Stacked Graphical Learning: Learning in Markov Random Fields using Very Short Inhomogeneous Markov Chains

William W. Cohen and **Zhenzhen Kou**
Machine Learning Dept
Carnegie Mellon University, Pittsburgh, PA USA 15213

Abstract

We described *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. The stacked graphical learning is efficient, especially during inference, capable of capturing dependencies easily, and can be constructed based on any kind of base learner. In experiments on two classification problems, stacked graphical learning generally achieved comparable accuracy to other graphical models via much faster inference.

1 Introduction

Traditional machine learning methods assume that instances are independent while in reality there are many relational datasets, such as hyperlinked webpages, scientific literatures with dependencies among citations, and social networks. The dependencies among data are complex. The instances could also have varying structures, for example, papers have different numbers of authors.

Graphical models are attractive tools to model the dependency among variables. Some graphical models such as Bayesian networks [10] and Conditional Random Fields [8] have fixed structures. Recently there have been studies on relational graphical models, such as relational dependency networks [6], and relational Markov networks [12, 1]. Compared to the traditional model, relational graphical models can capture the relations among instances, and do not require a fixed graphical structure.

Learning and inference of graphical models are usually expensive, especially when exact inference is infeasible. We proposed a meta-learning method, *stacked graphical learning*, for the learning and inference on relational data. In stacked graphical learning, a base learner is augmented by providing the predicted labels of related instances. That is, first, a base learner is applied to the training data 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.

A stacked graphical model is a form of short *inhomogeneous* Markov chains and does not require a fixed structure. One advantage of stacked graphical learning is that the inference is not iterative and thus very efficient. In relational domains, the dependencies can be captured easily using a *relational template* which finds the related instances given one example. Stacked graphical learning can be constructed based on any base learning algorithm, i.e., the base learner does not have to be a graphical model. Stacked graphical learning is easy to implement.

2 Preliminaries

We consider here collective classification tasks, in which the goal is to "collectively" classify some set of instances. In our notation, a collection of instances is a vector $\mathbf{x}$, and the labels for $\mathbf{x}$ are encoded as a parallel vector $\mathbf{y}$. Hence a collection is a pair $(\mathbf{x}, \mathbf{y})$ where each element of $\mathbf{x}$ is an instance–i.e., is itself a high-dimensional vector—and each element of $\mathbf{y}$ is a label from a small set $\mathcal{Y}$.

Collective classification can be formulated as an inference problem over graphical models. Here we will consider collective classification in the context of Markov random fields (MRFs). Concretely, we assume the final model is a Markov field in which each label y_i depends on the labels $y_{i_1}, \ldots, y_{i_L}$ of some set of "neighboring" or "related" instances, as well as the instance x_i. We let $\mathbf{Y}_{-i}$ denote the set of all variables with indices $\{j : j \neq i\}$. We will let MB_i denote variables in the Markov blanket (set of related instances) for y_i, i.e., the set such that $\Pr(Y_i|\mathbf{Y}_{-i}, \mathbf{X}) = \Pr(Y_i|\mathrm{MB}_i, \mathbf{X})$.

Here 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. When it is necessary to make a distinction, MB_i will denote a set of random variables, and $\mathrm{MB}_i(\mathbf{y}')$ will denote the projection of the concrete assignment $\mathbf{y}'$ onto the variables in MB_i.

Inference in MRFs is intractible, in the general case. One common scheme for approximate inference is *Gibbs sampling*, in which variables Y are initialized to some starting point, and then repeatedly updated[1] until convergence:

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

Learning θ^+ from a dataset $D = \{(\mathbf{x}, \mathbf{y})\}$ is typically expensive, as it requires probabilistic inference (using Gibbs or some other procedure) in order to assess $\Pr(D|\theta)$. Typically, this inference is the inner loop of some optimization procedure. An alternative approach which is often used is to maximize the *pseudo-likelihood* of the data, i.e., let θ^+ be

$$\theta^+ = \mathrm{argmax}_\theta \prod_{(\mathbf{x},\mathbf{y})\in D} \prod_{i=1}^{n} \Pr(Y_i|\mathbf{Y}_{-i}, \mathbf{x}, \theta) \tag{1}$$

This is quite tractible, as it simply requires learning separate conditional models for each i: for instance, the model for each Y_i might be a logistic regression (aka maximum entropy) model depending on some set of features $F_i(\mathbf{x}, \mathbf{y})$ that is computed using only y_i, $\mathbf{x}$, and $\mathrm{MB}_i(\mathbf{y})$. Gibbs sampling for an MRF with parameters learned to maximize pseudo-likelihood is closely related to *conditional dependency networks* [5].

