialised to default values and wouldn't change. This would result in a mechanical and flat performance, allowing no dynamic nor time delay, and consequently lacking any emotion.

One could argue that the musical score actually gives information on how to perform the piece, but this information is only qualitative in nature (e.g. *p*, *f*, *cresc.*, *ral.*, *acc.*) and therefore it cannot be readily implemented in the computer representation of the score.

The purpose of the present work is to use machine learning techniques in order to enable the computer to perform a piece of music in the way a human would, and preferably in the style of one particular performer.

Previous work such as [6] presented a technique for machine performance based on an inductive rule-learning algorithm. Thus, they were able to predict the dynamics of performances, as well as the changes of tempo. However their technique was restricted as they did not consider the specificities of individual performers. They sought commonalities across all performers in the change in loudness and in tempo, to establish general rules. Besides, their method did not consider combined changes in loudness and tempo, and harmonies were not considered in the analysis of scores.

We seek here to achieve machine performance by correlating musical scores with performances of these scores. We assume that these two objects are two views of the same musical semantic content *S*. Note that there is only one view of *S* in the form of a symbolic representation (score) but there can be several views of *S* in the form of a performance transcription (MIDI or audio recording for instance), as two different pianists will perform differently the same piece of music or the same pianist could perform differently at different moments.

Although it should also be possible to use MIDI recordings as performance transcriptions (made by the Computer Grand Piano for example), we choose to work with audio recordings which are more common especially when we look for recordings from famous pianists. From these audio recordings we extract performance worms as introduced by [3]. Thereafter, a performance will be associated to its worm and we won't have to consider audio recordings again.

Worms are sequences of 2 real values that give the evolution of tempo and loudness in time in the performance. Each (tempo, loudness) pair in the worm corresponds to a beat on the score. Figure 1 shows a graphical view of a worm as a 2D trajectory on a graph. Pianists' specificity's show with characteristic worm shapes, whereas a machine's performance worm would be immobile.

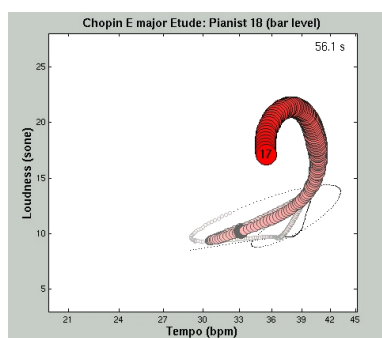


Figure 1: *Smoothed graphical view of a worm*

Beat	Tempo (bpm)	Loudness (sone)
1	22.3881	3.2264
2	22.3881	2.3668
3	21.4286	6.7167
4	19.0597	4.2105
5	28.1426	8.3444
6	30.0000	10.2206
7	26.7857	14.1084
8	25.8621	14.0037
9	35.7143	7.8521
...

Figure 2: *Machine representation of a worm*

In our approach we confine ourselves to expressive music with continuous upper part melody and accompaniment², as we believe this type of music to be simpler to analyse for the purpose of performance. It may be interesting to consider scores that make use of *rubato* as this implies important variations in tempo, at the discretion of the performer. It would make it easier to see whether our technique yields good results. ‘Etude 3 Opus 10’ by Chopin (Figure 3) is a prime example of the kind of music we consider as it has all the characteristics mentioned above.

For such music we can easily extract the melody, and then the harmonies: we assign to each note of the melody the chord made up of the notes played in the accompaniment while the note of the melody stays on. Thus we create a sequence of (note, chord) pairs occurring one after another, which we call ‘feature representation’.

This is musically relevant as it captures both the ‘vertical’ and the ‘horizontal’ dimensions of music. Figure 4 shows an example of the extracted feature representation of a musical score. We will use our learning algorithms with these feature representations instead of scores in the note matrix format. The principal advantage of the feature representation is that, unlike the score, no event can occur while another event is already on.



Figure 3: First two bars of Etude 3 Opus 10 by Chopin

Beat	Melody	Chord
1	B3	B3
2	E3	[E2 B2 G#3 B3 E4]
3	D#3	[E2 B2 G#3 D#3]
...

