A General MCMC Method for Bayesian Inference in Logic-Based Probabilistic Modeling

Taisuke Sato
Tokyo Institute of Technology
2-12-1 Ôokayama Meguro-ku Tokyo Japan 152-8552

Abstract

We propose a general MCMC method for Bayesian inference in logic-based probabilistic modeling. It covers a broad class of generative models including Bayesian networks and PCFGs. The idea is to generalize an MCMC method for PCFGs to the one for a Turing-complete probabilistic modeling language PRISM in the context of statistical abduction where parse trees are replaced with explanations. We describe how to estimate the marginal probability of data from MCMC samples and how to perform Bayesian Viterbi inference using an example of Naive Bayes model augmented with a hidden variable.

1 Introduction

MCMC (Markov chain Monte Carlo) methods have been widely used in Bayesian inference. They are the key technology being used for complex probabilistic models dealing with high-dimensional data and there exist well-known MCMC tools such as WinBugs [Lunn *et al.*, 2000]. However, mostly, they are designed for continuous models and there are not many general-purpose MCMC methods for Bayesian inference applicable to complex discrete models.

In this paper, we present such an MCMC method for PRISM, a Turing-complete logic-based probabilistic modeling language [Sato and Kameya, 2001]. It covers a broad class of generative models including BNs (Bayesian networks) and PCFGs (probabilistic context free grammars), thereby eliminating the need of inventing and implementing a new MCMC method for each model. We have only to declaratively describe a probabilistic model by a set of definite clauses. The marginal and posterior probability is automatically estimated from MCMC samples generated by the method integrated into PRISM. Note that models can be arbitrarily complex as long as they are described by PRISM programs. Thus Bayesian inference is possible at no extra cost even if they go beyond traditional BNs and PCFGs. Later we report Bayesian inference for a PLCG (probabilistic left-corner grammar) which has not been attempted yet to our knowledge.

The motivation behind our proposal is to develop an MCMC method for an expressive declarative language. Declarativeness is crucial. It abstracts away low level details such as control and memory management and keeps readability of complex models. In addition it separates models from their inference and learning methods, and conversely, allows us to combine them freely. PRISM uses logic programs to specify models and the user can choose the best one from a variety inference and learning methods such as MLE (maximum likelihood estimation), MAP (maximum a posterior) inference, VB (variational Bayes), Viterbi inference and now MCMC and associated Bayesian Viterbi inference.

There already exist MCMC based probabilistic modeling languages [Milch *et al.*, 2005; McCallum *et al.*, 2009; Goodman *et al.*, 2008]. Unlike these languages however, our MCMC is logic-based and furthermore not the only inference method but the latest one added to PRISM that works on top of PRISM's rich functionalities such as exact inference and VB learning.

2 PRISM for statistical abduction

PRISM is a language for statistical abduction. We here review the relationship between abduction and PRISM for self-containedness. Abduction is a form of logical inference which infers the best explanation E for a given observation G such that $DB, E \vdash G$ where DB is a logic program consisting of definite clauses and $DB \wedge E$ is consistent. Usually E is a conjunction of ground atoms called *abducible*s. When there are multiple explanations $\mathcal{E} = \{E_1, \ldots, E_N\}$ for G, all equally entailing G, we have to select the best one. One possible way is to assign probabilities to abducibles and choose $E^* = \mathrm{argmax}_{E \in \mathcal{E}} P(E \mid DB)$ that has the highest probability as the best explanation for G. We assume here $P(E \mid DB)$ is computable from the probabilities assigned to abducibles which we subsequently call *parameter*s. However the real problem starts here: how to obtain these parameters? Oftentimes there are thousands of parameters. Statistical abduction solves this problem by statistically learning them from data by applying a parameter learning algorithm such as the EM algorithm to logical formulas made up of abducibles.

PRISM is a probabilistic extension of Prolog for statistical abduction [Sato and Kameya, 1997; 2001]. It provides

msw[1] atoms as probabilistic abducibles to simulate basic probabilistic events such as dice throwing, together with various types of parameter learning procedures. An *explanation* in PRISM, denoted by ϵ, is a conjunction of msw atoms. When a goal G is given to a program DB, PRISM searches for all explanations for G w.r.t. DB by top-down tabled search. The resulting explanations are organized as a compact set of AND/OR formulas called an *explanation graph* so that common subformulas are shared. This sharing mechanism makes possible efficient probability computation of G by dynamic programming applied to the generated graph. Every inference and learning procedure in PRISM works for it and the user does not need to modify the program when he or she changes to another inference and learning mode.

