TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Diese Dissertation haben begutachtet:

....

DISSERTATION

Distance Based Learning in a Relational Setting And its Application to Expressive Music Performance

ausgeführt zum Zwecke der
Erlangung des akademischen Grades
Doktor der technischen Wissenschaften

unter der Leitung von

Univ. Prof. Dipl.-Ing. Dr. techn. Gerhard Widmer
Institut für Computational Perception
Johannes Kepler Universität Linz

eingereicht an der

Technischen Universität Wien
Fakultät für Informatik

von

Asmir Tobudic
Matrikelnummer: 9427455
Meiselstrasse 8/4/22, 1150 Wien

Wien, am 11. Februar 2008

Vienna, February 11, 2008.
This manuscript was typeset with LaTeX 2_ε.

Kurzfassung

Diese Arbeit beschreibt Forschungen im Bereich des Machinellen Lernens und deren Anwendungen auf musikwissenschaftlichen Fragen. Es wird ein neuer Lernalgorithmus namens DISTALL präsentiert sowie, darauf aufbauend, ein automatisches Lernsystem, das ein schwer fassbares Phänomen in der Musik analysieren und modellieren soll, nämlich ausdrucksvolle Musikinterpretation.

Der Lernalgorithmus DISTALL ist im Bereich des relationalen 'Instance-based Learning' (IBL) angesiedelt. Obwohl distanz- und instanzbasierte Lernalgorithmen im Bereich des Maschinellen Lernens immer sehr beliebt waren – vor allem mit propositionalen, attributbasierten Repräsentationssprachen –, ist IBL in mächtigeren relationalen (auf Prädikatenlogik basierenden) Repräsentationen wesentlich schwieriger und weniger erforscht. In der Dissertation werden diese beiden Lernformalismen diskutiert, Vorteile des relationalen Lernens aufgezeigt und die These aufgestellt, dass der kritischste Teil eines relationalen IBL-Lernalgorithmus sein Ähnlichkeitsmaß (zwischen Mengen von Termen) ist. Verschiedene Mengen- Ähnlichkeitsmaße werden rekapituliert, und es wird der Schluss gezogen, dass ein Ähnlichkeitsmaß, das auf optimalem Matching zwischen Mengen basiert, aus zwei wesentlichen Gründen am vielversprechendsten ist: (1) wegen seiner intuitiv überzeugenden und nachvollziehbaren Aspekte und (2) wegen seine klaren theoretischen Eigenschaften. Auf einem solchen Ähnlichkeitsmaß aufbauend wird DISTALL implementiert und im Detail beschrieben.

DISTALL wird sodann auf ein schwieriges Lernproblem aus dem Forschungsgebiet der Musikwissenschaft angewendet: Ausgehend von einer großen Zahl von Interpretationen (Aufnahmen) von Konzertpianisten soll der Computer lernen, Musik ausdrucksvoll zu spielen. Die Aufgabe wird als Mehrebenen-Dekompositions- und Vorhersage-Problem modelliert, und es wird gezeigt, dass dieses als relationales Lernproblem darstellbar ist und mittels relationalem IBL bewältigt werden kann. Experimente mit realen Daten, die aus einer beachtlichen Anzahl von Interpretationen eines Wiener Konzertpianisten gewonnen wurden, deuten die Brauchbarkeit unserer Methode an. Speziell wird gezeigt, dass die Vorhersagegenauigkeit von DISTALL die eines herkömmlichen propositionalen k-NN-Algorithmus übertrifft. Direkte experimentellen Vergleiche mit RIBL, einem modernen relationalen Lernalgorithmus, demonstrieren auch die klare Überlegenheit von DISTALL gegenüber RIBL bei dieser Lernaufgabe. Verschiedene weitere Verbesserungen des Lernsystem werden vorgestellt. Eine der Verbesserungen – die explizite Darstellung zeitlicher Beziehun-

gen – demonstriert deutlich die Mächtigkeit des relationalen Lernformalismus. In qualitativer Hinsicht stellt sich heraus, dass unser Lernsystem zumindest zum Teil erstaunlich gute Vorhersagen macht. Einige nach dem Lernen vom Konzertpianisten von DISTALL generierte 'Aufführungen' weisen erhebliche musikalische Qualität auf: eine davon gewann sogar einen Preis bei einem internationalen 'Computer Music Performance'-Wettbewerb.

Zwei weitere Anwendungen von DISTALL und seinem Ähnlichkeitsmaß werden sodann in der Dissertation vorgestellt: (1) Wir versuchen festzustellen, mit welchem Niveau an stilistischer 'Konsistenz' ein Wiener Konzertpianst verschiedene Mozartsonaten spielt. Mit Hilfe des Ähnlichkeitmaßes von DISTALL kann ein Konzept stilistischer Übereinstimmung definiert werden, das über einfache Notentext- Wiederholungen hinausgeht, und darauf aufbauend wird ein quantitatives Maß von Aufführungs-Übereinstimmung zwischen beliebigen Musikphrasen realisiert, das systematische quantitative Experimente zulässt. (2) Wir studieren eine der interessantesten Probleme, das in dieser Art von Forschung formuliert werden kann: Kann eine Maschine ein formales, prädiktives Modell des Spielstils eines berühmten Pianisten lernen? Wir erkunden, inwieweit die Maschine 'expressive Profile' großer Pianisten automatisch bilden kann, nur mit Hilfe von aus Audio-CDs gewonnenen Minimalinformationen und des Notentextes der gespielten Musik. Es stellt sich heraus, dass das auf DISTALL basierende Lernsystem tatsächlich in der Lage ist, 'ausdrucksvolle' Interpretationen neuer Musikstücke zu generieren, die zur echten Interpretation des 'Trainingspianisten' deutlich ähnlicher sind als zu den Interpretationen aller anderen Pianisten. Eine weitere interessante Anwendung unseres Lernalgorithmus wird schlussendlich noch besprochen: die automatische Erkennung berühmter Pianisten anhand ihres Spielstils. Wie die Experimente zeigen, sind auch bei diesem schwierigen Problem erstaunlich gute Resultate möglich.

Abstract

This thesis reports about work in the field of machine learning and its applications in musicology. We present a new machine learning algorithm called DISTALL and describe an automated learning system targeting one of the most elusive phenomena in music: to learn to play music expressively.

DISTALL is situated in the field of relational instance-based learning (IBL). Although distance- and instance-based learning methods have always been popular in machine learning, IBL in a richer, relational setting is more difficult and has been less explored. We contrast IBL in a propositional setting to relational IBL, discuss advantages of relational representations, and argue that the most critical part of a relational IBL is its set distance measure. After different set distance measures are discussed, we conclude that the set distance measure based on optimal matching is the most appealing for both its intuitive properties and strong theoretical aspects. We construct DISTALL, our new relational IBL algorithm around the optimal matching set distance measure and discuss its algorithmic implementation in detail.

DISTALL is applied to a difficult real-world learning task from expressive music performance research: 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 argue that relational IBL is indeed appropriate to address it. 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 DISTALL operating on structured, relational data outperforms a propositional k-NN algorithm. Experiments with a direct comparison to the state of the art relational learner RIBL clearly show DISTALL's superiority to RIBL on this learning task. Various improvements to the learning system are proposed, one of them – temporal representation – nicely demonstrating the power of relational formalisms. In qualitative terms, we end up with a system which at least partly makes surprisingly good predictions. Some of the piano performances produced by DISTALL after learning from human artist are of substantial musical quality; one even won a prize in an international 'computer music performance' contest.

Two further applications of DISTALL and its distance measure are presented: (1) We will try to assess the level of 'consistency' of a Viennese concert pianist in playing different Mozart sonatas. With the help of DISTALL's similarity measure we are able

to define a concept of consistency which goes beyond simple score repetitions. The level of performer consistency will be assessed between any tho phrases, regardless of similarity /dissimilarity of the pieces they belong to. (2) We address one of the most interesting questions one can consider in this kind of research: Can a machine build a formal model of the playing style of great pianists? We investigate to what extent a machine can automatically build 'expressive profiles' of famous pianists using only minimal performance information extracted from audio CD recordings by these pianists and the printed score of the music. It turns out that the learning system built around DISTALL is able to generate expressive performances on unseen pieces which are substantially closer to the real performances of the 'trainer' pianist than those of all others. Finally, another interesting application is discussed: recognizing pianists from their style of playing - a difficult learning problem tackled in the recent literature. We show that surprisingly high accuracy rates can be achieved by using expressive performance profiles predicted by DISTALL for artist recognition.

Acknowledgements

This work was carried out within the framework of a large-scale research project "Computer-Based Music Research: Artificial Intelligence Models of Musical Expression" at the Austrian Research Institute for Artificial Intelligence (Österreichisches Forschungsinstitut für Artificial Intelligence, ÖFAI), Vienna. This project has been financed through the START programme from the Austrian Federal Ministry for Education, Science, and Culture (Grant No. Y99–INF) in form of a generous research prize to Gerhard Widmer (`http://www.ofai.at/research/impml`). The ÖFAI acknowledges basic financial support from the Austrian Federal Ministry for Education, Science, and Culture and the Austrian Ministry for Transport, Innovation and Technology. Parts of this work were additionally financed through the financial support from the *Vianna Science and Technology Fond (WWTF)* (project 'Interfaces to Music').

I want to thank Gerhard Widmer, the head of the ÖFAI music group and my supervisor, for his engaging and never fatiguing spirit while pursuing novel scientific ideas of AI methods for exploring music expression, and even more for promoting the work of each of the members of our young research group. It is solely his merit that the interesting problems and applications of music expression have recently been established in the mainly technically oriented AI community. I owe him thanks for the opportunity I had: To learn and gain knowledge in scientific fields of AI and machine learning, in my mind seminal and surely future-oriented technical fields, while at the same time working on interesting and uncommon problems regarding music. I would like to thank Georg Dorffner, the head of the ÖFAI neural computation and robotics group and my second supervisor, for his careful proofreading and insightful suggestions regarding the presented work. I am grateful to all my colleagues at the ÖFAI music group: Simon Dixon, for his advices, insights, and his teaching in scientific thinking, Werner Goebl, especially for performing the harmonic and phrase structure analysis of the Mozart sonatas and for his friendly advices regarding my ignorance of musicology in general, Elias Pampalk, and Søren Tjagvad Madsen for making overall friendly and pleasant atmosphere. Special thanks to Robert Trappl, the head of ÖFAI for his patience in recruiting research money and allowing the young researcher at ÖFAI to concentrate on more interesting scientific work.

I would like to express my thanks to all my friends and family, who were supporting me during all the time through my work.

The work on this thesis took much more than it should have. After stopping

working at ÖFAI, the struggle of everyday life and my business career seriously put the finishing of the thesis in question. All the time however, there were three persons in the world which contribution to this thesis is so huge that the authorship can easily be transferred to.

The first person is my wife. Her smile, endless patience and understanding, even when no one in the world would reasonably understand me, made this work possible. I just have no idea why I am so lucky to be with you. The second and third persons are my mom and dad. Once during my childhood, my parents told me that one day I may be graceful for their suggestions regarding the importance of education. That day came already long ago. Thank you for being with me all the time, during last thirty years of my education.

Zahvalnica

Ovaj rad je dio dugogodisnjeg naucnog projekta 'Computer-basirano Istrazivanje u Musici: Metode Umjetne Inteligencije za Istrazivanje Musickog Izrazavanja' ("Computer-Based Music Research: Artificial Intelligence Models of Musical Expression"), koji se odvija na Austrijskom Naucnom Institutu za Umjetnu Inteligenciju (Österreichisches Forschungsinstitut für Artificial Intelligence, ÖFAI), Bec, Austrija. Projekt je financiran kroz START program Austrijskog Federalnog Ministerija za Edukaciju, Nauku, i Kulturu (projekt broj Y99–INF), u formi naucne nagrade dodijeljene Gerhardu Widmer-u (web adresa naucne grupe na institutu je `http://www.ofai.at/research/impml`). ÖFAI prima takode i osnovnu financijsku podrsku od Austrijskog Federalnog Ministarstva za Edukaciju, Nauku, i Kulturu, kao i od Austrijskog Ministarstva za Transport, Inovaciju, i Tehnologiju. Dijelovi ovog rada su dodatno financirani kroz financijsku podrsku grada Beca, Beckog Fonda za Nauku i Technologiju (*Vianna Science and Technology Fond (WWTF)*), kroz projekt 'Interfaces to Music'.

Kao prvo zelim da se zahvalim Gerhardu Widmer-u, vodi muzicke grupe na ÖFAI institutu i mom supervizoru, na njegovom uzornom i nikad posustajucem duhu u stalnom iznalazenju novih naucnih ideja primjene umjetne inteligencije za istrazivanje muzicke ekspresije, i jos vise za promovisanje radova svakog od clanova nase mlade umjetnicke grupe. To se moze svrstati u iskljucivo njegovu zaslugu, da su interesatni problemi i aplikacije u vezi sa muzickom ekspresijom u zadnje vrijeme nasli svoje cvrsto mjesto u pretezno tehnicki orijentisanoj nauci umjetne inteligencije. Ja mu dugujem hvala za priliku koju sam imao: Da skupljam znanja u naucnim oblastima umjetne inteligencije i masinskog ucenja – po mom misljenju kolosalnim, i sigurno ka buducnosti usmjerenim naucnim disciplinama –, dok istovremeno radeci na interesantnim i nesvakidasnjim problemima iz oblasti muzicke umjetnosti. Zelim da zahvalim i Georg-u Dorffner-u, vodi grupe za neuralnu informatiku i robotiku na ÖFAI institutu i mom drugom supervizoru, za pazljivo citanje i brojne korisne prijedloge u vezi moga rada. Zahvalan sam takode i svim mojim kolegama u muzickoj grupi ÖFAI instituta: Simon Dixon-u, za njegove savjete, uvide, i poducavanje u naucnom razmisljanju, Werner Goebl-u, posebno za manuelno analiziranje fraza i harmonije Mozart sonata koje su mi bile potrebne za rad, kao i za njegova prijateljska savjetovanja u vezi mog neznaja iz muzikologije, Elias Pampalk-u, i Søren Tjagvad Madsen-u, za pravljenje prijateljske i ugodne radne atmosfere. Specijalno hvala i Robert Trappl-u, vodi ÖFAI instituta, za njegovo strpljenje i neumorno

istrazivanje svih mogucnosti za financijelnu potporu, na taj nacin dozvoljavajuci mladim naucnicima na ÖFAI-u da se koncentrisu na mnogo interesantniji naucni rad.

Zelio bih da izrazim svoju zahvalnost svim mojim prijateljima i familiji, koji su me podrzavali za cijelo ovo vrijeme kroz moj rad.

A rad na ovom doktoratu je trajao mnogo duze nego sto je trebao. Nakon sto sam prestao raditi na ÖFAI institutu, obaveze u svakodnevnici i moje poslovna karijera su bili ozbiljno ugrozili privodenje kraju ove doktorske teze. Svo vrijeme medutim su postojale tri osobe cije su zasluge za njegov zavrsetak bile toliko velike, da bi im se slobodno moglo prenijeti pravo na autorstvo ovoga rada.

Prva osoba je moja zena. Njen smijeh, neizmjerno strpljenje i razumjevanje, cak i onda kada me logicki niko na svijetu nebi mogao razumjeti, su napravili zavrsetak ovog rada mogucim. Ja samo nemam pojma zasto sam toliko sretan da mogu biti sa tobom. Druga i treca osoba su moja mama i tata. Jednom prilikom za vrijeme mog djetinjstva, moji roditelji su mi rekli da cu jednog dana mozda biti zahvalan na njihovim sugestijama u vezi vrijednosti koje ima skupljanje znanja. Taj dan je vec davno dosao. Hvala vam sto ste bili sa mnom cijelo ovo vrijeme, u zadnje trideset tri godine mog obrazovanja.

Contents

Chapter 1

Introduction

1.1 Artificial Intelligence and Machine Learning

Very few sciences have such a controversial name as *Artificial Intelligence*. It is not at least due to the fact that the name comprises the term *intelligence*, which is on its own very hard to define. The difficulties are reflected by the fact that many different definitions of the field of artificial intelligence exist. In order to apply the technical point of view, and avoiding somewhat philosophical discussion, for the purposes of this introduction we will adopt the following nearly trivial definition: Artificial Intelligence as a scientific field is about creating machines that perform functions that require intelligence when performed by people [Russell and Norvig, 2003]. A nice overview of several definitions, classified along several dimensions is given in [Russell and Norvig, 2003].

While it is difficult to say what intelligence is exactly, some aspects of being intelligent can be identified relatively easily: ability to reason, to be creative, to find solutions to non-trivial problems, etc. One of the central aspects of being intelligent is the ability *to learn* from experience. Again, there are several definitions of learning. According to [Langley, 1995], learning is the improvement in some environment through the acquisition of knowledge resulting from experience in that environment.

Machine learning is a subfield of Artificial Intelligence. Given the definition of learning, it can be (trivially) defined as a study of how to make machines learn. Technically, a machine learning 'environment' consists of a learning task and a set of examples. In order to assess the system's learning abilities, one also defines the *performance measure* on the given learning task and then measures the improvement produced by the learning ([Mitchell, 1997]). In other words, learning can be technically defined as seeking to reduce an error (given the performance measure) on the learning task ([Mitchell, 1997]).

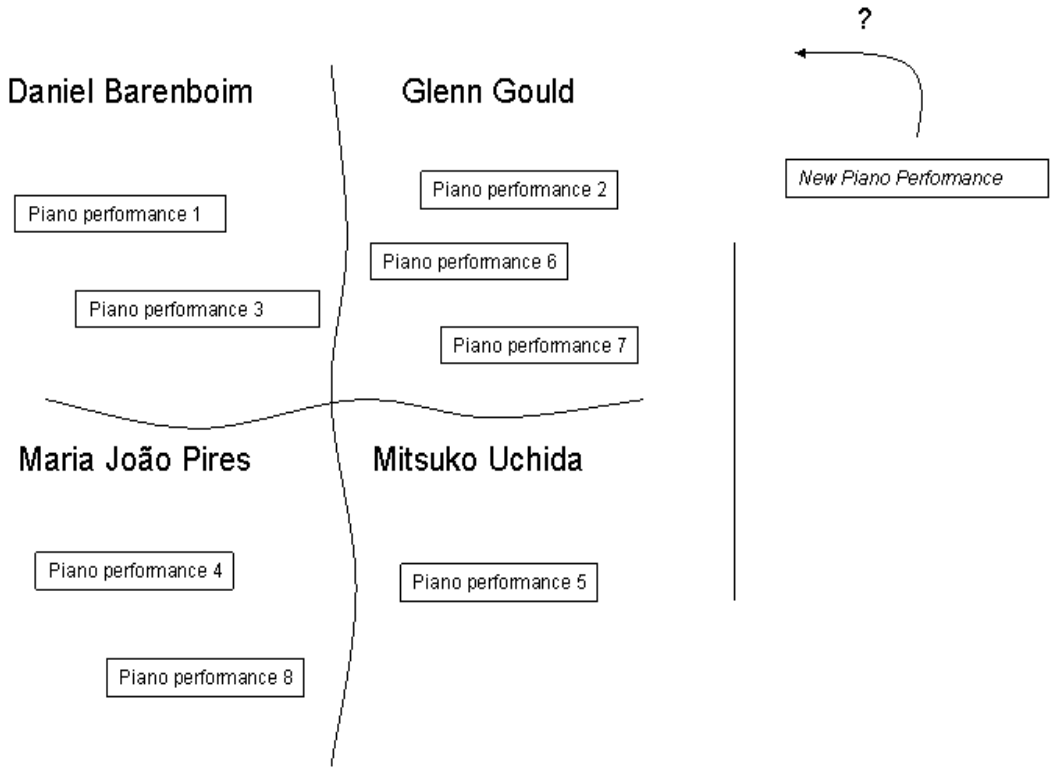

Figure 1.1:

1.2 Supervised vs. Unsupervised Learning

Machine learning methods can be divided along several dimensions. The most general classification is into supervised and unsupervised learning.

In *supervised learning* setting, every learning example has a *target value* attached to it. The learning program is given a set of examples together with the information about their target values. These examples are called the *training set*. The task of the system is to predict the target values of *unseen examples*.

E.g., consider Figure 1.1. In this example, the system is given eight performances of four great pianists. Along with the performances, the system is given the information which performance is produced by which pianist. After learning, the task of the system is to classify a new, unseen performance. Throughout this text we are mainly concerned with supervised learning.

Another important class of learning systems is *unsupervised learning*. In this setting, the learner operates with examples which – in the context of the learning task – do not have explicit classes. The task of the system is then to form 'natural groupings' of the examples. Although it might at the first sight not be obvious how

unsupervised learners can learn something meaningful at all, they are for various reasons indeed very important for the field of machine learning. To name just a few: Collecting and labeling a large set of training patterns can be costly. If a learner can be designed on a small set of labeled samples and then scaled up to run without supervision on a large unlabeled set, much time can be saved. One can also proceed another way around ([Duda *et al.*, 2000]): train with large amounts of unlabeled data, and then label the groupings found. This is actually the primary reason for importance of unsupervised learning methods and is closely related to *data mining and knowledge discovery* ([Adriaans and Zantinge, 1997; Witten and Frank, 2000]), the scientific fields that recently became very popular in both the scientific and the industrial communities. Unsupervised methods can also be used to find *features* with high 'information gain'. These 'right features' can then be used for classification. Indeed, *feature selection* – choosing the subset of features which allows the best generalization performance given the limited number of training examples – is one of the most difficult problems in machine learning, especially in domains with very high *dimensionalities* (e.g. including various important tasks in the field of bio-informatics, such as learning protein structures and functions). For an excellent overview of unsupervised learning methods see [Duda *et al.*, 2000].

1.3 Distance-based Learning Methods

Besides the classification into supervised/unsupervised approaches, machine learning methods can be divided along several different dimensions. E.g. there is in principle a difference between methods which are governed by the laws of *probability calculus* and those governed by the laws of *logic*. We also distinguish between systems operating basically on numerical vs. symbolic data representations, systems building a model that can describe the input data (*descriptive models*), and those which rather 'skip' directly to mapping inputs to the correct answer (*discriminative models*), etc.

An important class of learning methods are those which make use of a *similarity* or *distance measure* in some form. These learning approaches are generally called distance-based learning methods. Distance-based learning typically relies upon two subtasks. First, one has to define an appropriate distance measure for the learning task. Then, on top of the distance measure the algorithm can be constructed in such a way that it makes optimal use of the provided distance measure.

Instance-based learning is a supervised machine learning method which makes explicit use of the defined distance function. The most basic instance-based learning algorithm is *k-nearest neighbor* ([Duda *et al.*, 2000]). The basic variant of the algorithm predicts the class of an unseen example as the (weighted) majority class of the k nearest training examples. For the purposes of the research presented here, we will use the (variants of) of the k-NN.

Yet another successful supervised instance-based learning algorithm is *locally*

weighted regression (see e.g. [Mitchell, 1997]). It is a method for the problem of approximating real-valued functions. It uses distance-weighted training examples to form an explicit approximation to the target function over a local region surrounding an unseen example. The approximation in the neighborhood surrounding an unseen example might be a linear function, a quadratic function or take some other functional form.

Clustering is an unsupervised method which often relies upon some explicit distance measure. Although we will not use it in this work, it is a very important technique in machine learning. The task is to partition a set of examples into clusters such that examples in the same cluster are as similar as possible and examples in different clusters are as different as possible, according to the defined distance measure.

While distance-based method such as instance-based learning and clustering are very popular and well-investigated in the standard attribute-value representation, the application of these methods is more difficult and less explored for more expressive representation languages. The field of *inductive logic programming* (ILP) ([Nienhuys-Cheng *et al.*, 1997; Muggleton and Raedt, 1994] is a subfield of machine learning which is concerned with learning in a richer *first-order logic* (FOL) or *relational* setting. It is related to the work presented here, since in this text we will mainly use relational formalisms. Yet in presenting our novel learning algorithm, we will be mainly concerned with defining a *distance measure* in a relational setting and applying a straightforward k-NN algorithm for learning. As such, the work is situated in the field of *relational instance-based learning.* Somewhat related is also the field of *case-based reasoning* (CBR).

While the distance measure can be defined explicitly – on the space of examples, it can be also represented implicitly. *Kernel* methods and learning algorithms based on them, such as e.g. *support vector machines* ([Cristianini and Shawe-Taylor, 2000]), are also related to distance-based learning, since they make use of distance functions defined implicitly in a higher dimensional space. Although not directly related to our work, we mention them for the sake of completeness.

1.4 Music Performance Research and Expressive Music Performance

According to Seashore, a pioneer of musical performance research, music generally relies on

> 'artistic deviation from the fixed and regular: from rigid pitch, uniform intensity, fixed rhythm, pure tone and perfect harmony' [Seashore, 1938].

Music performance is the act of interpreting, structuring and physically realising a piece of music. The fact that it is indisputably a complex human activity

– involving physical, acoustic, physiological, psychological, social, and artistic aspects [Widmer and Goebl, 2003] – is reflected by the broad range of subfields that music performance research involves. As discussed in [Gabrielsson, 1999, 2003], the broad range of topics covered by music performance research include (listed in chronological order): Performance planning, sight reading, improvisation, feedback in performance, measurements of performance, models of music performance, physical factors in performance, psychological and social factors, and performance evaluation.

This text will focus on applying machine learning methods to one specific aspect of music performance, namely, building *computational models* of *expressive* music performance. Expressive music performance is the art of shaping a musical piece by continuously varying important parameters like tempo, loudness, intonation, articulation, tone envelope, timbre, etc., while playing (or singing) a piece of music. In this work we will concentrate on piano music and two of the most important expression parameters: tempo and loudness (dynamics). 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 but at the same time it is absolutely essential for the music to sound alive.

From the point of view of musicology, this thesis can be regarded as a continuation of the line of research pursued by our research group in Vienna over the past years (see e.g. [Widmer, 1995a,b, 1996, 2000, 2002; Widmer and Tobudic, 2003; Widmer *et al.*, 2003]). The work pursued by our research group is to be regarded as basic research: Our intention is not to engineer computer programs that generate music performances that sound as human-like as possible. Rather, we investigate to what extent a machine can automatically build, via inductive learning from 'real-world' data, operational models of certain aspect of performance (e.g. predictive models of tempo and dynamics in the case of this thesis). By analysing the models induced by the machine, we hope to get new insight into fundamental principles underlying the complex phenomenon of expressive music performance, and in this way contribute to the growing body of scientific knowledge in this area.

1.5 Outline

This thesis has two main contributions. The first central contribution is a proposal for a new distance measure which can be applied to various learning tasks represented in *relational formalisms*. While distance-based learning on its own is a very broadly explored subfield of machine learning, instance- and distance-based learning approaches in a relational setting are more difficult and much less explored. On top of our distance measure, any distance-based algorithm – either supervised, such as e.g. *k-nearest neighbor* or unsupervised, e.g. *k-means clustering* – can be constructed.

The second contribution of the thesis is the introduction of a difficult, real-world learning task from expressive performance music research. We aim at automatically inducing expressive models of certain aspects of expressive piano performance, namely models of expressive *tempo* (i.e. timing) and *dynamics* deviations. The starting material for automatically inducing such models will be a relative large amount of empirical 'expressive performance' data – precisely measured performances by skilled musicians. We will then introduce a methodology for decomposing this rather 'unstructured' performance data into well-defined training examples, which are then used as an input to our relational instance-based learner. The experimental sections of the thesis discuss various tasks from music research for which our learning approach can be applied, involving predicting the 'courses' of expressive tempo and dynamics curves on unseen pieces, examining the degree of 'consistency' of the same pianist playing different musical pieces, recognizing great pianists from their style of playing and even learning to 'play' a piece of music like the great pianists.

Since instance-based learning is a central learning framework used throughout this thesis, we will review it in chapter 2. After introduction in section 2.1, we review the basic concepts regarding k-nearest neighbor (section 2.2). We will also discuss the notions of distance and metric in general in section 2.3. First-order logic as a representation formalism is presented in section 2.4. This section also contains a discussion how learning examples in FOL are most commonly represented in practical learning systems. It turns out that *sets of elements* are an important concept for representing learning examples in practical relational learning systems, and thus distance measures between sets of elements are of central interest for our work. In section 2.5 we give an in-depth overview of existing set distance measures, one of them being the main part of our learning algorithm.

Chapter 3 gives detailed description of our relational instance-based learning algorithm DISTALL. We introduce our motivation in designing DISTALL in section 3.1. In section 3.2 we explain the representation of the learner input. Sections 3.3 and 3.4 are concerned with the learner's central aspects from the technical perspective: the mechanism for set formation given the learning input and efficient computation of the set-based optimal matching distance. In section 3.5 we put all pieces together and give the detailed description of our structural similarity measure which is then built in the new relational instance-based learner DISTALL. In section 3.6 we contrast DISTALL with its ancestor RIBL, by showing via an example how they implement different notions of structural similarity.

In Chapter 4 we introduce a learning task from expressive piano performance research. After short introduction in section 4.1, we recapitulate the related previous work undertaken in our research group: induction of note-level performance rules with the PLCG algorithm (section 4.2). This thesis can be considered as a continuation of that research. In section 4.3 we then present the main learning task which will be used for evaluation of our algorithm: Learning to play music expressively. In this section we also review the concept of expressive performance curves and present a method for converting these 'unstructured' curves into well-defined

training examples, which in turn will be used as input for relational instance-based learning. The overall learning system used for experiments is presented in section 4.4.

Chapter 5 is dedicated to the main bundle of experiments we have conducted during evaluation of our approach. In section 5.1 we present the experiments with a standard propositional k-NN. Besides making the reader familiar with our dataset, training and evaluation procedures, it also gives a first impression of how difficult the learning problem is. We also discuss various ways in which the results can be improved, resulting in a system which even won a prize in a recent 'computer music performance' contest in Tokyo. Section 5.2 shows the experimental results on the same learning problem achieved with our new relational instance-based learning algorithm and contrasts it with both standard propositional k-NN, and related learner RIBL. Comparing our new learning algorithm with the state of the art relational instance-based learner RIBL on the difficult real-world learning task illuminates this section as the most interesting part of this thesis from the technical, machine learning point of view. In section 5.3 we recapitulate the most interesting results and conclusions from our experiments.

Chapter 6 presents a further interesting application of our similarity measure: we will try to assess the level of 'consistency' of a Viennese concert pianist in playing different pieces from a corpus of Mozart's piano sonatas from the classical period. After introducing the problem in section 6.1, in section 6.2 we present two similarity measures which will help us assess the level of consistency in piano performances: score- and performance-based similarity. The data and the procedure for experiments is presented in section 6.3 and results in section 6.4. Finally, we discuss the setup and results and connect them to the ongoing work in our lab in section 6.5.

From the application point of view, Chapter 7 represent a highlight of the thesis in a certain sense. Here we investigate to what extent a machine – with the methodology described earlier in the thesis – can automatically build a model of a playing style of great pianists. For the work presented in this chapter we used only minimal performance information derived from audio CD recordings by truly great pianists and the printed score of the music. In section 7.1 we introduce the problems we are interested in. Section 7.2 explains the data and methodology used for the experiments. We then discuss the experimental evaluation of building 'expressive profiles' of the pianists in section 7.3. Section 7.4 discuss another interesting task: recognizing famous pianists from their style of playing. Finally, in section 7.5 we shortly discuss the automatic style replication.

