MetaBayes: Bayesian Meta-Interpretative Learning using Higher-Order Stochastic Refinement

Stephen H. Muggleton, Dianhuan Lin, Jianzhong Chen and Alireza Tamaddoni-Nezhad

Department of Computing, Imperial College London

Abstract. Recent papers have demonstrated that both predicate invention and the learning of recursion can be efficiently implemented by way of abduction with respect to a meta-interpreter. This paper shows how Meta-Interpretive Learning (MIL) can be extended to implement a Bayesian posterior distribution over the hypothesis space by treating the meta-interpreter as a Stochastic Logic Program. The resulting $MetaBayes$ system uses stochastic refinement to randomly sample consistent hypotheses which are used to approximate Bayes' Prediction. Most approaches to Statistical Relational Learning involve separate phases of model estimation and parameter estimation. We show how a variant of the MetaBayes approach can be used to carry out simultaneous model and parameter estimation for a new representation we refer to as a Super-imposed Logic Program (SiLPs). The implementation of this approach is referred to as $MetaBayes_{SiLP}$. SiLPs are a particular form of ProbLog program, and so the parameters can also be estimated using the more traditional EM approach employed by ProbLog. This second approach is implemented in a new system called $MilProbLog$. Experiments are conducted on learning grammars, family relations and a natural language domain. These demonstrate that $MetaBayes$ outperforms $MetaBayes_{MAP}$ in terms of predictive accuracy and also outperforms both $MilProbLog$ and $MetaBayes_{SiLP}$ on log likelihood measures. However, $MetaBayes$ incurs substantially higher running times than $MetaBayes_{MAP}$. On the other hand, $MetaBayes$ and $MetaBayes_{SiLP}$ have similar running times while both have much shorter running times than $MilProbLog$.

1 Introduction

In [19] grammars are learned using a special-purpose Prolog meta-interpreter. Hypotheses are generated along various SLD derivation paths by abduction. The approach is generalised in this paper by 1) replacing the special-purpose meta-interpreter by a general-purpose meta-interpreter which interprets user-provided meta-rules and 2) treating the meta-rules as a Stochastic Logic Program (SLP) [15, 3]. In this setting we can view the hypotheses as being derived using Stochastic Refinement [23]. Figure 1 illustrates a Stochastic Refinement tree [23] for constructing a Finite State Acceptor (FSA). In this tree each path leading to a hypothesis can be interpreted either a) as a series of refinements leading to a headless Horn clause, representing the negation $\neg H$ of the ground abductive hypothesis H (see Figure 1a) or b) as the derivation of a finite state acceptor

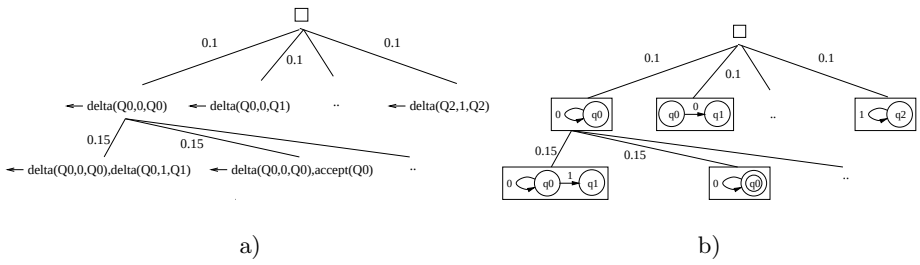


Fig. 1. Stochastic Refinement tree showing a) clause containing arcs (delta) and acceptors, b) corresponding Finite State Acceptors. Stochastic Refinement tree edge labels represent selection probabilities.

by a meta-interpreter applied to an SLP (Figure 1b). Furthermore, as in [2], we can view the SLP as a structural Bayes' prior over the hypothesis space. In this case, the posterior is formed by using the positive and negative examples to prune sub-trees from the prior. Following pruning, selection probabilities for each sub-tree are renormalised in the posterior.

1.1 Bayesian MIL versus Probabilistic ILP

According to Bayesian learning theory [7, 4], maximal predictive accuracy in learning is achieved by using a diversity of models, with predictions weighted according to the sum of posterior probability of the corresponding hypothesis. The implementation of such a posterior probability as a stochastic refinement graph thus provides a direct way of using hypothesis sampling approaches to approximate maximal accuracy machine learning. The relationship between Bayesian MIL (BMIL) and traditional Probabilistic ILP [22] is illustrated in Figure 2.

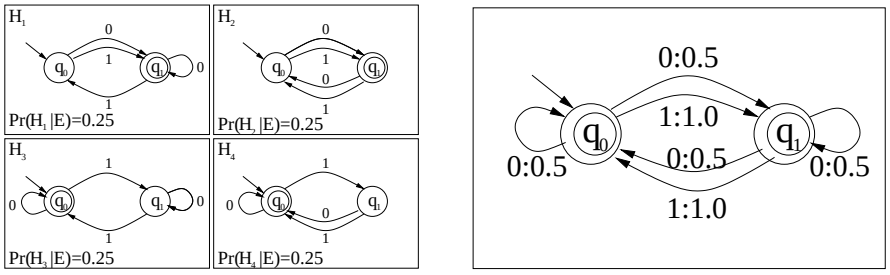


Fig. 2. a) Finite State Acceptor hypotheses generated by BMIL from examples $e^+ = 101011011$ and $e^- = 111101$. b) Super-imposed Logic Program formed from hypotheses in a).

