Investigation of Full-Sequence Training of Deep Belief Networks for Speech Recognition

Abdel-rahman Mohamed [1*], *Dong Yu*[2], *Li Deng*[2]

[1] Department of Computer Science, University of Toronto, Toronto, ON Canada
[2] Microsoft Research, Redmond, WA USA

asamir@cs.toronto.edu, {dongyu,deng}@microsoft.com

Abstract

Recently, Deep Belief Networks (DBNs) have been proposed for phone recognition and were found to achieve highly competitive performance. In the original DBNs, only frame-level information was used for training DBN weights while it has been known for long that sequential or full-sequence information can be helpful in improving speech recognition accuracy. In this paper we investigate approaches to optimizing the DBN weights, state-to-state transition parameters, and language model scores using the sequential discriminative training criterion. We describe and analyze the proposed training algorithm and strategy, and discuss practical issues and how they affect the final results. We show that the DBNs learned using the sequence-based training criterion outperform those with frame-based criterion using both three-layer and six-layer models, but the optimization procedure for the deeper DBN is more difficult for the former criterion.

Index Terms: Deep Belief Networks, phone recognition, discriminative training, full-sequence optimization

1. Introduction

Research in speech recognition has explored layered architectures in the recognizer design for quite some time (e.g., [1][2][3][5][15][17][18][19][20]), motivated partly by the desire to capitalize on some analogous properties in the human speech generation and perception systems. In these studies, learning of model parameters has been one of the most prominent and difficult problems and has thus received much attention. In parallel with the development in speech recognition research, a recent progress in learning methods from neural network research has renewed the interest in further exploring deep structured models. One major advance is the development of highly effective learning techniques for the deep belief networks (DBNs), which are densely connected, directed belief nets with many hidden layers [8][9].

Although deep networks typically have higher modeling power than their shallow counterparts, learning in the deep networks is significantly harder because it is difficult to infer the posterior distribution over the hidden variables when given a data vector and because the simple back-propagation algorithm does not perform effectively due to the significantly increased chance of trapping into a local optimum. The recently proposed learning strategy -- generative pre-training using restricted Boltzmann machines (RBMs) and contrastive divergence followed by discriminative fine-tuning – is a significant advance in learning DBNs. This strategy has proved to be effective in a number of applications including coding and classification for speech, audio, and image [6][8][9][12][13][14].

In a nut-shell, DBNs can be considered as a highly complex nonlinear feature extractor where each layer of the hidden units learns to represent features that capture higher order correlations in the original input data. DBNs have been recently proposed for phone recognition and were found to achieve very competitive performance [13]. In these DBNs, only frame-level information was used for training DBN weights; the transition parameters and language model (LM) scores were obtained from an HMM-like approach and were trained independently of the DBNs. However, speech recognition is a sequential or full-sequence learning problem and it has been well known that discriminative information at the sequence level contributes to improving recognition accuracy (e.g., [7]). Using neural networks to discriminate between sequences has indeed been proposed in the past (e.g., [4][10][11][16], but it was done in shallow architectures and not in an integrative manner. In this paper, we investigate approaches to optimizing the DBN weights, transition parameters, and language model (LM) scores jointly using a discriminative training criterion at the sequence level designed specifically for DBNs.

The rest of the paper is organized as follows. In Section 2 we introduce the DBN, with the RBMs as its constituents, and the general learning strategy for the DBN. In Section 3 we describe and analyze the sequential learning algorithm we developed for the DBN and discuss the practical issues and training procedures. We present the experimental results on the TIMIT phonetic recognition task as our initial exploration in Section 4, and conclude the paper in Section 5.

2. Deep Belief Network Basics

DBNs are densely connected, directed belief nets with many hidden layers for which learning is a difficult problem. In this section we introduce an effective training algorithm that learns each layer greedily by treating each pair of layers as an RBM before doing a joint optimization of all the layers.

2.1. Restricted Boltzmann Machines

An RBM is a particular type of Markov random field (MRF) that has one layer of (typically Bernoulli) stochastic hidden units and one layer of (typically Bernoulli or Gaussian) stochastic visible units. RBMs can be represented as bipartite graphs since all visible units are connected to all hidden units, but there are no visible-visible or hidden-hidden connections.

