Mach Learn (2006) 64:5–24
DOI 10.1007/s10994-006-8260-4

Relational IBL in classical music

Asmir Tobudic · Gerhard Widmer

Received: 25 June 2004 / Revised: 17 February 2006 / Accepted: 2 March 2006 / Published online: 8 May 2006

Abstract It is well known that many hard tasks considered in machine learning and data mining can be solved in a rather simple and robust way with an instance- and distance-based approach. In this work we present another difficult task: learning, from large numbers of complex performances by concert pianists, to play music expressively. We model the problem as a multi-level decomposition and prediction task. We show that this is a fundamentally relational learning problem and propose a new similarity measure for structured objects, which is built into a relational instance-based learning algorithm named DISTALL. Experiments with data derived from a substantial number of Mozart piano sonata recordings by a skilled concert pianist demonstrate that the approach is viable. We show that the instance-based learner operating on structured, relational data outperforms a propositional *k*-NN algorithm. In qualitative terms, some of the piano performances produced by DISTALL after learning from the human artist are of substantial musical quality; one even won a prize in an international 'computer music performance' contest. The experiments thus provide evidence of the capabilities of ILP in a highly complex domain such as music.

Keywords Relational instance-based learning · Music

1. Introduction

Instance-based learning has always been very popular within machine learning and data mining. During the long research history on IBL, countless studies have stressed its strong aspects: algorithmic simplicity, incrementality, almost obvious extensions for handling noisy

Editors: Tamás Horváth and Akihiro Yamamoto

A. Tobudic
Austrian Research Institute for Artificial Intelligence, Vienna

G. Widmer (✉)
Department of Computational Perception, Johannes Kepler University Linz,
and Austrian Research Institute for Artificial Intelligence, Vienna
e-mail: gerhard.widmer@jku.at

examples and/or attributes, the ability to deal with discrete as well as with continuous attributes, and often surprisingly good performance. Although most research on IBL has been done in a propositional setting, recently there has been a growing interest in transferring the successful IBL framework to the richer first-order logic (FOL) resentation. A number of instance-based learners operating in the FOL framework have already been developed—e.g. KBG (Bisson, 1992), RIBL (Emde & Wettscherek, 1996), STILL (Sebag & Rouveirol, 1997)—and shown to work well on a number of tasks.

This paper introduces another difficult task for relational IBL, from the area of music research. We would like to automatically build, via inductive learning from 'real-world' data (i.e., real performances by highly skilled musicians), predictive models of certain aspects of performance (e.g. expressive tempo, timing, dynamics, etc.). Previous research has shown that computers can indeed find and describe interesting and useful regularities at the level of individual notes. Using a new machine learning algorithm (Widmer, 2003), we succeeded in discovering a small set of simple, robust and highly general rules that predict a substantial part of the note-level choices of a performer (e.g., whether (s)he will shorten or lengthen a particular note) with high precision (Widmer, 2002). However, music performance is a highly complex activity, and the note level is far from sufficient as a basis for a complete model of expressive performance. The goal of our ongoing work is to build quantitative models of musical expression at different levels of abstraction: we would like to learn tempo and dynamics strategies at levels of *hierarchically nested phrases*.

In this paper we show how relational IBL can be applied to learn expressive tempo and dynamics patterns at different phrase levels. We also propose a new similarity measure for structured objects described in FOL, which is a fairly straightforward modification of existing measures and can be regarded as a combination of two techniques: (1) RIBL's (Emde & Wettscherek, 1996) strategy for assessing similarity between FOL objects by computing the similarity between objects' properties and the similarity of the objects related to them, and (2) a definition of distance between two sets based on the notion of transport networks, as proposed in Ramon and Bruynooghe (1998).

Our similarity measure has been built into a relational instance-based learner named DISTALL and applied to our music task. DISTALL predicts timing and dynamics patterns for phrases in a new piece by analogy to the most similar phrases in the training set. Empirical evaluation shows that the relational IBL approach is viable in the complex real-world domain—the pieces learned by DISTALL are of substantial musical quality; one even won a prize in an international 'computer music performance' contest. Experiments also show that DISTALL produces clearly better results than both the related algorithm RIBL and a propositional k-NN. The latter provides additional evidence for the benefits of relational instance-based learning.

The paper is organized as follows: Section 2 introduces the notion of expressive music performance and its representation via performance curves. We also show how hierarchically nested musical phrases are represented in FOL, and how complex tempo and dynamics curves can be decomposed into well-defined training instances for the instance-based learning algorithm. In Section 3 we describe our distance measure in detail. Section 4 gives empirical results on DISTALL's performance and a comparison with both, RIBL, and a propositional k-NN algorithm. In Section 5 we show two ways of improving the learning performance. Section 6 talks about the qualitative, musical side of the results. Future research plans are then discussed in the final Section 7.

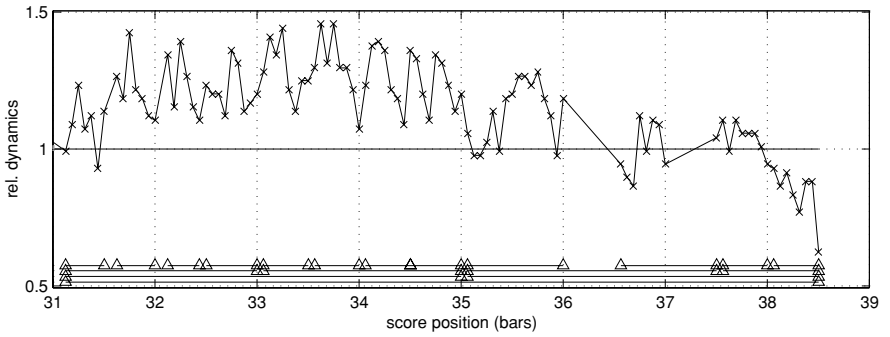

Fig. 1 Dynamics curve (relating to melody notes) of performance of Mozart Sonata KV.279, 1st movement, mm. 31–38, by a Viennese concert pianist

2. Real-world task: Learning to play music expressively

The work presented here is part of a large research project that studies the fundamentals of *expressive music performance* via AI and, in particular, machine learning (Widmer et al., 2003). Expressive music performance is the art of shaping a musical piece by continuously varying important parameters like tempo, loudness, etc. while playing a piece. Instead of playing a piece of music with constant tempo or loudness, (skilled) performers rather speed up at some places, slow down at others, stress certain notes or passages etc. The way this 'should be' done is not specified precisely in the written score,[1] but at the same time it is absolutely essential for the music to sound alive. The aim of this work is learning predictive models of two of the most important expressive parameters: *timing* (tempo variations) and *dynamics* (loudness variations).

The tempo and loudness variations can be represented as curves which quantify the variations of these parameters for each note relative to some reference value (e.g. average loudness or tempo of the same piece). Figure 1 shows a *dynamics curve* of a small part of the Mozart piano Sonata K.279 (C major), 1st movement, as played by a Viennese concert pianist (computed from recordings on a Bösendorfer SE290 computer-monitored grand piano[2]). Each point represents the relative loudness with which a particular melody note was played (relative to an average loudness of the piece); a purely mechanical (unexpressive) rendition of the piece would correspond to a flat horizontal line at $y = 1.0$. Tempo variations can be represented in an analogous way.

A careful examination of the figure reveals some trends in the dynamics curve. For instance, one can notice an up-down, *crescendo-decrescendo* tendency over the presented part of the piece and relatively consistent smaller up-down patterns embedded in it. This is not an accident since we chose to show a part of the piece which is a musically meaningful unit: a high-level *phrase*. This phrase contains a number of lower-level phrases, which are apparently also 'shaped' by the performer. The hierarchical, four-level phrase structure of this passage is indicated by four levels of brackets at the bottom of the figure. The aim of our work is

[1] The *score* is the music as actually printed.