Here is a simple PRISM program for a PCFG that has two rules S → SS and S → a.

```
values(s,[[s,s],[a]],set@[0.4,0.6]).

s(X,Z) :-
  msw(s,RHS),
  ( RHS = [s,s] -> s(X,Y),s(Y,Z)
  ; RHS = [a] -> X = [a|Z] ; true ).
```

Figure 1: A PCFG program for S → SS | a

values(s, [[s, s], [a]], set@[0.4, 0.6]) is a declaration which introduces two probabilistic abducibles msw(s, [s, s]) and msw(s, [a]) with parameters 0.4 and 0.6 respectively. They represent a probabilistic choice of a CFG rule in {S → SS, S → a}. In the program s(X, Z) means s spans a difference list X−Z. msw(s, RHS) returns RHS = [s, s] with probability 0.4 and [a] with probability 0.6. Figure 2 depicts part of the explanation graph for the goal s([a, a, a], []) representing a sentence "aaa". The goal has two proofs corresponding to the two disjuncts in Figure 2. They share the same subgoal s([a, a], [a]). All explanations for s([a, a, a], []) are found by transforming it into DNF using these if-and-only-if formulas.

$$
\begin{aligned}
&\mathtt{s}([\mathtt{a},\mathtt{a},\mathtt{a}],[]) \Leftrightarrow \\
&\qquad (\mathtt{s}([\mathtt{a},\mathtt{a},\mathtt{a}],[\mathtt{a},\mathtt{a}]) \wedge \mathtt{s}([\mathtt{a},\mathtt{a}],[]) \wedge \mathtt{msw}(\mathtt{s},[\mathtt{s},\mathtt{s}])) \\
&\quad \vee\ (\mathtt{s}([\mathtt{a},\mathtt{a},\mathtt{a}],[\mathtt{a}]) \wedge \mathtt{s}([\mathtt{a}],[]) \wedge \mathtt{msw}(\mathtt{s},[\mathtt{s},\mathtt{s}])) \\
&\mathtt{s}([\mathtt{a},\mathtt{a},\mathtt{a}],[\mathtt{a},\mathtt{a}]) \Leftrightarrow \mathtt{msw}(\mathtt{s},[\mathtt{a}]) \\
&\quad \mathtt{s}([\mathtt{a},\mathtt{a}],[]) \Leftrightarrow \mathtt{s}([\mathtt{a},\mathtt{a}],[\mathtt{a}]) \wedge \mathtt{s}([\mathtt{a}],[]) \wedge \mathtt{msw}(\mathtt{s},[\mathtt{s},\mathtt{s}]) \\
&\qquad \cdots
\end{aligned}
$$

Figure 2: part of the explanation graph for s([a, a, a], [])

In general, a discrete random variable X_i taking a value in $V_i = \{v_1, \ldots, v_N\}$ is represented in PRISM by a family i of ground msw atoms $\{\mathtt{msw}(i, v) \mid v \in V_i\}$. msw$(i, v)$ becomes exclusively true with probability (parameter) $\theta_{i,v}$ ($\sum_{v \in V_i} \theta_{i,v} = 1$). Let DB be a program and $\boldsymbol{\theta}$ the set of all parameters in DB. In terms of the distribution semantics of PRISM [Sato and Kameya, 2001], $P_{DB}(G \mid \boldsymbol{\theta})$, the probability of a goal G w.r.t. DB, is calculated as a sum-product of parameters:

$$
\begin{aligned}
P_{DB}(G \mid \boldsymbol{\theta}) &= \sum_{\epsilon \in \mathrm{expl}(G)} P_{DB}(\epsilon \mid \boldsymbol{\theta}) \\
P_{DB}(\epsilon \mid \boldsymbol{\theta}) &= \prod_{(i,v):\mathtt{msw}(i,v) \in \epsilon} {\theta_{i,v}}^{\sigma_{i,v}(\epsilon)}
\end{aligned}
$$

where ϵ is an explanation for G such that $\epsilon, DB \vdash V, \mathrm{expl}(G)$ the set of all explanations for G and $\sigma_{i,v}(\epsilon)$ the number of times msw(i, v) appears in ϵ. Conversely it is also possible to estimate $\boldsymbol{\theta}$ from an iid sample given as a list of goals like [s([a, a, a], []), s([a], []), ...].

3 Bayesian PRISM

We now introduce priors over parameters and make Bayesian inference. Let $\boldsymbol{\theta}_i = \theta_{i,v_1}, \ldots, \theta_{i,v_N}$ be parameters associated with a family of msw atoms $\mathtt{msw}(i, v_1), \ldots, \mathtt{msw}(i, v_N)$ and $\boldsymbol{\theta}$ the set of all parameters in a program DB. As usual we put a Dirichlet distribution $P_{\mathrm{Dir}}(\boldsymbol{\theta}_i \mid \alpha_{i,v_1}, \ldots, \alpha_{i,v_N}) \propto {\theta_{i,v_1}}^{\alpha_{i,v_1}-1} \cdots {\theta_{i,v_N}}^{\alpha_{i,v_N}-1}$ over $\boldsymbol{\theta}_i$. $\boldsymbol{\alpha}_i = \alpha_{i,v_1}, \ldots, \alpha_{i,v_N}$ are called the *hyper parameter*s for $\boldsymbol{\theta}_i$. We use $\boldsymbol{\alpha}$ for the set of all hyper parameters in DB.