The starting point for the Gibbs process, as given here, is defined by the parameters θ^0. One common choice is a θ^0 that assigns random values to $\mathbf{y}^0$. Another plausible choice for θ^0 is an ML or MAP estimate for parameters of some model which is restricted to the "non-relational" features—i.e., in the subset of features from F_i that depend on y_i and $\mathbf{x}$, but not MB_i. This MAP would also be learned from a dataset, i.e., θ^0 would be chosen as

$$\theta^0 = \mathrm{argmax}_\theta \prod_{(\mathbf{x},\mathbf{y})\in D} \prod_{i=1}^{n} \Pr(Y_i|\mathbf{x}, \theta) \tag{2}$$

3 Short inhomogeneous Markov chains

3.1 The general case

One disadvantage of Gibbs sampling is its computational cost. Assume that we wish to limit inference time by restricting the number of iterations performed by Gibbs to some small number K,

[1] This version of Gibbs sampling is slightly non-standard. It is more usual to base y_i^t on the most recently computed values of $\mathbf{y}$—i.e., to use y_j^t rather than y_j^{t-1} for $j < i$. However, similar asymptotic convergence properties hold for this version.

where $K \ll T$. To compensate for this severe restriction, we propose to allow the parameters used at each stage t of the sampling procedure to differ—i.e., we propose to use a short *inhomogeneous* Markov chain to approximate $\Pr(\mathbf{Y}|\mathbf{X})$, rather than a long homogeneous chain. This leads to the following variant of Gibbs sampling:

- for $i = 1 \ldots n$, pick $y_i^0 \sim \Pr(Y_i|\mathbf{X} = \mathbf{x}, \theta^0)$
- for $k = 1 \ldots K$
  - for $i = 1 \ldots n$, pick $y_i^k \sim \Pr(Y_i|\mathbf{Y}_{-i} = \mathbf{y}_{-i}^{k-1}, \mathbf{X} = \mathbf{x}, \theta^k)$

Notice that this is identical to standard Gibbs, except that it is based on $\theta^0, \theta^1, \ldots, \theta^K$ rather than θ^0, θ^+. We begin by posing the question: can one efficiently learn an MLE estimate of the parameters $\theta^0, \theta^1, \ldots, \theta^K$ above? We claim that the answer to this question is "no", even if the Markov blanket of every Y_i is small. This can be demonstrated by viewing the series of models at $k = 1 \ldots K$ as a dynamic Bayes network (DBN) with $t = 1 \ldots K$, in which $y_i^t \sim \Pr(Y_i|\mathbf{Y}_{-i} = \mathbf{y}_{-i}^{t-1}, \mathbf{X} = \mathbf{x}, \theta^t)$ specifies the state transition probability. In general, learning an MLE estimate of parameters in DBNs is expensive [4].

3.2 A greedy method

Let $Q^t(\mathbf{Y}|\mathbf{X})$ be the distribution over $\mathbf{Y}^t$ defined by the inhomogeneous process above. One approach would be learning $\theta^0, \theta^1, \ldots, \theta^K$ in a greedy fashion. Begin as before, by setting θ^0 to maximize data likelihood, using a model restricted to the "non-relational" features, as shown in Equation 2.

Then, for $k = 1, \ldots, K$, set θ^k to maximize likelihood of the data under the assumption that Q^{k-1} is fixed, and k is the *final* iteration to be performed: in other words, choose θ^k as

$$\theta^k = \text{argmax}_\theta \prod_{(\mathbf{x},\mathbf{y}) \in D} \prod_{i=1}^{n} \Pr(Y_i = y_i|\mathbf{Y}_{-i}, \mathbf{x}, \theta) \cdot Q^{k-1}(\mathbf{Y}_{-i}|\mathbf{x}) \tag{3}$$

Let θ_i^k be parameters of the model for $\Pr(Y_i)$ that is used at stage k. Naively, one might first learn each θ_i^0 (e.g., using logistic regression), and in each later step, learn θ_i^k using values of $\mathbf{Y}_{-i}$ that have been sampled from Q^{k-1}. One practical difficulty with this is that, while modern logistic regression learning methods produce reasonably well-calibrated probability estimates on unseen test data, their probability estimates on *training* data are biased. Thus, one cannot simply use the learned θ^{k-1} to estimate the necessary values of Q^{k-1} during training.

