Plausible Explanations and Instance–Based Learning in Mixed Symbolic/Numeric Domains

Gerhard Widmer

Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, and
Austrian Research Institute for Artificial Intelligence,
Schottengasse 3, A–1010 Vienna, Austria
e–mail: gerhard@ai.univie.ac.at

Abstract

The paper is concerned with supervised learning of numeric target concepts. The task is to learn to predict or determine the exact values of some numeric target variables. Training examples may be described by both symbolic and numeric predicates. General domain knowledge may be available in qualitative form. The paper presents a general learning model for such domains. The model integrates a symbolic learning component, which is based on a multi–instance plausible explanation algorithm, and an instance–based learning component, which stores instances with precise values and predicts new values by interpolation. The symbolic component can use available qualitative background knowledge; it learns sub–concepts that partition the space for the underlying instance–based method. A realization of the model in a system named IBL–Smart is then described. The system has been applied to a complex task from the domain of tonal music, and some experimental results are reported that demonstrate the effectiveness of the method.

Key words: Knowledge–based learning, instance–based learning, integrated learning, qualitative models.

1 Introduction

It is being recognized by more and more researchers that qualitative background knowledge is naturally available in many domains, and that learning algorithms are needed that can effectively use such knowledge, even if it is incomplete and inconsistent, and generally abstract and imprecise. Some approaches to this problem have been proposed in the recent past, most of them centering around the notion of incomplete or plausible explanations (see, e.g., Tecuci, 1991; Tecuci & Michalski, 1991; Widmer, 1991). All these methods and systems assume that the target concepts are discrete classes of objects, to be described by classification rules which assign a new object to its appropriate class.

However, there are also many learning problems with numeric target concepts, i.e., where the task is to predict more or less precisely the values of some numeric variables. The training instances may be described by both symbolic and numeric predicates. In such domains, too, general domain knowledge may be available

that relates certain parameters, but maybe only in a qualitative, imprecise way. As an example, consider typical prediction tasks such as stock market prediction or the prediction of energy consumption or demand in some power plant. One can easily conceive of partial qualitative models of these domains that would capture some of the relevant domain knowledge. Intelligent learners should be able to utilize such abstract knowledge.

With few exceptions (e.g., regression trees – Breiman et al., 1984), 'classical' symbolic learning methods cannot be used for such numeric problems (unless the domain of the numeric target concept can be abstracted into discrete, qualitative subranges without loss of relevant information). In particular, plausible explanation methods capable of utilizing qualitative domain knowledge, like those mentioned above, are not applicable to such domains; they assume discrete target concepts, and it is not clear how the qualitative background knowledge should be related to the precise numeric information in the data.

The topic of this paper is a new learning model (and an implemented system) that can learn numeric target concepts while taking maximum advantage of available qualitative domain knowledge, given that the problem and the target concepts satisfy some basic assumptions. The approach consists essentially in using a *symbolic learner* to partition the space for an *instance–based, numeric method* that is used to predict precise values of the target variables. The symbolic learner produces *plausible explanations* for discrete subconcepts; the explanations (and the extracted rules) are based both on qualitative background knowledge and on empirical information from the training data. The underlying instance–based method stores the examples in several *independent instance spaces* and uses the learned symbolic rules to decide which instance space is relevant to a given example. Values for new examples are then predicted by numeric interpolation in those instance spaces that are classified as relevant by the associated symbolic rules.

The motivation for this research was a practical and complex problem in the domain of tonal music, namely, learning to apply *expressive interpretation* to a given piece of music, i.e., to dynamically vary tempo and dynamics in order to produce a musically satisfying performance. The target concepts in this problem are necessarily numeric (exactly how much variation should be applied to a given note), and there is some natural domain knowledge that is relevant to the task. The domain knowledge comes from music theory and is inherently qualitative and incomplete, but describable in explicit form.

The model has been implemented in a learning system named IBL–Smart (for reasons that will become obvious soon). We will first present the general learning model, then describe its realization in the system IBL–Smart, and illustrate its applicability with a description of our particular musical application and some experimental results. Our approach was strongly inspired by ideas presented in (DeJong, 1989), and the last section will relate our system to that and other work.