Figure 4: Feature representation of the score in Figure 3

The audio recordings used initially are all chosen from the same performer, which style we try to learn, and there is only one recording per score. Hence, the data used in our system is a paired data-set containing first, score feature representations and second, performance worms. The system learns how a famous pianist performs by relating the worm to the structure of the music as it is written on the score, and then perform new pieces of music using the learned style. We use KCCA to correlate the two views and to learn the common “semantic space” in which they are to be projected. When dealing with a new score, we seek to generate a worm sequence that maximises the similarity with the score in the common semantic space (i.e. the inner product in that space). The worm coordinates are then used to set the timing of notes and their velocity, thus generating a performance – limited, since the attack of a key on the piano cannot be summarised with just one value for the velocity.

In order to apply KCCA to our data, we first need to design a kernel that applies to sections of worm and a kernel that applies to sections of musical score, which we call ‘Music Kernel’ and we present in this paper.

2 Music Kernel

We begin by introducing some notations that will be used throughout the paper. Let s_1 denote a sequence of events in time $(s_{1,j})_j$ such that

$$(j_1 < j_2) \Rightarrow (s_{1,j_1} \text{ happens before } s_{1,j_2}). \quad (1)$$

Each event e is characterised by its position in time – also called “onset time” – and its attributes that we assume to be vectors of real numbers. Events occur one after another,

²Continuous: no rests; Upper part: at any given time, the note with the highest pitch that can be heard is a note of the melody; Accompaniment: the notes which are not part of the melody and which are used to create harmonies.

and cannot overlap. Therefore, the duration of an event can be determined from its onset time and the onset time of the following event.

We design a kernel that is invariant to shifts of sequences s_i across time. Hence, the kernel should only be sensitive to the first derivative of the onset times of the events i.e. the durations of the events. For more convenience, we can shift all the sequences s_i so that they all start at the same onset time – say instant 0 – which does not affect the kernel products.

2.1 Events of same durations

Let s_1, s_2 be time sequences with the same number of events. We also assume in this section that the events in s_1 and s_2 have the same durations:

$$\forall j, \text{duration}(s_{1,j}) = \text{duration}(s_{2,j}). \quad (2)$$

The sequences starting both at instant 0, their events have respectively the same onset times and thus it makes sense to compare them 2 by 2.

Let κ_{j_0} be a measure of similarity between such two sequences, around a particular location j_0 in time (called point of reference). Let d_{j_0} denote a distance between sequences with j_0 as point of reference, and d_e a distance between single events.

Distance between events d_e The distance between sequences is based on a distance between events. Since events' attributes are vectors of real numbers, the distance between events can be set to be a combination of the Euclidian distances between the attributes.

Let $\hat{s}_{1,j}$ denote the single attribute of an event $s_{1,j}$. We may want to consider the first difference of the attribute with respect to the indices, $\hat{s}'_{1,j}$, rather than the attribute itself (see Equation 3)³. For instance, [1] consider the ‘melodic interval’ for the analysis of melodies, which is the first difference of the pitch of a musical note. This would allow us to better capture the structure of the time sequence data as we would focus on the changes of the data in time instead of the data itself.

$$\hat{s}'_{1,j} = \frac{\hat{s}_{1,j} - \hat{s}_{1,j-1}}{j - (j-1)} = \hat{s}_{1,j} - \hat{s}_{1,j-1} \quad (3)$$

$$d_e(s_{1,j}, s_{2,j}) = \|\hat{s}_{1,j} - \hat{s}_{2,j}\| \quad (4)$$

$$\text{or } \|\hat{s}'_{1,j} - \hat{s}'_{2,j}\|. \quad (5)$$

Distance between sequences d_{j_0} As time is the important dimension of the sequences we consider, we give more importance to events that are close to the point of reference than to events that are far away from it. For this, we apply a Gaussian decay factor to distances between single events depending on the number of events that separate them from the point of reference:

$$d_{j_0}(s_1, s_2)^2 = \sum_j e^{-\frac{(j-j_0)^2}{\sigma^2}} d_e(s_{1,j}, s_{2,j})^2. \quad (6)$$

