A Viterbi-like algorithm and EM learning for statistical abduction

Taisuke SATO
Dept. of Computer Science, Graduate School of Information Science and Engineering
Tokyo Institute of Technology
Ookayama 2-12-1 Meguro-ku Tokyo Japan 152-8552
sato@mi.cs.titech.ac.jp

Yoshitaka KAMEYA

Abstract

We propose statistical abduction as a first-order logical framework for representing and learning probabilistic knowledge. It combines logical abduction with a parameterized distribution over abducibles. We show that probability computation, a Viterbi-like algorithm and EM learning for statistical abduction achieve the same efficiency as specialized algorithms for HMMs (hidden Markov models), PCFGs (probabilistic context-free grammars) and sc-BNs ( singly connected Bayesian networks).

1 Introduction

Abduction is a form of inference that generates best hypotheses explaining an observed fact. For example, if one notices a fact that the grass is wet in the yard, he/she might abduce that it rained last night, or the sprinkler was on, by using general rules such as "if it rains, things get wet." Abduction has been used for diagnosis systems, planning, natural language processing, etc. in AI.

It is straightforward to formalize (part of) abduction in the logic programming framework [6, 9]. We have a background theory $T$ consisting of clauses$^1$ and an observed fact $G$ (usually a ground atom$^2$) to be explained, and the task of abduction is to generate a hypothesis $H = \{ h_1, \ldots, h_n \}$, i.e. an explanation, by choosing $h_i$s from a particular set of hypothesis candidates called abducibles,$^3$ such that $T \cup H \models G$ and $T \cup H$ is consistent.$^4$ The quality of $H$, the abduced hypothesis, is evaluated by various criteria such as simplicity, abduction cost, and so on [9, 8, 18].

While the above framework is logically sound, it is obviously incomplete. Especially it entirely ignores the problem of uncertainty in the real world. Our observations are often partial, inconsistent or contaminated by noise. So the abduced hypothesis should be treated as being true only to some degree. Also it must be noticed that our observation are always finite but potentially infinite (we may have another observation indefinitely), and we would like to know the eventual outcome of infinitely many observations. It is therefore, even desirable, to build a new abductive framework that unifies the logical framework with a statistical framework such as Bayesian networks.

This amalgamation is made possible by introducing probabilistic elements into logic programs. More exactly, it is made possible by introducing a (parameterized) probability distribution over abducibles, which enables us to identify the best hypothesis as the most likely hypothesis and the likelihood can be maximized by statistical learning. The resulting logical-statistical system will benefit both abductive logic programming and Bayesian networks, as the former is expected to become robust against noise and missing data and the latter gets first-order expressiveness as knowledge representation language. We term this unification statistical abduction.

There are however theoretical and practical problems to achieve the unification. First of all, statistical abduction must deal with infinitely many objects sanctioned by the language of first-order logic and their joint distribution, which gives rise to a theoretical problem of constructing an infinite joint distribution of structured objects. It requires to extend the realm of traditional Bayesian networks considerably.

Secondly, to use statistical abduction in practice, we need to know all values of statistical parameters just

$^1$Syntactically, a clause (in implicative form) is a logical formula $L_0 \rightarrow L_1 \land \ldots \land L_m$ ($m \geq 0$) where $L_i$ ($0 \leq i \leq m$) is an atom or its negation, and all variables are universally quantified at the outermost position. $L_0$ is called a head and $L_1 \land \ldots \land L_m$ the body. A definite clause is one such that every $L_i$ is an atom. Note that we adopt Prolog conventions in this paper, so a conjunction $A \land B$ is expressed as $(A, B)$, and variables begin with an upper case letter.

$^2$A ground atom is one that contains no variables.

$^3$Usually abducibles are ground atoms.

