Feature Space Maximum A Posteriori Linear Regression for Adaptation of Deep Neural Networks

Zhen Huang[1], Jinyu Li[2], Sabato Marco Siniscalchi[1,3], I-Fan Chen[1], Chao Weng[1], Chin-Hui Lee[1]

[1] School of ECE, Georgia Institute of Technology, Atlanta, GA. USA
[2] Microsoft Corporation, One Microsoft Way, Redmond, WA. USA
[3] Department of Telematics, Kore University of Enna, Enna, Italy

Abstract

We propose a feature space maximum *a posteriori* (MAP) linear regression framework to adapt parameters for context dependent deep neural network hidden Markov models (CD-DNN-HMMs). Due to the huge amount of parameters used in DNN acoustic models in large vocabulary continuous speech recognition, the problem of over-fitting can be severe in DNN adaptation, thus often impair the robustness of the adapted DNN model. Linear input network (LIN) as a straight-forward feature space adaptation method for DNN, similar to feature space maximum likelihood linear regression (fMLLR), can potentially suffer from the same robustness situation. The proposed adaptation framework is built based on MAP estimation of the LIN parameters by incorporating prior knowledge into the adaptation process. Experimental results on the Switchboard task show that against the speaker independent CD-DNN-HMM systems, LIN provides 4.28% relative word error rate reduction (WERR) and the proposed fMAPLIN method is able to provide further 1.15% (totally 5.43%) WERR on top of LIN.

Index Terms: deep neural network, speaker adaptation, maximum *a posteriori* estimation

1. Introduction

Recent successes in adopting context dependent deep neural network hidden Markov models (CD-DNN-HMMs) in automatic speech recognition (ASR) have demonstrated promising performance improvements over the conventional Gaussian mixture model (GMM) HMMs [1] in various tasks and datasets [2, 3, 4, 5, 6, 7]. Despite the advances by DNN, CD-DNN-HMMs still suffer from the same performance degradations as in CD-GMM-HMMs due to potential acoustic mismatches between the training and testing conditions.

Over the past, a large number of adaptation methods for CD-GMM-HMMs have been extensively investigated to address the aforementioned robustness problem [8], and many of these methods have had significant impacts on either research or commercial use. A recent survey on these adaptation methods can be found in [9]. For the conventional CD-GMM-HMMs systems, two families of adaptation parameters, direct model parameters and indirect transformation matrix parameters, have been developed. Techniques, such as maximum likelihood linear regression (MLLR) [10] and the maximum $a\ posteriori$ (MAP) [11], and their corresponding feature space variations (feature space MLLR (fMLLR) [12, 13], and feature space MAPLR (fMAPLR) [14, 15]) have also been widely used. For general artificial neural network (ANN) [16] HMM hybrid, known as connectionist systems [17] in which the CD-DNN-HMM is a special case, there are also many adaptation attempts, such as transformation-based [18, 19, 20, 21], conservative-based [22, 23, 24], subspace-based [25] and activation function based [26] approaches. Contrary to CD-GMM-HMM adaptation, the parameters to be adapted in CD-DNN-HMMs for large vocabulary continuous speech recognition (LVCSR) are often structurally embedded in a deep (multiple hidden layers) and wide (large number of neurons per layer) network and use a large output layer designed to model senones (tied-triphone states) [27, 4] directly. When the amount of adaptation data is limited, which is quite common in rapid speaker adaptation tasks, over-fitting is a severe issue. To address this problem, [28] adds KullbackLeibler divergence (KLD) regularization to the objective criterion to obtain a conservative adaptation scheme. Transformation based methods have also been employed in [29], and adaptation is only performed on the transform network in order to reduce the number of parameters to be optimized.