[2] The SE290 is a full concert grand piano with a special mechanism that measures every key and pedal movement with high precision and stores this information in a format similar to MIDI. From these measurements, and from a comparison with the notes in the written score, the tempo and dynamics curves corresponding to the performances can be computed.

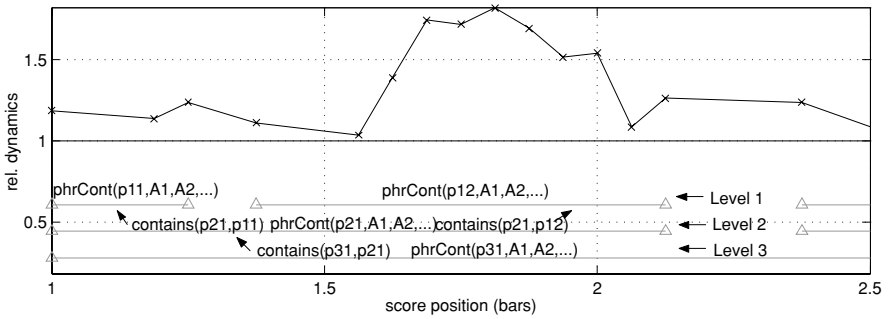

Fig. 2 Phrase representation used by our relational instance-based learning algorithm

the automatic induction of tempo and dynamics strategies, at different levels of the phrase structure, from large amounts of real performances by concert pianists. The heart of our system, the relational instance-based learning algorithm described below, recognizes similar phrases from the training set and applies their expressive patterns to a new (test) piece. In this section we will describe the steps which precede and succeed the actual learning: First we show how hierarchically nested phrases are represented in first-order logic. We then show how complex tempo and dynamics curves as measured in real performances can be decomposed into well-defined training instances for the learner. Finally, we discuss the last step: at prediction time, the shapes predicted by the learner for nested phrases at different levels must be combined into a final performance curve that can be used to produce a computer-generated 'expressive' performance.

2.1. Representing musical phrases in FOL

Phrases are segments of music heard and interpreted as coherent units; they are important structural building blocks of music. Phrases are organized hierarchically: smaller phrases are grouped into higher-level phrases, which are in turn grouped together, constituting a musical context at a higher level of abstraction etc. The phrases and relations between them can be naturally represented in first-order logic.

Consider Fig. 2. It shows the dynamics curve corresponding to a small portion (2.5 bars) of a Mozart sonata performance, along with the piece's underlying phrase structure. For all scores in our data set phrases are organized at three hierarchical levels, based on a manual phrase structure analysis. The musical content of each phrase is encoded in the predicate *phrCont/11*. It has the form *phrCont(Id, A1, A2, . . .)*, where *Id* encodes the phrase identifier and *A1, A2, . . .* are attributes that describe very basic phrase properties. The first seven of the ten attributes are numeric: the length of a phrase, the relative position of the highest melodic point (the 'apex') within a phrase, the melodic interval between starting note and apex, the melodic interval between apex and ending note, metrical strengths of starting note, apex, and ending note. The last three attributes are symbolic: the harmonic progression between start, apex, and end, and two boolean attributes which state whether the phrase ends with a 'cumulative rhythm', and whether it ends with a cadential chord sequence.

Relations between phrases are specified via the predicate *contains(Id1,Id2)*, which states that the bigger phrase *Id1* contains the smaller one *Id2*. Note that smaller phrases (consisting only of a few melody notes) are described in detail by the predicate *phrCont/11*. For the bigger phrases—containing maybe several bars—the high-level attributes in *phrCont/11* are

not sufficient for a full description. But having links to the lower-lever phrases through the *contains/2* predicate and their detailed description in terms of *phrCont/11*, we can also obtain detailed insight into the contents of bigger phrases.

In ILP terms, the description of the musical scores through the predicates *phrCont/11* and *contains/2* defines the background knowledge of the domain. What is still needed in order to learn are the training examples, i.e. for each phrase in the training set, we need to know how it was played by a musician. This information is given in the predicate *phrShape(Id,Coeffs)*, where *Coeffs* encode information about the way the phrase was played by a pianist. This is computed from the tempo and dynamics curves, as described in the following section.

2.2. Deriving the training instances: Multilevel decomposition of performance curves

Given a complex tempo or dynamics curve (see Fig. 1) and the underlying phrase structure, we need to calculate the most likely contribution of each phrase to the overall observed expression curve, i.e., we need to decompose the complex curve into basic expressive phrase 'shapes'. As approximation functions to represent these shapes we decided to use the class of second-degree polynomials (i.e., functions of the form $y = ax^2 + bx + c$), because there is ample evidence from research in musicology that high-level tempo and dynamics are well characterized by quadratic or parabolic functions (Todd, 1992). Decomposing a given expression curve is an iterative process, where each step deals with a specific level of the phrase structure: for each phrase at a given level, we compute the polynomial that best fits the part of the curve that corresponds to this phrase, and 'subtract' the tempo or dynamics deviations 'explained' by the approximation. The curve that remains after this subtraction is then used in the next level of the process. We start with the highest given level of phrasing and move to the lowest. As tempo and dynamics curves are lists of multiplicative factors (relative to a default tempo), 'subtracting' the effects predicted by a fitted curve from an existing curve simply means dividing the y values on the curve by the respective values of the approximation curve.

More formally, let $N_p = \{n_1, \ldots, n_k\}$ be the sequence of melody notes spanned by a phrase p, $O_p = \{onset_p(n_i) : n_i \in N_p\}$ the set (sequence) of relative note positions of these notes within phrase p (on a normalized scale from 0 to 1), and $E_p = \{expr(n_i) : n_i \in N_p\}$ the part of the expression curve (i.e., tempo or dynamics values) associated with these notes. Fitting a second-order polynomial onto E_p then means finding a function $f_p(x) = a^2x + bx + c$ such that

$$D(f_p(x), N_p) = \sum_{n_i \in N_p} [f_p(onset_p(n_i)) - expr(n_i)]^2$$

is minimal.

Given an expression curve (i.e., sequence of tempo or dynamics values) $E_p = \{expr(n_1), \ldots, expr(n_k)\}$ over a phrase p, and an approximation polynomial $f_p(x)$, 'subtracting' the shape predicted by $f_p(x)$ from E_p then means computing the new curve

$$E'_p = \{expr(n_i)/f_p(onset_p(n_i)) : i = 1, \ldots, k\}.$$

Figure 3 illustrates the result of the decomposition process on the last part (mm. 31–38) of the Mozart Sonata K.279, 1st movement, 1st section. The four-level phrase structure our music analyst assigned to the piece is indicated by the four levels of brackets at the

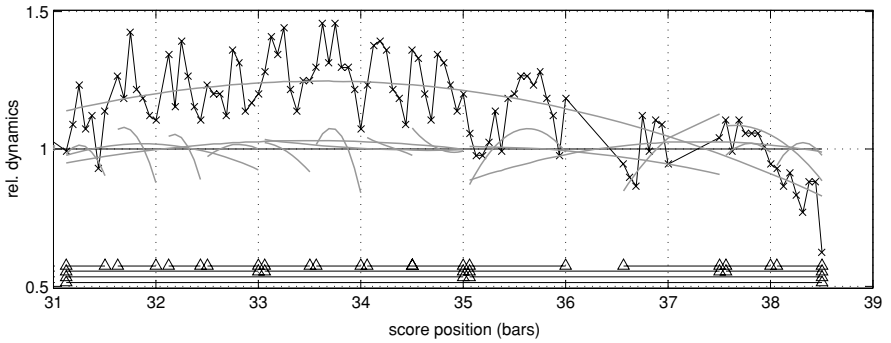

Fig. 3 Multilevel decomposition of dynamics curve of performance of Mozart Sonata K.279:1:1, mm. 31-38.: original dynamics curve plus the second-order polynomial shapes giving the best fit at four levels of phrase structure. The hierarchical phrase structure of this passage is indicated by four levels of brackets at the bottom of the figure

bottom of the plot. The elementary phrase shapes (at four levels of hierarchy) obtained after decomposition are plotted in gray.

