


AUTHOR QUERY FORM

	Journal: NEUCOM Article Number: 13731	Please e-mail or fax your responses and any corrections to: E-mail: corrections.esch@elsevier.macipd.com Fax: + 44 1392 285878
---	--	---

Dear Author,

Please check your proof carefully and mark all corrections at the appropriate place in the proof (e.g., by using on-screen annotation in the PDF file) or compile them in a separate list. Note: if you opt to annotate the file with software other than Adobe Reader then please also highlight the appropriate place in the PDF file. To ensure fast publication of your paper please return your corrections within 48 hours.

For correction or revision of any artwork, please consult <http://www.elsevier.com/artworkinstructions>.

Any queries or remarks that have arisen during the processing of your manuscript are listed below and highlighted by flags in the proof. Click on the [Q](#) link to go to the location in the proof.

Location in article	Query / Remark: click on the Q link to go Please insert your reply or correction at the corresponding line in the proof
Q1	Please confirm that given names and surnames have been identified correctly and are presented in the desired order.
Q2	Please check the telephone number and e-mail address (ksyao@hotmail.com) of the corresponding author, and correct if necessary.
Q3	Please check the years 2011 and 2013 in the biography of author Dong Yu.
Q4	Please provide biography for the author “Yifan Gong”.
Q5	Please provide heading for the first column in Tables 4 and 5.

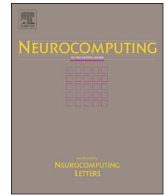
Thank you for your assistance.

Please check this box or indicate your approval
if you have no corrections to make to the PDF file



Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A fast maximum likelihood nonlinear feature transformation method for GMM–HMM speaker adaptation

Q1 Kaisheng Yao^{a,*}, Dong Yu^b, Li Deng^b, Yifan Gong^a

^a Online Service Division, Microsoft Corporation, One Redmond Way, Redmond 98052, WA, USA

^b Microsoft Research, One Redmond Way, Redmond 98052, WA, USA

ARTICLE INFO

Article history:

Received 4 September 2012

Received in revised form

22 December 2012

Accepted 12 February 2013

Keywords:

Extreme learning machine

Neural networks

Maximum likelihood

Speech recognition

Speaker adaptation

Hidden Markov models

ABSTRACT

We describe a novel maximum likelihood nonlinear feature bias compensation method for Gaussian mixture model–hidden Markov model (GMM–HMM) adaptation. Our approach exploits a single-hidden-layer neural network (SHLNN) that, similar to the extreme learning machine (ELM), uses randomly generated lower-layer weights and linear output units. Different from the conventional ELM, however, our approach optimizes the SHLNN parameters by maximizing the likelihood of observing the features given the speaker-independent GMM–HMM. We derive a novel and efficient learning algorithm for optimizing this criterion. We show, on a large vocabulary speech recognition task, that the proposed approach can cut the word error rate (WER) by 13% over the feature maximum likelihood linear regression (fMLLR) method with bias compensation, and can cut the WER by more than 5% over the fMLLR method with both bias and rotation transformations if applied on top of the fMLLR. Overall, it can reduce the WER by more than 27% over the speaker-independent system with 0.2 real-time adaptation time.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Automatic speech recognition (ASR) systems rely on powerful statistical techniques such as Gaussian mixture model–hidden Markov models (GMM–HMMs) [1] and deep neural network (DNN)–HMMs [2] to achieve good performance. Even though ASR systems are trained on large amounts of data, mismatches between the training and testing conditions are still unavoidable due to speaker and environment differences. Much like other systems built upon statistical techniques, ASR systems often fail to produce the same level of performance when tested under mismatched conditions.

A substantial amount of work has been conducted to improve ASR systems' performance under mismatched conditions. A comprehensive review of existing techniques can be found in [3]. The key idea of these techniques is to adapt either the model or the feature so that the mismatch between the training and testing conditions can be reduced. Here we briefly review the three techniques that motivated this work.

