Learning in the Deep-Structured Conditional Random Fields

Dong Yu, Li Deng
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{*dongyu, deng*}*@microsoft.com*

Shizhen Wang[1]
University of California
405 Hilgard Avenue
Los Angeles, CA 90095
szwang@ee.ucla.edu

Abstract

We have proposed the deep-structured conditional random fields (CRFs) for sequential labeling and classification recently. The core of this model is its deep structure and its discriminative nature. This paper outlines the learning strategies and algorithms we have developed for the deep-structured CRFs, with a focus on the new strategy that combines the layer-wise unsupervised pre-training using entropy-based multi-objective optimization and the conditional likelihood-based back-propagation fine tuning, as inspired by the recent development in learning deep belief networks.

1 Introduction

Conditional random fields (CRFs) are *discriminative* models that directly estimate the probabilities of the state sequence conditioned on the whole observation sequence. This is in contrast to the *generative* models such as the hidden Markov models (HMMs) that describe the joint probability of the observation and the states. Given their discriminative nature and their high flexibility in choosing features, CRFs have been widely and successfully used to solve sequential labeling problems, notably those in natural language processing [1] [2] and speech processing [3].

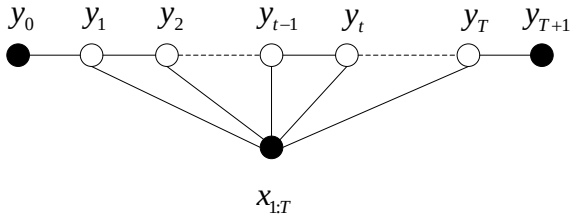

Figure 1. The graphical representation of the linear-chain CRF

The linear-chain CRF depicted in Figure 1 is the most popular CRF due to its simplicity and efficiency. Given a T-frame observation sequence $\boldsymbol{x} = (x_1, x_2, \cdots, x_T)$, the conditional probability of the state sequence $\boldsymbol{y} = (y_1, y_2, \cdots, y_T)$ (which may be augmented with a special start (y_0) and end (y_{T+1}) state) is formulated as

[1] Shizhen Wang contributed to this work while he was an intern at Microsoft Research.

$$p(\boldsymbol{y}|\boldsymbol{x};\Lambda) = \frac{exp\left(\sum_{t,i}\lambda_i f_i(y_{t-1}, y_t, \boldsymbol{x}, t)\right)}{Z(\boldsymbol{x};\Lambda)} \tag{1}$$

where we have used $f_i(y_{t-1}, y_t, \boldsymbol{x}, t)$ to represent both the observation features $f_i(y_t, \boldsymbol{x}, t)$ and the state transition features $f_i(y_t, y_{t-1}, t)$. The partition function

$$Z(\boldsymbol{x};\Lambda) = \sum_{\boldsymbol{y}} exp\left(\sum_{t,i}\lambda_i f_i(y_{t-1}, y_t, \boldsymbol{x}, t)\right) \tag{2}$$

is used to normalize the exponential form so that it becomes a valid probability measure. The model parameters $\Lambda = \{\lambda_i\}$ are typically optimized to maximize the L_2 regularized conditional state sequence log-likelihood

$$J_1(\Lambda, X) = \sum_k \log p\left(\boldsymbol{y}^{(\boldsymbol{k})}|\boldsymbol{x}^{(\boldsymbol{k})};\Lambda\right) - \frac{\|\Lambda\|^2}{2\sigma^2} \tag{3}$$

where σ^2 is a parameter that balances the log-likelihood and the regularization term and is typically tuned using a development set. The derivatives of $J_1(\Lambda, X)$ over the model parameters λ_i are given by

$$\begin{aligned}\frac{\partial J_1(\Lambda, X)}{\partial \lambda_i} &= \tilde{E}[f_i(\boldsymbol{y}, \boldsymbol{x})] - E[f_i(\boldsymbol{y}, \boldsymbol{x})] - \frac{\lambda_i}{\sigma^2} \\ &= \sum_k f_i\left(\boldsymbol{y}^{(\boldsymbol{k})}, \boldsymbol{x}^{(\boldsymbol{k})}\right) - \frac{\lambda_i}{\sigma^2} - \sum_k \sum_{\boldsymbol{y}} p\left(\boldsymbol{y}|\boldsymbol{x}^{(\boldsymbol{k})};\Lambda\right) f_i\left(\boldsymbol{y}, \boldsymbol{x}^{(\boldsymbol{k})}\right),\end{aligned} \tag{4}$$

which can be efficiently estimated using the forward-backward (sum-product) algorithm [1] [4]. The model parameters in the CRFs can thus be optimized using algorithms such as generalized iterative scaling (GIS) [5], gradient and conjugate gradient (e.g. L-BFGS) ascent [6], and RPROP [7].

