Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Artificial Intelligence 146 (2003) 129–148

Artificial Intelligence

www.elsevier.com/locate/artint

Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries

Gerhard Widmer [a,b]

[a] *Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, Vienna, Austria*
[b] *Austrian Research Institute for Artificial Intelligence, Vienna, Austria*

Received 22 November 2001; received in revised form 8 November 2002

Abstract

This article presents a new rule discovery algorithm named PLCG that can find simple, robust partial rule models (sets of classification rules) in complex data where it is difficult or impossible to find models that completely account for all the phenomena of interest. Technically speaking, PLCG is an *ensemble learning method* that learns multiple models via some standard rule learning algorithm, and then combines these into one final rule set via clustering, generalization, and heuristic rule selection. The algorithm was developed in the context of an interdisciplinary research project that aims at discovering fundamental principles of expressive music performance from large amounts of complex real-world data (specifically, measurements of actual performances by concert pianists). It will be shown that PLCG succeeds in finding some surprisingly simple and robust performance principles, some of which represent truly novel and musically meaningful discoveries. A set of more systematic experiments shows that PLCG usually discovers significantly simpler theories than more direct approaches to rule learning (including the state-of-the-art learning algorithm RIPPER), while striking a compromise between coverage and precision. The experiments also show how easy it is to use PLCG as a meta-learning strategy to explore different parts of the space of rule models.

Keywords: Machine learning; Data mining; Rule discovery; Ensemble methods; Meta-learning; Partial models; Expressive music performance

E-mail address: gerhard@ai.univie.ac.at (G. Widmer).
URL address: http://www.ai.univie.ac.at/~gerhard (G. Widmer).

0004-3702/03/$ – see front matter
doi:10.1016/S0004-3702(03)00016-X

1. Introduction

In application-oriented sciences like data mining and machine learning, it is often the case that challenges posed by some concrete practical application problem lead to the development of new, often general solutions with new properties. The work to be described in the present article is a prime example of that. What will be presented here is a new rule learning algorithm that can discover extremely simple partial rule models from large amounts of complex, real-world data. The motivating application problem, in this case, comes from the (unusual?) domain of music.

The contribution of this article is two-fold: on the one hand, a new, general-purpose rule discovery algorithm is presented and systematically compared to alternative rule learning methods. The distinctive features of the new algorithm are that it is a meta-algorithm that can be 'wrapped around' any given rule learning algorithm, and that it provides an elegant way to explore the trade-off between the generality (coverage), the precision, and the complexity of the theories (rule sets) looked for.

On the other hand, this article provides proof that automated discovery techniques can also fruitfully be applied to 'soft' sciences like music, and can in fact make significant new discoveries. So far, AI methods, particularly inductive learning, have been used as tools for scientific discovery predominantly in the natural sciences (e.g., [13,15,16,21,23–25]). There has been far less work in the so-called humanities of softer sciences, which at first sight would seem to defy attempts at machine-based discovery because of a lack of strict laws that can be discovered. The present article attempts to show that AI can be just as successful in such areas, even in 'artistic' domains like music.

The context for this work is a long-term inter-disciplinary research project situated at the intersection of the scientific disciplines of Artificial Intelligence and Musicology [29,31].[1] The goal is to use AI methods to study the complex phenomenon of *expressive music performance*. We want to understand what great musicians do when they interpret and play a piece of music, and to what extent an artist's musical choices are constrained or 'explained' by (a) the structure of the music, (b) common performance practices, and (c) cognitive aspects of music perception and comprehension. Formulating formal, quantitative models of expressive performance is one of the big open research problems in contemporary empirical musicology. Our project develops a new direction in this field: we use *inductive learning algorithms* to discover general and valid expression principles from (large amounts of) real performance data.

The main purpose of this research is *knowledge discovery*. We search for simple, general, interpretable models of aspects of expressive music performance (such as tempo and expressive timing, dynamics, articulation). To that end, we have compiled what is most probably the largest set of performance data (precise measurements of timing, dynamics, etc. of real musical performances) ever collected in empirical performance research. Specifically, we are analyzing large sets of recordings by highly skilled concert pianists, with the goal of discovering explainable patterns in the way the music is played. The results described here represent a first small, but significant, step towards this goal.

[1] See also the project web page at http://www.oefai.at/music/.

In Section 2, we first explain some basic concepts of our application domain, and then report on some problems encountered when standard machine learning algorithms were applied in a straightforward way. These experiences prompted us to develop the new rule algorithm PLCG, which is described in Section 3. The main purpose of PLCG is to find simple, robust partial theories (sets of classification rules that explain at least a part of the data) in complex data sets. PLCG achieves this by learning multiple theories via some standard rule learning algorithm, and then combining these theories into one final rule set via clustering, generalization, and heuristic rule selection. Section 4 demonstrates the potential of the approach by describing some extremely simple and general performance rule sets discovered by PLCG—some of the learned rules represent truly novel and musically meaningful discoveries. Section 5 then presents several more systematic experiments that compare PLCG to more 'direct' rule learning methods. The results indicate that PLCG finds more compact theories than state-of-the-art rule learners, while striking a compromise between generality and precision. They also demonstrate how easy it is to use PLCG as a meta-learning strategy to explore different parts of the space of possible theories.

2. The goal: Discovering fundamental principles of musical expression

2.1. The target: 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. Rather, they speed up at some places, slow down at others, stress certain notes or passages by various means, and so on. The expressive nuances added by an artist are what makes a piece of music come alive (and what makes some performers famous). The most important dimensions available to a performer (a pianist, in particular) are tempo, dynamics (loudness variations), and articulation (the way successive notes are connected).

Expressive variation is more than just a 'distortion' of the original (notated) piece of music. In fact, the opposite is the case: the notated music score is but a small part of the actual music. Not every intended nuance can be captured in a limited formalism such as common music notation, and the composers were and are well aware of this. The performing artist is an indispensable part of the system, and expressive music performance plays a central role in our musical culture. That is what makes it a central object of study in the field of musicology.

Our approach to studying this complex phenomenon is to collect large corpora of performance data (i.e., exact measurements of onset and offset times and loudness of each note as played in a performance), and to apply inductive learning algorithms to find models that compactly characterize various classes of situations that are treated in a similar way by the performer (such as 'situations where the performer slows down' vs. 'situations where s/he speeds up'). At the moment, we limit ourselves to classical piano music.

2.2. Data and target concepts

The data used in our experimental investigations is by far the largest corpus of precisely measured and documented performances ever studied in musical performance research. We currently work with recordings of 18 complete Mozart piano sonatas (some six hours of music, more than 150,000 notes) by two different concert pianists, as well as performances of selected Chopin pieces by 22 different pianists. The pieces were played on a Bösendorfer SE290 computer-monitored grand piano. The Bösendorfer SE290 is a full concert grand piano with a special mechanism that measures and records every key and pedal movement with high precision. These measurements, together with the notated score in machine-readable form, provide us with all the information needed to compute expressive variations (e.g., tempo fluctuations). Collecting and preparing this data set was a formidable project in itself and in fact forced us to develop a range of novel intelligent music processing algorithms [1,2,6,7].