To avoid this problem, we used an idea suggested by a meta-learning scheme, *stacking* [13]. We let θ_i^k be the MAP estimate for $\Pr(Y_i = y_i|\text{MB}_i(\hat{\mathbf{y}}^{k-1}), \mathbf{x})$, where $\hat{y}_j^{k-1}$ is a prediction obtained in a cross-validation test of a logistic regression learner on the learning problem associated with θ_j^{k-1}.

Another issue is how to model and capture the dependencies. We use *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.

After we incorporate cross-validated predictions and the relational template into the learning algorithm, we end up with the inference and learning methods of Figure 1 for collective classification using short inhomogeneous Markov chains. The relational template can be extended to include aggregation functions based on $\hat{\mathbf{y}}$ and x_i. We will demonstrate the use of aggregations in Section 4.

If the data is produced by a homogeneous Markov chain that converges geometrically, i.e., if $|\Pr^n(\mathbf{Y}|\mathbf{X}) - \Pr^{\text{inf}}(\mathbf{Y}|\mathbf{X})| < (1-\epsilon)^n$ for all $\mathbf{X}$, $\mathbf{Y}$, n and initial distributions $\Pr^0$ (here $\Pr^k$ is the distribution after k iterations of the homogeneous Markov chain), the stacking method will converge at least as fast, i.e., $|Q^n(\mathbf{Y}|\mathbf{X}) - \Pr^{\text{inf}}(\mathbf{Y}|\mathbf{X})| < (1-\epsilon)^n$. This can be proven as follows:

- 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 $k = 0$:
    Return $f^0 = A(D^0)$. 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 $\mathbf{x} \in D$ as follows:
       (a) Split D into J equal-sized disjoint subsets $D_1...D_J$.
       (b) For $j = 1...J$, let $f_j^{k-1} = A(D^{k-1} - D_j^{k-1})$.
       (c) For $\mathbf{x} \in D_j, \hat{\mathbf{y}}^{k-1} = f_j^{k-1}(\mathbf{x}^{k-1})$.
    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. Return $f^k = A(D^k)$.
- 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

It is obvious for n=0 since the starting point is θ^0, same as the homogeneous Markov chain that converges. If it is true for Q^{n-1}, by the convergence of the target chain, there is some θ^n that reduces the error by a factor of $1 - \epsilon$ starting with Q^{n-1}, namely, θ^+, the homogeneous chain parameters. The stacking algorithm will take the θ^n that reduces error the most on Q^{n-1} on the data; in the limit, i.e., ignoring overfitting issues, this will reduce the error by at least $(1 - \epsilon)^n$.

3.3 An alternative approach

A further approximation is to set $\mathbf{y}^1$, $\mathbf{y}^2$, ..., $\mathbf{y}^K$ to all be equivalent, and all equal to $\mathbf{y}^0$. This approximation is conceptually useful, as it allows us to "unroll" the functions defined by θ^1, θ^2, ..., θ^K, and enumerate the features that contribute to each. Let MB_i^2 be the set of all variables Y_ℓ such that $Y_\ell \in \mathrm{MB}_j$ for some $Y_j \in \mathrm{MB}_i$—i.e., MB_i^2 is the Markov blanket of the Markov blanket of Y_i. Likewise, let MB_i^k be the "order k Markov blanket of i", or set of all variables Y_ℓ such that $Y_\ell \in \mathrm{MB}_j$ for some $Y_j \in \mathrm{MB}_i^{k-1}$. Intuitively, MB_i^K is a diameter-K subgraph centered around Y_i—a sort of generalized "sliding window".