Although great performance has been observed using the single-layer CRFs, limitations associated with their shallow structure are also noticeable. For example, the single-layer CRFs typically require manual construction of many different features to achieve good performance and require a large amount of training data to obtain the generalization ability. They lack the ability to automatically generate robust discriminative internal features from the raw features. As an example, we have shown [8][9] that when continuous features are used, better performance can be achieved by imposing constraints on the distribution of the features, which is equivalent to expanding each continuous feature $f_i(y_{t-1}, y_t, \boldsymbol{x}, t)$ into L features

$$f_{il}(y_{t-1}, y_t, \boldsymbol{x}, t) = a_l\left(f_i(y_{t-1}, y_t, \boldsymbol{x}, t)\right) f_i(y_{t-1}, y_t, \boldsymbol{x}, t), \tag{5}$$

where $a_l(.)$ is a weight function whose definition can be found in [8][10][11]. However, the single-layer CRFs cannot learn these expanded features automatically.

Motivated by the recent advances in deep learning developed by the neural network community [12][13][14][15][16], we have recently proposed the deep-structured CRFs for sequential labeling and classification and observed promising results on the text labeling [2] and language identification tasks [9]. In the deep-structured CRFs, multiple layers of simple CRFs are stacked together to achieve a much more powerful modeling and discrimination ability.

Using multiple layers of CRFs to improve the modeling power is not new. Several flavors of hierarchical CRFs have been proposed in the literature [3][17][18]. Those models typically aim at tackling the granularity problem at different representation layers and use the lower layer CRFs as the building blocks for the higher layer CRFs. The deep-structured CRF discussed in this paper distinguishes itself from the conventional hierarchical models in that it aims at learning discriminative intermediate representations from the raw features and at combining all sources of information to obtain a superior classification ability.

The purpose of this paper is to summarize the learning strategies and algorithms we have

developed for the deep-structured CRFs, with a focus on the new strategy that combines the entropy-based layer-wise unsupervised pre-training and the conditional likelihood-based back-propagation fine tuning. We first describe the architecture of the deep-structured CRF in Section 2. We then illustrate the layer-wise supervised learning strategy, and the strategy that combines the layer-wise unsupervised pre-training and the likelihood back-propagation fine tuning in Sections 3 and 4, respectively. We provide some experimental results in Section 5 and summarize the paper in Section 6.

2 Architecture of Deep-Structured CRF

The architecture of the deep-structured CRF discussed in this paper is depicted in Figure 2, where the final layer is a linear-chain CRF and the lower layers are zero-th-order CRFs that do not use state transition features. Using zero-th-order instead of linear-chain CRFs in the lower layers can significantly reduce the computational cost while only slightly degrades the classification performance. In the deep-structured CRF, the observation sequence at layer j consists of two parts: the previous layer's observation sequence $\boldsymbol{x^{j-1}}$ and the frame-level marginal posterior probabilities $p\left(y_t^{j-1}\middle|\boldsymbol{x^{j-1}}\right)$ from the preceding layer $j-1$. This is inspired by the tandem structure used in some automatic speech recognition systems [19]. Note that the features constructed on the observations may use only part of the input information though.