The first class of techniques adapts the model or feature by applying the physical model that causes the mismatch [4–6]. For example, the parallel model combination (PMC) method [4] adapts the speech models under noisy environment by combining

clean speech models and noise models. The joint additive and convolutive noise compensation (JAC) [5] method jointly compensates for the additive background noise and channel distortion by applying the vector Taylor series expansion of the physical distortion model around the mean vectors of clean speech, noise and channel. Methods in this category differ on the type of physical models used and the approximation applied. For instance, the JAC method may be further improved by including both the magnitude and phase terms in the physical model that describes the relationship between speech and noise [6].

The second class of techniques [7,8] adapts the model or feature by learning the distortion model from stereo data, i.e., data recorded simultaneously over a “clean” and “noisy” channel. Methods in this category differ in how the distortion model is estimated and used to compensate for the mismatches during testing.

In contrast to the above techniques, the third class of technique does not need an explicit model that describes how speech is distorted. Instead it learns a transformation directly from the adaptation data and the associated transcriptions that may contain errors. This transformation is then used either to modify the test features to make it closer to the training data or to adjust the HMM model so that it better represents the test data. Example methods in this category include stochastic matching [9] and the popular maximum likelihood linear regression (MLLR) [10]. This class of techniques is more flexible than the first class of techniques since it does not rely on a physical model, which may not be available when, for instance, the HMMs are trained on noisy

Q2 * Corresponding author. Tel.: +1 469 360 6986.
E-mail addresses: kaisheny@microsoft.com, ksyao@hotmail.com (K. Yao).

speech. This class of techniques is also less demanding than the second class of techniques since it does not require stereo data which are rarely available. For these reasons this class of techniques has been widely used in real world large vocabulary speech recognition (LVSR) systems while the first and second class of techniques are mainly successful in small tasks such as digits recognition in noisy conditions [11,12]. Due to the large amount of HMM parameters involved in LVSR systems, feature transformation is preferred over model adaptation [10] to reduce computational cost.

Effort has been made to extend the third class of techniques to deal with nonlinear distortions. For example, in [13] a single-hidden-layer neural network (SHLNN) was used to estimate the nonlinear transformations to adapt the HMM model or feature. A gradient ascent method was developed to estimate both the upper- and lower-layer weights of the neural network, which unfortunately, was slow to convergence and easy to overfit to the training data [13], especially since the adaptation dataset is typically very small. Furthermore, the neural network in [13] was trained to maximize the likelihood of the transformed feature instead of the observed feature, which requires considering the Jacobian of the feature transformation.

Recently, extreme learning machine (ELM) [14] was proposed to address the difficulty and inefficiency in training single-hidden-layer neural networks. There are two core components in the ELM: using randomly generated “fixed” connection between the input and hidden layer (called lower-layer weights) and using linear output units so that for minimum mean square error criterion closed-form solution is available when the lower-layer weights are fixed. The randomly generated lower-layer weights are independent of the training data and so ELM is less likely to overfit to the training data than the conventional neural networks. ELM has been used as building blocks for example in the deep convex network (DCN) [15,16].

In this paper we propose a novel nonlinear feature adaptation framework for LVSR exploiting the core ideas in ELM. More specifically, we estimate a nonlinear time-dependent bias using an ELM-like SHLNN to compensate for the mismatch between the training and testing data. In this SHLNN, the lower-layer weights are randomly generated and the output layer contains only linear units just as in the ELM. Different from ELM, however, the SHLNN in our task is optimized to maximize the likelihood of either the transformed feature, for which a closed-form solution is derived, or the observed feature, for which an efficient second-order Gauss-Newton method is developed.