In BMIL the model consists of a set of logic programs, each with an associated probability. By contrast, in Probabilistic ILP approaches such as ProbLog [8], Bayesian Logic Programs [12] and Stochastic Logic Programs [16, 17] the model consists of a single logic program with probabilistic parameters associated with individual clauses. These representations use implicit independence assumptions to support probabilistic inference. Thus, according to [22] "A ProbLog program defines a distribution over logic programs by specifying for each clause the probability that it belongs to a randomly sampled program, and these probabilities are mutually independent." Figure 2a illustrates a uniform posterior distribution over four two-state FSA hypotheses consistent with a pair $\langle e^+, e^- \rangle$ of positive and negative examples. By contrast, Figure 2b shows a Super-imposed Logic Program (SiLP) which can be viewed as a summary of the distribution in Figure 2a. The SiLP is formed by labelling each arc by the sum of the posterior probabilities of hypotheses in Figure 2a containing that arc. Note that a SiLP is a ProbLog program since the label for each clause (the arcs and acceptors) represents the probability that it belongs to a randomly sampled logic program drawn from the posterior distribution shown in Figure 2a.

In order to show how Bayes' prediction avoids certain forms of error associated with Probabilistic ILP representations, we note that the Bayes' prediction for the negative example $e^- = 111101$ based on the distribution in Figure 2a is zero since, by construction, no one of the FSAs accepts this string. However, the ProbLog program illustrated in Figure 2b predicts this sequence has a probability greater than zero. The discrepancy derives from the fact that, contrary to the ProbLog assumption, the arcs in Figure 2b are clearly not mutually independent within the FSAs in Figure 2a.

1.2 Multiple and single models

By consideration of Figures 2a and 2b we now compare the relative advantages of a multiple model predictor versus a single-model predictor.

Multiple models. The key advantage here is the *maximal expected predictive accuracy* offered by Bayes' prediction. It is assumed the target theory is selected randomly according to the hypothesis prior over the hypothesis space $\mathcal{H}$. Prediction that instance $x = True$ is based on the sum of posterior probabilities of all consistent hypotheses in $\mathcal{H}$ which make this prediction. This approach is infeasible in the case of $\mathcal{H}$ being a large or infinite space, though in this case it can be approximated by making predictions based on a sample of hypotheses.

Single model. This has the advantages of *increased understandability*. We can view a SiLP as providing a summarisation of the hypothesis space. In Figure 2a we see that most (actually all) consistent hypotheses have 1-arcs from state q_0 to q_1 and q_1 to q_0. By contrast, all the 0-arcs have probability 0.5, meaning they are as likely to be true as false.

The paper is organised as follows. Section 2 describes the MetaBayes Refinement framework. The implementation of the systems MetaBayes, MetaMAP, $\text{MetaBayes}_{\text{SiLP}}$ and MilProbLog (not to be confused with MetaProbLog [14]) are then given in Section 3. Experiments on binary prediction (MetaBayes vs MetaMap) and probabilistic prediction (MetaBayes vs $\text{MetaBayes}_{\text{SiLP}}$ vs MilProbLog) are conducted on various datasets, including Finite State Automata

(FSAs), the ancestor relation for the Russian Royal Family and learning language semantics. In Section 5 we provide a comparison to related work. Lastly we conclude the paper and discuss future work in Section 6.

2 MetaBayes Refinement framework

2.1 Setting

The setting for Meta-Interpretive Learning (MIL) [19] assumes as input a specialised Meta-interpreter B_M together with two sets of ground atoms representing background knowledge B_A and examples E respectively. The result of learning is a revised form of the background knowledge containing the original background knowledge B_A augmented with additional ground atoms representing a hypothesis H. We assume H is derived from $B = B_A$ and E. Applying Inverse Entailment $B, H \models E$ is equivalent to $B, \neg E \models \neg H$. In this form we see that $B, \neg E$ is given to the meta-interpreter where $\neg E$ is a goal and the resulting abduced program $\neg H$ represents a headless Horn clause such as those shown in Figure 1a.

2.2 Generalised meta-interpreter

A series of specific variants of special-purpose meta-interptreters are given in [19, 18] for Regular grammars ($Metagol_R$), Context-free grammars ($Metagol_{CF}$) and a fragment of Dyadic definite clause logic ($Metagol_D$). Figure 3a shows a generalised Meta-Interpreter which can emulate each special-purpose meta-interpreter using a set of domain specific meta-rules such as those shown for finite state acceptors (Figure 3b) [19] and the fragment of dyadic definite clauses (Figure 3c) investigated in [18]. As discussed in [18] a meta-rule is a higher-order wff

$$\exists \mathcal{S} \forall \mathcal{T} P(s_1, .., s_m) \leftarrow .., Q_i(t_1, .., t_n), ..$$

where $\mathcal{S}, \mathcal{T}$ are disjoint sets of variables, $P, Q_i \in \mathcal{S}$ and $s_j, t_k \in \mathcal{T}$. For instance, the second finite state acceptor meta-rule in Figure 3 indicates that with suitable higher-order ground substitition for the existentially quantified variables $\mathcal{S} = \{P, C, Q\}$ the higher-order atom $delta(P, C, Q)$ can be interpreted as the first-order clause $P([C|X], Y)$:- $Q(X, Y)$. In this way higher-order abduction of a set of atoms can be interpreted as first-order induction of a definite program.

2.3 Stochastic refinement

According to [23] a downward *stochastic unary refinement operator* is a function $\sigma : G \rightarrow 2^{G \times [0,1]}$ defined as follows: $\sigma(C) = \{\langle D_i, p_i \rangle | D_i \in \rho(C),\ p_i \in [0,1]$ and $\sum p_i = 1$ for $1 \leq i \leq |\rho(C)|\}$ and $\sigma^*(C) = \{\langle D_i, p_i \rangle | D_i \in \rho^*(C),\ p_i \in [0,1]$ and $\sum p_i = 1$ for $1 \leq i \leq |\rho^*(C)|\}$. In [23] it is shown that the n-step stochastic refinements of a clause represent a probability distribution. In the context of the meta-interpreter of Figure 3 we can consider the refinement function ρ to consist of the selection of a consistent meta-rule followed by the related abduction of a higher-order atom[1]. Stochastic refinement with respect to a meta-interpreter involves making selections according to a probability distribution over the meta-rules.

