ILP with Noise and Fixed Example Size: A Bayesian Approach

Eric McCreath[*] and **Arun Sharma**[†]
Department of Artificial Intelligence
School of Computer Science and Engineering
The University of New South whales
Sydney NSW 2052, Australia
Email : {ericm,arun}@cse.unsw.edu.au

Abstract

Current inductive logic programming systems are limited in their handling of noise, as they employ a greedy covering approach to constructing the hypothesis one clause at a time. This approach also causes difficulty in learning recursive predicates. Additionally, many current systems have an implicit expectation that the cardinality of the positive and negative examples reflect the "proportion" of the concept to the instance space.

A framework for learning from noisy data and fixed example size is presented. A Bayesian heuristic for finding the most probable hypothesis in this general framework is derived. This approach evaluates a hypothesis as a whole rather than one clause at a time. The heuristic, which has nice theoretical properties, is incorporated in an ILP system, LIME. Experimental results show that LIME handles noise better than FOIL and PROGOL. It is able to learn recursive definitions from noisy data on which other systems do not perform well. LIME is also capable of learning from only positive data and also from only negative data.

1 Introduction

Most ILP systems like GOLEM [Muggleton and Feng, 1990] and FOIL [Quinlan, 1990] employ a greedy covering heuristic to build hypotheses. They try to find the clause that covers maximum number of positive examples without covering any or few negative examples. A new set of positive examples is created by removing the covered positive examples and the process is repeated with this new set of positive examples and the negative examples until there are no uncovered positive examples left. While this approach has lead to efficient learning in many applications, there are situations in which it fails to perform well.

Consider the problem of noise handling. GOLEM [Muggleton and Feng, 1990] has a rudimentary noise handling facility as each clause is allowed to cover a fixed number of negative examples in addition to as many positive examples as possible. This approach is inflexible as it requires adjusting the noise parameter with changes in sample size and noise level. Inflexibility aside, handling of noise at the clause level, in many cases, appears to result in a poor overall hypothesis, mainly because of overfitting. FOIL [Quinlan, 1990] employs an MDL/MML [Rissanen, 1978; Wallace and Freeman, 1987] like approach to noise handling. While this is better than an *ad hoc* approach, its greedy covering strategy of building the hypothesis one clause at a time can lead to overfitting in the presence of noise (for example, see Figure 2 that describes FOIL's performance on a simple predicate `plus2`).[1] A single clause only yields information about overgeneralization errors, information about data not covered by the hypothesis requires looking at the complete hypothesis.

Another problem with current ILP systems arises while learning predicates that require recursive definitions. In these situations, for successful learning to take place, a system has to be provided with a complete initial sequence of the data, including the base case. However, if the data is sparse, it becomes very difficult for these systems to learn hypotheses with recursive clauses. Again, an approach that evaluates complete hypotheses instead of individual clauses does a better job of overcoming such deficiencies in the data.

Apart from the above mentioned difficulties, many applications of MDL/MML like heuristic have an implicit expectation that the distribution of examples received by

[*]Supported by an Australian Postgraduate Award.
[†]Supported by Australian Research Council grant A49530274.

[1]mFOIL [Lavrac *et al.*, 1996] has an improved noise handling capability, but it still suffers from the greedy covering approach. Noise handling in LINUS [Lavrac and Dzeroski, 1992] is more about taking advantage of noise handling techniques from attribute value learning.

the learner matches the true "proportion" of the underlying concept to the instance space. However, in many learning situations this assumption is unjustified. Usually the cardinality of positive and negative examples is fixed and independent of the concept being learned. As an example, consider a learner presented with a set of 100 positive and 100 negative examples of cancer patients. It is very unlikely that this set of examples is representative of the population from where the examples are drawn. Moreover, many systems require a minimum number of positive and negative examples.