Suppose an iid sample of goals $\mathcal{G} = G_1, \ldots, G_T$ is given. Let ϵ_t denote an explanation for G_t $(1 \leq t \leq T)$. Put $\mathcal{E} = \epsilon_1, \ldots, \epsilon_T$ and $\sigma_{i,v}(\mathcal{E}) = \sum_{t=1}^{T} \sigma_{i,v}(\epsilon_t)$. Then the marginal likelihood of $\mathcal{E}$ and that of $\mathcal{G}$ are respectively calculated as

$$
\begin{aligned}
&P_{DB}(\mathcal{E} \mid \boldsymbol{\alpha}) \\
&\quad = \int \left(\prod_{t=1}^{T} P_{DB}(\epsilon_t \mid \boldsymbol{\theta}) \right) \prod_i P_{\mathrm{Dir}}(\boldsymbol{\theta}_i \mid \boldsymbol{\alpha}_i)\, d\boldsymbol{\theta}_i \\
&\quad = \prod_i Z_i^{-1} \left(\frac{\prod_{v \in V_i} \Gamma(\sigma_{i,v}(\mathcal{E}) + \alpha_{i,v})}{\Gamma(\sum_{v \in V_i} \sigma_{i,v}(\mathcal{E}) + \alpha_{i,v})} \right) \qquad (1) \\
&\qquad \text{where } Z_i = \frac{\prod_{v \in V_i} \Gamma(\alpha_{i,v})}{\Gamma(\sum_{v \in V_i} \alpha_{i,v})} \\
&P_{DB}(\mathcal{G} \mid \boldsymbol{\alpha}) \\
&\quad = \sum_{\mathcal{E} \in \mathrm{expl}(G_1) \times \cdots \times \mathrm{expl}(G_T)} P_{DB}(\mathcal{E} \mid \boldsymbol{\alpha}) \qquad (2)
\end{aligned}
$$

where $\Gamma(\cdot)$ is the gamma function. It is immediate from the equation (2) that exact computation of $P_{DB}(\mathcal{G} \mid \boldsymbol{\alpha})$ is intractable due to the summation of combinatorially many $\mathcal{E}$s. We avoid this difficulty by switching to MCMC and perform approximate Bayesian inference.

Bayesian inference treated in this paper is the computation of the marginal likelihood $P_{DB}(\mathcal{G} \mid \boldsymbol{\alpha})$ which is useful for model selection and *Bayesian Viterbi inference* that computes the *Bayesian Viterbi explanation* $\epsilon^*_{\mathrm{new}} = \mathrm{argmax}_{\epsilon_{\mathrm{new}} \in \mathrm{expl}(G_{\mathrm{new}})} P_{DB}(\epsilon_{\mathrm{new}} \mid G_{\mathrm{new}}, \mathcal{G}, \boldsymbol{\alpha})$ for a new goal G_{new}. The latter is applied to classification problems. We notice the following equations.

[1] msw is a short hand for "multi-valued switch."

$$P_{DB}(\mathcal{G} \mid \boldsymbol{\alpha})$$
$$= \frac{P_{\mathrm{Dir}}(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) P_{DB}(\mathcal{G} \mid \boldsymbol{\theta})}{\sum_{\mathcal{E}} P_{DB}(\boldsymbol{\theta} \mid \mathcal{E}, \boldsymbol{\alpha}) P_{DB}(\mathcal{E} \mid \mathcal{G}, \boldsymbol{\alpha})} \quad (3)$$
$$\mathrm{argmax}_{\epsilon_{\mathrm{new}} \in \mathrm{expl}(G_{\mathrm{new}})} P_{DB}(\epsilon_{\mathrm{new}} \mid G_{\mathrm{new}}, \mathcal{G}, \boldsymbol{\alpha})$$
$$= \mathrm{argmax}_{\epsilon_{\mathrm{new}} \in \mathrm{expl}(G_{\mathrm{new}})} P_{DB}(\epsilon_{\mathrm{new}} \mid \mathcal{G}, \boldsymbol{\alpha}) \quad (4)$$
$$P_{DB}(\epsilon_{\mathrm{new}} \mid \mathcal{G}, \boldsymbol{\alpha})$$
$$= \sum_{\mathcal{E}} P_{DB}(\epsilon_{\mathrm{new}} \mid \mathcal{E}, \boldsymbol{\alpha}) P_{DB}(\mathcal{E} \mid \mathcal{G}, \boldsymbol{\alpha}) \quad (5)$$

Here $\boldsymbol{\theta}$ in (3) is arbitrary. $P_{DB}(\boldsymbol{\theta} \mid \mathcal{E}, \boldsymbol{\alpha})$ is the posterior distribution of $P_{\mathrm{Dir}}(\boldsymbol{\theta} \mid \boldsymbol{\alpha})$ given $\mathcal{E}$ and $\mathcal{E}$ ranges over $\mathrm{expl}(G_1) \times \cdots \times \mathrm{expl}(G_T)$. (3), (4) and (5) assure that, as we show later, we can approximately compute the marginal likelihood of $\mathcal{G}$ and the Bayesian Viterbi explanation if we get enough samples from $P_{DB}(\mathcal{E} \mid \mathcal{G}, \boldsymbol{\alpha})$.