$^4$Induction also has similar formalization, and the distinction between induction and abduction is a subtle issue [7].
like conditional probabilities for Bayesian networks. Determining a large number of statistical parameters is a tough task, known as the where-do-the-numbers-come-from problem. Although one might hope that the problem is mitigated by using learning techniques, there has been little work on statistical learning in the literature of logical framework of abduction.

The objective of this paper is to point out that we have now a theoretical foundation for statistical abduction, an efficient algorithm for computing probabilities, an efficient EM algorithm for parameter learning, and finally a Viterbi-like algorithm that generates the most likely hypothesis. Since the subject is broad and the space is limited, descriptions are necessarily sketchy, and proofs and details are left to a full paper we are preparing. Also, we assume that the reader is familiar with the basics of logic programming [6].

In what follows, after a short review of background in Sec. 2, we explain PRISM programs in Sec. 3. Sec. 4 contains the description of efficient computation of probabilities, OLDT search, EM learning and a Viterbi-like algorithm, followed by the complexity analysis of PRISM programs for HMMs, PCFGs (probabilistic context-free grammars) and so BNs (singly-connected Bayesian networks). Sec. 5 is a conclusion.

2 Background

The use of first-order clauses in relation to Bayesian networks has been pursued in the past decade in two strands. In the KBMC (knowledge-based model construction) approach [3, 1, 11, 14], initiated by Breese in [3], they are used as a macro language to compactly represent similar Bayesian networks. Basically, a knowledge base KB contains clauses representing general rules and CPTs (conditional probability tables). Each time a set of evidence and context is given as ground atoms, a specialized Bayesian network is constructed by tracing logical/probabilistic dependencies in KB to compute the probability of a query atom. Uncertain parameters associated with CPTs can be learned by applying the EM learning algorithm for Bayesian networks [4] to the constructed network [11]. The KBMC approach is characterized as a local approach, implicitly defining a set of local distributions as Bayesian networks, each of which corresponds to a query, evidence and context.

Statistical abduction on the other hand takes a global approach which defines a single distribution over ground atoms. It was begun by Poole as "probabilistic Horn abduction" [15]. In his approach, a program contains non-probabilistic definite clauses and probabilistic disjoint declarations. A disjoint declaration is of the form disjoint([h1:p1,...,hn:pn]). It says hi, an abducible atom, becomes exclusively true with probability pi (0 ≤ i ≤ n). Abducibles in different declarations are supposed to be independent. The probability of a non-abducible ground atom is then calculated by reduction in a top-down manner through program clauses to a DNF formula made out of abducibles in the disjoint declarations. The probabilistic Horn abduction not only defines probabilities of atoms without referring to Bayesian networks but can represent Bayesian networks themselves [15].

While the probabilistic Horn abduction opened a new vista on extending Bayesian networks to first-order language, it left behind some problems. First it assumed a priori the acyclicity condition5 and the covering property6 of programs. These assumptions are hard to verify and could be severe restrictions in programming.7 Also the defined probability measure is not guaranteed to be completely additive. As a result, for instance, lim_{m→∞} Prob(∪i=1...m pi(t)) = Prob(∪i=1...m pi(z)) where the ti's are ground terms, is not guaranteed. More serious is the problem of determining parameters in disjoint declarations. How can we get them? It remained unanswered.

Aiming at providing a broader theoretical basis and a learning algorithm of uncertain parameters for the global approach, Sato proposed distribution semantics [19] and developed a first-order statistical modeling language PRISM9 [19, 20]. The proposed semantics defines an infinite joint distribution over ground atoms as the denotation of a given program containing abducibles with statistical parameters declared just like those in Poole’s disjoint declarations. It is a probabilistic extension of the least Herbrand model semantics for definite clause programs [6] to the possible world semantics with a completely additive probability measure. Since our extension is based on the least Herbrand model semantics, we don’t need the covering assumption or the acyclicity condition (because every definite program has a least Herbrand model and the iff completion [6] holds in it.). Similarly, neither the range-restrictedness condition nor normalization is necessary in our approach, which is required in proof-theoretic approaches to assigning probabilities to logic programs [5, 13].

