# A Logic-based Approach to Generatively Defined Discriminative Modeling

Taisuke Sato[1], Keiichi Kubota[1], and Yoshitaka Kameya[2]

[1] Tokyo institute of Technology, Japan
{sato,kubota}@mi.cs.titech.ac.jp
[2] Meijo University, Japan
ykameya@meijo-u.ac.jp

**Abstract.** Conditional random fields (CRFs) are usually specified by graphical models but in this paper we propose to use probabilistic logic programs and specify them generatively. Our intension is first to provide a unified approach to CRFs for complex modeling through the use of a Turing complete language and second to offer a convenient way of realizing generative-discriminative pairs in machine learning.
We implemented our approach as the D-PRISM language by modifying PRISM, a logic-based probabilistic modeling language for generative modeling, while exploiting its dynamic programming mechanism for efficient probability computation. We tested D-PRISM with logistic regression, a linear-chain CRF and a CRF-CFG and empirically confirmed their excellent discriminative performance compared to their generative counterparts, i.e. naive Bayes, an HMM and a PCFG.

## 1 Introduction

Conditional random fields (CRFs) [1] are probabilistic models for discriminative modeling defining a conditional distribution $p(y \mid x)$ over output $y$ given input $x$. They are quite popular for labeling sequence data such as text data and biological sequences [2]. Although they are usually specified by graphical models, we here propose to use probabilistic logic programs and specify them generatively. Our intension is first to provide a unified approach to CRFs for complex modeling through the use of a Turing complete language and second to offer a convenient way of realizing generative-discriminative pairs [3] in machine learning.

The use of logical expressions to specify CRFs is not new but they have been used solely as feature functions [4, 5]. For example in Markov logic networks (MLNs)[5], weighted clauses are used as feature functions to define (conditional) Markov random fields and probabilities are obtained by Gibbs sampling. Contrastingly in our approach implemented by a generative modeling language PRISM [6, 7], clauses have no weights; they simply constitute a logic program *DB* that computes possible output $y$ from input $x$ by proving a top-goal $G_{x,y}$ relating $x$ to $y$ from *DB*, and probabilities are exactly computed by dynamic programming. However *DB* contains special atoms of the form $\mathtt{msw}(i, v)$, inherited from PRISM and called `msw` atoms, have weights $\exp(\lambda_{i,v} \cdot f_{i,v})$ where $\lambda_{i,v}$

is a real number and $f_{i,v}$ is a binary feature (0 or 1) respectively. $i$ and $v$ are arbitrary terms. They may depend on $x$ and $y$, and so may do $f_{i,v}$. We define the weight $q(x, y)$ of a top-goal $G_{x,y}$ as a sum-product of such weights appearing in a proof of $G_{x,y}$ and consider $q(x, y)$ as an unnormalized distribution. By modifying the dynamic programming mechanism of PRISM slightly, we can efficiently compute, when possible and feasible, the unnormalized marginal distribution $q(x) = \sum_y q(x, y)$ and obtain a CRF $p(y \mid x) = q(x, y)/q(x)$. We implemented our idea by modifying PRISM and termed the resulting language D-PRISM (discriminative PRISM). D-PRISM is a general programming language that generatively defines CRFs and provides built-in predicates for parameter learning and Viterbi inference of CRFs.

Our approach to CRFs is general in the sense that, like other statistical relational learning (SRL) languages for CRFs [5, 8], programs in D-PRISM have no restriction such as the exclusiveness condition in PRISM [7] except for the use of binary features and we can write any program, i.e. we can write arbitrary CRFs as long as they are computationally feasible. We point out that binary features are most common features and they can encode basic CRF models such as logistic regression, linear-chain CRFs and CRF-CFGs [9, 10, 2]. Furthermore by dynamic programming, probabilistic inference can be efficiently carried out with the same time complexity as their generative counterparts like the case of linear-chain CRFs and hidden Markov models (HMMs).

In machine learning it is well-known that naive Bayes and logistic regression form a generative-discriminative pair [3]. That is, any conditional distribution $p(y \mid \boldsymbol{x})$ computed from a joint distribution $p(\boldsymbol{x}, y) = p(\boldsymbol{x} \mid y)p(y)$ defined generatively by naive Bayes, where $y$ is a class and $\boldsymbol{x}$ is a feature vector, can also be defined directly by logistic regression and vice versa. As is empirically demonstrated in [3], classification accuracy by discriminative models such as logistic regression is generally better than their corresponding generative models such as naive Bayes when there is enough data but generative models reach their best performance more quickly than discriminative ones w.r.t. the amount of available data. Also the theoretical analysis in [11] suggests that when a model is wrong in generative modeling, the deterioration of prediction accuracy is more severe than in discriminative modeling. It seems therefore reasonable to say "...For any particular data set, it is impossible to predict in advance whether a generative or a discriminative model will perform better" [2]. Hence what is desirable is to provide a modeling environment in which the user can test both types of modeling smoothly without pains and D-PRISM provides such an environment that makes it easy to test and compare discriminative modeling and generative modeling for the same class or related family of probabilistic models.

