Notes on Stacked Graphical Learning for Efficient Inference in Markov Random Fields

Zhenzhen Kou **William W. Cohen**

January 2007
CMU-ML-07-101

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In *collective classification*, classes are predicted simultaneously for a group of related instances, rather than predicting a class for each instance separately. Collective classification has been widely used for classification on relational datasets. However, the inference procedure used in collective classification usually requires many iterations and thus is expensive. We propose *stacked graphical learning*, a meta-learning scheme in which a base learner is augmented by expanding one instance's features with predictions on other related instances. Stacked graphical learning is efficient, especially during inference, capable of capturing dependencies easily, and can be implemented with any kind of base learner. In experiments on eight datasets, stacked graphical learning is 40 to 80 times faster than Gibbs sampling during inference. We also give theoretical analysis to better understand the algorithm.

Keywords: machine learning, stacked graphical learning, collective classification

1 Introduction

Traditional machine learning methods assume that instances are independent, while in reality there are many relational datasets, such as hyperlinked webpages, scientific literature with dependencies among citations, and social networks. The dependencies among instances in relational data can be complex.

Collective classification has been widely used for classification on relational datasets. In collective classification, classes are predicted simultaneously for a group of related instances, rather than predicting a class for each instance separately. Recently there have been studies on relational models for collective inference, such as relational dependency networks [1], relational Markov networks [2], and Markov logic networks[3]. Collective classification can be formulated as an inference problem over graphical models. Consider collective classification in the context of Markov random fields (MRFs). Inference in MRFs is intractible, in the general case. One common scheme for approximate inference is *Gibbs sampling*[4]. Gibbs sampling for an MRF with parameters learned to maximize pseudo-likelihood is closely related to *conditional dependency networks* [4]. However, Gibbs sampling usually takes many iterations to converge and thus graphical models are usually expensive, especially when exact inference is infeasible.

We propose a meta-learning method, *stacked graphical learning*, for learning and inference on relational data. In stacked graphical learning, a base learner is augmented by providing the predicted labels of related instances. First a base learner is applied to the training data in a cross-validation-like way to make predictions. Then we expand the features by adding the predictions of related examples into the feature vector. Finally the base learner is applied to the expanded feature set to obtain a stacked model.

One advantage of stacked graphical learning is that the inference is very efficient. Experimental results show that compared to Gibbs sampling, stacked graphical learning can achieve similar performance in one or two iterations, while Gibbs sampling usually converges only after 100 iterations.

In stacked graphical learning, the dependencies among data can be captured easily using a *relational template* which finds the related instances given one example. Stacked graphical learning can be implemented with any base learning algorithm, i.e., the base learner does not have to be a graphical model. Stacked graphical learning is also easy to implement.

2 Algorithm

2.1 Stacked Graphical Learning

We consider here collective classification tasks, in which the goal is to "collectively" classify some set of instances. In our notation, a dataset is $D = \{(\mathbf{x}, \mathbf{y})\}$. An instance is a pair of $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ is itself a high-dimensional feature vector and $\mathbf{y}$ is a label from a small set $\mathcal{Y}$. In this paper we use upper case letters such as Y for random variables and their bold-faced equivalents (e.g., $\mathbf{Y}$) for vectors of random variables. We use lower case letters for concrete assignments to these variables.

We consider a model that captures the dependency by expanding the feature of an instance x_i

- Parameters: a relational template C and a cross-validation parameter J.
- Learning algorithm: Given a training set $D = \{(\mathbf{x}, \mathbf{y})\}$ and a base learner A:
  - Learn the local model, i.e., when $k = 0$:
    Let $f^0 = A(D^0)$. Please note that $D^0 = D, \mathbf{x}^0 = \mathbf{x}, \mathbf{y}^0 = \mathbf{y}$.
  - Learn the stacked models, for $k = 1...K$:
    1. Construct cross-validated predictions $\hat{\mathbf{y}}^{k-1}$ for $\mathbf{x} \in D^{k-1}$ via calling the subroutine in Figure 2.
    2. Construct an extended dataset $D^k = (\mathbf{x}^k, \mathbf{y})$ by converting each instance x_i to x_i^k as follows: $x_i^k = (x_i, C(x_i, \hat{\mathbf{y}}^{k-1}))$, where $C(x_i, \hat{\mathbf{y}}^{k-1})$ will return the predictions for examples related to x_i such that $x_i^k = (x_i, \hat{y}_{i_1}^{k-1}, ..., \hat{y}_{i_L}^{k-1})$.
    3. Let $f^k = A(D^k)$.
- Inference algorithm: given $\mathbf{x}$:
  1. $\hat{\mathbf{y}}^0 = f^0(\mathbf{x})$.

  For $k = 1...K$,
  2. Carry out Step 2 above to produce $\mathbf{x}^k$.
  3. $\mathbf{y}^k = f^k(\mathbf{x}^k)$.

  Return $\mathbf{y}^K$.

Figure 1: Stacked Graphical Learning and Inference

with "predicted" labels for the related instances. We use predicted labels instead of true labels since during inference there is no way to get true labels. We use a *relational template C* to pick up the related instances. A relational template is a procedure that finds all the instances related to a given example and returns their indices. For instance x_i, $C(x_i)$ retrieves the indices $i_1, ..., i_L$ of instances $x_{i_1}, ..., x_{i_L}$ that are related to x_i. Given predictions $\hat{\mathbf{y}}$ for a set of instances $\mathbf{x}$, $C(x_i, \hat{\mathbf{y}})$ returns the predictions on the related instances, i.e., $\hat{y}_{i_1}, ..., \hat{y}_{i_L}$. Since the relation between x_i and x_j might be one-to-many, for example, webpages link to different numbers of webpages, we allow aggregation functions to combine predictions on a set of related instances into a single feature.