In the RBMs, the joint distribution $\mathrm{p}(\mathbf{v}, \mathbf{h}; \theta)$ over the visible units $\mathbf{v}$ and hidden units $\mathbf{h}$, given the model parameters θ, is defined in terms of an energy function $\mathrm{E}(\mathbf{v}, \mathbf{h}; \theta)$ of

$$\mathrm{p}(\mathbf{v}, \mathbf{h}; \theta) = \frac{exp\big(-\mathrm{E}(\mathbf{v}, \mathbf{h}; \theta)\big)}{Z}, \quad (1)$$

* This investigation was carried out during his internship at Microsoft Research, Redmond.

where $Z = \sum_u \sum_h exp\big(-\mathrm{E}(\mathbf{v}, \mathbf{h}; \theta)\big)$ is a normalization factor or partition function, and the marginal probability that the model assigns to a visible vector $\mathbf{v}$ is

$$\mathrm{p}(\mathbf{v}; \theta) = \frac{\sum_{\boldsymbol{h}} exp\big(-\mathrm{E}(\mathbf{v}, \mathbf{h}; \theta)\big)}{Z}. \quad (2)$$

For a Bernoulli (visible)-Bernoulli (hidden) RBM, the energy is

$$\mathrm{E}(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij}\, v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j, \quad (3)$$

where w_{ij} represents the symmetric interaction term between visible unit v_i and hidden unit h_j, b_i and a_j the bias terms, and V and H are the numbers of visible and hidden units. The conditional probabilities can be efficiently calculated as

$$p\big(h_j = 1|\mathbf{v}; \theta\big) = \sigma\left(\sum_{i=1}^{V} w_{ij}\, v_i + a_j\right), \quad (4)$$

$$p(v_i = 1|\mathbf{h}; \theta) = \sigma\left(\sum_{j=1}^{H} w_{ij}\, h_j + b_i\right). \quad (5)$$

where $\sigma(x) = 1/\big(1 + exp(x)\big)$.

Similarly, for a Gaussian-Bernoulli RBM, the energy is

$$\mathrm{E}(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij}\, v_i h_j + \frac{1}{2}\sum_{i=1}^{V}(v_i - b_i)^2 - \sum_{j=1}^{H} a_j h_j, \quad (6)$$

The corresponding conditional probabilities become

$$p\big(h_j = 1|\mathbf{v}; \theta\big) = \sigma\left(\sum_{i=1}^{V} w_{ij}\, v_i + a_j\right), \quad (7)$$

$$p(v_i|\mathbf{h}; \theta) = N\left(\sum_{j=1}^{H} w_{ij}\, h_j + b_i, 1\right). \quad (8)$$

where v_i takes real values and follows a Gaussian distribution with mean $\sum_{j=1}^{H} w_{ij}\, h_j + b_i$ and variance one. Gaussian-Bernoulli RBMs can be used to convert real-valued stochastic variables to binary stochastic variables which can then be further processed using the Bernoulli-Bernoulli RBMs

Following the gradient of the log likelihood $\log p(\mathbf{v}; \theta)$ we obtain the update rule for the weights as [8]:

$$\Delta w_{ij} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}, \quad (9)$$

where $\langle v_i h_j \rangle_{data}$ is the expectation observed in the training set and $\langle v_i h_j \rangle_{model}$ is that same expectation under the distribution defined by the model. Unfortunately, $\langle v_i h_j \rangle_{model}$ is extremely expensive to compute exactly so the contrastive divergence (CD) approximation to the gradient is used where $\langle v_i h_j \rangle_{model}$ is replaced by running the Gibbs sampler initialized at the data for one full step [9].

2.2. Deep Belief Networks

From the decoding point of view, a DBN can be treated as a multi-layer perceptron with many layers. The input signal is processed layer by layer with Eq. (4) till the final layer, whose output is then transformed into a multinomial distribution using the softmax operation

$$\mathrm{p}(l = k|\mathbf{h}; \theta) = \frac{exp\big(\sum_{i=1}^{H} \lambda_{ik} h_i + a_k\big)}{Z(\boldsymbol{h})}, \quad (10)$$

where $l = k$ denotes the input been classified into the k-th class, and λ_{ik} is the weight between hidden unit h_i at the last layer and class label k.

As the baseline system, we use the conventional frame-level criterion to train the DBNs. Specifically, we adopt the strategy proposed in [13][8][9] for training DBN weights. That is, we first train a stack of RBMs in a generative manner and then fine-tune all the parameters jointly using back-propagation algorithm by maximizing the frame-level cross-entropy between the true and the predicted probability distributions over class labels.