We end up with a training example for each phrase in the training set—a predicate *phrShape*(*Id*, *Coeff*), where $Coeff = \{a, b, c\}$ are the coefficients of the polynomial fitted to the part of the performance curve associated with the phrase.

2.3. Combining multi-level phrase predictions

Input to the learning algorithm are the (relational) representation of the musical scores plus the training examples (i.e. timing and dynamics polynomials), for each phrase in the training set. Given a test piece the learner assigns the shape of the most similar phrase from the training set to each phrase in the test piece. In order to produce final tempo and dynamics curves, the shapes predicted for phrases at different levels must be combined. This is simply the inverse of the curve decomposition problem. Given a new piece to produce a performance for, the system starts with an initial 'flat' expression curve (i.e., a list of 1.0 values) and then successively multiplies the current values by the multi-level phrase predictions.

3. Instance-based learning in FOL

This section presents the relational IBL learner DISTALL and briefly contrasts it with its ancestor RIBL (Emde & Wettscherek, 1996), by showing via an example how they implement different notions of structural similarity.

3.1. A structural similarity measure

Before explaining DISTALL in detail, we recall some definitions concerning first order logic and inductive logic programming from (Helft, 1989; Muggleton & De Raedt, 1994).

*Definition 1 (*Linked clause). A clause is linked if all of its variables are linked. A variable v is linked in a clause c if and only if v occurs in the head of c, or there is a literal l in c that contains the variables v and w ($v \neq w$) and w is linked in c.

Example 1. Clause $p(A) \leftarrow r(B)$ is not linked, while $p(A) \leftarrow q(A, B), r(B, C), u(D, C)$ is.

*Definition 2 (*Level of term). The level $l(t)$ of a term t in a linked clause c is 0 if t occurs as an argument in the head of c; and $1 + \min l(s)$ where s and t occur as arguments in the same literal of c.

Example 2. The variable F in $father(F, C) \leftarrow male(F), parent(F, C)$ has level 0, the variable G in $grandfather(F) \leftarrow male(F), parent(F, C), parent(C, G)$ has level 2.

The algorithm to be presented here can be regarded as a generalization of the propositional k-NN for examples described in first-order logic. In FOL, examples are usually represented as sets of ground facts. The heart of a relational IBL algorithm is thus a distance function between sets of elements. A number of distances on sets already exist, e.g. the Hausdorff metric, symmetric difference distances, distances based on relations between sets, etc. For our algorithm, we adopt the distance proposed in Ramon and Bruynooghe (1998); Ramon & Bruynooghe (2000), which is defined via transport networks. The concept of transport networks is used in order to efficiently compute the distance between two sets of elements based on maximal matching. In the following we briefly recapitulate the transport network set distance. For a more formal description (see Ramon and Bruynooghe, 1998).

First, the appropriate transport network is constructed. The network has two groups of vertices $\{a_i\}$ and $\{b_i\}$ corresponding to the elements of the two sets A and B; a starting and an ending vertex (source s and sink t); and two additional vertices, let us call them a_- and b_-. For all edges in the network, *capacities* and *weights* are defined, where capacities represent the maximal amount of 'units' which can 'flow' through a connection and the weights are the distances (transport costs) between particular vertices.

The distance between two sets of elements is then defined as the solution of the maximum flow minimal weight problem: one would like to transport as much as possible from s to t with minimal costs. In other words, one wants to maximally match elements from one set with the elements of the other and achieve the minimal possible distance. By setting the weights of the edges between set elements and the two additional vertices a_- and b_- to a big constant (e.g. bigger than all other edge weights), a 'penalty' is modeled: all elements of one set which do not match with any element of the other cause big costs. By associating appropriate capacities with the edges in the network one can generalize the notion of cardinality in such a way that sets of different cardinality can be scaled appropriately (e.g. allowing the elements of the smaller set to match up more than one element of the other and avoiding that the distance of two sets with vastly different cardinalities is expressed mainly through penalty). An example of a distance network is given in Fig. 4.

Although the maximal matching distance defined via transport networks can be calculated in polynomial time, applying it directly on examples described in FOL (containing maybe hundreds of ground facts) would cause impractically high computational costs. Another problem is that given an example described by many facts, the relevance of particular facts is hard to tune (e.g. by weighting them differently). We avoid these problems by collecting the facts derived from background knowledge into hierarchical *subsets*. By doing so we develop a context-sensitive similarity measure where terms with a lower linkage level to the objects whose distance we are interested in can be made to influence the distance more. In the following we describe our algorithm in more detail. The difference between our approach

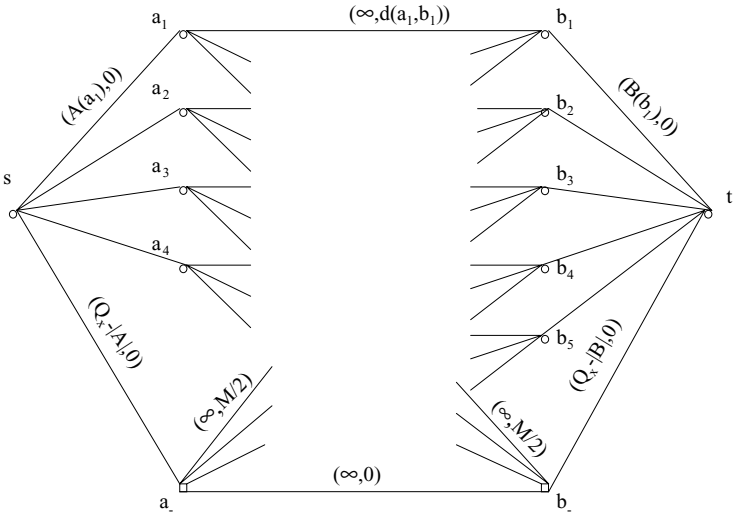

Fig. 4 A distance network (see Ramon & Bruynooghe, 2000)

and the RIBL algorithm, which uses a similar strategy for grouping facts, is illustrated on an example from our learning problem in Section 3.3.

3.2. Computation of similarity in DISTALL

The first step of the algorithm is building so-called starting clauses. The building of starting clauses is a well known method in ILP employed for reasons of computational complexity and implemented in many systems (CLINT (De Raedt, 1992), GOLEM (Muggleton & Feng, 1990), ITOU (Rouveirol, 1992), RIBL (Emde & Wettscherek, 1996)). For each (training and test) example we collect literals that contain terms linked to the example and group them into subsets according to linkage levels. The building of starting clauses is in our case guided by types and modes of literals. Types and modes are used in ILP to restrict the search space. The distance between a test and training example is computed as the set distance between all literals found in the test and training starting clause at linkage level 1. In other words, we find the solution of the maximal flow minimal weight problem, where the transport network vertices are literals containing terms which are directly linked to the examples. The weights of the edges (i.e. distances between individual literals) are computed as the Manhattan distance defined over the literals' arguments (or set to 1 if the literals have different functors). If the arguments are object identifiers, the distance between them is computed by expanding them into a new transport network where the vertices are literals also containing these objects, found one linkage level deeper in the starting clauses. At the lowest level, the distance between objects is calculated as the distance between discrete values.