One practical difficulty to obtain predictions for training examples is that, while learning methods produce reasonably well-calibrated probability estimates on unseen test data, their probability estimates on *training* data are biased. Thus, to obtain the "predictions" for training examples, we apply a cross-validation-like technique suggested by a meta-learning scheme, *stacking* [5]. The procedure to obtain the predictions for training examples is shown in Figure 2.

Given a training set $D = \{(\mathbf{x}, \mathbf{y})\}$ and a base learner A, construct cross-validated predictions $\hat{\mathbf{y}}$ for $\mathbf{x} \in D$ as follows:

1. Split D into J equal-sized disjoint subsets $D_1 ... D_J$.
2. For $j = 1...J$, let $f_j = A(D - D_j)$. That is, train a classifier f_j, based all the data from D except the subset D_j.
3. For $\mathbf{x} \in D_j, \hat{\mathbf{y}} = f_j(\mathbf{x})$. That is, for data in D_j, apply the classifier f_j to obtain its prediction.

Figure 2: A cross-validation-like technique to obtain predictions for training examples

Finally we end up with the inference and learning methods of Figure 1 for collective classification. The relational template can be extended to include aggregation functions based on $\hat{\mathbf{y}}$ and x_i. We will demonstrate the use of this algorithm and aggregations in Section 3.1.

3 Experimental Results

3.1 Datasets

We evaluated stacked graphical learning on several classification problems. The first problem we studied is the task of text region detection in the system called the Subcellular Location Image Finder (SLIF) [8, 9]. SLIF is a system which extracts information from both figures and the associated captions in biological journal articles. Usually there are multiple *panels* (independently meaningful sub-figures) within one figure. Finding the text regions, i.e., the regions in panels containing their labels, is one important task in SLIF. The problem studied in this paper is to classify if the candidate regions found via image processing are text regions or not. The text region detection dataset contains candidate regions found in 1070 panels from 207 figures.

There are dependencies among the locations of candidate regions. Intuitively, if after image processing a candidate text region was found at the upper-left corner of panel B and two candidate

regions were found in panel A, one located at the upper-left corner, another in the middle, it is more likely the candidate region at the upper-left of panel A is the real text region. We define the *neighbor* of a candidate text region to be the region located in the "same" position in adjacent panels in the same figure and consider the neighbors on four directions (left, right, up, and down). We also consider the dependency among candidate regions within the same panel, called *competitors*. Figure 3 is an example figure in SLIF which demonstrates candidate regions, neighbors and competitors.