3. Full-Sequence Training of DBNs

The standard discriminative back-propagation step as used in [8][9][13] optimizes the log posterior probability $p(l_t|v_t)$ of class labels given the current input, both at time-frame t (which may be a fixed local block of frames). We call this method of training DBNs a frame-based approach, because it uses the frame (or frame-block) only to predict the class labels. It does not explicitly use the fact that the neighboring frames (or frame-blocks) have smaller distances between the assigned probability distributions over class labels. To take this fact into account we model the probability of the whole sequence of labels given the whole utterance $p(l_{1:T}|v_{1:T})$.

The approach we take in this paper is to consider the top-most layer of the DBN as a linear-chain conditional random field (CRF) with $\boldsymbol{h}_t$ as input features from the lower layer at time t. This new model can be seen as a modification of the deep-structured CRF of [19][20] where the lower multiple layers of CRFs are replaced by DBNs. The resulting architecture is shown in Fig. 1, which can be equivalently seen as shared DBNs unfolding over time.

The conditional probability of the full-sequence labels given the full-sequence input features in this sequential model is

$$p(l_{1:T}|v_{1:T}) = p(l_{1:T}|\boldsymbol{h}_{1:T}) = \frac{exp\big(\sum_{t=1}^{T}\gamma_{ij}\phi_{ij}(l_{t-1}, l_t) + \sum_{t=1}^{T}\sum_{d=1}^{D}\lambda_{l_t d} h_{td}\big)}{Z(\boldsymbol{h}_{1:T})}, \quad (11)$$

where the transition feature is

$$\phi_{ij}(l_{t-1}, l_t) = \begin{cases} 1 & \text{if } l_{t-1} = i \text{ and } l_t = j \\ 0 & \text{otherwise,} \end{cases} \quad (12)$$

γ_{ij} is the parameter associated with this transition feature, h_{td} is the d-th dimension of the hidden unit value at the t-th frame at the last layer $\boldsymbol{h}_t$, D is the dimension of (or number of units) at that hidden layer.

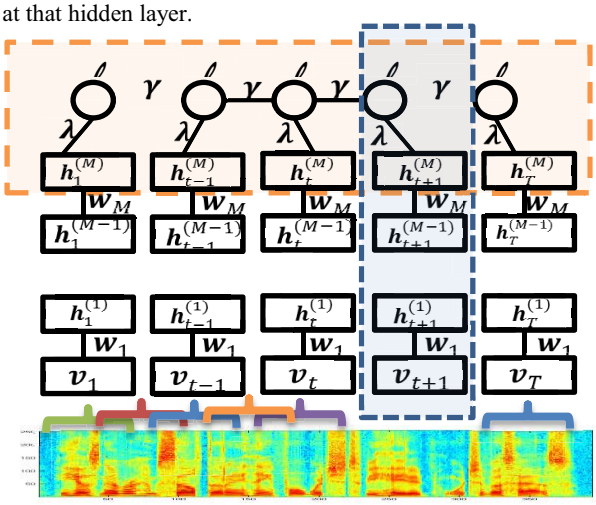

Fig. 1: Architecture of the DBN developed and evaluated in this work.

To optimize the log conditional probability $p(l_{1:T}^n|v_{1:T}^n)$ of the n-th utterance, we take the gradient over the activation parameters λ_{kd}, transition parameters γ_{ij}, and M-th-layer weights $w_{ij}^{(M)}$ as

$$\frac{\partial \log p(l_{1:T}^n|v_{1:T}^n)}{\partial \lambda_{kd}} = \sum_{t=1}^{T} \left(\delta(l_t^n = k) - p(l_t^n = k|v_{1:T}^n)\right) h_{td}^{(M),n} \quad (13)$$

$$\frac{\partial \log p(l_{1:T}^n|v_{1:T}^n)}{\partial \gamma_{ij}} = \sum_{t=1}^{T} \left(\delta(l_{t-1}^n = i, l_t^n = j) - p(l_{t-1}^n = i, l_t^n = j|v_{1:T}^n)\right) \quad (14)$$

$$\frac{\partial \log p(l_{1:T}^n|v_{1:T}^n)}{\partial w_{ij}^{(M)}} = \sum_{t=1}^{T} \left(\lambda_{l_t d} - \sum_{k=1}^{K} p(l_t^n = k|v_{1:T}^n)\lambda_{kd}\right) \cdot h_{td}^{(M),n}\left(1 - h_{td}^{(M),n}\right) h_{ti}^{(M-1),n} \quad (15)$$

Note that the gradient $\partial \log p(l_{1:T}^n|v_{1:T}^n)/\partial w_{ij}^{(m)}$ can be considered as back-propagating the error $\delta(l_t^n = k) - p(l_t^n = k|v_{1:T}^n)$ vs. $\delta(l_t^n = k) - p(l_t^n = k|v_t^n)$ in the frame-based training algorithm.