2 The General Model

2.1 Statement of the learning problem

This paper deals with supervised learning of numeric target concepts. More precisely, the class of learning problems we are interested in can be defined as follows:

Given: a set E of training examples, described in terms of a set of operational predicates P, where we distinguish symbolic predicates PS and numeric predicates (attributes) PN. Thus, $PS \cup PN = P$. Also attached to each example $e \in E$ is a numeric attribute $T(e,v)$ (the *target attribute*) with known value v. (This replaces the *classification* in symbolic supervised concept learning.) As v is a function of (the description of) the instance, we will also write $v = T(e)$. Note that so far, there are no negative instances in this scenario.

Find: a set of general rules that predict, for any given object o described by predicates $\in P$, a numeric value $v = T(o)$, based on the description of o.
(As in symbolic concept learning, we might require these rules to be *complete* (predict a value for every example) and *correct* (predict the correct value for each example) with respect to the training data (cf. Michalski, 1983). However, this may not be 100% desirable or feasible in every application domain.)

In addition, there may be some domain–specific *background knowledge (BK)* relating the target concept $T(X,V)$ (or some abstractions of T – see below) to some of the operational predicates P in specific ways, possibly via some intermediate non–operational predicates. This knowledge might be in the form of rules, as in standard EBL domain theories (Mitchell et al., 1986) or in the form of qualitative knowledge items as in (Widmer, 1993). The knowledge need not be correct or complete, nor need it be quantitative and precise. An additional constraint then is to find solutions (rules) that conform as closely as possible to BK while also consistently describing the training data E.

The learning model we are going to introduce in the next section includes a symbolic learning component that can utilize qualitative background knowledge for generating plausible explanations. For this method to be applicable, we need to make the following

Assumptions:

1) We assume that there are some discrete, qualitative sub–concepts $Ti(X)$ of the target concept $T(X,V)$ that can naturally be distinguished, where a sub–concept is defined by a more or less clearly distinguished subrange of the function value V.
2) We further assume that it is these discrete sub–concepts that are related to operational predicates P by the available background knowledge BK.
3) Finally, we assume that examples of the discrete sub–concepts Ti can be distinguished using the operational predicates P.

For example, in our energy demand prediction task, such subconcepts might be extremely_low(Demand) or higher_than_capacity(Demand); in the musical domain described below, there are natural qualitative subconcepts such as crescendo(Note) and diminuendo(Note) (increase or decrease, respectively, in loudness relative to the current level) or accelerando(Note) and ritardando(Note) (increase or decrease in tempo). [1)]

The motivation for this assumption is that these discrete, qualitative, symbolic sub–concepts will be the target concepts for the symbolic

---

[1)] The boundaries between these subconcepts will sometimes have to be defined somewhat arbitrarily. This is not necessarily a problem, as the results of the symbolic learning component are not used for classification, but only to find appropriate sets of instances for comparison. Section 2.2 will make that clearer.

learning component. Each of the original training instances will be assigned to one of the sub–concepts T_i, depending on its value $v = T(e)$, and the symbolic learner will learn general rules for each sub–concept. Note that in this way we also introduce negative instances for each target concept T_i, namely, all examples assigned to some T_j where $j \neq i$.

2.2 The learning model

Returning to our general learning problem, one way to approach it would be to simply do instance–based learning in the entire description space spanned by all the available attributes P, symbolic and numeric. That is, training instances would be stored along with their complete descriptions, and the value $v = T(o)$ for some new object o would be predicted by some nearest neighbor method in the space of stored instances, possibly with some numeric interpolation. There are several problems with this approach. First, it is not always clear how to devise a similarity metric that combines symbolic and numeric attributes in a meaningful way, especially when the attributes are of various types and inhomogeneous with respect to domain size etc. Second, the only way to integrate available qualitative background knowledge into the learning process is via the similarity metrics. This may not be the most natural way to express one's domain knowledge. Moreover, instance–based approaches suffer from the problem that they do not produce comprehensible concept descriptions. On the other hand, it is clear that some kind of instance–based interpolation component is needed in such domains, as the task is to predict numeric values from continuous domains, which is impossible with discrete, symbolic concept descriptions.

The model were are proposing here consists of two components: a *symbolic learning component* that learns to distinguish different types of situations and can utilize all the available domain knowledge, and an *instance–based* component which stores the instances with their precise numeric attribute values and can predict the target value for some new object by numeric interpolation over known instances. The connection between these two components is as follows: each rule (conjunctive hypothesis) learned by the symbolic learning component describes a subset of the instances; these are assumed to represent one particular subtype of the concept to be learned. All the instances covered by a rule are given to the instance–based learner to be stored together in a separate instance space. Predicting the target value for some new object then involves matching the object against the symbolic rules and using only those numeric instance spaces (interpolation tables) for prediction whose associated rules are satisfied by the object. In this way, the system learns several distinct instance spaces where different laws and regularities may apply. In fact, different instance spaces may contain examples with conflicting values.