In what follows, we review CRFs in Section 2 and also review three basic models, i.e. logistic regression, linear-chain CRFs and CRF-CFGs in Section 3. We then introduce D-PRISM in Section 4 which is a discriminative version of PRISM. We empirically verify the effectiveness of our approach in Section 5 using the three basic models. Section 6 contains related work and discussion and Section 7 is the conclusion.

## 2 Conditional random fields

Conditional random fields (CRFs) [1] are popular probabilistic models defining a conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ over the output sequence $\boldsymbol{y}$[3] given an input sequence $\boldsymbol{x}$ which takes the following form.

$$p(\boldsymbol{y} \mid \boldsymbol{x}) \equiv \frac{1}{Z(\boldsymbol{x})} \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(\boldsymbol{x}, \boldsymbol{y})\right\} \tag{1}$$

Here $f_k(\boldsymbol{x}, \boldsymbol{y})$ and $\lambda_k$ $(1 \leq k \leq K)$ are respectively a real valued function (*feature function*) and the associated weight (*parameter*) and $Z(\boldsymbol{x})$ a normalizing constant. As $Z(\boldsymbol{x})$ is the sum of exponentially many terms, the exact computation is generally intractable and takes $\mathcal{O}(M^{|y|})$ time where $M$ is the maximum number of possible values for each component of $\boldsymbol{y}$ and hence approximation methods have been developed [2]. However when $p(\boldsymbol{y} \mid \boldsymbol{x})$ has recursive structure of specific type as a graphical model like linear-chain CRFs, $Z(\boldsymbol{x})$ is efficiently computable by dynamic programming.

Now let $D = \{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \cdots, (\boldsymbol{x}^{(T)}, \boldsymbol{y}^{(T)})\}$ be a training set. The conditional log-likelihood $l(\boldsymbol{\lambda} \mid D)$ of $D$ is given by

$$\begin{aligned} l(\boldsymbol{\lambda} \mid D) &\equiv \sum_{t=1}^{T} \log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(t)}) - \frac{\mu}{2} \sum_{k=1}^{K} \lambda_k^2 \\ &= \sum_{t=1}^{T} \left\{ \sum_{k=1}^{K} \lambda_k f_k(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) - \log Z(\boldsymbol{x}^{(t)}) \right\} - \frac{\mu}{2} \sum_{k=1}^{K} \lambda_k^2 \end{aligned}$$

where $\boldsymbol{\lambda} = \lambda_1, \ldots, \lambda_K$ are parameters and $\frac{\mu}{2} \sum_{k=1}^{K} \lambda_k^2$ is a penalty term. Parameters are then estimated as the ones that maximize $l(\boldsymbol{\lambda} \mid D)$, for example by Newton's or quasi-Newton methods. The gradient required for parameter learning is computed as

$$\frac{\partial l(\boldsymbol{\lambda} \mid D)}{\partial \lambda_k} = \sum_{t=1}^{T} \left\{ f_k(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) - E(f_k \mid \boldsymbol{x}^{(t)}) \right\} - \mu \lambda_k.$$

The problem here is that the expectation $E(f_k \mid \boldsymbol{x}^{(t)})$ is difficult to compute and hence a variety of approximation methods such as stochastic gradient descent (SDG) [2] have been proposed. However in this paper we focus on cases where exact computation by dynamic programming is possible and use an algorithm that generalizes the outside probability computation in probabilistic context free grammars (PCFGs) [12].

[3] Bold italic letters are random vectors in this paper.