In this paper, a feature space maximum *a posteriori* (MAP) linear regression (LR) framework is proposed to adapt the parameters of CD-DNN-HMMs based on a linear transformation network (LIN) [18, 19]. We refer to the proposed technique as fMAPLIN. The LIN adaptation method, also called feature space LIN (fLIN), can be seen as a DNN version of conventional fMLLR in CD-GMM-HMM. Similar to fMLLR, it learns a feature transformation during adaptation. Such adaptation schemes in the feature space have the advantage of involving less parameters than those techniques that adapt either all DNN parameters or only the hidden/output layers which often have much more parameters than the input layer. Even when adaptation is only applied to the feature transform network, the over-fitting problem could still arise. The proposed adaptation framework is built based on MAPLR estimation of the LIN parameters by incorporating prior knowledge into the adaptation process which is similar to fMAPLR [15] in CD-GMM-HMM. It will be shown in this paper that applying the proposed fMAPLIN adaptation is equivalent to adding a regularization term to the loss function in DNN training. With the simple assumption of a standard joint normal prior density, the proposed method can be reduced to L2-regularized adaptation [30].

LIN and LIN-like adaptation methods, such as feature space discriminative linear regression (fDLR) [29, 31], and regularized conservative adaptation methods [28] have been proven to be effective for CD-DNN-HMM systems. We would like to investigate the way to do feature space MAP adaptation for CD-DNN-HMM systems. Experiments on the Switchboard task show that against the speaker independent CD-DNN-HMM systems, LIN provides 4.28% relative word error reduction and the proposed fMAPLIN method is able to provide further 1.15% (totally 5.43%) relative word error reduction on top of LIN. The

experiment results exhibit the advantage of MAP adaptation which incorporates essential prior information. In our formulation the prior also serves as a regularization and thus achieves conservative adaptation training.

2. Training of CD-DNN-HMMs

In this work, the input to DNN is a splice of a central frame (whose label is that for the splice) and its n context frames on both left and right sides, *e.g.,* $n = 10$. The hidden layers were constructed by sigmoid units, and the output layer is a softmax layer. The basic structure of a deep model is shown in Fig. 1. Specifically, the values of the nodes can be expressed as:

$$\mathbf{x}^i = \begin{cases} W_1\mathbf{o}^t + \mathbf{b}_1, & i = 1 \\ W_i\mathbf{y}^i + \mathbf{b}_i, & i > 1 \end{cases}, \tag{1}$$

$$\mathbf{y}^i = \begin{cases} \mathrm{sigmoid}(\mathbf{x}^i), & i < n \\ \mathrm{softmax}(\mathbf{x}^i), & i = n \end{cases}, \tag{2}$$

where W_1, and W_i are the weight matrices, $\mathbf{b_1}$, and $\mathbf{b_i}$ are the bias vectors, n is the total number of the hidden layers, and both $\mathrm{sigmoid}$ and $\mathrm{softmax}$ functions are element-wise operations. The vector $\mathbf{x}^i$ corresponds to pre-nonlinearity activations, and $\mathbf{y}^i$ and $\mathbf{y}^n$ are the vectors of neuron outputs at the i^{th} hidden layer and the output layer, respectively. The softmax outputs were considered as an estimate of the senone posterior probability:

$$P(C_j|\mathbf{o}_t) = \mathbf{y}_t^n(j) = \frac{\exp(\mathbf{x}_t^n(j))}{\sum_i \exp(\mathbf{x}_t^n(i))}, \tag{3}$$

where C_j represents the j^{th} senone and $\mathbf{y}^n(j)$ is the j^{th} element of $\mathbf{y}^n$ in Fig. 1.

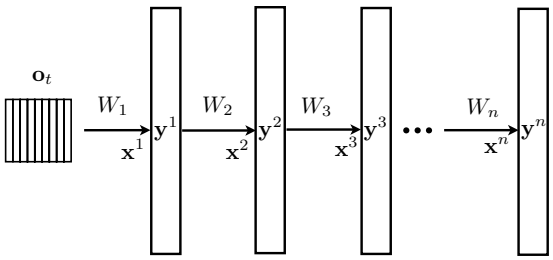

Figure 1: Structure of a deep neural network: W_i is weight matrix at the i^{th} layer, note that the bias terms are omitted for simplicity.