More precisely, the target concepts for the *symbolic learning component* are the discrete, qualitative sub–concepts T_i mentioned in section 2.1. The symbolic learner learns general conditions that characterize or discriminate between these discrete classes. These conditions may refer to both symbolic and numeric predicates. The symbolic learner tries to use all the available qualitative background knowledge. The result produced by this component is a set of general rules that group the examples into clusters by assigning them to different sub–classes of the target concept.

The *numeric, instance–based component* takes the original training instances *E* as clustered by the symbolic learner, and creates a separate instance space from each cluster. Instances are stored with all their numeric attributes and with their precise numeric target values. The dimensions of such an instance space are thus defined by the numeric attributes *PN*. For some new object *o*, the target value $v = T(o)$ can then be predicted by selecting the appropriate instance space (by using the generated symbolic rules as filters), and applying some numeric interpolation method over the stored instances.

Several comments seem to be in order here: First, we assume that the symbolic learning component may refer to predicates from both *PS* and *PN* for its hypotheses. It is not realistic (nor necessary) to expect that the discrete sub–concepts T_i can always be distinguished by reference to symbolic predicates only. Second, we do assume that after clustering the examples according to sub–concepts (and sub–sub–concepts, if these are disjunctive), interpolation over the *numeric* attributes in the resulting instance spaces is sufficient to predict sensible target values. In other words, we assume that the rules learned for the sub–concepts T_i contain all the relevant *symbolic* information. The dimensions of the instance spaces are only attributes from *PN*. Any other solution would require some non–standard interpolation scheme to arrive at numeric prediction values. If additional domain knowledge about attribute relevance is available, the number of numeric dimensions may still be reduced, or some specialized similarity measures may be used for interpolation.

3 Realization of the Model: IBL–Smart

The general method has been implemented in a system named IBL–Smart and has been tested in the context of a complex musical problem. In accordance with the model, IBL–Smart consists of two components. The first of these — the symbolic learner — has been specifically designed to be able to use qualitative domain knowledge.

3.1 The symbolic learning component

The symbolic learner in IBL–Smart is a multiple–instance plausible explanation system based on the search algorithm of ML–Smart (Bergadano & Giordana, 1988). It performs top–down discrimination, integrating and interleaving deductive and inductive operationalization steps. The basics of the search are described below (section 3.1.1). For IBL–Smart, we have extended ML–Smart's discrimination algorithm to also use qualitative background knowledge in the form of *general dependency statements* and *directed qualitative dependency relations*. This is described in section 3.1.2.

3.1.1 The basic search algorithm

The learner starts with a nonoperational definition of the target concept (some discrete sub–concept T_i) and performs stepwise operationalization (specialization) by growing a heuristic best–first search tree. Each node/partial hypothesis in the search tree is accompanied by its extension, i.e., the positive and negative examples covered by the operational part of the expres-

sion. This makes it possible to use coverage measures as part of the search heuristic.

As in ML–Smart, each step in the search is either

(1) a *deductive application of a rule* from the domain theory — replacing a non–operational literal by its sufficient conditions as defined by the rule;

(2) an *inductive generalization step* — dropping a predicate when the node covers too few positive instances and thus the hypothesis seems too restricted; or

(3) an *inductive specialization step* — adding some predicate to the operational part of the hypothesis in order to exclude some negative instances.

Deductive operationalization steps (1) are preferred. Inductive specialization (3) is done when deductive operationalization is not possible (e.g., when no rule is available or applicable to the examples). Inductive generalization (2) is attempted whenever the current node covers too few positive instances (according to some threshold) and thus the hypothesis seems too restrictive. The system then looks for a condition that, if dropped, would increase the number of instances covered by the hypothesis. All in all, the search algorithm integrates deduction and induction in a fine–grained manner.

The search is guided by a heuristic measure (*evaluation function*) H that measures the relative 'goodness' of nodes. The heuristic decides both which node is to be expanded next, and how. Among other things (see below), it takes into account the coverage of the expression, i.e., the ratio positive / negative instances covered by the node, and also the absolute number of positive instances covered.

The discrimination algorithm has been extended to utilize also numeric attributes in discrimination steps. For numeric attributes, the system looks for a binary split point that best discriminates between positive and negative instances, as it is done in some decision tree learners (e.g., Cestnik et al., 1987; Fayyad & Irani, 1992). The general evaluation function H of the search algorithm is used to determine what is the best split.