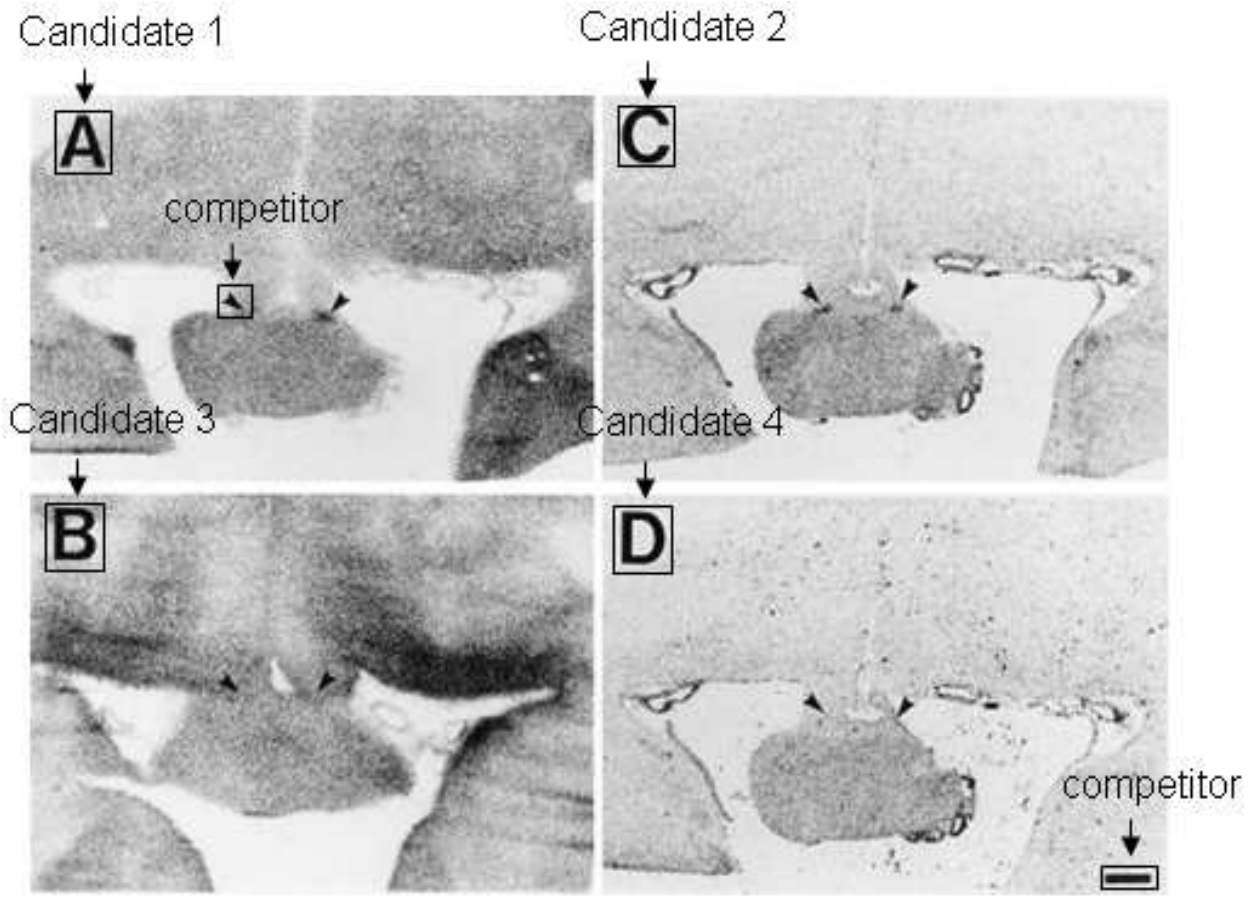

Figure 3: An example figure in SLIF

Let x_i be a candidate region, $\mathbf{x}$ be a vector of candidate regions from one figure. The relational template returns the predictions on x_i's neighbors and competitors. If a neighbor does not exist, 0 is assigned to the corresponding feature. Since one candidate region can have several competitors from the same panel, we apply an EXISTS aggregator to the competitors, i.e., as long as there is one competitor which is predicted to be a text region, we assign 1 to the corresponding feature added during stacking. For instance, considering Candidate 1 in Figure 3, if Candidate 2(right neighbor) has been predicted as 1, Candidate 3(down neighbor) as 1, and the competitor as 0, $C(x_i, \hat{\mathbf{y}})$ returns (1, 0, 0, 1, 0).

We use a maximum entropy learner as the base learner. The features for the base learner are obtained via image processing and contain binary features indicating whether Optical Character Recognition(OCR) extracts a character or not from the candidate region and its neighbors [8].

The second problem is the document classification problem. We consider the webpage classification on the WebKB dataset[10], which contains webpages from four computer science departments, and paper classification on the Cora dataset and the CiteSeer dataset[11]. The WebKB data contains aproximately 3800 webpages labelled from 6 categories and 8000 hyperlinks. The relational template applies the COUNT aggregator and returns the number of outgoing and incoming links in each category, given one webpage. The Cora data[12] contains 2708 papers labelled from seven categories and 5429 citations. If paper A cites paper B, we consider there is a link from paper A to paper B. The relational template applies the COUNT aggregator and returns the number of outgoing and incoming links in each category to one paper. The Citeseer data[13] contains 3312 papers labelled from six categories and 4732 citations. The relational template is the same as the template for the Cora data.

We use a maximum entropy learner as the base learner in stacked graphical learning for document classification and a bag-of-word feature set.

The third problem we study is named entity extraction from Medline abstracts and emails. We used three datasets to evaluate our method for protein name extractions. The University of Texas, Austin dataset contains 748 labeled abstracts[14]; the GENIA dataset contains 2000 labeled abstracts[15]; and the YAPEX dataset contains 200 labeled abstracts[16]. We also study person name extraction from the email message corpus. The CSpace corpus we used in this paper contains 216 email messages collected from a management course at Carnegie Mellon University[17].

We use conditional random fields [18] as the base learner and the feature set described in our previous paper [19] for protein name extraction and the feature set described in [20] for person name extraction. The relational template will retrieve the predictions for the nearby words (with window size 3) and for the same word appearing in one abstract, apply the COUNT aggregator, and return the number of words in each category, given one word. That is, let x_i be the word in a document. For words $x_j = x_i$ in the same document, we count the number of times x_j appearing with label y and use it as one of the stacked features for x_i.

3.2 Accuracy of Stacked Graphical Learning

To evaluate the effectiveness of stacked graphical learning, we compare five models. The first model is a competitive graphical model. For the SLIF and document classification problems, we compare to relational dependency network (RDN) models [1]. The RDN model uses the same features as the stacked model, but learns via a pseudo-likelihood method, and does inference with Gibbs sampling, which usually converge after 100 iterations. For name extraction, we compare to a stacked sequential CRF model [7]. The second model is a local model, i.e., the model trained with the base learner. For the SLIF and document classification problem, the local model is a MaxEnt model. For the name extraction, the local model is a CRF model. The third and fourth models are stacked graphical models. The fifth model is a probabilistic upper-bound (noted as ceiling model in Table 1) for the stacked graphical model, i.e., we use the stacked graphical model but allow true labels of related instances to be added during the feature extension at both training and testing time.

Table 1: Evaluation on five models. The accuracy for "SLIF" and "Document classification" and F1-measure for named entity extraction are reported. We compared stacked graphical model to a local model, another relational model, and its probabilistic ceiling. The local models for "SLIF" and "Document classification" are MaxEnt models, and the local models for "named entity extraction" are CRFs. The competitive relational models for "SLIF" and "Document classification" are RDN models, and the competitive relational models for "named entity extraction" are stacked-CRFs.

	SLIF	Document classification			Named Entity Extraction			
		WebKB	Cora	CiteSeer	UT	Yapex	Genia	CSpace
Local model								
MaxEnt	77.2	58.3	63.9	55.3	-	-	-	-
CRFs	-	-	-	-	73.1	65.7	72.0	80.3
Competitive Model								
RDNs	86.7	**74.2**	72.9	58.7	-	-	-	-
Stacked Sequential CRFs	-	-	-	-	76.8	66.8	77.1	81.2
Stacked model (k=1)	**90.1**	73.2	73.8	**59.8**	78.3	**69.3**	77.9	**82.5**
Stacked model (k=2)	**90.1**	72.1	**73.9**	**59.8**	**78.4**	69.2	**78.0**	82.4
Ceiling for stacked model	96.3	73.6	76.9	62.3	80.5	70.5	80.3	84.6

This can not be implemented in practice but gives some idea of what performance is theoretically achievable using collective classification with our model.