In Chapter 8 we conclude. Limitations of both the learning algorithm and our overall methodology for learning expressive performance models are discussed, as well as motivation and possible directions for further work.

Chapter 2

Instance-based Learning in a Relational Setting

This chapter introduces the basic concepts of instance-based learning in a relational formalism.

The first section of this chapter gives a short overview of instance-based learning. We then briefly discuss one particular instance-based learning algorithm, namely k-nearest neighbor, one of the most popular techniques in the field of machine learning. The notion of distance in general, and its most common implementation on learning examples described in propositional languages are then discussed. We then review the formalism of first-order logic, as generally used in ILP, and show how learning examples in a relational setting are most commonly represented in practical learning systems. It turns out that representation via *sets of elements* is one of the most common and natural form of description. Accordingly, a distance measure operating on sets of elements has to be the central aspect of distance-based learning algorithm operating on relational data. In the last section we thus discuss different distance measures defined over sets of elements, one of them being the main 'ingredient' of our learning algorithm described in the next chapter.

2.1 Instance-based Learning

Instance-based learning is one of the most popular techniques in machine learning and data mining. It is a conceptually straightforward class of methods for approximating discrete- or real-valued functions. Learning in these algorithms consist of simply storing the presented training data. When a new test instance is encountered, a set of similar instances is retrieved from memory and used to classify the new query instance. Because of the delayed processing until a new instance must be classified, instance-based methods are sometimes referred to as 'lazy' learning ([Mitchell, 1997]).

One key property of these methods is that they can construct a different approximation to the target function for each distinct test instance that must be classified.

Given:
- set E_{train} of training instances x,
  along with their class label $f(x) = c$, $c \in C$
- query instance x_q to be classified

k-NN for classification:
- find $x_1, x_2, \ldots, x_k$, the k instances from E_{train}
  that are nearest to x_q
- return

$$\hat{f}(x_q) = argmax_{c \in C} \sum_{i=1}^{k} \sigma(c, f(x_i)) \tag{2.1}$$

  where $\sigma(a, b) = 1$ if $a = b$ and $\sigma(a, b) = 0$ otherwise

Figure 2.1: k-nearest neighbor for classification ((adapted from [Mitchell, 1997]).

In fact, they construct only a *local approximation* to the target function that applies in the neighborhood of the new test instance, and never construct an approximation designed to perform well over the entire instance space. This has significant advantages when the target function is very complex, but can still be described by a collection of less complex local approximations ([Mitchell, 1997]).

The main disadvantage of instance-based approaches is their computational complexity – both in terms of space (storage of instances) and time (search). This is due to the fact that nearly all computation takes place at classification time rather than when the training examples are first encountered.

In this work we will use the most popular form of instance-based learning, namely the k-nearest neighbor algorithm. It will be briefly recapitulated in the next section.

2.2 *k*-Nearest Neighbor Learning

The k-nearest neighbor algorithm is suitable for approximating both discrete- and continuous-valued target functions. Its most striking feature is its conceptual simplicity: When the class of a new example x_q should be predicted, the algorithm first collects the k training examples that are most similar to x_q. The test instance is then classified as the most frequent class value of the k nearest training examples. Figure 2.1 shows a high level description of the algorithm (adapted from [Mitchell, 1997]).

Approximating continuous-valued target functions in the k-nearest neighbor framework is straightforward: To accomplish this, the algorithm has to calculate the mean value of the k nearest training examples rather than the most frequent class value. In this case, the final line of the above algorithm would be:

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} f(x_i)}{k} \tag{2.2}$$

k-nearest neighbor is a highly effective learning method for many practical problems. Its algorithmic simplicity is just one of its strong aspects. Taking the (distance) weighted average of the k neighbors nearest to the test point can smooth out the impact of noisy training examples and/or attributes. Dealing with noisy training examples is one of the most important practical issues in machine learning. Other strong aspects of the k-nearest neighbor are the ability to deal with discrete as well as with continuous attributes, incrementality, and often surprisingly good generalization performance, which has been proved on a number of different learning tasks.

The main drawback of the nearest neighbor algorithm is its computational complexity. In the most naive approach the algorithm has to inspect each training example in turn and calculate its distance to the test example, retaining the identity only of the closest one. If we have n training examples, and each distance calculation is $O(d)$ (where d is the dimensionality of the feature space, i.e., the number of attributes), this search is thus $O(dn^2)$ ([Duda *et al.*, 2000]). Since for large training sets and/or complex distance computation, the algorithm becomes intractable, reducing its search complexity has received a great deal of analysis. Generally, there are three techniques for reducing its computational burden: computing partial distances, prestructuring, and pruning the stored prototypes (see also [Duda *et al.*, 2000]).

In the *partial distance* approach, the distance between test and training instances is computed using some subset of the features. If the partial distance is too great, the algorithm does not compute further. Intuitively speaking, partial distance methods assume that what one knows about the distance in a subspace is indicative of the full space. *Prestructuring* creates some form of *search tree* in which training instances are selectively linked. During classification, the distance of the test instance to one (or a few) stored 'root' prototype(s) is computed and only the instances linked to it (them) are considered. Of these, only the closest to the test instance is found and recursively, only subsequent linked prototypes are considered. If the tree is properly structured, the total number of instances that need to be searched is reduced. The third method for reducing the search complexity is to eliminate 'useless' instances during training, a technique known as *pruning*. A simple method is to eliminate instances that are surrounded by training points of the same category. This leaves the decision boundaries (and errors) unchanged, while reducing the search complexity.

The basic assumption of the k-nearest neighbor is that the class/value of the test instance will be similar to the classes/values of the similar training instances. Thus, we see that the algorithm heavily relies on the *distance* between instances. In the following section we discuss the notion of distance in general and describe some basic and well understood distance functions.

2.3 Distances

In the previous section we argued that the distance function is the most important component of k-nearest neighbor and more generally all instance-based learning methods. Therefore, one can improve these algorithms by improving the distance measures they use. In this section we first formally define the more general notion of metric. In the following we then review the best understood and most commonly used distance functions.

2.3.1 The Notion of Metric

A distance measure is a function that yields smaller values for more similar objects and larger values for more distant objects. While the definition of distance function depends on the description language used to formulate the learning task, the notion of *metric* is far more general. A metric $D(\cdot, \cdot)$ is merely a function that gives a scalar distance between two argument patterns and satisfies the following definition:

Definition 2.1 [**Metric**] Let S be a set. A function $D : S \times S \rightarrow \Re$ is a *metric* iff:

- the function is positive: $\vee x, y \in S : D(x, y) \geq 0$
- the function is reflexive: $\vee x, y \in S : D(x, y) = 0$ if and only if $x = y$
- the function is symmetric: $\vee x, y \in S : D(x, y) = D(y, x)$
- the function satisfies the triangle inequality:
  $\vee x, y, z \in S : D(x, y) + D(y, z) \geq D(x, z)$

It is straightforward to intuitively interpret the above conditions: The first two say that there should be a common minimal value for pairs of equal objects. Besides the obvious symmetry property, the triangle inequality states that the distance between any two objects (e.g., x and z) should be minimal if it is calculated directly (e.g. $D(x, z)$). Any indirect path (e.g. $D(x, y)$, $D(y, z)$) must be either equal or greater.

While not all distance measures commonly used in machine learning satisfy the above conditions, these are desirable properties, since the measures that satisfy them often produce more intuitive results and also allow many optimizations in algorithms using the measure.

2.3.2 Distance between objects in a propositional setting

Propositional logic or *attribute value language* is the most commonly used description language in machine learning and artificial intelligence in general ([Mitchell, 1997; Russell and Norvig, 2003]). It is beyond the scope of this work to give a full formal description of propositional logic (a nice overview can be found in [Russell and

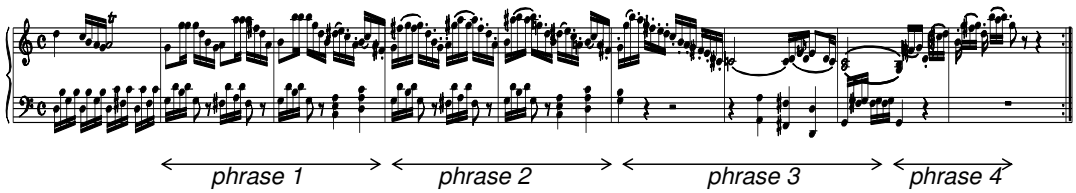

Figure 2.2: Score of the Mozart Sonata KV. 279 (C major), 1st movement, mm. 31-38. Phrase boundaries are indicated by brackets at the bottom of the figure.

Norvig, 2003] or [Nienhuys-Cheng *et al.*, 1997]). We will use the term *attribute value setting*, since in the bulk of machine learning research, learning instances are represented as tuples of values. The values are particular quantifications of the set of attributes which describe the instance.

More formally, the space of possible examples is the product $E = A_1 \times A_2 \times ... \times A_n$ of the n domains $A_1, A_2, ..., A_n$ of the attributes. Hence, each example e is an n-tuple $\langle attr_1(e), attr_2(e), ..., attr_n(e)\rangle \in E$. E is generally called the 'instance space'.

Example 2.1 Figure 2.2 shows the score of the Mozart Sonata KV. 279 (C major), 1st movement, mm. 31-38. This part of the piece can be divided into four *phrases* – segments which are considered as important building blocks of classical music. If we want to apply a machine learning algorithm on the level of phrases, in the attribute-value setting they would be represented as tuples $\langle attr_1(phr), attr_2(phr), ..., attr_n(phr)\rangle$, where $attr_1(phr), attr_2(phr), ...$ are values of the attributes that describe musical properties of the phrase, like the length of a phrase, mean and variance of the size of melodic intervals between the melody notes within the phrase, information about the presence of trills in the phrase etc. Some attributes have continuous domains (e.g. such as the mean of the size of melodic intervals between melody notes), and some are discrete (e.g., the basic information about the presence of trills in the phrase would belong to the set of values $\{true, false\}$).

The distance measures in attribute value setting are thoroughly investigated and understood. Most widely used is the Euclidean distance.

Definition 2.2 [Eucledean distance] Let e_1 and e_2 be arbitrary learning examples described by tuples $\langle attr_1(e_1), attr_2(e_1), ..., attr_n(e_1)\rangle$ and $\langle attr_1(e_2), attr_2(e_2), ..., attr_n(e_2)\rangle$, respectively. The Euclidean distance between e_1 and e_2 is then defined as

$$d_e(e_1, e_2) = \sqrt{\sum_{i=1}^{n} d_a(attr_i(e_1), attr_i(e_2))^2} \tag{2.3}$$

For the attributes with discrete domains, the distance $d_a(a_1, a_2)$ is usually defined as

$$d_a(a_1, a_2) = 1 - \delta(a_1, a_2) \tag{2.4}$$

where $\delta(a_1, a_2) = 1$ if $a_1 = a_2$, and $\delta(a_1, a_2) = 0$ otherwise.
For attributes with continuous domains the distance $d_a(a_1, a_2)$ is simply

$$d_a(a_1, a_2) = a_1 - a_2 \tag{2.5}$$

The Euclidean distance is a special case of the general class of distances called *Minkowski distance* ([Duda *et al.*, 2000]).

Definition 2.3 [Minkowski distance] Let e_1 and e_2 be arbitrary learning examples described by tuples $\langle attr_1(e_1), attr_2(e_1), ..., attr_n(e_1)\rangle$ and $\langle attr_1(e_2), attr_2(e_2), ..., attr_n(e_2)\rangle$, respectively. The Minkowski distance between e_1 and e_2 is then defined as

$$d_e(e_1, e_2) = (\sum_{i=1}^{n} |d_a(attr_i(e_1), attr_i(e_2))|^k)^{\frac{1}{k}} \tag{2.6}$$

Minkowski distance is also referred to as L_k norm, the Euclidean distance being thus the L_2 norm ([Duda *et al.*, 2000]). In the further text we will also make use of the L_1 norm, also called *Manhattan* or *city block* distance, which can be considered as the shortest path between two instances e_1 and e_2, if one has to 'travel' along the paths parallel to coordinate axes.

2.4 Relational Setting

The propositional setting is the most commonly used setting in machine learning, and a large literature corpus describes learning in this setting. However, for a number of applications, it is very problematic to represent the learning task in the attribute value setting ([Russell and Norvig, 2003; Raedt, 1998]). For such applications, trying to represent the learning task in attribute value form would either cause loss of information or create huge redundancies. In order to overcome these problems, in such cases machine learning researchers use *first-order logic* (FOL) or *relational setting*.

The basic assumption of first-order logic is that the world consists of *objects* and that *relations* hold between these objects (thus the name *relational setting*). The formalism was initially introduced by Gottlob Frege ([Frege, 1986]), and further developed by Alfred North Whitehead and Bertrand Russel ([Whitehead and Russel, 1910 1913]). The semantics of first-order logic was developed by Alfred Tarski ([Tarski, 1936, 1956]). It would be far beyond the scope of this thesis to give complete introduction to the concepts of FOL. In the next section we will thus formally describe just those concepts which we will use in the following text. For a more in-depth introduction to FOL and basic learning approaches in this setting see [Nienhuys-Cheng *et al.*, 1997].

2.4.1 Basic Syntactic Definitions

In the following we will specify an *alphabet*, i.e. the set of all symbols which can be used in forming syntactical structures in FOL (if not stated otherwise, all definitions in the following text are recapitulated from [Nienhuys-Cheng *et al.*, 1997]).

Definition 2.4 [Alphabet] An alphabet of first-order logic consists of the following symbols:

- A set of *constants*: $\{a, b, c, ...\}$.
- A set of *variables*: $\{U, V, W, ...\}$.
- A set of *function symbols*: $\{f, g, h, ...\}$. Each function symbol has a natural number (its *arity*) assigned to it. The arity defines the number of arguments the function has.
- A non-empty set of *predicate symbols*: $\{P, Q, R, ...\}$. Each predicate symbol has a natural number (its arity) assigned to it.
- The following five *connectives*: $\neg, \wedge, \vee, \rightarrow$, and $\leftrightarrow$.
- Two *quantifiers*: $\exists$ (called the *existential* quantifier) and $\forall$ (called the *universal* quantifier).
- Three *punctuation* symbols: '(', ')', ','.

With the above defined alphabet we can define *terms*, *atoms*, and *literals*.

Definition 2.5 [Term] Terms are defined as follows:

- A constant is a term.
- A variable is a term.
- If f is an n-ary function symbol and $t_1, t_2, ..., t_n$ are terms, then $f(t_1, t_2, ..., t_n)$ is a term.

Definition 2.6 [Atom] If P is an n-ary predicate symbol and $t_1, t_2, ..., t_n$ are terms, then $P(t_1, t_2, ..., t_n)$ is called an atom.

Definition 2.7 [Literal] If $P(t_1, t_2, ..., t_n)$ is an atom, then both, $P(t_1, t_2, ..., t_n)$ and $\neg P(t_1, t_2, ..., t_n)$ are called literals.

Informally speaking, terms (constants, variables and functions) refer to the *objects* in the world. On the other hand, predicates (atoms) are used to denote *properties* or *relations* between objects.

Example 2.2 With the terms $phrId1$, $phrId2$, $phrId3$, and $phrId3$ we could represent four phrases from the example in Figure 2.2. Further, we could use the predicate $PhrLength/2$ to describe the length of each phrase: e.g., $PhrLength(phrId1, 4)$ would state that the first phrase lasts four bars. It is also possible to define predicates which depict relations between phrases. E.g., the relational predicate $Succeed(phrId1, phrId2)$ would state that the second phrase follows the first.

In order to define the full first-order language, we have first to define formulas and ground formulas.

Definition 2.8 [**Formula**] Formulas are defined as follows:

- An atom is a formula.
- If ϕ is a formula, then $\neg\phi$ is a formula.
- If ϕ and ψ are formulas, then $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$ are formulas.
- If ϕ is a formula and X is a variable, then $\exists X\ \phi$ and $\forall X\ \phi$ are formulas.

Definition 2.9 [**Ground term/formula**] A ground term (respectively ground formula) is a term (resp. formula) which does not contain any variables.

With the above defined concepts it is straightforward to define the first-order language:

Definition 2.10 [**First-order language**] The first-order language given by an alphabet is set of all formulas which can be constructed from the symbols of the alphabet.

The above definitions do not give the complete formal description of first-order logic. Many important concepts from FOL – such as the *scope* of the quantifiers and variables in a formula, a *bound occurrence* of a variable in a formula, or *closed formulas* – are not defined. The introduced concepts serve merely as a background for reading the further text. Some of them will be used in the next section, where we show how the introduced relational formalism is commonly used in practical applications in machine learning for representation of learning examples.

2.4.2 Describing Learning Examples in a Relational Setting

In section 2.3.2 we have shown how a part of Mozart Sonata consisting of four musical phrases (Figure 2.2) could be represented in propositional logic. However, there is one major problem with the propositional representation of this particular example [1]: In classical music phrases are commonly organized *hierarchically*. Smaller

[1] In fact, there are several problems with such a representation, but for the purposes of the following text we focus on the presented.

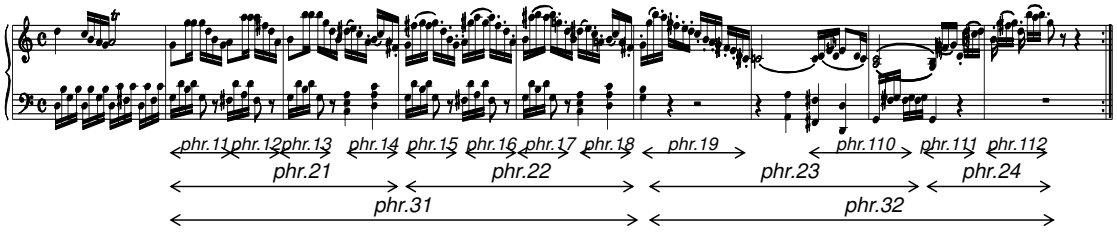

Figure 2.3: Score of the Mozart Sonata KV. 279 (C major), 1st movement, mm. 31-38. The hierarchical, three level phrase structure of the passage is indicated by brackets at the bottom of the figure.

phrases are grouped into higher-level phrases, which are in turn grouped together, constituting a musical context at a higher level of abstraction etc. Figure 2.3 reproduces the same part of the Mozart Sonata given in Figure 2.2 together with the hierarchical, three-level phrase structure indicated by three levels of brackets at the bottom of the figure. Phrase identifiers in the figure have the form $phr.LT$, L denoting the level of 'musical abstraction' to which a phrase belongs, and T its 'temporal order' within a particular level (e.g. $phr110$ refers to the tenth phrase of the first phrasing level).

If we want our relational learning algorithm to operate on the phrase level, we need to represent each phrase in a relational formalism. In relational IBL this is most commonly done by collecting all atoms from the database relevant to the object we want to represent and grouping them in a set, which is in turn the 'raw' description for the particular object and its *relevant context*.

Example 2.3 Suppose we want to represent two second-level phrases $phr21$ and $phr24$. Suppose further that in addition to the phrase identifier, our language consists of the two predicates $PhrLength/2$ and $Contains/2$. $PhrLength(PhrId, Length)$ describes the length of a phrase (e.g. in bars), and $Contains(PhrId1, PhrId2)$ states that the higher-level phrase $PhrId2$ contains the lower-level phrase $PhrId1$. Our goal would be to collect all predicates from the database which are 'relevant' to the phrases $phr21$ and $phr24$. The formal definition of this procedure will be given in section 3.3. For now, assume that 'relevant predicates' are predicates which are 'connected' via the same phrase identifier. After this procedure, the phrases $phr21$ and $phr24$ would be represented with the following two sets (since the lengths of the phrases are not important for the purposes of the example, we omit the actual numbers):

S_{phr21}=[$PhrLength(phr21, \cdot)$, $Contains(phr11, phr21)$, $Contains(phr12, phr21)$, $Contains(phr13, phr21)$, $Contains(phr14, phr21)$, $PhrLength(phr11, \cdot)$, $PhrLength(phr12, \cdot)$, $PhrLength(phr13, \cdot)$, $PhrLength(phr14, \cdot)$, $Contains(phr21, phr31)$, $PhrLength(phr31, \cdot)$, $Contains(phr22, phr31)$,

$PhrLength(phr22, \cdot)$, $Contains(phr15, phr22)$, $Contains(phr16, phr22)$, $Contains(phr17, phr22)$, $Contains(phr18, phr22)$, $PhrLength(phr15, \cdot)$, $PhrLength(phr16, \cdot)$, $PhrLength(phr17, \cdot)$, $PhrLength(phr18, \cdot)$].

and

S_{phr24}=[$PhrLength(phr24, \cdot)$, $Contains(phr111, phr24)$, $Contains(phr112, phr24)$, $PhrLength(phr111, \cdot)$, $PhrLength(phr112, \cdot)$, $Contains(phr24, phr32)$, $PhrLength(phr32, \cdot)$, $Contains(phr23, phr32)$, $PhrLength(phr23, \cdot)$, $Contains(phr19, phr23)$, $Contains(phr110, phr23)$, $PhrLength(phr19, \cdot)$, $PhrLength(phr110, \cdot)$].

From the example discussed above, one can see the obvious advantage of using relational representation: Objects (i.e., learning instances) are described not only by their attributes, but also by their relations to other objects. In the above example, the phrases are represented by their attributes (like in the propositional representation), and additionally, by relations to other phrases (upper or lower in the hierarchy). These other phrases are also described by their attributes. In this way, the distance between two phrases depends not only on their attributes but also on their *context*. As the reader can imagine (and we will prove in the experimental section), introduction of 'context' in a distance measure can be of much value for real-world learning tasks.

Since grouping atoms into sets of elements (and structuring the sets in some way) is one of the most commonly used techniques in describing learning examples in practical relational systems, defining the distance measure operating on sets of elements is obviously a crucial part of a distance-based relational algorithms. In the following section we will introduce the most common set distance measures. We will see that there is a trade off between the set sizes and computational feasibility of set distance measures in practical applications.

2.5 Distances Between Sets of Elements

In this section we discuss distances between sets of elements. However, before we introduce the most common distance measures between sets, we will define two concepts which we will use in the subsequent chapters. The first is the *weighted set* as a generalization of the *set* concept. The second is the *size* of the weighted set, as a generalization of the notion of cardinality (adapted from [Ramon and Bruynooghe, 2001]).

Definition 2.11 [Weighted set] Let X be a set. A weighted set W is a function $W[X] : X \rightarrow \Re$.

Definition 2.12 [Size under weighting function]. The size of a set X under

a weighting function for W is defined as $s_W(X) = \sum_{x \in X} W[X](x)$.

While for the most commonly used mathematical concept of a set, an element belongs to the set or not, a weighted set can be thought of as assigning to each element a degree to which it belongs to the set. Ordinary sets are special cases of weighted sets: one can see a weighted function W of a set X as a function that maps all elements $x \in X$ onto 1 and all other elements onto 0 in this special case.

In the following sections, we will assume that, given the set of all possible elements U, the distance measure between individual elements of U, d_U is defined. On top of this elementary distance measure d_U, the distances between sets of elements will be defined. We will start with simpler distance measures, and end up with a distance measure based on maximal matching, which is built in our learning algorithm described in the next chapter.

2.5.1 Symmetric Difference Distance

A straightforward way to compare two sets is to consider which elements are in one set and not in the other. The set of such elements is called the symmetric difference ([Eiter and Mannila, 1997]). More formally,

Definition 2.13 [Symmetric difference]. Let A and B be two sets. The symmetric difference of A and B is defined as

$$A \Delta B = (A/B) \cup (B/A) \tag{2.7}$$

Then, the following distance measure is well known:

Definition 2.14 [Symmetric difference distance] ([Ramon and Bruynooghe, 1998]). Let A and B be two sets and W their weighting function. The symmetric difference distance between A and B is then defined as:

$$D_\Delta(A, B) = s_W(A \Delta B) \tag{2.8}$$

Although the symmetric difference distance satisfies all metric properties (see e.g. [Ramon and Bruynooghe, 1998]) and it can be computed very efficiently, its applicability in real tasks is very limited: The distance is basically computed as (weighted) number of elements which are in one set and not in another, without taking into account any elementary distances between elements of sets.

2.5.2 Minimum Distance

The definition of the minimum distance is very straightforward: The distance between two sets is simply the minimum of all possible distances between any two elements ([Eiter and Mannila, 1997]).

Definition 2.15 [**Minimum distance**]. Let A and B be two sets and d_U the distance between their individual elements. The minimum distance function $D_{min}(A, B)$ is defined by:

$$D_{min}(A, B) = min_{a \in A, b \in B} d_U(a, b) \tag{2.9}$$

The minimum distance can be computed efficiently: In [Toussaint and Bhattacharya, 1983] it is shown that the minimum distance can be computed in time $O(mlog(m))$, where $m = max(\#A, \#B)$.

However, beside the fact that the minimum distance is not a metric, its major drawback are the unintuitive results. E.g., let $A = \{a, b\}$ and $B = \{a\}$. Then $D_{min}(A, B) = 0$ while A and B are not identical.

2.5.3 Distance Based on the Sum of Minimal Distances

A distance based on minimum distance which takes all elements in both sets into account is the sum of minimal distances (discussed in [Eiter and Mannila, 1997]):

Definition 2.16 [**Distance based on the sum of minimal distances**]. Let A and B be two sets and d_U the distance between their individual elements. The distance based on the sum of minimal distances between A and B is defined by:

$$D_{\sum_{min}}(A, B) = \frac{1}{2}(\sum_{a \in A}(min_{b \in B} d_U(a, b)) + \sum_{b \in B}(min_{a \in A} d_U(a, b))) \tag{2.10}$$

While a distance defined in this way looks more intuitive, it is in general not a metric. In particular, it does not satisfy the triangle inequality:

Example 2.4 Let $A = \{1, 2\}$, $B = \{3, 4\}$, and $C = \{5, 6\}$. Let $d_U = |\cdot - \cdot|$. Then, $D_{\sum_{min}}(A, B) = 3$, $D_{\sum_{min}}(B, C) = 3$, and $D_{\sum_{min}}(A, C) = 7$, hence $D_{\sum_{min}}(A, B) + D_{\sum_{min}}(B, C) < D_{\sum_{min}}(A, C)$ – the triangle inequality does not hold.

2.5.4 The Hausdorff metric

One of the most used set distances is the Hausdorff metric ([Eiter and Mannila, 1997]). It is defined as follows:

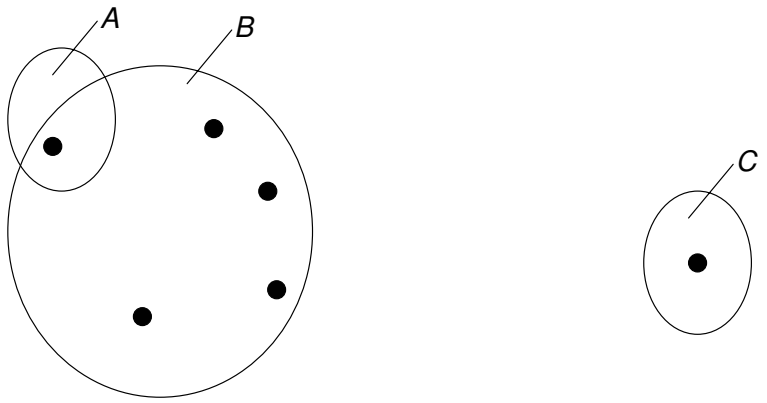

Figure 2.4: Assuming the distance between individual set elements d_U being planar Euclidean distance, the sets A and B are – according to the Hausdorff distance – equally distant from C.

Definition 2.17 [**Hausdorff distance**]. Let A and B be two sets and d_U the distance between their individual elements. The Hausdorff distance is then defined as:

$$D_H(A,B) = max(max_{a\in A}(min\,\{d_U(a,b)|b \in B\}), max_{b\in B}(min\,\{d_U(a,b)|a \in A\})) \tag{2.11}$$

The Hausdorff distance has been considered in the area of computational geometry, where the distance between geometric entities has to be computed (see, e.g. [Huttenlocher and Kedem, 1990]). Put simply, the distance between two sets is determined by the distance of the most distant elements of both sets to the nearest neighbor in the other set. The advantages of the Hausdorff distance are its computability in polynomial time and the fact that it satisfies all properties of a metric ([Eiter and Mannila, 1997]). One of the problems with the Hausdorff distance is that it is sensitive to extreme points in the sets. Consider the example given in Figure 2.4. If we assume the distance between individual set elements d_U being planar Euclidean distance, the sets A and B are equally distant from C according to the Hausdorff distance.

The major drawback of the Hausdorff distance is that the distance between two sets depends just on the distance between two points in the sets. A more sophisticated set distance measure would combine information about the elementary distances between many elements of both sets. In the following we will discuss such a distance, which is based on optimal matching between sets.

2.5.5 Distance Based on Optimal Matching

It is a rather intuitive idea that a set distance measure should first find corresponding elements in both sets and then use this information to compute the distance between the sets.

Example 2.5 Suppose we want to compare two families. The first family consists of a father f_1, a mother m_1, and a child c_1. The second family consists of a father f_2, mother m_2, a child c_2, a grandfather gf_2, and a grandmother gm_2. When comparing these two families, it would be natural to look for corresponding family members. One would then compare both fathers f_1 and f_2, both mothers m_1 and m_2, children c_1 and c_2 and finally take into account that in family 1 there are no grandparents gm and gf like in family 2.

In the following we will introduce a distance measure rooted in this idea, which is based on optimal matching between two sets. First we review a definition of matching and maximal matching between two sets (from [Grimaldi, 1998]).

Definition 2.18 [Matching]. Let A and B be two sets. A relation $f \subseteq A \times B$ between two sets A and B is a matching iff

$$\forall (a,b),(c,d) \in f : (a = c \Leftrightarrow b = d) \tag{2.12}$$

so each element of A is associated with at most one element of B and vice versa.

Definition 2.19 [Maximal matching]. A maximal matching f between A and B is a matching for which there is no $(a,b) \in A \times B$ such that $f \cup \{(a,b)\}$ is a matching between A and B.

Figure 2.5 illustrates the concepts of matching and maximal matching. Based on these concepts, the optimal matching distance can be defined
([Ramon and Bruynooghe, 2001]):

Definition 2.20 [Optimal matching distance]. Let A and B be two sets, $m(A,B)$ the set of all maximal matchings between A and B, and d_U the distance between their individual elements. The optimal matching distance is then defined as:

$$D_{OM}(A,B) = (min)_{r \in m(A,B)} d(r,A,B) \tag{2.13}$$

where

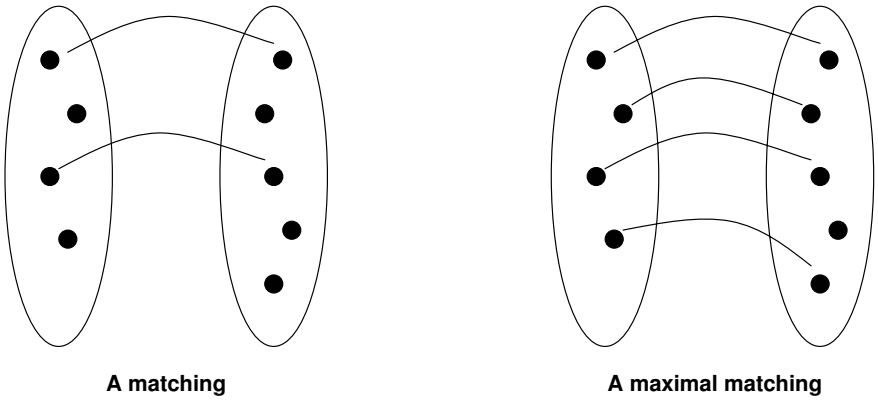

Figure 2.5: Examples of a matching and maximal matching between two sets.

$$d(r, A, B) = \left[\sum_{(x,y)\in r} d(x, y) \right] + \frac{M}{2} \cdot (\#A + \#B - 2 \cdot \#r) \tag{2.14}$$

and $\#A$, $\#B$, and $\#r$ denote cardinalities of sets A, B, and r, respectively.