DNN is trained by maximizing the log posterior probability over the training frames. This is equivalent to minimizing the cross-entropy objective function. Let $\mathcal{X}$ be the whole training set, which contains N frames, *i.e.* $\mathbf{o}_{1:N} \in \mathcal{X}$, then the loss with respect to $\mathcal{X}$ is given by

$$\mathcal{L}_{1:N} = -\sum_{t=1}^{N}\sum_{j=1}^{J} \mathbf{d}_t(j) \log P(C_j|\mathbf{o}_t), \tag{4}$$

where $P(C_j|\mathbf{o}_t)$ is defined in Eq. (3); $\mathbf{d}_t$ is the label vector of frame t. In real practices of DNN systems, the label vector $\mathbf{d}_t$ is often obtained by a forced alignment with an existing system resulting in only the target entry that is equal to 1.

The objective function is minimized by using error backpropagation [32] which is a gradient-descent based optimization method developed for neural networks. Specifically, taking partial derivatives of the objective function with respect to the pre-nonlinearity activations of output layer $\mathbf{x}^n$, the error vector to be backpropagated to the previous hidden layers is generated:

$$\boldsymbol{\epsilon}_t^n = \frac{\partial \mathcal{L}_{1:N}}{\partial \mathbf{x}^n} = \mathbf{y}_t^n - \mathbf{d}_t, \tag{5}$$

the backpropagated error vector at the previous hidden layer is:

$$\boldsymbol{\epsilon}_t^i = W_{i+1}^T \boldsymbol{\epsilon}_t^{i+1} * \mathbf{y}^i * \left(\mathbf{1} - \mathbf{y}^i\right), i < n \tag{6}$$

where $*$ denotes element-wise multiplication. With the error vectors at certain hidden layers, the gradient over the whole training set with respect to the weight matrix W_i is given by

$$\frac{\partial \mathcal{L}_{1:N}}{\partial W_i} = \mathbf{y}_{1:N}^{i-1}(\boldsymbol{\epsilon}_{1:N}^i)^T. \tag{7}$$

Note that in the above equation, both $\mathbf{y}_{1:N}^{i-1}$ and $\boldsymbol{\epsilon}_{1:N}^i$ are matrices, which are formed by concatenating vectors corresponding to all the training frames from frame 1 to N, *i.e.* $\boldsymbol{\epsilon}_{1:N}^i = [\boldsymbol{\epsilon}_1^i, \ldots, \boldsymbol{\epsilon}_t^i, \ldots, \boldsymbol{\epsilon}_N^i]$. The batch gradient descent updates the parameters with the gradient in (7) only once after each sweep through the whole training set and in this way parallelization can be easily conducted. However, stochastic gradient descent (SGD) [33] usually works better in practice where the true gradient is approximated by the gradient at a single frame t, *i.e.*, $\mathbf{y}_t^{i-1}(\boldsymbol{\epsilon}_t^i)^T$, and the parameters are updated right after seeing each frame. The compromise between the two, the mini-batch SGD [34], is more widely used, as the reasonable size of mini-batches makes all the matrices fit into the GPU memory, leading to a computationally efficient learning process. Here we use mini-batch SGD to update the parameters.

Training a neural network directly from a set of randomly initialized parameters usually results in a poor local optimum when performing error backpropagation, especially when the neural network is deep [35]. To cope with this, pre-training methods have been proposed for a better initialization of the parameters [36] by growing the neural network layer by layer without using the label information. Treating each pair of layers in the network as a restricted Boltzmann machine (RBM), each layer of the neural network can then be trained using an objective criterion called contrastive divergence [36].

3. fMAPLIN adaptation of CD-DNN-HMMs

In this following, we incorporate prior information into plain LIN adaptation in a Bayesian MAPLR framework. This extension from LIN to fMAPLIN in CD-DNN-HMMs is similar to the extension from fMLLR to fMAPLR in CD-GMM-HMMs.