Table 1 shows the accuracy[1] for each of the five models on eight real-world datasets. We used 5 fold cross validation (except for WebKB data, where we used 4 fold cross validation by departments).

We use paired t-tests to access the significance of the accuracy. The t-tests compare the stacked graphical models with k=1 to each of the other four models. The null hypothesis is that there is no difference in the accuracy of the two models. On all of the eight datasets, stacked graphical learning improves the performance of the base learner significantly ($p < .05$). On all the tasks, stacked graphical learning achieves statistically indistinguishable results to the competitive models, except that on the SLIF data stacked graphical learning is statistically significantly better than RDNs. On the WebKB and Yapex datasets, stacked graphical learning achieves comparable results to the ceiling models.

On all the tasks, there is no significant difference ($p < .05$) in accuracy for k=1 and k=2 in stacking, which suggests that stacking converges very quickly and does not require many iterations.

3.3 Efficiency of Stacked Graphical Learning

One advantage of stacked graphical learning is that the inference is very efficient. We compared the accuracy and computational cost of inference in stacked graphical models (with one iteration) to that of Gibbs sampling in RDNs with 50 iterations and 100 iterations, evaluating on the SLIF prob-

[1]For the third and fourth problem, we report the F1 accuracy.

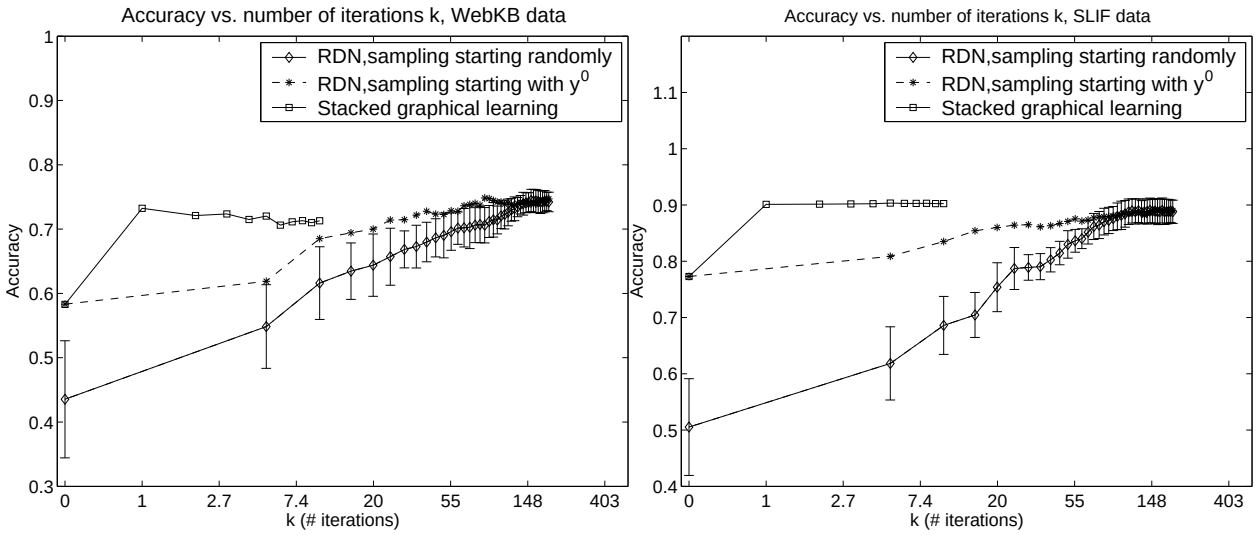

Figure 4: Convergence rate of stacking and Gibbs sampling

lem and the document classification problems, to demonstrate this. Table 2 shows the speedup, i.e., in the table "39.6" means the inference in stacked graphical learning is 39.6 times faster than Gibbs sampling. If the accuracy of stacked graphical learning is statistically significantly better than the accuracy of Gibbs sampling, there is a "+" marked by the number indicating the speedup. If there is no significant difference, there is no mark.

Table 2 shows that compared to Gibbs sampling with 50 iterations, stacked graphical learning generally achieves better accuracy but is about 40 times faster during inference. Compared to Gibbs sampling with 100 iterations, stacked graphical learning can achieve competitive or better accuracy but is more than 80 times faster during inference.

Table 2: Comparison on performance and efficiency. "39.6" means that the inference in stacked graphical learning is 39.6 times faster than Gibbs sampling. "+" means that the accuracy of stacked graphical learning is statistically significantly better than the accuracy of Gibbs sampling.

	Gibbs 50	Gibbs 100
SLIF	39.6^+	79.3^+
WebKB	43.4^+	87.0
Cora	42.7^+	85.4
Citeseer	43.6^+	87.3
Average speed-up	42.3	84.8

Figure 4 shows the convergence rate of stacking compared to Gibbs sampling on RDNs. The plots were generated using SLIF data and WebKB data. We run 10 iterations of stacking and 150 iterations of Gibbs sampling on RDNs and recorded the accuracy. We created the plots using a natural logarithmic scale of iteration number k. In addition to the Gibbs sampling with random

starting points, we also evaluated Gibbs sampling starting with same $\mathbf{y}^0$ as the corresponding stacked graphical models, i.e., with predictions of local models as starting points.