Our approach belongs to the third class of adaptation technique just reviewed. It requires neither physical models nor stereo data and can be easily applied to LVSR tasks. Our proposed approach is also more powerful than the previously developed approaches because it can take advantage of the neighboring frames (e.g., [4,6,5,10,13] cannot) and uses nonlinear transformations (e.g., [17] does not). We show, on a large vocabulary speech recognition task, that the proposed approach can cut the word error rate (WER) by 13% over the feature maximum likelihood linear regression (fMLLR) method with bias compensation, and can cut the WER by more than 5% over the fMLLR method with both bias and rotation transformations if applied on top of the fMLLR. Overall, it can reduce the WER by more than 27% over the speaker-independent system.

The rest of the paper is organized as follows: in Section 2, we discuss the problem of feature space transformation for speaker adaptation. In Section 3 we present the training algorithms of the ELM-like SHLNN for maximizing the likelihood of the transformed feature and observed feature and describe a tandem scheme that applies the nonlinear transformation on top of the linearly transformed feature. Experimental results are reported in Section 4 to

demonstrate how hidden layer size and context window lengths affect the adaptation effectiveness and how the proposed approach outperforms fMLLR on an LVSR task. We conclude the paper in Section 5.

2. Speaker adaptation through nonlinear feature transformation

In contrast to the traditional feature space transformation [9,10,17] that is a linear transform of observation vector \mathbf{x}_t at time t , the method presented in this paper is a nonlinear feature transformation method.

The nonlinear feature transformation can be defined as $f(\bar{\mathbf{x}}_t, \boldsymbol{\phi}_t)$, where $\boldsymbol{\phi}_t$ is a meta feature vector that contains information such as signal-to-noise ratio and utterance length, $\bar{\mathbf{x}}_t \in R^{LD}$ is an expanded observation vector centered at $\mathbf{x}_t \in R^D$ with a context length of L . For example, for a context length of $L=3$, $\bar{\mathbf{x}}_t = (\mathbf{x}_{t-1}^T \mathbf{x}_t^T \mathbf{x}_{t+1}^T)^T$, where superscript T denotes transpose. Usually, $\bar{\mathbf{x}}_t$ is augmented with element 1 to become $(\bar{\mathbf{x}}_t^T 1)^T$.

For speech recognition, the transformation $f(\bar{\mathbf{x}}_t, \boldsymbol{\phi}_t)$ is typically optimized to improve the likelihood of the observed feature $p(\mathbf{x}_t | \theta, \Lambda_x)$. Here θ denotes parameters of the nonlinear transformation and Λ_x denotes speaker independent HMM parameters that include (GMM) parameters $\{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m; m = 1, \dots, M\}$ and state transition probabilities $\{c_{ij}; i, j = 1, \dots, S\}$, where M is the total number of Gaussian components, S is the number of states, and $\pi_m, \boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ are the weight, mean vector and diagonal covariance matrix, respectively, of the m -th Gaussian component.

Directly improving the above likelihood is difficult. We instead iteratively estimate θ through the following auxiliary function:

$$Q(\theta; \tilde{\theta}) = -\frac{1}{2} \sum_{m,t} \gamma_m(t) (f(\bar{\mathbf{x}}_t, \boldsymbol{\phi}_t) - \boldsymbol{\mu}_m)^T \boldsymbol{\Sigma}_m^{-1} (f(\bar{\mathbf{x}}_t, \boldsymbol{\phi}_t) - \boldsymbol{\mu}_m) + \sum_{m,t} \gamma_m(t) \log |\mathbf{J}_t| \quad (1)$$

where $\tilde{\theta}$ is the previous estimate of θ , $\gamma_m(t)$ is the posterior probability of Gaussian m at time t , given HMM parameters Λ_x and the previous estimates of the nonlinear transformation parameters $\tilde{\theta}$. $\mathbf{J}_t = (\partial f(\bar{\mathbf{x}}_t, \boldsymbol{\phi}_t)) / \partial \boldsymbol{\phi}_t$ is the Jacobian of the feature transformation at time t . Note that we assume that the feature transformation is not dependent on a particular Gaussian m .