The *capacities* associated with the edges in the network can be used to control the 'virtual' cardinalities of sets (and, accordingly, the influence of penalty on the set distance). They can also be used to give different importance to the predicates in sets.

The basic principle of the algorithm, let us call it DISTALL (DIstance on SeTs of Appropriate Linkage Level), is illustrated in Fig. 5. In the example, the distance between objects Ob_1 and Ob_2 is calculated as the solution of the maximal flow minimal weight problem on

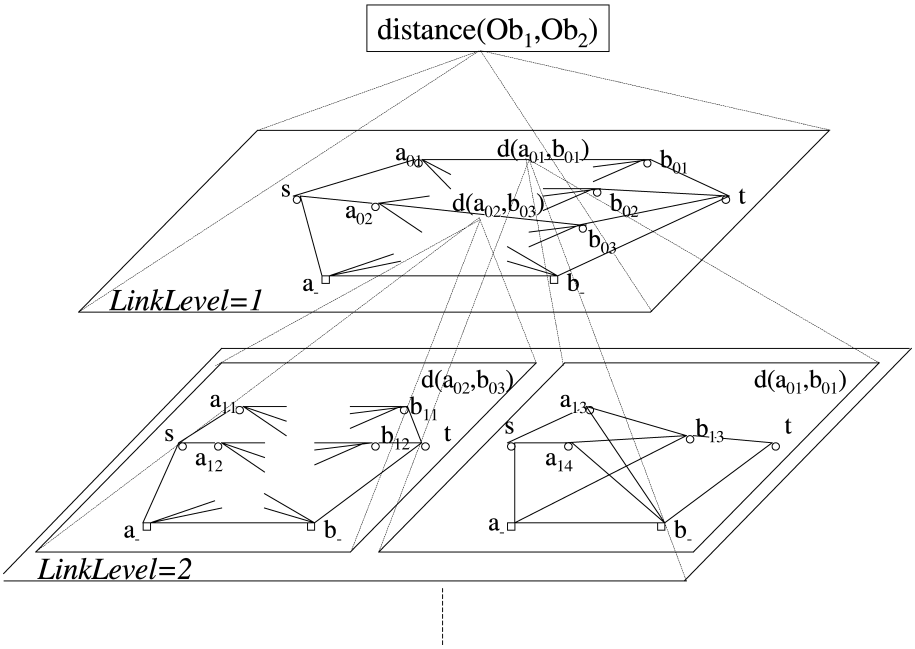

Fig. 5 Basic principle of DISTALL's similarity measure

the sets of literals found at $LinkLevel = 1$ in the starting clauses built for Ob_1 and Ob_2. The weights $d(a_i, b_j)$ of edges connecting literals containing no object identifiers are computed directly (via Manhattan distance, see above). In the example, the literals a_{01} and b_{01} as well as a_{02} and b_{03} have same functors and object identifiers as arguments. The weights of edges between them are thus defined as distance network problems involving the literals containing these objects, found one linkage level deeper in the starting clause. The procedure continues recursively, until the depth of the starting clauses is reached. The computational cost is kept small, since the algorithm solves many hierarchically nested transport network problems with a small number of vertices in each network.

3.3. DISTALL vs. RIBL

DISTALL can be regarded as a continuation of the line of research initiated in Bisson (1992), where a clustering algorithm together with its similarity measure was presented. This work was later improved in Emde and Wettscherek (1996), in the context of the relational instance-based learning algorithm RIBL. The main idea behind RIBL's similarity measure is that the similarity between two objects is determined by the similarity of their attributes and the similarity of the objects related to them. The similarity of the related objects depends in turn on their attributes and related objects.

DISTALL combines this idea with a set distance function based on the notion of transport networks, which was proposed in Ramon and Bruynooghe (1988). While RIBL's similarity measure permits several literals in one example to match the same literal in the other example, DISTALL's netflow distance strongly favors matchings that are as complete as possible and penalizes literals left unmatched. We argue that the so defined distance is more natural in structured domains and works better in practice. First we show the main difference between

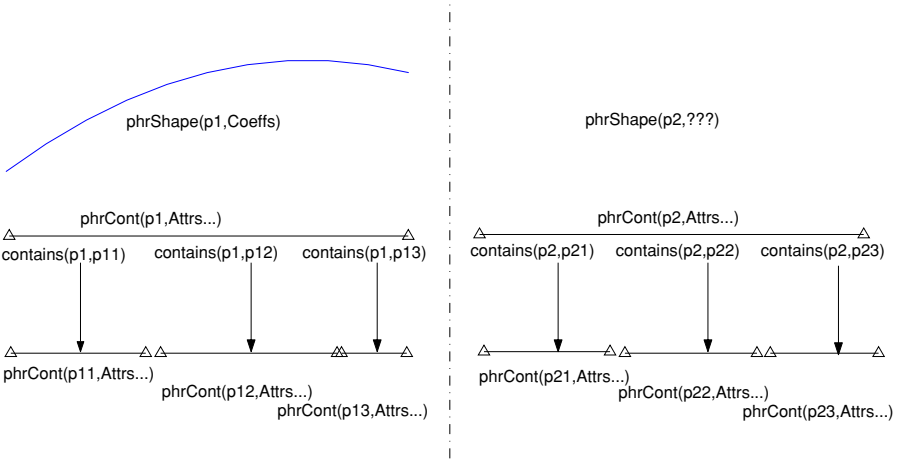

Fig. 6 An example of relational learning situation: training example (left) and new test case (right)

RIBL's and DISTALL's behavior in one constructed situation from our musical application domain. In the next section we present DISTALL's empirical results and provide a direct comparison with RIBL.

Consider the situation given in Fig. 6. We are interested in predicting the 'expressive shape' of the high-level phrase $p2$ and thus want to calculate the distance between phrases $p1$ and $p2$. Each of the two phrases is described via the attributes stored in the phrase-content predicate $phrCont$. They also contain lower-level phrases (predicate $contains$), which are in turn described with their phrase-content predicates. RIBL would compute the similarity between $p1$ and $p2$ as a (weighted) sum of the similarities between the $phrCont$ and $contains$ predicates, where for each $contains$ predicate of $p2$, the most similar $contains(p1, X)$ predicate is found (by finding the most similar $phrCont$ and $contains$ predicate at the lower level). Imagine the situation where the short lower-level phrase $p13$ is a $prototype$ of all lower-level phrases of $p2$ (i.e. the sum of the distances between $p13$ and all $p2x$ phrases is minimal) and the other two lower-level phrases $p11$ and $p12$ are completely different from all phrases $p2x$. By matching all $p2x$ phrases to $p13$ RIBL would obtain a relatively high similarity between $p1$ and $p2$. This is not what we want, as the internal details of the *whole* high-level phrase $p1$ are 'responsible' for the expressive shape and not just a small fraction. In DISTALL, on the other hand, such a matching that leaves two of three subphrases unmatched would receive a high penalty and thus result in a low similarity rating.

4. Experiments

In the following we present detailed empirical results achieved with DISTALL on a complex real-world dataset derived from piano performances of classical music. We also provide a comparison with the results achieved by RIBL as well as with a simple propositional k-NN.

4.1. The data

The data used for the experiments was derived from performances of Mozart piano sonatas by a Viennese concert pianist on a Bösendorfer SE 290 computer-controlled grand piano.