The data set that will serve as a basis for rule discovery in the particular experiments to be described here (i.e., the *training data*) consists of 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. The resulting dataset consists of more than 106,000 performed notes and represents some four hours of music.

The experiments described here were performed on the melodies (usually the soprano parts) only, which gives an effective training set of 41,116 notes. Each note is described by 29 attributes (10 numeric, 19 discrete) that represent both intrinsic properties (such as scale degree, duration, metrical position) and some aspects of the local context (e.g., melodic properties like the size and direction of the intervals between the note and its predecessor and successor notes, and rhythmic properties like the durations of surrounding notes etc., and some abstractions thereof).

Our goal at this stage of the project is to discover explanatory (partial) models of categorical performer actions such as speeding up vs. slowing down, at a very local, note-to-note level. In terms of performance parameters, we are looking at (local) tempo or timing, dynamics, and articulation. We defined the following discrete target classes:

(1) in the tempo dimension, a note N is assigned to class *ritardando* if the local tempo at that point is significantly (>2%) slower than the tempo at the previous note; the opposite class *accelerando* contains all cases of local speeding up;
(2) in dynamics, a note N is considered an example of class *crescendo* if it was played louder than its predecessor, and also louder than the average level of the piece; class *diminuendo* (growing softer) is defined analogously;
(3) in articulation, three classes were defined: *staccato* if a note was sounded for less than 80% of its nominal duration, *legato* if the proportion is greater than 1.0 (i.e., the note overlaps the following one), and *portato* otherwise; we will only try to learn rules for the classes *staccato* and *legato*.

A performed note is considered a counter-example to a given class if it belongs to one of the competing classes. (Note that due to some details of our class definitions, there will be some notes that are neither examples nor counter-examples of some concept.) The total

Table 1
Numbers of training examples of various categories

	Timing		Dynamics		Articulation		
	longer	shorter	louder	softer	staccato	legato	portato
slow 2/2	790	786	677	580	1,374	473	625
slow 3/4	1,209	1,169	1,081	942	1,371	1,326	1,177
slow 4/4	903	916	836	708	961	1,212	821
slow 3/8	157	153	151	100	150	216	125
slow 6/8	598	667	523	469	1,037	458	530
fast 2/2	1,945	1,925	1,662	1,447	3,227	1,305	1,394
fast 2/4	2,148	2,206	1,851	1,376	4,270	887	1,345
fast 3/4	2,048	1,972	1,838	1,467	3,611	1,387	1,289
fast 4/4	2,107	2,078	1,781	1,413	3,486	1,338	1,537
fast 3/8	738	711	623	435	1,387	337	343
fast 6/8	767	724	606	492	1,258	317	542
slow	3,657	3,691	3,268	2,799	4,893	3,685	3,278
fast	9,753	9,616	8,361	6,630	17,239	5,571	6,450
total	13,410	13,307	11,629	9,429	22,132	9,256	9,728

numbers of training examples (notes pertaining to each category) that result from this are summarized in Table 1, separately for different global tempi and time signatures of the corresponding sonata sections; this particular partitioning of the data will play a role in the following.

2.3. Previous results and problems

Musically speaking, this local, note-to-note definition of the target concepts and the low-level representation of the training examples (only single notes with very limited information about their context) is clearly unsatisfactory. We cannot expect an artist's expressive actions to be explainable by reference to such low-level features of the music only. More abstract structural aspects of the music, like phrase structure and harmonic structure, definitely play an important role and should be added to the representation of the data. Unfortunately, these aspects of the music are difficult to capture in a formal system, and we currently have no algorithms that could reliably compute them from the information given in the score—and given the sheer amount of data we are working with, performing a 'manual' structural analysis of the pieces is infeasible. Consequently, we cannot expect to find even close to perfect models of expressive performance at this level of representation.

Nevertheless, in a first suite of experiments [28], we succeeded in showing that even at this low level, there is structure in the data; learning algorithms like C4.5 [20] were able to find rule sets that predict the performer's choices with better than chance probability.

However, the improvement over the baseline accuracy was generally rather small (though statistically significant), which indicates that there are severe limits as to how much of a performer's behaviour can be explained at the note level. Moreover, the learned models were extremely complex. For instance, a decision tree discriminating between *accelerando*

(speeding up) and *ritardando* (slowing down) with 58.09% accuracy had 3037 leaves (after pruning)! That is clearly not desirable if our goal is knowledge discovery.

Now apart from the musical limitations mentioned above, the problem can also be attributed in part to an *inappropriate bias* of the learning algorithm employed. Decision tree learners attempt to build a global discriminative model that fully distinguishes between the members of the various classes and that makes predictions everywhere in the instance space. What seems more sensible in our application domain is to search for *partial* models that only explain what *can* be explained, and simply ignore those parts of the instance space where no compact characterization of the target classes seems possible. Moreover, given the nature of our data and target phenomena, we cannot expect very high levels of discriminative accuracy—we cannot assume the artist to be perfectly consistent and predictable.

In the following, we describe a rule learning algorithm named PLCG that was developed for this purpose. It will be shown that PLCG can find very simple partial models that still characterize a number of interesting subclasses of expressive performance behaviour. (Indeed, we will show that 4 simple rules are sufficient to predict 22.89% of the instances of local *ritardando* in our large data set, which contrasts nicely with the decision tree with 3037 leaves mentioned above.)

3. PLCG: Discovering simple partial models

3.1. The PLCG meta-algorithm

Given the goal of learning partial models, an obvious choice is to apply rule learning algorithms of the *set covering* variety (also known as *separate-and-conquer* learners [10]), such as FOIL [19] or RIPPER [4] (see Fig. 1 for a sketch of the basic structure). These algorithms learn theories one rule at a time, in each rule refinement step selecting a literal that maximizes some measure of discrimination (e.g., information gain). A rule is specialized until a given stopping criterion (typically based on the rule's purity or precision) is satisfied, and the overall learning process stops when no more rules can be found that satisfy this purity criterion (or, more generally, when some user-defined stopping criterion is satisfied). The stopping criteria are thus the natural entry point for the user to influence the generality and precision of the induced rules. In the context of our problem, we would require rather low levels of precision in the RULESTOPCRITERION (see Fig. 1). The degree of coverage of the resulting rules would then follow automatically, dictated by the data.