While the basic optimization algorithm with gradient descent can be succinctly described by Eqs. (13), (14), and (15), which compute the gradients in analytical forms, there are many practical issues to consider in the algorithm implementation.

First, the top-layer CRF's state transition parameters can form a transition matrix, which is different from that of the HMM. In fact, it is a combination of the transition matrix and the bi-phone LM scores. Without proper constraints, the transition matrix may have low likelihoods of being transitioning between states that are prohibited in the left-to-right three-state HMMs even though the training data does not support such transitions. To prevent this from happening so that a sharper model may be built, we enforce this constraint in the training by setting transition weights that are prohibited in the HMMs to have a very large negative value.

Second, since the weights in the DBNs are jointly optimized together with CRF's transition parameters, the optimization problem is no longer convex. For this reason, good initialization is crucial. The DBN weights can be initialized using the frame-based discriminative training. The transition parameters can be initialized from the combination of the HMM transition matrices and the LM scores, and can be further optimized by tuning the transition features while fixing the DBN weights before the joint optimization.

Third, there are two ways of doing decoding using the DBNs trained above. The simpler approach is to feed the log marginal probability $\log p(l_t|v_{1:T})$ as the activation scores to the conventional HMM decoder and use the HMM transition matrices and LM scores in the normal way. This approach may work when the full-sequence training can improve the quality of $\log p(l_t|v_{1:T})$. This approach is not optimal since it does not closely match the criterion we are optimizing for. A more effective approach is to generate the state sequence first and then to map the state sequence to the phoneme sequence. The decoding result may be further improved if we allow for insertion penalties, which can be easily integrated into the decoder by modifying the transition parameters by

$$\hat{\gamma}_{ij} = \rho\gamma_{ij} + \varphi \quad (16)$$

if state i is the final state of a phone and state j is the first state of a phone, where φ is the insertion penalty, and ρ is the scaling factor.

4. Experiments and Results

4.1. Experimental Setup

Phone recognition experiments were performed on the TIMIT corpus. The standard training set consisting of 462 speakers was used for training DBNs, with all SA removed. The standard development set of 50 speakers was used for model tuning. Results are reported using the standard 24-speaker core test set consisting of 192 sentences with 7,333 phone tokens. The speech was analyzed using a 25-ms Hamming window with a 10-ms fixed frame rate. In all the experiments, we represented the speech using first- to 12th-order Mel frequency cepstral coefficients (MFCCs) and energy, along with their first and second temporal derivatives. The data were normalized to have zero mean and unit variance. All experiments used a context window or block of 11 frames as the visible states. We used 183 target class labels (i.e., three states for each of the 61 phones). After decoding, the 61 phone classes were mapped to a standard set of 39 classes for scoring using the HResults tool in HTK, in the same way as other standard TIMIT experiments (e.g., [5]).

4.2. DBN Training

All of our experiments used a bigram language model over phones, which was jointly estimated with all DBN parameters using the training set. The decoder parameters were tuned to optimize performance on the development set for each run using grid search. Training of the DBNs was accelerated by exploiting a graphics processor. A single pass over the entire training set during pre-training took about 3-4 minutes per layer. An epoch of fine-tuning with back-propagation took around 10 and 25 minutes with frame-based, and sequence-based criteria, respectively.