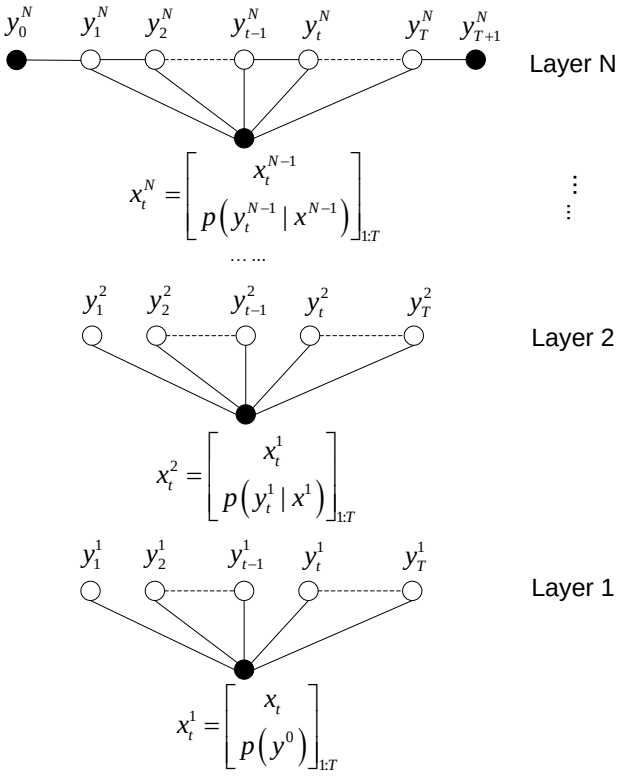

Figure 2. The graphical representation of the deep-structured CRF.

In the deep-structured CRF, the state sequence inference is carried out layer-by-layer in a bottom-up manner so that the computational complexity is limited to at most linear to the number of layers used. The model parameter estimation is more complicated. At the final layer the number of states can be directly determined by the problem to be solved and the parameters can be learned in the supervised way. However, parameter learning can be tricky for the intermediate layers, which serve as abstract internal representations of the original observation and may have completely different number of states than the final layer.

In the following sections, we will describe two learning strategies for the deep-structured CRFs. In the layer-wise supervised learning, we restrict the number of states at intermediate layers to be the same as that in the final layer. so that the same label used to train the final layer

can be used to train all the intermediate layers. In the second strategy of entropy-based layer-wise unsupervised pre-training followed by conditional likelihood-based back propagation learning, we allow for an arbitrary number of states in the intermediate layers. This learning scheme first learns each intermediate layer separately in an unsupervised manner, and then fine-tunes all the parameters jointly.

3 Layer-wise Supervised Learning

If we restrict the number of states at intermediate layers to be the same as that in the final layer and treat each state at intermediate layers the same as that in the final layer, we can train the intermediate layers layer-by-layer using the same label used to train the final layer. Note that the output of the deep-structured CRF model is a state sequence; so the parameters in the final layer are optimized by maximizing the regularized conditional log-likelihood (3) at the state-sequence level. In contrast to the highest layer, all remaining layers are trained by maximizing the frame-level marginal log-likelihood of

$$J_2(\Lambda, X) = \sum_{k,t} \log p\left(y_t^{(k)} | \boldsymbol{x}^{(k)}; \Lambda\right) - \frac{\|\Lambda\|^2}{2\sigma^2} \tag{6}$$

since this marginal probability is the only additional information passed into the higher layers. This criterion, however, is equivalent to the state-sequence level criterion $J_1(\Lambda, X)$ when the zero-th-order CRF is used in the intermediate layers since

$$\begin{aligned} J_1(\Lambda, X) &= \sum_{k} \log p\left(\boldsymbol{y}^{(\boldsymbol{k})} | \boldsymbol{x}^{(\boldsymbol{k})}; \Lambda\right) - \frac{\|\Lambda\|^2}{2\sigma^2} \\ &= \sum_{k} \log \frac{exp\left(\sum_{t,i} \lambda_i f_i\left(y_t^{(\boldsymbol{k})}, y_{t-1}^{(\boldsymbol{k})}, \boldsymbol{x}^{(\boldsymbol{k})}, t\right)\right)}{Z(\boldsymbol{x}^{(\boldsymbol{k})}; \Lambda)} - \frac{\|\Lambda\|^2}{2\sigma^2} \\ &= \sum_{k} \sum_{t,i} \lambda_i f_i\left(y_t^{(\boldsymbol{k})}, \boldsymbol{x}^{(\boldsymbol{k})}, t\right) - \log Z\left(\boldsymbol{x}^{(\boldsymbol{k})}; \Lambda\right) - \frac{\|\Lambda\|^2}{2\sigma^2} \\ &= \sum_{k,t} \sum_{i} \lambda_i f_i\left(y_t^{(\boldsymbol{k})}, \boldsymbol{x}^{(\boldsymbol{k})}, t\right) - \log Z\left(\boldsymbol{x}^{(\boldsymbol{k})}; \Lambda\right) - \frac{\|\Lambda\|^2}{2\sigma^2} \\ &= J_2(\Lambda, X). \end{aligned} \tag{7}$$

$J_2(\Lambda, X)$ can be optimized in a complexity of $O(TY)$, where T is the number of frames and Y is the number of states. Since the output of each frame in the zero-th-order CRF is independent of each other, the process can be further speeded up using parallel computing techniques.