After some experimentation, we have chosen to pursue a more complex approach. The basic idea is to learn several models in parallel (from subsets of the data), search for groups of similar rules in these models, generalize these into summarizing rules (of varying degrees of generality), and then select those generalizations for the final model that optimize some (possibly global) user-defined criterion (which will typically be a trade-off function between coverage and precision). This strategy gives us more direct control over the overall coverage and precision of the induced models, and at the same time helps ameliorate one of the major problems of the greedy literal selection strategy of the underlying rule learner: the danger of selecting sub-optimal conditions due to the local

```
procedure RL(E)

Theory := {};
while not THEORYSTOPCRITERION(Theory,E) do
    Rule := FINDBESTRULE(E);
    E := E \COVERS(Rule,E);
    Theory := Theory ∪ Rule;
return(Theory);

procedure FINDBESTRULE(E)

Rule := {};
repeat
    Rule := Rule ∪ FINDBESTCONDITION(E);
    E := COVERS(Rule,E);
until RULESTOPCRITERION(Rule,E);
return(Rule);
```

Fig. 1. RL: A standard sequential covering algorithm for rule learning. FINDBESTCONDITION finds the most discriminating condition according to some heuristic criterion (e.g., FOIL's information gain criterion).

maximization of a given discrimination measure. In this sense, our approach — let us call it the PLCG (**P**artition + **L**earn + **C**luster + **G**eneralize) strategy — is inspired by the success of *ensemble methods* in machine learning (see [5] for a good overview). The corresponding algorithm is given in more detail in Fig. 2. The advantages or properties of this algorithm will be analyzed in more detail in Section 5.

3.2. An instantiation of PLCG

PLCG is really a *meta-algorithm*. It needs to be instantiated with three concrete algorithms (see Fig. 2): a rule learning algorithm L, a hierarchical clustering algorithm H, and a rule selection criterion or strategy S. For the following experiments, PLCG will be instantiated as follows:

For *rule learning*, PLCG uses the simple, straightforward sequential covering algorithm RL sketched in Fig. 1, with the standard information gain heuristic as used in FOIL [19] and the following stopping criteria:

(1) RULESTOPCRITERION(r,E) = `true` if purity $P(r, E) = p/(p+n) \geqslant MP_{RL}$,
(2) THEORYSTOPCRITERION(*Theory*,E) = `true` if no more rule r can be found with purity $P(r, E) = p/(p+n) \geqslant MP_{RL}$ and positive coverage $p \geqslant MC_{RL}$,

where p and n are the numbers of positive and negative examples, respectively, covered by a rule r on a set of examples E, and the required minimum purity MP_{RL} and minimum coverage MC_{RL} are user-specified parameters.[2] Obviously, high values of MP_{RL} will

[2] The subscript $_{RL}$ serves to distinguish these parameters from the purity and coverage parameters MP_{PLCG} and MC_{PLCG} used by PLCG's rule selection algorithm (see below).

Given:

- a set of training instances D
- a target concept (class) c
- a rule learning algorithm L
- a hierarchical clustering algorithm H
- a rule selection criterion S

Algorithm:

(1) Separate (randomly or according to a particular scheme) the training examples D into n subsets D_i, $i = 1, \ldots, n$, s.t. $\bigcup D_i = D$;
(2) Learn partial rule models $R_i = \{r_{ij}\}$ for class c from each of these subsets D_i separately, using the learning algorithm L.
(3) Merge the rule sets R_i into one large set R: $R = \bigcup R_i$.
(4) Use the clustering algorithm H to cluster the rules in R into a tree of clusters C_i, $i = 1, \ldots, k$, of similar rules;
(5) For each cluster C_i, compute the *least general generalization* of all the rules in C_i: $\hat{r}_i = lgg(\{r_{ij} \mid r_{ij} \in C_i\})$. The resulting tree T of rules $\hat{r}_i$ represents generalizations of various degrees of the original rules.
(6) From this generalization tree T, select those rules $\hat{r}_i$ that optimize the given selection criterion S.

Fig. 2. The PLCG (**P**artition + **L**earn + **C**luster + **G**eneralize) rule learning strategy.

produce more precise theories with possibly lower coverage, and lowering MP_{RL} will lead to more general theories that also cover a larger number of negative examples.

For *rule clustering*, we use a standard bottom-up hierarchical agglomerative clustering algorithm [12] that produces a binary cluster tree, with the individual rules forming the leaves of the tree, and the root containing all the rules (Fig. 3). The *measure of distance* δ between two rules is simply the number of generalization operations needed to compute the *least general generalization* (*lgg*) of the two rules. Given our standard propositional representation of instances and rules (see Section 4 below for an example), the definition of the *lgg* is quite straightforward: essentially, a discrete attribute with two different values in the two rules generalizes to the empty condition (`true`), while conditions involving numeric attributes (these are of the form $a_i \leqslant x$ or $a_i > x$) generalize to the most specific inequality expression that covers both of the original intervals.

The distance between *clusters* is then computed as the shortest distance from any member of one cluster to any member of the other cluster—that is, we perform what is known as *single-link clustering* [14].

As for the *rule selection criterion* (step 6 of the PLCG algorithm), we currently use another greedy set-covering algorithm that starts with the empty rule set and always adds the rule that has maximum purity on the as yet uncovered instances (Fig. 4). Actually, instead of purity, we use the *Laplace estimate* $L = (p + 1)/(p + n + 2)$, which is related to purity, but penalizes rules with a low coverage on the positive examples. Again, the selection is terminated when no rule with purity (Laplace) greater than some user-

```
function CLUSTER(E)
Clusters := {{e_i} | e_i ∈ E};
while |Clusters| > 1 do
   (c_i, c_j) := MOSTSIMILARCLUSTERPAIR(Clusters);
   Clusters := (Clusters\{c_i, c_j}) ∪{{c_i, c_j}};
return(Clusters);

function MOSTSIMILARCLUSTERPAIR(Cs)
   return (c_i, c_j) s.t. c_i, c_j ∈ Cs, i ≠ j and
   DISTANCE(c_i, c_j) = min{DISTANCE(c_k, c_l) | c_k, c_l ∈ Cs, k ≠ l};

function DISTANCE(Cluster1, Cluster2) =
   min{δ(o_i, o_j) | o_i ∈ Cluster1, o_j ∈ Cluster2};
```

Fig. 3. A standard bottom-up agglomerative (single-link) clustering algorithm; see the main text for a definition of the distance δ between two objects/rules.

```
procedure RULESEL(R,E)
FinalTheory := {};
while not FINALTHEORYSTOPCRITERION(FinalTheory,R,E) do
   Rule := FINDBESTRULE(R,E);
   Covered := COVERS(Rule,E);
   R := R \ {Rule};
   E := E \Covered;
   FinalTheory := FinalTheory ∪ {Rule};
return(FinalTheory);

function FINDBESTRULE(R,E) = argmax_{r∈R} LAPLACE(r,E);

function LAPLACE(r,E) = (p + 1)/(p + n + 2),
   where p (n) = number of positive (negative) examples ∈ E
   covered by rule r;

function FINALTHEORYSTOPCRITERION(R,E) =
   there is no r ∈ R with purity P(r, E) = p/(p + n) ⩾ MP_PLCG
   and positive coverage p ⩾ MC_PLCG
```

Fig. 4. The heuristic procedure employed by PLCG to select the rules for the final theory; MP_{PLCG} and MC_{PLCG} are user-defined parameters.

defined MP_{PLCG} and coverage greater than some minimum coverage MC_{PLCG} can be found.

This is just one of many possible rule selection strategies. Many others are conceivable that could use different criteria for trading coverage against precision, or that might aim at optimizing other aspects of the evolving rule set (e.g., minimum overlap or a minimum number of contradictions between rules).