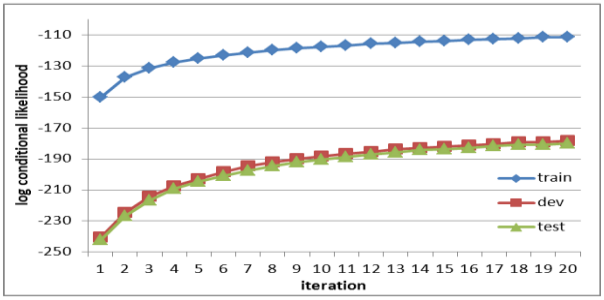

Fig. 2: Log conditional likelihood values in fine tuning

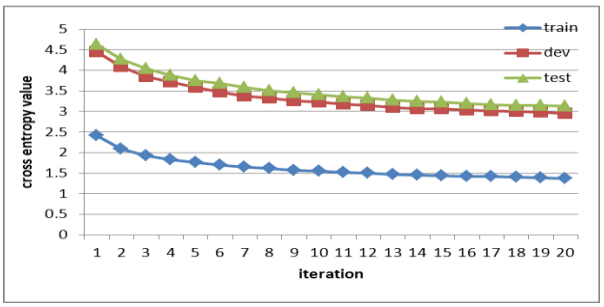

Fig. 3: Cross-entropy values observed in fine tuning

The behavior of the training process is carefully examined. As an example, we plot in Fig. 2 the training objective -- the log conditional likelihood -- as a function of fine-tuning epochs in the training, development, and test sets, respectively. The same plot for cross entropy is shown in Fig. 3.

4.3. Phonetic Recognition Results

Table 1 shows TIMIT phone recognition accuracy using frame-based and sequence-based training criteria with three-layer DBNs (i.e., one hidden layer). The new full-sequence training improves phone recognition accuracy for both dev and test sets, although more for the dev set. A relatively small amount of tuning is required to achieve such improvement.

Table 1. Comparative TIMIT phone recognition accuracy with three-layer DBNs

Three-layer DBN	Dev	Core Test
Frame-based	75.60%	73.87%
Sequence-based	76.82%	74.10%

When the DBN goes deeper to six layers, the recognizer's accuracy is sharply increased to over 77% for the frame-based system, as shown in the first row of Table 2, consistently with the finding in [13]. Without careful tuning of the training parameters, the full-sequence training was not able to improve recognition accuracy in the initial set of experiments. Careful adjustment of a set of parameters based on the dev set gives rise to higher accuracy. One of the parameters is the mini-batch size (i.e. the number of sentences in each mini batch) used in the gradient descent training of the DBN weights and integrated CRF/LM transition weights. With adjustment of the corresponding learning rate, use of increasing mini-batch sizes improves the accuracy to 77.81% on the core test set, as shown in Table 2. We also observed the importance of constraining the transition parameters. Without such a constraint, the accuracy drops to 76.22%.

Further optimization of the pre-training and fine-tuning steps [13] for frame-based training is able to improve the core-set accuracy to 77.77%. A rather different kind of optimization on the training procedure enables the sequence-based training to improve the core-set accuracy to 78.25% in our initial exploration. One step in this new training procedure is to use the DBN weights learned from sequence-based training to replace the randomization in the pre-training.

Table 2. Comparative TIMIT phone recognition accuracy with six-layer DBNs.

Six-layer DBN		Dev	Core Test
Frame-based		78.89%	77.15%
Sequence-based	MB Size=1	78.61%	76.42%
	MB Size=5	79.30%	77.31%
	MB Size=150	79.01%	77.62%
	MB Size=800	78.79%	77.81%

5. Conclusions

This paper presents our recent work on extending the DBN method of [13] from frame-based training to sequence-based training. Specifically, we develop and present a full-sequence discriminative learning approach to jointly optimizing the DBN weights, state-to-state transition parameters, and language model scores.

Setting up the full-sequence discriminative criterion to train the DBN weights gives rise to an architecture that can be considered as a modification of the deep-structured CRF described in of [19][20]. The top layer's linear-chain CRF remains the same but the lower layers of deep CRFs that are fully discriminative have now become largely generative during the pre-training stage with bi-directionally connected DBNs. Not only does this change overcome the difficulty for unsupervised learning in the intermediate layers of deep CRFs, but it also allows the use of longer windows in the input data. Further, DBNs appear to apportion the power of discriminative and generative modeling in a more balanced way than deep CRF. Further improvement includes using the most likely label sequences that are consistent with the reference during optimization rather than sticking to a single alignment as the reference label sequence. We are currently also exploring other successful techniques in speech recognition in the DBN framework, such as the use of object functions better correlating with errors than MMI [7], use of context dependency [1][2], and conditioning the DBN weights on specific auxiliary factors that control the variability in the speech data [21][22].