3.1. LIN adaptation

The original LIN adaptation as described in [18] for a ANN-HMM system plays the same role as fMLLR in a GMM-HMM system. It learns a linear transformation between the input feature vectors to the speaker independent (SI) ANNs. During recognition, the transformed vector is used as the input to the SI-ANN. When performing adaptation for a new speaker, the weights in the transformation network are initialized to an identity matrix. The SI-ANN network is kept "frozen" while only

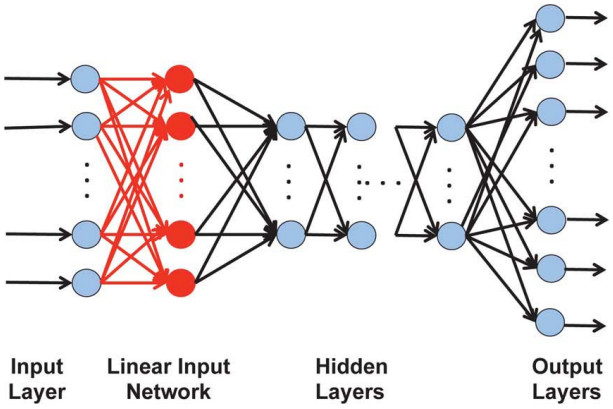

Figure 2: An architecture for adapting ANN-HMM models based on LIN. During adaptation, the parameters (weights) associated to the red links are estimated using the adaptation utterances while all other weights are kept fixed. The activation function for each LIN neuron (red node) is linear.

the transformation matrix is updated during adaptation by the conventional error back-propagation (BP). The basic concept of the LIN adaptation is illustrated in Fig. 2, in which the parameters of the red part of the DNN model are modified.

3.2. fMAPLIN adaptation

As described in Section 3.1, plain LIN adaptation is the same as general DNN training with the difference that only the transformation network's weights are updated. During adaption training, the LIN transformation network is trained to maximize the log posterior probability, or minimize the cross-entropy, over training samples with the loss function in Eq. (4). The label vector $\mathbf{d}_t$ is often obtained by a forced alignment from an existing transcription. Thus, a simplified loss function is obtained:

$$\mathcal{L}_{1:N}^{xent} = -\sum_{t=1}^{N} \log p(C_{tg}|\mathbf{o}_t), \tag{8}$$

where C_{tg} is the target senone at time t. Taking into account the weight matrix W_{upd} representing the transformation network's weights in the context of LIN adaptation and the entire network's weights in the general case of DNN training, Eq. (8) becomes:

$$\mathcal{L}_{1:N}^{xent} = -\sum_{t=1}^{N} \log p(C_{tg}|\mathbf{o}_t, W_{upd}), \tag{9}$$

and a complete posterior formulation should be:

$$\begin{aligned} \mathcal{L}_{1:N}^{MAP} &= -\sum_{t=1}^{N} \log p(C_{tg}, W_{upd}|\mathbf{o}_t) \\ &= -\log p(W_{upd}) - \sum_{t=1}^{N} \log p(C_{tg}|\mathbf{o}_t, W_{upd}) \\ &= -\log p(W_{upd}) + \mathcal{L}_{1:N}^{xent} \end{aligned} \tag{10}$$

where $p(W_{upd})$ is a joint prior probability density of the weights in W_{upd}. Similar to conventional MAP learning, the prior density can be scaled by a factor λ to control the influence degree of the prior [30]. Now Eq. (10) becomes,

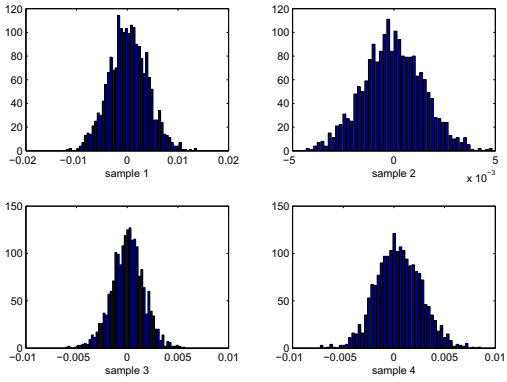

Figure 3: histograms of 4 sample weights

$$\mathcal{L}_{1:N}^{MAP} = -\lambda \log p(W_{upd}) + \mathcal{L}_{1:N}^{xent}. \tag{11}$$

Eq. (11) formulates the proposed MAP learning idea by adding a term of prior density $p(W_{upd})$ in the objective function to be optimized. Here we assume that the joint probability density of the weights to be Gaussian and all the weights are independent with each other (the reason for the assumptions will be explained later in Section 4).

