

# Discovering Simple Rules in Complex Data: A Meta-learning Algorithm and Some Surprising Musical Discoveries

Gerhard Widmer

Department of Medical Cybernetics and Artificial Intelligence, University of Vienna,  
and

Austrian Research Institute for Artificial Intelligence, Vienna  
gerhard@ai.univie.ac.at

## 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).

The article will show 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.

## 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 rather unusual (at least in data mining) domain of music.

The research described here is part of a large, long-term inter-disciplinary research project situated at the intersection of the scientific disciplines of Musicology and AI [19]<sup>1</sup> The goal is to use intelligent data analysis methods to study the complex phenomenon of *expressive music performance*. We want to understand what great musicians do when they interpret

---

<sup>1</sup>See also the project web page at  
<http://www.ai.univie.ac.at/oefai/ml/music/musicproject.html>.

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 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 work described here represents a first major step towards this goal.

The contribution of this article is two-fold: it presents a new learning algorithm called PLCG for inducing simple, comprehensible partial rule models (i.e., models that characterize those parts of a phenomenon that can be readily characterized), and it describes some interesting and surprising discoveries made by this algorithm in the domain of empirical music research. But the contribution goes beyond this particular application; PLCG is a domain-independent, general-purpose method that could be of use in many other applications in machine learning and data mining.

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 theories (sets of classification rules) in complex data where neither high coverage nor high precision can be expected. 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 principles and 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 rule models.

## **2 The Motivating Application: 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. 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.

The research described here is part of a large, long-term basic research project that aims at developing quantitative models of expressive performance (e.g., in the form of prediction rules for tempo, dynamics, and articulation) with machine learning and data mining methods [19]. 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 learning and 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 categori-

	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

Table 1: Numbers of training examples of various categories.

cal 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 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 [18], we succeeded in showing that even at this low level, there is structure in the data; learning algorithms like C4.5 [15] 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 note lengthening 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 [14] or RIPPER [4]. (see Figure 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

---

```

procedure RL( $E$ )
   $Theory := \{\}$ ;
  while not THEORYSTOPCRITERION( $Theory, E$ ) do
     $Rule :=$  FIndBESTRULE( $E$ );
     $E := E \setminus$  COVERS( $Rule, E$ );
     $Theory := Theory \cup Rule$ ;
  return( $Theory$ );

procedure FIndBESTRULE( $E$ )
   $Rule := \{\}$ ;
  repeat
     $Rule := Rule \cup$  FIndBESTCONDITION( $E$ );
     $E :=$  COVERS( $Rule, E$ );
  until RULESTOPCRITERION( $Rule, E$ );
  return( $Rule$ );

```

---

Figure 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).

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 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 figure 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 Figure 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

---

**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  $D_i$ ,  $i = 1 \dots n$ , 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  $C_i$ ,  $i = 1 \dots k$ , of similar rules;
5. For each cluster  $C_i$ , compute the *least general generalization* of all the rules in  $C_i$ :  $\hat{r}_i = \text{lgg}(\{r_{ij} | r_{ij} \in C_i\})$ . 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  $S$ .

---

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

RL sketched in Figure 1, with the standard information gain heuristic as used in FOIL [14] and the following stopping criteria:

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

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.<sup>2</sup> Obviously, high values of  $MP_{RL}$  will 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.

---

<sup>2</sup>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).

---

```

function CLUSTER( $E$ )
   $Clusters := \{\{e_i\} | e_i \in E\}$ ;
  while  $|Clusters| > 1$  do
     $(c_i, c_j) := \text{MOSTSIMILARCLUSTERPAIR}(Clusters)$ ;
     $Clusters := (Clusters \setminus \{c_i, c_j\}) \cup \{\{c_i, c_j\}\}$ ;
  return  $(Clusters)$ ;

function MOSTSIMILARCLUSTERPAIR( $Cs$ )
  return  $(c_i, c_j)$  s.t.  $c_i, c_j \in Cs, i \neq j$  and
   $\text{DISTANCE}(c_i, c_j) = \min\{\text{DISTANCE}(c_k, c_l) | c_k, c_l \in Cs, k \neq l\}$ ;

function DISTANCE( $Cluster1, Cluster2$ ) =
   $\min\{\delta(o_i, o_j) | o_i \in Cluster1, o_j \in Cluster2\}$ ;

```

---

Figure 3: A standard bottom-up agglomerative (single-link) clustering algorithm (see the main text for a definition of the distance  $\delta$  between two objects/rules).

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 (Figure 3). The *measure of distance*  $\delta$  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 nominal (symbolic) 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  $a_i \leq x$  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* [13].