Letting $\tilde{Q}^k$ be approximation to Q^k that is obtained in this way, it is easy to see that

$$\begin{array}{lll}
\tilde{Q}^1(Y_i = y_i|\mathbf{x}) & \text{depends only on the values of} & \mathbf{x}, y_i, \mathrm{MB}_i(\mathbf{y}^0) \\
\tilde{Q}^2(Y_i = y_i|\mathbf{x}) & \text{depends only on the values of} & \mathbf{x}, y_i, \mathrm{MB}_i^2(\mathbf{y}^0) \\
\cdots & \cdots & \cdots \\
\tilde{Q}^K(Y_i = y_i|\mathbf{x}) & \text{depends only on the values of} & \mathbf{x}, y_i, \mathrm{MB}_i^K(\mathbf{y}^0)
\end{array}$$

This suggests a new approximate learning method, in which we set θ^0 as in the greedy method above; construct, in some way, an expanded set of features that relate the value of y_i to the values of $\mathbf{x}$ and $\mathrm{MB}_i^K(\mathbf{y^0})$; and finally find parameters $\tilde{\theta}$ that are MAP estimates for the dataset D using these features. This approximation leads to an inhomogeneous chain that is only two steps long.

In prior work [2], this sort of approximation was found to be effective for certain sequential classification problems—a special case of the collective classification task considered here in which the Markov network is a linear chain. However, there are reasons to believe that this "window" approximation will be less appropriate for general graphs. Consider a simple case, where the expanded

feature set includes only a single edge between y_i and each $y_j \in \mathrm{MB}_i$, the Markov network has a maximum clique size of 2, and every $|\mathrm{MB}_i|$ is bounded by some small constant b. Let n be the number of parameters in the model θ^+. It is easy to see that the number of features used in this "unrolled" 2-step chain can grow rapidly with K: because $|\mathrm{MB}_i^K|$ can grow as b^K, the 2-step chain can have up to $n \cdot b^K$ parameters. This means that learning and evaluating the classifiers used in the second step of the chain will be expensive, and that the learner will be prone to overfit. While with linear chains, such as sequential classification problems, MB_i^K contains only $O(K)$ variables, which is probably why overfitting is less of a problem in [2].

Under the same assumptions, the method of Figure 1 will use only $n \cdot K$ parameters: even though $Q^K(Y_i = y_i|\mathbf{x})$ also depends on the values of $\mathbf{x}, y_i, \mathrm{MB}_i^K(\mathbf{y}^0)$, this dependency is "funnelled through" small Markov blankets involving variables $\mathbf{y}^1, \ldots, \mathbf{y}^{K-1}$. In the experiments, we will show that the method of Figure 1 is less liable to overfit than the two-step approximation.

4 Experimental Results

4.1 Datasets

We evaluated stacked graphical learning on two classification problems. The first problem we studied is the task of text region detection in the system called the Subcellular Location Image Finder (SLIF) [7, 11]. 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, upper, and lower. We also consider the dependency among candidate regions within the same panel, called *competitors*. Figure 2 is an example figure in SLIF which demonstrates candidate regions, neighbors and competitors.

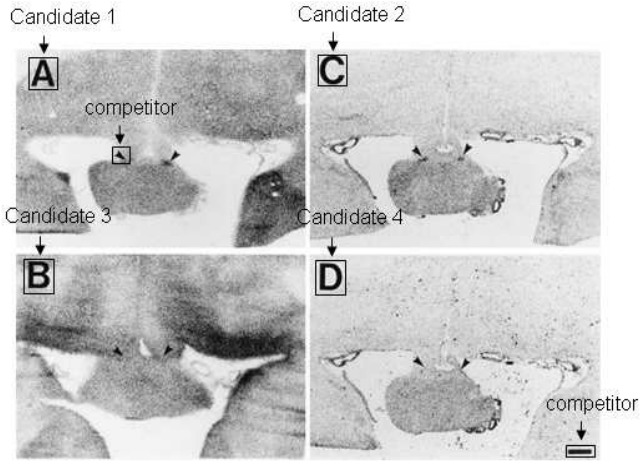

Figure 2: An example figure in SLIF

The relational template returns the predictions on one candidate region's neighbors and competitors. 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.

In the real-world SLIF dataset, there are usually a few (usually 2$\sim$10) panels in one figure. Thus the connected sub-graphs are relatively small. We generated a synthetic SLIF dataset with the same distributions over features and labels as the real-world data set, allowing more panels in one figure. We evaluated our approach on this synthetic dataset too. The synthetic dataset contains approximately 7500 candidate regions from 3200 panels in 200 figures.

The second dataset is the WebKB data[3] containing webpages from four computer science departments. The webpages were manually labelled with one of the six categories: course, faculty, student, staff, research projects, or other. The data contains aproximately 3800 webpages 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.