Applying a Gaussian to the distance d_{j_0} gives the music kernel as:

$$\kappa_{j_0}(s_1, s_2) = \exp\left(-\frac{d_{j_0}(s_1, s_2)^2}{\tau^2}\right). \quad (7)$$

³Another possibility is to consider the derivative with respect to time:

$$\frac{\hat{s}_{1,j} - \hat{s}_{1,j-1}}{\text{time}(s_{1,j}) - \text{time}(s_{1,j-1})}$$

2.2 Events of different durations

It does not make much sense to compare events from s_1 and s_2 when they do not have the same durations, because events will not happen at the same times within the sequence. For this reason, we transform one of the two sequences to be compared, say s_1 , so that its events happen at the exact same times in the sequence as those of s_2 , i.e. they have same durations. We call such an operation a ‘projection’ and denote the resulting sequence by $s_{(1\rightarrow 2)}$.

Figure 5 shows the behaviour of the projection algorithm on two examples. Each event is a rectangle, time is represented on a horizontal axis, and the attributes of the events are represented with the brightness of the rectangles’ colours.

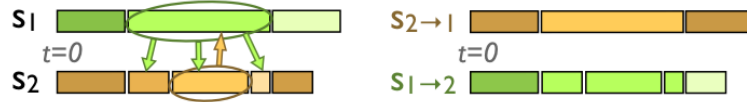


Figure 5: Illustration of the projection algorithm

The sequence $s_{(1\rightarrow 2)}$ can be used instead of s_1 in Equation 7 to compute a distance with s_2 . However, we may lose events of the original sequence when projecting it, as shown in Figure 5. In order not to lose the specificities of s_1 and s_2 when extending our distance to sequences of events of different durations, we define our new distance to be

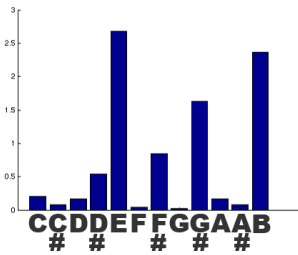
$$d_{j_0}(s_1, s_2) = \sqrt{d_{j_0}(s_1, s_{(2\rightarrow 1)})^2 + d_{j_0}(s_{(1\rightarrow 2)}, s_2)^2} \quad (8)$$

Note that projection also makes a certain error since the durations pattern of $s_{(1\rightarrow 2)}$ is the same as the one of s_2 , which is likely to be different from the durations pattern of s_1 .

A measure of that error, $e_{1\rightarrow 2}$, can be given by comparing the duration of each event $s_{(1\rightarrow 2),j}$ with the duration of the event e chosen in s_1 it originates from. Errors $e_{1\rightarrow 2}$ and $e_{2\rightarrow 1}$ can then both be incorporated into the new distance d_{j_0} defined in Equation 8.

2.3 Application to scores

A score’s feature representation (Figure 4) is a time sequence in which events are (note, chord) pairs. The performance of the Music Kernel is dependent on the attributes of these events as the kernel is based on the distance between events which is based on a distance between the attributes. It is quite natural to consider the pitch, which is an integer, as the attribute for a note. However it is not obvious which numerical attribute should be considered for a chord and could account its harmonic characteristics.



[2] developed a 12 dimensional representation of chords based on psycho-acoustic considerations. When a chord is played, not only the notes that make up the chord can be heard but also the harmonics of these notes. For instance, when a $C3$ is played, a $G4$ can be heard as a harmonic, with a lesser loudness. We first determine the loudnesses of all audible notes when the chord is sounded. Then, for each pitch-class ($C, C\#, D, \dots$) the loudnesses of the notes that belong to it are summed, this resulting in a vector of 12 real values that we consider as the attribute

for the chord. It appears then that if two chords correspond to two different inversions of the same harmony, their attributes will be similar even though the chords could be made of different notes (e.g.: $[E2 G\#3 B3 E4]$ and $[B2 E3 D2 G\#2]$ for E major). The figure above shows the attribute of the chord $[E2 B2 G\#3 B3 E4]$ as a histogram.