3.1.2 Using qualitative background knowledge

The search algorithm as described above corresponds closely to the original ML–Smart method as presented in (Bergadano & Giordana, 1988). In our system IBL–Smart, the algorithm has been extended so as to also utilize *qualitative* background knowledge, where available. IBL–Smart domain theories may contain two types of qualitative knowledge items:

(1) *General dependency statements* of the form depends_on(Q,Ps) simply state that some predicate Q may be operationalized by using a set of specified predicates Ps. This type of general knowledge items has already been proposed in (Bergadano, Giordana and Ponsero, 1989). In IBL–Smart, such statements tell the search algorithm to use an entire set of predicates in one operationalization step: successors of a node are created for *all possible combinations* of values for the predicates Ps occurring in some positive instances covered by the node.

Such dependency statements are similar, but not identical, to *determinations* (Russell, 1987). They permit IBL–Smart to perform strictly constrained forms of look–ahead, and thus help overcome blindness

effects that would arise if the algorithm performed purely empirical step–wise specialization. For instance, they can be used to describe *relational clichés* as proposed in (Silverstein & Pazzani, 1991).

(2) *Directed dependency statements* of the form q+(A,B) can be paraphrased as "the values of A and B are positively proportionally related" or "high (or low) values of A tend to produce high (or low) values of B, all other things being equal". Negative dependency (q−(A,B)) is defined analogously. Such statements are, of course, restricted to functional predicates (or *attributes*) that assign values to objects. They were already used in (Widmer, 1991) and are similar to Michalski's *M–descriptors* (Michalski, 1983). The notation was borrowed from Forbus' *qualitative proportionalities* (Forbus, 1984).

In the search algorithm of IBL–Smart, directed dependencies are used like general dependencies (create successors for all possible value combinations), and the additional knowledge about the *direction* of influence is used in the *search heuristic H*: when evaluating some operationalization based on a q+ or q− relation, the heuristic also rates the degree to which the particular values involved match the direction of the dependency (which is assumed to be linear). Knowing that q+(A,B) and operationalizing condition B(.)= b, an operationalization B(.)= b because A(.) = a for specific values b and a will be regarded the more plausible the more the relative positions of b and a in their respective domains agree: B(.)= high because A(.) = high is rated as more plausible than B(.)= high because A(.) = low (see also Widmer, 1993.). Hypotheses constructed by IBL–Smart will tend to include those attributes that most closely approximate such linear constraints between the data and the background knowledge.

Note that, as with strict deductive rules, such qualitative dependency statements need not necessarily be entirely correct in order to have a positive impact on the search. If a dependency statement is correct, it will lead to faster convergence; if it is too general (the given predicates are not sufficient to completely discriminate between positive and negative instances), subsequent empirical discrimination steps will refine it. And if it is overly restrictive (some predicates are not necessary), this may be repaired by empirical generalization steps, where the predicates that are too restrictive are removed from the hypothesis to arrive at a more general partial concept description.

By taking into account both such inference–dependent plausibility measures and information about the numbers of positive and negative instances covered by a node, the search heuristic combines weak, imprecise background knowledge with empirical information from the training data, producing hypotheses that tend to correspond to the background knowledge as much as the data permit and overriding the background knowledge if the data are in conflict with the knowledge.

3.2 The numeric instance–based component

The result of this learning step is a concept hypothesis for a discrete, qualitative sub–concept in the form of a DNF expression, where each conjunct describes one particular subtype of the sub–concept. The instance–based learner now collects all the training instances covered by a particular conjunct and builds an instance store