We use a maximum entropy learner as the base learner in stacked graphical learning. On each of the two real-world data sets, we have two feature sets. One is a base feature set and another one is a stronger feature set. The base feature set for SLIF data is constructed as follows: an Optical Character Recognition (OCR) software is applied to the candidate regions and the OCR output is converted to a binary feature, i.e., if a character is recognized from the candidate region, we assign 1, otherwise, 0. We join the binary features of a candidate region and all its neighbors, and the EXISTS aggregation result on its competitors to form a feature vector for the candidate region. Besides the base features, the stronger feature set for SLIF data includes more features obtained via image processing, such as the region size, the mean of the gray values, the standard deviation of gray values, whether the candidate region is considered clean or not. The base feature set for WebKB data includes features about the URL, school, and a set of bag-of-word feature. The stronger feature set for WebKB data includes more features about the number of incoming and outgoing links.

4.2 Performance of Stacked Graphical Learning

To evaluate the effectiveness of stacked graphical learning, we compare four models. The first model is a relational dependency network (RDN) model [6]. The RDN model uses the same features as the stacked model, but learning via a pseudo-likelihood method and inference with Gibbs sampling. The second model is a local model, i.e., the model trained with the base learner. The third model is the stacked graphical model. The fourth 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. Table 1 shows the accuracy for each of the four models on two real-world datasets and the synthetic dataset. For SLIF data, we used 5 fold cross validation, for WebKB, 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 three models. The null hypothesis is that there is no difference in the accuracy of the two models. The differences that are statistically significant at a $p < .05$ level are reported in the table with - or +.

On four of the tasks, the stacked graphical models improve the performance of the base learner significantly. Classification on WebKB data with the strong feature set is the only variant where stacking does not give a significant improvement. Yet in this case, there is no significant difference between the performance of the four models. On all the tasks, stacked graphical learning achieves statistically indistinguishable results to RDNs and on the WebKB tasks, stacked graphical learning

Table 1: Evaluation on four models

	SLIF data		Synthetic Data	WebKB	
	basic features	+additional features	basic features	basic features	+additional features
RDNs	86.7	88.9	91.3	74.2	75.7
Local model	77.2^{-}	85.2^{-}	85.3^{-}	58.3^{-}	75.3
Stacked model (k=1)	90.1	89.8	92.5	73.2	75.9
Stacked model (k=2)	90.1	90.3	92.8	72.1	75.9
Ceiling for stacked model	96.3^{+}	95.5^{+}	97.6^{+}	73.6	76.5