In this formula, the constant M represents the maximal possible distance between two points in both sets. This means that one sums up the distances of the pairs of elements which are in r and adds a penalty $M/2$ for each element that does not match with an element from the other set. The optimal matching distance is then the minimal distance of all possible maximal matching distances. Besides its strong aspects, namely rooting in a rather intuitive idea and utilising as much information about both sets as possible, it has been shown that for positive M it satisfies all properties of a metric (for the proof that an optimal matching distance satisfies all four metric conditions stated in definition 2.1, see [Ramon and Bruynooghe, 2001]).

The main problem with the optimal matching distance is its computational complexity. In order to calculate the distance between two sets a naive approach would be to calculate distances between two sets for all possible maximal matchings and return the minimal one. Due to the combinatorial nature of such an approach, such a computation would have exponential complexity and thus would be intractable even for moderate numbers of set elements.

Fortunately, in the next chapter we will see that the problem of finding the optimal matching distance can be restated in a form suitable for applying concepts from *graph theory*. One tool from graph theory, namely transport networks, is shown to be able to solve the problem of optimal matching distance in time polynomial in the size of both sets. In the next chapter this technique is discussed. We will then introduce our new similarity measure based on the optimal matching distance and show how it is built into our relational instance-based learning algorithm DISTALL.

Chapter 3

DISTALL

This chapter describes our new instance-based learner DISTALL and briefly contrasts it with its ancestor RIBL [Emde and Wettschereck, 1996], by showing via an example how they implement different notions of structural similarity. A large part of the material presented here has been already published in [Tobudic and Widmer, 2003a] and [Tobudic and Widmer, 2006].

3.1 Introduction

In designing our structural distance measure we have been guided by three basic principles:

- the new structural distance measure should be intuitive, taking into account as much relevant information as possible
- it should have a solid theoretical background
- it should be computationally feasible for real world problems

This premises are not trivial to fulfil. We have seen in chapter 2 that learning examples in a relational setting are commonly described as sets of ground facts. However, a number of set distance measures are both very limited, taking into account only a small part of available information in both sets, and often do not satisfy the properties of metric. In section 2.5 we introduced the set distance measure based on optimal matching and argued that it has roots in an intuitive idea. Additionally, it utilizes a lot of information about distances between elementary set elements. Furthermore, it is a metric. On the other hand, the main drawback of the optimal matching distance is its computational complexity. In this chapter we will show how to effectively compute the optimal matching distance measure in time polynomial in the size of both sets.

Having a distance measure which runs in polynomial time does not satisfy our third guideline on its own: the computational feasibility for real world problems.

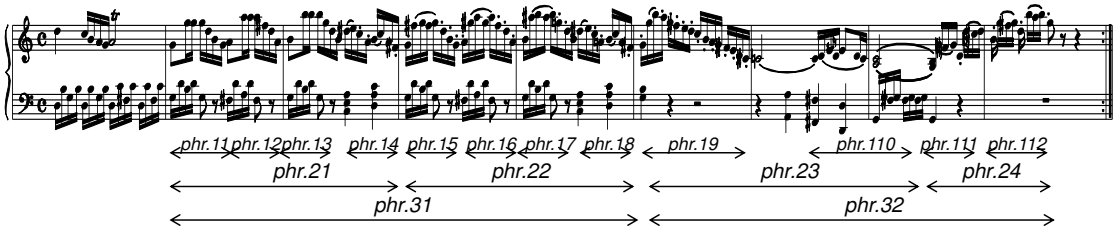

Figure 3.1: Score of the Mozart Sonata KV. 279 (C major), 1st movement, mm. 31-38. The hierarchical, three level phrase structure of the passage is indicated by brackets at the bottom of the figure.

Applying such a distance measure directly on real world examples – each containing maybe hundreds of ground facts or more – would cause impractically high computational costs. To avoid this problem we group the ground facts into *hierarchical subsets* and apply the optimal matching distance on these substantially smaller sets. The mechanism for subset formation is described in subsequent sections. In section 3.5 we put all pieces together and describe our structural similarity measure which is then built in our relational instance-based learner DISTALL. Finally, we contrast it with its ancestor RIBL, the state of the art relational instance-based learner.

None of DISTALL's basic 'ingredients' is new. How to effectively compute optimal matching distance is shown in [Ramon and Bruynooghe, 2001]. Grouping ground facts into subsets is a well known technique in relational learning (see e.g. [Raedt, 1992], GOLEM [Muggleton and Feng, 1990], RIBL [Emde and Wettschereck, 1996]) and a technique for computing a distance by forming *hierarchical subsets* is used in RIBL [Emde and Wettschereck, 1996]. However, the novel aspect of this work is that our structural distance measure combines these elementary techniques in a unique way, resulting in a new relational instance-based learner. In chapter 5 we then report about experiments assessing DISTALL's performance and contrasting it with its 'ancestor' RIBL on a difficult, real-world learning task derived from the field of expressive music performance research.

3.2 Representation of the Learning Input

In section 2.4 we have discussed the main concepts regarding relational formalism. We have argued that the basic assumption of this formalism is that the world (and the learning problem) consists of objects and that relations hold between these objects. A 'raw' input to our learning algorithm thus consists of a database containing predicates which describe objects and relations which hold between them.

Consider our music example from section 2.4.2. The Figure 2.3 describing the phrasing structure is repeated in Figure 3.1. Consider further, that our learning problem consists of just two predicates, $PhrLength/2$ and $Contains/2$. As in the Example 2.3, $PhrLength(PhrId, Length)$ describes the length of a phrase $PhrId$

in bars, and $Contains(PhrId1, PhrId2)$ states that the higher-level phrase $PhrId2$ contains the lower-level phrase $PhrId1$. The starting point for our learning algorithm would then be an unstructured database of facts as given in the following :

DB=[$PhrLength(phr11, 0.5)$, $PhrLength(phr12, 0.5)$, $PhrLength(phr13, 0.5)$, $PhrLength(phr14, 0.5)$, $PhrLength(phr15, 0.5)$, $PhrLength(phr16, 0.5)$, $PhrLength(phr17, 0.5)$, $PhrLength(phr18, 0.5)$, $PhrLength(phr19, 1.0)$, $PhrLength(phr110, 1.0)$, $PhrLength(phr111, 0.5)$, $PhrLength(phr112, 0.5)$, $PhrLength(phr21, 2.0)$, $PhrLength(phr22, 2.0)$, $PhrLength(phr23, 2.5)$, $PhrLength(phr24, 1.0)$, $PhrLength(phr31, 4.0)$, $PhrLength(phr32, 3.5)$, $Contains(phr11, phr21)$, $Contains(phr12, phr21)$, $Contains(phr13, phr21)$, $Contains(phr14, phr21)$, ...
$Contains(phr21, phr31)$, $Contains(phr22, phr31)$, $Contains(phr23, phr32)$, $Contains(phr24, phr32)$].

In addition to the database, predicate and type declarations are given. The predicate declarations define the arity of the predicates as well as the sorts and mode declarations of their arguments. Sorts are used to declare types of the predicate arguments and mode declarations specify which arguments are input or output arguments. Type definitions are used to differentiate between arguments that represent an object in a domain (e.g., a phrase) and attribute values (e.g., the phrase length). This information is used in the same manner as it is used in other ILP learners (e.g., CLINT [Raedt, 1992], GOLEM [Muggleton and Feng, 1990] and RIBL [Emde and Wettschereck, 1996]).

To illustrate this concepts, the predicate and type declaration concerning our example would be:

- $PhrLength/2$:! $< phrase >, < barLength >$.
- $Contains/2$:! $< phrase >$, ! $< phrase >$.

- $type : phrase : object$.
- $type : barLength : number$.

The first declaration states that the two arguments of the predicate $PhrLength$ are of sorts $phrase$ and $barLength$. The '!' declares the first argument as predicate input. By omitting the sign '!' we have implicitly declared the second argument as predicate output. Similary, $Contains$ consists of two arguments, both being input arguments of the sort $phrase$. Type definitions state that arguments of sort $phrase$ should be treated as objects and arguments of sort $barLength$ as numerical values.

The first step of our algorithm is building sets describing learning examples and grouping the sets into smaller groups which facilitate faster computation of set

distances. The sets are extracted from the database and the process is guided by predicate and type definitions. In the following section we describe it in a more detail.

3.3 Generation of Structured Sets Describing Learning Examples

Before explaining the process of generation of structured sets which describe learning examples, we recall some definitions from [Helft, 1989] and [Muggleton and Raedt, 1994].

Definition 3.4 [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 3.1 Clause $p(A) \leftarrow r(B)$ is not linked, while $p(A) \leftarrow q(A, B), r(B, C), u(D, C)$ is.

Definition 3.5 [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 3.2 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 set of literals describing an learning example (i.e. an object) can be generated from the database and structured into subsets according to their appropriate linkage levels (depths) by iterating the following procedure:

Given an object (like the phrase $phr21$ from the example given in Figure 3.1), we first collect all literals from the database containing the object identifier (e.g., $phr21$) as one of their arguments. These literals would make up the subset at depth 0. Then, we can determine the set of all new objects contained in the subset of depth 0. In the following we collect all literals from the database containing the new objects, and group them in a set at linkage level 1. This procedure can be recursively repeated until some depth parameter is reached.

Example 3.3 Consider again our music example given in Figure 3.1 describing the phrasing structure of part of the Mozart Sonata KV. 279 and two predicates $PhrLength(PhrId, Length)$ and $Contains(PhrId1, PhrId2)$. In this example, the set of literals describing the phrase $phr21$ would be divided into four subsets according to their linkage levels:

$LinkLevel_0$=[$PhrLength(phr21, 2.0)$, $Contains(phr11, phr21)$, $Contains(phr12, phr21)$, $Contains(phr13, phr21)$, $Contains(phr14, phr21)$, $Contains(phr21, phr31)$]

$LinkLevel_1$=[$PhrLength(phr11, 0.5)$,$PhrLength(phr12, 0.5)$, $PhrLength(phr13, 0.5)$, $PhrLength(phr14, 0.5)$, $PhrLength(phr31, 4.0)$, $Contains(phr22, phr31)$]

$LinkLevel_2$=[$PhrLength(phr22, 2.0)$, $Contains(phr15, phr22)$, $Contains(phr16, phr22)$, $Contains(phr17, phr22)$, $Contains(phr18, phr22)$]

$LinkLevel_3$=[$PhrLength(phr15, 0.5)$, $PhrLength(phr16, 0.5)$, $PhrLength(phr17, 0.5)$, $PhrLength(phr18, 0.5)$]

This process is restricted by user specified argument sorts, types, and modes. Information of argument sorts ensures that an object referred to by an argument A at some depth N is only 'linked' to the literals at depth $N + 1$ if these contain A as one of their arguments and if that argument is sort compatible with the original argument A. For example, a phrase with the object identifier $obj21$ would not be linked to the trill identifier $obj21$. Information on argument types is also utilized to prevent linking of non-object arguments. According to this restriction, the numerical argument 2.0 in $PhrLength(phr21, 2.0)$ would not be linked to any argument one level deeper. Finally, argument modes enable linkage only along input arguments. Output arguments are considered to be determined by the input arguments of the predicate and, therefore, offer no additional information. For more information on sorts, types and modes see [Raedt, 1992; Muggleton and Feng, 1990; Emde and Wettschereck, 1996]. The generated structured set for each given object is in the ILP literature referred as *starting clause*. In the following we will also used this term.

3.4 Efficiently Computing the Optimal Matching Distance

In previous section we showed how structured sets describing learning examples can be generated from the domain database. In section 2.5 we introduced the set distance measure based on optimal matching. Since the distance measure represent the core part of our learning algorithm, in the following we present a technique for efficient computation of set distance measure based on optimal matching. It turns out that the optimal matching distance measure can be computed in time polynomial in the size of both sets.

The optimal matching distance between two sets of elements can be efficiently computed via transport networks, a concept rooted in graph theory. In the following we will review in more detail how it can be done (recapitulated from [Ramon and

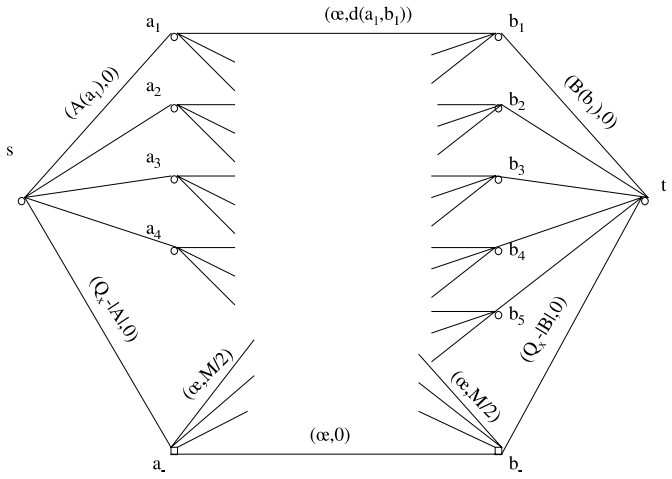

Figure 3.2: A distance network (see [Ramon and Bruynooghe, 2001])

Bruynooghe, 2001]), without introducing transport networks in more detail (for more information on graph theory and the concept of transport networks see [Mehlhorn, 1984]).

In order to efficiently compute the distance based on optimal matching between two sets of elements, one has to first construct the appropriate transport network: 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 or equal than the maximal weight in the network), 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. An example of a distance network is given in Figure 3.2.

More formally, given the definitions of the weighted set and the size under weighting function (defined in section 2.5), the distance network can be formalized as follows [Ramon and Bruynooghe, 2001]:

Definition 3.1 [Distance network] Let X be a set, and d a metric on X. Let M

be a constant, W be a weighting function for X and $Q_X^W = max_{A \in 2^X} size_W(A)$. Then for all finite $A, B \in 2^X$ with $A = \{a_1, ..., a_m\}$ and $B = \{b_1, ..., b_n\}$, we define a distance network between A and B for d, M and W in X to be N[X,d,M,W,A,B]=N(V,E,cap,s,t,w) with $V = A \cup B \cup \{s, t, a_-, b_-\}, E = (\{s\} \times (A \cup \{a_-\})) \cup ((B \cup \{b_-\}) \times \{t\}) \cup ((A \cup \{a_-\}) \times (B \cup \{b_-\}))$, $\forall a \in A, \forall b \in B : w(s,a) = w(b,t) = w(s,a_-) = w(b_-,t) = w(a_-,b_-) = 0 \wedge w(a,b) = d(a,b) \wedge w(a_-,b) = w(a,b_-) = M/2$ and $\forall a \in A, \forall b \in B : cap(s,a) = W[A](a) \wedge cap(b,t) = W[B](b) \wedge cap(s,a_-) = Q_X^W - size_W(A) \wedge cap(b_-,t) = Q_X^W - size_W(B) \wedge cap(a,b) = cap(a_-,b) = cap(a,b_-) = cap(a_-,b_-) = \infty$ (see Figure 3.2).

The distance between sets of elements is then defined as follows:

Definition 3.2 [Netflow distance] Let X be a set, d a metric on X, M a constant, and W a weighting function for X. For all $A, B \in 2^X$, the netflow distance from A to B under d, M and W in X, denoted $d^N_{X,d,M,W}(A,B)$, is the weight of the minimal weight maximal flow from s to t in N[X,d,M,W,A,B].

In [Mehlhorn, 1984] it has been shown that if W has integer values, the maximal flow minimal weight problem can be solved in time polynomial in $size_W(A)$ and $size_W(B)$.

If the weights of the netflow distance are normalized (in the interval [0,1]), the netflow distance can also be normalized:

Definition 3.3 [Normalized netflow distance] Let X be a set and d a normalized metric on X. Then, the normalized netflow distance based on d is a distance function $d^{N,n}_{X,d,M,W}(A,B) : X \times X \rightarrow \Re$ defined by:

- 0 if (X=0 and Y=0)
- $\frac{2 \times d^N_{X,d,M,W}(A,B)}{d^N_{X,d,M,W}(A,B) + (size_W(X) + size_W(Y))/2}$ otherwise

The equivalents of this rather abstract description of transport networks for purposes of our learning algorithm would be as in the following:

Each of the vertices $\{a_i\}$ and $\{b_i\}$ corresponds to one literal from the sets A and B for which we want to compute the distance. *Weights* between individual vertices are the distances between individual literals (see next section for description how to compute the distance between individual literals). *Capacities* between vertices can be set to scale the sets to approximately same 'virtual cardinality', e.g. by allowing the literals of the smaller set to match up more than one literal of the bigger set. This would avoid the fact that the distance of two sets with vastly different cardinalities is expressed mainly through penalty. The set distance between sets of literals A and B would then correspond to the solution of the maximum flow minimal weight problem.

In the following section we describe how the individual techniques introduced so far are put together and built in our relational instance based learner DISTALL.

3.5 The DISTALL Algorithm

Since DISTALL algorithm incorporates some rather complex concepts (like transport networks described above), we will first introduce it by explaining its precise algorithmic description (given in Figure 3.3). We will then use the intuitive graphical description (Figure 3.4) to further clarify its most important issues. In the end of the section we give a toy example of distance computation between two 'instances' from our music domain.

For each (training and test) example (i.e. object like phrase $phr21$), DISTALL first generates their starting clauses as described in section 3.3. For each test instance it then finds the k most similar training examples and makes prediction based on them (lines 1–8 in Figure 3.3). The *distance* between a test and training example is computed as set distance between all literals found in the test and training starting clause at linkage level 0 (line 6 in Figure 3.3). In other words, we find the solution of the maximal flow minimal weight problem (see Definition 3.2), where the transport network vertices are literals containing terms which are directly linked to the training and test instances. In order to find the solution of the maximal flow minimal weight problem we need to calculate values of all edge weights in the network (line 12 in Figure 3.3). The weights of the edges are defined as distances between individual literals. The distance between literals is set to 1 if they have different functors (line 16 in Figure 3.3). If the literals have the same functors, the distance between them is computed as the average distance over all their arguments (line 25 in Figure 3.3). The distance over literals' arguments is computed as the Manhattan distance if the arguments are numeric (line 20 in Figure 3.3)), or as a discrete distance if the arguments are symbolic (line 21 in Figure 3.3)). However, if the arguments are object identifiers (line 22 in Figure 3.3)), 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. As a consequence of this procedure, the optimal matching distance is computed recursively, each time on the sets at the one linkage-level 'deeper'. That is why we termed the algorithm DISTALL: DIstance on SeTs of Appropriate Linkage Level. At the lowest level, the distance between objects is calculated as the distance between discrete values (line 10 in Figure 3.3).

The basic principle of DISTALL is illustrated in Figure 3.4. 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 the sets of literals found at $LinkLevel = 0$ 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 the 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 one network.

Notice that the distances between vertices are normalized. With a normalized distance between vertices we can apply Definition 3.3 and normalize the netflow distance. Normalized netflow distance is in turn used in the computation of the (normalized) distance between literals in the 'higher-level' transport network.

Example 3.4 Suppose we want to calculate distance between two phrases from out music example, let say $phr21$ and $phr23$ illustrated in Figure 3.5. In order to keep our example comprehensible, we simplified the real hierarchical phrase organization (given in Figure 3.1 and in the Example 3.3) in that there are just two additional higher-level phrases which contain phrases $phr21$ and $phr23$, namely phrases $phr31$ and $phr32$, with no other phrases in the musical passage. Assuming there is just one more predicate (in addition to $Contains(PhrId1, PhrId2)$), which describes the length of the phrases ($PhrLength(PhrId, Length)$), the starting clauses of the two phrases would be:

$$SC(phr21)=\{0{:}[PhrLength(phr21, 2.0), Contains(phr21, phr31)],1{:}[phrLength(phr31, 4.0]\}$$

$$SC(phr23)=\{0{:}[PhrLength(phr23, 2.5), Contains(phr23, phr32)],1{:}[phrLength(phr32, 3.5]\}$$

Each starting clause consists of two literals at depth 0 and one literal at depth 1. The distance between $phr21$ and $phr23$ is calculated as optimal matching distance on sets at depth 0. These are $[PhrLength(phr21, 2.0)$, $Contains(phr21, phr31)]$ and $[PhrLength(phr23, 2.5)$, $Contains(phr23, phr32)]$ for this example. In order to compute the set distance, we must calculate all edge weights between the elements of the two sets. These weights are the distances between individual literals. Distances between individual literals are 1 for literals with different functors: E.g., edge weight between $PhrLength(phr21, 2.0)$ and $Contains(phr23, phr32)$ would be set to 1. The distance between literals with the same functors is the average of the distances between their arguments. In our example, the edge weight between $PhrLength(phr21, 2.0)$ and $PhrLength(phr23, 2.5)$ would be the Manhattan distance between their second arguments and would be 0.5. [1] Note that the distance in this case would not be the average of the first and second argument distances, since the object arguments for which the transport network is made (i.e. $phr21$ and $phr23$) do not contribute to the distance between the literals. It would not make sense to take this distance into account, since it is the distance we were interested in the first place!

We have the similar situation in determining the distance (edge weight) between literals $Contains(phr21, phr31)$ and $Contains(phr23, phr32)$. The distance between these two literals fully depends on the distance between the arguments $phr31$ and

[1] In the real implementation of DISTALL we scale the distance with the total range of the argument in the data set (see line 20 of in Figure 3.3). This scaling is irrelevant to this illustrative example.

$phr32$. Since these arguments are object identifiers, the distance between them is calculated 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. In our example, the new transport network would contain only two vertices, namely literals $PhrLength(phr31, 4.0)$ and $PhrLength(phr32, 3.5)$. The optimal matching set distance for this transport network would be straightforward to compute: we need to assign just one edge weight, namely the distance between literals $PhrLength(phr31, 4.0)$ and $PhrLength(phr32, 3.5)$. This literal distance would again be the Manhattan distance between their second argument.

On this example we see the intuitive idea behind DISTALL's distance measure: Distance between phrases $phr21$ and $phr23$ is based on: 1) Their similarities (e.g. their bar lengths – reflected in the edge weight computation between literals $PhrLength(phr21, 2.0)$ and $PhrLength(phr23, 2.5)$ in the transport network on depth level 0). 2) The similarities between objects which are related to them (e.g. the bar lengths of the phrases $phr31$ and $phr32$, which contain the initial two phrases). In this way DISTALL incorporates the *contextual* information in its distance measure.

Of course, this example is intended to be an instructive and not very realistic one. The 'direct' similarities between phrases in real application would depend on many phrase attributes and not just on their lengths (e.g. melodic intervals between notes in the phrases, the harmonic development in the phrases etc.). Similarly, the phrases would be in 'relations' with many more phrases, e.g., they would contain many phrases, which would themselves contain other phrases etc. However, the example shows the basic principles of DISTALL's similarity measure.

3.6 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 Wettschereck, 1996], in the context of the relational instance-based learning algorithm RIBL. DISTALL and RIBL share the main idea behind their similarity measures, namely that the similarity between two objects should be 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.

Where DISTALL and RIBL differ is the set distance measure. DISTALL incorporates an intuitively elegant set distance measure based on optimal matching which is then efficiently computed by applying the concept of transport networks (first proposed in [Ramon and Bruynooghe, 2001]). As a consequence of its similarity computation, DISTALL's set distance strongly favors matchings that are as complete as possible and penalizes literals left unmatched. On the other hand, RIBL permits several literals in one set to match the same literal in the other. It

is beyond this text to give a in-depth description of RIBL's similarity measure (for detailed description see [Emde and Wettschereck, 1996]). However, since DISTALL is in some sense inspired by RIBL, we explain the main difference between RIBL's and DISTALL's behavior in one constructed situation from our musical application domain. In chapter 5 we present DISTALL's empirical results and provide a direct comparison with RIBL.

Consider again our music example from sections 2.4.2 and 3.3. A slightly modified situation is reproduced in Figure 3.6. In this example we want to calculate the distance between phrases $phr31$ and $phr32$. Each of the two phrases is described via the attributes stored in the predicate $PhrCont$, which describe the musical 'content' of each phrase. Both phrases also contain lower-level phrases (predicate $Contains$), which are in turn described with their phrase-content predicates. RIBL would compute the similarity between $phr31$ and $phr32$ as a (weighted) sum of the similarities between the $PhrCont$ and $Contains$ predicates, where for each $Contains$ predicate of $phr32$, the most similar $Contains(X, phr31)$ 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 $phr23$ is a *prototype* of all lower-level phrases of $phr32$ (i.e. the sum of the distances between $phr23$ and $\{phr24, phr25, phr26\}$ phrases is minimal, and the other two lower-level phrases $phr21$ and $phr22$ are completely different from all phrases $\{phr24, phr25, phr26\}$). By matching all $\{phr24, phr25, phr26\}$ phrases to $phr23$ RIBL would obtain a relatively high similarity between $phr31$ and $phr32$. This is not what we want, as the internal details of the *whole* high-level phrase $phr31$ are 'responsible' for the similarity/dissimilarity 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.

Given:

- a set $e \in E$ of training examples
- database with background knowledge DB
- test instance i, number of NNs k and starting-clause depth $depth$

1. DISTALL($i, E, DB, k, depth$):$prediction$
2. $DIST = \{\}$
3. calculate starting clause sc_i for i (from DB)
4. for all $e \in E$
5. calculate starting clause sc_e for e (from DB)
6. $dist$=TRANSPORT-NET(e,i,sc_e,sc_i,0,$depth$)
7. $DIST$=$DIST \cup \{< e, dist >\}$
8. $prediction$ = combine predictions of k NNs from $DIST$

9. TRANSPORT-NET($e, i, sc_e, sc_i, linkLevel, depth$):$dist$
10. IF ($linkLevel > depth$): $dist$=(e==i)
11. construct transport network TN with literals from $sc_e(linkLevel)$ and $sc_i(linkLevel)$ containing objects e and i
12. FIND-WEIGHTS($TN, e, i, sc_e, sc_i, linkLevel, depth$)
13. $dist$ is solution to the maximal flow minimal weight problem on TN

14. FIND-WEIGHTS($TN, e, i, sc_e, sc_i, linkLevel, depth$)
15. for all edges $(u, v) \in TN$
16. IF u and v have different functors: $weight(u, v)$=1
17. ELSE
18. $dist = \{\}$
19. for all arguments a_i in (u,v)
20. CASE a_i numeric: $dist_i$=$|u_i - v_i|/range_i$
21. CASE a_i discrete: $dist_i$=(u_i==v_i)
22. CASE a_i object:
23. $dist_i$=TRANSPORT-NET($u_i, v_i, sc_e, sc_i, linkLevel + 1, depth$)
24. $dist = dist \cup dist_i$
25. $weight(u, v)$=average($dist$)

Figure 3.3: The high-level algorithmic description of DISTALL

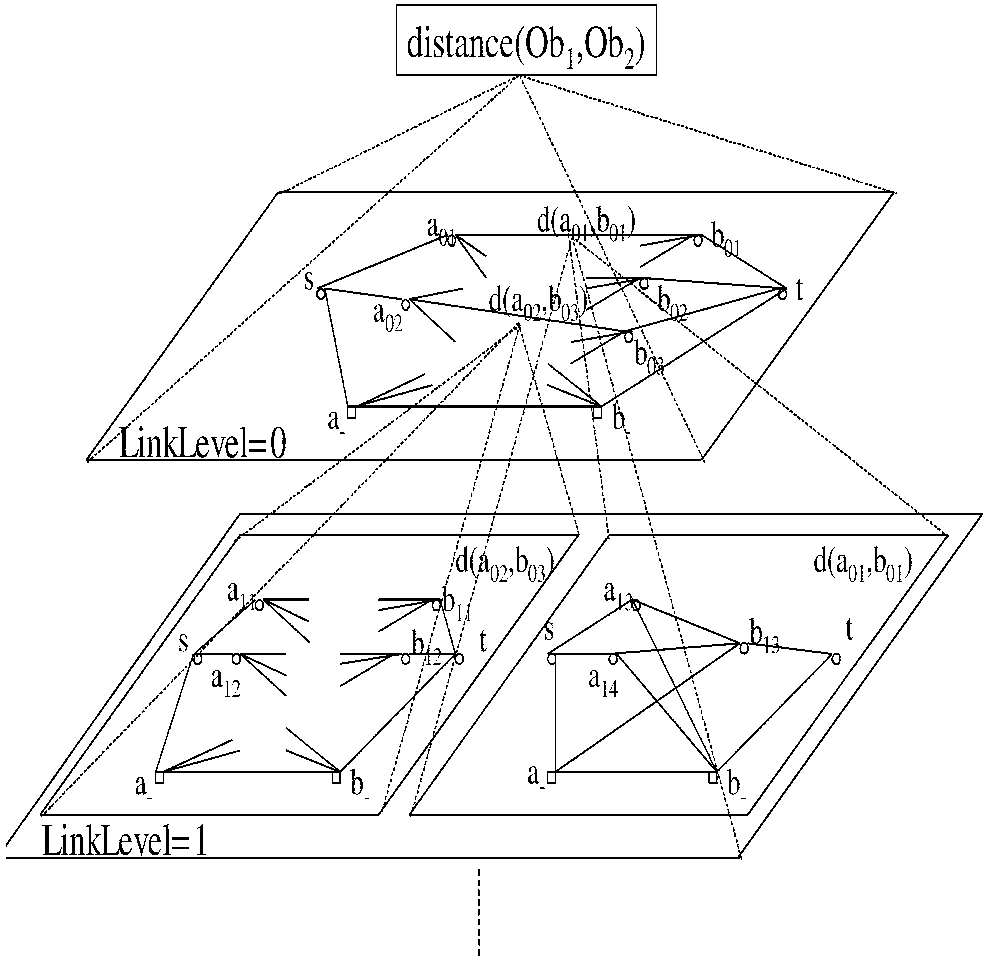

Figure 3.4: Basic principle of DISTALL's similarity measure

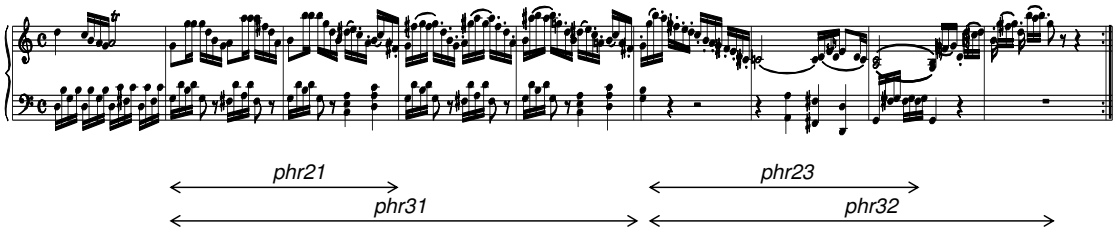

Figure 3.5: Simplified (and incomplete) version of the hierarchical phrase structure of the Mozart Sonata KV. 279 (C major), 1st movement, mm. 31-38. For details see Example 3.4. The purpose of the example is to show distance computation by DISTALL, in this case between phrases $phr21$ and $phr23$.