3 Correlation Analysis

Proposed by Hotelling in 1936, Canonical Correlation Analysis (CCA) is a technique for finding pairs of basis vectors that maximise the correlation between the projections of paired variables onto their corresponding basis vectors. Correlation is dependent on the chosen coordinate system, therefore even if there is a very strong linear relationship between two sets of multidimensional variables this relationship may not be visible as a correlation. CCA seeks a pair of linear transformations one for each of the paired variables such that when the variables are transformed the corresponding coordinates are maximally correlated. Due to limited space we refer the reader to [5] for detailed account of CCA and kernel CCA (KCCA). We use KCCA to learn the correlation between short pieces of scores and their respective worms. These pieces should stay short so that their respective kernels are able to detect similarities between them.

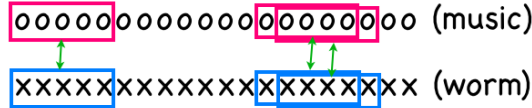
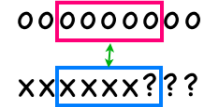


Figure 6: KCCA Training

Training examples are taken at many different onset times, in order to have a whole variety of pieces of music and score. They are windows in time delimited by a left and right limit around a point of reference. Figure 6 shows a view of some windows used as training examples. Each character ‘o’ or ‘x’ corresponds to one beat. The KCCA algorithm is provided with the kernel matrices of the two views of these training examples. Its outputs are the canonical correlation matrices for the two views.

We use KCCA for the task of worm completion: the original worm is known only until a certain point in time. The unknown section of worm is represented by ‘?’ characters. We use a greedy algorithm to determine the worm coordinates to replace the ‘?’s with: we consider a window of the score and of the worm such that it only includes the first ‘?’ and we replace the unknown point by several test worm points; the one we retain is the point for which the correlation between the worm and the score was maximal. We repeat the operation until there is no ‘?’ left.



The correlation value is given by the inner product between the representation of the worm and of the score in the semantic space. Equation 9 shows how to obtain the representation of an object te in the semantic space, also called its semantic vector sv . A is the canonical correlation matrix for the view that te corresponds to, and is obtained after training the system with KCCA.

$$sv = A \begin{pmatrix} \kappa_{j_0}(te, tr_1) \\ \kappa_{j_0}(te, tr_2) \\ \dots \\ \kappa_{j_0}(te, tr_n) \end{pmatrix}, \text{ where } tr_i \text{ are the training examples.} \quad (9)$$

4 Experiments

Experiments on the automated completion of the worm, which leads to machine performance, have been carried out on the first page of ‘Etude 3 Opus 10’. We used the first half for training (bars 1 to 8) and the beginning of the second half for tests (bars 9 to 12). The worm used for training, and for comparison with the result of worm completion on the test section, was extracted from a recording of Claudio Arrau in 1956. The system is

only tested on rote learning – that we wish to be successful before testing generalisation in the future – which explains why the test section has been chosen to be very similar to the training section. We expect the machine to mimic the style of the pianist it has learnt from training, hence to produce a worm similar to the original “pianist’s worm”.

Parameters. In the previous sections we have introduced several system parameters, namely: the relative importance of melody, chords and errors of projection in the music kernel ; the parameters σ and τ , as defined in Equations 6 and 7 ; the left and right limits of the window for scores and worms. We choose to intuitively tune these parameters, noting that the quality of the results is highly affected by the value of σ . The correct approach for tuning the parameters would be through a cross validation procedure.

Measure of the quality of a performance. We compare worm points (in 2 dimensions) at each instant in time between the worm produced by the machine and the original pianist’s worm. Comparison between two worm points is given by the inner product between these points normalised by the norm of the original worm’s point. These are then added for each pair of worm points at each instant in time and divided by the number of instants in time considered. Thus, we obtain 1 when comparing the original worm with itself.