5It says that every ground atom A must be assigned a unique integer s(A) such that s(A) > s(B1),...,s(Bm) for every ground instance of a clause of the form A ← B1,...,Bm.

6It requires that when there are finite ground instances A → α1,...,Ai,...,im (i ≤ m) about a ground atom A in the program, A → α1 ∨ ... ∨ αm holds. The case of infinitely many ground instances is not mentioned. Intuitively the property ensures every observation has an explanation. Logically it is equivalent to assuming the iff completion [6].

7Under the acyclicity condition, when a clause includes local variables like Y in p(X) ← q(X,Y),...we cannot write recursive clauses such as mbr(X,cons(2,Y)) ← mbr(X,Y). Also it is not clear how to check the covering property, as it specifies the relationship between what is potentially observable (which depends on nature) and what we can explain (which depends on the program).

9URL is http://sato-www.cs.titech.ac.jp/prism/
Theoretically, we may use as many constant symbols, function symbols and predicate symbols as we need, and can write whatever program. Of course, in actual programming, we have to care about efficiency, termination, etc.

Syntactically, our program $DB$ is a set of clauses $F \cup R$ where $F$ is a set of unit clauses and $R$ is a set of definite clauses. For the purpose of theoretical explanation however, we consider $DB$ as a set of infinitely many ground clauses made up of all possible ground instances of original clauses in $DB$. $F$ then is a set of infinitely many ground atoms. We associate with $F$ an infinite joint distribution $P_F$. So atoms in $F$ are probabilistically true and random sampling determines a set of true atoms $F^i$. Then think of a new definite clause program $DB' = F' \cup R$ and its least Herbrand model $M(DB')$ [6]. $M(DB')$ determines the truth values of all ground atoms in $DB$, which implies that every ground atom $A$ is a random variable (taking 1 when $A$ true and 0 otherwise), and therefore a joint distribution $P_{DB'}$ for all ground atoms is definable. $P_F$ is constructed from a collection of finite joint distributions $P^{(n)}_F (A_1 = x_1, \ldots, A_n = x_n) (n = 1, 2, \ldots) \text{ where } A_\mathbb{S}$ are random variables (abducibles). The resulting $P_{DB'}$ becomes a $\sigma$-additive probability measure over $\{0, 1\}$ (the set of all countably infinite sequences of 0 and 1).

For statistical abduction to be useful however, we need efficient methods for

- computing the probabilities of observations,
- computing the most likely explanation for the given observation, and
- learning parameters

As for HMMs, these methods correspond to the forward procedure, the Viterbi algorithm, and the Baum-Welch algorithm respectively [17].

PRISM, an implementation of the statistical abduction with $P_F$ being constrained to a specific form, has been developed as a modeling language designed for complex symbolic statistical phenomena [19, 20]. It comes with a built-in EM algorithm\footnote{The EM algorithm is a standard statistical method for MLE (maximum likelihood estimation) [23].} for parameter learning, but still lacks efficiency compared to HMMs.

In what follows, we show that there exist algorithms for PRISM programs that run as efficiently as specialized ones when they are applied to HMMs, PCFGs and sc-BNs.

3 PRISM programs

A PRISM program is defined as a definite clause program $DB = F \cup R$ which satisfies the following conditions on facts $F$, their distribution $P_F$ and rules $R$.

1. $F$ is a set of ground atoms of the form $msw(i, n, v)$. The arguments $i$ and $n$ are called group-id (or switch name) and trial-id, respectively. We assume that a finite set $V_i$ of ground terms is associated with each $i$, and $v \in V_i$ holds.\footnote{We consider $DB$ as a countably infinite set of ground clauses, and $i$ and $n$ are arbitrary ground terms in the Herbrand universe.} $V_i$ corresponds to a set of values of switch $i$.