Note that the observation features at each layer can be constructed differently, and possibly across different frames than the previous layer also. This allows for the great flexibility of the higher layers to incorporate longer-span features from lower-layer decoding results. Allowing for long-span features can be helpful for speech recognition [20] [21][22][23] tasks.

We now describe some desirable theoretical properties of this training strategy.

Theorem 1: The objective function $J_1(\Lambda, X)$ on the training set will not decrease as more layers are added in the deep-structure CRF.

Proof: Let's consider the extension from an N-layer deep-structured CRF to an N+1 layer deep-structured CRF. The parameters for the first N-1 layers are the same for both systems. For the N-layer system, the observation features at the final layer are constructed on $\boldsymbol{x}^{\boldsymbol{N}}$ and the corresponding parameter set is Λ^N. For the N+1-layer system, the observation features are constructed on the observations that are augmented by $p'(y_t^N | \boldsymbol{x}^{\boldsymbol{N}})$ at each frame, where we use p' to indicate that the probability is estimated using the N-th layer in the N+1-layer system. The corresponding parameter set at the final layer in the N+1-layer system is $\Lambda^{N+1} \supseteq \Lambda^N$. Since

$$\max_{\Lambda^N} J_1(\Lambda^N, X) \le \max_{\Lambda^{N+1}} J_1(\Lambda^{N+1}, X) \tag{8}$$

and the optimization problem is convex at each layer, the learning algorithm which achieves the global optimum enabled by the convexity can always find a parameter set in the N+1-layer system that gives a higher value of $J_1(\Lambda, X)$.■

It directly follows that

Corollary 1: The deep-structured CRF performs no worse than the single-layer CRF on the training set.

Note that the conditional log-likelihood increase in the training set can be carried over to the test set with a properly chosen regularization term. However, as the number of intermediate layers continues to grow, the gain will eventually saturate.

4 Layer-wise Unsupervised Learning with Fine Tuning

The layer-wise supervised training paradigm described in Section 3 works only when the number of states in the intermediate layers is the same as that in the final layer so that the same supervision can be used to train each layer. This requirement, however, significantly restricts the potential of the deep-structured CRF for extracting powerful, optimization-driven internal representations automatically from the original data sequence. In this section, we relax this constraint and allow for completely different internal representations with vastly different number of states in the intermediate layers. This relaxation requires a different training algorithm with different objective function(s) as an intermediate step.

A conceptually simple approach to train the deep-structured CRF with arbitrarily configured intermediate layers is to train all the model parameters jointly. However, it has been shown [13][14][15][16] that when the number of layers increases, joint training can be very inefficient and leads to poor local optimum. Alternatively, one can train the intermediate layers one by one in an unsupervised manner, for example, in a generative way by optimizing the association between the input and the output for each intermediate layer.

In this paper, we propose a layer-wise unsupervised learning strategy with a discriminative flavor where we cast the intermediate layer learning problem into a multi-objective programming (MOP) one. More specifically, we minimize the average frame-level conditional entropy and maximize the state occupation entropy at the same time. Minimizing the average frame-level conditional entropy forces the intermediate layers to be sharp indicators of subclasses (or clusters) for each input vector, while maximizing the occupation entropy guarantees that the input vectors be represented distinctly by different intermediate states. The training of this MOP problem is carried out in a similar way to that described in [19]. Specifically, we start from maximizing the state occupation entropy. We then update the parameters by alternating between minimizing the frame-level conditional entropy and maximizing the average state occupation entropy. At each epoch, we optimize one objective by allowing the other one to become slightly worse within a limited range. This range is gradually tightened epoch by epoch. The model parameters are then fine tuned using the conditional likelihood-based back propagation we will describe shortly.