[1] abduce/3 only adds a higher-order atom a to a program P to give P' when $a \notin P$.

a) Generalised meta-interpreter

```
prove([], Prog, Prog).
prove([Atom|As], Prog1, Prog2) :-
        metarule(RuleName, HO_Sub, (Atom :- Body), OrderTest),
        OrderTest,
        abduce(metasub(RuleName, HO_Sub), Prog1, Prog3),
        prove(Body, Prog3, Prog4),
        prove(As, Prog4, Prog2).
```

b) Meta-rules for finite state acceptors

```
metarule(acceptor, [Q], ([Q, [], []] :- []), (term(Q))).
metarule(delta, [P, C, Q], ([P, [C|X], Y] :- [[Q, X, Y]]),
        (nonterm(Q), nonterm(P))).
```

c) Meta-rules for dyadic fragment

```
metarule(instance, [P, X, Y], ([P, X, Y] :- []), (pred(P))).
metarule(base, [P, Q], ([P, X, Y] :- [[Q, X, Y]]),
        (pred_above(P, Q), obj_above(X, Y))).
metarule(tailrec, [P, Q], ([P, X, Y] :- [[Q, X, Z], [P, Z, Y]]),
        (pred_above(P, Q), obj_above(X, Z), obj_above(Z, Y))).
metarule(chain, [P, Q, R], ([P, X, Y] :- [[Q, X, Z], [R, Z, Y]]),
        (pred_above(P, R), obj_above(X, Z), obj_above(Z, Y))).
```

Fig. 3. Prolog representation of a) Generalised meta-interpreter, b) Regular Grammar meta-rules and c) Dyadic fragment meta-rules

2.4 Prior, Likelihood and Posterior

The prior of H relative to background knowledge B can now be defined as $Pr(H|B) = \sum_{\langle H,p\rangle \in \sigma^*(\neg B)} p$ and $Pr(H) = Pr(H|\emptyset)$. The likelihood of examples E with respect to the background knowledge B and hypothesis H is $Pr(E|B,H) = \begin{cases} 1 & \text{if } B, H \models E \\ 0 & \text{otherwise} \end{cases}$. Using Bayes' theorem the posterior is

$$Pr(H|B,E) = \frac{Pr(H|B)Pr(E|B,H)}{c}$$

where c is a normalisation constant. A hypothesis H is said to be MAP in the case that $H \in \text{argmax}_H Pr(H|B,E)$. A Bayes' prediction of instance x is defined by the function $\text{BayesP}(x) = \begin{cases} 1 & \text{if } \sum_H Pr(H|B,E) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$.

3 Implementation

Below we describe the implementation of four systems: MetaBayes, MetaMAP, $\text{MetaBayes}_{\text{SiLP}}$ and MilProbLog. MetaBayes and MetaBayes_{MAP} are variants of the generalised meta-interpreter where stochastic refinement of the meta-rules is assumed to be conducted using a uniform distribution at each internal node of the refinement tree. $\text{MetaBayes}_{\text{SiLP}}$ is based on MetaBayes. Both $\text{MetaBayes}_{\text{SiLP}}$ and MilProbLog output probabilistic programs.

3.1 MetaBayes

This algorithm carries out an approximation to Bayes' prediction based on averaging over the posterior probabilities of a set $\mathcal{H}$ consisting of a sample of consistent hypotheses. The set is generated using a method we refer to as *Regular Sampling*, in which hypotheses are generated based on a series of fractions from the sequence $0, \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}, ...$ This sequence has the property of being evenly distributed in the unit interval $[0, 1]$ without repeating the same fractional value twice. Considering the consistent hypotheses to be ordered $H_1, H_2, ..$ left-to-right in SLD order within the derivation tree, the fraction p_i is used to find the rightmost H_j such that $\sum_{k=1}^{j} Pr(H_k|B, E) \leq p_i$. This is achieved efficiently by considering that the cumulative posterior probability (the sum of posterior probabilities of hypothesis preceding a given hypothesis in the derivation tree) associated with hypotheses found in the sub-trees under each node of the stochastic refinement tree is partitioned into equally sized intervals. Starting at the root of the refinement tree H_j will be found in the sub-tree whose cumulative posterior probability interval $[\min, \max]$ is such that $\min \leq p_i < \max$. Within this sub-tree we repeat by selecting the sub-tree containing the probability $(p_i - \min)(\max - \min)$. The iteration is terminated by the hypothesis returned by the base case of the meta-interpreter. By bounding the posterior probability sum of the sample and ignoring duplicates the approach can be made to achieve the effect of sampling without replacement. Although slightly more complex to program than an alternative implementation of sampling with replacement, the Regular Sampling approach achieves higher efficiency by minimising duplicate sampling due to the spread of hypotheses chosen by the sequence of fractions.

3.2 MetaBayes$_{MAP}$

This algorithm carries out predictions based on the leftmost consistent hypothesis at minimal depth in the stochastic refinement tree. This hypothesis can be found efficiently using iterative deepening of derivations from the generalised meta-interpreter.