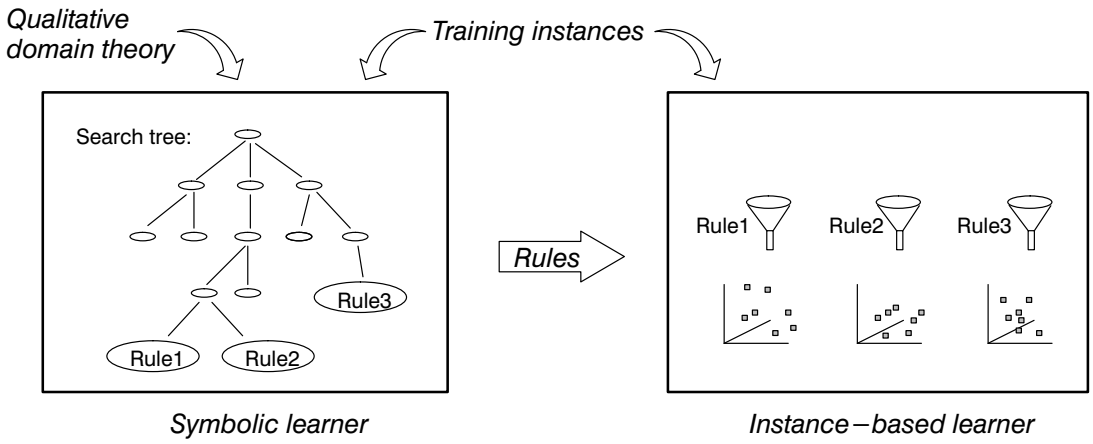

Fig. 1: Sketch of IBL–Smart

in the form of an *interpolation table*, using these examples. In the absence of knowledge about the relevance of the numeric attributes (*PN)* to the target value, the *dimensions* of the interpolation table are chosen to be all the numeric attributes ($\subseteq PN$) shared by the selected instances (not all instances may have defined values for all attributes), and the output dimension is the value of the target variable $V = T(X)$.

When given a new instance for which to predict the value of the target variable, the system matches the instance against all learned rules, retrieves those instance spaces whose associated rules are matched, and computes a value for the instance's target value by interpolation in each of the retrieved spaces. If the instance matches more than one rule, and thus target values are computed in several spaces, the target values are simply averaged. Lacking more specific knowledge about the relationships between the various numeric parameters, we use the Euclidean distance as the similarity measure and perform linear interpolation. Figure 1 summarizes the basic structure of IBL–Smart.

4 An Application of IBL–Smart: Learning Expressive Interpretation

IBL–Smart has been applied to a complex problem from the domain of tonal music, namely, expressive performance or interpretation of written music. By this we understand the variations in tempo and loudness that a performer applies (consciously or unconsciously) to the notes of a piece during performance. When played exactly as written, most pieces of tonal music would sound utterly mechanical and lifeless.

There are basically three dimensions to expressive performance: variations in tempo (*"rubato"*), in loudness (*"dynamics"*) and in the duration of notes as actually played, as opposed to the notated length (*"articulation"*).

In this presentation, we will restrict ourselves to the dimension of *dynamics*. As mentioned in the introduction, this concept is inherently numeric, as the task is to decide not just whether or not to play some note louder or softer, but exactly by *how much.* Nevertheless, there are two discrete, qualitative sub–concepts that can naturally be distinguished: crescendo(Note) and diminuen-

do(Note) – whether a note is to be played louder or softer, respectively, than some standard level. These are the target concepts for the plausible explanation component. The precise amounts by which the loudness is to be varied are numeric multiplication factors that are to be learned by the instance–based component. [2)]

Training instances are derived from actual performances of piano pieces recorded on an electronic piano via a MIDI interface. At the moment, we are restricting ourselves to single line melodies (with additional information about the underlying harmonic structure of the piece). That is, the input is a sequence of notes, described in terms of various predicates and accompanied by explicit information about the degree of crescendo or diminuendo that was applied to it by the performer. Each note of a played piece is a training instance.

The *description language* consists of predicates that describe various features of a note and structural features of its surroundings. There are currently 41 operational predicates, of which 21 are symbolic (like followed_by_rest(Note)) and 20 are numeric (like duration(Note,X)). Some of these predicates are computed by a pre–processing component which performs a music–theoretic analysis of the given piece in terms of some relevant musical structures (e.g., phrases and various types of 'processes' such as linear melodic lines (ascending or descending), rhythmic patterns, etc.). Many numeric attributes then describe the relative position of a note in a phrase or in a 'process'. Note that the number of attributes defined for a given note varies: some notes occur in many patterns, others only in some. So not all numeric attributes are defined for every note.

The background knowledge for this problem is mainly in the form of directed and undirected dependency statements. The domain theory is a hierarchy of such dependency statements and some crisp rules. The top level of the theory relates the phenomenon of loudness variations to some abstract musical notions by a set of dependencies like

```
depends_on( crescendo(Note,X),
                        [ salience(Note,Y)]).
depends_on( crescendo(Note,X),
                [ goal_directedness(Note,Y)]).
depends_on( crescendo(Note,X),
                        [ closure(Note,Y)]).
```

The first of these can be paraphrased as

"Whether crescendo should be applied to a note (and if so, the exact amount X) depends, among other things, on the structural importance (salience) Y of the note."

and analogously for the other ones.