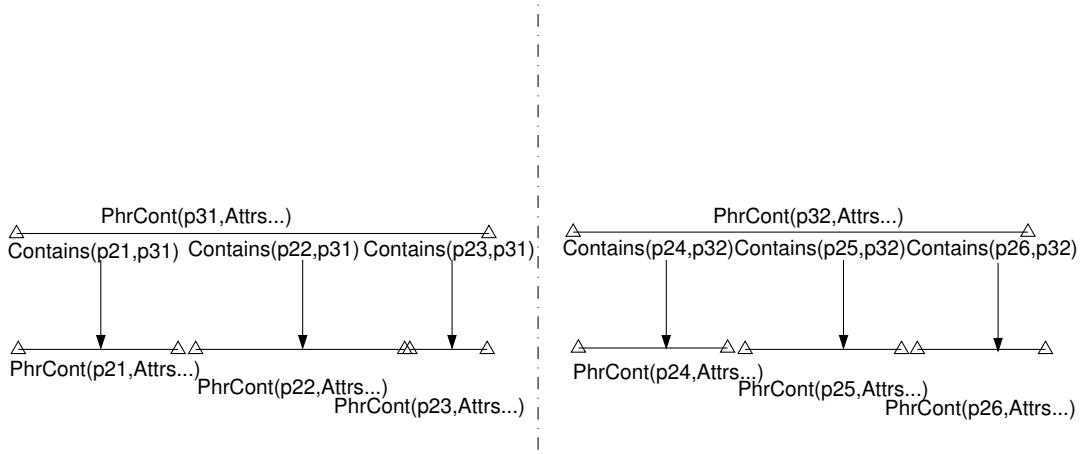

Figure 3.6: An example of relational learning situation: training example (left) and new test case (right)

Chapter 4

Application of Relational IBL to Classical Music

In the last chapter we have discussed our new relational instance-based learner DISTALL. In the following we will introduce a difficult real-world problem from expressive music performance research, which we will use to experimentally assess DISTALL's efficiency: learning, from large numbers of complex performances by concert pianists, to play music expressively.

After defining the context in section 4.1, section 4.2 discusses some of the related previous work undertaken in our research group: induction of note-level performance rules with the PLCG algorithm. Although PLCG has proved to be very successful in the context of its learning task – some rules induced by it are novel musicological discoveries! –, in terms of building a predictable models of expressive performance however, there is an obvious limitation of this work: local, low-level rules can not explain all data in such a complex domain as concert-level expressive piano music. This thesis can thus be regarded as a continuation of this work in terms that we complement the note-level rule model with a predictive models of musical expression at higher levels of the musical structure, namely the level of *phrases*. To this end we apply our relational instance based learner described in last chapter. Section 4.3 discuss the notion of expressive music performance and its representation via performance curves, as well as the data representation we will use for learning. We also show (section 4.3.3) how the training examples for the learner are derived – by decomposing complex performance curves into elementary 'expressive shapes' that can be associated with musical phrases at different levels. Finally, the overall picture of the system used for the experiments described in chapter 5 is given.

This chapter reports about research already published in [Tobudic and Widmer, 2003c,a, 2006; Widmer and Tobudic, 2003]

4.1 Context: Expressive Music Performance

Expressive music performance is the art of shaping a musical piece by continuously varying important parameters like tempo, dynamics, etc. Human musicians do not play a piece of music mechanically, with constant tempo or loudness, exactly as written in the printed music score. Rather, they speed up at some places, slow down at others, stress certain notes or passages by various means, and so on. The most important parameter dimensions available to a performer (a pianist, in particular) are tempo and continuous tempo changes, dynamics (loudness variations), and articulation (the way successive notes are connected). Most of this is not specified in the written score, but at the same time it is absolutely essential for the music to be effective and engaging. As such, expressive performance is a phenomenon of central interest in contemporary (cognitively oriented) musicology.

The work described in the following is another step in a long-term research endeavour that aims at building quantitative models of expressive music performance via Artificial Intelligence and, in particular, inductive machine learning methods [Widmer *et al.*, 2003]. This is to be regarded as basic research. We do not intend to engineer computer programs that generate music performances that sound as human-like as possible. Rather, our goal is to investigate to what extent a machine can automatically build, via inductive learning from 'real-world' data (i.e., real performances by highly skilled musicians), operational models of certain aspects of performance (e.g., predictive models of tempo, timing, dynamics, etc.). In this way, we hope to get new insights into fundamental principles underlying the complex phenomenon of expressive music performance, and contribute to the growing body of scientific knowledge about performance (see [Gabrielsson, 1999] for an excellent overview of current knowledge in this area).

Previous research has shown that computers can indeed find interesting regularities of musical performance. A new machine learning algorithm [Widmer, 2003] succeeded in discovering a small set of simple, robust, and highly general rules that predict a substantial part of the note-level expressive choices of a performer (e.g., whether she will shorten or lengthen a particular note) with surprisingly high precision (see section 4.2).

However, it became equally clear (actually, it was clear from the outset), that the level of single notes is far from sufficient as a basis for a complete model of expressive performance. Music performance is a highly complex activity, with performers tending to shape the music at many different levels simultaneously (see below). The goal of the work described in this text is thus to build a predictive model of musical expression at higher levels of the musical structure, the level of musical *phrases*. An instance-based learning system is described that recognizes performance patterns at various abstraction levels and learns to apply them to new pieces (phrases) by analogy to known performances.

4.2 The Previous Work: Induction of Note-Level Performance Rules with the PLCG Algorithm

The following two sections are recapitulation from [Widmer *et al.*, 2003] and [Widmer, 2003].

4.2.1 Inductive Learning of Classification Rules with the PLCG Algorithm

The induction of classification rules is one of the most studied topics in machine learning. The most common strategy for learning classification rules is known as *sequential covering*, or separate-and-conquer (see e.g. [Fürnkranz, 1999]). According to this strategy, the rules are learned one at a time and, after having learned a rule, all the examples covered by the rule are removed. Ideally, this process is repeated until no examples are left that are not covered by any rule. However, the data of real life problems is *noisy* (i.e. contains errors), meaning that the given data and rule representation languages usually do not even permit the building of perfectly consistent theories; so the state of the art rule learning algorithms perform some kind of pruning to avoid overfitting.

PLCG is a meta-algorithm that relies on such a sequential covering algorithm described above. PLCG uses a rule learning algorithm to induce several partly redundant theories and then combines these theories into one final rule set. In this sense, PLCG is motivated by the successful framework of ensemble methods ([Dietterich, 2000]). Basically, the construction of theories by PLCG proceeds in several stages. It is here that the acronym PLCG can be explained. It stands for partition + learn + cluster + generalize. In a first step, the training examples are partitioned into several subsets. A set of classification rules is then learned from each of them. In the clustering step, the learned rule sets are merged into one large set, and a hierarchical clustering of the rules into a tree of rule sets is performed, where each set contains rules that are somehow similar. Each of these rule sets is then replaced with the least general generalization of all the rules in the set (generalize). Finally, a heuristic algorithm selects the most promising rules from this generalization tree and joins them into the final rule set that is then returned.

The main motivation behind this rather complex procedure is that the advantage of *ensemble learning* – improved prediction accuracy due to the uncorrelated errors of single classifiers – is combined with the benefit of inducing comprehensible theories. In contrast to most common ensemble learning methods such as bagging [Breiman, 1996], stacking [Wolpert, 1992], or boosting [Freund and Schapire, 1996], which only combine the predictions of single classifiers in order to improve prediction accuracy, PLCG combines theories directly and produces one final rule set that can be interpreted. As we will see in the next section, the discovery of comprehensible rules is especially important in the context of our musical project. A more in depth

discussion of the algorithm is given in [Widmer, 2003].

4.2.2 Learning Note-Level Performance Rules

In previous work in our research lab, PLCG was applied to the task of learning note-level performance rules. The term *note-level* reveals that the rules predict how a musician is going to play a particular note in a piece – louder or softer than its predecessor, slower or faster than notated in the musical score, or staccato or legato. The training data used for experiments contained precise information of how each individual note was performed by a pianist, recorded on a Bösendorfer SE290 – a computer-monitored 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.

For the experiments with PLCG, recordings of 13 complete piano sonatas by W. A. Mozart (K.279-284, 330-333, 457, 475, and 533) performed by the Viennese concert pianist Roland Batik were used. The resulting data set comprises more than 106,000 performed notes (of which only melody notes were used, resulting in an effective training set of 41,116 notes), or some 4 hours of music. Each of the melody notes was described by 29 attributes (10 numeric, 19 discrete) that describe both intrinsic properties (such as duration, metrical position etc.) and some aspects of local context of the note (e.g., melodic and rhythmic properties such as the durations of surrounding notes, and the size and direction of the intervals between the note and its predecessor and successor notes etc.).

The following target concepts have been learned: Two tempo concepts – *ritardando* (slowing down) and *accelerando* (speeding up)–, two dynamics concepts – *crescendo* (if the note is played louder than its predecessor) and *diminuendo* (growing softer)–, and two articulation concepts – *staccato* (if a note was sounded for less than 80% of its nominal duration) and *legato* (the note overlaps the following one). From the training set, PLCG learned a small set of 17 simple classification rules (6 rules for tempo changes, 6 rules for local dynamics, and 5 rules for articulation) that predict a large number of the note-level choices of the pianist. Some of the rules represent truly novel musical discoveries (for a more in depth discussion on the rules from the perspective of musicology see [Widmer, 2002]).

4.3 Real-World Task: Learning to Play Music Expressively

As stated above, note-level model can not be sufficient if one aims to build a comprehensive model of expressive music performance. Music performance is a highly complex activity, with performers tending to shape the music at many different levels simultaneously. The goal of the work presented in this thesis is thus 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 the following we show how relational IBL can be applied to learn expressive tempo and dynamics patterns at different phrase levels. After some basic concepts regarding expressive music performance and performance curves, we discuss how phrases are described and represented in our learning system. We further show a method for decomposing complex expression curves into elementary patterns that can be associated with individual phrases (at different phrase levels). Instance-based learning in general, and DISTALL in particular, can then predict timing and dynamics patterns for phrases in a new piece by analogy to the most similar phases in the training set. Generating complex expression curves on 'unseen' test pieces is then a rather straightforward technique, namely the inverse of the curve decomposition problem.

4.3.1 Expressive Music Performance Curves

In section 4.1 we argued that a (skilled) performer, in order to produce interesting and engaging music, relies on varying different parameters, like timing, loudness, articulation etc. In this thesis, we restrict ourselves to piano performances and two of the most important parametric dimensions: *timing* (tempo variations) and *dynamics* (loudness variations). The tempo and loudness variations applied by a musician over the course of a piece (if we can measure them, which is a problem in its own right) can be represented as *tempo* and *loudness curves*, respectively. For instance, Figure 4.1 shows *dynamics curves* – the dynamics patterns produced by three different pianists in performing the same piece. Each point represents the relative loudness with which a particular melody note was played (relative to an averaged 'standard' loudness); a purely mechanical, unexpressive rendition of the piece would correspond to a perfectly flat horizontal line at $y = 1.0$. Variations in tempo can be represented in an analogous way.

Musically trained readers will already notice certain high-level patterns or trends in the curves in Figure 4.1 that seem to correlate with lower- and higher-level phrases of the piece (e.g., a global up-down, *crescendo-decrescendo* tendency over the large phrase that covers the first four bars). Extracting and learning to apply such high-level expressive patterns is the goal of the work presented here.

4.3.2 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. As stated in chapter 2, the phrases and relations between them can be naturally represented in first-order logic.

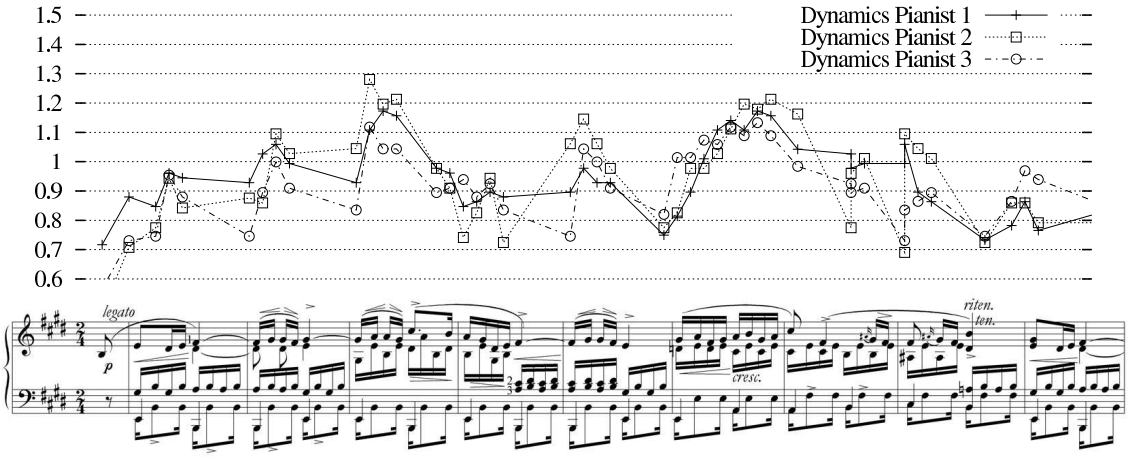

Figure 4.1: Dynamics curves (relating to melody notes) of performances of the same piece (Frédéric Chopin, Etude op.10 no.3, E Major) by three different Viennese pianists (computed from recordings on a Bösendorfer SE290 computer-monitored grand piano).

Consider Figure 4.2. 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 four hierarchical levels, based on a manual phrase structure analysis. The musical content of each phrase is encoded in the predicate *phrCont*. 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 following list gives the names, data types and descriptions of all phrase attributes used by our learning system:

1. **phraseLength** [*numeric*]: the length of the phrase

2. **relPosNumApex** [*numeric*]: the relative position of the highest melodic point (the 'apex') within a phrase

3. **intFirstApex** [*numeric*]: the melodic interval between starting note and apex

4. **intApexLast** [*numeric*]: the melodic interval between apex and ending note

5. **metrStrFirst** [*numeric*]: metrical strength of starting note

6. **metrStrLast** [*numeric*]: metrical strength of ending note

7. **metrStrApex** [*numeric*]: metrical strength of apex

8. **harmonyTriple** [*symbolic*]: the harmonic progression between start, apex, and end

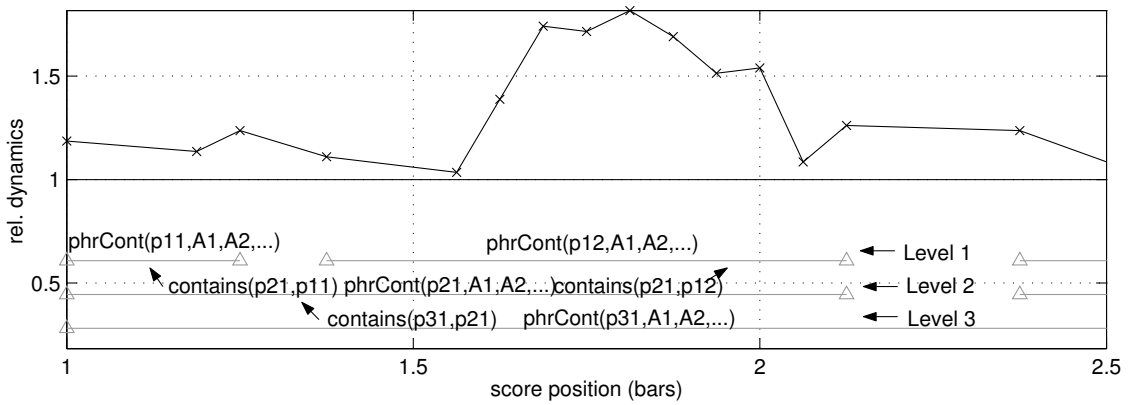

Figure 4.2: Phrase representation used by our relational instance-based learning algorithm.

9. **endsWithCumRhythm** [*boolean*]: state whether the phrase ends with a 'cumulative rhythm'

10. **endsWithCadence** [*boolean*]: state whether the phrase 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*. For the bigger phrases — containing maybe several bars — the high-level attributes in *phrCont* are not sufficient for a full description. But having links to the lower-lever phrases through the *contains* predicate and their detailed description in terms of *phrCont*, 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* and *contains* 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.

4.3.3 Deriving the Training Instances: Multilevel Decomposition of Performance Curves

Our starting material is the scores of musical pieces plus measurements of the tempo and dynamics variations applied by a pianist in a particular performance. These variations are given in the form of *tempo* and *dynamics curves* and represent the local tempo and the relative loudness of each melody note of the piece, respectively.

Both tempo and loudness are represented as multiplicative factors, relative to the average tempo and dynamics of the piece. For instance, a tempo indication of 1.5 for a note means that the note was played 1.5 times as fast as the average tempo of the piece, and a loudness of 1.5 means that the note was played 50% louder than the average loudness of all melody notes.

In addition, the system is given information about the *hierarchical phrase structure* of the pieces, currently at four levels of phrasing. Phrase structure analysis is currently done by hand, as no reliable algorithms are available for this task.

Given a performance (dynamics or tempo) curve, the first problem is to extract the *training examples* for phrase-level learning. Remember that we want to learn how a performer 'shapes' phrases at different structural levels by tempo and dynamics 'gestures'. To that end, the complex curve must be decomposed into basic expressive 'gestures' or 'shapes' that represent the most likely contribution of each phrase to the overall expression curve.

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 previous research in musicology that high-level tempo and dynamics are well characterized by quadratic or parabolic functions [Todd, 1992; Repp, 1992; Kronman and Sundberg, 1987]. 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 approximations. 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 by our definitions, tempo and dynamics curves are lists of multiplicative factors, '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 $N_p = \{n_1, ..., n_k\}$ 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) $E_p = \{expr(n_1), ..., expr(n_k)\}$ 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

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

To illustrate, Figure 4.3 shows the dynamics curve of the last part (mm.31–38) of the Mozart Piano Sonata K.279 (C major), 1st movement, first section. The four-level phrase structure our music analyst assigned to the piece is indicated by the four levels of brackets at the bottom of each plot. The figure shows the stepwise approximation of the expression curve by polynomials at these four phrase levels. The red line in level (e) of the figure shows how much of the original curve is accounted for by the four levels of approximations.

4.3.4 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.

Formally, for a given note n_i that is contained in m hierarchically nested phrases $p_j, j = 1...m$, the expression (tempo or dynamics) value $exp(n_i)$ to be applied to it is computed as

$$exp(n_i) = \prod_{j=1}^{m} f_{p_j}(onset_{p_j}(n_i)),$$

where f_{p_j} is the approximation polynomial predicted as being best suited for the j^{th}-level phrase p_j by the relational instance-based learner.

4.4 The Overall System

Figure 4.4 depicts the overall system used for experiments in the next chapter. Given the scores of the training and the test pieces, our musicologist [1] first analyzed their phrase structure. Each phrase in the training and the test pieces is then represented in FOL as discussed in section 4.3.2. The training performances (for which the system is given expressive tempo and dynamics curves) are then decomposed into 'elementary shapes' as described in section 4.3.3.

[1] Thanks to Werner Goebl for performing the harmonic and phrase structure analysis of the Mozart sonatas.

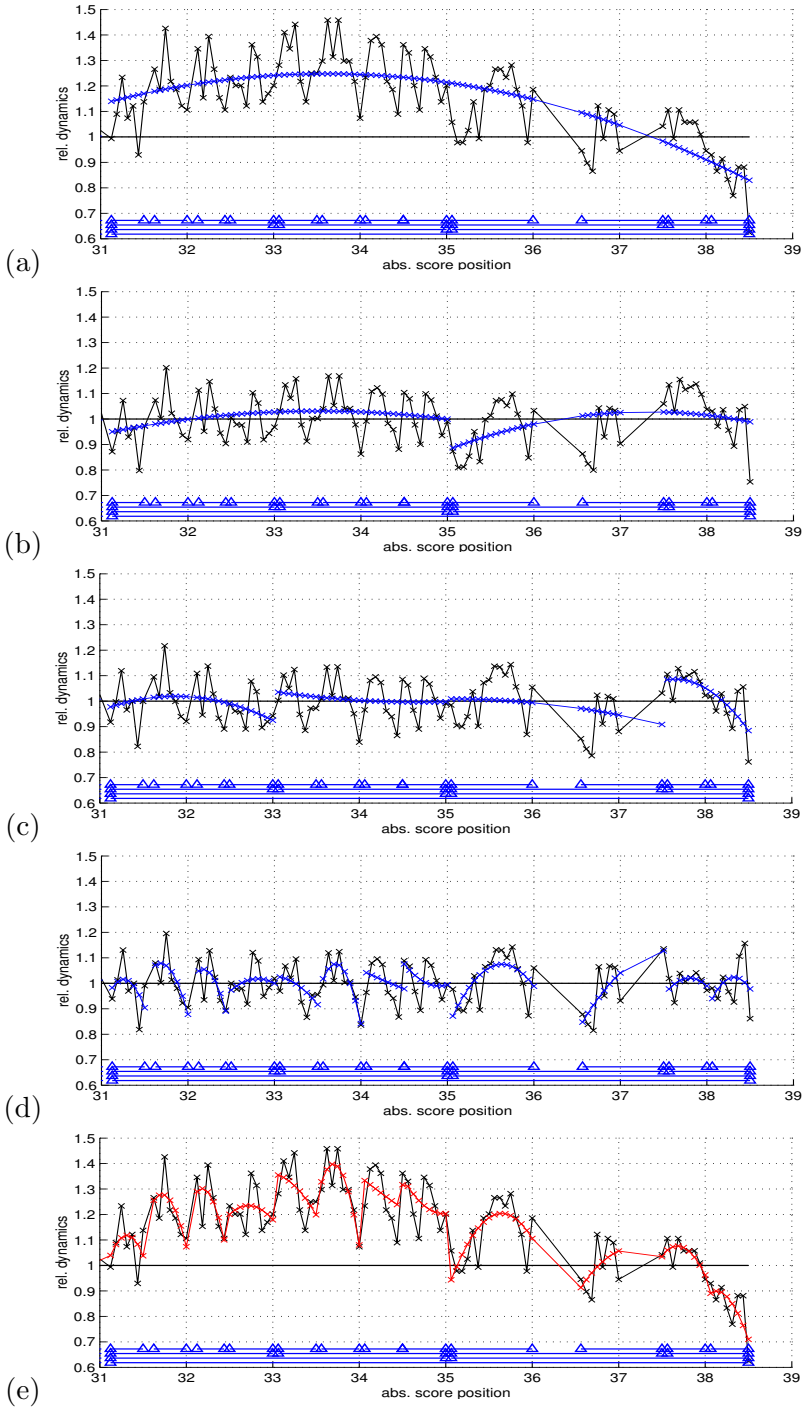

Figure 4.3: [best viewed in color] Multilevel decomposition of dynamics curve of performance of Mozart Sonata K.279:1:1, mm.31–38. Level (a): original dynamics curve plus the second-order polynomial giving the best fit at the top phrase level (blue); levels (b–d) each show, for successively lower phrase levels, the dynamics curve after 'subtraction' of the previous approximation, and the best-fitting approximations at this phrase level; Level (e): 'reconstruction' (red) of the original curve by the four levels of polynomial approximations.

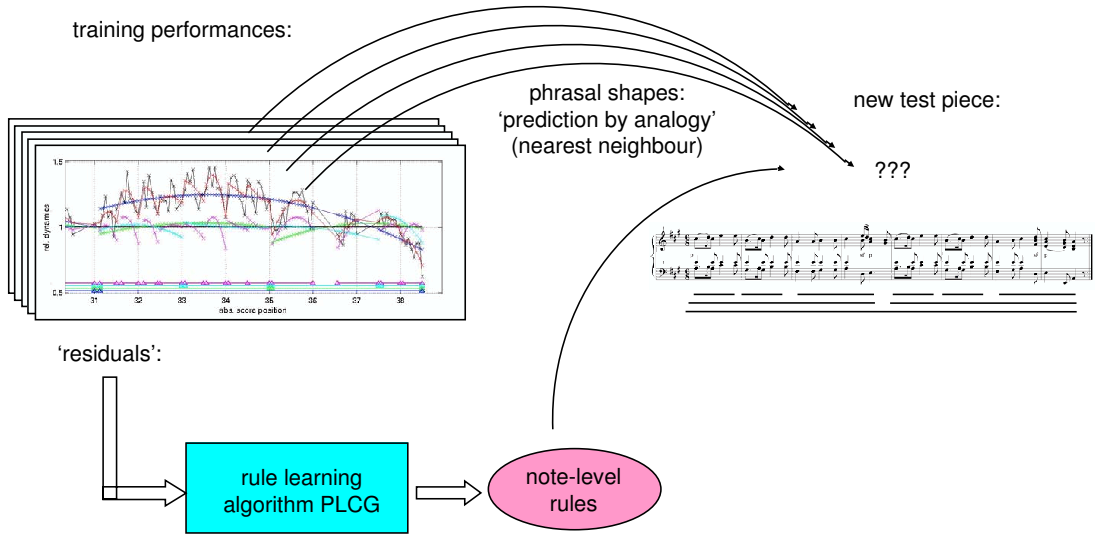

Figure 4.4: The overall system used for experiments.

For each phrase in the test piece, the most similar phrase in the 'training database' is retrieved (via instance-based learning) and its expressive 'shape' is assigned to the test phrase. All phrase shapes in the test piece are then combined as described in section 4.3.4, giving the predicted tempo and dynamics expressive curve for the test piece.

In order to build as complete an expressive model as possible, we also complement the output of the instance-based learning, namely predicted tempo and dynamics expressive curves, with the rules of the note-level learning via PLCG-algorithm. These rules will be learned from what we call the 'residuals', i.e. those aspects of observed performance curves that are not explained by higher-level expressive shapes (see also section 5.1.3 below). Combining the rules with a simple numeric prediction scheme again based on a k-NN algorithm produces a partial model of note-level nuances that predicts local timing and dynamics changes to be applied to some individual notes. Tempo and dynamics deviations for those notes which are 'covered' by rules produced by PLCG are further 'adjusted' by amount suggested by PLCG (i.e., for these notes the system multiplies the expressive values of phrase-level predictions with the note-level expressive values predicted by PLCG). Tempo and dynamics for those notes which are not affected by PLCG-rules will stay unchanged

after PLCG-step, and are thus affected only by instance-based learning.

For those experiments, where the goal was to directly compare different instance-based learning algorithms (DISTALL and RIBL in first place), the PLCG note-level learning was switched off.

Chapter 5

Experimental Evaluation

After introducing a difficult real-world problem in the last chapter, in the following we want to present the experimental evaluation of our approach in general, and the results of our relational instance-based learning algorithm DISTALL in particular. This chapter is divided into three sections.

Section 5.1 presents the empirical results achieved with a staightforward propositional k-nearest neighbor on our problem. Besides being interesting for introducing our dataset, learning and evaluation procedure in its own right, this section also gives the reader a flavor of how difficult the problem really is. While the results of a first quantitative experiment turn out to be rather disappointing, we will show various ways in methodology in which the results can be improved, finally resulting in a system that at least partly makes surprisingly good predictions and even won a prize in a recent 'computer music performance' contest. The material presented in this section has already been published in [Widmer and Tobudic, 2002, 2003; Tobudic and Widmer, 2003c,b].

In section 5.2 we test our relational instance-based learner DISTALL on the same problem and show how it outperforms a simple propositional k-NN. We also present experimental results of the direct comparison between DISTALL and its 'ancestor' RIBL. It turns out that DISTALL produces clearly better results than RIBL, at least on this learning problem. We also present two ways of further improving our prediction results, one of them nicely demonstrating the power of ILP. Finally, a brief discussion of the musical quality of learned performances is presented. A short summary is given in section 5.3. This section contains material already published in [Tobudic and Widmer, 2003a, 2004, 2006].

5.1 Experiment with a Propositional k-NN

5.1.1 The Data

The data used for the experiments was derived from performances of Mozart piano sonatas by the Viennese concert pianist Roland Batik on a Bösendorfer SE 290

Table 5.1: Mozart sonata sections used in experiments (to be read as <sonataName>:<movement>:<section>); *notes* refers to 'melody' notes.

sonata section		notes	phrases at level 1	2	3	4
kv279:1:1	fast 4/4	391	50	19	9	5
kv279:1:2	fast 4/4	638	79	36	14	5
kv280:1:1	fast 3/4	406	42	19	12	4
kv280:1:2	fast 3/4	590	65	34	17	6
kv280:2:1	slow 6/8	94	23	12	6	3
kv280:2:2	slow 6/8	154	37	18	8	4
kv280:3:1	fast 3/8	277	28	19	8	4
kv280:3:2	fast 3/8	379	40	29	13	5
kv282:1:1	slow 4/4	165	24	10	5	2
kv282:1:2	slow 4/4	213	29	12	6	3
kv282:1:3	slow 4/4	31	4	2	1	1
kv283:1:1	fast 3/4	379	53	23	10	5
kv283:1:2	fast 3/4	428	59	32	13	6
kv283:3:1	fast 3/8	326	53	30	12	3
c kv283:3:2	fast 3/8	558	79	47	19	6
kv332:2	slow 4/4	477	49	23	12	4
Total:		5506	714	365	165	66

computer-controlled grand piano. The SE 290 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.

A manual, multi-level phrase structure analysis of some sections of these sonatas was carried out manually by a musicologist. Phrase structure was marked at four hierarchical levels. The resulting set of annotated pieces available for our experiment is summarized in Table 5.1. As the reader can see, we have compiled an substantial set of music for our evaluation experiments, consisting of 16 pieces with over 5000 melody notes and 1310 musical phrases. The 16 pieces comprise about 30 minutes of piano music. Furthermore, the scores and performances are quite complex and different in character; automatically learning expressive strategies from them is a challenging task.

In the following section we will discuss our first experiment with a propositional k-NN, where we used one nearest neighbor for prediction. In the further text we will then show various improvements of the learning procedure, including varying numbers of neighbors and phrase levels, making more homogeneous training sets, and including the learning of local rules via the PLCG algorithm.