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 (Figure 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-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).

---

```

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

function FINDBESTRULE( $R, E$ ) =  $argmax_{r \in R}$  LAPLACE( $r, E$ );

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

function FINALTHEORYSTOPCRITERION( $R, E$ ) =
  there is no  $r \in R$  with purity  $P(r, E) = p/(p + n) \geq MP_{PLCG}$ 
  and positive coverage  $p \geq MC_{PLCG}$ 

```

---

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

## 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,<sup>3</sup> 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 [21]). Let us illustrate the types of patterns found by looking at just one of the learned rules:

---

<sup>3</sup>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 proved experimentally [18]) 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.

**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’  $\leq 1$ ).”

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.

A few other interesting rules were discovered, such as two pairs of timing and articulation rules that nicely characterize the pianist’s consistent treatment of certain types of melodic leaps and rhythmic patterns. These discoveries and their relation to other theories of expressive performance in the musicological literature are discussed in [21].

## 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.

Category	#rules	True Positives	False Positives	$\pi$
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

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);  $\pi$  = precision =  $TP/(TP + FP)$ .

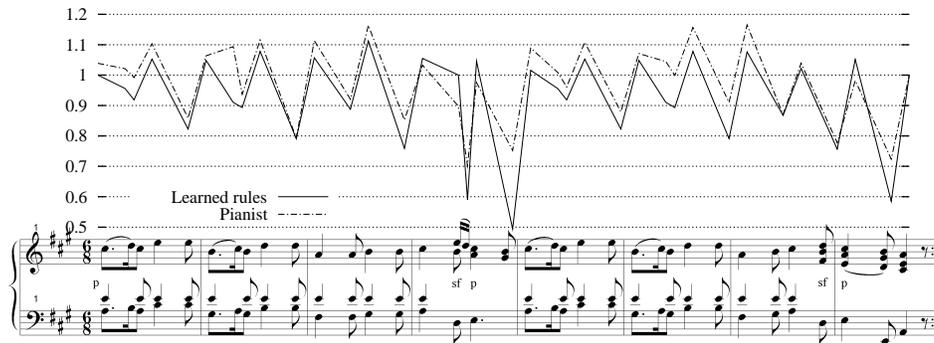


Figure 5: Mozart Sonata K.331, 1st movement, 1st part, as played by pianist and learner. The curve plots the relative tempo at each note — notes above the 1.0 line are shortened relative to the tempo of the piece, notes below 1.0 are lengthened. A perfectly regular performance with no timing deviations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

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 well 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, Figure 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 simple rules (one for note lengthening (*ritardando*), one for shortening (*accelerando*)) that produce the system’s timing curve.<sup>4</sup>

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, we tested them 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 (and the merits of the PLCG rule discovery method).

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,

<sup>4</sup>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 [16] and [17].

Category	#rules	True Positives	False Positives	$\pi$
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 3: Prediction results on test data (Mozart performances by P.Entremont).

Category	#rules	True Positives	False Positives	$\pi$
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

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

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 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

	Coverage ( $TP\%$ )		$\pi$	No. of rules	
RL <sub>0.9</sub>	12731/62017	(20.53 %)	.854	2475	(8.25)
RL <sub>0.7</sub>	28358/62017	(45.73 %)	.708	4094	(13.65)
PLCG <sub>0.7/0.9</sub>	18767/62017	(30.26 %)	.741	1707	(5.69)

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

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: RL<sub>0.9</sub> learns rules directly from the data with a required purity level of  $RP_{RL} = 0.9$  (i.e., RL<sub>0.9</sub> is exactly the same algorithm as the one used within PLCG’s inner loop); RL<sub>0.7</sub> uses the more relaxed minimum purity 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<sub>0.9</sub> and RL<sub>0.7</sub> 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<sub>0.9</sub>) that were then combined, generalized, and selected from. We lack the space to present the full results table with  $60 \times 18$  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<sub>0.9</sub>) yields theories with higher precision, but lower coverage than learning with a lower required precision (RL<sub>0.7</sub>). 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<sub>0.9</sub>’s and lower than RL<sub>0.7</sub>’s. Conversely, it reaches a precision lower than RL<sub>0.9</sub>’s and higher than RL<sub>0.7</sub>’s.

The interesting result is that PLCG achieves this with significantly *fewer rules* than *either* of the two base-level learners, RL<sub>0.9</sub> and RL<sub>0.7</sub>. In other words, PLCG covers more instances than RL<sub>0.9</sub> with fewer (more general) rules, while still retaining a higher precision than RL<sub>0.7</sub>, 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.