Table 1 Mozart sonata sections used in experiments (to be read as <sonataName>:<movement>:<section>); *notes* refers to 'melody' notes. The number of phrases at each level for a section is also shown

Sonata section		Notes	phr. at level 1	2	3
kv279:1:1	fast 4/4	391	50	19	9
kv279:1:2	fast 4/4	638	79	36	14
kv280:1:1	fast 3/4	406	42	19	12
kv280:1:2	fast 3/4	590	65	34	17
kv280:2:1	slow 6/8	94	23	12	6
kv280:2:2	slow 6/8	154	37	18	8
kv280:3:1	fast 3/8	277	28	19	8
kv280:3:2	fast 3/8	379	40	29	13
kv282:1:1	slow 4/4	165	24	10	5
kv282:1:2	slow 4/4	213	29	12	6
kv282:1:3	slow 4/4	31	4	2	1
kv283:1:1	fast 3/4	379	53	23	10
kv283:1:2	fast 3/4	428	59	32	13
kv283:3:1	fast 3/8	326	52	30	12
kv283:3:2	fast 3/8	558	78	47	19
kv332:2	slow 4/4	477	49	23	12
Total:		5506	712	365	165

A multi-level phrase structure analysis of the musical score was carried out manually by a musicologist. Phrase structure was marked at four hierarchical levels; three of these were finally used in the experiments. The sonatas are divided into sections, which can be regarded as coherent pieces. The resulting set of annotated pieces is summarized in Table 1. The pieces and performances are quite complex and different in character; automatically learning expressive strategies from them is a challenging task.

4.2. A quantitative evaluation of DISTALL

A systematic *leave-one-piece-out* cross-validation experiment was carried out. Each of the 16 sections was once set aside as a test piece, while the remaining 15 pieces were used for learning. DISTALL uses one nearest neighbor for prediction, with the starting clause depth set to 2 (i.e. just those phrases whose relationship order to the phrase in question is ≤ 2 can influence the distance measure).

The expressive shapes for each phrase in a test piece were predicted by DISTALL and then combined into a final tempo and dynamics curve, as described in Section 2.3. The curves predicted by the system were then compared to the *approximation* curves—i.e., the curves implied by the three levels of quadratic functions—of the actual expression curves produced by the pianist. The following performance measures were computed: the *mean squared error* of the system's prediction on the piece relative to the approximation curve ($MSE = \sum_{i=1}^{n} (pred(n_i) - expr(n_i))^2/n$), the *mean absolute error* ($MAE = \sum_{i=1}^{n} |pred(n_i) - expr(n_i)|/n$), and the *correlation* between predicted and approximated curve.[3] MSE and MAE were also computed for a *default* curve that would correspond to a

[3] We could have computed the error of the system's prediction on the piece relative to the curve corresponding to the pianist's actual performance. However, this would be somewhat unfair, since the learner was given not the actual performance curves but an *approximation*, namely the polynomials fitted to the curve at various phrase levels. Correctly predicting these is the best the learner could hope to achieve.

Table 2 Results, by sonata sections, of cross-validation experiment with DISTALL (*depth=2*, *k=1*). Measures subscripted with *D* refer to the 'default' (mechanical, inexpressive) performance, those with *L* to the performance produced by the learner. The cases where DISTALL is better than the default are printed in bold

	Dynamics					Tempo				
	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$
kv279:1:1	.0341	**.0193**	.1571	**.0959**	.7055	.0161	.0223	.0879	.0894	.3147
kv279:1:2	.0282	**.0254**	.1394	**.1168**	.6587	.0106	.0168	.0720	.0796	.4226
kv280:1:1	.0264	**.0117**	.1332	**.0842**	.7704	.0136	**.0066**	.0802	**.0552**	.7730
kv280:1:2	.0240	.0328	.1259	**.1238**	.5285	.0125	.0126	.0793	**.0705**	.5536
kv280:2:1	.1534	**.1227**	.3493	**.2660**	.5308	.0310	**.0082**	.1128	**.0693**	.8685
kv280:2:2	.1405	**.0596**	.3170	**.1892**	.7731	.0323	.0360	.1269	**.1220**	.5383
kv280:3:1	.0293	**.0083**	.1452	**.0705**	.8516	.0188	**.0070**	.0953	**.0558**	.7969
kv280:3:2	.0187	.0224	.1124	**.1057**	.5785	.0196	**.0151**	.1033	**.0822**	.6423
kv282:1:1	.0956	**.0317**	.2519	**.1304**	.8298	.0151	.0187	.0905	**.0724**	.4583
kv282:1:2	.0781	**.0391**	.2277	**.1425**	.7858	.0090	.0246	.0741	.0877	.3981
kv282:1:3	.1047	**.0408**	.2496	**.1624**	.7846	.0938	**.0376**	.2236	**.1337**	.8287
kv283:1:1	.0255	**.0184**	.1379	**.0909**	.7866	.0094	**.0090**	.0664	.0668	.4998
kv283:1:2	.0333	**.0151**	.1560	**.0901**	.7705	.0097	**.0095**	.0691	**.0661**	.5675
kv283:3:1	.0345	**.0092**	.1482	**.0695**	.8883	.0116	**.0076**	.0696	**.0533**	.6857
kv283:3:2	.0371	**.0189**	.1572	**.0951**	.7461	.0100	.0134	.0745	**.0727**	.4700
kv332:2	.0845	**.0674**	.2476	**.2118**	.5119	.0146	.0536	.0718	.1524	.2269
WMean	.0437	**.0279**	.1664	**.1163**	.7001	.0141	.0175	.0811	**.0800**	.5215

purely mechanical, unexpressive performance (i.e., an expression curve consisting of all 1's). That allows us to judge if learning is really better than just doing nothing. The results of the experiment are summarized in Table 2, where each row gives the results obtained on the respective test piece when all others were used for training. The last row ($WMean$) shows the weighted mean performance over all pieces (individual results weighted by the relative length of the pieces).

We are interested in cases where the *relative errors* (i.e., MSE_L/MSE_D and MAE_L/MAE_D) are less than 1.0, that is, where the curves predicted by the learner are closer to the approximation of the pianist's performance than a purely mechanical rendition. In the dynamics dimension, this is the case in 14 out of 16 cases for MSE, and in 16 out of 16 for MAE. The results for tempo are worse: in 8 cases for MSE and 11 for MAE is learning better than no learning.

On some pieces DISTALL is able to predict expressive curves which are surprisingly close to the approximations of the pianist's ones—witness, e.g., the correlation of 0.89 in kv283:3:1 for dynamics.[4] On the other hand, DISTALL performs poorly on some pieces, especially on those that are rather different in character from all other pieces in the training set (e.g. correlation of 0.23 for kv332:2, tempo).

4.3. DISTALL vs. RIBL and propositional *k*-NN

In order to put these results into context, we present a comparison with RIBL (Emde & Wettscherek, 1996). Since RIBL is not publicly available, in the experiments of this section

[4] Such a high correlation between predicted and observed curves is even more surprising taking into account that kv283:3:1 is a fairly long piece with over 90 hierarchically nested phrases containing over 320 melody notes.

Table 3 Comparison between propositional k-NN, RIBL and DISTALL. The table shows weighted mean errors over all test pieces. Measures subscripted with D refer to the 'default' (mechanical, inexpressive) performance, those with L to the performance produced by the learner. All learners use one nearest neighbor for prediction. RIBL's and DISTALL's depth parameter is set to $depth = 2$. All attribute and predicate weights used by the learners (and especially capacities of the distance networks used by DISTALL) are set to 1. The results for DISTALL are repeated from Table 2 (last row). The cases where a learner is better than default are printed in bold

	Dynamics					Tempo				
	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$
prop. k-NN	.0437	**.0335**	.1664	**.1309**	.6369	.0141	.0177	.0811	.0878	.4628
RIBL	.0437	**.0303**	.1664	**.1241**	.6866	.0141	.0215	.0811	.0888	.4765
DISTALL	.0437	**.0279**	.1664	**.1163**	.7001	.0141	.0175	.0811	**.0800**	.5215

we used a self-implemented version. We also compare DISTALL and RIBL, which operate on relational data representation, with the performance of the standard propositional k-NN,[5] since it has been shown that a richer relational representation need not always be a guarantee for better generalization performance (Dzeroski et al., 1998). We can represent phrases in propositional logic by describing each phrase in the data set with the attributes $A1, A2, \ldots$ from the predicate $phrCont(Id, A1, A2, \ldots)$ together with the 'target' polynomial coefficients *Coeffs* from the predicate *phrShape*(*Id*, *Coeffs*). By doing so we lose information about hierarchical relations between phrases and obtain an attribute-value representation which can be used by the k-NN algorithm.