This measure of the quality of the performance can be used to test our system against random worm completion. The latter randomly chooses the deltas with the previous worm point at each instant it is completing the worm. Since the deltas with previous worm point observed in the training data seem to have a Gaussian distribution, we use the learnt Gaussian in order to draw random deltas in random worm completion.

Unfortunately, it has not been possible yet to compare our system to the one of [6] and this remains a subject of future work.

Results. The system as it has been described until here produced worms which coordinates went off the extremes observed in the pianist’s performance. Consequently, the norms of the worms were considerably bigger than the norm of the original pianist’s worm. The reason for this is still unclear, however, we were able to bring modifications to the system in order to correct its bad behaviour: incorporation of the Gaussian prior that was mentioned before in the choice of the worm point to replace ‘?’ with – this privileges small worm deltas – and dynamic re-scaling of the worm coordinates in case they are off the extremes, as it is shown in Figure 7 in one dimension.



Figure 7: Illustration of dynamic re-scaling that replaces x_{next} by x'_{next}

Using the music kernel with KCCA we were able to obtain better results than with a randomly generated worm. The worm inferred by the machine had a measure of quality of 0.95 whereas for 100 runs of the random worm completion the mean value was 0.51. The music kernel was also specifically tested and proved successful in detecting the similarities between test sections of musical score of length 6 beats. Similarities were shown in terms of melodic contour and chords progressions independently, and were in concordance with intuitive notions of musical similarities.

5 Conclusions

The proposed music kernel was shown to be able to capture the structure of music and allowed the algorithm of worm completion to perform better than random, after corrections

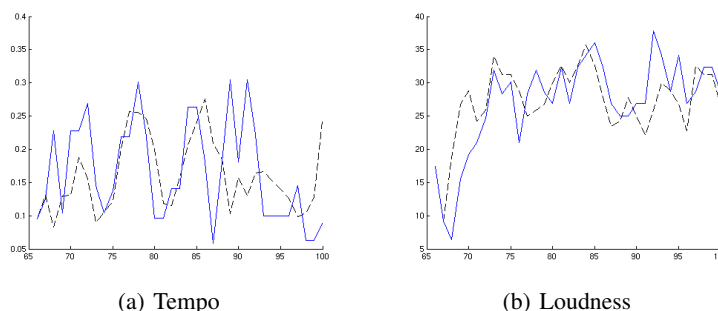


Figure 8: Variations in time of the performance parameters for the machine (blue line) and for the pianist (dotted line)

were made. However, although these results look encouraging, the resulting performances were not pleasant to listen to.

Future work should investigate the causes of the bad behaviour of the system that forced us to bring corrections. One suggestion is the greedy character of the algorithm which optimises the worm at only one instant and could consequently favour important worm deltas. It would be worth while to use dynamic programming techniques in order to optimise the worm completion on segments of several instants and this way look ahead on the score, as in fact a human would do.

Acknowledgements This work was supported by the EPSRC project ‘Learning the Structure of Music’ (LeStruM)⁴, EP-D063612-1. The authors would also like to thank Gerhard Widmer for providing the famous pianists’ performance worms.

References

- [1] M. Bergeron and D. Conklin. Representation and discovery of feature set patterns in music. In *Proceedings of the International Workshop on Artificial Intelligence and Music, 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1–12, Hyderabad, India, 2007. 4
- [2] Jean-François Paiement, Douglas Eck, Samy Bengio, and David Barber. A graphical model for chord progressions embedded in a psychoacoustic space. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005. 5
- [3] S. Dixon, W. Goebel, and G. Widmer. 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. 1, 2
- [4] Tuomas Eerola and Petri Toiviainen. Mir in matlab: The midi toolbox. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, 2004. 1
- [5] David R. Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: an overview with application to learning methods. *Neural Computation*, 16:2639–2664, 2004. 6
- [6] G. Widmer, S. Dixon, W. Goebel, E. Pampalk, and A. Tobudic. In search of the Horowitz factor. *AI Magazine*, 3(24):111–130, 2003. 2, 7

⁴LeStruM project website <http://www.lestrum.org/>.