Motivated by the above problems with existing ILP systems, the present paper introduces a general framework for noise handling in learning systems, and derives a Bayesian heuristic for inducing a hypothesis with the maximum posterior probability. The framework assumes that the number of examples is fixed independent of the concept (and certainly not representative of the actual proportion of the concept to the instance space). Hence, the heuristic can be employed to learn from only positive data and from only negative data in addition to the usual combination of positive and negative data. An interesting theoretical result about the framework is that it formalizes the intuitive importance of an extra positive example over an extra negative example for concepts that are "small" with respect to the instance space. A similar result for the importance of negative examples over positive examples holds for concepts that are "large" with respect to the instance space. Empirical evidence in keeping with these theoretical results are presented.

The heuristic is adapted for ILP and implemented in the LIME system that considers complete candidate hypotheses rather than single clauses. We would like to note that another system that looks at complete hypotheses instead of single clauses is TRACY [?]. A theoretical bound is derived that yields a bound on the search space for the most probable hypothesis. It is shown that LIME equipped with this heuristic handles noise better than FOIL and PROGOL. Empirical results also show the effectiveness of this heuristic in handling recursive definitions from sparse data. Additional results are presented to show that this heuristic is capable of learning from only positive data and from only negative data.

2 Noise model and the Bayesian Heuristic

We describe a framework of learning for modeling noise and fixed example size. Within this framework, we derive a Bayesian heuristic for the optimal hypothesis.

Let X denote a countable class of instances. Let D_X be a distribution on the instance space X. Let $\mathcal{C} \subseteq 2^X$ be a countable concept class. Let $D_{\mathcal{C}}$ represent the distribution on $\mathcal{C}$. Let H be a hypothesis space and P be the distribution (prior) over H. The concept represented by a hypothesis h is referred to as the extension of h (written: ext(h)). Further, let $\mathcal{C}$ and H be such that:

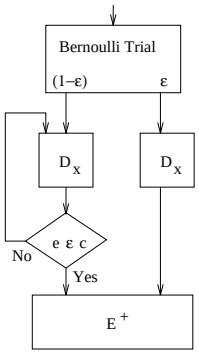

Figure 1: Model of positive example generation

- for each $C \in \mathcal{C}$, there is an $h \in H$ such that $C = \text{ext}(h)$; and
- for each $C \in \mathcal{C}$, $D_{\mathcal{C}}(C) = \sum_{\{\, h \in \mathcal{H} | C = \text{ext}(h) \,\}} P(h)$.

Let $\theta(C)$ denote the "proportion" of the concept C with respect to the instance space X, that is, $\theta(C) = \sum_{x \in C} D_X(x)$.

We assume that a concept C is chosen with the distribution $D_{\mathcal{C}}$. Let $\epsilon \in [0, 1]$ be the level of noise. Suppose we want to generate m positive examples and n negative examples (the reader should note that in the fixed noise model, m and n are independent of the concept C).

Now, each of the m positive examples are generated as follows: With probability ϵ, a instance is randomly choose from X and made a positive example (this could possibly introduce noise). With probability $1 - \epsilon$, an instance is repeatedly selected randomly from X until the instance is an element of the concept. This instance is the positive example generated. Figure 1 illustrates this process. The generation of negative examples is done similarly[2].

The fixed example size aspect of the framework makes it more flexible than other frameworks that have an implicit expectation that the proportion of positive and negative examples reflects the concept. The reader should note that this allows learning to take place from only positive data and from only negative data in addition to the usual combination of positive and negative data.

We now derive a Bayesian heuristic for finding the most probable hypothesis h given the example set E.[3]

[2]The level of noise ϵ can be made different for the positive and negative examples, but for simplicity we take it to be the same.

[3]All references to example sets are actually references to *example multisets*.