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

	Coverage ( $TP\%$ )		$\pi$	No. of rules	
RL <sub>0.9</sub>	12731/62017	(20.53 %)	.854	2475	(8.25)
RL <sub>0.7</sub>	28358/62017	(45.73 %)	.708	4094	(13.65)
<b>RIPPER</b>	<b>30106/62017</b>	<b>(48.54 %)</b>	<b>.670</b>	<b>2181</b>	<b>(7.27)</b>
PLCG <sub>0.7/0.9</sub>	18767/62017	(30.26 %)	.741	1707	(5.69)
<b>PLCG-RIP<sub>0.7</sub></b>	<b>14877/62017</b>	<b>(23.99 %)</b>	<b>.757</b>	<b>484</b>	<b>(1.61)</b>
<b>PLCG-RIP<sub>0.65</sub></b>	<b>20307/62017</b>	<b>(32.74 %)</b>	<b>.692</b>	<b>642</b>	<b>(2.14)</b>

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.

via different *rule selection strategies*. In fact, one can perform the rule learning and clustering steps once and then apply any number of different rule selection algorithms on the resulting rule cluster tree. This will be demonstrated in the following section.

## 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<sub>0.9</sub> and RL<sub>0.7</sub>.

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<sub>0.7</sub> and PLCG-RIP<sub>0.65</sub>, respectively. They were also run on our 60 test problems. Table 6 summarizes the results and also includes once more the figures pertaining to RL and PLCG from Table 5.

A comparison of RIPPER’s results with the simpler rule learners RL<sub>0.9</sub> and RL<sub>0.7</sub> confirms that RIPPER does indeed perform more aggressive pruning: RIPPER’s results are similar to RL<sub>0.7</sub>, 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<sub>0.9/0.7</sub> compares to RIPPER similarly as it does to RL<sub>0.7</sub>: it achieves lower coverage and higher precision, still with fewer rules than RIPPER.

Comparing now the PLCG variants based on RIPPER (PLCG-RIP<sub>0.7</sub> and PLCG-RIP<sub>0.65</sub>) 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 627 rules, which is 2.14 rules per run! It is quite remarkable to see how much PLCG manages to simplify theories.

To repeat, the point of this experiment is not to show that either of these two algorithms, RIPPER or PLCG, is better. It really depends on the application whether one is more inter-

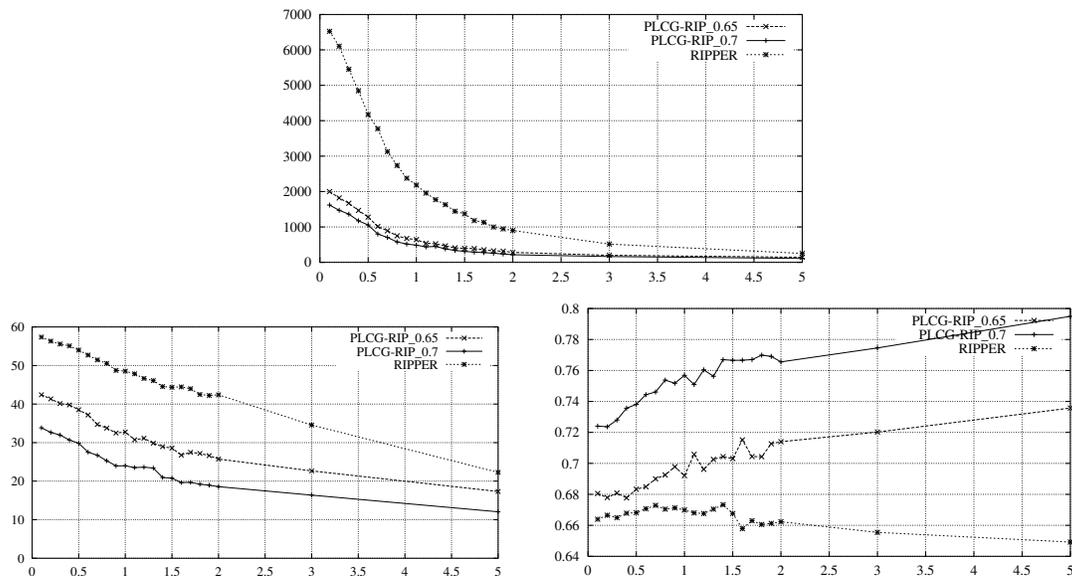


Figure 6: RIPPER vs. two PLCG-RIP variants (differing in the  $MP_{PLCG}$  values) over the range of RIPPER’s pruning parameter  $-S$ : numbers of rules (top), coverage (bottom left), precision (bottom right).

ested in the coverage, the 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 easily explore lots of different parts of the space of rule models. We will briefly illustrate that with a set of systematic experiments involving PLCG and RIPPER.