The abstract notions salience, goal_directedness, and closure are then again related to lower–level musical effects, all the way down to some surface features of training instances, for example:

[2)] A clarifying remark to readers who feel that we are trivializing the artistic phenomenon of expressive musical performance by claiming that a computer program can easily learn to replicate such behaviour, or that these phenomena can be explained by some simple domain theory: We are not talking here about the highly artistic details in variation that distinguish a great pianist or other performer. We are convinced, however (and there is much support for this hypothesis from various areas of musicology), that expressive performance does have a large 'rational' component, in that one of its purposes is to convey an understanding of musical structure to a listener. It is this rational part for which we can find partial plausible explanations and which we can expect a computer to learn, provided it is equipped with the necessary musical knowledge and a suitable vocabulary.

Fig. 2: Beginnings of three little minuets by J.S.Bach

```
q+( metrical_strength(Note,X),
    stability(Note,Y))                and
q+( harmonic_stability(Note,X),
    stability(Note,Y))
```

"The degree of stability Y of a note is positively proportionally related (among other things) to the metrical strength X of the note" etc.

where metrical_strength is a numeric and harmonic_stability is a symbolic attribute (with a discrete, ordered domain of qualitative values). Both are defined as operational.

Given this domain theory and some played pieces, the plausible explanation component learns mixed symbolic/numeric rules that discriminate various types of situations where a crescendo or a diminuendo occurs. These rules are sets (disjunctions) of conjunctive conditions; each conjunct describes a particular class of crescendo/diminuendo situations. For each conjunct, a numeric interpolation table (instance space) is created which contains all the instances covered by the conjunct. The set of all numeric attributes shared by all the instances covered by a conjunct defines the dimensions of the respective interpolation table.

5 An Experiment

Several experiments with comparatively simple piano pieces have been performed. In one experiment, we chose three well–known minuets from J.S.Bach's *Notenbüchlein für Anna Magdalena Bach* as training and test pieces. The beginnings of the three minuets are shown in Figure 2. All three pieces consist of two parts. The second part of each piece was used for training: they were played on an electronic piano by the author, and recorded through a MIDI interface. After learning, the system was tested on the first parts of the same pieces. In this way, we combined some variation in the training data (three different pieces) with some uniformity in style (three pieces from the same period and with similar characteristics; test data from the same pieces as training data, though different).

The training input consisted in 212 examples (notes), of which 79 were examples of crescen-

do, and 120 were examples of diminuendo (the rest were played in a neutral way). The system learned 14 rules (conjuncts) and, correspondingly, 14 interpolation tables characterizing crescendo situations, and 15 rules for diminuendo. Quite a number of instances were covered by more than one rule. For illustration, here is a simple rule for crescendo:

```
crescendo(Note,X) :–
        metrical_strength(Note,S),
        S > 4.0,
        harmony_stability(Note,high),
        previous_interval(Note,I1),
        direction(I1,up),
        next_interval(Note,I2),
        direction(I2,down).
```

"Apply some crescendo to the current note if the metrical strength of the note is > 4 and the underlying harmony is stable and the direction of the melody from the previous to the current note is up and the direction of the melody from the current note to the next is down"

The quality of the learning results is not easy to measure, as there is no precise criterion to decide whether some performance is right or wrong. Judging the correctness is a matter of listening. Unfortunately, we cannot attach a recording to this paper so that the reader can appreciate the results. Instead, Figure 3 depicts a part of one of the training pieces (the second part of the first minuet in G major as played by the author), and also shows the performance created by the system for a test piece (the first part of the same minuet) after learning. The figures plot the relative loudness with which the individual notes were played. A level of 1.0 would be neutral, values above 1.0 represent crescendo (increased loudness), values below 1.0 diminuendo.

The reader familiar with standard music notation may appreciate that there are strong similarities in the way similar types of phrases are played by the human teacher and the learner. (Note, for instance, the crescendo in lines rising by stepwise motion, and the decrescendo patterns in measures with three quarter notes). Generally, the results were very good, given the limited amount of training data and the surface differences between training and test pieces. Readers not familiar with music notation will have to take our word for it. We are planning experiments with other, non–musical domains where the results will be more easily interpretable and testable.