3.3 MetaBayes$_{SiLP}$

This algorithm superimposes the set of hypothesises sampled by MetaBayes. Specifically, the posterior probabilities of sampled hypotheses are renormalised. Then the summation is carried out for each clause C present in the set of sampled hypothesises. It follows Equation 1, where C denotes a clause, $p(H_i|E)$ is the posterior probability of a hypothesis H_i, $p(C|H_i)$ means the probability of C being true given that H_i is true, thus $p(C|H_i)$ is either 1 or 0, depending on whether C is part of H_i. This equation is essentially the same as that of Bayesian prediction on atoms, except predicting the probabilistic labels on clauses instead of atoms.

$$p(C|E) = \sum p(C|H_i) * p(H_i|E) \tag{1}$$

3.4 MilProbLog

This algorithm is based on ProbLog but loaded with a meta-interpreter and all possible meta-substitutions, which essentially provides all possible hypothesis

clauses. In this way, a learning task requiring simultaneous model and parameter estimation is reduced to only parameter estimation. If a clause is assigned with probability zero, then it implies that this clause is hypothesised as *not* part of the learned structure.

4 Experiments

In this section we describe experiments which compare MetaBayes to MetaMAP, as well as the comparison among MetaBayes, MetaBayes$_{\mathrm{SiLP}}$ and MilProbLog.

4.1 Binary prediction - MetaBayes vs. MetaMap

We first consider the following two Null hypothesis which compares MetaBayes to MetaMAP in terms of predictive accuracy and running time. We use datasets of learning FSAs and learning the concept of ancestor.

Null Hypothesis 1.1 MetaBayes does not have higher expected predictive accuracy than MetaMAP.

Null Hypothesis 1.2 MetaBayes does not have longer expected running time than MetaMAP.

Learning FSAs

Materials and Methods 200 randomly chosen FSA were generated using Metagol$_R$ [19]. Specifically, a set of sequences were randomly chosen from Σ^* for $\Sigma = \{0, 1\}$, then they are used as training examples for Metagol$_R$ to learn FSAs. The target FSAs derived in this way are guaranteed to be minimal, since Metagol$_R$ finds a minimal hypothesis. For each target grammar, 20 training examples were randomly chosen from Σ^* for $\Sigma = \{0, 1\}$. Another 1000 test examples were also randomly sampled without replacement. The examples are half positive and half negative, therefore the default accuracy is 50%. Predictive accuracies and associated learning times were averaged over the 200 FSAs. We plot the learning curves at training sizes $\{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$.

The learning systems being compared are MetaBayes and MetaMAP. The MetaMAP system makes binary prediction, while MetaBayes makes probabilistic prediction. To make them comparable, we use 0.5 as threshold to discretise the prediction by MetaBayes. Specifically, if a prediction made by MetaBayes is larger than 0.5, it is regarded as the positive, otherwise equal to 0.5 or smaller than 0.5 are considered as the negative. MetaBayes' performance varies with the size of sampled hypotheses and the given prior. Therefore we run the experiment with sample sizes as 10, 100, 500, 750 and 1000. For the prior, we considered priors which are exponential, polynomial and uniform with respect to the description length of a hypothesis. We also considered an informative prior, which is similar to the exponential prior except being given additional information about the size of a target hypothesis. The informative prior P_{inf} is defined as below, where $dl/1$ is a function which returns the description length of a hypothesis. In the case there is no sampled hypotheses having the same size as a target hypothesis, the informative prior reduces to the exponential prior.

$$P_{inf}(H) = \begin{cases} 1 & dl(H) = dl(TargetH) \\ (1/2)^{dl(H)} & dl(H) \neq dl(TargetH) \end{cases}$$

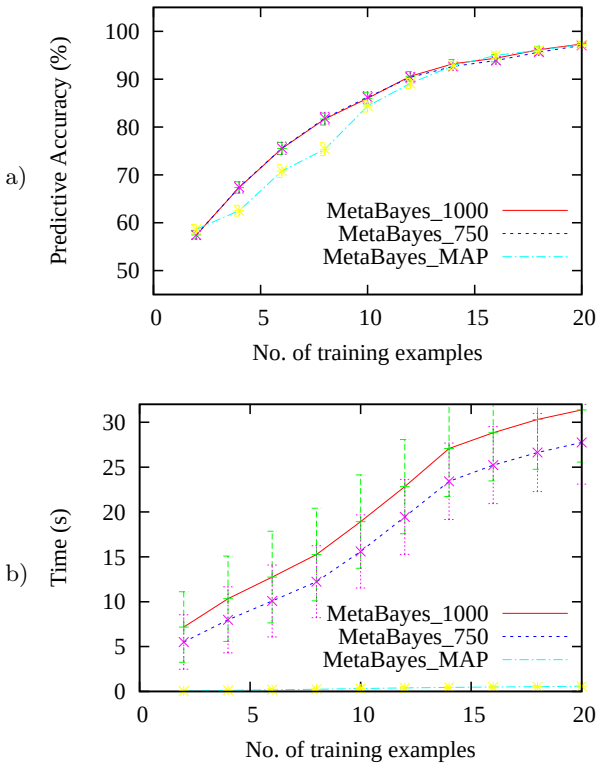


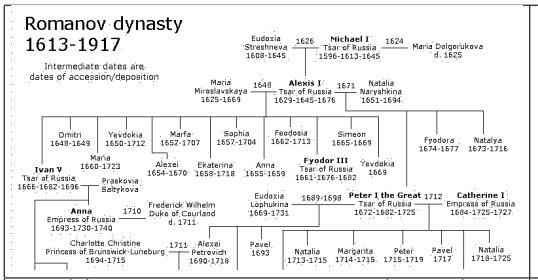
Fig. 4. Average results for learning FSAs showing a) Predictive accuracies (informative prior) and b) Running times

Results and Discussion Figure 4(a) shows that MetaBayes given an informative prior has significantly higher predictive accuracies than that of MetaMAP. MetaBayes with higher sample rate 1000 also has slightly higher accuracies that of 750. However, the improvement on accuracy comes at cost of running time. As shown in Figure 4(b), the total running time of MetaBayes with sample size 1000 is about 30 times longer than that of MetaMAP. The increase of sample rate in MetaBayes also significantly increase the running time. Therefore both Null hypothesis 1.1 and 1.2 are refuted by the experiment of learning 200 randomly chosen FSAs. Other results of MetaBayes with different sample sizes and priors which does not significantly outperform MetaMAP are not plotted due to limited space.