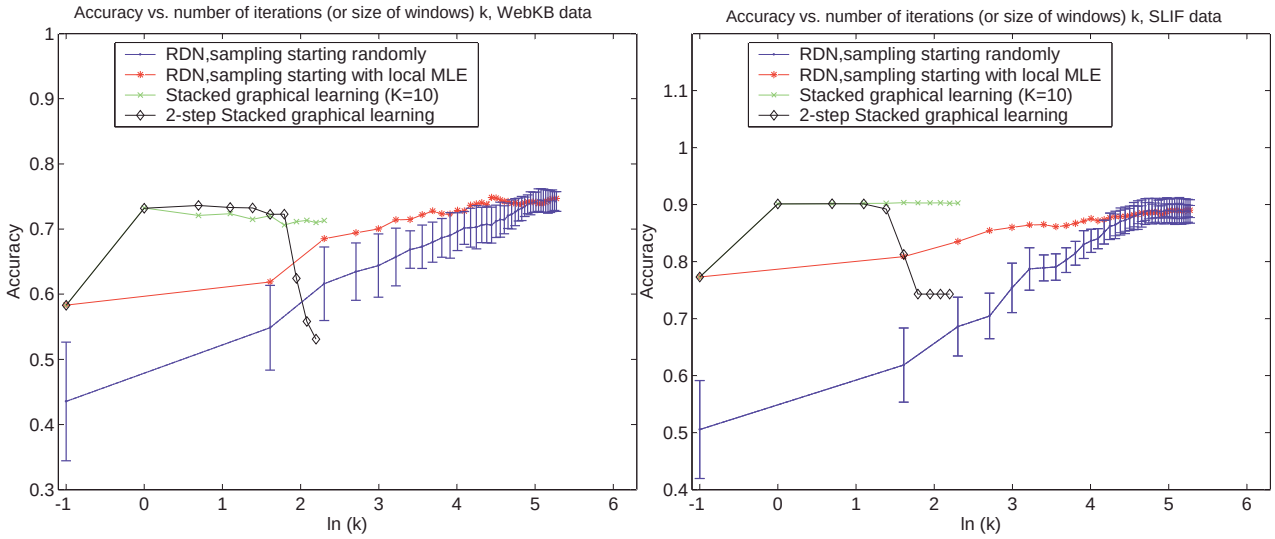

Figure 3: Convergence rate of stacking and Gibbs sampling

achieves comparable results to the ceiling models. With one more iteration of stacking, usually there is no significant improvement of accuracy. On the WebKB task with the base feature set, the RDN model obtains an average accuracy of 74.2% while the ceiling model obtains an average accuracy of 73.6%. However, the difference is not statistically significant.

We notice that on the two classification problems, the base learner obtains a higher accuracy with the additional features. However, the differences in the accuracies of RDNs, stacked graphical models, and ceiling models on different feature sets are not statistically significant.

4.3 Convergence of Stacked Graphical Learning and Gibbs Sampling

Figure 3 shows the convergence rate of stacking compared to Gibbs sampling on RDNs. The plots were generated using SLIF data and WebKB data with the basic feature set. We created the plots with a natural logarithm of k, where $ln(k) = -1$ corresponds to the initials where $k = 0$, and $ln(k) = 0$ corresponds to $k = 1$. We observe that stacked models (green curves) 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 (red curves). Stacked graphical model 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 keeping at the same level. We observe that Gibbs sampling converges to a same level after much more iterations and the convergence rate when k is small depends much 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 initials. The standard deviation is calculated in each fold, and averaged to get the error bar.

The further approximation described in Section 3.3 suggests that a multi-stage stacking is related to a single-stage stacking with features from an order-k "window". Figure 3 shows that this is true when k is small, while the two-step approximation tends to overfit when the window size k grows (black curves).

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. We showed that stacked graphical learning is a kind of short inhomogeneous Markov chains. 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 becomes extremely important. The results on two classification problems 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.

Cohen and Carvalho introduced stacked sequential learning in their paper [2]. In this paper, we extend the stacked sequential model to a more general case, where relational data is considered as the application, and demonstrate that stacked graphical models are short inhomogeneous Markov chains. McCallum and Sutton introduced parameter independence diagrams for introducing additional independence assumptions into parameter estimation for efficient training of undirected graphical models[9]. 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.

Future work will compare stacked models to more graphical models such as relational Markov networks, and explore more on the relational template design and base learner selection. For example, integrating an online learning algorithm will enable fast training of stacked graphical models. Also there are more ways to design the relational template and expand the features during stacking. For example, when there are many features in the original feature set, simply "adding" the predictions might not work well. We are also considering the application to inter-related classification problems in an information extraction system.

References

[1] R. Bunescu and R. J. Mooney. Relational markov networks for collective information extraction. In *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning (SRL-2004)*, Banff, Canada, July 2004.

[2] W. W. Cohen and V. R. Carvalho. Stacked sequential learning. In *Proceedings of Nineteenth International Joint Conferences on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[3] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, 1998.

[4] Z. Ghahramani. Learning dynamic Bayesian networks. *Lecture Notes in Computer Science*, 1387:168–197, 1998.

[5] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. M. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

[6] D. Jensen and J. Neville. Dependency networks for relational data. In *Proceedings of 4th IEEE International Conference on Data Mining (ICDM-04)*, Brighton, UK, 2004.

[7] Z. Kou, W. W. Cohen, and R. F. Murphy. Extracting information from text and images for location proteomics. In *Proceedings of the BIOKDD 2003*, Washington D.C., 2003.

[8] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML-2001)*, Williams, MA, 2001.

[9] A. McCallum and C. Sutton. Piecewise training with parameter independence diagrams: Comparing globally- and locally-trained linear-chain crfs. Technical Report IR-383, Center for Intelligent Information Retrieval, University of Massachusetts, 2004. Available from http://www.cs.umass.edu/ mccallum/papers/lcrf-nips2004.pdf.

[10] K. P. Murphy. Bayes net toolbox for matlab. *Computing Science and Statistics*, 33, 2001.

[11] 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. In *Proceedings of the IASTED International Conference on Knowledge Sharing and Collaborative Engineering*, St. Thomas, US Virgin Islands, 2004.

[12] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)*, Edmonton, Canada, 2002.

[13] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.