In a comparative experiment, we also tested a system restricted to learning only in an instance–based way, that is, with interpolation tables, but without the symbolic explanation component. This learner used *all* the available attributes, both numeric and symbolic. The following distance metric was used: all numerical attributes were scaled between 0 and 1, and for symbolic attributes, the distance was defined to be 0 in the case of a match and 1 otherwise. As not all training instances share all numeric dimensions, the system learned as many interpolation tables as there were combinations of numeric attributes occurring in the training data (18 for crescendo, 12 for decrescendo). The results on the same data were considerably worse. The learner did not distinguish as well between different types of situations, and the results are rather blurred, as can also be seen from Figure 4, which shows the same test piece as played by the second system.

Fig. 3: Parts of a training piece as played by teacher (top) and test piece as played by learner after learning (bottom)

Fig. 4: Part of test piece as played after instance–based learning only

6 Discussion, Related Work, and Related Matters

First, let us briefly recapitulate the main characteristics of the learning model: (1) The model can learn precise numeric concepts via an instance–based method while using available qualitative background knowledge through a symbolic learning component. (2) The symbolic learning component defines and separates different independent regions in numeric instance space where different regularities may apply. This allows the instance–based learner to build specialized instance stores, which may yield very specific prediction behaviour. And (3), as a side effect, learning rules for discrete sub–concepts clusters the examples around meaningful abstractions, which may be useful for other tasks.

The definition of abstract sub–concepts *Ti* introduces a natural distinction between symbolic and numeric learning, and also produces negative instances for the symbolic learner. That the background knowledge is used only by the symbolic component seems natural, given that it is qualitative and thus may explain abstract, symbolic concepts (at best), but certainly not precise numeric values and relationships. [3] Of course, this does not preclude the use of additional knowledge to guide or constrain numeric learning in the instance–based component.

[3] For instance, in music, we may be able to explain why a performer applied some crescendo at a certain point (for instance, in order to stress a musically important event), but we can never explain why she chose exactly that *precise degree* of crescendo. A system can only record these precise degrees and try to replicate the same behaviour in similar situations. What is similar is determined by the rules learned by the symbolic component.

It should be remembered that this is a general learning model: the system presented here — IBL–Smart — is just one particular incarnation of a more general approach. We have found it convenient to use a best–first search algorithm like the ML–Smart learner as the basis for our plausible explanation component, as it explicitly constructs a search tree and allows us to integrate various sources of knowledge into the learning process via the search heuristic (evaluation function). However, with appropriate modifications and extensions, other symbolic learners capable of utilizing incomplete and inconsistent knowledge — for instance, FOCL (Pazzani & Kibler, 1992) — might be used just as well in this framework.

Similarly, more elaborate strategies could be used in the instance–based component. (Aha et al., 1991) have described a number of instance–based learning methods that could be applied within a framework such as ours. Also, available domain knowledge about the relative degree of relevance of numeric attributes or about the domains and typical values of numeric variables could be used to devise more sophisticated similarity metrics, tailored to the particular application.

With respect to related work, we acknowledge the important influence on this project by some of the ideas expressed in (DeJong, 1989). DeJong had presented a system that combined a very weak notion of plausible inference over single cases with numeric variables. Our approach departs from his, among other things, in the variety of types of background knowledge and in the use of a heuristically guided, search–based, multi–instance explanation algorithm that allows much more control over the learning process. Not only does this search introduce a strong notion of *empirical plausibility* by taking

into account the distribution of instances; the use of an explicit search heuristic also makes it possible to exploit the qualitative knowledge contained in *qualitative dependencies* (q+, q−) to compute the relative plausibility of arguments. The best–first search is very likely to find explanations that are most plausible overall (both with respect to the knowledge and the data). DeJong's system, on the other hand, simply assumed that the syntactically simplest explanation was also the most plausible one.

As an additional advantage of this multi–instance explanation approach, we note also that there is a natural way to deal with certain types of *noise* in the training data. The evaluation function of the search algorithm incorporates two thresholds: it accepts only nodes (conjuncts) that cover some minimum number of positive instances, and the termination criterion allows the search to halt when a certain percentage (< 100 %) of positive instances are covered. Thus, the system can ignore rare instances that look like exceptions, but are really the result of noise. By varying these thresholds, the system can be tuned to the characteristics of different application domains.

In fact, the musical experiments described in the previous section were characterized very strongly by noise in the data, originating from the author's imperfect piano technique, from the imprecise boundaries between the abstract sub–concepts crescendo and diminuendo, and from imprecision inherent in the domain itself (there are simply no 100% laws as to how some passage must and will be played; variation will invariably happen). The system concentrated on learning typical variations.