The experimental setup and the error measures stayed the same as in the previous section. All learners use one nearest neighbor for prediction. RIBL's and DISTALL's depth parameter is set to $depth = 2$. All attribute and predicate weights used by the learners, and especially capacities of the distance networks used by DISTALL are set to 1. Table 3 shows the results in terms of weighted mean errors over all test pieces. The equivalent results for DISTALL are repeated from Table 2 (last row). A summary of wins/losses between learning and no learning for all learners is given in Table 4.

Both tables provide evidence of the capabilities of ILP in our domain: Generally, both relational learners outperform the propositional k-NN, having lower errors, higher correlations and higher numbers of wins of learning vs. no learning (the only exception being RIBL's MSE and MAE for tempo, which are higher than those of the propositional k-NN). Additionally, DISTALL outperforms RIBL in terms of all performance measures, being the only learner which has a lower mean MAE for tempo than the mechanical performance. Since DISTALL solves the maximal flow minimal weight problem hierarchically, expanding unknown weights into transport network problems at a lower level, the number of elements in each network—and accordingly, the computational complexity of solving the network distance problem—is kept low, making DISTALL's runtimes only slightly higher than RIBL's (for the presented experiment involving 16 pieces containing about 1240 phrases—about 30 minutes of music—and 16-fold cross-validation, the approximate runtimes on our system are 4 h and 5 h for RIBL and DISTALL, respectively). Certainly, the performance measures expressed in terms of weighted mean numbers given in Table 3 do not imply that DISTALL outperforms RIBL on all pieces in the dataset. It turns out that DISTALL-predicted curves

[5] For a detailed study on the performance that can be achieved by fine-tuning the straightforward propositional k-NN on the same learning problem (see Tobudic & Widmer, 2003a).

Table 4 Summary of wins vs. losses, over all test pieces, between learning and no learning for the propositional k-NN, RIBL and DISTALL

	Dynamics		Tempo	
	MSE	MAE	MSE	MAE
prop. k-NN	12+/4−	14+/2−	6+/10−	7+/9−
RIBL	14+/2−	15+/1−	7+/9−	8+/8−
DISTALL	14+/2−	16+/0−	8+/8−	11+/5−

are closer to the pianist approximations (in terms of MSE and correlation) than those predicted by RIBL in 9 cases for dynamics and 11 cases for tempo (meaning RIBL outperforms DISTALL on 7 pieces for dynamics and 5 for tempo).

5. Two ways of improving learning performance

Although some aspects of the above results were encouraging, e.g. in the dynamics domain, the results for tempo are rather disappointing. Even keeping in mind that artistic performance of difficult music like Mozart piano sonatas is a complex and certainly not entirely predictable phenomenon, DISTALL's inability to learn tempo expression curves which are closer to the performer's in more cases than the mechanical performance in terms of MSE (Table 4) is rather discouraging. In the following sections we explore two ways in which the results can be improved, one of them nicely demonstrating the power of ILP.

5.1. A richer representation of phrases: Statistical features

A certain amount of improvement can be achieved by optimising the feature-based representation of the phrases. The experiments in the previous section were based on the same simple phrase representation that was already used in our previous work with propositional and relational learning algorithms (Tobudic & Widmer, 2003a,b), which makes the results directly comparable. But this is by no means the only possible way of describing musical phrases. More recent experiments indicated that the results can be improved substantially by adding various statistical attributes that capture some global musical characteristics of phrases. Tables 5 and 6 show the results obtained after adding, in addition to some simple information about global tempo and the presence of trills, the following numeric attributes: mean and variance of the durations of the melody notes within a phrase (as rough indicators of the general 'speed' of events and of durational variability), and mean and variance of the sizes of the melodic intervals between the melody notes (as measures of the 'jumpiness'

Table 5 Comparison between propositional k-NN, RIBL and DISTALL. All three learners have access to the extended set of phrase features (see text). The table shows weighted mean errors over all test pieces. The cases where a learner is better than default are printed in bold

	Dynamics					Tempo				
	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$
prop. k-NN	.0437	**.0279**	.1664	**.1210**	.6742	.0141	.0185	.0811	.0870	.4806
RIBL	.0437	**.0289**	.1664	**.1222**	.6740	.0141	.0239	.0811	.0920	.5349
DISTALL	.0437	**.0248**	.1664	**.1116**	.7302	.0141	.0248	.0811	.0891	.5786

Table 6 Summary of wins vs. losses between learning and no learning for the propositional k-NN, RIBL and DISTALL. All three learners have access to the extended set of phrase features (see text)

	Dynamics		Tempo	
	MSE	MAE	MSE	MAE
prop. k-NN	15+/1−	16+/0−	6+/10−	7+/9−
RIBL	15+/1−	15+/1−	8+/8−	9+/7−
DISTALL	15+/1−	15+/1−	12+/4−	13+/3−

of the melodic line). As can be seen, all learners generally benefit from this richer phrase representation, but DISTALL achieve the greatest performance gain in terms of wins/losses in the problematic tempo domain.

5.2. The power of FOL: Temporal relationships

The results of predicting performer's tempo decisions are not satisfying, even with the extended set of phrase features. Actually, in terms of weighted mean errors over all test pieces, the learning performances of both relational learners given the full set of phrase features even degrade in the tempo domain (compare Tables 3 and 5). With the new feature set, no learner is able to produce performances which are better on average than the mechanical one in the tempo domain. We suspect that the main reason for that is predicting of incoherent successive phrases. That is, in a lot of cases, for two successive phrases from the test piece, the learner suggests two phrases from fully different pieces from the training set as being the most similar. These incoherent successive shapes cause 'leaps' in the resulting expressive curves and accordingly higher errors.

This is also counterintuitive from a musical point of view, since it is obvious that the way the performer interprets a phrase strongly depends of the preceding music. On the other hand, 'preparation' for the succeeding phrase also significantly influences the 'shaping' of the current phrase. In other words, an essential aspect of music is its temporal nature, which was not presented to the learner so far.