Learning the concept of ancestor

Materials and Methods We used the family tree of Russian royal family (Romanov dynasty 1613-1917), which involves 12 generations and 119 persons. Part of the family tree is shown in Figure 5a. This dataset has previously been

used in [21]. The background knowledge contains only facts of $father/2$ and $mother/2$. The examples consist of only $ancestor/2$. No example of $parent/2$ is given, therefore to learn the target hypothesis in Figure 5b would require not just recursion, but also predicate invention. 60 training examples with half positive and half negative are randomly chosen. They were divided into 5 folds with size 12. Results from the 5 folds were averaged. There are 1000 test examples. The default accuracy is 50%. We plot the learning curves at training sizes $\{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$. The rest are the same as that in learning FSAs.



(a) Russian royal family tree (partial)

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z),
                 ancestor(Z,Y).
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).
```

(b) Target hypothesis of ancestor

Fig. 5. Learning the concept of ancestor

Results and Discussion Similar to the accuracy graph in Figure 4a, Figure 6a also shows that MetaBayes given an informative prior has significantly higher predictive accuracies than that of MetaMAP. The running time difference between MetaBayes and MetaMAP is even larger than that of learning FSAs, as shown in Figure 6b. This is due to the dramatic increase in hypothesis space. In contrast, the concept of ancestor requires H_2^2 representation (Dyadic Datalog programs), which has Universal Turing Machine expressivity [18]. Considering MetaBayes samples from the entire hypothesis space while MetaMAP only searches through part of the hypothesis space to find the shortest hypothesis, the increase of hypothesis space has larger impact on MetaBayes than MetaMAP. Therefore both Null hypothesis 1.1 and 1.2 are also refuted by the above results. The running time graph in Figure 6b has very large deviations. Since one of the folds contains examples of ancestor involving 10 generations, which leads to significantly longer running time than the other four folds.

4.2 Probabilistic prediction - MetaBayes vs. MetaBayes$_{\mathbf{SiLP}}$ vs. MilProbLog

In this subsection, we investigate Null hypothesis 2.1, 2.2 and 2.3 about the comparison among MetaBayes, MetaBayes$_{\text{SiLP}}$ and MilProbLog. Considering all the three systems make probabilistic predictions, we use likelihood instead of accuracy as the criterion for comparison.

Null Hypothesis 2.1 MetaBayes does not have higher expected likelihood than MetaBayes$_{\text{SiLP}}$.

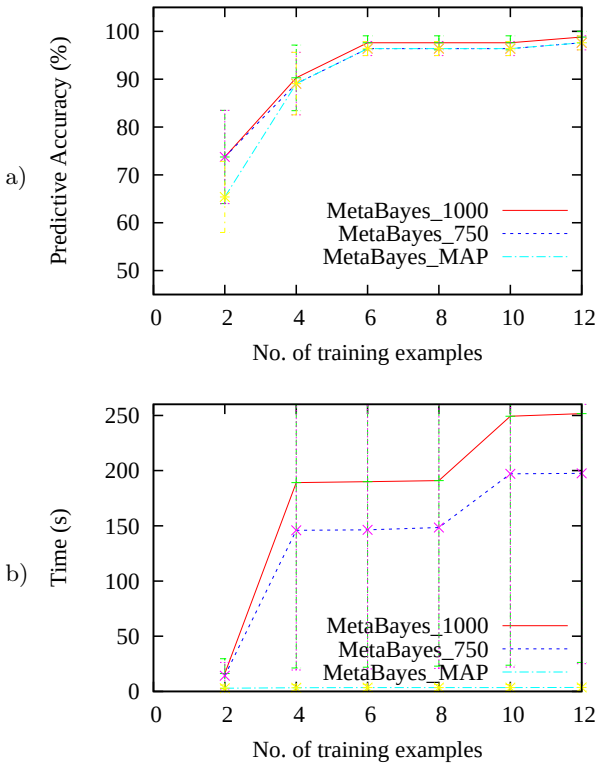


Fig. 6. Average results for learning the concept of ancestor showing a) Predictive accuracies (informative prior) and b) Running times

Null Hypothesis 2.2 MetaBayes does not have higher expected likelihood than MilProbLog.

Null Hypothesis 2.3 $\text{MetaBayes}_{\text{SiLP}}$ does not have higher likelihood than MilProbLog.

Learning FSAs

Materials and Methods Considering MilProbLog requires the input of all possible hypothesis clauses, we have to constrain the hypothesis space to be enumerable. Therefore we considered learning FSAs with the number of maximum states N_s as 2, 3 and 4, respectively. We used regular sampling to randomly sample 100 different pairs of sequences from Σ^* for $\Sigma = \{0, 1\}$ with maximum length 10. For each pair of sequences, one is used as the positive while the other is used as the negative training examples. All possible FSAs with maximum N_s states that can be derived from the pairs are included in the set of target FSAs. For example, if $e^+ = 101011011$ and $e^- = 111101$ is one of the pairs of sampled sequences and $N_s = 2$ (only two states q_0 and q_1 are allowed), then there are

four FSAs generable, as depicted in Fig. 2 (a). Then all the four FSAs are part of the target FSAs. We used all the sequences with maximum length 10 as test examples, thus the size of test examples is 2047. MilProbLog was run with 10 iterations.

Results and Discussion Figure 7 shows an example of the probabilistic programs output by MetaBayes$_{\text{SiLP}}$ and MilProbLog. Since MetaBayes$_{\text{SiLP}}$ carries out simultaneous model and parameter estimation, it does not require the provision of candidate hypothesis clauses, but only generates those candidates from examples in a data-driven fashion. That is why there are two clauses marked as *not_generated* in Figure 7a. In contrast, MilProbLog only works when given structures. Therefore it requires all candidate clauses being provided as input, even though some of the clauses will not be used for explaining the positive examples or inconsistent with negative examples. That is why there are parameters assigned as 0 in Figure 7b. Therefore MetaBayes$_{\text{SiLP}}$ is more efficient than MilProbLog, especially when the hypothesis space is large.