	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	.0383	.0409	.1643	.1543	.6170	.0348	.0420	.1220	.1496	.3095
kv279:1:2	.0318	.0736	.1479	.1978	.4157	.0244	.0335	.1004	.1317	.2536
kv280:1:1	.0313	.0275	.1432	.1238	.6809	.0254	.0222	.1053	.1071	.4845
kv280:1:2	.0281	.0480	.1365	.1637	.4517	.0250	.0323	.1074	.1255	.3124
kv280:2:1	.1558	.0831	.3498	.2002	.7168	.0343	.0207	.1189	.1111	.7235
kv280:2:2	.1424	.0879	.3178	.2235	.6980	.0406	.0460	.1349	.1463	.4838
kv280:3:1	.0334	.0139	.1539	.0936	.7656	.0343	.0262	.1218	.1175	.5276
kv280:3:2	.0226	.0711	.1231	.2055	.4492	.0454	.0455	.1365	.1412	.3006
kv282:1:1	.1076	.0480	.2719	.1751	.7454	.0367	.0390	.1300	.1303	.3166
kv282:1:2	.0865	.0508	.2420	.1759	.6887	.0278	.0479	.1142	.1571	.2560
kv282:1:3	.1230	.0757	.2595	.2364	.6698	.1011	.0529	.2354	.1741	.8104
kv283:1:1	.0283	.0236	.1423	.1206	.5907	.0183	.0276	.0918	.1196	.2409
kv283:1:2	.0371	.0515	.1611	.1625	.4469	.0178	.0274	.0932	.1197	.1972
kv283:3:1	.0404	.0314	.1633	.1311	.6061	.0225	.0216	.1024	.1083	.4260
kv283:3:2	.0424	.0405	.1688	.1466	.5255	.0256	.0261	.1085	.1130	.2961
kv332:2	.0919	.0824	.2554	.2328	.5599	.0286	.0436	.1110	.1529	.1684
WMean:	.0486	.0506	.1757	.1662	.5584	.0282	.0332	.1108	.1285	.3192

Table 5.2: Results of piece-wise cross-validation experiments with a propositional k-NN ($k = 1$). Measures subscripted with D refer to the 'default' (mechanical, inexpressive) performance, those with L to the performance produced by the learner. $WMean$ is the weighted mean (individual results weighted by the relative length (number of notes) of the pieces.

5.1.2 Quantitative Results of A First Experiment

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. We used a propositional k-NN with $k = 1$. The learned phrase-level shapes were then applied to the test piece, and the following measures were computed: the *mean squared error* of the learner's predicted curve relative to the actual performance curve produced by the pianist ($MSE = \sum_{i=1}^{n} (pred(n_i) - expr(n_i))^2/n$), the *mean absolute error* ($MAE = \sum_{i=1}^{n} |pred(n_i) - expr(n_i)|/n$), and the *correlation* between predicted and 'true' curve. MSE particularly punishes curves that produce a few extreme 'errors'. MSE and MAE were also computed for a *default* curve that would correspond to a 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 5.2, where each line gives the results obtained on the respective test piece when all others were used for training. The last line ($WMean$) shows the weighted mean

performance over all pieces (individual results weighted with the relative length of the pieces).

At a first glance, the results look rather disappointing. We are interested in cases where the *relative errors* (i.e., $\text{MSE}_L/\text{MSE}_D$ and $\text{MAE}_L/\text{MAE}_D$) are less than 1.0, that is, where the curves predicted by the learner are closer to the pianist's actual performance than a purely mechanical rendition. In the dynamics dimension the learner produces encouraging results, in 11 out of 16 cases for MSE and in 12 out of 16 for MAE. Tempo seems basically unpredictable: only in 5 (MSE) and 3 (MAE) cases, respectively, did learning produce an improvement over no learning, at least in terms of these purely quantitative, unmusical measures. Also, the correlations vary between 0.77 (kv280:3:1, dynamics) and only 0.17 (kv332:2, tempo).

Averaging over all 16 experiments shows the same behavior, dynamics seems learnable at least to some extent (the weighted relative errors being $RMSE = 1.041$ and $RMAE = 0.945$ respectively), while tempo seems unpredictable (all relative errors are above 1.0).

5.1.3 Improving the Results

The above results were rather disappointing. Even keeping in mind that artistic performance of difficult music like Mozart sonatas is a complex and certainly not entirely predictable phenomenon, we had hoped that there would be something predictable about phrase-level tempo and dynamics that a learner could pick up. But the above results are not the end of the story, and in the following sections we explore ways in which they can be improved – at the end we will end up with a system that at least partly makes surprisingly good predictions and even won a prize in a performance contest.

More homogeneous training set

One way of improving the results is by noting that Mozart piano sonatas are highly complex music, with a lot of diversity in character. Splitting this set of rather different pieces into more homogeneous subsets and performing learning within these subsets should make the task easier for the learner. For instance, it is known in musicology that absolute tempo has quite an impact on what performance patterns sound acceptable. And indeed, it turns out that simply separating the pieces into fast and slow ones and learning in each of these sets separately considerably increases the number of cases where learning produces an improvement over no learning, both in the dynamics and the tempo domain. Table 5.3 summarizes the results in terms of wins/losses between learning and no learning for both learning settings. The improvement is obvious. However, the tempo domain is still a problem, with only 7 wins out of 16 cases.

	MSE/dynamics	MAE/dynamics	MSE/tempo	MAE/tempo
all pieces	11+/5-	12+/4-	5+/11-	3+/13-
slow / fast	14+/2-	14+/2-	7+/9-	7+/9-

Table 5.3: Summary of wins vs. losses between learning and no learning; + means curves predicted by the learner better fit the pianist than a flat curve (i.e. relative error < 1), - means opposite. First line: piece-level cross validation over all pieces; second line: learning on fast and slow pieces separately

Varying numbers of neighbours and phrase levels

All the results so far were produced by a k-NN learner with k=1. We initially chose k=1 because we could not think of a meaningful way to combine the predictions of several neighbours – simple averaging of polynomial coefficients or curves seemed not very sensible from a musical point of view. But in experiments it turned out that in the absence of a more informed combination strategy, even simple averaging of several neighbours' predictions can substantially improve the quality of the predicted curves. Table 5.4 shows the results obtained by increasing the number k of neighbours used in the prediction. The dynamics results in particular show substantial improvement – the $RMSE$ (MSE_L/MSE_D) drops from 1.041 for $k = 1$ to 0654 for $k = 10$, the $RMAE$ from 0.946 to 0.787, and the correlation improves. There is also some improvement in the tempo dimension, with at least the $RMSE$ dropping below 1.0. The attendant slight drop in correlation indicates that with increasing k, the learner tends to reproduce fewer of the local tempo changes of the pianist, while improving the overall fit at higher levels.

In further experiments, it turned out that the highest level of phrasing that was marked by our musicologist – extended phrases that span several, sometimes many, bars – was not well mirrored in the performances by our pianist. Ignoring the highest phrase level and learning and predicting only at the lower three phrase levels leads to even better result, as shown in last 5 rows in table 5.4. The results for the same learner variants in terms of wins vs. losses between learning and no learning is given in table 5.5. Finally, learning beats no learning even in the tempo dimension for both, MSE and MAE (3 phrase levels, 10NN).

Improving the musical quality by learning local rules

Trying to predict how a pianist 'shapes' phrases tend to reconstruct the larger trends in a performance curve quite well, but cannot describe all the detailed local nuances added by the pianist, e.g., the emphasis on particular notes. Local nuances will be left over in what we call the *residuals* – the tempo and dynamics fluctuations left unexplained by the phrase-level polynomials. These can be expected to represent a mixture of noise and meaningful or intended local deviations.

In order to also learn a model of these intended deviations, we applied a rule

	dynamics					tempo				
Variant	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$
4 levels, 1NN	.0486	.0506	.1757	.1662	.5584	.0282	.0332	.1108	.1285	.3192
4 levels, 2NN	.0486	.0395	.1757	.1520	.5637	.0282	.0299	.1108	.1239	.3105
4 levels, 3NN	.0486	.0354	.1757	.1466	.5918	.0282	.0297	.1108	.1231	.2871
4 levels, 5NN	.0486	.0336	.1757	.1424	.6114	.0282	.0292	.1108	.1208	.2786
4 levels, 10NN	.0486	.0318	.1757	.1382	.6166	.0282	.0276	.1108	.1157	.2960
3 levels, 1NN	.0486	.0385	.1757	.1447	.5980	.0282	.0318	.1108	.1231	.3260
3 levels, 2NN	.0486	.0359	.1757	.1446	.5913	.0282	.0303	.1108	.1203	.2963
3 levels, 3NN	.0486	.0341	.1757	.1425	.6017	.0282	.0302	.1108	.1215	.2767
3 levels, 5NN	.0486	.0324	.1757	.1399	.6167	.0282	.0270	.1108	.1156	.3454
3 levels, 10NN	.0486	.0312	.1757	.1380	.6096	.0282	.0271	.1108	.1136	.2937

Table 5.4: Varying the numbers of neighbours and phrase levels. The table shows weighted means over all test pieces.

learning algorithm to the residuals. The goal was to induce note-level rules that predict when the pianist will significantly lengthen or shorten a particular note relative to its context, or play it significantly louder or softer. The learning algorithm used was PLCG (see section 4.2), which has been shown to be quite effective in distinguishing between signal and noise and discovering reliable rules when only a part of the data can be explained [Widmer, 2003]. Combining the learned rules with a simple numeric prediction scheme again based on a k-NN algorithm produces a partial model of note-level nuances that predicts local timing and dynamics changes to be applied to some individual notes.

Combining these note-level predictions with the phrase-level predictions yields an additional slight reduction in MSE and MAE both for tempo and dynamics, but the difference is almost negligible (though consistently in favour of the combined learner). The interesting fact is that the correlation values improve significantly. For instance, combining the note-level model with the $3\ levels, 10\ NN$ learner yields (weighted mean) correlations of 0.6182 for dynamics and 0.3588 for tempo – for tempo in particular, this is significantly higher than any of the values in table 5.4. Obviously, the note-level model captures some important local choices of the pianist (which also strongly contribute to the musical quality of the performance).

A fairer comparison

A final way of 'improving' the results is to note that the error measures we used so far may not be quite appropriate. What was compared was the performance (tempo or dynamics) curve produced by composing the polynomials predicted by the learner, with the curve corresponding to the pianist's actual performance. However, what the k-NN learner learned from was not the actual performance curves, but an *approximation*, namely, the polynomials fitted to the curve at various phrase

Variant	MSE/dynamics	MAE/dynamics	MSE/tempo	MAE/tempo
4 levels 1NN	11+/5-	12+/4-	5+/11-	3+/13-
4 levels 2NN	12+/4-	13+/3-	7+/9-	4+/12-
4 levels 3NN	12+/4-	14+/2-	6+/10-	2+/1=/13-
4 levels 5NN	14+/2-	14+/2-	8+/8-	5+/11-
4 levels 10NN	14+/2-	15+/1-	10+/6-	6+/10-
3 levels 1NN	12+/4-	14+/2-	7+/9-	6+/10
3 levels 2NN	13+/3-	15+/1-	6+/10-	5+/11-
3 levels 3NN	14+/2-	15+/1-	8+/8-	5+/11-
3 levels 5NN	15+/1-	15+/1-	9+/7-	7+/9-
3 levels 10NN	15+/1-	15+/1-	11+/1=/4-	9+/7-

Table 5.5: Summary of wins vs. losses between learning and no learning for different numbers of phrase levels and neighbours.

levels. And maybe this approximation is not very good to begin with. This is partly confirmed by a look at table 5.6, which summarizes how well the four-level decompositions (without the residuals) reconstruct the respective training curves. [1] The dynamics curves are generally better approximated by the four levels of polynomials than the tempo curves, and the difference is dramatic. That may explain in particular why our tempo results were so poor.

The finding implied by table 5.6 has implication for musicology, where it has hitherto been believed (but never systematically tested with large numbers of real performances) that quadratic functions are a reasonable model class for expressive timing (e.g., [Todd, 1989; Windsor and Clarke, 1997]). But it also suggests that the above way of computing prediction error was not entirely fair. It would seem more appropriate to compare the predicted curves not to the actual performance curve, but to the approximation curve that is implied by the four levels of quadratic functions that were used as training examples. [2] Correctly predicting these is the best the learner could hope to achieve. Table 5.7 shows the error figures we obtain in this way, for the four variants of the k-NN learners described above.

As can be seen, the situation indeed now looks better for our learner (compare this to table 5.4 above). Note the substantially higher correlations in the tempo domain - it is obviously easier to predict approximated curves than real curves. There is also some improvement in terms of the number of wins vs. losses against the default. For example, with 3 levels of phrasing and 10 nearest neighbours (last line in the table 5.7) we get win/loss rations of 15+/1- for dynamics (both for MSE and MAE) and 11+/1=/4- (MSE) and 10+/6- (MAE) for tempo.

[1] That is, we look not at the performance of the learning system, but only at the effectiveness of approximating a given curve by four levels of quadratic functions.

[2] Actually, the most direct comparison would be between the predicted 'true' polynomial coefficients, but numeric errors and correlations at this level would be hard to interpret intuitively or musically.

	MSE_D	MSE_P	RMSE	MAE_D	MAE_P	RMAE	$Corr_L$
dynamics	.0486	.0049	.1008	.1757	.0501	.2851	.9397
tempo	.0282	.0144	.5106	.1108	.0755	.6814	.6954

Table 5.6: Summary of fit of four-level polynomial decomposition on the training data. Measures subscripted with D refer to the 'default' (mechanical, inexpressive) performances (repeated from table 5.2), those with P to the fit of the curves reconstructed by the polynomial decompositions.

	dynamics					tempo				
Variant	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$
4 levels, 1NN	.0437	.0457	.1665	.1543	.5936	.0141	.0190	.0811	.0959	.4517
4 levels, 10NN	.0437	.0268	.1665	.1249	.6571	.0141	.0135	.0811	.0831	.4137
3 levels, 1NN	.0437	.0335	.1665	.1309	.6369	.0141	.0177	.0811	.0878	.4628
3 levels, 10NN	.0437	.0262	.1665	.1246	.6489	.0141	.0130	.0811	.0806	.4155

Table 5.7: Summary of errors resulting from comparing the learner's predictions to the 'reconstructed' training curve rather than the actual performance curve produced by the pianist. Shown are weighted means over all training expamples.

Of course, the 'trick' of changing the definition of error does not change the musical quality of the results, but it gives a more realistic picture of the capabilities of the propositional nearest-neighbour learning in this domain.

5.1.4 Musical Results

The musical quality of the results is hard to describe in a text. Generally, the quality varies strongly between pieces, and even within pieces – passage that are musically sensible are sometimes followed by rather extreme errors, at least in musical terms. One incorrect shape can seriously compromise the quality of a composite performance curve that would otherwise be perfectly musical. The qualitative measures MSE, MAE, and correlation are not necessarily indicative of the quality of the listening experience.

Figure 5.1 gives the reader an impression of predictions of the learning system on two examples for dynamics domain, the first section of the third movement of the Mozart piano sonata K.283 (Figure 5.1 (a)) and the second movement of the Mozart piano sonata K.332 (Figure 5.1 (b)). The first example is the case where prediction worked quite well concerning both, the higher-level aspects and the local, lower-level patterns. On the other hand, the dynamics prediction for K.332 is rather poor, which partly can be explained by the fact that K.332 is a piece drastically different in character than all other pieces from our dataset.

Tempo and dynamics prediction curves as shown in Figure 5.1 can be used to

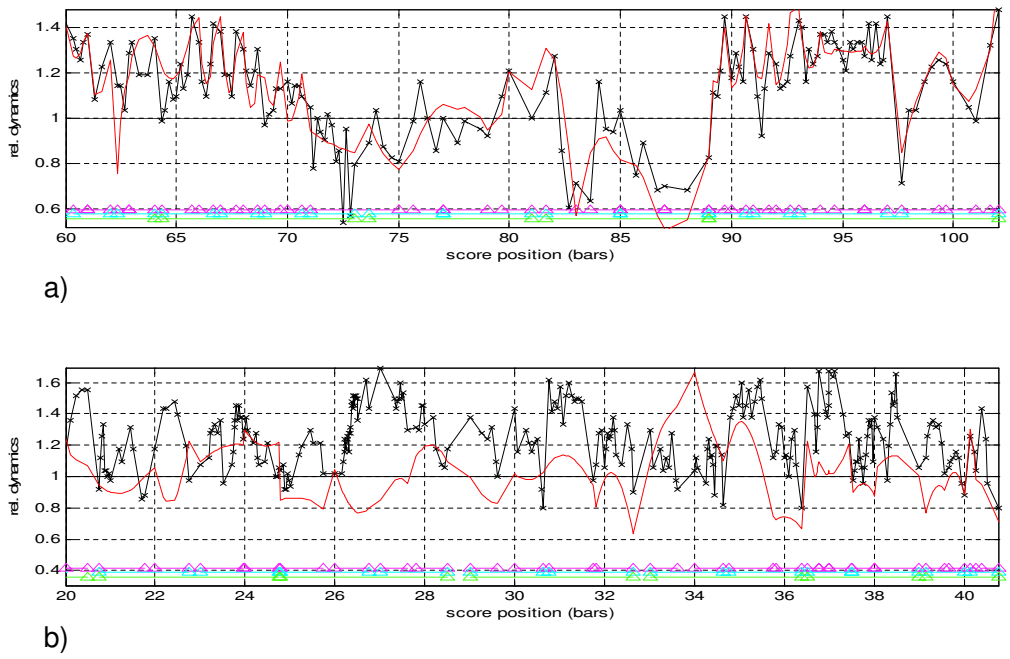

Figure 5.1: Two examples of performance curves and the predictions of the learning system for the dynamics domain: (a) Mozart Sonata K.283, third movement, first section; (b) Mozart Sonata K.332, second movement.

'automate' the generation of expressive music. We used tempo and dynamics predictions by the 1-NN learning algorithm for Mozart piano sonata K.280 in F major to produce the recording of a fully computer generated expressive performance. The recording was submitted to an International Computer Piano Performance Rendering Contest [3] (RENCON'02) in Tokyo in September 2002, where it won Second Prize behind a rule-based rendering system that had been carefully tuned by hand. The rating was done by a jury of human listeners. While this result does in no way 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 system.

[3] yes, there is such a thing ...

5.2 Experimental Evaluation of DISTALL (vs. propositional k-NN and RIBL)

In the following section we present detailed empirical results achieved with DISTALL on the same learning problem discussed so far. We also provide a direct comparison with the results achieved with the simple propositional k-NN presented in the last section, as well as those achieved by the DISTALL's 'ancestor' RIBL. Two ways of further improving results of our learning system are further discussed, one of them exploring the representational power of first-order logic. This section contains material already published in [Tobudic and Widmer, 2003a, 2004, 2006].

5.2.1 The Data

The data used for the experiments was the same as described in the last section and presented in table 5.1. Since our experiments consistently showed that the highest level of phrasing was not well mirrored in the performances by our pianist – as discussed in section 5.1.3 – we ignore this level of phrasing in further experiments. Thus, all results presented in the following sections are achieved by learning and predicting only at the lower three phrase levels.

5.2.2 A Quantitative Evaluation of DISTALL

As in the experiments with the propositional k-NN, 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. For details, see section 3.5). 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 4.3.4.

In evaluating DISTALL's performance, there is a single change in the definition of error measures, compared with the first experiments with the propositional k-NN, discussed in section 5.1.2 and presented in table 5.2. The curves predicted by the system were compared not to the pianist's actual performance curves, but 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. As discussed in section 5.1.3, 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. Error measures based on the approximation curves give in our eyes thus a more realistic picture of the capabilities of the nearest-neighbour approach in this domain. Of course, in comparing DISTALL's performance with those of propositional k-NN

	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

Table 5.8: 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.

and RIBL later in this section, this error measure will be applied to all algorithms equally.

The same performance measures as discussed in section 5.1.2 were computed: the MSE (*mean squared error*) of the system's prediction on the piece relative to the approximation curve, the MAE (*mean absolute error*), and the *correlation* between predicted and approximated curve. Again, MSE and MAE were also computed for a *default* curve that would correspond to a purely mechanical, unexpressive performance, which allows us to judge if learning is really better than just doing nothing. The results of the experiment are summarized in table 5.8, where again 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).

Again, 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

	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

Table 5.9: 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 5.8 (last row), those from propositional k-NN are reproduced from the third row from table 5.7. The cases where a learner is better than default are printed in bold.

	MSE/dynamics	MAE/dynamics	MSE/tempo	MAE/tempo
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-

Table 5.10: Summary of wins vs. losses, over all test pieces, between learning and no learning for the propositional k-NN, RIBL and DISTALL. 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.

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).

5.2.3 DISTALL vs. RIBL and Propositional k-NN

In order to put these results into context, we present a comparison with RIBL [Emde and Wettschereck, 1996]. Since RIBL is not publicly available, in the experiments

[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.

of this section 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 discussed above. It has been shown that a richer relational representation need not always be a guarantee for better generalization performance [Džeroski *et al.*, 1998].

The experimental setup and the error measures stayed the same as described 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 5.9 shows the results in terms of weighted mean errors over all test pieces. The equivalent results for DISTALL are repeated from Table 5.8 (last row). A summary of wins/losses between learning and no learning for all learners is given in Table 5.10.

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 (see section 3.5), 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 4h and 5h for RIBL and DISTALL, respectively). Certainly, the performance measures expressed in terms of weighted mean numbers given in Table 5.9 do not imply that DISTALL outperforms RIBL on all pieces in the dataset. It turns out that DISTALL-predicted curves 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).

5.2.4 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 still rather poor. 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 5.10) is still rather discouraging. In the following sections we explore two ways in which the results can be improved,

	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 5.11: 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.

	MSE/dynamics	MAE/dynamics	MSE/tempo	MAE/tempo
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-

Table 5.12: 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).

one of them nicely demonstrating the power of ILP.

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 simple phrase representation described in section 4.3.2. But describing phrases with this set of features is by no means the optimal way of describing musical phrases. Our experiments indicated that the results can be improved substantially by adding various statistical attributes that capture some global musical characteristics of phrases. By adding features describing global tempo, the presence of trills, 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' of the melodic line), we constructed a more representative feature set. The effect of this change on all 3 learners can be seen in Tables 5.11 and 5.12. As can be seen, all learners generally benefit from this richer phrase representation. Interestingly, DISTALL achieves the greatest performance gain in terms of wins/losses in the problematic tempo domain.

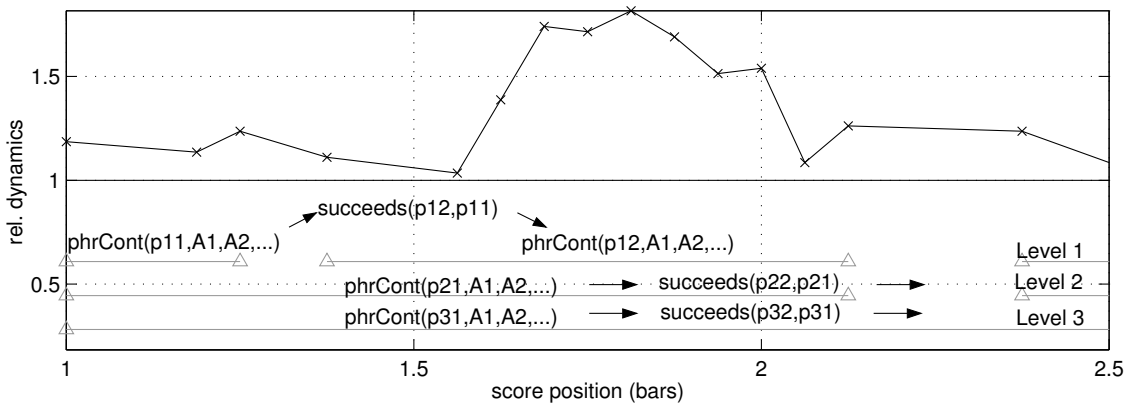

Figure 5.2: New phrase representation with the temporal *succeeds/2* predicate.

The Power of FOL: Temporal Relationships

The results of predicting the performer's tempo decisions are not satisfactory, 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 5.9 and 5.11). 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 learners 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 5.2 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 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

	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

Table 5.13: Results, by sonata sections, of cross-validation experiment with DISTALL (*depth=4*, *k=1*) and new temporal phrase representation (see Figure 5.2). The cases where DISTALL is better than the default are printed in bold.

are given in Table 5.13. The experiments with the new representation were also rerun with RIBL (with the same setting, *depth=4*, *k=1*). Table 5.14 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 5.15 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 in the tempo dimension. We suspect that such a result in a complex artistic domain would not be possible without the expressive power of FOL.

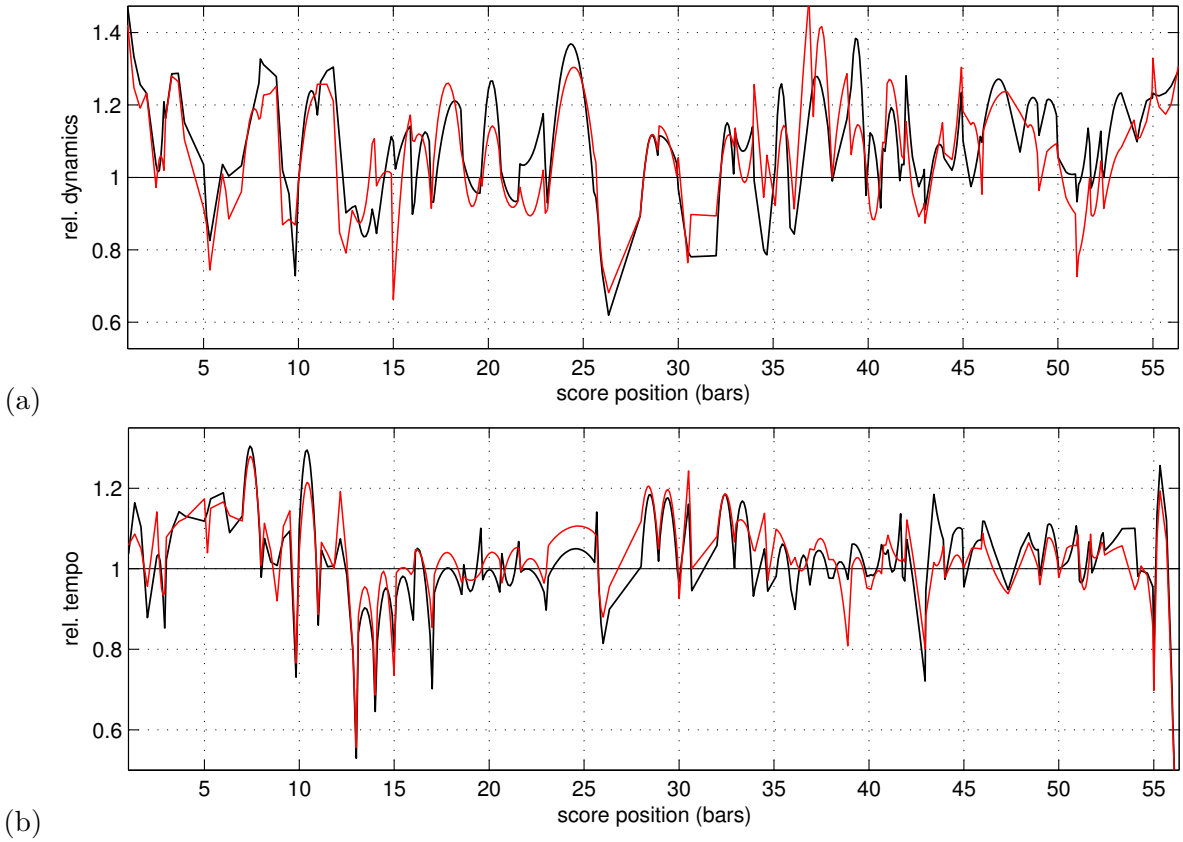

Figure 5.3: Expressive curves for dynamics (a) and tempo (b) of the Mozart Sonata KV.280, 1^{st} movement, 1^{st} section. Black curves represent approximations of the actual expression curves produced by the pianist, implied by the three levels of quadratic functions. DISTALL *Temp*'s predictions are given in red.

	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 5.14: 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.

	MSE/dynamics	MAE/dynamics	MSE/tempo	MAE/tempo
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-

Table 5.15: 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.

5.2.5 Musical Results

A look at Table 5.13 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 of the slow pieces exhibit astonishingly high correlations in both dimensions (e.g. KV.280, 2^{nd} movement, 1^{st} 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, 1^{st} movement, 1^{st} section is shown in Figure 5.3. For this performance, the correlations between learner and pianist are 0.79 for dynamics and 0.90 for tempo.

5.3 Summary

In this chapter we discussed the experimental results of our system applied on difficult real-world learning task derived from music performance research. For our experiments we compiled a substantial dataset containing 16 sections of Mozart piano sonatas comprising of about 30 minutes of piano music. The music is played by the Viennese concert pianist Roland Batik on a Bösendorfer SE 290 computer-controlled grand piano. We derived objective performance measures based on MSE, MAE and correlations between predicted dynamics and tempo curves and actual dynamics and tempo curves 'played' by the pianist. We also compared the effect of learning and 'doing nothing', i.e. purely mechanical performances.

The results of the first experiments with straightforward k-NN were rather disappointing, showing how difficult the learning problem actually is. Especially the tempo domain turned out rather unpredictable: purely mechanical performances were closer to the pianist's tempo curves clearly better than the tempo curves produced by the k-NN. We then showed various improvements of the system including grouping training and test sets differently (according to the absolute tempo), varying numbers of k in the k-NN algorithm and taking into account a subset of the different phrase levels considered in the initial experiment. We further improved musical quality of the performances by learning local, note-level rules with the PLCG algorithm. In the end we showed a system which made surprisingly good predictions in both domains at least on some pieces from our dataset. After learning from other pieces, tempo and dynamics predictions for one such a piece – Mozart piano sonata K.280 in F major – were used to produce the recording which can be considered as the fully computer generated expressive piano performance. The recording won second prize at the RENCON'02 in Tokyo, where the rating was done by a jury of human listeners.

We then applied our new relational instance-based learning algorithm DISTALL to the same problem. We showed that by applying DISTALL, a substantial progress in the learning performance over k-NN can be made. As the main contribution of this chapter to the field of machine learning we consider the direct comparison between DISTALL and RIBL. It tuned out that DISTALL was indeed able to outperform RIBL in terms of practically all performance measures we derived for our learning problem. DISTALL has been able to improve learning performance even in tempo domain, e.g. resulting in significantly more wins over mechanical performance in terms of MAE for tempo (11+/5-) than RIBL (8+/8-) (Table 5.10).

We have also shown two additional ways in further improving our results. First, we improved the phrase representation by including more statistical features. Second, we demonstrated the power of FOL and reformulated our relational predicate in the way that it includes temporal relationships between phrases. Especially the second improvement resulted in a further performance leap in terms of all error measures. Direct comparison with RIBL revealed again the DISTALL's superiority on this learning problem: After applying the same phrase representation to the both

learners, DISTALL's wins/losses for tempo domain were 12+/4- in the case of MSE (9+/7- by RIBL) and 14+/2- in the case of MAE (11+/5- by RIBL) (Table 5.15). While further experiments need to be done in order to further evaluate strengths and weaknesses of both learners, we consider this experimental success of DISTALL vs. RIBL on the difficult real-world task as a small contribution to the machine learning community.