Supplying the propositional k-NN with such temporal information would be very difficult, if possible at all. On the other hand, FOL allows us to express the temporal relationship between successive phrases naturally. Figure 7 shows the new relational representation of the phrases. The predicate *contains/2* is replaced with the new temporal predicate *succeeds(Id2,Id1)*, which simply states that the phrase *Id2* succeeds the same-level phrase *Id1*. With the new relational predicate, the similarity of two phrases will strongly depend on the

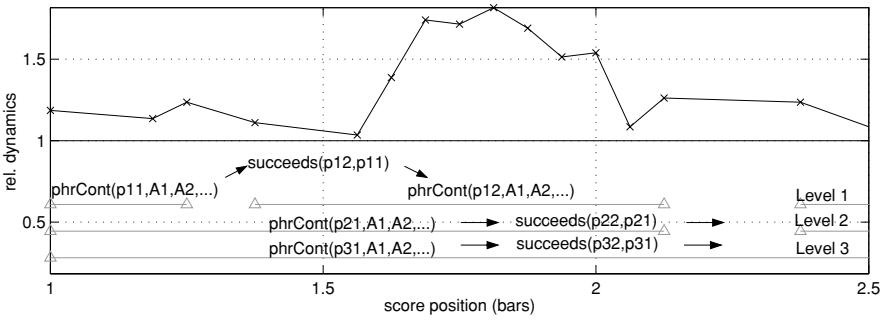

Fig. 7 New phrase representation with the temporal *succeeds/2* predicate

Table 7 Results, by sonata sections, of cross-validation experiment with DISTALL (*depth=4*, *k=1*) and new temporal phrase representation (see Fig. 7). The cases where DISTALL is better than the default are printed in bold

	Dynamics					Tempo				
	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$
kv279:1:1	.0341	**.0178**	.1571	**.0941**	.7147	.0161	**.0125**	.0879	**.0707**	.5800
kv279:1:2	.0282	**.0245**	.1394	**.1099**	.6062	.0106	.0107	.0720	**.0647**	.5977
kv280:1:1	.0264	**.0105**	.1332	**.0789**	.7933	.0136	**.0026**	.0802	**.0418**	.8999
kv280:1:2	.0240	.0255	.1259	**.1119**	.5607	.0125	**.0073**	.0793	**.0609**	.7247
kv280:2:1	.1534	**.0441**	.3493	**.1686**	.8603	.0310	**.0077**	.1128	**.0707**	.8919
kv280:2:2	.1405	**.0564**	.3170	**.1863**	.8044	.0323	**.0178**	.1269	**.0988**	.7427
kv280:3:1	.0293	**.0078**	.1452	**.0699**	.8642	.0188	**.0158**	.0953	**.0724**	.7034
kv280:3:2	.0187	**.0168**	.1124	**.0951**	.6468	.0196	**.0156**	.1033	**.0828**	.6360
kv282:1:1	.0956	**.0246**	.2519	**.1091**	.8665	.0151	**.0088**	.0905	**.0561**	.6583
kv282:1:2	.0781	**.0268**	.2277	**.1205**	.8358	.0090	.0193	.0741	.0829	.4287
kv282:1:3	.1047	**.0332**	.2496	**.1539**	.8290	.0938	**.0867**	.2236	**.2138**	.3125
kv283:1:1	.0255	**.0089**	.1379	**.0695**	.8462	.0094	**.0079**	.0664	**.0590**	.5810
kv283:1:2	.0333	**.0159**	.1560	**.0919**	.7419	.0097	**.0080**	.0691	**.0633**	.5949
kv283:3:1	.0345	**.0093**	.1482	**.0698**	.8869	.0116	**.0063**	.0696	**.0477**	.7123
kv283:3:2	.0371	**.0214**	.1572	**.1018**	.6968	.0100	.0104	.0745	**.0632**	.5430
kv332:2	.0845	**.0612**	.2476	**.1947**	.6403	.0146	.0434	.0718	.1525	.1827
WMean	.0437	**.0233**	.1664	**.1075**	.7229	.0141	**.0135**	.0811	**.0729**	.6066

similarities of their same-level predecessors and successors, which will hopefully produce more coherent performances.

The experiments from previous sections were rerun with DISTALL and the new relational representation. Again, DISTALL used one nearest neighbor for prediction. The starting clause depth was set to 4, i.e. the distance measure for a phrase can be influenced by its four predecessors and four successors. The detailed results are given in Table 7. The experiments with the new representation were also rerun with RIBL (with the same setting, *depth=4*, *k=1*). Table 8 summarizes the results, in terms of weighted mean errors over all test pieces, of the propositional approach, DISTALL and RIBL operating on the relational representation given in the previous section, and DISTALL and RIBL operating on the new temporal phrase representation, denoted as DISTALL*Temp* and RIBL*Temp*, respectively. Table 9 summarizes the results in terms of wins/losses between learning and no learning.

It seems that the temporal predicate *succeeds/2* indeed enables both relational learners to produce accurate results in both domains. However, DISTALL outperforms RIBL in terms of almost all performance measures, especially in terms of wins/losses in the tempo domain. For the first time, DISTALL is able to clearly beat the default performances in terms of weighted MSE and MAE as well as in terms of piecewise wins/losses by tempo. We suspect that such a result in a complex artistic domain would not be possible without the expressive power of FOL.

6. Musical results

A look at Table 7 reveals that a substantial number of pieces predicted by DISTALL*Temp* show high correlations with the expression curves produced by the pianist. Even some predictions

Table 8 Summary of errors between propositional k-NN, DISTALL and RIBL operating on the relational representation given in the previous section and DISTALL and RIBL operating on the new temporal phrase representation, denoted as DISTALL*Temp* and RIBL*Temp*, respectively. The table shows weighted mean errors over all test pieces. The cases where a learner is better than default are printed in bold

	Dynamics					Tempo				
	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$
prop. k-NN	.0437	**.0279**	.1664	**.1210**	.6742	.0141	.0185	.0811	.0870	.4806
RIBL	.0437	**.0289**	.1664	**.1222**	.6740	.0141	.0239	.0811	.0920	.5349
DISTALL	.0437	**.0248**	.1664	**.1116**	.7302	.0141	.0248	.0811	.0891	.5786
RIBL*Temp*	.0437	**.0268**	.1664	**.1177**	.6686	.0141	**.0123**	.0811	**.0743**	.5615
DISTALL*Temp*	.0437	**.0233**	.1664	**.1075**	.7229	.0141	**.0135**	.0811	**.0729**	.6066

Table 9 Summary of wins vs. losses between learning and no learning for the propositional k-NN, DISTALL and RIBL operating on the relational representation given in the previous section and DISTALL and RIBL operating on the new temporal phrase representation, denoted as DISTALL*Temp* and RIBL*Temp*, respectively

	Dynamics		Tempo	
	MSE	MAE	MSE	MAE
prop. k-NN	15+/1−	16+/0−	6+/10−	7+/9−
RIBL	15+/1−	15+/1−	8+/8−	9+/7−
DISTALL	15+/1−	15+/1−	12+/4−	13+/3−
RIBL*Temp*	15+/1−	15+/1−	9+/7−	11+/5−
DISTALL*Temp*	15+/1−	16+/0−	12+/4−	14+/2−

of the slow pieces exhibit astonishingly high correlations in both dimensions (e.g. KV.280, 2nd movement, 1st section, featuring correlations of 0.86 for dynamics and 0.89 for tempo), although the data set contains only six slow pieces.

On the other hand, high correlations and low errors are not a guarantee for good musical quality. Sometimes, relatively small errors at musically sensitive places can seriously compromise the musical quality of the whole piece. Nevertheless some of the performances produced by DISTALL are of substantial musical quality. One of them, Mozart Sonata KV.280, 1st movement, 1st section is shown in Fig. 8. For this performance, the correlations between learner and pianist are 0.79 for dynamics and 0.90 for tempo.[6]

A recording of this performance was submitted to an International Computer Piano Performance Rendering Contest (RENCON'02) in Tokyo in September 2002, where it won second prize behind a rule-based rendering system that had carefully been tuned by hand. The rating was done by a jury of human listeners. While these results do not imply that a machine will ever be able to learn to play music like a human artist, we do consider it a nice success for a machine learning, and in particular, ILP.