By expressing the matrix W_{upd} as a vector w with each entry representing a particular weight, we have $p(W_{upd})$ as:

$$p(W_{upd}) = \frac{1}{(2\pi)^{M/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(w-\mu)^T\Sigma^{-1}(w-\mu)) \tag{12}$$

where only the diagonal entries of the covariance matrix Σ are non-zero (from the independence assumption of the weights). Now the loss function can be written as,

$$\mathcal{L}_{1:N}^{MAP} = \frac{\lambda}{2}(w-\mu)^T\Sigma^{-1}(w-\mu) + \mathcal{L}_{1:N}^{xent}. \tag{13}$$

A close look at Eq. (13), when the prior distribution is standard Gaussian $N(0|I)$, MAP learning will degenerate to conventional L2-regularized training.

The gradient of $\mathcal{L}_{1:N}^{MAP}$ with respect to w can now be expressed as:

$$\frac{\partial \mathcal{L}_{1:N}^{MAP}}{\partial w} = \lambda(w-\mu)^T diag(\Sigma^{-1}) + \frac{\partial \mathcal{L}_{1:N}^{xent}}{\partial w}, \tag{14}$$

where $diag(\Sigma^{-1})$ is a vector composed by the diagonal entries of Σ^{-1}) and $\frac{\partial \mathcal{L}_{1:N}^{xent}}{\partial w}$ can be computed using the conventional BP algorithm following Eq. (7).

4. Prior density estimation

The joint prior probability distribution $p(W_{upd})$ of the weights in W_{upd} is an essential part of the proposal fMAPLIN adaptation framework. To analyze and estimate the prior density, we utilized the training data of the baseline DNN. We treated each speaker in the training set as a sample speaker and a single iteration of supervised plain LIN adaptation was performed using 20 utterances of that speaker with a learning rate of 1e-04. After that, we can get a transformation matrix for each speaker. Using

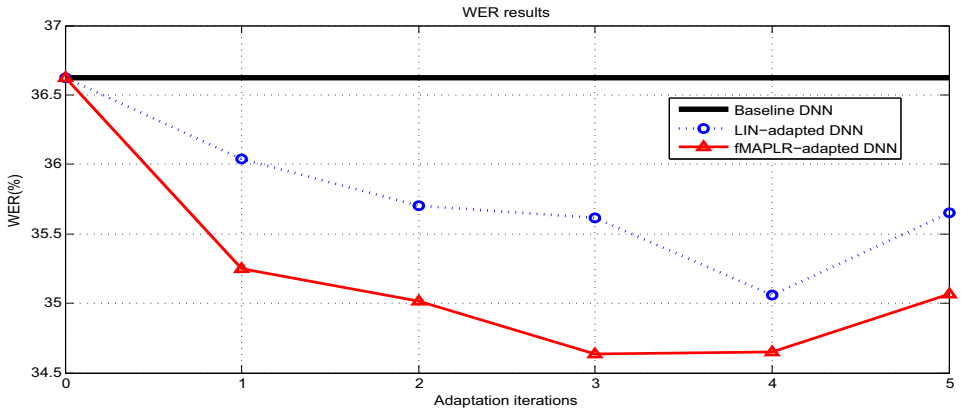

Figure 4: WER results for of LIN-adapted DNN and the proposed fMAPLIN adapted DNN.

the transformation matrices obtained for each speaker, we first examine the prior distribution of some weight parameters.

Fig. 3 shows the histograms of 4 sample weights over the speakers (nearly 2000 speakers). It can be seen that the distribution is quite like Gaussian, and the same result applies to all the sample weights that we do not report for the sake of saving space. This observation motivated us to assume the prior density of the weights to be joint Gaussian as discussed earlier. The other assumption that all the weights are independent of each other may not holds well in practice, but it leads to a diagonal precision matrix Σ^{-1} that reduces the computational burden.