We sample $\mathcal{E} \sim P_{DB}(\mathcal{E} \mid \mathcal{G}, \boldsymbol{\alpha})$ by MCMC following the work done by Johnson et al. [Johnson *et al.*, 2007]. They construct a Markov chain for Bayesian PCFGs by M-H (Metropolis-Hastings) sampling whose state is a vector of parse trees for each sentence. A sate transition occurs by randomly choosing a sentence and by replacing according to the M-H acceptance probability its parse tree in the current state with a new one that is sampled from the set of parse trees of the sentence using inside probabilities. The M-H sampling algorithm for PRISM is obtained by generalizing sentences to goals and parse trees to proof trees, or equivalently to explanations for goals (because proof trees are recoverable from explanations).

Let $\mathcal{G} = G_1, \ldots, G_T$ be observed goals and $\mathcal{E}^{(k)} = \epsilon_1^{(k)}, \ldots, \epsilon_T^{(k)}$ their corresponding explanations constituting a state at step k. We denote by $\mathcal{E}_{-t}^{(k)}$ $\mathcal{E}^{(k)}$ with $\epsilon_t^{(k)}$ deleted and put $\bar{\boldsymbol{\theta}}_{-t}^{(k)} = \mathbf{E}[\boldsymbol{\theta} \mid \mathcal{E}_{-t}^{(k)}, \boldsymbol{\alpha}]$ which is the average of parameters by the posterior conditioned on $\mathcal{E}_{-t}^{(k)}$. We construct a Markov chain over $\mathcal{E}$s by M-H sampling as follows.

[Initialize] Set $\mathcal{E}^{(0)} = \epsilon_1^{(0)}, \ldots, \epsilon_T^{(0)}$ for appropriate $\epsilon_t^{(0)}$s.
[Repeat] for $k = 0, 1, \ldots$
Let $\mathcal{E}^{(k)} = \epsilon_1^{(k)}, \ldots, \epsilon_T^{(k)}$ be the current state.
Choose randomly t in $[1, T]$.
Sample a new explanation $\epsilon_t' \sim P_{DB}(\epsilon_t' \mid G_t, \bar{\boldsymbol{\theta}}_{-t}^{(k)})$.
Let $\mathcal{E}'$ be $\mathcal{E}^{(k)}$ with $\epsilon_t^{(k)}$ replaced by ϵ_t'.
Compute the acceptance probability

$$A = \min\left(1, \frac{P_{DB}(\mathcal{E} \mid \mathcal{E}', \alpha) P_{DB}(\mathcal{E}' \mid \mathcal{G}, \bar{\theta}_{-t}^{(k)}, \alpha)}{P_{DB}(\mathcal{E}' \mid \mathcal{E}, \alpha) P_{DB}(\mathcal{E} \mid \mathcal{G}, \bar{\theta}_{-t}^{(k)}, \alpha)}\right).$$

Move to a new state $\mathcal{E}^{(k+1)} = \mathcal{E}'$ with probability A
else $\mathcal{E}^{(k+1)} = \mathcal{E}^{(k)}$.

Figure 3: M-H sampling for PRISM

Sampling $\epsilon_t' \sim P_{DB}(\epsilon_t' \mid G_t, \bar{\boldsymbol{\theta}}_{-t}^{(k)})$ is carried out exactly as in [Johnson *et al.*, 2007] but we use generalized inside probabilities in PRISM [Sato and Kameya, 2001] instead of inside probabilities for PCFGs (details omitted).

As the initial state, we select (non-Bayesian) Viterbi explanations. Namely we put $\epsilon_t^{(0)} = \mathrm{argmax}_{\epsilon_t \in \mathrm{expl}(G_t)} P_{DB}(\epsilon_t \mid G_t, \bar{\boldsymbol{\theta}}_{\mathrm{VB}})$ $(1 \leq t \leq T)$ where $\bar{\boldsymbol{\theta}}_{\mathrm{VB}}$ is the posterior mean of $\boldsymbol{\theta}$ by the approximate posterior distribution of $P_{DB}(\boldsymbol{\theta} \mid \mathcal{G})$ computed by the VB routine in PRISM.

4 Estimating marginal likelihood

In this section we estimate the marginal likelihood of real and artificial data.