4.1 Maximize the state occupation entropy

For simplicity, let us denote by $\boldsymbol{x}$, $\boldsymbol{h}$, and $\Lambda^{\mathrm{h}} = \{\lambda_i^h\}$ the input, output, and parameters of an intermediate layer, respectively. The intermediate layer state occupation entropy is defined as

$$H(h) = -\sum_h p(h)\, log p(h) \tag{9}$$

where

$$p(h) = \frac{1}{K}\sum_{k}\sum_{t} p\left(h_t = h\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right). \tag{10}$$

The derivative of $H(h)$ with respect to λ_i^h can be calculated as

$$\begin{aligned}\frac{\partial H(h)}{\partial \lambda_i^h} &= -\frac{\partial p(h)}{\partial \lambda_i^h} logp(h) - \frac{\partial logp(h)}{\partial \lambda_i^h} p(h) \\ &= -[logp(h) + 1]\frac{\partial p(h)}{\partial \lambda_i^h} \\ &= -\frac{1}{K}[logp(h) + 1]\sum_{k}\sum_{t}\frac{\partial p\left(h_t = h\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right)}{\partial \lambda_i^h}.\end{aligned} \tag{11}$$

Since

$$\frac{\partial p\left(h_t = h\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right)}{\partial \lambda_i^h} = \left[p\left(h_t\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right) - p^2\left(h_t\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right)\right] f_i\left(h_t, \boldsymbol{x}^{(\boldsymbol{k})}, t\right) \tag{12}$$

we obtain the final gradient

$$\frac{\partial H(h)}{\partial \lambda_i^h} = -\frac{1}{K}[logp(h) + 1]\sum_{k}\sum_{t}\left[p\left(h_t\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right) - p^2\left(h_t\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right)\right] f_i\left(h_t, \boldsymbol{x}^{(\boldsymbol{k})}, t\right). \tag{13}$$

4.2 Minimize the frame-level conditional entropy

The frame-level conditional entropy at the intermediate layer can be written as

$$H\left(h\middle|\boldsymbol{x}, \Lambda^{\mathrm{h}}\right) = -\sum_{k}\sum_{h} p\left(h\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right) logp\left(h\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right). \tag{14}$$

Following the similar procedure we compute the derivative of $H\left(h\middle|\boldsymbol{x}, \Lambda^{\mathrm{h}}\right)$ with respect to λ_i^h as

$$\begin{aligned}\frac{\partial H\left(h\middle|\boldsymbol{x}, \Lambda^{\mathrm{h}}\right)}{\partial \lambda_i^h} &= -\sum_{k}\sum_{t}\left[logp\left(h_t\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right) + 1\right]\frac{\partial p\left(h\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right)}{\partial \lambda_i^h} \\ &= -\sum_{k}\sum_{t}\left[logp\left(h_t\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right) + 1\right]\left[p\left(h\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right) - p^2\left(h\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{h}}\right)\right] f_i\left(h_t, \boldsymbol{x}^{(\boldsymbol{k})}, t\right)\end{aligned} \tag{15}$$

4.3 Fine tuning with conditional likelihood-based back propagation

In the fine tuning step, we aim to optimize the state sequence log-likelihood

$$\begin{aligned}L\left(\Lambda^{\mathrm{N}}, \Lambda^{\mathrm{h_{N-1}}}, \cdots, \Lambda^{\mathrm{h_1}}\right) &= \sum_{k} logp\left(\boldsymbol{y}^{(\boldsymbol{k})}\middle|\boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{N}}, \Lambda^{\mathrm{h_{N-1}}}, \cdots, \Lambda^{\mathrm{h_1}}\right) \\ &= \sum_{k} L^{(\boldsymbol{k})}\left(\Lambda^{\mathrm{N}}, \Lambda^{\mathrm{h_{N-1}}}, \cdots, \Lambda^{\mathrm{h_1}}\right).\end{aligned} \tag{16}$$

jointly for all parameters conditioned on all the layers, where Λ^{N} is the parameter set for the final layer, and $\Lambda^{\mathrm{h_{N-1}}}, \cdots, \Lambda^{\mathrm{h_1}}$ are parameters for the $N-1$ hidden layers. The observation as the input to the final layer is

$$\left[x_t \quad f_t^{\mathrm{h_1}} \quad f_t^{\mathrm{h_2}} \quad \cdots \quad f_t^{\mathrm{h_{N-1}}}\right], t = 1, \cdots, T \tag{17}$$

where the hidden layer's frame-level log-likelihood is

$$f_t^{\mathrm{h_n}} = \log p\left(h_t^{\mathrm{n}}\middle|\boldsymbol{x}, \boldsymbol{f}^{\mathrm{h_1}}, \cdots, \boldsymbol{f}^{\mathrm{h_{n-1}}}, \Lambda^{\mathrm{h_n}}\right) \quad \text{if n} > 1 \tag{18}$$

and

$$f_t^{\mathrm{h_n}} = \log p\left(h_t^{\mathrm{n}} \middle| \boldsymbol{x}, \Lambda^{\mathrm{h_n}}\right) \quad \text{if n} = 1. \tag{19}$$