For speaker adaptation, observation vectors \mathbf{x}_t are from a particular speaker. Increasing the above auxiliary function scores corresponds to learning a speaker dependent transformation that increases the likelihood of observation \mathbf{x}_t , given the GMM parameters.

Theoretically the transformation function $f(\bar{\mathbf{x}}_t, \boldsymbol{\phi}_t)$ can be any function. In this paper we focus on compensating for the bias change, i.e.,

$$\mathbf{y}_t = f(\bar{\mathbf{x}}_t, \boldsymbol{\phi}_t) = \mathbf{x}_t + g(\bar{\mathbf{x}}_t, \boldsymbol{\phi}_t), \quad (2)$$

where $g(\bar{\mathbf{x}}_t, \boldsymbol{\phi}_t)$ is a nonlinear function compensating for the bias and is defined by the SHLNN shown in Fig. 1. Similar to the ELM, the SHLNN consists of a linear input layer, a sigmoid nonlinear hidden layer, and a linear output layer. The SHLNN parameters θ include a $D \times K$ matrix \mathbf{U} , a $K \times LD$ matrix \mathbf{W} and a scalar value α , where \mathbf{U} is the upper layer weights connecting the hidden layer and the output layer, \mathbf{W} is the lower layer weights connecting the input layer and the hidden layer, and α is the parameter to control steepness of the sigmoid function at the hidden layer. Since $\boldsymbol{\phi}_t$ can be considered as part of the observation we merge $\boldsymbol{\phi}_t$ into $\bar{\mathbf{x}}_t$ from now on to simplify notation.

Eq. (2) can be further rewritten and simplified to

$$\mathbf{y}_t = \mathbf{x}_t + \mathbf{U}\mathbf{h}_t \quad (3)$$

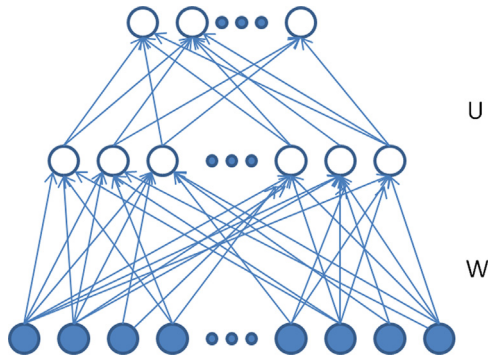


Fig. 1. Neural network architecture.

where $\mathbf{h}_t = \theta(\alpha \mathbf{W} \bar{\mathbf{x}}_t)$ is the hidden layer output and $\theta(x) = 1/(1 + \exp(-x)) = \exp(x)/(1 + \exp(x))$ is the sigmoid function, by noting that, following ELM, the lower-layer weight \mathbf{W} is randomly generated and is independent of the adaptation data and the training criterion. The scalar value α is typically set to a value around 1.0. In this paper, it is set to 0.6.¹

3. The algorithms for maximum likelihood estimation

As we discussed in the last section, the only parameter set we need to learn is \mathbf{U} , whose algorithm will be derived in this section.

Substituting (3) into (1), we have

$$Q(\theta; \tilde{\theta}) = -\frac{1}{2} \sum_{m,t} \gamma_m(t) (\mathbf{U} \mathbf{h}_t - \boldsymbol{\rho}_{mt})^T \boldsymbol{\Sigma}_m^{-1} (\mathbf{U} \mathbf{h}_t - \boldsymbol{\rho}_{mt}) + \sum_t \gamma_m(t) \log |\mathbf{J}_t|, \quad (4)$$

where $\boldsymbol{\rho}_{mt} = \boldsymbol{\mu}_m - \mathbf{x}_t$. The Jacobian $\mathbf{J}_t = (\partial \mathbf{y}_t / \partial \mathbf{x}_t) \in \mathbb{R}^{D \times D}$ is

$$\mathbf{J}_t = \mathbf{I}_{D \times D} + \alpha \mathbf{U} \mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c \quad (5)$$

where $\mathbf{W}_c \in \mathbb{R}^{K \times D}$ is a submatrix of \mathbf{W} , whose rows correspond to the center observation \mathbf{x}_t in the context window, $\mathbf{H}_t \in \mathbb{R}^{K \times K}$ is a diagonal matrix with element $\theta(\alpha \mathbf{W} \bar{\mathbf{x}}_t)$, and $\mathbf{I}_{\times l} \in \mathbb{R}^{l \times l}$ is an identity matrix.