After parameter learning, we apply our model to prediction tasks and infer the most-likely output $\hat{\boldsymbol{y}}$ for an input sequence $\boldsymbol{x}$ which is given by (2) below. As naively computing $\hat{\boldsymbol{y}}$ is straightforward but too costly, we again consider only cases where dynamic programming is feasible and apply a variant of the Viterbi algorithm for HMMs.

$$\hat{\boldsymbol{y}} \equiv \mathrm{argmax}_y p(\boldsymbol{y} \mid \boldsymbol{x}) \;=\; \mathrm{argmax}_y \sum_{k=1}^{K} \lambda_k f_k(\boldsymbol{x}, \boldsymbol{y}) \tag{2}$$

# 3 Basic models

## 3.1 Logistic regression

Logistic regression specifies a conditional distribution $p(y \mid \boldsymbol{x})$ over a class variable $y$ given the input $\boldsymbol{x} = x_1, \ldots, x_K$, a vector of attributes. It assumes $\log p(y \mid \boldsymbol{x})$ is a linear function of $\boldsymbol{x}$ and given by

$$p(y \mid \boldsymbol{x}) \equiv \frac{1}{Z(\boldsymbol{x})} \exp\left\{\lambda_y + \sum_{j=1}^{K} \lambda_{y,j} x_j\right\}. \tag{3}$$

We here confirm that logistic regression is a CRF. Rewrite $\lambda_y = \sum_{y'} \lambda_{y'} \mathbf{1}_{\{y'=y\}}$ and $\lambda_{y,j} x_j = \sum_{y'} \lambda_{y',j} \mathbf{1}_{\{y'=y\}} x_j$[4] and substitute them for $\lambda_y$ and $\lambda_{y,j} x_j$ in (3). We then obtain

$$p(y \mid \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left\{\sum_{y'} \lambda_{y'} \mathbf{1}_{\{y'=y\}} + \sum_{y'} \sum_{j=1}^{K} \lambda_{y',j} \mathbf{1}_{\{y'=y\}} x_j\right\}. \tag{4}$$

By considering $\mathbf{1}_{\{y'=y\}}$ and $\mathbf{1}_{\{y'=y\}} x_j$ as feature functions (of $y$ and $\boldsymbol{x}$), we can see logistic regression is a CRF.

## 3.2 Linear-chain CRFs

CRFs [1] are generally intractable and a variety of approximation methods such as sampling and loopy BP have been developed. There is however a tractable subclass called *linear-chain CRF*s defining a conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ shown in (5) below over sequences $\boldsymbol{y}$ given the sequence $\boldsymbol{x}$ such that $|\boldsymbol{x}|$[5]$= |\boldsymbol{y}|$ and feature functions are restricted to the form $f(\boldsymbol{x}, y_i, y_{i-1})$ $(2 \leq i \leq |\boldsymbol{y}|)$. As a result exact probability computation is possible in time linear in the input length $|\boldsymbol{x}|$ by a variant of the forward-backward algorithm. They are considered as a generalized and undirected version of HMMs which enable us to use far richer feature functions other than transition probabilities and emission probabilities in HMMs.

$$p(\boldsymbol{y} \mid \boldsymbol{x}) \equiv \frac{1}{Z(\boldsymbol{x})} \exp\left\{\sum_{k=1}^{K} \lambda_k \sum_{i} f_k(\boldsymbol{x}, y_i, y_{i-1})\right\} \tag{5}$$

[4] $\mathbf{1}_{\{y'=y\}}$ is a binary function of $y$ taking 1 if $y = y'$, otherwise 0.
[5] $|\boldsymbol{x}|$ denotes the length of vector $\boldsymbol{x}$.

### 3.3 CRF-CFGs

PCFGs [12] are a basic class of probabilistic grammars extending CFGs by assigning selection probabilities $\boldsymbol{\theta}$ to production rules. In PCFGs, the probability of a sentence is the sum of probabilities of parse trees and the probability of a parse tree is the product of probabilities associated with production rules used in the tree. PCFGs are generative models and parameters are usually learned by maximum likelihood estimation (MLE). So given parse trees $\tau_1, \ldots, \tau_T$ and the corresponding sentences $s_1, \ldots, s_T$, parameters are estimated as $\boldsymbol{\theta}^* = \mathrm{argmax}_\theta \prod_{t=1}^{T} p(\tau_t, s_t \mid \boldsymbol{\theta})$.

Seeking better parsing accuracy, Johnson attempted parameter learning by maximizing conditional likelihood: $\boldsymbol{\theta}^{'} = \mathrm{argmax}_\theta \prod_{t=1}^{T} p(\tau_t \mid s_t, \boldsymbol{\theta})$ but found the improvement is not statistically significant [9]. Later Finkel et al. generalized PCFGs to *conditional random field context free grammar*s (CRF-CFGs) where the conditional probability $p(\tau \mid s)$ of a parse tree $\tau$ given a sentence $s$ is defined by (6) [10].

$$p(\tau \mid s) \equiv \frac{1}{Z(s)} \exp\left\{ \sum_{k=1}^{K} \lambda_k \sum_{r \in \tau} f_k(r, s) \right\} \tag{6}$$

Here $\lambda_1, \ldots, \lambda_K$ are parameters and $r \in \tau$ is a CFG rule (enriched with other information) appearing in the parse tree $\tau$ and $f_k(r, s)$ is a feature function.