We observe that stacked models converge more quickly than Gibbs sampling and achieve a satisfactory performance much faster, even if the Gibbs sampling starts with same $\mathbf{y}^0$ as the corresponding stacked graphical models. Stacked graphical models can achieve significant improvement over the base learner after the first iteration. More iterations of stacking do not seem to be more helpful, with the performance staying at about the same level. We observe that Gibbs sampling converges to the same level after many more iterations and the convergence rate when k is small depends heavily on the starting points. We plot error bars along the curve for Gibbs sampling with random starting points. The error bars are calculated over 5 randomly initial samples, i.e., in each fold, Gibbs sampling is run 5 times with random initial starting points.

4 Formal Analysis

The previous sections show that stacked graphical learning works well in practise. In this section we will now formally analyze an idealized version of the algorithm, in order to better understand its performance. We use upper case letters and their bold-faced equivalents, such as Y and $\mathbf{Y}$ for (vectors of) random variables. We use lower case letters for (vectors of) concrete assignments to these variables.

4.1 Gibbs sampling and dependency networks

We will assume that the data to be modeled $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$, can be generated by a homogeneous Markov chain. In other words, we assume that each $\mathbf{y}_i$ is drawn from a distribution $\pi(\mathbf{Y}|\mathbf{X} = \mathbf{x}_i)$, where $\pi(\mathbf{Y}|\mathbf{X})$ can be defined as the limit of the following process as $T \to \infty$:

1. for $i = 1 \ldots n$, pick $y_i^0 \sim \Pr(Y_i|\mathbf{X} = \mathbf{x}, \theta^0)$
2. for $t = 1 \ldots T$
   (a) for $i = 1 \ldots n$,
   pick $y_i^t \sim \Pr(Y_i|\mathbf{Y}_{-i} = \mathbf{y}_{-i}^{t-1}, \mathbf{X} = \mathbf{x}, \theta^+)$

Under relatively mild conditions, this limit will exist, and will be independent of $\mathbf{y}^0$ (and hence θ^0).[2]

In the algorithm above, $\mathbf{Y}_{-i}$ represents the values of all random variables in $\mathbf{Y}$ other than Y_i, θ^0 is some set of parameters that define the initial choice of values for y_i, and θ^+ is a set of parameters that defines a process for incrementally updating Y_i given estimated values for $\mathbf{Y}_{-i}$ and $\mathbf{x}$. We will assume that θ^+ and θ^0 are shorthand for a set of n probabilistic classifiers, one that predicts each Y_i.

[2]However, our experiments show that different choices of θ^0 do greatly effect the convergence rate of the Gibbs sampler. It is also convenient to introduce it in the context of the following remarks.

Often Y_i does not depend on all the other variables $\mathbf{Y}$, but on only a few. Let $\mathrm{MB}_i(\mathbf{Y})$ denote the *Markov blanket* of Y_i, i.e., a subset of the values from $\mathbf{Y}$ such that Y_i is conditionally independent of $\mathbf{Y}_{-i}$ given $\mathbf{x}$ and $\mathrm{MB}_i(\mathbf{Y})$:

$$\Pr(Y_i|\mathbf{Y}_{-i}, \mathbf{X} = \mathbf{x}, \theta^+) = \Pr(Y_i|\mathrm{MB}_i(\mathbf{Y}), \mathbf{X} = \mathbf{x}, \theta^+)$$

To simplify our notation let us introduce these abbreviations:

$$\begin{aligned}
P_i(Y_i|\mathbf{x};\theta) &\equiv \Pr(Y_i|\mathbf{X} = \mathbf{x}, \theta) \\
P_i(Y_i|\mathbf{x},\mathbf{y};\theta) &\equiv \Pr(Y_i|\mathrm{MB}_i(\mathbf{Y}) = \mathrm{MB}_i(\mathbf{y}), \mathbf{X} = \mathbf{x}, \theta) \\
P(\mathbf{Y}|\mathbf{x};\theta) &\equiv \prod_i P_i(Y_i = y_i|\mathbf{x};\theta) \\
P(\mathbf{Y}|\mathbf{x},\mathbf{y}';\theta) &\equiv \prod_i P_i(Y_i = y_i|\mathbf{x},\mathbf{y}';\theta)
\end{aligned}$$

The process above can now be re-written as follows.

Definition 1 *Gibbs sampling is the following stochastic process:*

1. *for* $i = 1 \ldots n$, *pick* $y_i^0 \sim P_i(\mathbf{x};\theta^0)$
2. *for* $t = 1 \ldots T$
   (a) *for* $i = 1 \ldots n$, *pick* $y_i^t \sim P_i(\mathbf{x}, \mathbf{y}^{t-1};\theta^+)$

(The limit of) Gibbs sampling is one means of approximating inference in a conditionally-defined Markov random field. It is intuitively appealing, as it simply requires iteratively applying and re-applying a set of conditional probability models—such as could be learned by logistic regression, probabilistic decision trees, probabilistic SVMs, etc—each of which predicts a single variable Y_i from $\mathbf{x}$ and some set of "related" variables, as defined by MB_i.

To make these definitions more concrete, let us use them to define the following well-known learning method.