Of course (and this is also implied by the name), our system also owes a lot to the work on integrated deductive–inductive learning in ML–Smart (Bergadano & Giordana, 1988). We have extended the ML–Smart algorithm to also utilize background knowledge in the form of directed dependency statements (which are a very natural kind of knowledge in many domains).

With respect to the system described in (Widmer, 1991; 1993), which also constructs plausible explanations of individual training instances on the basis of qualitative background knowledge, we note that explaining multiple instances at a time adds a strong empirical justification to plausible explanations. The price is non–incrementality. However, it is likely that, using techniques described in (Widmer, 1989), IBL–Smart can be made to learn incrementally without losing too much in effectiveness.

Acknowledgments

I would like to thank Johannes Fürnkranz for helpful comments on this paper. Thanks also to the anonymous reviewers for very precise and stimulating comments. This research is sponsored in part by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant P8756–TEC. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry for Science and Research.

References

Aha, D.W., Kibler, D., and Albert, M.K. (1991). Instance–Based Learning Algorithms. *Machine Learning 6(1)*, pp. 37–66.

Bergadano, F. and Giordana, A. (1988). A Knowledge Intensive Approach to Concept Induction. In *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, pp. 305–317.

Bergadano, F., Giordana, A., and Ponsero, S. (1989). Deduction in Top–Down Inductive Learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, N.Y., pp. 23–25.

Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth.

Cestnik, B., Kononenko, I., and Bratko, I. (1987). ASSISTANT 86: A Knowledge Elicitation Tools for Sophisticated Users. In I.Bratko & N.Lavrac (Eds.), *Progress in Machine Learning*. Wilmslow, U.K.: Sigma Press.

Collins, A. and Michalski. R.S. (1989). The Logic of Plausible Reasoning: A Core Theory. *Cognitive Science 13(1)*, pp. 1–49.

DeJong, G. (1989). Explanation–Based Learning with Plausible Inferencing. In *Proceedings of the Fourth European Working Session on Learning (EWSL–89)*, Montpellier, France, pp. 1–10.

Fayyad, U. and Irani, K. (1992). On the Handling of Continuous–Valued Attributes in Decision Tree Generation. *Machine Learning 8(1)*, pp. 87–102.

Forbus, K.D. (1984). Qualitative Process Theory. *Artificial Intelligence 24(1–3)*, pp. 85–169.

Michalski, R.S. (1983). A Theory and Methodology of Inductive Learning. In R.S.Michalski, J.G.Carbonell, and T.M.Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach, vol. I*. Palo Alto, CA: Tioga.

Mitchell, T.M., Keller, R.M., and Kedar–Cabelli, S.T. (1986). Explanation–Based Generalization: A Unifying View. *Machine Learning 1(1)*, pp. 47–80.

Pazzani, M. and Kibler, D. (1992). The Utility of Knowledge in Inductive Learning. *Machine Learning 9(1)*, pp. 57–94.

Russell, S.J. (1987). *Analogical and Inductive Reasoning*. Ph.D. thesis, Report STAN–CS–87–1150, Stanford University, Stanford, CA.

Silverstein, G. and Pazzani, M.J. (1991). Relational Clichés: Constraining Constructive Induction During Relational Learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Ill., pp. 203–207.

Tecuci, G.D. (1991). Learning as Understanding the External World. In *Proceedings of the First International Workshop on Multistrategy Learning*, Harper's Ferry, W.VA, pp. 49–64.

Tecuci, G.D. and Michalski, R.S. (1991). A Method for Multistrategy Task–adaptive Learning Based on Plausible Justifications. In *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Ill., pp. 549–553.

Widmer, G. (1989). An Incremental Version of Bergadano & Giordana's Integrated Learning Strategy. In *Proceedings of the Fourth European Working Session on Learning (EWSL–89)*, Montpellier, France, pp. 227–238.

Widmer, G. (1991). Learning by Plausible Reasoning and its Application to a Complex Musical Problem. In *Proceedings of the First International Workshop on Multistrategy Learning (MSL–91)*, Harper's Ferry, W.VA.

Widmer, G. (1993). Learning with a Qualitative Domain Theory by Means of Plausible Explanations. In R.S.Michalski and G.Tecuci, eds., *Machine Learning: A Multistrategy Approach, vol. IV.* Los Altos, CA: Morgan Kaufmann. (in press)