4. Musical discoveries made by PLCG

Let us first look at some of PLCG's discoveries from a musical perspective. Section 5 below we will then take a more systematic experimental look at PLCG's behaviour relative to more 'direct' rule learning.

4.1. Performance principles discovered

When run on the complete Mozart performance data set (41,116 notes) for each of the six target concepts defined above,[3] PLCG (with parameter settings $MP_{PLCG} = .7$, $MC_{PLCG} = .02$, $MP_{RL} = .9$, $MC_{RL} = .01$) selected a final set of 17 performance rules (from a total of 383 specialized rules) —6 rules for tempo changes, 6 rules for local dynamics, and 5 rules for articulation. (Two rules were selected manually for musical interest, although they did not quite reach the required coverage and precision, respectively.) Some of these rules turn out to be discoveries of significant musicological interest. We lack the space to list all of them here (see [32]). Let us illustrate the types of patterns found by looking at just one of the learned rules:

RULE TL2:
```
abstract_duration_context = equal-longer
& metr_strength ⩽ 1
⇒ ritardando
```

"Given two notes of equal duration followed by a longer note, lengthen the note (i.e., play it more slowly) that precedes the final, longer one, if this note is in a metrically weak position ('metrical strength' $\leqslant 1$)."

This is an extremely simple principle that turns out to be surprisingly general and precise: rule TL2 correctly predicts 1,894 cases of local note lengthening, which is 14.12% of all the instances of significant lengthening observed in the training data. The number of incorrect predictions is 588 (2.86% of all the counterexamples). Together with a second, similar rule relating to the same type of phenomenon, TL2 covers 2,964 of the positive examples of note lengthening in our performance data set, which is more than one fifth (22.11%)! It is highly remarkable that one simple principle like this is sufficient to predict such a large proportion of observed note lengthenings in a complex corpus such as Mozart sonatas. This is a truly novel (and surprising) discovery; none of the existing theories of expressive performance were aware of this simple pattern.

[3] In this experiment, the data were not split into subsets randomly; rather, 10 subsets were created according to global tempo (fast or slow) and time signature (3/4, 4/4, etc.) of the sonata sections the notes belonged to. We chose these two dimensions for splitting because it is known (and has been shown experimentally [28]) that global tempo and time signature strongly affect expressive performance patterns. As a result, we can expect models that tightly fit (overfit?) these data partitions to be quite different, and diversity should be beneficial to an ensemble method like PLCG.

A number of other interesting rules were discovered, such as two pairs of timing and articulation rules that concisely characterize the pianist's consistent treatment of certain types of melodic leaps. One pair of rules is particularly noteworthy in that it points to a rhythmic distinction (between 2:1 and 3:1 duration ratios) that seems to be strongly perceptually relevant: one of the rules predicts a 'softening' of rhythmic contrasts in duration ratios of 2:1, the other a 'sharpening' in 3:1 duration ratios, which seems to indicate that performers indeed use this strategy to aid the listener in distinguishing these two types of rhythms [22]. More details on this as well as the other discovered rules and their relation to theories of expressive performance in the musicological literature can be found in [32].

4.2. Fit and generality of discovered principles

As our primary goal is knowledge discovery, we are first of all interested in how much of the given (training) data is explained by the learned model—in other words, how well the induced models capture the pianist's performance style. Thus, contrary to more 'standard' machine learning applications, the degree of *fit* on the training set is relevant. (Of course, we will also be looking at *generalization accuracy* on unseen data—see below.) As a quantification of fit, we measure the *coverage* (i.e., the number of positive examples correctly predicted) and the *precision* (the proportion of predictions that are correct) of the rule sets on the training data, separately for each prediction category. Table 2 gives the results.

A detailed discussion of the results and their musical interpretation is beyond the scope of this article. Generally, it turns out that certain sub-classes of note lengthening (local *ritardando*), *staccato*, and to a lesser extent the local dynamics variations (*crescendo* and *diminuendo*) are surprisingly predictable, and with extremely few (and simple) rules. On the other hand, categories like *accelerando* and *legato* seem more difficult to predict—at least at the level of individual notes. Uncovering the reasons for this will require more specialized investigations.

To give the reader an impression of just how effective a few simple rules can be in predicting a pianist's behaviour in certain cases, Fig. 5 compares the tempo variations predicted by our rules to the pianist's actual timing in a performance of the well-known Mozart Sonata K.331 in A major (first movement, first section). In fact, it is just two

Table 2
Fit of rule sets on training data (13 Mozart sonatas); True Positives (TP) = correct predictions; False Positives (FP) = incorrect predictions (relative to total number of positive and negative instances, respectively); π = precision = $TP/(TP+FP)$

Category	#rules	True Positives		False Positives		π
ritardando	4	3069/13410	(22.89%)	1234/20551	(6.00%)	.713
accelerando	2	397/13307	(2.98%)	179/20550	(0.87%)	.689
crescendo	3	1318/11629	(11.33%)	591/18260	(3.24%)	.690
diminuendo	3	625/9429	(6.63%)	230/20113	(1.14%)	.731
staccato	4	6916/22132	(31.25%)	1089/18984	(5.74%)	.864
legato	1	687/9256	(7.42%)	592/31860	(1.86%)	.537

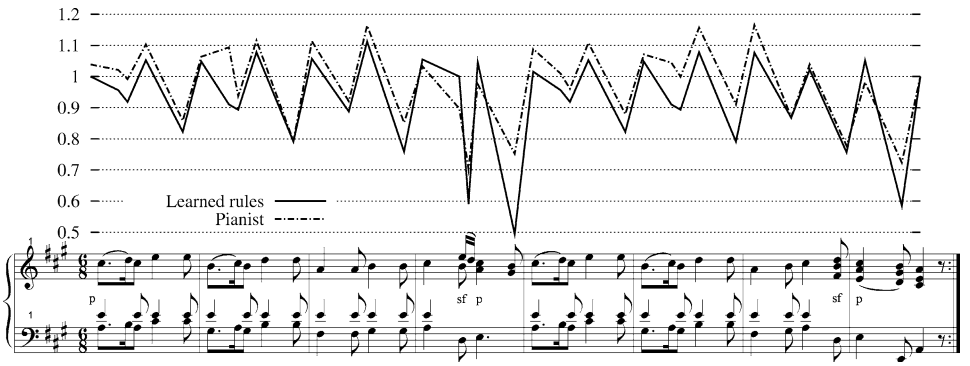

Fig. 5. Mozart Sonata K.331, 1st movement, 1st part, as played by pianist and learner. The curve plots the relative tempo at each note—notes above the 1.0 line are shortened relative to the tempo of the piece, notes below 1.0 are lengthened. A perfectly regular performance with no timing deviations would correspond to a straight line at $y = 1.0$.

simple rules (one for note lengthening (*ritardando*), one for shortening (*accelerando*)) that produce the system's timing curve.[4]

The next question concerns the *generality* of the discovered rules. How well do they transfer to other pieces and other performers? To assess the degree of *performer-specificity* of the rules, they were tested on performances of the same pieces, but by a different artist. The test pieces in this case were the Mozart sonatas K.282, K283 (complete) and K.279, K.280, K.281, K.284, and K.333 (second movements), performed by the renowned conductor and pianist Philippe Entremont, again on a Bösendorfer SE290. The results are given in Table 3.