Definition 2 *The pseudo-likelihood learning method [22] for a Gibbs sampler and a class of models $\mathcal{M}$ is defined as follows.*

$$\begin{aligned}
\hat{\theta}^0 &= \mathit{argmax}_{\theta\in\mathcal{M}} \prod_{(\mathbf{x},\mathbf{y})\in D} P(\mathbf{y}|\mathbf{x},\theta) \\
\hat{\theta}^+ &= \mathit{argmax}_{\theta\in\mathcal{M}} \prod_{(\mathbf{x},\mathbf{y})\in D} P(\mathbf{y}|\mathbf{x},\mathbf{y},\theta) \\
&= \mathit{argmax}_{\theta\in\mathcal{M}} \prod_{(\mathbf{x},\mathbf{y})\in D} \prod_i P_i(y_i|\mathbf{x},\mathbf{y},\theta)
\end{aligned}$$

The "argmax" here means that $\hat{\theta}^+$ will be the maximum-likelihood (ML) model in $\mathcal{M}$ for each Y_i. The optimization over $\Pr(y_i|\mathrm{MB}_i(\mathbf{y}),\mathbf{x},\theta)$ means that in the ML optimization used to train each probabilistic classifier for Y_i, the values for the "related" values MB_i will be taken from values for $\mathbf{y}$ seen in the training data D. The model learned by this method is sometimes called a *dependency network* [4].

4.2 An idealized version of stacked graphical learning

Pseudo-likelihood/dependency net learning approximates the unknown Gibbs sampling parameters θ^0, θ^+ with estimates $\hat{\theta}^0$, $\hat{\theta}^+$ that will be used in a Gibbs sampler of the same form. We propose to approximate the unknown Gibbs sampler with a different samping procedure:

Definition 3 *Inhomogeneous Gibbs sampling is the following stochastic process:*

1. *for* $i = 1 \ldots n$, *pick* $y_i^0 \sim P_i(\mathbf{x}; \theta^0)$
2. *for* $k = 1 \ldots K$
   (a) *for* $i = 1 \ldots n$, *pick* $y_i^k \sim P_i(\mathbf{x}, \mathbf{y}^{k-1}; \theta^k)$

Note that this sampling procedure is defined by $K+1$ sets of parameters, $\theta^0, \theta^1, \ldots, \theta^K$, rather than only two. We will use $Q^k(\mathbf{x}; \theta^0, \ldots, \theta^k)$ to denote the distribution produced after k iterations of inhomogeneous Gibbs sampling.

Clearly, inhomogeneous Gibbs samplers include all ordinary Gibbs samplers, since it could be that K is large and $\theta^1 = \theta^2 = \ldots = \theta^+$. Hence inhomogeneous samplers produce a larger class of distributions. There are several potential advantages of considering this larger class. First, there may be practical problems that are better approximated by the larger class. Second, it is possible that even distributions generated by a long homogeneous Gibbs sampler can be well-approximated by a *short* inhomogeneous Gibbs sampler—notice that a short inhomogeneous Gibbs sampler can be executed quickly.

Finally, the larger class of inhomogeneous samplers may be computationally more efficient to learn. This may seem counter-intuitive, but recall that there are many instances of learning tasks which are made computationally easier by expanding the class of possible models (e.g., [23]).

To learn the parameters of an inhomogeneous Gibbs sampler, we propose the following method.

Definition 4 *Idealized stacked graphical learning is the following learning method.*

1. *Let* $\hat{\theta}^0 = \mathit{argmax}_{\theta \in \mathcal{M}} \prod_{(\mathbf{x},\mathbf{y}) \in D} \Pr(\mathbf{y}|\mathbf{x}, \theta)$
2. *For* $k = 1, \ldots, K$*:*
   (a) *Create* D^k *by replacing each* $(\mathbf{x}, \mathbf{y}) \in D$ *with* $(\mathbf{x}, \mathbf{y}')$ *where* $\mathbf{y}'$ *was drawn from* $Q^{k-1}(\mathbf{x}; \hat{\theta}^0, \ldots, \hat{\theta}^{k-1})$. *Let* $\mathbf{y}'^{k-1}_{\mathbf{x}}$ *denote the* $\mathbf{y}'$ *so drawn.*
   (b) *Set*
   $$\hat{\theta}^k = \mathit{argmax}_{\theta \in \mathcal{M}} \prod_{(\mathbf{x},\mathbf{y}) \in D} Q(\mathbf{y}|\mathbf{x}, \mathbf{y}'^{k-1}_{\mathbf{x}}, \theta)$$

The optimization over $\Pr(y_i|\mathrm{MB}_i(\mathbf{y}), \mathbf{x}, \theta)$ means that in the ML optimization used to train the k-th probabilistic classifier for Y_i, the values for the "related" values MB_i will be taken from values for $\mathbf{y}'$ constructed in Step 2a.

4.3 Stacked graphical learning as greedy learning of an inhomogeneous sampler

To understand the idealized stacked graphical learning method better, first notice that the learning algorithm picks $\hat{\theta}^0$ as in pseudo-likelihood training. Let us now consider how $\hat{\theta}^1$ is chosen. We will show that the method picks $\hat{\theta}^1$ so as to force the distribution $Q^1(\mathbf{x})$ to be as close as possible to the unknown $\Pr(\mathbf{Y}|\mathbf{X})$ on the data D.

Intuitively, in the unknown target inhomogeneous Gibbs sampler that we are trying to learn, θ^1 will be applied to values $(\mathbf{x}, \mathbf{y}')$ where $\mathbf{y}'$ is generated according to $Q^0(\mathbf{x}; \theta^0)$. We do not know θ^0, but we can approximate θ^0 with $\hat{\theta}^0$, and pick θ^1 by maximizing the empirical probability of $\Pr(\mathbf{y}|\mathrm{MB}_i(\mathbf{y}'), \mathbf{x}, \theta)$ where $\mathbf{y}'$ is sampled from $Q^0(\mathbf{x}; \hat{\theta}^0)$:

$$\hat{\theta}^1 = \text{argmax}_{\theta \in \mathcal{M}} \Pr(\mathbf{y}|\mathrm{MB}_i(\mathbf{y}'), \mathbf{x}, \theta)$$

The idealized stacked graphical learning method does exactly this, and then continues this process for K further iterations.