0.5 :: q0 →	0 :: q0 →
0.5 :: q0 → 0 q0	0 :: q0 → 0 q0
1. 0 :: q0 → 1 q1	1. 0 :: q0 → 1 q1
1.0 :: q1 → 1 q0	1.0 :: q1 → 1 q0
0.5 :: q1 → 0 q1	0.95 :: q1 → 0 q1
0.5 :: q1 →	1.0 :: q1 →
not_generated	0 :: q0 → 1 q0
not_generated	0 :: q1 → 1 q1
0.5 :: q1 → 0 q0	0.99 :: q1 → 0 q0
0.5 :: q0 → 0 q1	0.99 :: q0 → 0 q1
(a) MetaBayes$_{\text{SiLP}}$	(b) MilProbLog

Fig. 7. Probabilistic programs of learning FSA from $e^+ = 101011011$ and $e^- = 111101$.

Table 1 shows the negloglikelihoods of three systems. The closer to zero the better, since *likelihood* being 1 corresponds to *loglikelihood* being 0. Therefore MetaBayes has significantly better prediction (smaller negloglikelihood) than both MetaBayes$_{\text{SiLP}}$ and MilProbLog no matter what the value N_s is. Therefore Null hypothesis 2.1 and 2.2 are refuted. This is consistent with Bayesian learning theory that Bayesian prediction being optimal.

MetaBayes$_{\text{SiLP}}$ and MilProbLog both have significantly worse prediction than MetaBayes, but MetaBayes$_{\text{SiLP}}$ has increasing better prediction than MilProbLog with the increasing of N_s. This is consistent with the fact that the increasing of N_s leads to the exponential increase of hypothesis space and enlarges the difference between the search spaces of MetaBayes$_{\text{SiLP}}$ and MilProbLog. Since MetaBayes$_{\text{SiLP}}$ is able to constrain its search space to version space, while MilProbLog has to estimate the probabilities on all candidate clauses. According to Blumer bound, the larger search space leads to higher predictive error. Therefore Null hypothesis 2.3 is refuted. In terms of running time, MilProbLog took at least 10 times longer running time than that of MetaBayes$_{\text{SiLP}}$. Such results shows that MetaBayes$_{\text{SiLP}}$ has at least competitive likelihood to that of MilProbLog while having much shorter running time.

N_s	**MetaBayes**	**MetaBayes$_{\mathbf{SiLP}}$**	**MilProbLog**
2	856.70±19.03	1361.43±25.84	1317.14±7.66
3	976.64±5.76	1249.72±9.65	1323.38±1.66
4	820.60±10.20	1034.30±16.70	1306.40±2.58

Table 1. NegLogLikelihoods of learning FSAs

Learning language semantics Consider a task of validating a hypothesised food web using domain literature. For example, we might want to know whether the statement 'foxes eat rabbits' is supported by a piece of text from the literature. The challenge of this task lies in the richness of natural language. Specifically, there are many different ways to convey the same meaning. For instance, the sentences 'foxes are the predator of rabbits' express the same meaning as that of 'foxes eat rabbits'. In [5] such validation task was done manually by human beings. Human beings are capable of understanding the meaning of a text and extract relevant information from the text, but it is too time consuming to read through all literature. More importantly, human beings are capable of learning the meaning of text if encounter new words or phrases.

Materials and Methods In this experiment we consider learning semantics from texts, in particular, learning alternative phrases for expressing the same meaning. We use the representation of Definite Clause Translation Grammars (DCTG) [1] to allow both syntactic and semantic paring. Definite Clause Translation Grammars are triadic. It is similar to Definite Clause Grammars, but different in terms of a third argument for carrying the corresponding semantic. Figure 8 gives an example of DCTG which can parse the Text-Semantic pairs given in Figure 9. The target hypothesis of this experiment is a DCTG like the one in Figure 8. To learn such a grammar would require learning recursion and predicate invention. An invented predicates like s_2 can be interpreted as the set of phrases for express the meaning of 'eat', while s_1 and s_3 correspond to predator and prey, respectively.

```
s0(Text1,Text3,[M|Meaning]):- s1(Text1,Text2,M),s0(Text2,Text3,Meaning).
s0(Text1,Text3,[M|Meaning]):- s2(Text1,Text2,M),s0(Text2,Text3,Meaning).
s0(Text1,Text3,[M|Meaning]):- s3(Text1,Text2,M),s0(Text2,Text3,Meaning).
s0([Word|Text1],Text2,Meaning):- s0(Text1,Text2,Meaning).

s1([foxes|Text],Text,fox).
s1([fox|Text],Text,fox).
s2([predator,of|Text],Text,eats).
s2([eat|Text],Text,eats).
s3([rabbit|Text],Text,rabbit).
s3([rabbits|Text],Text,rabbit).
```

Fig. 8. Definite Clause Translation Grammars for parsing Text-Semantic pairs in Figure 9

Six positive examples were gathered from real texts. Another ten negative examples were derived from positive examples by removing words like 'eat' and 'fed'. Figure 10 shows part of the examples. There are only 16 examples in total,

```
s0([foxes, are, the, predator, of, rabbits],[],[fox,eats,rabbits]).
s0([foxes, eat, rabbits],[],[fox,eats,rabbits]).
```

Fig. 9. Artificial examples of Text-Semantic pairs

therefore we used leave-one-out cross-validation. We constrain the candidate clauses to the phrases with maximum length 2. MilProbLog was run with 10 iterations.