6. References

[1] Baker, J., Deng, L., Glass, J., Khudanpur, S., Lee, C. Morgan, N., and O'Shgughnessy, D. "Research developments and directions in speech recognition and understanding," *IEEE Sig. Proc. Mag.,* vol. 26, May 2009, pp. 75-80.

[2] Baker, J., Deng, L., Khudanpur, S., Lee, C., Glass, J., and Morgan, N. "Updated MINDS report on speech recognition and understanding," *IEEE Sig. Proc. Mag.*, July 2009, vol. 26.

[3] Bilmes, J. and Bartels, C. "Graphical model architectures for speech recognition," *IEEE Sig. Proc. Mag.*, vol. 22, Sept. 2005, pp. 89–100.

[4] Bridle, J. S. and Dodd, L., "An Alphanet approach to optimizing input transformations for continuous speech recognition" Proc. ICASSP, 1991.

[5] Deng, L., Yu, D., and Acero, A. "Structured speech modeling," IEEE Trans. Audio, Speech & Language Proc., vol. 14, Sept. 2006, pp. 1492-1504.

[6] Deng, L., Seltzer, M., Yu, D., Acero, A., Mohamed, A., and Hinton, G. "Binary coding of speech spectrograms using deep auto-encoder," Proc. Interspeech, 2010.

[7] He, X., Deng, L., Chou, W. "Discriminative Learning in Sequential Pattern Recognition --- A Unifying Review for Optimization-Oriented Speech Recognition," *IEEE Sig. Proc. Mag.*, vol. 25, 2008, pp. 14-36.

[8] Hinton, G., Osindero, S., and Teh, Y. "A fast learning algorithm for deep belief nets," Neural Computation, vol. 18, 2006, pp. 1527-1554.

[9] Hinton, G. and Salakhutdinov, R. "Reducing the dimensionality of data with neural networks," Science, vol. 313. no. 5786, 2006, pp. 504 – 507.

[10] Kingsbury, B. "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling" Proc. ICASSP, 2009.

[11] Konig, Y. "REMAP: Recursive Estimation and Maximization of A Posteriori Probabilities in Transition-based Speech Recognition", Ph.D. thesis, Univ. California-Berkeley, 1996.

[12] Lee, H., Largman, Y., Pham, P., Ng, A. "Unsupervised feature learning for audio classification using convolutional deep belief networks," Proc. NIPS, Dec. 2009.

[13] Mohamed, A.-R. Dahl, G., Hinton, G. "Deep belief networks for phone recognition," Proc. NIPS Workshop, Dec. 2009.

[14] Mohamed, A.-R. and Hinton, G. "Phone recognition using restricted Boltzmann machines", Proc. ICASSP, 2010.

[15] Morgan, N., Qifeng, Z., A. Stolcke, K. Sonmez, S. Sivadas, T. Shinozaki, M. Ostendorf, P. Jain, H. Hermansky, D. Ellis, G. Doddington, B. Chen, O. Cretin, H. Bourlard, and M. Athineos, "Pushing the envelope—Aside," *IEEE Signal Process. Mag.*, vol. 22, no. 5, pp. 81–88, 2005.

[16] Prabhavalkar, R., Fosler-Lussier, E. "Backpropagation training for multilayer conditional random field based phone recognition", Proc. ICASSP 2010, pp. 5534-5537.

[17] Schwarz, P., Matjka, P., and Cernock, J. "Hierarchical structures of neural networks for phoneme recognition," *Proc. ICASSP*, 2006, pp. 325–328.

[18] Yu, D. and Deng, L. "Deep-structured hidden conditional random fields for phonetic recognition," Proc. Interspeech, 2010.

[19] Yu, D., Deng, L., and Wang, S., "Learning in the deep-structured conditional random fields," Proc. NIPS Workshop, Dec. 2009.

[20] Yu, D., Wang, S., Karam, Z., Deng, L. "Language recognition using deep-structured conditional random fields," Proc. ICASSP, 2010, pp. 5030-5033.

[21] Yu, D., Deng, L., Gong, Y. and Acero, A. "A novel framework and training algorithm for variable-parameter hidden Markov models," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 17, 2009, pp. 1348-1360.

[22] Yu, D. and Deng, L. "Solving nonlinear estimation problems using Splines," *IEEE Sig. Proc. Mag.*, vol. 26, 2009, pp. 86-90.