### 5.3 Wrapping PLCG Around RIPPER: A systematic Study

RIPPER has a parameter ( $-S$ ) that directly affects the degree of pruning. We performed a systematic comparison of PLCG, wrapped around RIPPER, with RIPPER itself, over the full (sensible) range of this pruning parameter. Figure 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 no setting of pruning level — reaches a precision as high as *any* of the PLCG runs. PLCG is consistently more precise, but less general.

Another interesting view on the results is given by what is known in information retrieval as a *recall-precision diagram*: Figure 7 plots the precision of the different rule sets over their coverage (‘recall’). What can be clearly seen here 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.

Figure 7 demonstrates quite clearly how the PLCG meta-strategy can be used to explore different regions in the space of possible rule models, simply by employing different criteria

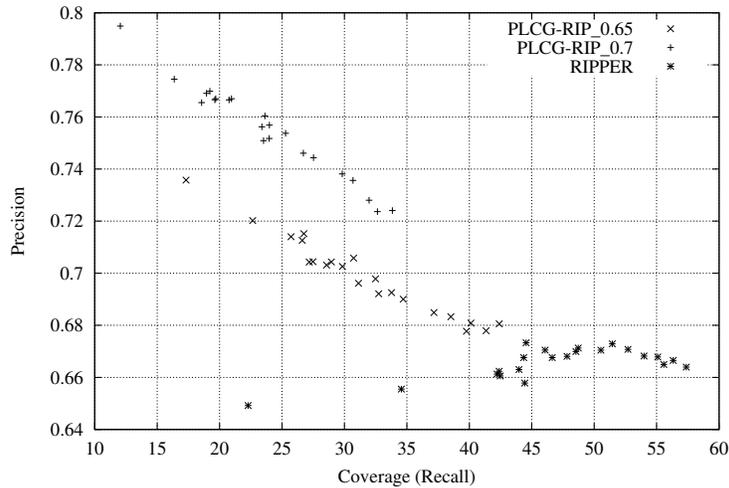


Figure 7: RIPPER vs. two PLCG-RIP variants: recall (coverage) vs. precision. The individual points correspond to different pruning (-S) parameter settings for RIPPER; each point is a summary over 60 5-fold cross-validation runs.

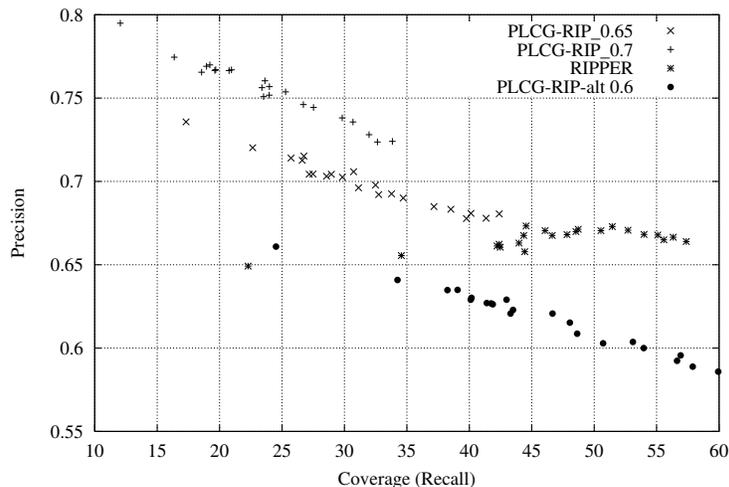


Figure 8: RIPPER vs. three PLCG-RIP variants: recall (coverage) vs. precision. The individual points correspond to different pruning (-S) parameter settings for RIPPER; each point is a summary over 60 5-fold cross-validation runs.

for the final selection of rules. The basic rule learning + clustering + generalization steps in PLCG need to be performed only once; the different rule selection procedures then simply operate on the resulting generalization tree.

All the PLCG results in Figure 7 have higher precision, but lower coverage than the rule sets found by the underlying learning algorithm RIPPER. Just to demonstrate that the opposite behaviour (theories with *higher coverage* than the underlying rule learner) can also be achieved, Figure 8 includes results for PLCG parametrized with a slightly different rule selection heuristic: PLCG-RIP-alt selects *all* maximally general rules from the generalization tree that exceed a given minimum precision, regardless of their coverage (i.e.,  $MC_{PLCG} = 0$ ). The price to be paid in this case is lower precision. That seems to indicate a kind of overfitting

— too many specialized rules from the generalization tree are collected in the final theory. There may be other rule selection heuristics that trade off coverage and precision in a better way.