2. Let $V_i = \{v_1, v_2, \ldots, v_{|i|}\}$. Then, one of the ground atoms $msw(i, n, v_1)$, $msw(i, n, v_2)$, ...., $msw(i, n, v_{|i|})$ becomes exclusively true (takes the value 1) on each trial. For each $i$, $\theta_i \in [0, 1]$ is a parameter of the probability of $msw(i, n, v)$ being true ($v \in V_i$), and $\sum_{v \in V_i} \theta_i = 1$ holds.

3. For each ground terms $i$, $\bar{i}$, $n$, $n'$, $v \in V_i$ and $v' \in V_{\bar{i}}$, random variable $msw(i, n, v)$ is independent of $msw(\bar{i}, n', v')$ if $n \neq n'$ or $i \neq \bar{i}$.

4. Define $head(R)$ as a set of atoms appearing in the head of $R$. Then, $F \cap head(R) = \emptyset$.

$msw/3$ in the first condition is to represent a basic probabilistic choice such as coin-tossing ($msw$ stands for multi-valued switch). A ground atom $msw(i, n, v)$ represents an event “a switch named $i$ takes on $v$ as a sample value on the trial $n$. “ The second and the third condition say that a logical variable $V$ in $msw(i, n, V)$ behaves similarly to a random variable which is realized to $v_k$ with probability $\theta_{ki}$ ($k = 1 \ldots |V_i|$).\footnote{Recall however that, in distribution semantics, logical variables themselves are not random variables.} Moreover, from the third condition, the logical variables $V_1$ and $V_2$ in $msw(i, n_1, V_1)$ and $msw(i, n_2, V_2)$ can be seen as independent and identically distributed (i.i.d.) random variables if $n_1$ and $n_2$ are different ground terms. From an abductive point of view, $msw$s are all abducibles, and the fourth condition requires that no $msw$ appears in the heads of $R$.

As an example of representing temporal processes in an abductive framework, we describe an HMM as a PRISM program. It looks like a usual Prolog program, but denotes a discrete stochastic process (of finite length, 3 in this case).

1) target(hmm/1).
2) data(\{\text{\textit{hmm.data}}\}).
3) table(\{\text{\textit{hmm/1, hmm/3}}\}).
4) values(\{\text{\textit{init.}}, \{s0, s1\}\}).
5) values(out(\{\text{\textit{\ldots}}, \{a, b\}\}).
6) values(tr(\{\text{\textit{\ldots}}, \{a, s1\}\}))
7) \text{\textit{hmm/3}}:- msw(init, null, Si), \text{\textit{hmm/1}}, \text{\textit{Si}}, \text{\textit{Cs}}.
8) \text{\textit{hmm(T, S, [C]|Cs)\)}}:- T<3, msw(out(S), T, C), msw(tr(S), T, NextS).
9) \text{\textit{hmm(T, \ldots, [\ldots])\)}}:- T>3.

Procedurally, the above HMM program simulates the generation process of strings.Clauses 7-9 represent the probabilistic behavior of the HMM. In clause 8, to output a symbol C, we use different switches
out(S) conditioned on the state S.\footnote{Generally, a CPT of a random variable X is representable as a switch msx(x, c1, c2, . . . , cn, z), where fX is the id of X, n is the number of conditional variables, ci (i = 1, . . . , n) is the value of each conditional variable Ci, and z is the value of X. Of course, X’s possible values should be declared in advance like Clause 4-6.} Note that T in msx(out(S), T, C) is used to guarantee independence among the choices at each time step. Recursive clauses like B are allowed in PRISM. Clauses 1-6 contain additional information about the program (they are called control declarations). Clause 1 declares only ground atoms containing h/m/1 are observable\footnote{This is just for the case of implementation.} h/m/(a, b, s1) being true means this HMM generates the string aba. Clause 2 specifies a file storing learning data. Clause 3 specifies the table predicates (described later) are h/m/1 and h/m/3. We can read that \( V_{\text{init}} = \{s0, s1\}, V_{\text{out}}(\cdot) = \{a, b\}, V_{\text{tr}}(\cdot) = \{s0, s1\} \) from clauses 4-6.