They conducted learning experiments with a CRF-CFG using the Penn Treebank [13]. Parameters are learned from parse trees $\tau_1, \ldots, \tau_T$ and corresponding sentences $s_1, \ldots, s_T$ in the corpus by maximizing conditional likelihood just like [9] but this time they obtained a significant gain in parsing accuracy [10]. Their experiments clearly demonstrate the advantage of extensive use of features and discriminative parameter learning.

## 4 D-PRISM

Having seen basic models of CRFs, we next show how they are uniformly subsumed by a logic-based modeling language PRISM [6, 7] with a simple modification of its probability computation. The modified language is termed *D-PRISM* (discriminative PRISM).

### 4.1 PRISM at a glance

Before proceeding further we quickly review PRISM[6]. PRISM is a high-level generative modeling language based on Prolog extended with a rich array of probabilistic built-in predicates for various types of probabilistic inference and parameter learning. Specifically it offers, in addition to MLE by the EM algorithm, Viterbi training (VT), variational Bayes (VB), variational VT (VB-VT) and MCMC for Bayesian inference. PRISM has been applied to music and

[6] `http://sato-www.cs.titech.ac.jp/prism/`

bioinformatics [14–16]. A PRISM program *DB* contains `msw` atoms of the form $\texttt{msw}(i, v)$ ($i$ and $v$ are terms) representing a probabilistic choice such that for given $i$, there is a set of possible values (outcomes) $V = \{v_1, \ldots, v_{|V|}\}$ and executing $\texttt{msw}(i, X)$ returns $v_k$ in $X$ with probability $\theta_k$ ($1 \leq k \leq |V|$) where $\sum_{k=1}^{|V|} \theta_k = 1$. $\{\theta_1, \ldots, \theta_{|V|}\}$ are called *parameters for* $\texttt{msw}(i, \cdot)$.

*DB* defines a probability measure $p_{DB}(\cdot)$ over Herbrand interpretations [7]. The probability $p_{DB}(G)$ of a top-goal $G$ then is computed as a sum-product of parameters in two steps. First $G$ is reduced using *DB* by SLD search to a disjunction $E_1 \vee \cdots \vee E_M$ such that each $E_j$ ($1 \leq j \leq M$) is a conjunction of `msw` atoms representing a sequence of probabilistic choices. $E_j$ is called an *explanation* for $G$ because it explains why $G$ is true or how $G$ is probabilistically generated. Let $\phi(G) \equiv \{E_1, \ldots, E_M\}$ be the set of all explanations for $G$. $p_{DB}(G)$ is computed as $p_{DB}(G) = \sum_{E \in \phi(G)} p_{DB}(E)$ and $p_{DB}(E) = \prod_{k=1}^{N} \theta_k$ for $E = \texttt{msw}_1 \wedge \cdots \wedge \texttt{msw}_N$, where $\theta_k$ is a parameter for $\texttt{msw}_k$ ($1 \leq k \leq N$).

Let $p(x, y)$ be a joint distribution over input $x$ (or observation) and output $y$ (or hidden state). Suppose we wish to construct $p(x, y)$ as a PRISM program. We write a program *DB* that probabilistically proves $G_{x,y}$, a top-goal that relates $x$ to $y$, using `msw` atoms, in such a way that $p(x, y) = p_{DB}(G_{x,y})$ holds. Since $(x, y)$ forms a complete data, $G_{x,y}$ has only one explanation $E_{x,y}$ for $G_{x,y}$, so we have $p_{DB}(G_{x,y}) = p_{DB}(E_{x,y}) = \prod_{i,v} \theta_{i,v}^{\sigma_{i,v}(E_{x,y})}$ where $\sigma_{i,v}(E_{x,y})$ is the count of occurrences of $\texttt{msw}(i, v)$ in $E_{x,y}$. Introduce $G_x = \exists y\, G_{x,y}$. Then the marginal probability $p(x)$ is obtained as $p_{DB}(G_x)$ because $p(x) = \sum_y p_{DB}(G_{x,y}) = p_{DB}(G_x)$ holds. Hence the conditional distribution $p(y \mid x)$ is computed as

$$p(y \mid x) = \frac{p_{DB}(G_{x,y})}{p_{DB}(G_x)} = \frac{p_{DB}(E_{x,y})}{p_{DB}(G_x)} = \frac{\prod_{i,v} \theta_{i,v}^{\sigma_{i,v}(E_{x,y})}}{\sum_{E_{x,y} \in \phi(G_x)} \prod_{i,v} \theta_{i,v}^{\sigma_{i,v}(E_{x,y})}}. \quad (7)$$