This induction can be formally expressed as follows.[4]

$$h_{\text{induced}} = \max_{h \in H} P(h|E) \quad (1)$$

Using Bayes' formula, $P(h|E)$ can be expressed as follows.

$$P(h|E) = \frac{P(h)P(E|h)}{P(E)} \quad (2)$$

We will apply Occam's razor in computation of $P(h)$, the prior probability of the hypothesis h, thereby assigning higher probabilities to simpler hypotheses. $P(E|h)$, the probability of the examples E given that hypothesis h represents the target concept, can be calculated by taking the product of the conditional probabilities of the positive and negative example sets. As each positive example is generated independently, $P(E^+|h)$ may be calculated by taking the product of the conditional probabilities of each positive example. $P^+(e|h)$, the conditional probability of a positive example e given hypothesis h, is computed as follows.

$$P^+(e|h) = \begin{cases} \frac{D_X(e)(1-\epsilon)}{\theta(\text{ext}(h))} + D_X(e)\epsilon, & \text{if } e \in \text{ext}(h); \\ D_X(e)\epsilon, & \text{if } e \notin \text{ext}(h). \end{cases} \quad (3)$$

A few words about the above equation are in order. Given that h represents the target concept, the only way in which $e \notin \text{ext}(h)$ is if the right hand path in Figure 1 was chosen. Hence, in this case the conditional probability of e given h is $D_X(e)\epsilon$. On the other hand, if $e \in \text{ext}(h)$ then either the left or right hand paths in Figure 1 could have been chosen. The contribution of the right hand path to $P^+(e|h)$ is then $D_X(e)\epsilon$. If the left hand path is taken, then the instance drawn is guaranteed to be from the target concept; hence $D_X(e)(1-\epsilon)$ is divided by $\theta(\text{ext}(h))$—the proportion of the target concept to the instance space. By a similar reasoning we compute $P^-(e|h)$, the conditional probability of a negative example e given hypothesis h.

$$P^-(e|h) = \begin{cases} \frac{D_X(e)(1-\epsilon)}{1-\theta(\text{ext}(h))} + D_X(e)\epsilon, & \text{if } e \notin \text{ext}(h); \\ D_X(e)\epsilon, & \text{if } e \in \text{ext}(h). \end{cases} \quad (4)$$

Now, $P(E|h)$ can be computed as follows.

$$P(E|h) = \prod_{e \in E^+} P^+(e|h) \prod_{e \in E^-} P^-(e|h) \quad (5)$$

We let TP denote the set of true positives $\{ e \in E^+ \mid e \in \text{ext}(h) \}$; TN denote the set of true negatives $\{ e \in E^- \mid e \notin \text{ext}(h) \}$; FPN denote the set of false positives and false negatives, $\{ e \in E^+ \mid e \notin \text{ext}(h) \} \cup \{ e \in E^- \mid e \in \text{ext}(h) \}$.

[4]The notation $\max_{h \in H} P(h|E)$ denotes a hypothesis $h \in H$ such that $(\forall h' \in H)[P(h|E) \geq P(h'|E)]$.

Substituting 3 and 4 into 5 and using TP, TN, and FPN, we get the following.

$$P(E|h) = \left(\prod_{e \in E^+ \cup E^-} D_X(e) \right) \left(\frac{1-\epsilon}{\theta(\text{ext}(h))} + \epsilon \right)^{|\text{TP}|} \left(\frac{1-\epsilon}{1-\theta(\text{ext}(h))} + \epsilon \right)^{|\text{TN}|} \epsilon^{|\text{FPN}|} \quad (6)$$

Now substituting 6 into 2 and 2 into 1 and performing additional arithmetic manipulation, we obtain the final h_{induced} as $\max_{h \in H} Q(h)$, where $Q(h)$ is defined as follows.

$$Q(h) = \lg(P(h)) + |\text{TP}| \lg \left(\frac{1-\epsilon}{\theta(\text{ext}(h))} + \epsilon \right) + |\text{TN}| \lg \left(\frac{1-\epsilon}{1-\theta(ext(h))} + \epsilon \right) + |\text{FPN}| \lg(\epsilon) \quad (7)$$

Hence, in our inductive framework, a learning system attempts to maximize $Q(h)$ (referred to as the *quality* of the hypothesis h). A number of theoretical results can be shown about this heuristic. For example, the following theorem formalizes the intuitive expectation that if the proportion of a concept to the instance space is small, then positive examples are more useful than negative ones.