5. Experiments and Results

We conducted a series of experiments on the Switchboard task using the Kaldi toolkit [37]. The baseline DNN model was trained using a 109-hour subset of the SWB-I training data [38] with an initial learning rate of 0.008 using cross-entropy objective function. It is initialized with stacked restricted Boltzmann machines (RBMs) by using layer by layer generative pretraining. An initial learning rate of 0.01 is used to train the Gaussian-Bernoulli RBM and a learning rate of 0.4 is applied to the Bernoulli-Bernoulli RBMs. The DNN has seven 2048-neuron hidden layers, and a 2979-neuron output softmax layer. The features used to train the DNN MFCC [39] with 39 dimensions, so the final input feature of 11 frames has 429 dimensions resulting in the size of LIN transformation matrix to be 429 x 429.

The adaptation and testing data were from the NIST 2000 Hub5 evaluation set [40]. There are 80 speakers in the Hub5 2000 testing dataset [40]. We selected the 40 CallHome [40] speakers for the experiments. The first 20 utterances of each selected speaker formed the adaptation set and the remaining utterances were used for testing. All adaptation experiments were performed in a supervised manner. The following adaptation scheme was established: 4 iterations of plain LIN adaptation to get a fairly good estimation of the feature transformation matrix for each speaker. Next, these transformation matrices were used to estimate a joint Gaussian prior distribution for the LIN weights. After getting the prior density, the proposed fMAPLIN adaptation is performed. The learning rate was set to 6.25e-05 for the adaptation iterations.

Fig. 4 shows that when no prior information is used (i.e., no regularization), LIN-based adaptation gets the best WER of 35.06% (4.28% relative WERR against the baseline DNN system) at the 4th iteration. In fMAPLIN adaptation, it reaches the best WER of 34.63% (5.43% WERR) at the 3th iteration. On top of LIN adaptation, the proposed fMAPLIN approach is able to provide a further 1.15% WERR.

6. Summary and Discussion

In this paper, a feature space maximum *a posteriori* linear regression framework is proposed for DNN adaptation. Experiments on the Switchboard task show that against the speaker independent CD-DNN-HMM systems, LIN provides 4.28% relative word error reduction and the proposed fMAPLIN method is able to provide further1.15% (totally 5.43%) relative WERR on top of LIN. The experiment results exhibit the advantage of MAP adaptation which incorporates essential prior information. In our formulation the prior also serves as a regularization and thus achieves conservative adaptation training.

Using the prior density form of the joint Gaussian distribution, we have a special case of the proposed framework with the commonly used L2-regularized training scheme, when the the joint Gaussian distribution is assumed to be standard (zero mean and identity covariance matrix). Other forms of prior densities, such as matrix variate normal density in [14], can also be adopted. In [41, 42], we formulated the L2 and L1 regularization on transformation matrix as the ridge and LASSO adaptation with enforced sparsity to handle limited adaptation data. We will connect the proposed method with those sparsity enforcement methods. In this paper, we estimate the prior density using the LIN transformation matrices obtained after several steps of plain LIN adaptation. Other schemes for estimating the prior density can also be employed, such as using a reliable hold-out development set. The strategy of performing fMAPLIN adaptation can also be flexible, for example, in a greedier way, the prior density can be re-estimated after each iteration of fMAPLIN.

7. Acknowledgment

The authors would like to thank our colleague You-Chi Cheng, Kehuang Li of Georgia Institute of Technology and Professor Ji Wu of Tsinghua University for valuable discussions and suggestions. We also want to thank Professor Bo Hong of Georgia Institute of Technology and his PhD student Jiadong Wu for helping us set up and utilize GPUs in DNN training.

8. References

[1] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[2] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Interspeech*, 2011, pp. 437–440.

[3] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A. Mohamed, "Making deep belief networks effective for large vocabulary continuous speech recognition," in *Proc. ASRU*, 2011, pp. 30–35.

[4] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[5] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[7] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. Interspeech*, 2013, pp. 2345–2349.

[8] C.-H. Lee and Q. Huo, "On adaptive decision rules and decision parameter adaptation for automatic speech recognition," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1241–1269, 2000.

[9] J. Li, L. D. Y. Gong, and R. Haeb-Umbach, "An overview of noise-robust automatic speech recognition," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 745–777, 2014.