It follows from (3) in Section 3 that the marginal likelihood is computed in terms of three quantities appearing in (3), i.e. $P_{\mathrm{Dir}}(\boldsymbol{\theta} \mid \boldsymbol{\alpha})$, $P_{DB}(\mathcal{G} \mid \boldsymbol{\theta})$ in the enumerator and the denominator. The first two are easy to compute. The third one, the denominator, is also not difficult to deal with. We can approximate it using samples $\mathcal{E}_1, \ldots, \mathcal{E}_K$ from $P_{DB}(\mathcal{E} \mid \mathcal{G}, \boldsymbol{\alpha})$ generated by the proposed MCMC method as follows.

$$\sum_{\mathcal{E}} P_{DB}(\boldsymbol{\theta} \mid \mathcal{E}, \boldsymbol{\alpha}) P_{DB}(\mathcal{E} \mid \mathcal{G}, \boldsymbol{\alpha}) \approx \frac{\sum_{m=1}^{K} P_{DB}(\boldsymbol{\theta} \mid \mathcal{E}_m, \boldsymbol{\alpha})}{K}$$

With a necessary tool being prepared, we estimate the marginal likelihood of real data. Hereafter we use ML as a shorthand for "marginal likelihood" to save space. We put $\boldsymbol{\theta} = \bar{\boldsymbol{\theta}}_{\mathrm{VB}}$ in (3) and choose as a dataset the ATR corpus [Uratani *et al.*, 1994] which contains 10,995 Japanese conversational sentences. There is a manually developed CFG containing 860 rules. Using this CFG, we conduct two experiments of estimating the ML, one for a PCFG model and the other for a PLCG model[2]. Due to space limitations however, we only report the latter because there is no literature on estimating the ML of PLCG model as far as we know.

A PRISM program for the PLCG model is comprised of 2,500 non-unit clauses and 21,000 unit clauses. Considering computational difficulty caused by such a large model, at each #Sentence, i.e. the number of given sentences, we run the MCMC algorithm for 1,000 iterations and use samples from the last 100 iterations to estimate the log ML of the ATR corpus. We repeat this process 10 times and take the average. The result is shown in Table 1, together with the average (of 10 times trials) of log VFE (variational free energy) computed by the VB routine in PRISM.

[2] A PLCG constructs a parse tree in a bottom-up manner. Parsing conflicts such as shift-reduce are resolved by a probabilistic choice.

#Sentence	Log ML		Log VFE	
	ave.	std.	ave.	std.
10	-627.27	5.77	-642.56	0.87
100	-4597.84	27.79	-4738.51	3.44
1000	-37249.42	29.59	-37320.85	26.08
5000	-176296.93	63.98	-175999.83	74.90
10000	-333198.23	23.44	-333056.56	105.22

Table 1: Average log marginal likelihood and log variational free energy of the ATR corpus by the PLCG model

It is seen from Table 1 that ML and VFE are very close though ML is always higher than VFE (this is convincing because VFE is a lower bound of ML). Also the effect of introducing prior is visible, witnessed by the non-linearity of MLs w.r.t. #Sentence. I.e. when #Sentence is small, the log ML is offset more positively[3].

Estimating ML is indispensable in Bayesian model selection. We next take up a profile-HMM and examine how the ML of data changes with the size of the HMM. A profile-HMM is a variant of HMM designed for multiple alignment of biological sequences. As illustrated in Figure 4, it has a base HMM (bottom layer) which is augmented with insert and del(-ete) states to simulate probabilistic changes in evolution. The *model length*, i.e. the number of states in the base HMM, must be tuned when aligning biological sequences and from a Bayesian viewpoint, it should maximizes the ML of the sequences.

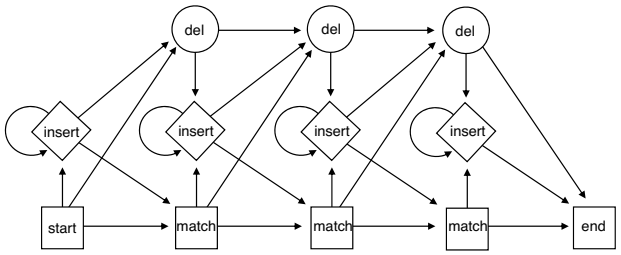

Figure 4: Profile-HMM

Using Rose [Stoye *et al.*, 1998] with true length set to 10, we randomly generate artificial protein sequences of size 100. They have varying length and the 64% of the sequences have length 10. We estimate the log ML of the data by running a PRISM program for a profile-HMM while changing the model length from 2 to 25. At each length, we run the MCMC algorithm for $6,000$ iterations and use the last $1,000$ samples to estimate the log ML and repeat this process 10 times.

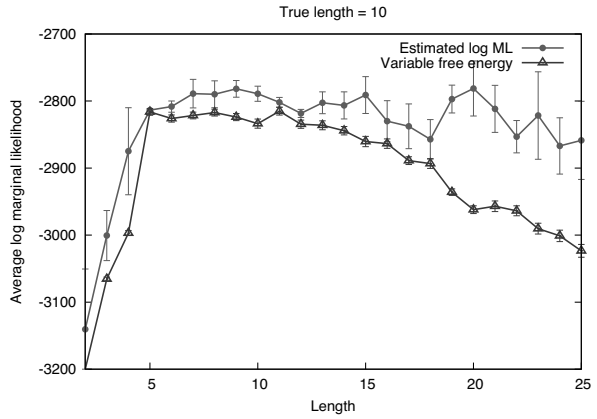

Figure 5: Average log ML vs. model length

[3]The same tendency is observed with the PCFG model but to a less degree.