Theorem 1 *Suppose c is a target concept and h_c is a hypothesis that represents c. Let $E = (E^+, E^-)$ be the given sets of positive and negative examples. Now, from E construct two new example sets $E_1 = (E^+ \cup \{ e_1 \}, E^-)$ and $E_2 = (E^+, E^- \cup \{ e_2 \})$ where e_1 is an instance from the concept c and e_2 is not. If $\theta(c) < 0.5$ then $P(h_c|E_1) > P(h_c|E_2)$.*

PROOF: We omit the proof.

A similar result can be established about the importance of negative examples over positive examples for concepts, c, with $\theta(c) > 0.5$.

3 LIME System

The above heuristic has been incorporated in an inductive logic programming system, LIME, as briefly describe in this section. We first define the standard setting for ILP and discuss how the heuristic can be adapted to this setting. We then discuss the architecture of LIME, ending the section with a few words on how LIME estimates prior probability of hypotheses.

A system with background knowledge B is given a set of positive ground facts, E^+, and a set of negative ground facts, E^-, about a target predicate. The system is required to come up with a hypothesis h' such that for all $e \in E^+$, $B \cup \{ h' \} \models e$ and for all $e \in E^-$, $B \cup \{ h' \} \not\models e$. In order to adapt the Bayesian heuristic

to this setting, we need to interpret h in Equations 1–6 as $B \cup \{ h' \}$.

LIME's functioning may be described in three stages: (a) preprocessing of the background knowledge, (b) generation of candidate clauses, and (c) search for the most probable hypothesis.

LIME preprocesses the backgound knowledge to identify functional dependencies, type information, and redundancies [McCreath and Sharma, 1995]. This step is very helpful in reducing the search space. Most ILP systems require such information to be explicitly provided together with data.

The second phase then uses the above information together with the examples and the background knowledge to construct a large set of candidate clauses. These candidates are selected on the basis of their potential to form part of the most probable hypothesis. Subsets of this set are searched for a set of clauses, h, such that the value of $Q(h)$ is maximized. Clearly, it is infeasible to examine all subsets, although, an exhaustive, but finite, search will find an optimal hypothesis. So, we digress a little to describe the following theorem that gives a bound on the search space for the most probable hypothesis.

Theorem 2 *Suppose* $h_{induced} = \max_{h \in H} Q(h)$. *Then for any* $h' \in H$,

$$2^{Q(h') - |E^+| \lg(\frac{1-\epsilon}{\psi(E^+)} + \epsilon) - |E^-| \lg(\frac{1-\epsilon}{\psi(E^-)} + \epsilon)} \leq P(h_{induced}),$$

where $\psi(E) = D_X(\min_{e \in E} D_X(e))$ *is the smallest probability according to the instance space distribution of examples from* E.

PROOF: We omit the proof.

The above result is useful because once the value of $Q(h)$ is known for any $h \in H$, the system has a bound on the prior probability of the most probable hypothesis, thereby restricting search to a finite well defined set of hypotheses. Since there are at least three such readily available hypotheses—the hypothesis that entails the entire instance space, the hypothesis that entails no memebr of the instance space, and the hypothesis that exactly entails the positive examples—such a bound can be determined.

The above discussion notwithstanding, it is easy to see that the bound still leaves an infeasible amount of search to be done. However, these sets form a lattice under the subset operation in such a way that the more general elements of the lattice have both lower prior probablity and are more general in terms of the instances they cover. This property gives structure to the search space, and allows each element in the lattice to be assigned a value that estimates an upper bound on the heuristic value of more general elements in the lattice. An A*-like search of this lattice produces the final hypothesis. Entire branches of the lattice may be pruned if the estimated upper bound for a branch is less than the heuristic's value of the best hypothesis so far. This pruning reduces expected execution time significantly. Also, before a hypothesis replaces the best hypothesis so far, a PROLOG interperator is used to determine the hypothesis' exact coverage, thereby providing a more accurate estimate of the heuristic. This step also removes poorly constructed recursive hypotheses, thereby enhancing LIME's ability to learn recursive logic programs. A detailed description of LIME will be given in a more expanded paper.