Comparing this to Table 2, we find no significant degradation in coverage and precision (except in category *diminuendo*). On the contrary, for some categories (*ritardando*, *crescendo*, *staccato*) the coverage is higher than on the original training set. The discriminative power of the rules —the precision—remains roughly at the same level. This (surprising?) result testifies to the generality of the discovered principles; PLCG seems to have successfully avoided overfitting the training data.

Another experiment tested the generality of the discovered rules with respect to *musical style*. They were applied to pieces of a very different style (Romantic piano music), namely, the Etude Op.10, No.3 in E major (first 20 bars) and the Ballade Op.38, F major (first 45 bars) by *Frédéric Chopin*, and the results were compared to performances of these pieces by 22 Viennese pianists. The melodies of these 44 performances amount to 6,088 notes. Table 4 gives the results.

This result is even more surprising. *Diminuendo* and *legato* turn out to be basically unpredictable, and the rules for *crescendo* are rather imprecise. But the results for the other classes are extremely good, better in fact than on the original (Mozart) data which

[4] To be more precise: the rules predict whether a note should be lengthened or shortened; the *precise numeric amount* of lengthening/shortening is predicted by a *k-nearest-neighbor* algorithm (with $k = 3$) that uses only instances for prediction that are covered by the matching rule, as proposed in [26] and [27].

Table 3
Prediction results on test data (Mozart performances by P. Entremont)

Category	#rules	True Positives		False Positives		π
ritardando	4	596/2036	(29.27%)	242/3175	(7.62%)	.711
accelerando	2	90/2193	(4.10%)	45/3013	(1.49%)	.667
crescendo	3	210/1601	(13.12%)	87/3055	(2.85%)	.707
diminuendo	3	53/1598	(3.32%)	45/2725	(1.65%)	.541
staccato	4	861/2192	(39.28%)	228/3996	(5.71%)	.791
legato	1	131/2827	(4.63%)	57/3361	(1.70%)	.697

Table 4
Prediction results on test data (Chopin performances by 22 pianists)

Category	#rules	True Positives		False Positives		π
ritardando	4	1752/2537	(69.06%)	327/2988	(10.94%)	.843
accelerando	2	1472/2767	(53.20%)	110/2746	(4.01%)	.930
crescendo	3	601/2392	(25.13%)	285/2578	(11.06%)	.678
diminuendo	3	0/2249	(0.00%)	0/2784	(0.00%)	–
staccato	4	950/2932	(32.40%)	166/2802	(5.92%)	.851
legato	1	17/2011	(0.85%)	27/3723	(0.73%)	.386

the rules had been learned from! The high coverage values, especially of the tempo rules, are remarkable. Remember also that the data represent a mixture of 22 different pianists. When looking at how well the rules fit individual pianists, we find that some of them are predicted extremely well (e.g., pianist #15: ritardando: $TP = 89/122$ (72.95%), $FP = 4/129$ (3.10%), $\pi = .957$; accelerando: $TP = 71/120$ (59.17%), $FP = 3/132$ (2.27%), $\pi = .959$). We are currently preparing recordings of a larger variety of Chopin pieces, which will permit more extensive investigations into the rules' general validity.

5. Systematic experimental evaluation

The above results show that PLCG can discover general and robust rules in complex data. In this section we take a more systematic look at the general properties of PLCG and its behaviour relative to more 'direct' approaches to rule learning.

5.1. PLCG vs. simple rule learning

The first experiment concerns a systematic comparison between PLCG and its underlying rule learner RL. Does PLCG's ensemble learning strategy produce any advantage over the more direct application of RL on the training data? The experimental setup was as follows: PLCG, with parameter settings $MP_{RL} = .9$ and $MC_{RL} = .01$ for base-level rule learning, and $MP_{PLCG} = .7$ and $MC_{PLCG} = .04$ for heuristic rule selection, was compared to two versions of the base-level learner RL: $\text{RL}_{0.9}$ learns rules directly from the data with a required purity level of $RP_{RL} = 0.9$ (i.e., $\text{RL}_{0.9}$ is exactly the same algorithm as the one used within PLCG's inner loop); $\text{RL}_{0.7}$ uses the more relaxed minimum purity

Table 5
PLCG vs. simple RL: summary of 60 cross-validation results; NR = total number of rules (summed over all 60 data sets × 5 folds) and average number of rules per learning run ($NR/60/5$); TP = true positives; π = precision

	Coverage (TP%)		π	No. of rules		lits/rule
$RL_{0.9}$	12731/62017	(20.53%)	.854	2475	(8.25)	5.27
$RL_{0.7}$	28358/62017	(45.73%)	.708	4094	(13.65)	4.76
$PLCG_{0.7/0.9}$	18767/62017	(30.26%)	.741	1707	(5.69)	3.69

threshold $RP_{RL} = 0.7$, which corresponds to the precision level used by PLCG in its rule selection phase. The purpose of the experiment was to study whether PLCG's multiple rule learning + generalization + selection approach yields any advantage over learning rules directly from the data with the corresponding parameter settings.

60 different experimental data sets were produced by partitioning our 41,116 performed and classified notes according to the general *tempo* (slow vs. fast) and the *time signature* (3/4, 4/4, etc.) of the Mozart sonata segments they belong to. This resulted in 10 training sets each for the 6 target concepts *accelerando*, *ritardando*, *crescendo*, *diminuendo*, *staccato*, and *legato*.

On each of these 60 data sets, the three learning algorithms were compared via a 5-fold (paired) cross-validation. Within each CV run, $RL_{0.9}$ and $RL_{0.7}$ were applied to the combined data from the four training folds, while PLCG used the four folds to learn four separate rule sets (via $RL_{0.9}$) that were then combined, generalized, and selected from. We lack the space to present the full results table with 60 × 18 entries here. Table 5 gives a summary of the results.

The results clearly reflect the expected trade-off: learning with a tighter precision threshold for individual rules ($RL_{0.9}$) yields theories with higher precision, but lower coverage than learning with a lower required precision ($RL_{0.7}$). PLCG, with its mixture of precision thresholds (high in the individual rule learning runs, lower in the rule selection phase) figures somewhere in between: its coverage is higher than $RL_{0.9}$'s and lower than $RL_{0.7}$'s. Conversely, it reaches a precision lower than $RL_{0.9}$'s and higher than $RL_{0.7}$'s.

The interesting result is that PLCG achieves this with significantly *simpler theories* than *either* of the two base-level learners, $RL_{0.9}$ and $RL_{0.7}$. Both the number of rules and their individual complexity (average number of conditions per rule—see last column in Table 5) are significantly lower. PLCG covers more instances than $RL_{0.9}$ with fewer (more general) rules, while still retaining a higher precision than $RL_{0.7}$, which used the same precision threshold MP_{RL} in its search for rules. That is indeed the desired kind of behaviour for our application, where the goal is to discover simple, general, robust (partial) theories that can be presented to and discussed with musicologists.