Chapter 6

A By-product: Consistency in Piano Performances

In the previous chapter we have presented experimental results comparing DISTALL, RIBL and a propositional k-nearest neighbor algorithm. Beside the task of building computational models of expressive music performance, as described above, the similarity measure built into our relational instance-based learner can also find other useful applications in musicology. In the following sections we will discuss one possibility: Applying DISTALL's similarity measure to segments of musical scores, we will try to assess the level of 'consistency' of a Viennese concert pianist playing different pieces from a corpus of Mozart piano sonatas. Although there is a work in musicology examining the consistency of playing immediate repetitions in piano music ([Palmer, 1989]), with the help of DISTALL's similarity measure we are able to define a concept of 'consistency' which goes beyond simple score repetitions. Actually, we can assess the level of performer consistency between any two phrases, regardless of similarity/dissimilarity of pieces they belong to.

The further text is organized as follows. After a short introduction to the task we are interested in, we describe two basic ingredients in assessing the consistency level of a performer: score- and performance based similarity measure (Section 6.2). Section 6.3 describes the methodology. Experimental results are given in Section 6.4. Limitations and further directions are discussed in Section 6.5.

6.1 Expressive Performance and Stylistic Consistency

The wealth of research on expression in music performance is represented in overview papers by [Palmer, 1997] and [Gabrielsson, 1999, 2003]. Skilled musicians are able to shape their performances with extremely high precision. Especially the shaping of immediate repetitions appears to be very similar (the differences to be within a few milliseconds, see e.g., [Palmer, 1989]). However, to our knowledge there were no

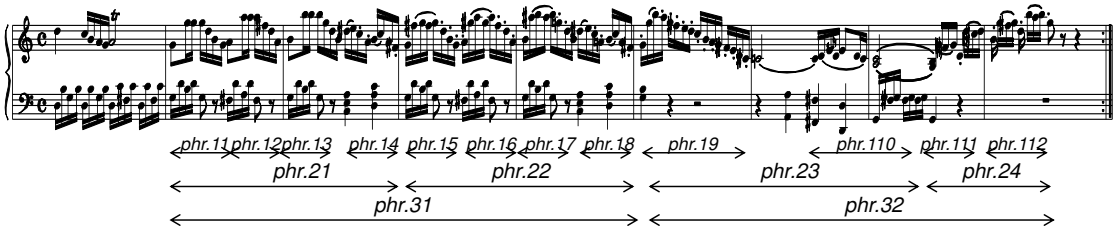

Figure 6.1: Score of the Mozart Sonata KV. 279 (C major), 1st movement, mm. 31-38. The hierarchical, three level phrase structure of the passage is indicated by brackets at the bottom of the figure.

studies that focused on the 'consistency' of particular performers in a more general sense. We define consistency as the property to perform phrases that are similar according to the score in a similar way (regarding timing and dynamics). In this paper we will provide (circumstantial) evidence of consistency of a Viennese concert-level pianist playing different pieces from a large corpus of Mozart piano sonatas. We examine the consistency at the level of musical phrases and try to empirically prove the hypothesis that similar phrases will be played in a similar way. Our observation goes beyond simple repeats: we look at 'similar' phrases regardless of where they are 'located' in our music corpus - sometimes maybe belonging to pieces which are different in character. For the purposes of this work we need two kinds of phrase similarity: (1) 'objective' similarity between two phrases according to their scores and (2) similarity between the ways in which two phrases are performed by a pianist. The first similarity measure can be regarded as a mathematical function of the phrase scores (termed score-based similarity further in this text) and is assessed with the DISTALL's similarity measure. The second similarity measure (performance-based similarity) is computed from the performances of a Viennese pianist recorded on a Bösendorfer SE290 computer-monitored concert grand piano. In a further step we show that a dependency between our 'mathematical', score-based similarities and the performer's expressive decisions exists not only for the most similar phrases: a certain level of dependency between the two measures seems to hold even for the whole 'similarity scale'.

6.2 Score- vs. Performance-based Similarities

6.2.1 Phrases and score-based similarity measure

Figure 6.1 shows the score of eight bars of Mozart's (C major) Sonata KV 279, 1st movement. The brackets at the bottom of the figure indicate phrases - segments of music which are heard and interpreted as coherent units.

Phrases are organized hierarchically: smaller phrases are grouped into higher-level phrases, which are in turn also grouped together, constituting a musical context

at a higher level of abstraction. Figure 6.1 shows phrases hierarchically nested at three levels of abstraction. Despite a lot of recent work on musical similarity (e.g., [Hewlett and Selfridge-Field, 1998; Cambouropoulos, 2001; Cambouropoulos and Widmer, 2001]), there is no known procedure in musicology which would objectively quantify similarity between two phrases (e.g. based on their scores). Instead, we use a technique discussed in this text so far. Motivated by a fact that phrases are organized hierarchically we use a similarity measure designed for the domains with distinctive structural characteristics. This similarity measure is an essential part of our algorithm DISTALL [Tobudic and Widmer, 2006], which we discussed in chapter 3.

For the purpose of the presented work, it is not of importance how DISTALL's score-based similarity measure works in detail. The similarity measure can be regarded as a black box, which - given the scores of two phrases as input - outputs their similarity in the range [0, 1], 1 meaning that the phrases are identical. The algorithm computes phrase similarities based on their musical properties (attributes) that are computed from the score. As discussed already in the last chapter these are: The length of a phrase, the relative position of the highest melodic point (the 'apex'), the melodic intervals between starting note and apex, and between apex and ending note, metrical strengths of starting note, apex, and ending note and the harmonic progression between start, apex, and end. Further descriptors provide information about global tempo and presence of trills and state whether the phrase ends with a 'cumulative rhythm', and whether it ends with a cadential chord sequence. The remaining attributes describe global characteristics of the phrases in statistical terms: mean and variance of the durations of the melody notes within the 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' of the melodic line). As discussed in the previous text, our similarity measures takes into account the sequential nature of music. We use relational predicate *succeeds(phrId2,phrId1)*, introduced in section 5.2.4. As stated earlier, the similarity between two phrases in this case depends also on the preceding and succeeding music, which is from the musical point of view a rather intuitive idea.

We will evade a lengthy and difficult discussion of whether this particular measure of score similarity makes musical sense. The very fact that our experimental data will show significant correlations between performance similarity and phrase similarity, as measured in this way, unequivocally shows that the score-based phrase similarity measure does capture something meaningful in a systematic way.

6.2.2 Performance-based similarities

The starting material for deriving performance-based similarities are recordings of music pieces performed by a Viennese concert pianist on a Bösendorfer SE290

computer-monitored concert grand piano.[1] 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. For a detailed reference on recording and playback functionality of SE290 see [Palmer and Brown, 1991; Repp, 1993; Goebl, 2001].

From the Bösendorfer SE290 recordings, the tempo and dynamics curves are extracted. These are then decomposed into elementary tempo and dynamics phrasal 'shapes' – we used the same procedure as for deriving the training instances for our relational instance-based learner (as described in section 4.3.3).

After the decomposition process, for each phrase in our data set, we end up with two triples of coefficients a_d, b_d, c_d and a_t, b_t, c_t for second-degree polynomials in the dynamics and tempo domain respectively, which approximate the pianist's interpretation of this particular phrase in a domain. We then define performance-based similarity between two phrases for both domains separately as the correlation between the performer's expressive shapes for these phrases in a domain. More formally:

$$perfSim(phr_1, phr_2) = corr(y_1 - 1, y_2 - 1)$$

where,

$$y_1 = a_1x^2 + b_1x + c_1, \quad y_2 = a_2x^2 + b_2x + c_2$$

and

$$corr(y_1, y_2) = \frac{\sum_{x=0,step=0.1}^{1} y_1(x)y_2(x)}{\sqrt{\sum_{x=0,step=0.1}^{1} y_1^2(x)}\sqrt{\sum_{x=0,step=0.1}^{1} y_2^2(x)}}$$

a_1, b_1, c_1 and a_2, b_2, c_2 are the expressive shape coefficients of the two phrases for either dynamics or tempo. We subtract 1 from the shape functions since all expressive values are related to an average value, either average loudness or average tempo of the piece, which is represented as $y = 1$. The denominator in the last equation ensures that the autocorrelation of any sequence is identically 1.

As a result, the performance-based similarity between two phrases for each domain is expressed as a number in the range [-1, 1]. Similarity values greater than 0 suggest that the general trends of both phrase shapes in a domain are similar: e.g., for both phrases the pianist 'stays' above the average loudness during the whole phrase or starts soft and proceeds to loud toward the end of the phrase. On the other hand, a performance-based similarity smaller than 0 reveals that the pianist's interpretations of the two phrases are rather opposite.

[1] These recordings were made prior to and independently of our study, for a CD production, and the pianist had no idea that his recordings would ever be used in a scientific study.

sonata section		notes	phrases at level 1	2	3
kv.279:1:1	fast 4/4	391	50	19	9
kv.279:1:2	fast 4/4	638	79	36	14
kv.280:1:1	fast 3/4	406	42	19	12
kv.280:1:2	fast 3/4	590	65	34	17
kv.280:2:1	slow 6/8	94	23	12	6
kv.280:2:2	slow 6/8	154	37	18	8
kv.280:3:1	fast 3/8	277	28	19	8
kv.280:3:2	fast 3/8	379	40	29	13
kv.282:1:1	slow 4/4	165	24	10	5
kv.282:1:2	slow 4/4	213	29	12	6
kv.282:1:3	slow 4/4	31	4	2	1
kv.283:1:1	fast 3/4	379	53	23	10
kv.283:1:2	fast 3/4	428	59	32	13
kv.283:3:1	fast 3/8	326	53	30	12
kv.283:3:2	fast 3/8	558	79	47	19
kv.332:2	slow 4/4	477	49	23	12
Total:		5506	714	365	165

Table 6.1: 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.

6.3 Methodology

6.3.1 Data

The Mozart piano sonatas which were used for the study are given in Table 6.1. The scores of the sonatas were coded manually into a symbolic representation, and a multi-level phrase structure analysis of each piece was carried out by a musicologist. For the experiments we used phrase structure marked at the three hierarchical levels. From the scores and phrase structure information, the basic phrase features were calculated, which were in turn used for deriving score-based similarities (see section 6.2).

The starting point for deriving performance-based phrase similarities are performances of the above mentioned Mozart sonatas by a Viennese concert pianist on the Bösendorfer SE290. From these recordings (and phrase structure information), dynamics and tempo performance curves are computed and, in a further step, dynamics and tempo phrasal shapes. These are used for deriving performance-based phrase similarities (see section 6.2).

6.3.2 Procedure

The result of the score- and performance-based phrase similarity computation are two matrices for each phrase level (1 to 3) and each dimension (dynamics and tempo). Matrix entries are similarities (score- or performance-based) for each possible pair of phrases at the same level. Figure 6.2 shows score- and performance-based matrices (712 x 712) for the dynamics dimension for phrases at level 1. Each unit on the axes represents a phrase. Within a piece phrases are ordered chronologically, and the pieces themselves are ordered according to Table 6.1. The first phrase of each piece is indicated by a tick on the respective axis. The brightness of each pixel in the figure corresponds to the similarity of the two corresponding phrases, white meaning that the phrases are identical.

At this place let us refer to the main goal of this work: we try to empirically show a dependency between an 'objective' similarity measure defined on music phrases (i.e. phrase similarities suggested by our score-based algorithm) and the ways the phrases are played by one and the same pianist. We hope to find this dependency even in a large corpus of complex, diverse music like our 16 sonata sections. Mathematically, the dependency is defined as the correlation between score- and performance-based matrices as follows:

First, some elementary data preprocessing is undertaken. The main diagonal and the lower triangular part of the matrices are discarded (both similarity measures satisfy reflexivity and symmetry relations). Both matrices are also normalized to have mean 0 and standard deviation 1. Ideally, we would like to be able to show high correlation between all pixel values in the upper triangular part of the score-based matrix and the upper triangular part of the performance-based matrix. However, this is unrealistic for many reasons: The score-based similarity measure is a mathematical function of a rather limited set of phrase features computed from the score. The performance-based matrices depend solely on the performer's decisions. High correlation between all pixels in the two matrices would mean that not only are similar phrases played in similar ways but also 'half-similar' in 'half-similar' and dissimilar in dissimilar ways, with the whole continuous mathematical similarity scale mapped 'correctly' to the performer-based similarity scale, which is a very unrealistic assumption. It is also not clear if quantifying the whole similarity scale between music phrases make sense in terms of musicology: while most musicologists would probably agree that two phrases can be regarded as similar or dissimilar, it would be difficult to quantify dissimilarity as e.g. 'rather dissimilar' (0.3) or 'fully dissimilar' (0.0).

For these reasons we first limit our investigation to those phrases which are suggested to be most similar by our score-based algorithm and show that those phrase pairs are (mostly) played similarly by the pianist. We then systematically repeat the same procedure, including more and more (less similar) phrase pairs, and examine if the high correlation between score- and performance-based values still holds. The same procedure is repeated until all phrase pairs are included in the

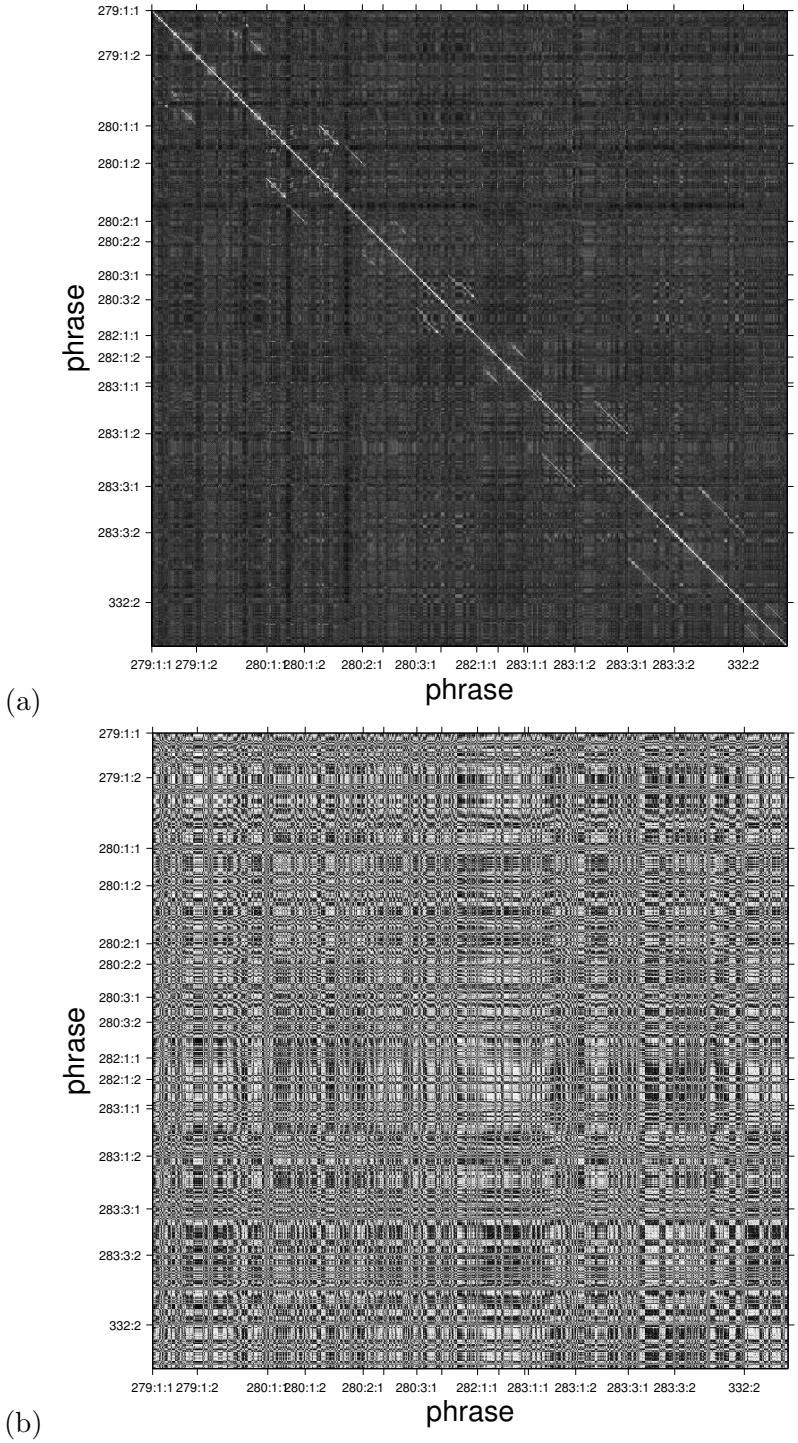

Figure 6.2: Score- (a) and performance-based (b) phrase similarity matrix for the dynamics dimension for all phrases at phrase level 1. Each unit on the axes represents a phrase. Phrases are ordered chronologically within a piece. The first phrase of each piece is indicated by a tick on the respective axis. The brightness of each pixel represents similarity of two corresponding phrases, white meaning the phrases are identical.

computation.

More formally, from the upper triangular part of the score-based similarity matrix we choose the n most similar phrase pairs (the n brightest pixels in the upper part of Figure 6.2). The corresponding pairs are then found in the performance-based similarity matrix and the correlation between these two sequences is computed as:

$$corr(s_n, p_n) = \frac{\sum_n s_n p_n}{\sqrt{\sum_n s_n^2}\sqrt{\sum_n p_n^2}}$$

where the s_n are n most similar score-based similarity values, and p_n are the n corresponding performance-based similarity values. The denominator again ensures that the autocorrelation of any sequence is normalized to 1. The given equation can be regarded as a measure of how consistently performance-based values 'follow' score-based similarities for a given set of phrase pairs. In the further steps we systematically increase the number of phrase pairs by including more pairs from the sorted score-based matrix and repeat the above described procedure. At the end we get the correlation between score- and performance-based similarities for all phrase pairs at one level: the case we discussed above where we examine if the score-based similarity measure correlates with the pianist's decisions on the whole similarity scale. The above described procedure is carried out for all three phrase levels and both domains (dynamics and tempo).

Note that even when we consider the most similar phrase pairs (as suggested by the score-based algorithm), high correlation is only possible if these pairs are (mostly) played in similar ways by the pianist, regardless of where they are in the music corpus. E.g., even if a pair consists of phrases which belong to pieces very different in character they have to be played rather similarly by the pianist in order to achieve a high correlation.

6.4 Results

6.4.1 Correlation between score- and performance-based similarities

The results of the correlation computation between score- and performance-based similarities as described in the last section are given in Figure 6.3.

The line marked '+' shows correlation progression between score- and performance-based similarities for different fractions (0.1% to 100%) of the total number of phrase pairs at each phrase level for both dimensions (levels 1, 2 and 3 have approximately 253 thousands, 66 thousands, and 13 thousands phrase pairs in total). In order to put these results in a context we repeated the same procedure for the samples of a normally distributed random variable (mean 0 and standard deviation 1) and the performance-based similarities. In other words we want to examine how much better than a random variable our score-based similarities correlate with the performer's

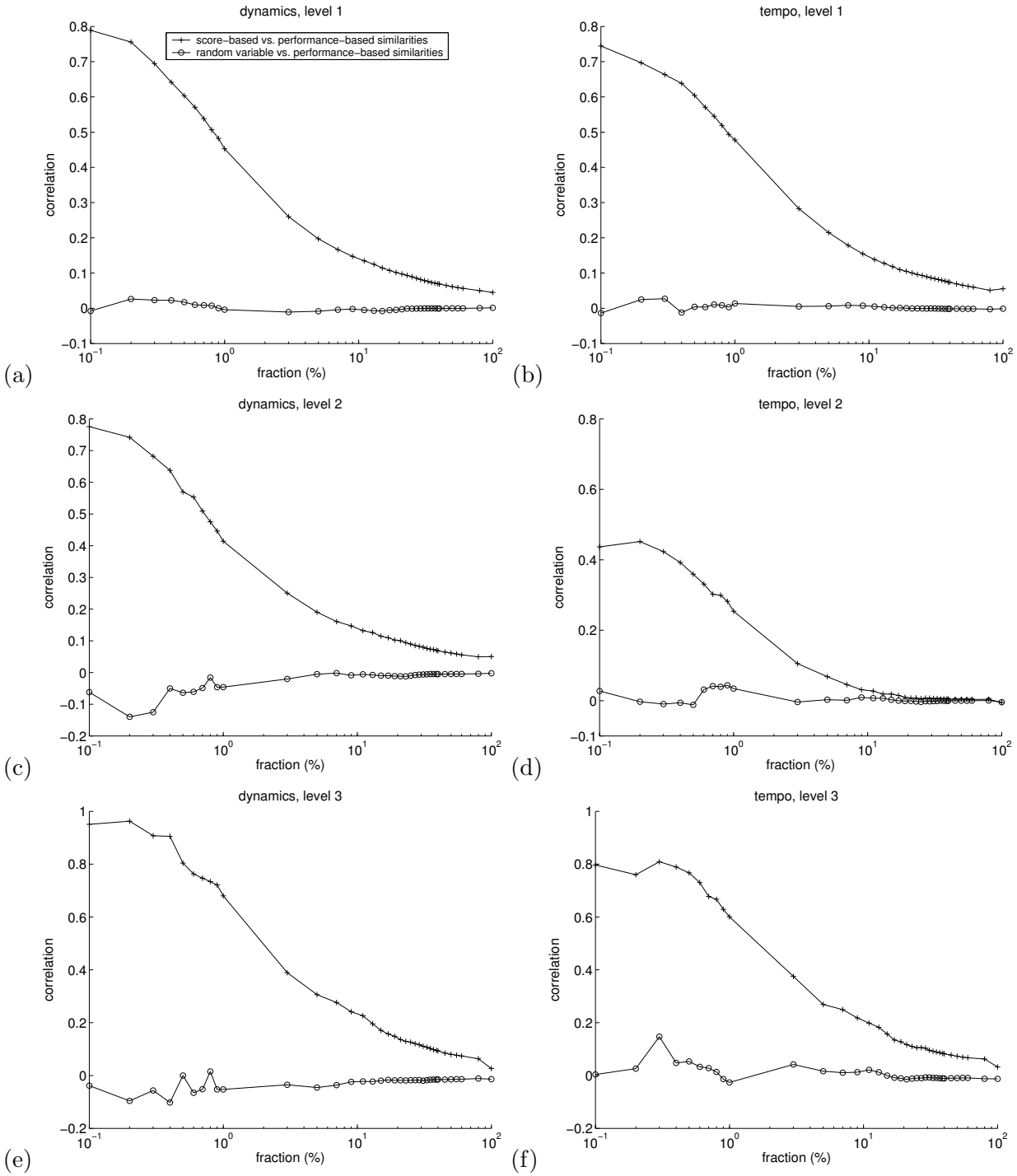

Figure 6.3: Correlation between score- and performance-based similarities ('+'-marked) for all phrase levels (1 to 3) and both dimensions (dynamics, tempo), for different fractions (indicated by the x-axis) of the most similar phrase pairs (as suggested by the score-based similarity algorithm). The results of the same procedure between a normally distributed random variable (mean 0, standard deviation 1) and performance-based similarities are marked 'o'.

decisions. These baseline correlations - which for large sample sizes converge to zero - are marked 'o'. A first look at Figure 6.3 reveals high correlation between score- and performance-based similarities for the 0.1% most similar phrase pairs at all phrase levels and for both dimensions. While it might look as a small fraction, 0.1% of the total number of phrase pairs is quite a large number - e.g. more than 250 phrase pairs for phrase level 1. Correlations of about 80% in dynamics and 75% in tempo show that the pianist indeed plays these phrase pairs rather similarly.

The very high correlations at the 0.1% mark for the phrase level 3 in both dimensions (almost 100% and 80% for dynamics and tempo respectively) should not be misinterpreted. They are a result of the smaller total number of phrases at this level and not due to the fact that the score-based suggestions correlate with the pianist's decisions better at the higher phrase levels (0.1% for level 3 corresponds to only 13 phrase pairs). Rather, the opposite is the case. Correlations between score- and performance-based similarities decrease at higher phrase levels if one considers absolute numbers of phrase pairs: E.g. a sequence of about 2500 phrase pairs corresponds to fractions of about 1% and 18% for levels 1 and 3 respectively - consider the correlation drop for these fraction values between levels 1 and 3 for both dimensions. The initial upward trend in the tempo domain for levels 2 and 3 can also be explained by the relatively small number of phrase pairs: a few uncorrelated similarities in a sequence of 0.1% of all phrase pairs at level 3 (13 pairs in total) can degrade the overall correlation, which is then improved given more pairs.

Figure 6.3 also reveals that the correlation between score- and performance-based similarity values is generally higher in the dynamics domain. This can partly be explained by the fact that quadratic functions which we use to model performer decisions (and performance-based similarities accordingly, see section 6.2) may not be as reasonable a model class for expressive timing as it has been believed in musicology (see also [Widmer and Tobudic, 2003]), the fact we already discussed in the last chapter.

As expected, the base line correlation between samples of a random variable and performance-based similarities oscillates around 0 since the two variables are independent (the 'most similar' phrase pairs are in this case chosen randomly). The deviations from 0 for the baseline correlation can be explained by higher variance for smaller sample sizes (this is the case for the lower fractions and/or for the higher phrase levels). Finally, Figure 6.3 also offers an answer to the question of how well the score-based similarity measure 'matches' performer decisions on the whole similarity scale. As can be seen from the right most points in the diagrams in Figure 6.3, there is a positive correlation between the score- and the performance-based similarities for the sequences of all phrase pairs for the smallest phrases at the level 1 (the phenomenon is less distinctive for larger phrases at levels 2 and 3). Although the correlations are rather small (about 4.5% and 5.5% for level 1 dynamics and tempo, respectively), it is still remarkable that any 'non-performance' similarity measure correlates above baseline with the performer decisions for all combinations of phrase pairs (253 thousands in total) in a rich music corpus as given in Table 6.1.

6.4.2 The most similar phrase pairs

The black pixels in Figure 6.4 (a), (b), and (c) show the distribution of the 1% most similar phrase pairs – as suggested by the score-based similarity measure – for phrase levels 1 to 3 respectively, among different pieces from our music corpus.

For each of these diagrams, the subset of the phrase pairs which are performed similarly by the pianist is plotted in Figure 6.4 (d), (e), and (f). A phrase pair is regarded as having been performed similarly if the correlation between the expressive shapes of the two phrases is above 0 for *both* domains, dynamics *and* tempo. As expected, most of the phrase pairs given in Figures 6.4 (a), (b), and (c) consist either of phrases which come from the same pieces (pixels around the main diagonal), or generally sections belonging to the same sonata movements (pixels belonging to the neighboring pieces, e.g. the crossing of kv.280:1:1 and kv.280:1:2 or kv.283:3:1 and kv.283:3:2). Figures 6.4 (d), (e), and (f) suggests that quite a number of them are also performed in a similar way.

On the other hand, some of the phrase pairs regarded as very similar by the score-based similarity measure consist of phrases which belong to pieces which are quite different in character, e.g. the pairs in the last column in the diagrams where one of the phrases from a pair belongs to kv.332:2 and the other one to some other piece. E.g. consider four phrase pairs (four black pixels) in the last column of Figure 6.4 (c), where one of the phrases from a pair belongs to the kv.332:2 and the other one either to kv.282:1:2 or kv.282:1:3. Three of the four pairs are also performed similarly by the pianist (they reappear in Figure 6.4 (f)), although our listening tests revealed that kv.332:2 is rather different in character than both kv.282:1:2 or kv.282:1:3. Actually it turns out that 50%, 46%, and 74% (for levels 1 to 3 respectively) of the phrase pairs which are regarded as the most similar by the score-based similarity algorithm are also performed in a similar way by the pianist. The baseline probability for a pair to have a correlation of expressive shapes above 0 for both, dynamics and tempo domain is 25% (50% for either domain).

6.5 Discussion

An empirical investigation of the dependency between similar phrases and the way they are played by the same pianist is presented. It turned out that in many cases our pianist performed phrases suggested to be similar by our score-based similarity measure in rather similar ways, sometimes regardless of the character of the pieces to which the phrases belong. The high correlations between score-based similarities and performer expressions for the most similar phrase pairs can be regarded as indication of stylistic consistency of the performer. On the other hand, it can be considered as a reliability proof of our score-based similarity measure. We have also shown

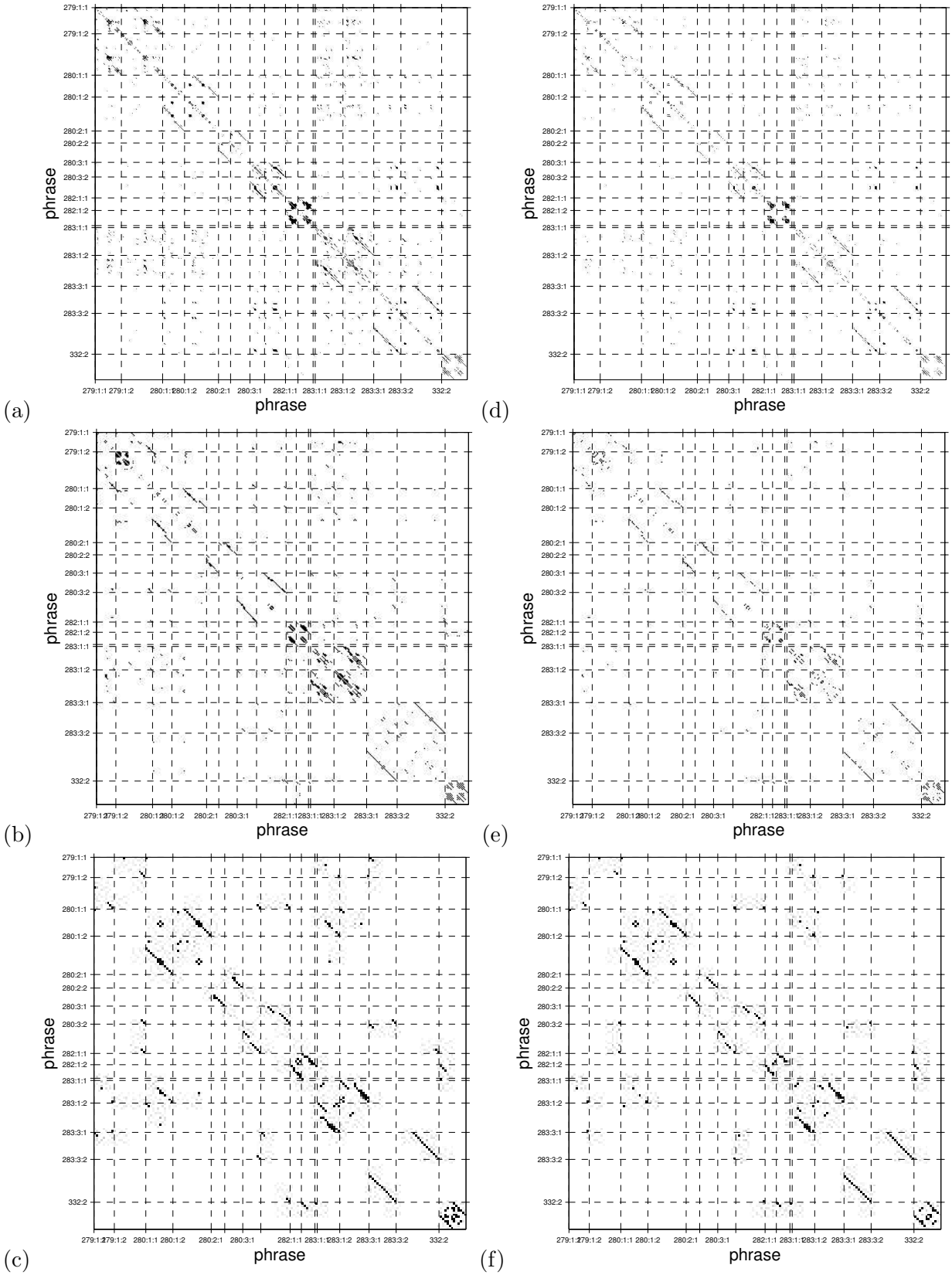

Figure 6.4: Black pixels show the distribution of the 1% of the most similar phrase pairs – as suggested by the score-based similarity measure – among pieces of the corpus given in Table 6.1 for the phrase levels 1 (a), 2 (b), and 3 (c). Figures (d), (e) and (f) show the subsets of the phrase pairs given in (a), (b), and (c) which are performed similarly by the pianist.

that a certain level of dependency between score- and performance-based similarity measures holds not only for the most similar phrases: it turns out that the two 'variables' are not statistically independent even if all combinations of phrase pairs in our data corpus are considered (about 253 thousands in total).