Let `T` stand for temperature, `H` humidity, and `S` season respectively. A PRISM program in Fig. 1 generatively defines a joint distribution $p(\texttt{[T,H]}, \texttt{S})$ by naive Bayes as the distribution of `nb([T,H],S)`. It first probabilistically generates a season `S` by executing a probabilistic built-in predicate `msw(season,S)`[7], then similarly generates a value `T` of temperature and a value `H` of humidity, each conditioned on `S`, by executing `msw(attr(temp,S),T)` and `msw(attr(humidity,S),H)` in turn. The correspondence to (7) is that $G_{x,y}$ = `nb([T,H],S)` and $G_x$ = `nb([T,H])` where $x$ = `[T,H]` and $y$ = `S`.

### 4.2 From probability to weight

The basic idea of our generative approach to discriminative modeling is to generalize (7) by generalizing probability to weight, i.e. by replacing parameters $\theta_{i,v}$ for $\texttt{msw}(i, v)$ with *weight*s of the form $\exp(\lambda_{i,v} f_{i,v}(G_x, E_{x,y}))$ where $f_{i,v}(G_x, E_{x,y})$ is a binary feature function taking 0 or 1. We compute $p_{DB}(G_x)$ as a sum-product

[7] `S` in `msw(season,S)` runs over a set {`spring`,`summer`,`fall`,`winter`} of seasons as specified by the `values/2` declaration `values(season,[spring,summer,fall,winter])`.

```
values(season,[spring,summer,fall,winter]).
values(attr(temp,_),[high,mild,low]).
values(attr(humidity,_),[high,low]).

nb([T,H],S):-                          % defines p([T,H],S)
    msw(season,S),                     % S from {spring,summer,fall,winter}
    msw(attr(temp,S),T),               % T from {high,mild,low}
    msw(attr(humidity,S),H).           % H from {high,low}
nb([T,H]):- nb([T,H],_).               % defines p([T,H])
```

**Fig. 1.** Naive Bayes program

of weights as before but with normalization, to have a CRF in the left-hand side of (7). More precisely, we first introduce an unnormalized distribution $q(x, y)$ defined by:

$$q(x,y) \equiv \exp\left(\sum_{i,v} \lambda_{i,v} f_{i,v}(G_x, E_{x,y}) \sigma_{i,v}(E_{x,y})\right)$$
$$\text{where } f_{i,v}(G,E) \;=\; \begin{cases} 1 : E \in \phi(G) \wedge \texttt{msw}(i,v) \in E \\ 0 : \text{otherwise.} \end{cases} \tag{8}$$

obtained from replacing probabilities with weights in the computation of $p_{DB}(G_{x,y})$ ($=p_{DB}(E_{x,y})$). Note that by setting $\lambda_{i,v} = \ln \theta_{i,v}$, $q(x, y)$ is reduced to $p(x, y) = p_{DB}(G_{x,y})$ again. Next we introduce $\eta_{i,v} \equiv \exp(\lambda_{i,v})$ and rewrite (7) as $p(y \mid x) = p(E_{x,y} \mid G_x) = \dfrac{q(x,y)}{\sum_y q(x,y)}$. We reach (9) and (10).

$$p(E_{x,y} \mid G_x) = \frac{1}{Z(G_x)} \exp\left(\sum_{i,v} \lambda_{i,v} \sigma_{i,v}(E_{x,y})\right) = \frac{1}{Z(G_x)} \prod_{i,v} \eta_{i,v}^{\sigma_{i,v}(E_{x,y})} \tag{9}$$

$$Z(G_x) = \sum_{E_{x,y} \in \phi(G_x)} \exp\left(\sum_{i,v} \lambda_{i,v} \sigma_{i,v}(E_{x,y})\right) \;=\; \sum_{E_{x,y} \in \phi(G_x)} \prod_{i,v} \eta_{i,v}^{\sigma_{i,v}(E_{x,y})} \tag{10}$$

(9) and (10) are fundamental equations for D-PRISM describing how a CRF $p(y \mid x)$ is defined and computed as $p(E_{x,y} \mid G_x)$. By comparing (7) to (9) and (10), we notice that the most computationally demanding task, computing $Z(G_x)$ in (10), can be carried out efficiently by dynamic programming just by replacing $\theta_{i,v}$ in PRISM with $\eta_{i,v}$, which gives the same time complexity as PRISM's probability computation.