4 Three basic tasks

To apply statistical abduction to the real world, we consider, based on the analogy of HMMs\footnote{\( \theta \) is the vector consisting of parameters associated with all abducibles which forms the explanations for the observed fact G or an observation in G.}, three basic tasks: (1) computing \( P_{DB}(G = 1|\theta) \),\footnote{For simplicity, we hereafter omit the subscript DB (similar abbreviation may occur later).} the probability of an atom G representing an observation, (2) finding \( S^* \), the most likely explanation for G, and (3) adjusting the parameters to maximize the probability of a given sequence \( G = \{G_1, G_2, . . . , G_T\} \) of observations. All solutions should be computationally tractable.

Poole\footnote{\( \text{co} \)} described a method for the first task. Let us consider the following if-and-only-if (iff) relation under \( \text{comp}(R) \), the Clark’s completion of the rules \( R \) [6]:

\[
\text{comp}(R) \equiv G \leftrightarrow \bigwedge_{i=1}^{m} S_i(\theta)
\]

He assumes that all abducibles in \( S_i(\theta) \) (a finite conjunction of abducibles) are independent and \( S_i(\theta) \) are exclusive to each other (we say DB satisfies the exclusiveness condition). Let \( \hat{v}_DB(G) \)\footnote{In order to derive efficient algorithms for the three tasks, comparable to the forward procedure and the Viterbi algorithm, we need to add a couple of assumptions about programs. We assume that DB satisfies the exclusiveness condition, and that m is finite in Eq. 1 (we say DB satisfies the finite support condition).\footnote{Further, we assume that the following if-relational formula holds for some finite ordered set \( \tau_DB(G) = \{\tau_1, . . . , \tau_K\} \) of table atoms (atoms containing the table predicate\footnote{OLDT stands for Ordered Linear Resolution for Definite clauses with Tabulation.}):}

\[
\begin{align*}
\text{comp}(R) & \equiv (G \leftarrow S_1, . . . , S_m) \\
& \lor (\tau_1 \leftarrow S_1, . . . , S_m) \\
& \lor \cdots \lor (\tau_K \leftarrow S_K, . . . , S_m),
\end{align*}
\]

where, letting G be a special table atom \( \tau_i \), each in \( \hat{v}_DB(\tau_i) = \{S_1, . . . , S_m\} \) (1 \( \leq K \) is a subset of \( F \cup \{\tau_1, . . . , \tau_K\} \) called a factorized-or tabulated-explanation for \( \tau_i \), and each of \( S_1, . . . , S_m \) (0 \( \leq K \) is a set of independent atoms.

\[ \hat{v}_DB(\tau_i) \] and \( \tau_DB(G) \) can be obtained by OLDT search\footnote{This condition says that every observation has a finite number of explanations. Without it, our search for explanations would not terminate.} [22], a complete refutation technique with tabulation for logic programs. We first translate the source PRISM program to Prolog similarly to definite clause grammars (DCGs) [21], or Poole’s Theories [16].

The following Prolog program is a translation of the HMM program in Sec. 3. Clauses Tr and Tj are generated from the clause j:

\[
\begin{align*}
& (T_1) \quad \text{top_hmm(Cs, X) :- tab_hmm(Cs, X, []).} \\
& (T_2) \quad \text{top_hmm(Cs, [I | Cs] X, X) :- h2mm(Cs, X, []).} \\
& (T_3) \quad \text{top_hmm(T, S, Cs, X) :- h2mm(T, S, X, []).} \\
& (T_4) \quad \text{msv(tr, T, S, X) :- tab_hmm(T, X, l, S, X, []).} \\
& (T_5) \quad \text{msv(tr, T, S, X) :- tab_hmm(T, X, l, S, X, []).} \\
& (T_6) \quad \text{msv(tr, T, S, X) :- tab_hmm(T, X, l, S, X, []).} \\
& (T_7) \quad \text{msv(tr, T, S, X) :- tab_hmm(T, X, l, S, X, []).} \\
& (T_8) \quad \text{msv(tr, T, S, X) :- tab_hmm(T, X, l, S, X, []).} \\
& (T_9) \quad \text{msv(tr, T, S, X) :- tab_hmm(T, X, l, S, X, []).}
\end{align*}
\]

As can be seen, in translation, we add two arguments to collect factorized explanations and rename the predicates of abducibles (msv) and table atoms. We then perform OLDT search, noting extra arguments do not influence the search procedure, while saving in the solution table a list of factorized explanations every time
Figure 1: Solution table for the observation $hmm([a,b,a])$.

the table atom is solved. For example, the OLDT search for the above translation yields Fig. 1. Finally, we extract $\psi$, the set of all factorized explanations from the solution table. The remaining task is to get totally ordered table atoms, i.e. the ordered set $\tau_{DB}(G)$, respecting the acyclicity in Eq. 2, which can be done by topological sorting.

4.2 Computing the observation probability and the most likely explanation

Given the iff-relational formula in Eq. 2, an efficient algorithm for computing $P_{DB}(G = 1|\bar{0})$ (the first task) is derivable based on the analogy of the inside probability in Baker’s Inside-Outside algorithm [2]. In our formulation however, the inside probability of a table atom $\tau$ is $P_{DB}(\tau = 1|\bar{0})$, the probability of $\tau$ being true. It should be noted that, when computing $P[\tau_i]$, the inside probabilities $P[\tau_k], P[\tau_{k-1}], \ldots, P[\tau_{i+1}]$ have already been computed. The computation terminates with $P[G] = P[\tau_0]$, the inside probability of $G$.

```
procedure Get-Inside-Probs (DB, G) begin
    Put $G = \tau_0$;
    for $k := K$ downto 0 do
        $P[\tau_i] := 
        \sum_{\mathcal{E} \in \hat{\psi}(\tau_i)} \prod_{m = i}^{i+\lambda} e^{\theta_i \cdot \tau} \prod_{\tau_j \in \{\tau_{i+1}, \ldots, \tau_K\}} P[\tau]$
    end.
```

Similarly, an efficient algorithm for computing the most likely explanation $S^*$ (i.e. the second task) is derived. The algorithm Get-ML-Expl first computes $\delta[\tau_i]$, the maximum probability of factorized explanations for each table atom $\tau_i$. $E[\tau_i]$, the most likely factorized explanation for $\tau_i$, is then constructed. Finally, we construct $S^*$ from $E[\tau_0]$. For an HMM program (see Sec. 3), Get-ML-Expl is equivalent to the Viterbi algorithm, a standard algorithm for finding the most likely state-transition path of HMMs. Furthermore, when we write a probabilistic context-free grammar (PCFG) in PRISM, it is easily shown that probabilistic parsers for PCFGs, which finds most likely parse of a given sentence, are equivalent to Get-ML-Expl.

```
1: procedure Get-ML-Expl(DB, G) begin
2:     Put $G = \tau_0$;
3:     for $k := K$ downto 0 do begin
4:         $\forall S \in \hat{\psi}(\tau_i)$ do
5:             $\delta[\tau_i, S] := 
6: \prod_{m = i}^{i+\lambda} \prod_{\tau_j \in \{\tau_{i+1}, \ldots, \tau_K\}} P[\tau]$
7:         end;
8:     end;
9:     $E[\tau_i] := \arg\max_{\tau_j \in \{\tau_{i+1}, \ldots, \tau_K\}} \delta[\tau_i, S]$;
10: end;
```

4.3 Learning algorithm

The learning of PRISM programs means MLE (maximum likelihood estimation) of statistical parameters of $\text{msw}$ embedded in a program, for which the EM algorithm is appropriate [23]. A new EM algorithm named graphical EM algorithm that takes advantage of the structure of the iff-relational formula in Eq. 2 has been derived on the analogy of computation of the outside probabilities in the Inside-Outside algorithm [10].