```
s0([in,the,laboratory,'Pollard(1968)',found,that,agonum,dorsale,would,climb,freely,
   and,fed,on,aphids,on,the,leaves,of,brussels,sprout,plants], [], [agonum,
   dorsale,eats,aphids]).
¬s0([in,the,laboratory,'Pollard(1968)',found,that,agonum,dorsale,would,climb,freely,
   and,fed,aphids,on,the,leaves,of,brussels,sprout,plants], [], [agonum,dorsale,eats,
   aphids]).
s0(['Dicker(1951)',noted,that,the,larvae,fed,on,the,strawberry,aphid,pentatrichopus,
   fragaefolii,cocker], [], [larvae,eats,aphid]).
s0([it,therefore,seems,likely,that,although,agonum,dorsale,will,eat,a,wide,range,of,
   food,',',aphids,are,preferred], [],
   [agonum,dorsale,eats,aphids]).
```

Fig. 10. Real-world examples of Text-Semantic pairs (subset of all sixteen examples)

Results and Discussion Figure 11 shows part of the probabilistic programs generated by MetaBayes$_{\text{SiLP}}$ and MilProbLog. It compares the probabilistic labels on the same clauses. Similar to that in Figure 7, there are clauses not considered by MetaBayes$_{\text{SiLP}}$, because they are either unnecessary for explaining the positive or inconsistent with the negative. For example, the hypothesised clause 's2([fed|Text],Text,eats)' would cover the negative example in Figure 10.

Table 2 compares the negloglikelihood of the three systems. Similar to the previous experiment, MetaBayes has the smallest negloglikelihood among the three systems, thus MetaBayes' predictor performs best. MetaBayes$_{\text{SiLP}}$ also performs significantly better than MilProbLog, as shown in Table 2. It is worth noting that the MetaBayes$_{\text{SiLP}}$'s negloglikelihood is significantly smaller than that of MilProbLog while taking much shorter running time. Therefore all Null hypotheses 2.1, 2.2 and 2.3 are refuted.

The reason that MetaBayes$_{\text{SiLP}}$ significantly outperform MilProbLog is the same as that in the previous experiment, that is, MetaBayes$_{\text{SiLP}}$ has much smaller search space than that of MilProbLog. For example, when given example-fold1 with 15 training examples, MilProbLog has 2179 clauses to be estimated, while there are only 46 for MetaBayes$_{\text{SiLP}}$.

5 Related work

According to Bayesian learning theory [7, 4], maximal predictive accuracy in learning is achieved by using a diversity of models, with predictions weighted

0.31 :: s2([fed,on|Text],Text,eats).	0.29 :: s2([fed,on|Text],Text,eats).
not_generated	0.79 :: s2([fed|Text],Text,eats).
not_generated	0.35 :: s2([on|Text],Text,eats).
0.31 :: s2([will,eat|Text],Text,eats).	0.47 :: s2([will,eat|Text],Text,eats).
(a) $\text{MetaBayes}_{\text{SiLP}}$	(b) MilProbLog

Fig. 11. Probabilistic programs of learning language semantic (partial)

	MetaBayes	**$\text{MetaBayes}_{\text{SiLP}}$**	**MilProbLog**
NegLogLikelihood	0	1.58	11.71

Table 2. NegLogLikelihoods of learning language semantic

according to the posterior probability of the corresponding hypothesis. In [11] error bounds for various Bayesian algorithms were analysed. The paper notes that while MAP maximises probability of exact identification of the target, it may have relatively high expected error. The paper goes on to show that the Gibbs algorithm, which randomly chooses a consistent hypothesis from the posterior distribution, has an error bound which is at most twice that of a Bayes's predictor (which is known to be optimal). These theoretical results are consistent with the experiments described in Section 4, and are also pertinent to a number of more *ad hoc* approaches to "model averaging" which have demonstrated significant predictive accuracy increases. These approaches are usually grouped under the title of *ensemble methods* and include *boosting* [9, 13] and *bagging* [6, 25]. Unlike the approach described in the present paper ensemble approaches use a more *ad hoc* approach to model-averaging, not based on an explicit Bayesian prior over the hypothesis space. However, the use of such a prior is directly comparable to the use of SLPs for sampling Bayes' nets investigated in [2]. The present paper extends this general approach to an ILP context, and demonstrates its predictive accuracy advantages in a context which supports the invention of relations and recursive programs. Within the ILP literature randomised search [24, 20] has been widely investigated. However, unlike the approaches described in this paper, these searches involve heuristic step-wise optimisation, rather than sampling and averaging predictions over a posterior distribution. The related areas of Probabilistic ILP (PILP) [22] and Statistical Relation Learning (SRL) [10] involve combining Bayesian inference and ILP, though this is in the context of Probabilistic Logic representations. The treatment of a set of meta-rules as an SLP is akin to this, though the logical reasoning is necessarily in terms of higher-order clauses rather than the probabilistic first-order representations used in PILP and SRL.

6 Conclusion and further work

This paper extends previous work on Meta-Interpretive Learning [19, 18] by demonstrating that a Bayesian prior can be implemented as a meta-interpreter over a stochastic logic program consisting of higher-order meta-rules. We use this approach to suggest a method for carrying out simultaneous structure and parameter estimation for a form of ProbLog program which we refer to as SiLPs.