Syntactically D-PRISM programs are PRISM programs. More precisely they are PRISM programs that prove top-goals $G_{x,y}$ for complete data $(x, y)$ and $G_x$

for incomplete data $x$. For example, the PRISM program in Fig. 1 for naive Bayes is also a D-PRISM program defining logistic regression.

In D-PRISM, parameters are learned discriminatively from complete data. Consider the (log) conditional likelihood $l(\boldsymbol{\lambda} \mid D)$ of a set of observed data $D = \{d_1, d_2, \cdots, d_T\}$ where $d_t = (G_{x^{(t)}}, E_{x^{(t)},y^{(t)}}) = (G_t, E_t)$ $(1 \leq t \leq T)$. $l(\boldsymbol{\lambda} \mid D)$ is given by

$$\begin{aligned} l(\boldsymbol{\lambda} \mid D) &\equiv \sum_{t=1}^{T} \log p(E_t \mid G_t) - \frac{\mu}{2} \sum_{i,v} \lambda_{i,v}^2 \\ &= \sum_{t=1}^{T} \left\{ \sum_{i,v} \lambda_{i,v} \sigma_{i,v}(E_t) - \log Z(G_t) \right\} - \frac{\mu}{2} \sum_{i,v} \lambda_{i,v}^2 \end{aligned} \tag{11}$$

and parameters, $\boldsymbol{\lambda}$, are estimated as those that maximize $l(\boldsymbol{\lambda} \mid D)$. Currently we use L-BFGS [17] to maximize $l(\boldsymbol{\lambda} \mid D)$. The gradient used in the maximization is computed as

$$\begin{aligned} \frac{\partial l(\boldsymbol{\lambda}|D)}{\partial \lambda_{i,v}} &= \sum_{t=1}^{T} \left\{ \sigma_{i,v}(E_t) - \frac{\partial}{\partial \lambda_{i,v}} \log Z(G_t) \right\} - \mu \lambda_{i,v} \\ &= \sum_{t=1}^{T} \left\{ \sigma_{i,v}(E_t) - E(\sigma_{i,v} \mid G_t) \right\} - \mu \lambda_{i,v}. \end{aligned} \tag{12}$$

Finally, Viterbi inference, computing the most likely output $y$ for the input $x$, or the most likely explanation $E^*_{x,y}$ for the top-goal $G_x$ is formulated as (13) in D-PRISM and computed by dynamic programming just like PRISM.

$$\begin{aligned} E^*_{x,y} &= \mathrm{argmax}_{E_{x,y} \in \phi(G_x)} p(E_{x,y} \mid G_x) \\ &= \mathrm{argmax}_{E_{x,y} \in \phi(G_x)} \sum_{i,v} \lambda_{i,v} \sigma_{i,v}(E_{x,y}) \end{aligned} \tag{13}$$

## 5 Experiments

In this section, we conduct learning experiments with CRFs. CRFs encoded by D-PRISM programs and their generative counterpart encoded by PRISM programs are used to compare their accuracy in discriminative tasks. We consider three basic models, logistic regression, a linear-chain CRF and a CRF-CFG.

### 5.1 Logistic regression with UCI datasets

We select four datasets with no missing data from the UCI Machine Learning Repository [18] and compare prediction accuracy, one by logistic regression written in D-PRISM and the other by a naive Bayes model written in PRISM.

**Table 1.** Logistic regression: UCI datasets and accuracy

| Dataset | Size | #Class | #Attr. | D-PRISM | PRISM |
|---|---|---|---|---|---|
| zoo | 101 | 7 | 16 | 95.19%(8.16) | 95.19%(8.16) |
| car | 1728 | 4 | 6 | 93.51%(1.35) | 85.36%(2.56) |
| kr-vs-kp | 3196 | 2 | 36 | 97.56%(0.93) | 87.49%(2.06) |
| nursery | 12960 | 5 | 8 | 92.62%(0.82) | 90.20%(0.47) |

We use the program in Fig. 1 with a slight generalization for accepting arbitrary attribute sets. The result by ten-fold cross-validation is shown in Table 1 with standard deviation in parentheses. D-PRISM there means logistic regression whereas PRISM means a naive Bayes model. Except for the zoo dataset, logistic regression by D-PRISM outperforms a naive Bayes model by PRISM. The difference is statistically significant by t-test at 0.05 significance level.

### 5.2 Linear-chain CRF with the Penn Treebank

We here compare a linear-chain CRF encoded as a D-PRISM program and an HMM encoded as a PRISM program using sequence data extracted from the Penn Treebank[13]. What we actually do is to write an HMM program in PRISM for complete and incomplete data and consider it as a D-PRISM program defining a linear-chain CRF, similarly to the case of the naive Bayes and logistic regression.