One can formalize the claim that idealized stacked graphical learning is a greedy learner for homogeneous Gibbs-sampler distributions. Let $\mathcal{M}$ be some set of parameter values θ, and let $\mathcal{P}^k_{\mathcal{M}}$ be the set of all distributions defined by inhomogeneous Gibbs samplers using parameters $\theta^0, \ldots, \theta^k$ from $\mathcal{M}$. Let $\mathcal{P}^k_{\mathcal{M}}(\tilde{\theta}^0, \ldots, \tilde{\theta}^{k-1}, *)$ be the set of all distributions defined by inhomogeneous Gibbs samplers using parameters $\theta^0 = \tilde{\theta}^0, \ldots, \theta^{k-1} = \tilde{\theta}^{k-1}$ and $\theta^k \in \mathcal{M}$. We have the following claim.

Theorem 1 *Let $\pi \in \mathcal{P}^K_{\mathcal{M}}$, and let dataset D be generated by picking each $\mathbf{x}$ from a fixed distribution, and picking the associated $\mathbf{y}$ according to $\pi(\mathbf{Y}|\mathbf{x})$. Let $\hat{\theta}^0, \ldots, \hat{\theta}^K$ be the parameters learned by idealized stacked graphical learning from dataset D, and consider the limit as $|D| \to \infty$. For all $i : 0 \leq i \leq K$,*

$$Q^i(\hat{\theta}^0, \ldots, \hat{\theta}^i) = \textit{argmin}_{Q \in \mathcal{P}^i_{\mathcal{M}}(\hat{\theta}^0, \ldots, \hat{\theta}^{i-1}, *)} KL(Q||\pi)$$

where $KL(Q||P)$ is the KL-divergence of Q and P.

Proof. Every distribution in $\mathcal{P}^i_{\mathcal{M}}(\hat{\theta}^0, \ldots, \hat{\theta}^{i-1}, *)$ is defined by an inhomogeneous Gibbs sampler with parameters $\hat{\theta}^0, \ldots, \hat{\theta}^{i-1}$ and some $\theta^i \in \mathcal{M}$. Let $\tilde{\pi}_{i-1}$ denote $Q^i(\hat{\theta}^0, \ldots, \hat{\theta}^{i-1})$, and let $\tilde{\pi}_i \in \mathcal{P}^i_{\mathcal{M}}(\hat{\theta}^0, \ldots, \hat{\theta}^{i-1}, *)$. For any such $\tilde{\pi}_i$ and any $\mathbf{x}$, $\tilde{\pi}_i(\mathbf{y}|\mathbf{x})$ is defined by

$$\tilde{\pi}_i(\mathbf{y}|\mathbf{x}) = \tilde{\pi}_{i-1}(\mathbf{y}'|\mathbf{x}) \cdot P(\mathbf{y}|\mathbf{x}, \mathbf{y}'; \theta^i)$$

In the limit as $|D| \to \infty$, minimizing KL-divergence to P is equivalent to maximizing the probability of a dataset drawn from from the distribution P. Thus minimizing $KL(\pi_i||\pi)$ can be accomplished by choosing θ^i to maximize probability of

$$\prod_{(\mathbf{x}, \mathbf{y}, \mathbf{y}') \in D'} P(\mathbf{y}|\mathbf{x}, \mathbf{y}'; \theta^i)$$

where $\mathbf{y}$ is drawn from $\pi(\mathbf{Y}|\mathbf{x})$ and $\mathbf{y}'$ is drawn from $\tilde{\pi}_{i-1}(\mathbf{Y}|\mathbf{x})$. Notice that Step 2a of idealized stacked graphical learning constructs the appropriate $\mathbf{y}'$ for $\mathbf{x}$, and Step 2b performs the appropriate optimization.

Notice that idealized stacked graphical learning is nearly identical to the algorithm used in this paper, if we assume that the relational template $C(x_i, \mathbf{y})$ returns $\mathrm{MB}_i(\mathbf{y})$. The main differences are the use of most-likely predictions from cross-validation rather than sampling to produce values $\mathbf{y}'^k_{\mathbf{x}}$, and the use of aggregation functions in our implementation.

5 Conclusions

In this paper we presented stacked graphical learning, a meta-learning scheme in which a base learner is augmented by expanding one instance's features with predictions on other related instances. Formally stacked graphical learning can be viewed as approximating a homogeneous Markov chain by greedily extending a short inhomogeneous Markov chain.

Compared to other graphical models, stacked graphical learning is efficient, especially during inference. This property allows it to be very competitive in applications where an efficient inference algorithm is extremely important. The evaluations on eight real-world datasets indicate that classification with stacked graphical models can improve the performance of a base learner significantly and achieve accuracy competitive to other graphical models via much faster inference.