We plot a curve of average (10 times trials) of log ML and that of log VFE w.r.t. model length as shown in Figure 5. As in the PLCG case, ML is higher than VFE and has a larger standard deviation at each data-point. It draws a rather ragged line and decreases more gently than VFE. However both lines have a peak near Length=10, the true model length.

5 Viterbi inference and classification

In this section, we deal with Viterbi inference that computes the Bayesian Viterbi explanation. We apply it to a variant of Naive Bayes model that has a hidden class variable HC as illustrated in Figure 6. The new Naive Bayes model is referred to as NBH hereafter. The idea of HC is to make attributes interdependent through HC thus relaxing the independence assumption of Naive Bayes. Similar improvement is reported in [Zhang *et al.*, 2005] but NBH is much simpler.

When building an NBH model, we have to decide $\#HC$, the number of hidden classes. This is a model selection problem and we can solve it by estimating the log ML of data w.r.t. the NBH model varying $\#HC$.

In the following, we first explain how Bayesian Viterbi explanation is computed in PRISM and then using the NBH model for a dataset in the UCI repository, we show how it is applied to classification problems.

5.1 Reranking

Suppose we have observed goals $\mathcal{G}$ and a new goal $G_{\rm new}$ is given. We wish to know the Bayesian Viterbi explanation $\epsilon^*_{\rm new}$ for $G_{\rm new}$ computed by the equations (4) and (5). Just like the case of estimating the ML, we can approximate it by using MCMC samples $\mathcal{E}_1,\ldots,\mathcal{E}_K$ from $P_{DB}(\mathcal{E} \mid \mathcal{G},\boldsymbol{\alpha})$ as follows.

$$\epsilon^*_{\rm new} = \mathrm{argmax}_{\epsilon_{\rm new}\in \mathrm{expl}(G_{\rm new})} P_{DB}(\epsilon_{\rm new} \mid \mathcal{G},\boldsymbol{\alpha}) \quad (6)$$

$$P_{DB}(\epsilon_{\rm new} \mid \mathcal{G},\boldsymbol{\alpha}) \approx \frac{\sum_{m=1}^{K} P_{DB}(\epsilon_{\rm new} \mid \mathcal{E}_m,\boldsymbol{\alpha})}{K} \quad (7)$$

$P_{DB}(\epsilon_{\rm new} \mid \mathcal{E}_m,\boldsymbol{\alpha})$ in (7) is easy to compute as it has a closed form but unfortunately (6) requires the computation of $P_{DB}(\epsilon_{\rm new} \mid \mathcal{G},\boldsymbol{\alpha})$ for every explanation in $\mathrm{expl}(G_{\rm new})$, which is virtually impossible except small problems. To make this computation feasible we take a reranking approach.

In our reranking approach, we use (6) but replace $\mathrm{expl}(G_{\rm new})$ with its small subset $\mathcal{S}$ called a *candidate set* that possibly contains $\epsilon^*_{\rm new}$. The choice of $\mathcal{S}$ is critical because it needs to be likely to contain $\epsilon^*_{\rm new}$ but must not be too big.

Expecting non-Bayesian Viterbi explanations for $G_{\rm new}$ and their Bayesian counterparts overlap enough, we choose as $\mathcal{S}$ the top M explanations $\epsilon_1,\ldots,\epsilon_M$ $(\in \mathrm{expl}(G_{\rm new}))$ in the descending order of $P_{DB}(\epsilon \mid \bar{\boldsymbol{\theta}}_{\rm VB},\mathcal{E}_m,\boldsymbol{\alpha})$. Here $\bar{\boldsymbol{\theta}}_{\rm VB}$ is the approximate posterior mean of $\boldsymbol{\theta}$ as explained before.

When $G_{\rm new}$ is non-ground and contains free variables as in the next subsection, we consider all possible ground instantiations and all their possible explanations. We order them as in the ground case and take the top M explanations as the candidate set to compute $\epsilon^*_{\rm new}$.