For simplicity we employ as features the number of occurrences of transitions and emissions. Fig. 2 is a D-PRISM program for a CRF with two states {`s0`, `s1`} and two emission symbols {`a`, `b`}. `hmm0/2` describes complete data and corresponds to $G_{x,y}$ in (7) whereas `hmm0/1`[8] is for incomplete data and corresponds to $G_x$ in (7). Binary feature functions over a sequence $\boldsymbol{x}$ of symbols and a sequence $\boldsymbol{y}$ of states are implicitly represented by ground `msw` atoms, i.e. `msw(init,`$s$`)`, `msw(tr(`$s$`),`$s'$`)` ($s, s' \in$ {`s0`, `s1`} ) and `msw(out(`$s$`,`$e$`)` ($e \in$ {`a`, `b`} ). For example `msw(tr(`$s$`),`$s'$`)` returns 1 (true) if $\boldsymbol{y}$ contains a transition from $s$ to $s'$, else 0 (false).

We conduct a comparison of prediction accuracy by a linear-chain CRF and an HMM using the D-PRISM program in Fig. 2 with appropriate `values/2` declarations. The task is to predict the pos-tag sequence given a sentence. As learning data, we use two sets of sentences and their pos-tag sequences extracted from the Penn Treebank [13], Section-02 in the WSJ (Wall Street Journal articles) corpus referred to here as WSJ02_ALL and its subset referred to as WSJ02_16, consisting of data of length less-than or equal to 16.

Table 2 contains prediction accuracy (%) by eight-fold cross-validation in which D-PRISM means a linear-chain CRF and PRISM means an HMM, together with some statistics of data. The table clearly demonstrates that we

[8] Using "`hmm0([X0|Xs]):- hmm0([X0|Xs],_)`" to define `hmm0/1` is possible and theoretically correct but kills the effect of tabling.

```
values(init,[s0,s1]). values(tr(_),[s0,s1]). values(out(_),[a,b]).

hmm0([X0|Xs],[Y0|Ys]):- msw(init,Y0),msw(out(Y0),X0),hmm1(Y0,Xs,Ys).
hmm1(_,[],[]).
hmm1(Y0,[X|Xs],[Y|Ys]):-  msw(out(Y0),X),msw(tr(Y0),Y),hmm1(Y,Xs,Ys).

hmm0([X|Xs]):- msw(init,Y0),msw(out(Y0),X),hmm1(Y0,Xs).
hmm1(_,[]).
hmm1(Y0,[X|Xs]):- msw(tr(Y0),Y1),msw(out(Y1),X),hmm1(Y1,Xs).
```

**Fig. 2.** Simple linear-chain CRF program

can achieve considerably better (significant by t-test at 0.05 significance level) performance only by switching from PRISM to D-PRISM while using similar programs.

**Table 2.** Linear-chain CRF: Penn Treebank data and labeling accuracy

| Dataset | Size | Max-len | Ave-len | #Tags | #Words | D-PRISM | PRISM |
|---|---|---|---|---|---|---|---|
| WSJ02_16 | 1087 | 16 | 9.69 | 40 | 3341 | 83.17%(1.23) | 77.23%(1.38) |
| WSJ02_ALL | 2419 | 105 | 19.28 | 45 | 8476 | 90.60%(0.32) | 87.27%(0.29) |

### 5.3 CRF-CFG with the ATR tree corpus

We here deal with probabilistic grammars which are beyond the scope of graphical models. We compare the parsing accuracy of a CRF-CFG described by a D-PRISM program and a PCFG described by a PRISM program. We do not use features other than the number of occurrences of a rule in the parsing tree. To save space, we omit programs though they are (almost) identical.

As a dataset, we use the ATR tree corpus and its associated CFG [19]. Their statistics are shown in Table 3. After parameter learning by MLE using conditional likelihood (6) for the CRF-CFG and the one by the usual likelihood for the PCFG, we compare their parsing accuracy by eight-fold cross-validation. The task is to predict a parse tree given a sentence and the predicted parse tree is considered correct when it exactly coincides with the one for the sentence in the ATR tree corpus.

D-PRISM (resp. PRISM) in Table 3 indicates the average accuracy of the CRF-CFG model (reps. PCFG model). Apparently as in the case of [10], shifting from PRISM (PCFG) to D-PRISM (CRF-CFG) yields a considerable difference (the difference is statistically significant by t-test at 0.05 significance level).