5.2. *PLCG vs./around* RIPPER

The above results are interesting, but one might object that the rule learner RL to which PLCG was compared is really too simple and cannot compete with state-of-the-art rule learning algorithms. In particular, RL's *pruning strategy* as embodied in its simple stopping

criteria RULESTOPCRITERION and THEORYSTOPCRITERION is very weak, which might account for the large numbers of rules built by $RL_{0.9}$ and $RL_{0.7}$.

To verify this, we also ran RIPPER [4], a state-of-the-art rule learning algorithm with a sophisticated pruning strategy [11], on our 60 data sets. Moreover, as PLCG is a general meta-algorithm, we can just as well wrap PLCG around RIPPER, i.e., use RIPPER as PLCG's base-level learner instead of RL. We did this with two different purity thresholds for final rule selection: $MP_{PLCG} = 0.7$ and $MP_{PLCG} = 0.65$. The corresponding algorithms are called PLCG-$RIP_{0.7}$ and PLCG-$RIP_{0.65}$, respectively. Table 6 summarizes the results. The figures pertaining to RL and PLCG from Table 5 are repeated here for comparison.

A comparison of RIPPER's results with the simpler rule learners $RL_{0.9}$ and $RL_{0.7}$ confirms that RIPPER does indeed perform more aggressive pruning: RIPPER's results are similar to $RL_{0.7}$, with an even higher coverage, but much fewer rules (2182 vs. 4094). On the other hand, RIPPER pays the price of lower precision for its general and simple theories.

$PLCG_{0.9/0.7}$ compares to RIPPER similarly as it does to $RL_{0.7}$: it achieves lower coverage and higher precision, still with fewer rules than RIPPER.

Comparing now the PLCG variants based on RIPPER (PLCG-$RIP_{0.7}$ and PLCG-$RIP_{0.65}$) to 'their' base-level learner RIPPER, we see the same effect as we saw in the PLCG-RL case: the PLCG variants again reduce the coverage of the underlying rule learner (RIPPER), but they improve precision, and now with really extremely simple theories. For instance, with $MP_{PLCG} = .65$, we get a coverage of 32.74% and a precision of .692 with only 642 rules, which is 2.14 rules per run! Moreover, the rules found by PLCG are generally simpler than RIPPER's. In machine learning, the simplicity of rule sets and individual rules is generally associated with the comprehensibility of the induced models. In a knowledge discovery setting as described in this article, where the results must be communicated to and discussed with experts of the domain, comprehensibility and simplicity are important values.

In general, how much precision one is willing to sacrifice for how much coverage and theory simplicity, and vice versa, will depend on the particular application. The important advantage of the PLCG approach is that it makes it easy to explore and control this trade-off via different *rule selection strategies*. This is demonstrated in the following section.

Table 6
PLCG vs. RIPPER: summary of 60 cross-validation results. The results for RIPPER were produced with parameter settings -aUnordered (learn an unordered rule set) and -S1.0 (standard pruning strength); all other parameters were set to their default values

	Coverage (*TP*%)		π	No. of rules		lits/rule
$RL_{0.9}$	12731/62017	(20.53%)	.854	2475	(8.25)	5.27
$RL_{0.7}$	28358/62017	(45.73%)	.708	4094	(13.65)	4.76
RIPPER	**30106/62017**	**(48.54%)**	**.670**	**2181**	**(7.27)**	**3.67**
$PLCG_{0.7/0.9}$	18767/62017	(30.26%)	.741	1707	(5.69)	3.69
PLCG-$RIP_{0.7}$	**14877/62017**	**(23.99%)**	**.757**	**484**	**(1.61)**	**2.15**
PLCG-$RIP_{0.65}$	**20307/62017**	**(32.74%)**	**.692**	**642**	**(2.14)**	**2.26**

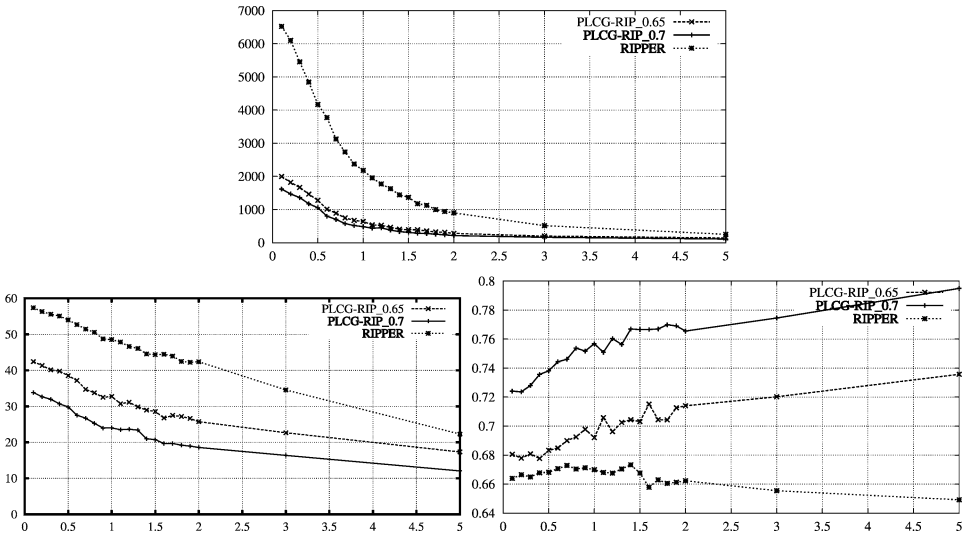

Fig. 6. RIPPER vs. two PLCG-RIP variants (differing in the MP_{PLCG} values) over the range of RIPPER's pruning parameter -S: numbers of rules (top), coverage (bottom left), precision (bottom right).

5.3. *Wrapping PLCG around* RIPPER*: A systematic study*

RIPPER has a parameter (-S) that directly affects the degree of pruning. We ran a systematic comparison of PLCG, wrapped around RIPPER, with RIPPER itself, over the full (sensible) range of this pruning parameter. Fig. 6 plots the results, over the different pruning levels, for RIPPER and two variants of PLCG-RIP, distinguished by different minimum purity levels (MP_{PLCG}) used in the final rule selection phase. Again, each point in these plots is a summary over 60 5-fold cross-validation experiments. The difference in theory complexity is dramatic, but also the difference in coverage and precision. RIPPER achieves a significantly higher coverage for all pruning levels, but also a significantly lower level of precision. In fact, it is remarkable that RIPPER *never*—for any setting of pruning level—reaches a precision as high as *any* of the PLCG runs. PLCG is consistently more precise, but less general.