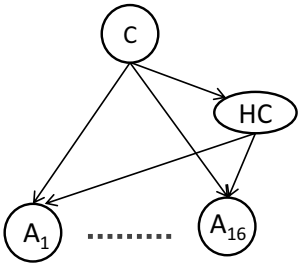

Figure 6: Bayesian network for an NBH model of the Voting Record dataset

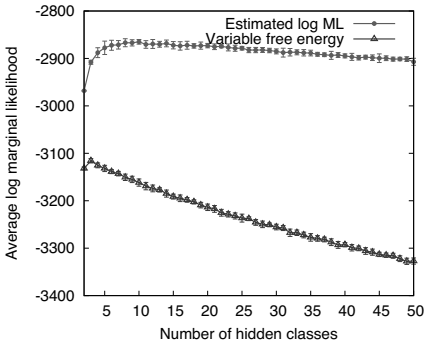

Figure 7: Average log marginal likelihood

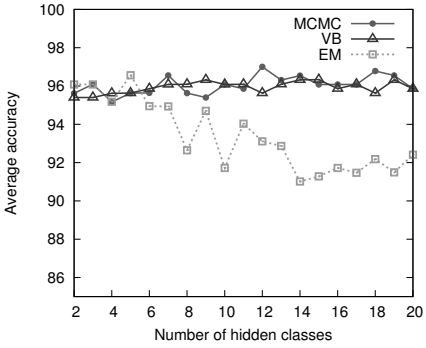

Figure 8: Average accuracy

5.2 Classification by the NBH model

We here apply Bayesian Viterbi inference to an NBH model to solve a classification problem of the the Voting Record dataset in the UCI repository. The dataset contains 435 tuples and each tuple has 16 binary attributes (yes, no) and a class (democrat, republican). Some attributes are missing. The size of the candidate set is set to 2.

Consider an NBH model in Figure 6. A PRISM program $DB_{\rm nbh}$ for the NBH model defines a predicate $\texttt{nbh}(\texttt{C},\texttt{As})$ that represents our observation where As is a vector (list) of 16 attributes and C denotes a class to which As belongs. Each tuple in the dataset is represented by a ground goal such as $\texttt{nbh}(\texttt{democrat},[\texttt{yes},\ldots,\texttt{no}])$.

Suppose a set of ground nbh atoms $\mathcal{G}'$ is given as observations and we would like to predict the class of new data $[\texttt{a}_1,\ldots,\texttt{a}_{16}]$ based on $\mathcal{G}'$. To do so, we put $G_{\rm nbh} = \texttt{nbh}(\texttt{C},[\texttt{a}_1,\ldots,\texttt{a}_{16}])$ (C is a variable and the $\texttt{a}_i$s are constant) and obtain the Bayesian Viterbi explanation $\epsilon^*_{\rm nbh}$ for $G_{\rm nbh}$ using $\mathcal{G}'$ as explained in the previous subsection. Since $\epsilon^*_{\rm nbh}$ uniquely entails a ground instantiation of $G_{\rm nbh}$, it uniquely determines the ground instantiation of C as well, either to democrat or to repbulican as the predicted class of the data $[\texttt{a}_1,\ldots,\texttt{a}_{16}]$. Thus we can solve the problem of classifying voting records into democrat or republican by the NBH model.

We draw curves for the average log ML and the average log VFE of the Voting Record dataset (20 times trials at each data point) varying the number of hidden classes #HC as in Section 4. We run the MCMC algorithm for 2,000 iterations and use samples from the last 1,000 iterations to estimate the log ML. They are shown in Figure 7. What is noticeable here is that the plotted curves are much smoother than those for the profile-HMM in Section 5. Intuitively this is because the NBH is much simpler and has much less parameters than the profile-HMM but a detailed analysis is left for future research. We also notice that while the log VFE curve has a global acute peak at #HC=3, the log ML curve has a very broad peak around #HC=12. As #HC increases, the gap between the two monotonically widens, which might suggest the quality of approximation by VB monotonically deteriorates.

Finally we compare classification accuracy of three classification methods applied to the NBH model, each temporarily referred to as MCMC, VB and EM respectively. MCMC means the classification by Bayesian Viterbi inference. VB means usual (non-Bayesian) Viterbi inference which uses the posterior mean $\bar{\boldsymbol{\theta}}_{\rm VB}$ to compute $\mathrm{argmax}_{\texttt{C}} P_{DB_{\rm nbh}}(\texttt{nbh}(\texttt{C},\texttt{As})) \mid \bar{\boldsymbol{\theta}}_{\rm VB})$ to decide the class C. EM is similar to VB but uses parameters learned by the EM algorithm. We vary #HC between 2 and 20 and at each #HC, we repeat 10-fold CV (cross-validation) 10 times to take the average of classification accuracy for each method. As a base line, we measure the average accuracy of Naive Bayes by 10-fold CV which turns out to be 90.1%.

The average classification accuracy for the three methods is plotted in Figure 8. Among the three, EM shows the poorest performance except for small #HC. Comparatively MCMC and VB demonstrate stable performance, around 96% accuracy through all values of #HC. The highest accuracy 97.0% is achieved by MCMC at #HC=12 which coincides with a broad peak in the average log ML curve. Contrastingly, although the average log VFE curve has its peak at #HC=3, the average accuracy curve for VB does not have a peak around that point. Nonetheless all three methods exhibit much better performance than Naive Bayes whose accuracy is 90.1%. Our experiment suggests the effectiveness of introducing a hidden variable to Naive Bayes though more experiments are needed to conclude it.