The derivative of the objective function over $\lambda_i^{h_n}$ is

$$\begin{aligned} \frac{\partial L\left(\Lambda^{\mathrm{N}}, \Lambda^{\mathrm{h_{N-1}}}, \cdots, \Lambda^{\mathrm{h_1}}\right)}{\partial \lambda_i^{h_n}} &= \sum_k \sum_{j=n}^{N-1} \frac{\partial L^{(\boldsymbol{k})}\left(\Lambda^{\mathrm{N}}, \Lambda^{\mathrm{h_{N-1}}}, \cdots, \Lambda^{\mathrm{h_1}}\right)}{\partial \boldsymbol{f}^{\mathrm{h_j}}} \frac{\partial \boldsymbol{f}^{\mathrm{h_j}}}{\partial \lambda_i^{h_n}} \\ &= \sum_k \left[1 - p\left(y_t | \boldsymbol{x}^{(\boldsymbol{k})}, \Lambda^{\mathrm{N}}, \Lambda^{\mathrm{h_{N-1}}}, \cdots, \Lambda^{\mathrm{h_1}}\right)\right] \sum_{j=n}^{N-1} \boldsymbol{\lambda}^{\mathrm{h_j}} \frac{\partial \boldsymbol{f}^{\mathrm{h_j}}}{\partial \lambda_i^{h_n}} \end{aligned} \tag{20}$$

where $\partial \boldsymbol{f}^{\mathrm{h_j}} / \partial \lambda_i^{h_n}$ can be recursively calculated as in (20) by noticing that $f_t^{\mathrm{h_j}}$ has the same form as the $L\left(\Lambda^{\mathrm{N}}, \Lambda^{\mathrm{h_{N-1}}}, \cdots, \Lambda^{\mathrm{h_1}}\right)$ except with fewer layers.

5 Experimental Results

Table 1 summarizes the recognition accuracy (RA) on a seven language/dialect recognition task using the layer-wise unsupervised learning with fine tuning approach, where the distribution constraint refers to the feature expansion approach described in [8], CRF refers to the single-layer linear-chain CRF, and DSCRF refers to the deep-structured CRF described in this paper. Note that the DSCRF with 128 hidden states and four-knot distribution constraint has the same number of parameters as the Gaussian mixture model (GMM) with 256 mixtures. Due to the page limit, readers are referred to [9] for the detailed experimental setup. As a comparison, the best configuration of GMM using the maximum mutual information (MMI) training contains 256 Gaussian mixtures and achieved 82.5% recognition accuracy on this task. It is clear from Table 1 that the deep-structured CRF significantly outperforms the single-layer CRF with recognition accuracies of 83.6% vs. 44.6% and 79.5% vs. 34.3% with and without the distribution constraint respectively before using the tandem features and the fine tuning. When the tandem feature is applied, the recognition accuracy can be improved to 85.1% which was further improved to 86.4% when the fine tuning is applied.

Additional results on the layer-wise supervised training and on other tasks such as natural language processing can be found in [2].

Table 1: Summary of the recognition accuracy (RA) on the seven language/dialect recognition task

Model	# States /Mixtures	Distribution Constraint	Tandem	RA(%)
CRF	-	no	-	34.3
CRF	-	yes	-	44.6
DSCRF+pretrain	128	no	no	79.5
	128	yes	no	83.6
	128	yes	yes	85.1
DSCRF+finetune	128	yes	yes	86.4

6 Summary

In this paper, we have described a deep-structured CRF model, in which multiple layers of CRFs are stacked together to achieve higher classification accuracy. We illustrated two approaches to learning the model parameters in the deep-structured CRF.

In its nut-shell, the deep-structured CRF shares many ideas from the deep belief network (DBN) [12][13]. However, it differentiates itself from the DBN in that the layer-wise pre-training is carried out in a discriminative flavor and that the sequential information is

integrated in the same way as used in the conventional CRF. The latter contrasts the DBN, which requires additional temporal processing mechanisms to model the sequential input data.