We now briefly discuss the computation of $Q(h)$ for a hypothesis h. ¿From Equation 7, it is clear that computation of $Q(h)$ requires (i) cardinality of TP, TN, and FPN; (ii) $\theta(\text{ext}(h))$; and (iii) $P(h)$—the prior probability of the hypothesis h. Each of the items in (i) can be estimated from the example sets, h, and a PROLOG interpreter. $\theta(\text{ext}(h))$ is estimated by randomly generating a set of instances and finding the proportion in this set of the concept represented by h with the help of a PROLOG interpreter. The computation of the prior $P(h)$ is somewhat involved. Each clause in a hypothesis is considered independent and as more clauses are added, the prior probability of a hypothesis decreases. The prior probability of a clause is computed from prior probability of literals, which in turn is computed from prior probability of variables. Current implementation of LIME uses the geometric distribution in the above computations. However, it turns out that as the number of examples increases, the prior probability of a hypothesis h becomes less and less important in the computation of $Q(h)$. This result can be formally established using the Borel-Cantelli lemma. The details of the $P(h)$ and the associated theoretical result will be presented in an expanded version of the paper.

4 Empirical Results

We present three sets of experiments to illustrate how LIME achieves its design goals of better noise handling, learning from fixed set of examples, and of learning recursive logic programs. Since these goals are our main focus here, we have omitted to include the time performance of LIME. It should be noted that LIME's time performance is of the same order as that of the other systems being compared in this study.

Noise

We first demonstrate LIME's superior noise handling capability for the simple concept `plus2`, which may be represented by the following logic program:

$$\texttt{plus2}(A, B) \leftarrow \texttt{inc}(A, C), \texttt{inc}(C, B).$$

In the above `inc` denotes the increment predicate available as background knowledge. A random selection of 50 positive and 50 negative examples are given to LIME.

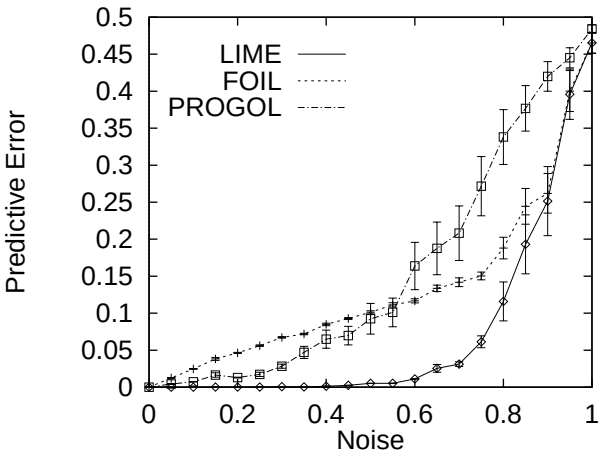

Figure 2: Predictive Error vs Noise for `plus2`

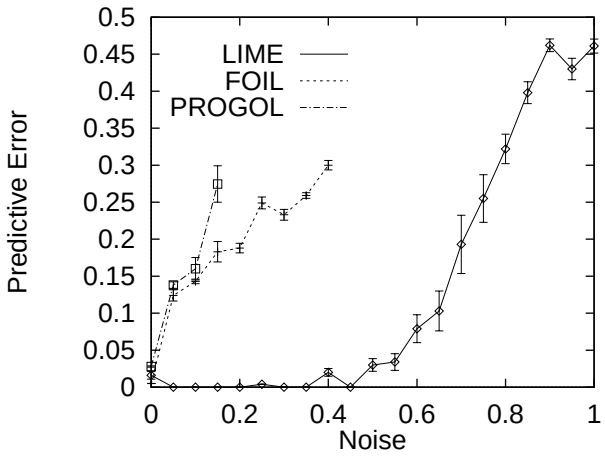

Figure 3: Predictive error vs Noise for `add`

These examples include noise. The predictive error of the induced hypothesis is measured against a noise-free test set that is generated by taking the "first" 20 positive examples and a random selection of 20 negative examples. This process is repeated 100 times to calculate the average predictive error. This is repeated with different noise levels and the results are shown in Figure 2. The error bars in the figure indicate the sample standard deviation. The results show that LIME is able to correctly learn the concept with noise levels of up to approxamatly 70%. The same test is carried out with FOIL and PROGOL.[5]

LIME performs better than FOIL and PROGOL for noise levels of up to approximately 70%. Here, FOIL overgenerizes inducing a less predictive hypothesis. This is mainly due to the covering approach which introduces unnecessary clauses. However, for noise levels higher than 70%, LIME, PROGOL and FOIL perform poorly.