The experiments described above are related to another work carried out in our research lab ([Madsen and Widmer, 2006]). The starting point of the work described in [Madsen and Widmer, 2006] was 12 recordings of a Schubert piano piece, played by great pianists. The goal was to assess both the artists *intra-piece consistency* and potential *similarities* between their playing styles. Beside the different starting material (one piece played by different performers in [Madsen and Widmer, 2006] vs. one performer playing different pieces in our work), the work described in [Madsen and Widmer, 2006] pursued the investigation in the reverse order: We have started from the score-based similarities and tried to answer the question if the most similar phrases (as suggested by the score) are played in similar ways. In [Madsen and Widmer, 2006] the first step was finding the sequences of greatest similarities in the *performances* and comparing the music behind.

The presented work has a serious limitation: in spite of their diversity, every piece used in the experiments is written in the same musical style, a piano sonata from the classical period. It would be interesting to examine whether the same level of dependency holds if the pieces belong to different styles, e.g. classical and romantic. On the other hand, we could investigate if the consistency level shown for the pianist used in our experiments holds for different playing skill levels, e.g. novices/hobby players/educated players. And of course, the most interesting observation would be if we could show the same high level of dependency between the 'mathematical variable' and the performer decisions for the case of great pianists. This would give us at least a (very) limited insight into the 'mystery' of their playing. Despite the difficulties of data acquisition for the case of great pianists, it will be our next step.

Chapter 7

Learning to Play Like the Great Pianists

In Chapter 5 we have discussed the application of instance-based learning in general, and our new relational instance-based learner in particular, for building predictive models of expressive piano performance. Probably the most interesting question one can consider in such kind of research is whether our technique can be applied to truly great pianists. Can a machine build a formal model of the playing style of great pianists? Can it learn to distinguish great pianists from each other? Or even, can it learn to replicate the style of, say, Arthur Rubinstein? Such questions are discussed in this chapter.

In particular, we use DISTALL to operate on data extracted from music CDs of great pianists. We investigate to what extent it can automatically build 'expressive profiles' of famous pianists using only minimal performance information derived from audio CD recordings by pianists and the printed score of the played music. It will turn out that the machine-generated expressive performances on unseen pieces are substantially closer to the real performances of the 'trainer' pianist than those of all others. We also discuss two order interesting applications of the work: recognizing pianists from their style of playing, and automatic style replication.

The rest of the text is laid out as follows. After a short introduction (section 7.1), section 7.2 describes the data and methods used in the further text. Experimental results and discussion of our first experiment, namely learning predictive performance models, are given in section 7.3. The other interesting question one can pose in this kind of research – can machine learn to recognize great pianists from their style of playing – is discussed in section 7.4. Finally, in section 7.5 we shortly discuss the automatic style replication.

This chapter reports research already published at the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05) [Tobudic and Widmer, 2005].

7.1 Introduction

The method of data acquisition used for the experiments presented so far in this text is unsuitable for studying great pianists: Our starting material was very precise measurements of when, how long and how loud *each individual note* was played by a pianist and how these measurements deviated from the nominal values prescribed in the written musical score. The main source of information was the Bösendorfer SE290, a special piano that precisely records each action by performer. As discussed already in the chapter 5, the SE290 is equipped 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 by comparing them to the notes as specified in the written score, every expressive nuance applied by a pianist can be computed.

Unfortunately, the SE290 is rather useless for studying truly great pianists (we cannot very well invite them all to Vienna to perform on the Bösendorfer SE290 piano). With great pianists, the only source of data is audio recordings (records and music CDs). However, it is impossible, with current signal-processing methods, to extract precise performance information about each individual note directly from audio data. Instead, for studying the great pianists we will use rather crude information which can be extracted from their audio CDs, namely the tempo and loudness information at the *beat* level. We will explore the question, if our system presented so far and operating with this very limited information, can learn models that are 'personal' enough to capture some aspects of the playing styles of six famous pianists?

Experiments show that the system indeed captures some aspect of the pianists' playing style: the machine's performances of unseen pieces are substantially closer to the real performances of the 'training' pianist than those of all other pianists in our data set. We also demonstrate an interesting by-product of the pianists' 'expressive models': the automatic identification of pianists based on their style of playing. And finally, the question of automatic style replication is briefly discussed.

7.2 Data and Methodology

The data used in this work was obtained from commercial recordings of famous concert pianists. We analyzed the performances of 6 pianists across 15 different sections of piano sonatas by W.A.Mozart. The pieces selected for analysis are complex, different in character, and represent different tempi and time signatures. Tables 7.1 and 7.2 summarize the pieces, pianists and recordings selected for analysis.

For learning tempo and dynamics models of the pianists in our data set, it would be ideal to have precise information about each note in the pianist's audio recording (start and end times, loudness, and so on). Unfortunately, it is impossible, with current signal-processing methods, to extract precise performance information about each individual note directly from audio data.

Section ID	Tempo descr.	#phrases
kv279:1:1	fast 4/4	460
kv279:1:2	fast 4/4	753
kv280:1:1	fast 3/4	467
kv280:1:2	fast 3/4	689
kv280:2:1	slow 6/8	129
kv280:2:2	slow 6/8	209
kv280:3:1	fast 3/8	324
kv280:3:2	fast 3/8	448
kv282:1:1	slow 4/4	199
kv282:1:2	slow 4/4	254
kv283:1:1	fast 3/4	455
kv283:1:2	fast 3/4	519
kv283:3:1	fast 3/8	408
kv283:3:2	fast 3/8	683
kv332:2	slow 4/4	549

Table 7.1: Mozart sonata sections selected for analysis. Section ID should be read as <sonataName> : <movement> : <section>. The total numbers of phrases are also shown.

ID	Pianist 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 7.2: Pianists and recordings used for experiments.

Instead, we extract time points from the audio recordings that correspond to *beat* locations. Finding beat locations in audio file is an extremely difficult task and open research problem[1], that forced our research group to develop a novel beat tracking algorithm called `BEATROOT` [Dixon, 2001a; Dixon and Cambouropoulos, 2000]. Experimental evaluations showed that the `BEATROOT` algorithm is one of the best beat tracking methods currently available [Dixon, 2001b].

From the (varying) time intervals between beat points, the beat-level tempo and its changes can be computed. Beat-level dynamics is computed from the audio signal as the overall loudness of the signal at the beat times as a very crude rep-

[1]Beat tracking, in a sense, is what human listeners do when they listen to a piece and tap their foot in time with music. As with many other perception and cognition tasks, what seems easy and natural for a human turns out to be extremely difficult for a machine.

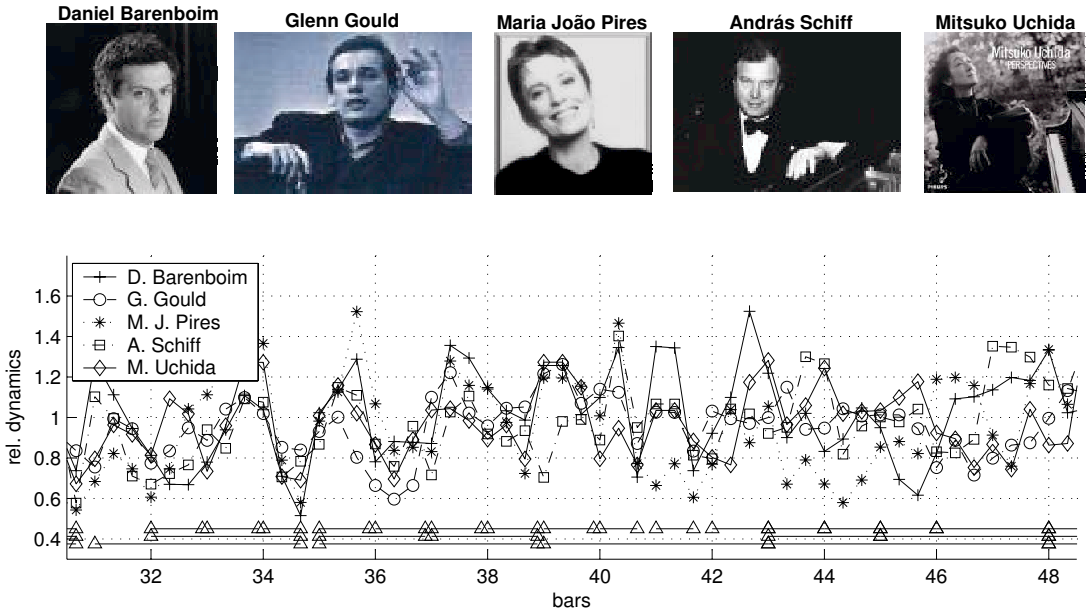

Figure 7.1: Dynamics curves of performances of five famous pianists for Mozart Sonata K.280, 1^{st} mvt., mm. 31–48. Each point represents the relative loudness at the beat level (relative to the average loudness of the piece by the same performer).

resentation of the dynamics applied by the pianists. After extracting beat points and calculating beat-level tempo and dynamics, computing pianists' dynamics and tempo performance curves, as discussed so far in this text, is rather straightforward. These curves, together with the musical score and the underlying phrase structure of the pieces is again the starting material for our experiments.

7.3 Learning Predictive Performance Models

Figure 7.1 shows the dynamics of Mozart Sonata K.280, 1^{st} movement, as played by five famous pianists. As with other examples discussed so far in this text, the musically informed reader can reveal certain trends common for all pianists. These trends often reflect the underlying phrase structure of the piece, which is indicated by three levels of brackets at the bottom of the figure. Despite the commonalities between the pianists, the curves are however not identical. There is also a significant difference between individual performers in shaping the phrases, which can be partly attributed to the performers' styles. In this section we will present experiments

which try to catch something from these deviations (and thus from the performer's styles), by exploiting the intra-pianist difference in shaping the phrases of Mozart music. We use DISTALL to identify 'characteristic' expressive phrase shapes for each individual pianist and apply them to a new piece. The phrase shapes are then combined into tempo and dynamics curves. Since DISTALL learns from each pianist separately, we get the learned tempo and dynamics curves for each 'training' pianist. If the learned curves are then closer to the actual curves of the trainer than to all other pianists, we could say that DISTALL has build a (very) limited pianist's 'expressive model'.

For each pianist, we conducted a systematic *leave-one-piece-out* cross-validation experiment: each of 15 pieces was once left aside as a test piece while the remaining 14 performances (by the same pianist) were used for learning. DISTALL's parameter for the number of nearest neighbors was set to 1. We used the temporal predicate *succeeds/2* (section 5.2.4). The parameter for the depth of starting clauses (section 3.5) is set to 4 (meaning that the distance between two phrases can be influenced by at most 4 preceding and 4 succeeding phrases).

The expressive shapes for each phrase in a test piece are predicted by DISTALL and then combined into a final tempo and dynamics curve, as described in section 4.3.3. The resulting curves are then compared to the real performance curves of all pianists (for the same test piece). As stated above, if the curve learned from the performances of one pianist is more similar to the real performance curve of the 'teacher' pianist than to those of all other pianists, we could conclude that the learner succeeded in capturing something of the pianist's specific playing style. The described procedure is repeated for all pieces and all pianists in our data set.

Correlation is chosen as a measure of how well the predicted curve 'follows' the real one. The curves are first normalized so that their autocorrelations are identically 1, giving a correlation estimate between curves as a number in the range [-1,1]. The results of the cross-validation experiment averaged over all pieces (weighted by the relative length of the pieces) are given in Table 7.3.

Interestingly, the system succeeded in learning curves that are substantially closer to the 'trainer' than all others, for all pianists. Some of the pianists are better 'predictable' than others, e.g. Daniel Barenboim and Mitsuko Uchida, which might indicate that they play Mozart in a more 'consistent' way. While at first sight the correlations may not seem impressive, one should keep in mind that artistic performance is far from predictable. Some pieces in our dataset are also quite different in character from the majority (notably, kv332:2). Predicting performance curves for such pieces is thus an extremely difficult task. One should also have in mind, that the numbers in Table 7.3 are averages over all pieces in our dataset. For instance, the system predicts Barenboim's expressive curves with an average correlation of .44 for both dynamics and tempo for all pieces in our data set, which is about half an hour of concert-level piano music.

Moreover, the correlation estimates in Table 7.3 are somewhat unfair, since we compare the performance curve produced by composing the polynomials predicted

	compared with					
learned from	DB	RB	GG	MP	AS	MU
DB	**.44**	.21	.26	.34	.38	.28
	.44	.27	.26	.32	.31	.31
RB	.21	**.32**	.09	.19	.19	.17
	.28	**.42**	.20	.22	.30	.27
GG	.25	.09	**.36**	.19	.21	.22
	.25	.18	**.32**	.23	.29	.28
MP	.33	.19	.19	**.39**	.33	.28
	.31	.23	.27	**.38**	.28	.34
AS	.36	.17	.20	.31	**.40**	.26
	.32	.29	.28	.25	**.41**	.32
MU	.27	.18	.21	.28	.26	**.38**
	.34	.30	.32	.36	.37	**.50**

Table 7.3: Results of piecewise cross-validation experiment. The table cells list correlations between learned and real curves, where rows indicate the 'training pianist', and columns the pianist whose real performance curves are used for comparison. The correlations are averaged over all pieces, weighted by the relative length of the piece. Each cell is further divided into two rows corresponding to *dynamics* and *tempo* correlations, respectively. The highest correlations in each row are printed in bold.

by the learner, with the curve corresponding to the pianists' *actual* performances. However, what DISTALL learned from was not the actual performance curves, but an *approximation* curve which is implied by the three levels of quadratic functions that were used as training examples. Correctly predicting these is the best the learner could hope to achieve.

Tables 7.4 - 7.9 show a more detailed, piecewise picture of the cross-validation experiments. In each of the 6 tables, the training pianist was one of the 6 pianists and the learned dynamics and tempo curves for each piece are compared to the actual curves from all 6 pianists for that piece. It turns out that DISTALL is indeed able to learn something from pianists' playing styles:

E.g., in 11 out of 15 cases for both domains the learner produces dynamics and tempo curves which are closer to Barenboim's playing than to any other pianist (Table 7.4). The system achieved surprisingly good results also for Uchida. Learning from Uchida results in curves which are closer to Uchida's playing than to any other pianist in 9 and 13 cases for dynamics and tempo respectively (Table 7.9). Some of the predicted curves exhibit high correlations of .7 and better with the Uchida's real performance curves (e.g., correlation of 0.78 in dynamics domain for kv283:3:1 and 0.78 for kv280:1:1 in tempo domain). The results are even more interesting if we recall that the learner is given a very crude, beat-level representation of the tempo and dynamics applied by the pianist, without any details about e.g. individual

voices or timing details below the level of beat.

On the other hand, the piecewise results revealed that some of the pianists seem to be less 'predictable' than Uchida or Barenboim with our approach. E.g., having Pires or Schiff as trainer makes the learning task apparantly more difficult: In 8 and 9 cases for dynamics and tempo respectively, the learner succeeds in producing dynamics and tempo curves closer to the respective teacher than any other pianist for both, Pires and Schiff (see Tables 7.7 and 7.8). The results for Glenn Gould being a teacher are mixed. He seems to be much more predictable in dynamics domain: In 14 our of 15 cases are learned curves closer to dynamics of Gould than to any other pianist. It is indeed the best piecewise result for all 6 teachers and both domains. On the other hand, the same learning goal succeeds in just 4 out of 15 pieces for tempo (Table 7.6). This is again the worst result for all 6 teachers, suggesting that Glenn Gould is much more 'consistent' in dynamics than in tempo domain.

Figure 7.2 shows an example of successful performance style learning. We see a passage from a Mozart piano sonata as 'played' by the computer after learning from recordings of other pieces by Daniel Barenboim (top) and Mitsuko Uchida (bottom), respectively. Also shown are the performance curves corresponding to these two pianists' actual performances of the test piece. In this case it is quite clearly visible that the curves predicted by the computer on the test piece are much more similar to the curves by the respective 'teacher' than to those by the other pianist.

Admittedly, this is a carefully selected example, one of the clearest cases of style replication we could find in our data. The purpose of this example is more to give an indication of the complexity of the curve prediction task and the difference between different artists' interpretations than to suggest that a machine will always be able to achieve this level of prediction performance.

7.4 Identification of Great Pianists

The primary goal of our work is learning predictive models of pianists' expressive performances.[2] But the models can also be used in a straightforward way for recognizing pianists. The problem of identifying famous pianists from information obtained from audio recordings of their playing has been addressed in the recent literature [Saunders *et al.*, 2004; Stamatatos and Widmer, 2002; Widmer and Zanon, 2004]. In [Widmer and Zanon, 2004], a number of low-level scalar features related to expressive timing and dynamics are extracted from the audio CD recordings, and various machine learning algorithms are applied to these. In [Saunders *et al.*, 2004],

[2]Note that learned tempo and dynamics curves as produced by our system can be used to build truly machine generated expressive performances: using the predicted tempo and dynamics curves (i.e. relative tempo and dynamics for each beat in the piece), it is straightforward to calculate tempo and dynamics for each note in the piece (e.g. by interpolation).

Piece	DB	RB	GG	MP	AS	MU
kv279:1:1	.21	.23	.21	.25	**.35**	.11
	.48	.28	.22	.23	.21	.29
kv279:1:2	.22	.11	.04	**.27**	.16	.18
	.43	.30	.27	.32	.31	.33
kv280:1:1	**.70**	.54	.04	.54	.56	.30
	.64	.28	.31	.46	.42	.54
kv280:1:2	**.38**	.24	.19	.30	.26	.22
	.34	.27	.16	.21	.23	.21
kv280:2:1	**.48**	.40	.20	.47	.43	.44
	.59	.39	**.59**	.54	**.59**	.56
kv280:2:2	**.42**	.41	.05	**.42**	.28	.31
	.50	.21	.32	.28	.45	.45
kv280:3:1	**.80**	.67	.34	.57	.62	.50
	.79	.74	.72	**.86**	.74	.81
kv280:3:2	**.72**	.64	.41	.61	.61	.64
	.41	**.44**	.34	.41	.42	.42
kv282:1:1	**.52**	.35	.35	.40	.34	.29
	.61	.26	.51	.50	.06	.40
kv282:1:2	**.47**	.38	.18	.31	.14	.17
	.45	.22	.32	.40	.23	.29
kv283:1:1	.28	-.08	.31	.32	**.38**	.23
	.20	.11	-.01	.25	.24	**.26**
kv283:1:2	**.22**	.12	.10	.15	.16	.16
	.13	.04	.12	**.23**	.13	.17
kv283:3:1	**.86**	-.19	.70	.48	.68	.36
	.78	.49	.33	.34	.59	.18
kv283:3:2	**.67**	.03	.62	.44	.61	.40
	.48	.17	.32	.20	.30	.09
kv332:2	.10	.02	.19	.01	.16	.11
	.22	.13	.02	.13	.07	.19
Total	**.44**	.21	.26	.34	.38	.28
	.44	.27	.26	.32	.31	.31

Table 7.4: Detailed results of the cross-validation experiment with Daniel Barenboim was the 'training' pianist. For each piece, the correlations between predicted and actual curves from all pianists are given. Each cell is divided into two rows corresponding to *dynamics* and *tempo* correlations, respectively. The highest correlations in each row are printed in bold. The average over all pieces is given in the last two rows (reproduced from the Table 7.3)

Piece	DB	RB	GG	MP	AS	MU
kv279:1:1	.35	.39	.19	**.41**	.35	.24
	.32	**.46**	.26	.30	.26	.40
kv279:1:2	.02	**.28**	.06	.15	.05	.07
	.30	**.43**	.18	.34	.28	.36
kv280:1:1	.49	**.58**	.03	.42	.38	.26
	.41	**.70**	-.08	.35	.45	.40
kv280:1:2	**.37**	.35	.06	.32	.35	.27
	.12	**.49**	.16	.07	.17	.10
kv280:2:1	.41	**.53**	.18	.42	.42	.46
	.22	**.56**	.34	.33	.48	.16
kv280:2:2	.31	**.43**	-.01	.24	.29	.29
	.40	**.54**	.34	.31	.51	.34
kv280:3:1	.69	**.79**	.43	.63	.61	.57
	.80	**.90**	.80	.72	.74	.76
kv280:3:2	.55	**.62**	.42	.51	.53	.56
	.28	**.37**	.23	.18	.27	.29
kv282:1:1	**.47**	.43	.32	.31	.35	.24
	.33	.56	.27	.38	.49	**.57**
kv282:1:2	**.37**	.30	.06	.23	.17	.12
	.29	.45	.30	.31	**.53**	.45
kv283:1:1	**.04**	**.04**	-.06	-.06	.03	.03
	.03	.00	.06	-.03	.16	**.17**
kv283:1:2	-.16	**.18**	-.22	-.18	-.09	-.24
	-.01	.02	**.17**	.02	.07	.05
kv283:3:1	**.07**	.04	.00	-.07	.03	-.08
	.43	**.78**	.23	.29	.49	.20
kv283:3:2	-.14	**.14**	.00	-.05	-.16	.11
	.32	**.45**	.17	.11	.24	.08
kv332:2	.07	.13	.15	.05	**.19**	.17
	.29	.10	.06	.08	.11	.25
Total	.21	**.32**	.09	.19	.19	.17
	.28	**.42**	.20	.22	.30	.27

Table 7.5: Detailed results of the cross-validation experiment with Roland Batik was the 'training' pianist. For each piece, the correlations between predicted and actual curves from all pianists are given. Each cell is divided into two rows corresponding to *dynamics* and *tempo* correlations, respectively. The highest correlations in each row are printed in bold. The average over all pieces is given in the last two rows (reproduced from the Table 7.3)

Piece	DB	RB	GG	MP	AS	MU
kv279:1:1	.19	.38	**.40**	.22	.27	.24
	.26	.12	.23	.17	.30	.36
kv279:1:2	**.05**	.03	-.18	-.01	-.04	.00
	.18	.17	.17	.09	**.42**	.22
kv280:1:1	.04	.00	**.29**	-.10	-.10	-.08
	.35	.22	.52	.47	.31	**.55**
kv280:1:2	.14	.10	**.24**	.06	.04	.11
	.27	.17	**.38**	.30	.24	.24
kv280:2:1	.07	-.04	**.34**	.15	.04	.04
	.33	.39	**.60**	.50	.54	.50
kv280:2:2	.19	.15	**.34**	.11	.13	.10
	.34	.24	.41	.23	**.44**	.27
kv280:3:1	.44	.51	**.61**	.42	.44	.55
	.61	.57	.52	**.67**	.55	.65
kv280:3:2	.34	.36	**.50**	.31	.27	.43
	.35	**.50**	.25	.28	.43	.30
kv282:1:1	.28	.16	**.55**	.22	.26	.21
	.44	.32	.49	.41	.28	**.54**
kv282:1:2	.24	.14	**.33**	.17	.04	.07
	.08	.02	.15	**.16**	.00	.11
kv283:1:1	.35	-.12	**.56**	.41	.34	.38
	.16	.23	.25	.05	**.34**	.27
kv283:1:2	.26	.09	**.51**	.31	.28	.36
	.00	-.03	.12	.10	.11	**.14**
kv283:3:1	.74	-.18	**.77**	.42	.59	.40
	.50	.23	**.68**	.30	.42	.26
kv283:3:2	.48	-.08	**.49**	.27	.45	.30
	.19	.03	**.52**	.24	.29	.13
kv332:2	.10	.03	**.28**	.10	.14	.20
	.13	.02	.01	.04	.01	.09
Total	.25	.09	**.36**	.19	.21	.22
	.25	.18	**.32**	.23	.29	.28

Table 7.6: Detailed results of the cross-validation experiment with Glenn Gould was the 'training' pianist. For each piece, the correlations between predicted and actual curves from all pianists are given. Each cell is divided into two rows corresponding to *dynamics* and *tempo* correlations, respectively. The highest correlations in each row are printed in bold. The average over all pieces is given in the last two rows (reproduced from the Table 7.3)

Piece	DB	RB	GG	MP	AS	MU
kv279:1:1	.35	.25	.05	**.44**	.37	.22
	.34	.29	.26	.29	.18	.31
kv279:1:2	.09	.19	.07	**.31**	.19	.23
	.25	.25	.23	**.33**	.30	.30
kv280:1:1	.53	.44	-.02	**.68**	.45	.27
	.41	.32	.41	**.67**	.38	.57
kv280:1:2	.26	.19	-.03	**.35**	.23	.18
	.37	.32	.30	**.46**	.34	.37
kv280:2:1	.46	.35	.21	**.61**	.48	.42
	.41	.33	.56	**.60**	.48	.56
kv280:2:2	.37	.33	.13	**.41**	.31	.34
	.45	.31	.45	**.51**	.46	.51
kv280:3:1	.67	**.69**	.41	.66	.64	.56
	.70	.61	.53	**.81**	.66	.75
kv280:3:2	.47	.50	.33	**.55**	.42	.45
	.46	.36	.39	.47	.41	**.49**
kv282:1:1	**.51**	.32	.33	.39	.30	.33
	.45	.32	.52	**.54**	.16	.53
kv282:1:2	**.33**	.16	.13	.23	.22	.14
	.36	.29	.33	**.48**	.23	.40
kv283:1:1	.24	-.05	**.35**	.30	.24	.28
	.29	.10	.09	.28	.17	.23
kv283:1:2	**.28**	.15	.16	.18	.23	.26
	.07	.01	.09	**.20**	.09	.11
kv283:3:1	.56	-.06	.48	**.65**	.51	.38
	.34	.19	.29	.39	**.41**	.29
kv283:3:2	.41	-.04	.40	.48	**.49**	.38
	.19	.12	**.24**	.23	.21	.20
kv332:2	.01	-.04	**.16**	-.04	.15	.11
	.09	.02	.03	.05	.05	**.17**
Total	.33	.19	.19	**.39**	.33	.28
	.31	.23	.27	**.38**	.28	.34

Table 7.7: Detailed results of the cross-validation experiment with João Pires was the 'training' pianist. For each piece, the correlations between predicted and actual curves from all pianists are given. Each cell is divided into two rows corresponding to *dynamics* and *tempo* correlations, respectively. The highest correlations in each row are printed in bold. The average over all pieces is given in the last two rows (reproduced from the Table 7.3)

Piece	DB	RB	GG	MP	AS	MU
kv279:1:1	.33	.14	.05	.40	**.48**	.16
	.37	.26	.32	.16	.34	.25
kv279:1:2	.18	.07	-.03	**.19**	.17	.16
	.25	.20	.24	.19	**.45**	.28
kv280:1:1	.48	.42	.12	.44	**.53**	.30
	.44	.44	.24	.51	.55	**.64**
kv280:1:2	**.27**	.15	.06	.20	**.27**	.16
	.31	**.38**	.26	.17	.30	.27
kv280:2:1	.45	.45	.20	**.52**	.46	.42
	.47	.52	.67	.50	**.73**	.56
kv280:2:2	.40	.39	.04	**.43**	.38	.33
	.47	.49	.43	.35	**.66**	.45
kv280:3:1	**.74**	.64	.35	.54	**.74**	.59
	.76	.73	.73	.80	**.85**	.72
kv280:3:2	.50	**.56**	.39	.54	.54	.55
	.31	.31	.30	.28	**.42**	.37
kv282:1:1	.33	.25	.18	**.40**	.28	.17
	.23	.53	.16	.31	.60	**.65**
kv282:1:2	**.42**	.32	.21	.25	.30	.27
	.06	.28	.19	.18	**.41**	.37
kv283:1:1	.28	-.04	.34	.32	**.38**	.27
	.05	.00	.02	.04	**.24**	.10
kv283:1:2	.14	**.21**	.05	.03	.14	.06
	.04	.00	**.18**	.10	.16	.10
kv283:3:1	.70	-.31	.60	.49	**.74**	.32
	.54	.39	.50	.38	**.72**	.34
kv283:3:2	.51	.00	.47	.36	**.61**	.28
	.39	.25	.35	.22	**.39**	.13
kv332:2	.12	-.01	.16	.08	**.25**	.18
	.38	.17	.14	.14	.11	.28
Total	.36	.17	.20	.31	**.40**	.26
	.32	.29	.28	.25	**.41**	.32

Table 7.8: Detailed results of the cross-validation experiment with András Schiff was the 'training' pianist. For each piece, the correlations between predicted and actual curves from all pianists are given. Each cell is divided into two rows corresponding to *dynamics* and *tempo* correlations, respectively. The highest correlations in each row are printed in bold. The average over all pieces is given in the last two rows (reproduced from the Table 7.3)

Piece	DB	RB	GG	MP	AS	MU
kv279:1:1	.27	.17	.18	.35	**.37**	.26
	.36	.31	.22	.36	.29	**.50**
kv279:1:2	-.10	.10	-.02	.11	-.04	**.13**
	.31	.35	.35	.28	.34	**.47**
kv280:1:1	.43	.36	.06	.36	.34	**.58**
	.40	.25	.26	.55	.42	**.78**
kv280:1:2	.23	.13	.07	.14	.17	**.36**
	.42	.37	.42	.37	.47	**.54**
kv280:2:1	.41	.40	.18	**.50**	.42	.44
	.53	.34	.52	.58	.59	**.69**
kv280:2:2	.37	**.43**	.04	.36	.34	.32
	.44	.13	.39	.42	.50	**.58**
kv280:3:1	.65	.65	.39	.54	.63	**.67**
	.72	.66	.62	**.82**	.68	.77
kv280:3:2	.37	.38	.31	.33	.45	**.56**
	.51	**.52**	.44	.49	.43	.51
kv282:1:1	**.35**	.22	.21	.27	.27	.28
	.38	.45	.36	.44	.44	**.72**
kv282:1:2	**.35**	.22	.17	.33	.18	.24
	.23	.36	.33	.42	.46	**.52**
kv283:1:1	.27	-.07	**.38**	.35	.28	.29
	.21	.10	.12	.16	.22	**.31**
kv283:1:2	.16	.16	.15	.17	.13	**.23**
	.17	.18	.21	.23	.22	**.32**
kv283:3:1	.55	.22	.58	.58	.41	**.78**
	.30	.28	.42	.42	.42	**.52**
kv283:3:2	.31	-.05	.39	.30	.29	**.46**
	.19	.15	.29	.21	.32	**.36**
kv332:2	.12	.05	.21	.12	.23	**.25**
	.32	.17	.14	.22	.16	**.35**
Total	.27	.18	.21	.28	.26	**.38**
	.34	.30	.32	.36	.37	**.50**