The approach supports sampling of hypotheses consistent with a given set of examples and background knowledge, and has been used to implement approximated Bayes' prediction in a system called MetaBayes. Similarly we implement a Bayes' MAP algorithm together with one caled MetaBayes$_{\text{SiLP}}$ which estimates structure and parameters of SiLPs. Our experiments indicate that approximated Bayes' prediction significantly outperform MAP on binary prediction tasks involving FSAs and prediction of ancestor relationships in the Russian Royal family dataset. The results are in line with theoretical predictions. Furthermore on probabilistic prediction our MetaBayes outperforms MetaBayes$_{\text{SiLP}}$ which in turn outperforms Problog on negative log likelihood prediction on both the FSA dataset and a natural language task involving ranking of probabilistic predictions.

Further work will address efficiency improvements in the algorithms as well as extensions to handle classification noise in the data and the use of Bayesian inference in active learning.

Acknowledgements

The authors would like to acknowledge the support of Syngenta in its funding of the University Innovations Centre at Imperial College. The first author would like to thank the Royal Academy of Engineering and Syngenta for funding his present 5 years Research Chair.

References

1. Harvey Abramson. Definite clause translation grammars. Technical report, Vancouver, BC, Canada, Canada, 1984.
2. N. Angelopoulos and J. Cussens. Markov chain Monte Carlo using tree-based priors on model structure. In *UAI-2001*, Los Altos, CA, 2001. Kaufmann.
3. A. Arvanitis, S.H. Muggleton, J. Chen, and H. Watanabe. Abduction with stochastic logic programs based on a possible worlds semantics. In *Short Paper Proceedings of the 16th International Conference on Inductive Logic Programming*. University of Corunna, 2006.
4. J.M. Bernardo and A.F.M Smith. *Bayesian theory*. Wiley, New York, 1994.
5. D.A. Bohan, G. Caron-Lormier, S.H. Muggleton, A. Raybould, and A. Tamaddoni-Nezhad. Automated discovery of food webs from ecological data using logic-based machine learning. *PloS ONE*, 6(12), 2011.
6. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
7. W. Buntine. *A Theory of Learning Classification Rules*. PhD thesis, School of Computing Science, University of Technology, Sydney, 1990.
8. L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A probabilistic prolog and its applications in link discovery. In R. Lopez de Mantaras and M.M Veloso, editors, *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 804–809, 2007.
9. Y. Freund and R. Shapire. A decision theoretic generalisation of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.
10. L. Getoor. Tutorial on statistical relational learning. In Stefan Kramer and Bernhard Pfahringer, editors, *Proceedings of the 15th International Conference on Inductive Logic Programming*, volume 3625, page 415, 2005.

11. D. Haussler, M Kearns, and R. Shapire. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. *Machine Learning Journal*, 14(1):83–113, 1994.
12. K. Kersting and L. De Raedt. Towards combining inductive logic programming with bayesian networks. In *Proceedings of the Eleventh International Conference on Inductive Logic Programming*, LNAI 2157, pages 118–131, Berlin, 2001. Springer-Verlag.
13. H. Lodhi and S.H. Muggleton. Modelling metabolic pathways using stochastic logic programs-based ensemble methods. In *Proceedings of the 2nd International Conference on Computational Methods in System Biology*. Springer-Verlag, 2004.
14. T. Mantadelis and G. Janssens. Nesting probabilistic inference. In *Proceedings of the International Colloquium on Implementation of Constraint and LOgic Programming Systems (CICLOPS)*, pages 1–16, Lexington, Kentucky, 2011. Springer-Verlag.
15. S.H. Muggleton. Stochastic logic programs. In L. de Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.
16. S.H. Muggleton. Stochastic logic programs. *Journal of Logic Programming*, 2001. Accepted subject to revision.
17. S.H. Muggleton. Learning structure and parameters of stochastic logic programs. *Electronic Transactions in Artificial Intelligence*, 6, 2002.
18. S.H. Muggleton and D. Lin. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. In *Proceedings of the 23rd International Joint Conference Artificial Intelligence (IJCAI 2013)*, pages 1551–1557, 2013.
19. S.H. Muggleton, D. Lin, N. Pahlavi, and A. Tamaddoni-Nezhad. Meta-interpretive learning: application to grammatical inference. *Machine Learning*, 94:25–49, 2014.
20. S.H. Muggleton and A. Tamaddoni-Nezhad. QG/GA: A stochastic search for Progol. *Machine Learning*, 70(2–3):123–133, 2007. DOI: 10.1007/s10994-007-5029-3.
21. N. Pahlavi and S.H. Muggleton. Towards efficient higher-order logic learning in a first-order datalog framework. In *Latest Advances in Inductive Logic Programming*. Imperial College Press, 2012. In Press.
22. L. De Raedt, P. Frasconi, K. Kersting, and S.H. Muggleton, editors. *Probabilistic Inductive Logic Programming*. Springer-Verlag, Berlin, 2008. LNAI 4911.
23. A. Tamaddoni-Nezhad and S.H. Muggleton. Stochastic refinement. In *Proceedings of the 20th International Conference on Inductive Logic Programming*, pages 222–237. Springer-Verlag, 2011.
24. F. Zelezny, A. Srinivasan, and D. Page. Lattice-search runtime distributions may be heavy-tailed. In S. Matwin and C. Sammut, editors, *Proceedings of the 12th International Conference on Inductive Logic Programming*, volume 2583, pages 333–345, 2003.
25. J. Zhu, H. Zou, S. Rosset, and T. Hastie. Multi-class adaboost. *Statistics and its Interface*, 2:349–360, 2009.