In this paper, we extend the stacked sequential model[7] to a more general case, where relational data is considered as the application. Krishnan and Manning[6] independently developed a "two stage" learning method for Named Entity Recognition, in which predictions from one CRF are used to generate predictions for another. This method is like Cohen and Carvalho's stacked CRFs [7], but in Krishnan and Manning's experiments, they used different functions to aggregate the predictions of the base classifier. Stacked graphical models are a generalization of Krishnan and Manning's method. McCallum and Sutton introduced parameter independence diagrams for introducing additional independence assumptions into parameter estimation for efficient training of undirected graphical models[26]. Their method obtained a gain in accuracy via training in less than one-fifth the time. Our work is focusing on an approach which is efficient in inference. In paper [27], we described an in-depth study of stacked graphical learning and applied it to the matching of two inter-related sub-tasks in SLIF system.

Future work will compare stacked models to more graphical models such as relational Markov networks, and further explore relational template design and base learner selection. For example, integrating an online learning algorithm will enable fast training of stacked graphical models. We are also considering more applications of stacked graphical learning to inter-related classification problems in an information extraction system.

References

[1] D. JENSEN AND J. NEVILLE , *Dependency Networks for Relational Data*, Proceedings of ICDM-04, Brighton, UK 2004.

[2] B. TASKAR AND P. ABBEEL AND D. KOLLER , *Discriminative Probabilistic Models for Relational Data*, Proceedings of UAI-02, Edmonton, Canada, 2002.

[3] M. RICHARDSON AND P. DOMINGOS, *Markov Logic Networks*, Machine Learning, 62, pp107–136, 2006.

[4] D. HECKERMAN, ET AL., *Dependency Networks for Inference, Collaborative Filtering, and Data Visualization*,Journal of Machine Learning Research, 1, pp49–75, 2000.

[5] D. H. WOLPERT, *Stacked generalization*, Neural Networks, vol. 5, pp241–259, 1992.

[6] V. KRISHNAN AND C. D. MANNING, *An Effective Two-Stage Model for Exploiting Non-Local Dependencies in Named Entity Recognition*, Proceedings of Coling/ACL2006, Sydney, Australia, 2006.

[7] V. R. CARVALHO AND W. W. COHEN, *Stacked Sequential Learning*, Proceedings of Proceedings of the IJCAI-05, Edinburgh, Scotland, 2005.

[8] Z. KOU, W. W. COHEN AND R. F. MURPHY, *Extracting Information from Text and Images for Location Proteomics*, Proceedings of the BIOKDD 2003.

[9] R. F. MURPHY, Z. KOU, J. HUA, M. JOFFE AND W. W. COHEN, *Extracting and Structuring Subcellular Location Information from on-line Journal Articles: the Subcellular Location Image Finder*, Proceedings of KSCE-04, St. Thomas, US Virgin Islands, 2004.

[10] M. CRAVEN, ET AL., *Learning to Extract Symbolic Knowledge from the World Wide Web*, Proceedings of AAAI-98, Madison, WI, 1998.

[11] Q. LU AND L. GETOOR, *Link-based Classification*, Proceedings of ICML-03, Washington, DC, 2003.

[12] A. MCCALLUM, K. NIGAM, J. RENNIE, AND K. SEYMORE, *Automating the construction of internet portals with machine learning*, Information Retrieval, 3, pp127-63.

[13] C. L. GILES, K. BOLLACKER, AND S. LAWRENCE, *Cite- Seer: An automatic citation indexing system*, ACM Digital Libraries 98.

[14] R. BUNESCU, ET AL., *Comparative Experiments on Learning Information Extractors for Proteins and their Interactions*, Artificial Intelligence in Medicine, 33, 2 (2005), pp 139-155.

[15] N. COLLIER, ET AL., *The GENIA project: Corpus-based knowledge acquisition and information extraction from genome research papers*, Proceedings of EACL-99, pp271-272.

[16] K. FRANZÉ, ET AL., *Protein names and how to find them*, International Journal of Medical Informatics, 2002, 67(1-3), pp49-61.

[17] W. W. COHEN AND S. SARAWAGI, *Exploiting Dictionaries in Named Entity Extraction: Combining Semi-Markov Extraction Processes and Data Integration Methods*, Proceedings of KDD 2004.

[18] J. LAFFERTY, A. MCCALLUM AND F. PEREIRA, *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*, Proceedings of ICML-2001, Williams, MA 2001.

[19] Z. KOU, W. W. COHEN, AND R. F. MURPHY, *High-Recall Protein Entity Recognition Using a Dictionary*, Proceedings of ISMB 2005.

[20] W. W. COHEN AND S. SARAWAGI, *Exploiting Dictionaries in Named Entity Extraction: Combining Semi-Markov Extraction Processes and Data Integration Methods*, Proceedings of KDD 2004.

[21] R. NEAL, *Probabilistic inference using Markov chain Monte Carlo methods*, CRGTR -93-1, Department of Computer Science, University of Toronto, 1993.

[22] J. BESAG, *Efficiency of pseudolikelihood estimation for simple Gaussian fields*, Biometrika, 64, pp616–618, 1977.

[23] R. L. RIVEST, *Learning Decision Lists*, Machine Learning, vol. 1, no. 2, pp229–246, 1987.

[24] V. CARVALHO AND W. W. COHEN , *On the Collective Classification of Email Speech Acts*, Proceedings of SIGIR 2005.

[25] Z. KOU AND W. W. COHEN, *Notes on Stacked Graphical Learning for Efficient Inference in Markov Random Fields*, ML-07-100, Machine Learning Department, Carnegie Mellon University, 2007.

[26] A. MCCALLUM AND C. SUTTON, *Piecewise Training of Undirected Models*, Proceedings of UAI 2005.

[27] Z. KOU, W. W. COHEN, R. F. MURPHY, *A Stacked Graphical Model for Associating Sub-Images with Sub-Captions*, Proceedings of PSB 2007.