We added an assumption that all observable atoms are exclusive to each other and the probabilities of these atoms sum up to one (we say DB satisfies the uniqueness condition).

The graphical EM algorithm first initializes the parameters $\theta$ and then iterates re-estimation until the likelihood saturates. In each iteration, the expected occurrence is computed first for each abducible atom $\text{msw}(i,n,v)$ from its parameter $\theta_{i,n,v}$, the inside probabilities and the outside probabilities, and then all parameters are simultaneously re-estimated by using such expected occurrences. Final values of parameters become learned ones.

4.4 Complexity

In this section, we primarily evaluate time complexity of our methods for the first and the second task due to space limitations. The method for the first (resp. the second) task comprises two phases — OLDT search and Get-Inside-Probs (resp. Get-ML-Expl). Hence, we should estimate each algorithm separately. Assuming that all table operations can be performed in $O(1)$ time, the computation time of OLDT search is measured by
the size of the search tree. As for the computation time of GET-INSIDE-PROBS and GET-ML-EXPL, it is $O(\xi_{\text{max}}N \sqrt{\lambda})$, where $\xi_{\text{max}} \equiv \max_{S \in \mathcal{S}} |S|$, and $\Delta \equiv \bigcup_{\epsilon \in \mathcal{D}} \mathcal{B}^\varepsilon (\sigma)$. In the case of the HMM program, $\xi_{\text{max}} = O(N^2 L)$ and $\xi_{\text{max}} = O(1)$. Hence, the computation time of GET-INSIDE-PROBS and GET-ML-EXPL is $O(N^2 L)$. This is the same order as that of the forward procedure and the Viterbi algorithm. So GET-ML-EXPL (resp. GET-INSIDE-PROBS) is a generalization of the Viterbi algorithm (resp. the forward procedure). For existing symbolic-statistical models, we summarize computation time as follows:

<table>
<thead>
<tr>
<th>Model</th>
<th>O(LDT time)</th>
<th>O(GP/GMLE time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMMs</td>
<td>$O(N^2 L)$</td>
<td>$O(N^2 L)$</td>
</tr>
<tr>
<td>sc-BNs</td>
<td>$O(</td>
<td>V</td>
</tr>
<tr>
<td>PCFGs</td>
<td>$O(M^3 L^3)$</td>
<td>$O(M^3 L^3)$</td>
</tr>
</tbody>
</table>

The second (resp. the third) column “O(LDT time)” (resp. “O(GP/GMLE time)” indicates that the computation time of LDT search (resp. GET-INSIDE-PROBS and GET-ML-EXPL) for the model in the first column. $N$, $L$, $T$, $|V|$ and $M$ are the number of states of the target HMM, the maximum length of input strings, the size of training data, the number of nodes in the target Bayesian network, and the number of non-terminal symbols in the target PCFG, respectively. The above table exemplifies that our general framework can subsume specific algorithms such as the Viterbi algorithm.

5 Conclusion

We have proposed statistical abduction as the combination of abductive logic programming with a distribution over abducibles. It has first-order expressive power and integrates current most powerful probabilistic knowledge representation frameworks such as HMMs, PCFGs and Bayesian networks. Besides, our general algorithms developed for the three tasks (probability computation, the search for the most likely explanation, and EM learning) achieve the same efficiency as specialized algorithms for above three frameworks.

Acknowledgments

This research is supported by Grant-in-Aid for Scientific Research on Priority Area “Discovery Science” of the Ministry of Education, Science, Sports and Culture, Japan.

References