Acknowledgments

We wish to thank Dr. Chin-Hui Lee at Georgia Institute of Technology and Dr. Xiao Li at Microsoft Research for helpful discussions.

References

[1] 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, pp. 282–289, 2001.

[2] D. Yu, S. Wang, and L. Deng, "Sequential labeling using deep-structured conditional random fields," submitted to IEEE Journal of Selected Topics in Signal Processing, 2009

[3] T. T. Truyen, "On conditional random fields: Applications, feature selection, parameter estimation and hierarchical modelling", Ph.D. dissertation, Curtin University of Technology, 2008.

[4] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, ISBN 0-387-31073-8, 2006.

[5] J. Darroch, and D. Ratcliff, "Generalized iterative scaling for log-linear models", Ann. Math. Statistics, 43:1470–1480, 1972.

[6] J. Nocedal, "Updating quasi-Newton matrices with limited storage", Mathematics of Computation, vol. 35, pp. 773-782, 1980.

[7] M. Riedmiller, and H. Braun, "A direct adaptive method for faster back-propagation learning: The RPROP algorithm", in proc. of IEEE ICNN, vol. 1, pp. 586-591. 1993.

[8] D. Yu, L. Deng, and A. Acero, "Using continuous features in the maximum entropy model", Pattern Recognition Letters. Vol. 30, Issue 8, June, 2009. doi:10.1016/j.patrec.2009.06.005.

[9] D. Yu, S. Wang, Z. Karam, L. Deng, "Language recognition using deep-structured conditional random fields", submitted to ICASSP 2010.

[10] D. Yu, L. Deng, Y. Gong, and A. Acero, "A novel framework and training algorithm for variable-parameter hidden Markov models", IEEE trans. on Audio, Speech, and Language Processing, vol. 17, no. 7, pp. 1348-1360, IEEE, September 2009.

[11] D. Yu, and L. Deng, "Solving nonlinear estimation problems using splines", IEEE Signal Processing Magazine, vol. 26, no. 4pp.86-90, July, 2009.

[12] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", Science, 313(5786), pp. 504–507, 2006.

[13] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets", Neural Computation, 2006, 18 (7). pp. 1527-1554.

[14] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks", NIPS'2006.

[15] M.'A. Ranzato, C. Poultney, S. Chopra, and Y. LeCun "Efficient learning of sparse representations with an energy-based model ", NIPS'2006.

[16] Y. Bengio, "Learning deep architectures for AI", Technical Report 1312, university of Montreal.

[17] L. Ladicky, C. Russell, P. Kohli, P. H. S. Torr, "Associative hierarchical CRFs for object class image segmentation", in Proc. ICCV 2009.

[18] L. Liao, D. Fox, and H. Kautz, "Hierarchical conditional random fields for GPS-based activity recognition", in Proc. of International Symposium of Robotis Research (ISRR), 2007.

[19] H. Hermansky, D. P. W. Ellis, S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems", in Proc. ICASSP, vol.3, pp. 1635-1638, 2000.

[20] L. Deng, D. Yu, and A. Acero, "A Bidirectional Target-Filtering Model of Speech Coarticulation and Reduction: Two-Stage Implementation for Phonetic Recognition", IEEE Trans. Audio, Speech & Language Proc, vol. 14, No. 1, pp 256-265, Jan 2006. doi: 10.1109/TSA.2005.854107.

[21] L. Deng, D. Yu, and A. Acero, "Structured speech modeling", IEEE Trans. on Audio, Speech and Language Processing. Vol. 14 No. 5, Sep 2006. pp. 1492- 1504.

[22] D. Yu, L. Deng, and A. Acero, "Evaluation of a Long-contextual-span Hidden Trajectory Model and Phonetic Recognizer Using A* Lattice Search", in Proc. of Interspeech, 2005, pp. 553-556.

[23] D. Yu, L. Deng, A. Acero, "A Lattice Search Technique for a Long-Contextual-Span Hidden Trajectory Model of Speech", Speech Communication, Elsevier. Volume: 48 Issue: 9, Sep 2006. pp. 1214-1226. doi:10.1016/j.specom.2006.05.002.

[24] S. Yaman and C.-H. Lee, "A flexible classifier design framework based on multi-objective programming," IEEE Trans. on Audio, Speech, and Language Processing, vol. 16, no. 4, pp. 779-789, 2008.