Additional insight is provided by Fig. 7, which displays the experimental results in two spaces: the graph on the left is a so-called *recall-precision diagram*, which plots the precision of the different rule sets over their coverage on the positive examples ('recall'). What can be clearly seen is that the results of RIPPER and PLCG occupy different regions in 'model space'. The PLCG results exhibit the expected trade-off between coverage and precision: models with higher coverage have lower precision, and vice versa.

The graph in the right half of Fig. 7 is known in machine learning (and other areas) as an *ROC curve*. It plots the ratio of true positives (TP) over the ratio of false positives (FP), for each of the rule sets.[5] *ROC Analysis* [17,18] can be used to study the relative superiority

[5] Note that the true positive rate $TP = Coverage/100$.

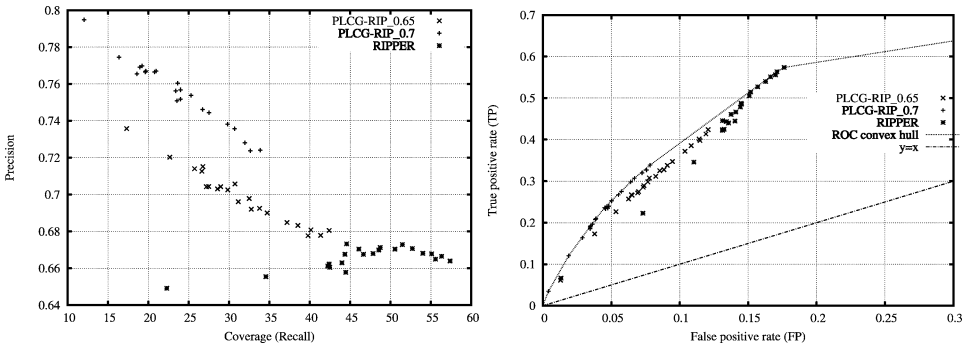

Fig. 7. RIPPER vs. two PLCG-RIP variants in two spaces: precision over recall (left) and ROC space (true positive rate over false positive rate, right). The individual points correspond to different pruning (-S) parameter settings for RIPPER; each point is a summary over 60 5-fold cross-validation runs.

of different classifiers under different class distributions or unequal misclassification costs. Informally, a classifier is better than another if it is to the 'northwest' (higher TP, lower FP, or both). The diagonal $y = x$ corresponds to the strategy of random guessing. The line connecting the 'northwestern-most' classifiers is called the *ROC Convex Hull*. ROC theory tells us that a classifier is optimal for some conditions if and only if it lies on the convex hull (and above the diagonal) [18].

An ROC analysis of the above results essentially shows that in terms of expected misclassification cost, PLCG-RIP$_{0.65}$ is suboptimal under all circumstances (all the points of PLCG-RIP$_{0.65}$ lie below the ROC convex hull). PLCG-RIP$_{0.7}$ and RIPPER both have areas in which they are superior. PLCG-RIP$_{0.7}$ dominates in the lower left part of the space, RIPPER's optimal points are in a space of higher TP (higher coverage) and higher FP. The minimum slope of the hull for which PLCG-RIP$_{0.7}$ dominates is 2.24—that means that if negative examples were 2.24 times as abundant as positive ones, or if the cost of false positive predictions were at least 2.24 times the cost of false negatives, then PLCG-RIP$_{0.7}$ would be the better classifier, otherwise it would be RIPPER. To put it simply, if false positives bother us a lot (at least 2.24 as much as false negatives), PLCG-RIP$_{0.7}$ is the method of choice in this domain, otherwise it is RIPPER.

A quantitative analysis of this type will be useful in applications where misclassification costs can be quantified (and can change), or where unequal class distributions are a problem. In our musical application, the misclassification costs for false positives and negatives cannot really be quantified. Intuitively, a reasonable predictive model of expressive performance should not make too many incorrect predictions, which in our case leads us to favour the more precise theories learned by PLCG.

Fig. 7 demonstrates how the PLCG meta-strategy can be used to explore different regions in the space of possible rule models, simply by employing different criteria for the final selection of rules.[6] The basic rule learning + clustering + generalization steps in

[6] We also experimented with other rule selection criteria, some of which in fact led PLCG to learn rule sets with higher coverage than RIPPER. These results were omitted from the graphs in order to avoid clutter.

PLCG need to be performed only once; the different rule selection procedures then simply operate on the resulting generalization tree.

The point of the above experiments was not to show that either of the two algorithms, RIPPER or PLCG, is better. It really depends on the application whether one is more interested in coverage, precision, or the complexity of a set of rules. PLCG is proposed here as an *alternative* or, better, as a *meta-strategy* that can be wrapped around any given rule discovery algorithm and can be used to explore the space of rule models.

6. Conclusion

To summarize, this article has presented a rule (meta-)learner named PLCG and its application to a complex musical knowledge discovery problem. PLCG has already made substantial and surprising discoveries, which were published in the musicological literature. More systematic experiments have shown that as a general-purpose rule learning and discovery algorithm, PLCG is broadly competitive with the state-of-the-art rule learning algorithm RIPPER, generally discovering theories with higher precision, but lower coverage. But beyond pure accuracy considerations, PLCG possesses a number of properties that make it an attractive algorithm for many knowledge discovery applications: (1) PLCG has a strong tendency to find very simple rule models; (2) by using different rule selection criteria, one can easily explore the trade-off between coverage, precision, and theory complexity; (3) the algorithm can be made to optimize global criteria other than coverage and precision, simply by changing the final rule selection procedure; (4) and as a meta-algorithm it can be used to combine the results of different base-level learning algorithms with possibly different biases.

In terms of related work, PLCG's bottom-up generalization of classification rules is reminiscent of Domingos' RISE algorithm [8], which performs a bottom-up generalization into more and more general rules, starting from the individual training instances. What the two have in common is the idea to learn rules only for those parts of the instance space where the concepts can be easily characterized. RISE caters for the remaining space with instance-based learning, while PLCG simply ignores it (because its focus is on finding comprehensible characterizations of sub-classes of the target). On the other hand, PLCG makes it easy to explore alternative (and arbitrarily complex) strategies for rule combination and selection, which is possible because it constructs an explicit tree of rules of varying generality.

PLCG is also reminiscent of the GROW method proposed in [3], in the context of rule pruning. In GROW, first a rule set is learned that heavily overfits the data, and this rule set is then extended with a large number of variants of the original rules obtained by different degrees of pruning. The rules for the final theory are then selected using a greedy selection procedure similar to PLCG's. The purpose of all this in GROW is to determine the optimal pruning level for the rules. In contrast, PLCG first learns several (possibly quite different) rule sets, from different subsets of the data, and in addition produces a generalization tree over these rules (which is somewhat similar to GROW's extension of a rule set with differently pruned rule variants). That gives it a richer variety of rules to choose from, which should make it less sensitive to wrong conditions possibly chosen by the underlying greedy rule learner.