We next show LIME's noise handling ability with `add` (the addition relation)—a more complex target predicate that requires a recursive definition. The target concept may be represented by the hypothesis:

$$\texttt{add}(A,B,C) \leftarrow \texttt{equal}(A,C),\texttt{zero}(B).$$
$$\texttt{add}(A,B,C) \leftarrow \texttt{inc}(D,B),\texttt{add}(A,D,E),\texttt{inc}(E,C).$$

This time we take a random selection of 200 positive and 200 negative examples but perform only 20 repeations at each noise level. Figure 3 shows the relationship between noise and predictive error measured against a noise-free test set of the "first" 25 positive examples and a random set of 25 negative examples. The results show that gap between LIME and other system widens further when the target concept requires a recursive definition. Experiments with FOIL and PROGOL were limited to 40% and 15% noise levels respectively because the quality of the programs output by these systems beyond these noise levels were difficlut to assess.

Learning from positive examples and from negative examples

Our second set of experiments shows empirical evidence for Theorem 1 which implies that positive examples are more useful than negative examples for a target concept that is "small" with respect to the instance space distribution. The experiments also give evidence for the converse that negative examples are more useful than positive examples for a target concept that is "large" with respect to the instance space distribution. These experiments also establish that LIME is capable of learning from only positive data and from only negative data.

We consider two concepts, the `plus2` and `notplus2` (the complement of `plus2`—that is, $\texttt{notplus2}(A,B)$ holds if $B \neq A+2$). It is easy to see that under reasonable assumptions, `plus2` is a "small" concept and `notplus2` is a "large" concept. LIME is run on examples of `plus2` and `notplus2` with identical background knowledge. The total number of examples is invariant over each test, however, the number of positive examples is increased as the number of negative examples is decreased. Each test is repeated 100 times and the results for both `plus2` and `notplus2` are shown in Figure 4.

Recursive Logic Programs

Table 1[6] summarizes experimental results on LIME's ability to learn a number of predicates that require recursive definitions. It should be noted that the data sets used in these experiments are noisy and not contiguous. FOIL cannot learn these predicates from such data sets.

[5] All our experiments are with FOIL, version 6.3 and with PROGOL, version 4.1.

[6] the column titled "incorrect" denotes

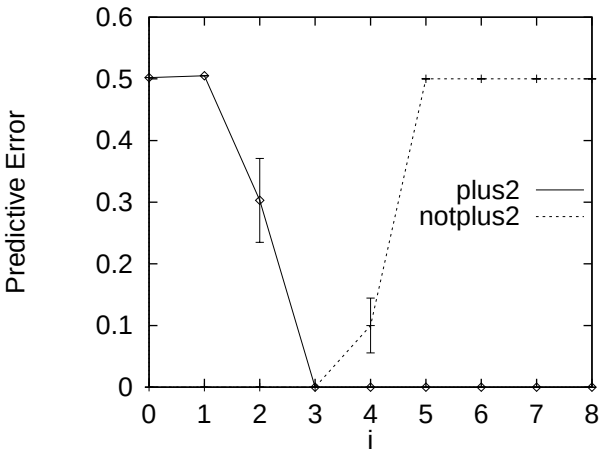

Figure 4: Error vs i (i = number of positive examples & 8-i = number of negative examples) for the `plus2` and `notplus2` logic programs