## 6 Conclusion

To summarize, this article has presented a rule (meta-)learner named PLCG that has the capacity to learn simple partial theories from complex data and offers a natural mechanism for exploring the coverage/precision/complexity tradeoff. It has also been demonstrated that PLCG is able to make interesting and surprising discoveries in a complex real-world domain.

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. Another significant difference is PLCG’s use of multiple models to arrive at a more stable theory.

PLCG is also reminiscent of the GROW method proposed in [3], in the context of rule set 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. While the majority of known ensemble methods like bagging, boosting, stacking, etc. only combine the *predictions* of the classifiers, there are a 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 [20] 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. Many thanks also to the L. Bösendorfer Company, Vienna, and in particular Fritz Lachnit for generous help with the Bösendorfer SE290. And finally, thanks to Johannes Fürnkranz for very helpful comments on an earlier version of this paper.

## References

- [1] Cambouropoulos, E. (2000). From MIDI to Traditional Musical Notation. In *Proceedings of the AAAI'2000 Workshop on Artificial Intelligence and Music*, 17th National Conference on Artificial Intelligence (AAAI'2000), Austin, TX. Menlo Park, CA: AAAI Press.
- [2] Cambouropoulos, E. (2001). Automatic Pitch Spelling: From Numbers to Sharps and Flats. In *Proceedings of the VIII Brazilian Symposium on Computer Music*, Fortaleza, Brazil.
- [3] Cohen, W. (1993). Efficient Pruning Methods for Separate-and-Conquer Rule Learning Systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, Chambéry, France.
- [4] Cohen, W. (1995). Fast Effective Rule Induction. In *Proceedings of the 12th International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann.
- [5] Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. In J. Kittler and F. Roli (Ed.), *First International Workshop on Multiple Classifier Systems*. New York: Springer Verlag.
- [6] Dixon, S. (2001). Automatic Extraction of Tempo and Beat from Expressive Performances. *Journal of New Music Research* (in press).

- [7] Dixon, S. and Cambouropoulos, E. (2000). Beat Tracking with Musical Knowledge. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000)*, IOS Press, Amsterdam.
- [8] Domingos, P. (1996). Unifying Instance-Based and Rule-Based Induction. *Machine Learning* 24, 141–168.
- [9] Domingos, P. (1998). Knowledge Discovery via Multiple Models. *Intelligent Data Analysis* 2, 187–202.
- [10] Fürnkranz, J. (1999). Separate-and-Conquer Rule Learning. *Artificial Intelligence Review* 13(1), 3–54.
- [11] Fürnkranz, J. and Widmer, G. (1994). Incremental Reduced Error Pruning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*. San Francisco, CA: Morgan Kaufmann.
- [12] Hartigan, J. (1975). *Clustering Algorithms*. Chichester, UK: John Wiley & Sons.
- [13] Johnson, S.C. (1967). Hierarchical Clustering Schemes. *Psychometrika* 2, 241–254.
- [14] Quinlan, J.R. (1990). Learning Logical Definitions from Relations. *Machine Learning* 5, 239–266.
- [15] Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. San Francisco, CA: Morgan Kaufmann.
- [16] Weiss, S. and Indurkha, N. (1995). Rule-based Machine Learning Methods for Functional Prediction. *Journal of Artificial Intelligence Research* 3, 383–403.
- [17] Widmer, G. (1993). Combining Knowledge-based and Instance-based Learning to Exploit Qualitative Knowledge. *Informatika* 17, 371–385.
- [18] Widmer, G. (2000). Large-scale Induction of Expressive Performance Rules: First Quantitative Results. In *Proceedings of the International Computer Music Conference (ICMC'2000)*. San Francisco, CA: International Computer Music Association.
- [19] Widmer, G. (2001a). Using AI and Machine Learning to Study Expressive Music Performance: Project Survey and First Report. *AI Communications* 14(3), 149–162.
- [20] Widmer, G. (2001b). The Musical Expression Project: A Challenge for Machine Learning and Knowledge Discovery. In *Proceedings of the 12th European Conference on Machine Learning (ECML'01)*, Freiburg, Germany. Berlin: Springer Verlag.
- [21] Widmer, G. (2001c). *Machine Discoveries: A Few Simple, Robust Local Expression Principles*. Submitted.