**Table 3.** CRF-CFG: ATR corpus and parsing accuracy

| Dataset | Size | Max-len | Ave-len | #Rules | D-PRISM | PRISM |
|---|---|---|---|---|---|---|
| ATR corpus | 10995 | 49 | 9.97 | 860 | 82.74%(1.62) | 79.06%(1.25) |

## 6 Discussion and future work

There are already discriminative modeling languages for CRFs such as Alchemy [20] based on MLNs and Factorie [8] based on factor graphs. To define models, the former uses weighted clauses whereas the latter uses imperatively defined factor graphs. Both use Markov chain Monte-Carlo (MCMC) for probabilistic inference. D-PRISM differs from them in that although programs define CRFs, they are purely generative, computing output from input, and probabilities are computed by dynamic programming.

Compared to PRISM, D-PRISM has no restriction on programs such as the uniqueness condition, exclusiveness condition and independence condition [7]. Consequently non-exclusive or is permitted in a program. Also computation is allowed to fail by constraints. Of course this freedom is realized at the expense of normalization cost which may be prohibitive even when dynamic programming is possible. However thanks to the removal of restrictive conditions, D-PRISM is now more amenable to structure learning in ILP than PRISM, which is expected to open up a new line of research of learning CRFs in ILP.

Currently only binary features are allowed. Generally a binary feature $f(x, y)$ is written as a conjunction like $F_{i,v}(x, y) \wedge \texttt{msw}(i, v)$ in which $F_{i,v}(x, y)$ succeeds if-and-only-if $x$ has a feature $F_{i,v}$. Introducing arbitrary features is a future work and so is a mechanism of parameter tying.

## 7 Conclusion

We have introduced D-PRISM, a logic-based generative language for discriminative modeling. As examples show, D-PRISM programs are just PRISM programs with probabilities replaced by weights. It is the first modeling language to our knowledge that generatively defines CRFs (and their extension to probabilistic grammars). We can freely build logistic regression, linear-chain CRFs, CRF-CFGs or more complex models generatively with the same modeling cost as PRISM while achieving better performance in discriminative tasks.

## References


1. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th International Conference on Machine Learning (ICML'01). (2001) 282–289
2. Sutton, C., McCallum, A.: An introduction to conditional random fields. Foundations and Trends in Machine Learning **4**(4) (2012) 267–373

3. Ng, A., Jordan, M.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In: NIPS. (2001) 841–848
4. Gutmann, B., Kersting, K.: TildeCRF: Conditional random fields for logical sequences. In: In Proceedings of the 15th European Conference on Machine Learning (ECML-06). (2006) 174–185
5. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62** (2006) 107–136
6. Sato, T., Kameya, Y.: PRISM: a language for symbolic-statistical modeling. In: Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97). (1997) 1330–1335
7. Sato, T., Kameya, Y.: Parameter learning of logic programs for symbolic-statistical modeling. Journal of Artificial Intelligence Research **15** (2001) 391–454
8. McCallum, A., Schultz, K., Singh, S.: Factorie: Probabilistic programming via imperatively defined factor graphs. In: Advances in Neural Information Processing Systems 22. (2009) 1249–1257
9. Johnson, M.: Joint and conditional estimation of tagging and parsing models. In: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-01). (2001) 322–329
10. Finkel, J., Kleeman, A., Manning, C.: Efficient, feature-based, conditional random field parsing. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08). (2008) 959–967
11. Liang, P., Jordan, M.: An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In: Proceedings of the 25th international conference on Machine learning (ICML'08). (2008) 584–591
12. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press (1999)
13. Marcus, M., Santorini, B., Marcinkiewicz, M.: Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics **19** (1993) 313–330
14. Sneyers, J., Vennekens, J., De Schreye, D.: Probabilistic-logical modeling of music. In: Proceedings of the 8th International Symposium on Practical Aspects of Declarative Languages (PADL'06), vol.3819, LNCS. (2006) 60–72
15. Biba, M., Xhafa, F., Esposito, F., Ferilli, S.: Stochastic simulation and modelling of metabolic networks in a machine learning framework. Simulation Modelling Practice and Theory **19**(9) (2011) 1957–1966
16. Mørk, S., Holmes, I.: Evaluating bacterial gene-finding hmm structures as probabilistic logic programs. Bioinformatics **28**(5) (2012) 636–642
17. Liu, D., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical Programming **45** (1989) 503–528
18. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
19. Uratani, N., Takezawa, T., Matsuo, H., Morita, C.: ATR integrated speech and language database. Technical Report TR-IT-0056, ATR Interpreting Telecommunications Research Laboratories (1994)
20. Kok,S. and Singla,P. and Richardson,M.and Domingos,P.: The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington (2005)