Logic Program	$\lvert E^+\rvert$ / $\lvert E^-\rvert$	% incorrect	Time (Sec)
$\texttt{add}(A,B,C) \leftarrow \texttt{equal}(A,C), \texttt{zero}(B).$ $\texttt{add}(A,B,C) \leftarrow \texttt{inc}(D,B), \texttt{add}(A,D,E), \texttt{inc}(E,C).$	16/10	7.8%	0.31
$\texttt{member}(A,B) \leftarrow \texttt{components}(B,A,C).$ $\texttt{member}(A,B) \leftarrow \texttt{components}(B,C,D), \texttt{member}(A,D).$	27/24	16.6%	41.89
$\texttt{mult}(A,B,C) \leftarrow \texttt{inc}(D,B), \texttt{add}(E,A,C), \texttt{mult}(A,D,E).$ $\texttt{mult}(A,B,C) \leftarrow \texttt{zero}(B), \texttt{zero}(C).$	16/24	2.5%	44.53
$\texttt{sort}(A,B) \leftarrow \texttt{components}(A,C,D), \texttt{insert}(C,E,B), \texttt{sort}(D,E).$ $\texttt{sort}(A,B) \leftarrow \texttt{empty}(A), \texttt{empty}(B).$	162/153	0.0%	578.63

Table 1: Some recursive logic programs

5 Discussion

Another approach to modeling noise in learning systems is due to [Angluin and Laird, 1987]. Their noise level parameter measures the percentage of data with the incorrect sign, that is, elements of the concept being mislabeled as negative data and vice versa. In their model 50% noise level means the data is truly random, whereas in our model truly random data is at noise level of 100%. Thus, in their model it is not useful to consider noise levels of greater than 50%. Our current model requires that the noise level be provided to the system. Although this may appear to be a weakness, in practice, a reasonable estimate suffices, and it can be shown that with increase in the example size, the impact of an inaccurate noise estimate diminishes. It should be noted that experiments reported in this paper always used a noise parameter of 10% in computing $Q(h)$ even if the actual noise in the data was considerably higher.

Recently, we have become aware of the work of [Muggleton, 1996] in which he has used a Bayesian heuristic for learning from only positive data. Interestingly, if we take the noise level to be 0 in our model and only consider positive examples, then our heuristic becomes identical to Muggleton's.

Future work will attempt to derive stochastic convergence in the limit results for the noise model presented in this paper in the style of Laird's [Laird, 1988] result for the Angluin-Laird noise model. Another direction would be to do adapt the predicative error analysis of [Muggleton, 1996] for the Bayesian heuristic with noise and fixed example size. On the empirical front, applicability of LIME on additional real-world domains will be investigated. To this end we would like to note that initial experiments with LIME on protein secondary structure data show comparable results to GOLEM.

ACKNOWLEDGEMENTS We would like to thank M. Bain for bringing [Muggleton, 1996] to our attention. We would also like to thank the referees for several valuable comments that have improved the paper.

References

[Angluin and Laird, 1987] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2:343–370, 1987.

[Laird, 1988] P. Laird. *Learning from Good and Bad Data.* Kluwer Academic Publishers, Boston, MA, 1988.

[Lavrac and Dzeroski, 1992] N. Lavrac and S. Dzeroski. Inductive learning of relations from noisy examples. In S Muggleton, editor, *Inductive Logic Programming*, pages 495–516. Academic Press, 1992.

[Lavrac *et al.*, 1996] N. Lavrac, S. Dzeroski, and I. Bratko. Handling imperfect data in inductive logic programming. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 48–64. IOS Press, 1996.

[McCreath and Sharma, 1995] E. McCreath and A. Sharma. Extraction of meta-knowledge to restrict the hypothesis space for ILP systems. In X. Yao, editor, *Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence*, pages 75–82. World Scientific, November 1995.

[Muggleton and Feng, 1990] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory, Tokyo*, pages 368–381. Ohmsa Publishers, 1990. Reprinted by Ohmsa Springer-Verlag.

[Muggleton, 1996] S. Muggleton. Learning from positive data. In *Proceedings of the Inductive Logic Programming Workshop*, 1996.

[Quinlan, 1990] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[Rissanen, 1978] J. Rissanen. Modeling by shortest data description. *Auotomatica*, 14:465–471, 1978.

[Wallace and Freeman, 1987] C. Wallace and P.R. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society (B)*, 49:240–265, 1987.