Table 7.9: Detailed results of the cross-validation experiment with Mitsuko Uchida was the 'training' pianist. For each piece, the correlations between predicted and actual curves from all pianists are given. Each cell is divided into two rows corresponding to *dynamics* and *tempo* correlations, respectively. The highest correlations in each row are printed in bold. The average over all pieces is given in the last two rows (reproduced from the Table 7.3)

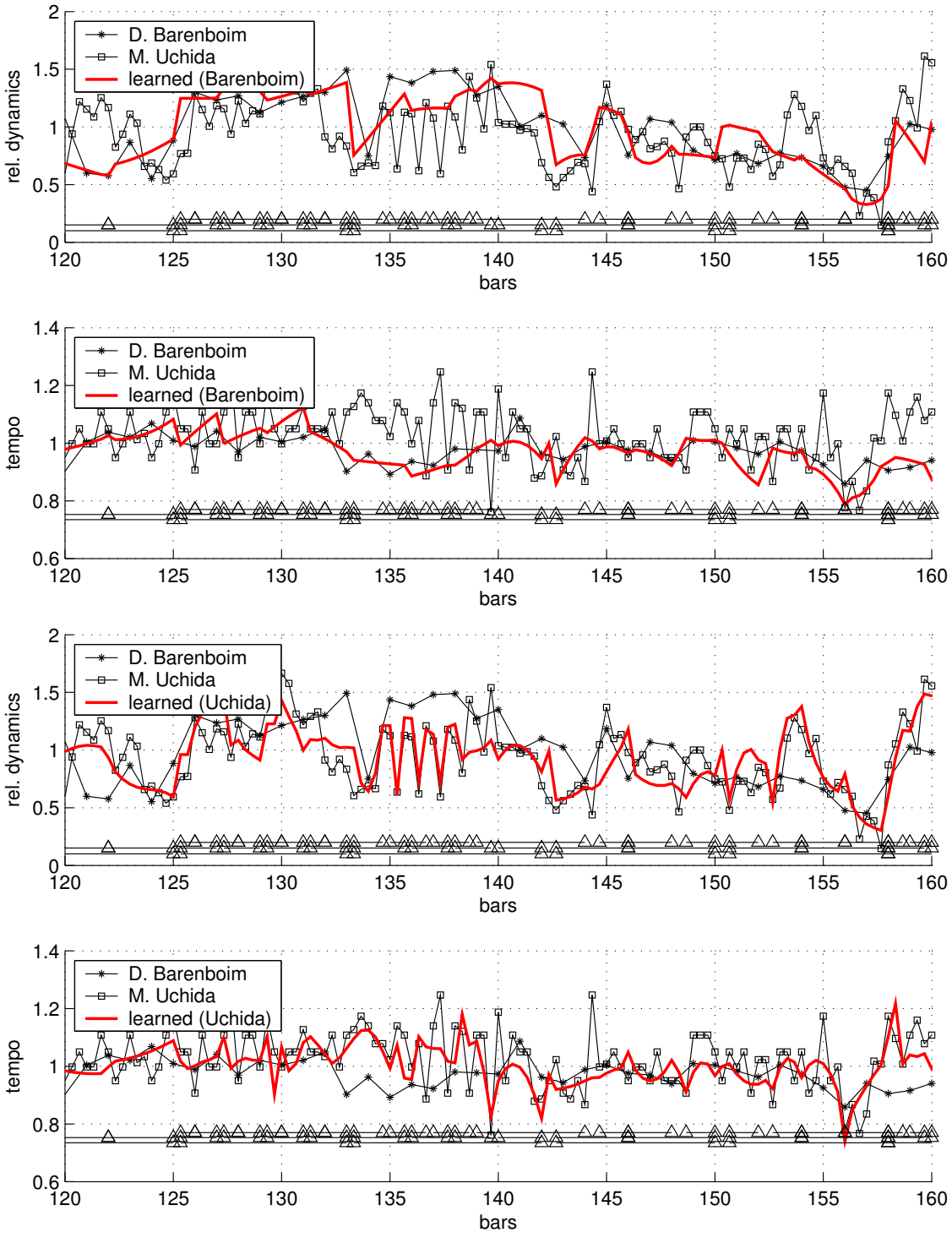

Figure 7.2: Dynamics and tempo curves produced by DISTALL on test piece (Sonata K.283, 3^{rd} mvt., 2^{nd} section, mm.120–160) after learning from Daniel Barenboim (top) and Mitsuko Uchida (bottom), compared to the artists' real curves as measured from the recordings.

the sequential nature of music is addressed by representing performances as strings and using string kernels in conjunction with kernel partial least squares and support vector machines. The string kernel approach is shown to achieve better performance than the best results obtained in [Widmer and Zanon, 2004]. A clear result from both works is that identification of pianists from their recordings is an extremely difficult task.

The pianists studied in the present paper are identical to those in [Widmer and Zanon, 2004] and [Saunders *et al.*, 2004]; unfortunately, the sets of recordings differ considerably (because manual phrase structure analyses, which are needed in our approach, were available only for certain pieces), so a direct comparison of the results is impossible. Still, to illustrate what can be achieved with a relational representation and learning algorithm, we briefly describe a classification experiment with DISTALL.

Each of the 15 pieces is set aside once. The 6 performances of that piece (one by each pianist) are used as test instances. A model of each pianist is built from his/her performances of the remaining 14 pieces. The result is two predicted curves per pianist for the test piece (for tempo and dynamics), which we call model curves. The final classification of a pianist, represented by his/her tempo and dynamics curves t_t and t_d on the test piece, is then determined as

$$c(t_t, t_d) = argmax_{p \in P}(\frac{corr(t_t, m_{pt}) + corr(t_d, m_{pd})}{2})$$

where P is set of all pianists and m_{pt} and m_{pd} are the pianists' model tempo and dynamics curves. In other words, the performance is classified as belonging to the pianist whose model curves exhibit the highest correlation (averaged over tempo and dynamics) with the test curves. For each pianist, DISTALL is tested on the 15 test pieces, which gives a total number of 90 test performances. The *baseline* accuracy – the success rate of pure guessing – is 15, or 16.67%. The confusion matrix of the experiment is given in Table 7.10.

Again, it turns out that the artists are identifiable to varying degrees, but the recognition accuracies are all clearly above the baseline. In particular, note that the system correctly identifies performances by Uchida in all but one case. Obviously, the learner succeeds in reproducing something of the artists' styles in its model curves. While these figures seem to compare very favourably to the accuracies reported in [Widmer and Zanon, 2004] and [Saunders *et al.*, 2004], they cannot be compared directly, because different recordings were used and, more importantly, the level of granularity of the training and test examples are different (movements in [Widmer and Zanon, 2004; Saunders *et al.*, 2004] vs. sections in this paper), which probably makes our learning task easier.

	prediction						
pianist	DB	RB	GG	MP	AS	MU	Acc.[%]
DB	11	0	0	2	2	0	73.3
RB	1	12	1	0	0	1	80.0
GG	1	1	10	0	0	3	66.7
MP	0	0	1	12	0	2	80.0
AS	1	0	2	0	10	2	66.7
MU	0	0	1	0	0	14	93.3
Total	-	-	-	-	-	-	**76.7**

Table 7.10: Confusion matrix of the pianist classification experiment. Rows correspond to the test performances of each pianist (15 per row), columns to the classifications made by the system. The rightmost column gives the accuracy achieved for all performances of the respective. The *baseline accuracy* in this 6-class problem is 16.67%.

7.5 Replicating Great Pianists?

Looking at Figure 7.2, one might be tempted to consider the possibility of automatic style replication: wouldn't it be interesting to supply the computer with the score of a new piece and have it perform it 'in the style of', say, Vladimir Horowitz or Arthur Rubinstein? This question is invariably asked when we present this kind of research to the public. Unfortunately (?), the answer is: while it might be interesting, it is not currently feasible.

For one thing, despite the huge effort we invested in measuring expressive timing and dynamics in recordings, the amount of available training data is still vastly insufficient vis-a-vis the enormous complexity of the behaviour to be learned. And secondly, the sort of crude beat-level variations in tempo and general loudness capture only a very small part of what makes an expressive interpretation; essential details like articulation (e.g., staccato vs. legato), pedalling, the shaping of individual voices, etc. are missing (and will be very hard to measure from audio recordings at all). A computer performance based only on applying these beat-level tempo and loudness changes will not sound anything like a human performance, as can be readily verified experimentally. Thus we have to admit that the title we chose for this chapter is a bit pretentious: the computer cannot be expected to play like the great pianists – at least not given the current methods and available training data. It can, however, extract aspects of personal style from recordings by great pianists, as has been shown in our experiments.

Chapter 8

Conclusion

The work presented in this thesis is situated on the crossroads between machine learning and musicology. On the machine learning side, we have introduced a new machine learning algorithm DISTALL. DISTALL belongs to the family of distance-based learning algorithms, and more precisely, relational instance-based learners. On the application side, we have presented the difficult real-world learning task rooted in the expressive music performance research. We have investigated to what extent a machine can automatically build expressive models of certain aspects of expressive piano performances. The focus was on the two most important dimensions of expressive performances, namely tempo and dynamics. Various interesting applications of our algorithm ranging from automated generation of expressive music performances to identifying great pianists were presented and discussed.

In Chapter 2 we discussed the basic framework for our algorithm – instance and distance based learning. We also introduced the relational formalism and discussed why distance based learning is more difficult in this representation than in the 'mainstream', propositional language. We discussed advantages and drawbacks of various set distance measures which potentially could be implemented in relational instance-based learner and argued that the set distance measure based on optimal matching have very strong theoretical properties: It is a metric, it takes advantage of all 'elementary' distances between set elements, and it is intuitive. At the same time it can be implemented in the way that the combinatorial explosion with increasing set sizes is avoided. Indeed, exponential increase in complexity with the growing set sizes is one of the main problems with the set distance measures in general.

Chapter 3 is where we presented our new learning algorithm. We have discussed our guiding principles for DISTALL and gave in-depth discussion of its main technical aspects: representation of the learning input, mechanisms for structuring the learning input and efficient computation of the optimal set matching distance. We than put all pieces together and gave the detailed algorithmic description of DISTALL's working principles.

The testbed for assessing DISTALL's learning capabilities is introduced in Chapter 4. The task is rooted in the expressive performance research and is a small step

in a research endeavour pursued in our lab that aims at building quantitative models of expressive music performance via inductive machine learning methods. The task is posed in the way that an instance-based learner can be applied to learn expressive tempo and dynamic shapes at a rather high level of the musical structure, namely the level of musical phrases. One of the main contribution of the Chapter 4 is that we have shown how rather unstructured expressive tempo and dynamics curves can be structured in a hierarchical way, resulting in training examples which in turn can be used by DISTALL (and by any other instance-based learner) to operate upon it. Indeed, one of the main difficulties in any real world machine learning task is to (meaningful) prepare the data in such a way that the learner can 'understand' and operate on. Additionally, a proposal of a complementary learning system constituting of instance-based phrase-level learner and PLCG note-level learner was made (and empirically tested later on).

Experimental evaluation of DISTALL and our learning system in general was presented in Chapter 5. After rather disappointing first experiments with the propositional k-nearest neighbor (where the reader could also get impression of the difficulty of the learning task), we have shown various incremental improvements in learning procedure resulting in the system which at least on some pieces have reached surprisingly good performance. Applying DISTALL to the same learning setting resulted in turn in performance leap regarding all objective error measures we have proposed. DISTALL also achieved a significantly better performance than the current state of the art relational instance-based learner RIBL. Further improvements were also presented, including richer phrase representation and, more importantly, exploiting the power of FOL by introducing temporal context information. It is the latter setting which made further significant performance leap for both relational learners. However, DISTALL has been able to better exploit the temporal information and has again clearly outperformed RIBL. The experiments with DISTALL in this last setting resulted in the system which has shown high level of predictive performances for both domains (tempo and dynamics), and almost all pieces from our dataset (e.g., see Figure 5.3 and Table 5.15).

Different further applications of our distance measure are presented in chapters 6 and 7. In Chapter 6 we have tried – with the help of the DISTALL's similarity measure – to assess the level of 'consistency' at the phrase level of a concert pianist playing different pieces of Mozart. The consistency is assessed via the level of correlation between 'objective', score-based similarity measure and the performance-based similarity measure (implied by the level of correlation between pianist interpretation of two different phrases). We have shown that the rather high level of correlation hold between these two measures, at least for the subset of the most similar phrase pairs (as indicated by the score-based similarity measure). While in a certain sense it can be considered as a reliability proof of our score-based similarity measure, it can on the other site be regarded as indication of stylistic consistency of the performer.

In Chapter 7 we have applied DISTALL for analyzing famous pianists. Interestingly, even with crude, beat-level performance representation extracted in semi-

automated manner from audio CDs, DISTALL was able to catch certain aspects of playing styles of great pianists. The technique of building expressive performance models is than applied in a straightforward manner for recognizing the pianists. As expected, some pianists were recognizable to the better degree than the others, but the overall results were surprisingly good and compare very favourably to the results published recently.

From the technical point of view, introducing a new learning algorithm is of course the most interesting part of the thesis. Moreover, presented experimental success of DISTALL regarding comparison to RIBL can be considered significant for at least three reasons: (1) the difficulty of the learning task at hand, (2) substantial volume of the dataset (containing more than 5500 melody notes and more than 1200 phrases, comprising half an hour of piano music), (3) consequent superior results in the direct comparison through all three experiment groups (initial representation, enriched phrase-level attributes and introduction of temporal context). Of course, the better results should be attributed to the optimal matching set distance measure, which has some compelling attributes, as we have argued in the text. On the other site, it is virtually the only difference between RIBL and DISTALL since the other aspects are rather similar. Especially the (sub)set formation, second main building block of both algorithms, works through the same hierarchical, recursive procedure. However, different set distance measures have a profound effect on the overall distance measures in such a hierarchical configuration, since the distance computed at one level forms the foundation for, and has the crucial influence on the distance which has to be computed on the higher level. It is thus clear that in the case of a hierarchical distance measure in general, special care should be taken in designing the distance measure at one level of hierarchy.

However, hierarchy is a key for efficient implementation of DISTALL, since it narrows the set sizes on which the optimal matching distance has to be applied. Without grouping the literals in an appropriate way, it would be intractable – with current computational resources – to apply the optimal matching set distance to nontrivial, real world problems. Hierarchy has been proven to be en efficient way in reducing complexity of the problems in many different technical systems. Recently, there is even increasing evidence from neuroscience that the hierarchical representations are utilized in some parts of biological neuronal systems, most notably mammalian (and human) neocortex ([Reisenhuber and Poggio, 1999; Wallis and Rolls, 1996]). It is speculated that the hierarchical representations are maintained in neocortex for similar reasons as we have argued above for the case of DISTALL, namely reduction of computational and memory complexity ([Reisenhuber and Poggio, 1999]). While, of course, we do not intend to imply that the representational formalism and working principles used by DISTALL bear any similarity with the representations maintained in the biological neural systems, we do believe that the hierarchical representation is indeed an efficient way for scaling up and dealing with complexity in general.

Of course, better learning performance on one learning task does not necessarily generalize to other learning tasks. Application of optimal matching set distance

on relatively small, hierarchical subsets allows for efficient implementation of DISTALL, which in turn allows DISTALL to be applied and tested for many complex, real-world problems. E.g., RIBL has been shown to produce superior generalization performance compared to other ILP algorithms for the problem of diterpene structure elucidation from ^{13}C NMR spectra (see [Dzeroski *et al.*, 1996]). Diterpenes are organic compounds of low molecular weight that are based on a skeleton of 20 carbon atoms. They are of significant chemical and commercial interest because of their use as lead compounds in the search for new pharmaceutical effectors. The structure elucidation of diterpenes based on ^{13}C NMR spectra is usually done manually by human experts with specialized background knowledge. In [Dzeroski *et al.*, 1996] it has been shown that RIBL's accuracy on this task is significantly higher than the accuracies achieved with all other relational (and propositional) algorithms and is in the range of the accuracy of human experts. It can be a topic of future work to examine, if performance gain achieved with DISTALL (as compared to RIBL) on our music task also translates to this other interesting task. Indeed, there are many domains and problems with structural representations, where the introduction of relational language (and algorithms) has been proven successful (e.g. see [King *et al.*, 2004] for the latest successes). It would be interesting to see if DISTALL can achieve any additional benefits at least on some of them.

Regarding musicology and, more precisely, expressive performance research, this thesis is to our knowledge the quantitatively largest attempt to model and generate expressive performances on the phrase level. There have been experiments with case-based learning for generating expressive phrases of good musical quality in jazz ballads ([Arcos and de Mantaras, 2001; de Mantaras and Arcos, 2002]). In addition to somewhat different assumptions (the system described in [de Mantaras and Arcos, 2002] made use of musical background knowledge), the work presented here was conducted on much larger data corpus. One interesting implication of our phrase-level decomposition procedure (section 4.3.3) regarding musicology is the experimental finding that quadratic functions are much less a reasonable model class for timing as it has been believed (but never systematically tested on such a large corpus of real performances) in musicology ([Todd, 1989; Windsor and Clarke, 1997]). While our decomposition process with quadratic functions worked very well in dynamics domain, it proved much less suited for timing (see Table 5.6). It would be interesting to address this issue in future work. Discovering a model class which better approximate real expressive performance curves for timing in a process akin to ours would certainly be interesting for musicology.

Further improvements regarding prediction ability of our expressive models in general could be achieved with introduction of additional phrase-level features, coupled with standard feature selection methods. We have shown in section 5.2.4 that given additional, richer phrase representation, a clear gain in learning performance is achieved. Currently, the phrase-level features are 'selected' in a rather ad hoc manner, based on opinions of our colleagues with background in musicology. We believe that with additional features and automated feature selection, the subset with best

generalization ability on our (limited) dataset could be found, which would improve the overall quality of the system. It would be also interesting to see if the prediction quality of the system transfers to other periods of music (other than Mozart sonatas from classical period).

The same could be said for the experiments reported in Chapter 7 regarding great pianists. Would the system be able to (learn to) recognize great pianists playing pieces from e.g., romantic period? Although we consider recognition accuracy of on average 76.7% on this difficult 6-class classification task surprisingly high (also compared with recent results reported in [Widmer and Zanon, 2004] and [Saunders *et al.*, 2004]), it should be noted that it is seriously compromised: Our data set is divided according to sections, rather than movements as done in [Widmer and Zanon, 2004; Saunders *et al.*, 2004]. Due to repetitions in different sections, our setting can clearly be considered easier to learn. On the other hand, all results reported in [Widmer and Zanon, 2004] and [Saunders *et al.*, 2004] refer to pairwise performer classifications, i.e., the results are assessed via many 2-class learning problems with 50% baseline accuracy. A direct, 6-class learning problem is unarguable much more difficult, and we are to our knowledge first who defined the learning task in this way. Unfortuantelly, we could not have used the same data as in [Widmer and Zanon, 2004] and [Saunders *et al.*, 2004], since the phrase analysis – which is needed for our system – has not been available for all pieces by the time of experiments. Further data gathering and experiments are thus needed in order to clearify, if our algorithm and learning method favourably compares with other machine learning methods on this interesting task, which – not at least due to efforts in our research group – recently became popular in machine learnig community.

Bibliography

Pieter Adriaans and Dolf Zantinge. *Data mining*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

Josep Lluis Arcos and Ramon Lopez de Mantaras. An interactive case-based reasoning approach for generating expressive music. *Applied Intelligence*, 14(1):115–129, 2001.

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

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Emilios Cambouropoulos and Gerhard Widmer. Automatic motivic analysis via melodic clustering. *Journal of New Music Research*, 29(4):303–317, 2001.

E. Cambouropoulos. Melodic cue abstraction, similarity, and category formation: A formal model, 2001.

Nello Cristianini and John Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

Ramon Lopez de Mantaras and Josep Lluis Arcos. Ai and music from composition to expressive performance. *AI Mag.*, 23(3):43–57, 2002.

Thomas G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.

Simon Dixon and Emilios Cambouropoulos. Beat tracking with musical knowledge. In *Europian Conference on Artificial Intelligence (ECAI'00)*, pages 626–630, 2000.

Simon Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 2001.

Simon Dixon. An empirical comparison of tempo trackers, 2001.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

Sašo Džeroski, Steffen Schulze-Kremer, Karsten R. Heidtke, Karsten Siems, Dietrich Wettschereck, and Hendrik Blockeel. Diterpene structure elucidation from 13c nmr spectra with inductive logic programming. *Applied Artificial Intelligence*, 12(5):363–383, July-August 1998. Special Issue on First-Order Knowledge Discovery in Databases.

Saso Dzeroski, Steffen Schulze-Kremer, Karsten R. Heidtke, Karsten Siems, and Dietrich Wettschereck. Applying ILP to diterpene structure elucidation from 13 c NMR spectra. In *Inductive Logic Programming Workshop*, pages 41–54, 1996.

Thomas Eiter and Heikki Mannila. Distance measures for point sets and their computation. *Acta Informatica*, 34(2):109–133, 1997.

Werner Emde and Dietrich Wettschereck. Relational instance-based learning. In *Proceedings of the 13th International Conference on Machine Learning*, 1996.

G. Frege. Begriffsschrift, eine der arithmetischen nachgebildete formelsprache des reinen denkens (gekuerzter nachdruck). In K. Berka and L. Kreiser, editors, *Logik-Texte: Kommentierte Auswahl zur Geschichte der Modernen Logik (vierte Auflage)*, pages 82–107. Akademie-Verlag, Berlin, 1986.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

Johannes Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

A. Gabrielsson. The performance of music. In *The Psychology of Music (2nd ed.)*, pages 501–602. Academic Press, San Diego, CA, 1999.

A. Gabrielsson. Music performance research at the millenium. *Psychology of Music*, 31(3):221–272, 2003.

Werner Goebl. Melody lead in piano performance: Expressive device or artifact? *Journal of the Acoustical Society of America*, 110(1):563–572, 2001.

Ralph P. Grimaldi. *Discrete and Combinatorial Mathematics: An Applied Introduction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

Nicolas Helft. Induction as nonmonotonic inference. In *Proceedings of the first international conference on Principles of knowledge representation and reasoning*, pages 149–156, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

W. B. Hewlett and E. Selfridge-Field. *Melodic Similarity: Concepts, Procedures, and Applications*. MIT Press, Cambridge, MA, 1998.

Daniel P. Huttenlocher and Klara Kedem. Computing the minimum hausdorff distance for point sets under translation. In *SCG '90: Proceedings of the sixth annual symposium on Computational geometry*, pages 340–349, New York, NY, USA, 1990. ACM Press.

R.D. King, K.E. Whelan, F.M. Jones, P.G.K. Reiser, C.H. Bryant, S.H. Muggleton, D.B. Kell, and S.G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 15 January 2004.

Ulf Kronman and Johan Sundberg. Is the musical ritard an allusion to physical motion? In Alf Gabrielsson, editor, *Action and Perception in Rhythm and Music*, volume 55, pages 57–68. Royal Swedish Academy of Music, Stockholm, 1987.

Pat Langley. *Elements of machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

Søren Tjagvad Madsen and Gerhard Widmer. Exploring pianist performance styles with evolutionary string matching. *International Journal on Artificial Intelligence Tools*, 15(4):495–514, 2006.

Kurt Mehlhorn. *Graph algorithms and NP-completeness*. Springer-Verlag New York, Inc., New York, NY, USA, 1984.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

S. Muggleton and C. Feng. Efficient induction of logic programs. 1990.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

Shan-Hwei Nienhuys-Cheng, R. De Wolf, and Ronald de Wolf. *Foundations of Inductive Logic Programming*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

C. Palmer and J.C. Brown. Investigations in the amplitude of sounded piano tones. *Journal of the Acoustical Society of America*, 90(1):60–66, 1991.

C. Palmer. Mapping musical thought to musical performance. *Journal of Experimental Psychology: Human Perception and Performance*, 15(12):331–346, 1989.

C. Palmer. Music performance. *Annual Review of Psychology*, 48:115–138, 1997.

Luc De Raedt. *Interactive theory revision: an inductive logic programming approach*. Academic Press Ltd., London, UK, UK, 1992.

Luc De Raedt. Attribute-value learning versus inductive logic programming: The missing links (extended abstract). In D. Page, editor, *ILP '98: Proceedings of the 8th International Workshop on Inductive Logic Programming*, volume 1446 of

Lecture Notes in Artificial Intelligence, pages 1–8, London, UK, 1998. Springer-Verlag.

J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. volume 1446, pages 271–280, 1998.

Jan Ramon and Maurice Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37(10):765–780, 2001.

M. Reisenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

Bruno H. Repp. Diversity and commonality in music performance: An analysis of timing microstructure in Schumann's "Träumerei". 92(5):2546–68, 1992.

B.H. Repp. Some empirical observations on sound level properties of recorded piano tones. *Journal of the Acoustical Society of America*, 93(2):1136–44, 1993.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

Craig Saunders, David R. Hardoon, John Shawe-Taylor, and Gerhard Widmer. Using string kernels to identify famous performers from their playing style. In *Proceedings of the 15th European Conference on Machine Learning*, 2004.

C.E. Seashore. *Psychology of Music*. McGraw-Hill (Reprinted 1967 by Dover Publications, New York.), 1938.

Efstathios Stamatatos and Gerhard Widmer. Music performer recognition using an ensemble of simple classifiers. In *Proceedings of the 15th European Conference on Artificial Intelligence*, 2002.

Alfred Tarski. Der wahrheitsbegriff in den formalisierten sprachen. *Studia Philosophica*, 1:261–405, 1936.

Alfred Tarski. *Logic, Semantics, Metamathematics*. Oxford University Press, 1956. Edited by J. H. Woodger.

A. Tobudic and G. Widmer. Relational IBL in music with a new structural similarity measure. In T. Horváth and A. Yamamoto, editors, *Proceedings of the 13th International Conference on Inductive Logic Programming (ILP'03), Szeged, Hungary*, volume 2835, pages 365–382, 2003.

Asmir Tobudic and Gerhard Widmer. Learning to play mozart: Recent improvements. In *Proceedings of the IJCAI'03 Workshop on Methods for Automatic Music Performance and their Applications in a Public Rendering Contest, 18th Joint International Conference on Artificial Intelligence (IJCAI'03), Acapulco, Mexico*, 2003.

Asmir Tobudic and Gerhard Widmer. Playing mozart phrase by phrase. In K. D. Ashley and D. G. Bridge, editors, *Proceedings of the 5th International Conference on Case-based Reasoning (ICCBR'03), Trondheim, Norway*, pages 552–566, 2003.

Asmir Tobudic and Gerhard Widmer. Case-based relational learning of expressive phrasing in classical music. In *Proceedings of the 7th European Conference on Case-based Reasoning (ECCBR'04), Madrid, Spain*, 2004.

Asmir Tobudic and Gerhard Widmer. Learning to play like the great pianists. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05), Edinburgh, Scorland.*, 2005.

Asmir Tobudic and Gerhard Widmer. Relational IBL in classical music. *Machine Learning*, 64(1-3):5–24, 2006.

McAngus N. Todd. Towards a cognitive theory of expression: The performance and perception of rubato. *Contemporary Music Review*, 4:405 – 416, 1989.

McAngus N. Todd. The dynamics of dynamics: A model of musical expression. *Journal of the Acoustical Society of America*, 91(6):3540–50, 1992.

G. T. Toussaint and B. K. Bhattacharya. Optimal algorithms for computing the minimum distance between two finite planar sets. *Pattern Recognition Letters*, 2:79–82, 1983.

G. Wallis and E. T. Rolls. A model of invariant object recognition in the visual system. *Prog Neurobiol.*, 51:167–194, 1996.

A. N. Whitehead and B. Russel. *Principia Mathematica*. Cambridge University Press, Cambridge, 2nd edition, 1910–1913.

Gerhard Widmer and Werner Goebl. Computational models of expressive music performance: The state of the art. *Journal of New Music Research*, 33(3):203–216, 2003.

Gerhard Widmer and Asmir Tobudic. Playing mozart by analogy: Learning multi-level timing and dynamics strategies. In *Proceedings of the ICAD Workshop on Performance Rendering Systems, 8th International Conference on Auditory Display (ICAD'2002), Kyoto, Japan*, 2002.

Gerhard Widmer and Asmir Tobudic. Playing mozart by analogy. *Journal of New Music Research*, 32(3):259–268, 2003.

Gerhard Widmer and Patrick Zanon. Automatic recognition of famous artists by machine. In *Proceedings of the 17th European Conference on Artificial Intelligence*, 2004.

Gerhard Widmer, Simon Dixon, Werner Goebl, Elias Pampalk, and Asmir Tobudic. In search of the Horowitz factor. *AI Magazine*, 24(3):111–130, 2003.

Gerhard Widmer. A machine learning analysis of expressive timing in pianists' performances of schumann's "träumerei". In A. Friberg and J. Sundberg, editors, *Proceedings of the KTH Symposion on Grammars for Music Performance*, pages 69–81, Stockholm, Sweden: Department of Speech Communication and Music Acoustics, 1995.

Gerhard Widmer. Modeling rational basis for musical expression. *Computer Music Journal*, 19:76–96, 1995.

Gerhard Widmer. Learning expressive music performance: The structure-level approach. *Journal of New Music Research*, 25:179–205, 1996.

Gerhard Widmer. Large-scale induction of expressive performance rules: First quantitative results. In I. Zannos, editor, *Proceedings of the 2000 International Computer Music Conference*, pages 344–347, Berlin, Germany, 2000.

Gerhard Widmer. Machine discoveries: A few simple, robust local expression principles. *Jorunal of New Music Research*, 31(3):37–50, 2002.

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

W. L. Windsor and E.F. Clarke. Expressive timing and dynamics in real and artificial musical performances: Using an algorithm as an analytical tool. *Music Perception*, 15(2):127 – 152, 1997.

Ian H. Witten and Eibe Frank. *Data mining: practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

Asmir Tobudic

Personal information:

- Date of birth: 12.01.1974
- Place of birth: Tuzla, Bosnia and Hercegovina
- Nationality: Austria
- Family Status: married

Education:

- 1980 – 1988: Elementary School, Tuzla
- 1982 – 1988: Elementary Music School, Classical Guitar, Tuzla
- 1988 – 1992: Informatics Gymnasium "Mesa Selimovic", Tuzla
- 1988 – 1992 Secondary Music School, Classical Guitar, Sarajevo
- 1993 – 1994: Faculty of Electrical Engineering, University of Tuzla
- 1995 – 2001: Faculty of Electrical Engineering and Information Technology, Vienna University of Technology
- 2001 – 2006: Ph.D. on TU-Vienna, research carried out on Austrian Research Institute for Artificial Intelligence

Research Interests:

- Computational Neuroscience, Machine Learning (ML), Artificial Intelligence
- Solving real-world problems with AI- and ML-methods

Positions:

- 2001 – 2006: Research position at Austrian Research Institute for AI
- 2006 – present: Director of ViennaClassic.com