6 Related work

Our work belongs to the lineage of SRL (statistical relational learning) [Getoor and Taskar, 2007] and PLL (probabilistic logic learning)[De Raedt and Kersting, 2008] that have been intensively studied in the past decade. They upgrade feature-based probabilistic modeling to more structured and complex modeling with the help of relation and logic. However complex models often require intractable probabilistic inference and MCMC sampling has been used to overcome the intractability problem across a variety of formalisms. In logic programming for example Angelopoulos and Cussens studied MCMC sampling in SLPs (stochastic logic programs)[Muggleton, 1996] to give priors for structured objects [Angelopoulos and Cussens, 2008]. In BNs,

Milch and Russell proposed a generic MCMC algorithm for a BN-based modeling language BLOG in which proposal distributions are plugged in by the user [Milch and Russell, 2006]. Turning to undirected graphical models, FACTORIE, a software library for a strongly-typed functional language Scala, performs MCMC sampling on imperatively defined factor graphs [McCallum *et al.*, 2009]. Church is another example of the use of MCMC sampling in a functional language [Goodman *et al.*, 2008]. It is a stochastic extension of a higher-order functional language Scheme, and supports construction of non-parametric Bayesian models such as Dirichlet process.

Recently Cussens proposed an approximation method for Bayesian inference in PRISM which is quite different from our MCMC [Cussens, 2010]. He adapted an ABC (approximate Bayesian computation) SMC (sequential Monte Carlo) method to PRISM. Since it does not need to compute the likelihood of data, much less of computational burden is expected.

7 Conclusion

We proposed a general MCMC method for Bayesian inference in a logic-based Turing-complete language PRISM. It is applicable to any PRISM program (and hence to a broad class of generative models) and probabilistically samples possible explanations in statistical abduction for observations given as goals. Samples are used to estimate the marginal likelihood of the observations and to infer the best explanation for a new goal in Bayesian sense. We applied the new MCMC method to a PCFG and a PLCG for the ATR corpus, a profile-HMM and a variant of Naive Bayes augmented with a hidden variable.

References

[Angelopoulos and Cussens, 2008] N. Angelopoulos and J. Cussens. Bayesian learning of Bayesian networks with informative priors. *Annals of Mathematics and Artificial Intelligence*, 54(1-3):53–98, 2008.

[Cussens, 2010] J. Cussens. Approximate Bayesian computation for the parameters of Prism programs. In *Proceedings of the 20th International Conference on Inductive Logic Programming (ILP'10)*, pages –, 2010.

[De Raedt and Kersting, 2008] L. De Raedt and K. Kersting. Probabilistic inductive logic programming. In L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, editors, *Probabilistic Inductive Logic Programming - Theory and Applications*, Lecture Notes in Computer Science 4911, pages 1–27. Springer, 2008.

[Getoor and Taskar, 2007] L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA, 2007.

[Goodman *et al.*, 2008] N.D. Goodman, V.K. Mansinghka, D. Roy, K. Bonawitz, and J.B. Tenenbaum. Church: a language for generative models. In *Proceedings of the Twenty fourth Conference on Uncertainty in Artificial Intelligence (UAI'08)*, 2008.

[Johnson *et al.*, 2007] M. Johnson, T. Griffiths, and S. Goldwater. Bayesian inference for PCFGs via Markov chain monte carlo. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL'07)*, pages 139–146, 2007.

[Lunn *et al.*, 2000] D.J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter. Winbugs – a Bayesian modelling framework: concepts, structure, and extensibility. *Statistics and Computing*, 10:325–337, 2000.

[McCallum *et al.*, 2009] A. McCallum, K. Schultz, and S. Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Advances in Neural Information Processing Systems 22*, pages 1249–1257, 2009.

[Milch and Russell, 2006] B. Milch and S. Russell. General-Purpose MCMC Inference over Relational Structures. In *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI'06)*, pages 349–358, 2006.

[Milch *et al.*, 2005] B. Milch, B. Marthi, S. Russell, D. Sontag, D.L. Ong, and A. Kolobov. BLOG: Probabilistic models with unknown objects. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 1352–1359, 2005.

[Muggleton, 1996] S. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.

[Sato and Kameya, 1997] T. Sato and Y. Kameya. PRISM: a language for symbolic-statistical modeling. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 1330–1335, 1997.

[Sato and Kameya, 2001] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

[Stoye *et al.*, 1998] J. Stoye, D. Evers, and F. Meyer. Rose: generating sequence families. *Bioinformatics*, 14(2):157–163, 1998.

[Uratani *et al.*, 1994] N. Uratani, T. Takezawa, H. Matsuo, and C. Morita. ATR integrated speech and language database. Technical Report TR-IT-0056, ATR Interpreting Telecommunications Research Laboratories, 1994. In Japanese.

[Zhang *et al.*, 2005] H. Zhang, L. Jiang, and J. Su. Hidden naive Bayes. In *Twentieth National Conference on Artificial Intelligence*, pages 919–924. AAAI Press, 2005.