In that sense, PLCG belongs to the family of so-called *ensemble methods* [5], which learn multiple models from subsets or modified versions of the training data, with one or several learning algorithms, and combine the resulting classifiers in some way. The majority of known ensemble methods like bagging, boosting, stacking, etc. only combine the *predictions* of the classifiers. There are few algorithms that try to combine the resulting multiple *models* into one coherent, comprehensible model. A prime example of this is CMM [9], a meta-learner that combines multiple models into a single theory by applying a learning algorithm to (artificially generated) training examples labeled by the n learned base models. According to the reported experimental results, CMM usually achieves higher accuracy than the base learner (C4.5RULES), but the models produced by CMM are typically 2–6 times more complex than the base learner's. In [9], it is also suggested that the accuracy/complexity tradeoff could be handled via the meta-learner's pruning parameters. Again, we think PLCG's way of explicitly addressing this tradeoff via a selection procedure that can select from a set of alternative rules (of various degrees of generality) is preferable.

Ultimately, we see the main asset of PLCG in the fact that as a meta-learning algorithm, it can be tailored to a wide range of learning and data mining tasks with different characteristics. For instance, it would be easy to modify PLCG's bias so that instead of recall and precision, it attempts to optimize other criteria of interest (e.g., the simplicity of individual rules, non-redundancy of rule sets, etc.). All one needs to do is change the heuristic for final rule selection. Any state-of-the-art rule discovery method can be used as the underlying learner. In fact, we need not limit ourselves to one rule learning algorithm; PLCG could also easily combine the results of different base-level learners, thus combining or selecting from different inductive biases.

Regarding our particular application problem—understanding expressive music performance—the research presented here is but a first small step in a long-term endeavour. There are many open questions and challenging problems [30,31] which provide lots of opportunities for exciting knowledge discovery research.

Acknowledgements

This research is supported by a very generous START Research Prize (project no. Y99-INF) by the Austrian Federal Government, administered by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung* (*FWF*), and by the FWF Project P12645-INF. The Austrian Research Institute for Artificial Intelligence acknowledges basic financial support by the Austrian Federal Ministry for Education, Science, and Culture. We are particularly grateful to the pianists Roland Batik and Philippe Entremont for allowing us to use their performances for our investigations. Thanks to Tom Fawcett for making his ROC/Convex Hull computing tool ROCCH publicly available. And finally, thanks to Johannes Fürnkranz for very helpful comments on an earlier version of this paper.

References

[1] E. Cambouropoulos, From MIDI to traditional musical notation, in: Proc. AAAI-2000 Workshop on Artificial Intelligence and Music, Austin, TX, 2000, pp. 19–23.

[2] E. Cambouropoulos, Automatic pitch spelling: From numbers to sharps and flats, in: Proc. VIII Brazilian Symposium on Computer Music, Fortaleza, Brazil, 2001.
[3] W. Cohen, Efficient pruning methods for separate-and-conquer rule learning systems, in: Proc. IJCAI-93, Chambery, France, 1993, pp. 988–994.
[4] W. Cohen, Fast effective rule induction, in: Proc. 12th International Conference on Machine Learning (ICML-95), Tahoe City, CA, 1995, pp. 115–123.
[5] T.G. Dietterich, Ensemble methods in machine learning, in: J. Kittler, F. Roli (Eds.), First International Workshop on Multiple Classifier Systems, Springer, New York, 2000, pp. 1–15.
[6] S. Dixon, Automatic extraction of tempo and neat from expressive performances, J. New Music Res. 30(1) (2001) 39–58.
[7] S. Dixon, E. Cambouropoulos, Beat tracking with musical knowledge, in: Proc. ECAI-2000 Berlin, Germany, 2000, pp. 626–630.
[8] P. Domingos, Unifying instance-based and rule-based induction, Machine Learning 24 (1996) 141–168.
[9] P. Domingos, Knowledge discovery via multiple models, Intelligent Data Analysis 2 (1998) 187–202.
[10] J. Fürnkranz, Separate-and-conquer rule learning, Artificial Intelligence Rev. 13(1) (1999) 3–54.
[11] J. Fürnkranz, G. Widmer, Incremental reduced error pruning, in: Proc. 11th International Conference on Machine Learning (ML-94), New Brunswick, NJ, 1994, pp. 70–77.
[12] J. Hartigan, Clustering Algorithms, Wiley, Chichester, 1975.
[13] L. Hunter (Ed.), Artificial Intelligence and Molecular Biology, AAAI Press, Menlo Park, CA, 1993.
[14] S.C. Johnson, Hierarchical clustering schemes, Psychometrika 2 (1967) 241–254.
[15] R.D. King, S. Muggleton, R.A. Lewis, M.J.E. Sternberg, Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationship of trimethoprim analogues binding to dihydrofolate reductase, Proc. Nat. Acad. Sci. 89 (1992) 11322–11326.
[16] S. Muggleton, R.D. King, M.J.E. Sternberg, Protein secondary structure prediction using logic-based machine learning, Protein Engineering 5(7) (1992) 647–657.
[17] F.J. Provost, T. Fawcett, Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions, in: Proc. KDD-97, Newport Beach, CA, 1997, pp. 43–48.
[18] F.J. Provost, T. Fawcett, Robust classification for imprecise environments, Machine Learning 42(3) (2001) 203–231.
[19] J.R. Quinlan, Learning logical definitions from relations, Machine Learning 5 (1990) 239–266.
[20] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Francisco, CA, 1993.
[21] J.W. Shavlik, G. Towell, M. Noordewier, Using neural networks to refine biological knowledge, Internat. J. Genome Res. 1(1) (1992) 81–107.
[22] J. Sundberg, How can music be expressive?, Speech Comm. 13 (1993) 239–253.
[23] R.E. Valdés-Pérez, Machine discovery in chemistry: New results, Artificial Intelligence 74 (1) (1995) 191–201.
[24] R.E. Valdés-Pérez, A new theorem in particle physics enabled by machine discovery, Artificial Intelligence 82 (1–2) (1996) 331–339.
[25] R.E. Valdés-Pérez, Principles of human-computer collaboration for knowledge discovery in science, Artificial Intelligence 107 (2) (1999) 335–346.
[26] S. Weiss, N. Indurkhya, Rule-based machine learning methods for functional prediction, J. Artificial Intelligence Res. 3 (1995) 383–403.
[27] G. Widmer, Combining knowledge-based and instance-based learning to exploit qualitative knowledge, Informatica 17 (1993) 371–385.
[28] G. Widmer, Large-scale induction of expressive performance rules: First quantitative results, in: Proc. Internat. Computer Music Conference (ICMC-2000), Berlin, Germany, 2000.
[29] G. Widmer, Using AI and Machine Learning to study expressive music performance: Project survey and first report, AI Comm. 14(3) (2001) 149–162.
[30] G. Widmer, The musical expression project: A challenge for machine learning and knowledge discovery, in: Proc. 12th European Conference on Machine Learning (ECML-01), Freiburg, Germany, 2001.
[31] G. Widmer, In search of the Horowitz Factor: Interim report on a musical discovery project, in: Proc. 5th Internat. Conference on Discovery Science (DS-02), Lübeck, Germany, 2002.
[32] G. Widmer, Machine discoveries: A few simple, robust local expression principles, J. New Music Res. 31 (1) (2002) 37–50.