[10] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Computer Speech & Language*, vol. 9, no. 2, pp. 171–185, 1995.

[11] J. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Trans. Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994.

[12] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[13] Y. Li, H. Erdogan, Y. Gao, and E. Marcheret, "Incremental online feature space MLLR adaptation for telephony speech recognition," in *Proc. Interspeech*, 2002.

[14] C. Chesta, O. Siohan, and C.-H. Lee, "Maximum a posteriori linear regression for hidden Markov model adaptation," in *Proc. Eurospeech*, 1999.

[15] X. Lei, J. Hamaker, and X. He, "Robust feature space adaptation for telephony speech recognition." in *Proc. Interspeech*, 2006.

[16] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[17] H. A. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*. Springer, 1994, vol. 247.

[18] J. Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, S. Renals, and T. Robinson, "Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system," in *Proc. Eurospeech*, 1995.

[19] V. Abrash, H. Franco, A. Sankar, and M. Cohen, "Connectionist speaker normalization and adaptation," in *Proc. Eurospeech*, 1995.

[20] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. D. Mori, "Linear hidden transformations for adaptation of hybrid ANN/HMM models," *Speech Communication*, vol. 49, no. 10, pp. 827–835, 2007.

[21] B. Li and K. C. Sim, "Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems," in *Proc. Interspeech*, 2010, pp. 526–529.

[22] J. Stadermann and G. Rigoll, "Two-stage speaker adaptation of hybrid tied-posterior acoustic models," in *Proc. ICASSP*, 2005.

[23] D. Albesano, R. Gemello, P. Laface, F. Mana, and S. Scanzio, "Adaptation of artificial neural networks avoiding catastrophic forgetting," in *Proc. IJCNN*, 2006, pp. 1554–1561.

[24] X. Li and J. Bilmes, "Regularized adaptation of discriminative classifiers," in *Proc. ICASSP*, 2006.

[25] S. Dupont and L. Cheboub, "Fast speaker adaptation of artificial neural networks for automatic speech recognition," in *Proc. ICASSP*, vol. 3, 2000, pp. 1795–1798.

[26] S. M. Siniscalchi, J. Li, and C.-H. Lee, "Hermitian polynomial for speaker adaptation of connectionist speech recognition systems," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2152–2161, 2013.

[27] M.-Y. Hwang and X. Huang, "Shared-distribution hidden Markov models for speech recognition," *IEEE Trans. Speech and Audio Processing*, vol. 1, no. 4, pp. 414–420, 1993.

[28] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *Proc. ICASSP*, 2013, pp. 7893–7897.

[29] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. ASRU*, 2011, pp. 24–29.

[30] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[31] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, and Y. Gong, "Adaptation of context-dependent deep neural networks for automatic speech recognition," in *Proc. Spoken Language Technology Workshop*, 2012, pp. 366–369.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*. MIT Press, Cambridge, MA, USA, 1988.

[33] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proc. ICML*, 2004, pp. 919–926.

[34] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction," in *Proc. ICML*, 2011, pp. 713–720.

[35] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Proc. AISTATS*, 2009, pp. 153–160.

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

[37] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovskỳ, G. Stemmer, and K. Veselỳ, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[38] J. J. Godfrey and E. Holliman, "Switchboard-1 release 2," *Linguistic Data Consortium, Philadelphia*, 1997.

[39] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[40] J. Fiscus, W. M. Fisher, A. F. Martin, M. A. Przybocki, and D. S. Pallett, "2000 NIST evaluation of conversational speech recognition over the telephone: English and mandarin performance results," in *Proc. Speech Transcription Workshop*, 2000.

[41] J. Li, Y. Tsao, and C.-H. Lee, "Shrinkage model adaptation in automatic speech recognition." in *Proc. INTERSPEECH*, 2010, pp. 1656–1659.

[42] J. Li, M. Yuan, and C.-H. Lee, "Lasso model adaptation for automatic speech recognition," in *Proc. ICML Workshop on Learning Architectures, Representations, and Optimization for Speech and Visual Information Processing*, 2011.