[6] Sound examples are accessible at `www.cp.jku.at/people/widmer/mlj_ilp`

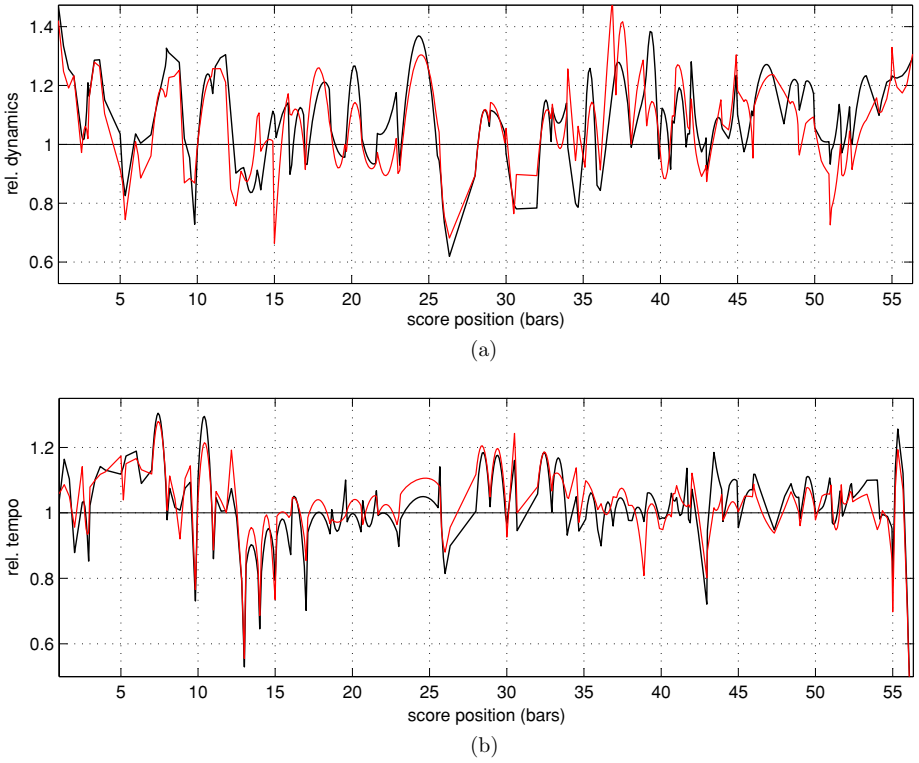

Fig. 8 Expressive curves for dynamics (a) and tempo (b) of the Mozart Sonata KV.280, 1st movement, 1st section. Black curves represent approximations of the actual expression curves produced by the pianist, implied by the three levels of quadratic functions. DISTALL*Temp*'s predictions are given in gray

7. Conclusion

We have presented a complex learning task from the domain of classical music: learning to apply musically 'sensible' tempo and dynamics variations to a piece of music, at different levels of the phrase hierarchy. The problem was modelled as a multi-level decomposition and prediction task and attacked via relational instance-based learning. A new structural similarity measure, based on a combination of two existing techniques, was presented and implemented in a learning algorithm named DISTALL. An experimental analysis showed that the algorithm produces better results in this domain than either the related algorithm RIBL or a propositional *k*-NN learner.

In addition to such quantitative evaluations, listening to the performances produced by the learner provides additional qualitative insight. Some of DISTALL's performances—although being the result of purely automated learning with no additional knowledge about music—sound indeed musically sensible. On the other hand, the musical success achieved is certainly affected by the uniformity of the pieces in the dataset with respect to music style. All pieces are sonatas written in the classical period. It is questionable if the basic strategy of phrase-level IBL is appropriate for pieces written in very different styles. In fact, in that case, the whole notion of phrase similarity would be musically dubious. The approach also relies on

manual phrase structure analysis, which is still far from being automatically feasible.

In Emde and Wettscherek (1996) it was argued that attribute weighting is an important issue in propositional IBL and even more important in a relational setting. DISTALL currently lacks a data-driven predicate/attribute weighting module.

Future work with DISTALL could also have an impact on musicology. The consequently worse results in the tempo domain suggest that other types of approximation functions may be worth trying, which might lead to better phrase-level tempo models. We also plan to try to empirically prove that a concert pianist plays similar phrases in similar ways (by showing a high correlation between expressive shapes for those phrases for which DISTALL suggests high similarity). One could also turn the question around and take the high correlation between expressive shapes attached to phrases which are suggested to have high similarity as a reliability proof of DISTALL's similarity measure.

Acknowledgments This research is supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* (project no. Y99-INF) and the *Vienna Science and Technology Fond (WWTF)* (project 'Interfaces to Music'). The Austrian Research Institute for Artificial Intelligence acknowledges basic financial support by the Austrian Federal Ministry for Education, Science, and Culture, and the Federal Ministry of Transport, Innovation, and Technology. Thanks to Werner Goebl for performing the harmonic and phrase structure analysis of the Mozart sonatas.

References

Bisson, G. (1992). Learning in FOL with a similarity measure. In *Proceedings of the 10th AAAI*, 1992.

De Raedt, L. (1992). *Interactive theory revision: An inductive logic programming approach*. Academic Press.

Duda, R., & Hart, P. (1967). *Pattern classification and scene analysis.* New York, NY: John Wiley & Sons.

Dzeroski, S., Schulze-Kremer, H. K. R., Siems, K., Wettschereck, D., & Blockeel, H. (1998). Diterpene structure elucidation from 13C NMR spectra with inductive logic programming. *Applied Artificial Intelligence: Special Issue on First-Order Knowledge Discovery in Databases*, *12*(5), 363–384.

Emde, D. & Wettschereck, D. (1996). Relational instance-base learning. In *Proceedings of the Thirteen International Conference on Machine Learning (ICML'96)* (pp. 122–130). San Mateo: Morgan Kaufmann.

Helft, N. (1989). Induction as nonmonotonic inference. In *Proceedings of the 1st international Conference on Principles of Knowledge Representation and Reasoning* (pp. 149–156). Kaufmann.

Muggleton, S. H. & Feng, C. (1990). Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*, Tokyo.

Muggleton, S., & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, *19,20*, 629–679.

Ramon, J., & Bruynooghe, M (1998). A framework for defining distances between first-order logic objects. In D. Page (ed.), *Proceedings of the 8th International Conference on Inductive Logic Programming*, vol. 1446 of Lecture Notes in Artificial Intelligence (pp. 271–280). Springer-Verlag.

Ramon, J. & Bruynooghe, M. (2001). A polynomial time computable metric between point sets. *Acta Informatica*, *37*, 765–780.

Rouveirol, C. (1992). Extensions of inversion of resolution applied to theory completion. In S. Muggleton (ed.), *Inductive logic programming*. London: Academic Press.

Sebag, M. & Rouveirol, C. (1997). Tractable induction and classification in FOL via stochastic matching. In *Proceedings of IJCAI-97*. Morgan Kaufmann.

Todd, N. McA. (1992). The dynamics of dynamics: A model of musical expression. *Journal of the Acoustical Society of America*, *91*, 3540–3550.

Tobudic, A. & Widmer, G. (2003a). Playing Mozart phrase by phrase. In *Proceedings of 5th International Conference on Case-Based Reasoning (ICCBR'03)*, Trondheim, Norway. Berlin: Springer Verlag.

Tobudic, A. & Widmer, G. (2003b). Relational IBL in music with a new structural similarity measure. In *Proceeding of 9th International Conference on Inductive Logic Programming (ILP'03)*. Szeged, Hungary. Berlin: Springer Verlag.

Widmer, G. (2002). Machine discoveries: A few simple, robust local expression principles. *Journal of New Music Research*, *31*(1), 37–50.

Widmer, G. (2003). Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence 146*(2), 129–148.

Widmer, G., Dixon, S., Goebl, W., Pampalk, E., & Tobudic, A. (2003). In search of the horowitz factor. *AI Magazine*, 24(3), 111–130.

Widmer, G., & Tobudic, A. (2003). Playing Mozart by analogy. *Journal of New Music Research*, *32*(3), 259–268.