\mathbf{U} can be learned with the gradient ascent algorithm, which iteratively updates an estimate of \mathbf{U} as

$$\mathbf{U} \leftarrow \mathbf{U} + \eta \frac{\partial Q}{\partial \mathbf{U}}, \quad (6)$$

where η is the step size,

$$\frac{\partial Q}{\partial \mathbf{U}} = -\sum_{m,t} \gamma_m(t) \boldsymbol{\Sigma}_m^{-1} (\mathbf{U} \mathbf{h}_t - \boldsymbol{\rho}_{mt}) \mathbf{h}_t^T + \sum_{m,t} \gamma_m(t) \frac{\partial \log |\mathbf{J}_t|}{\partial \mathbf{U}} \quad (7)$$

and

$$\begin{aligned} \text{vec} \left(\frac{\partial \log |\mathbf{J}_t|}{\partial \mathbf{U}} \right)^T &= \text{vec}(\mathbf{J}_t^{-T})^T \frac{\partial \mathbf{J}_t}{\partial \mathbf{U}} \\ &= \text{vec}(\mathbf{J}_t^{-T})^T [\alpha (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T \otimes \mathbf{I}_{D \times D}] \end{aligned} \quad (8)$$

where $\text{vec}(\cdot)$ is a vectorization operation to stack columns of a matrix. To obtain $(\partial \log |\mathbf{J}_t| / \partial \mathbf{U}) \in \mathbb{R}^{D \times K}$, we need to reshape the above vector to a $D \times K$ matrix.

¹ The scalar value α was experimented with 0.6 and 1.0. We did not observe much performance differences. Hence, this paper only reports results in Section 4 with $\alpha = 0.6$.

3.1. Maximum likelihood estimation on transformed features

We rewrite the auxiliary function as

$$Q(\mathbf{U}, \tilde{\mathbf{U}}) = -\frac{1}{2} \sum_{m,t} \gamma_m(t) (\mathbf{U} \mathbf{h}_t - \boldsymbol{\rho}_{mt})^T \boldsymbol{\Sigma}_m^{-1} (\mathbf{U} \mathbf{h}_t - \boldsymbol{\rho}_{mt}) \quad (9)$$

by ignoring the Jacobian, where $\tilde{\mathbf{U}}$ is the previous estimate of the upper layer weights \mathbf{U} . This simplified auxiliary function maximizes the likelihood of the transformed data $p(\mathbf{y}_t | \mathbf{U}, \Lambda_x)$, rather than the observed data $p(\mathbf{x}_t | \mathbf{U}, \Lambda_x)$.

A closed-form solution can be obtained for the above optimization problem by running one iteration of Gauss-Newton update and setting the initial $\mathbf{U} = 0$. This is because the criterion (9) is a quadratic function of \mathbf{U} and its gradient is linear. Using the Gauss-Newton method, we have

$$\begin{aligned} \text{vec}(\mathbf{U}) &= -\left(\frac{\partial^2 Q}{\partial \mathbf{U}^2} \right)^{-1} \text{vec} \left(\frac{\partial Q}{\partial \mathbf{U}} \right) \Big|_{\mathbf{U}=0} \\ &= -\left(\frac{\partial^2 Q}{\partial \mathbf{U}^2} \right)^{-1} \text{vec} \left(\sum_{m,t} \gamma_m(t) \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\rho}_{mt} \mathbf{h}_t^T \right) \Big|_{\mathbf{U}=0} \end{aligned} \quad (10)$$

where

$$\frac{\partial^2 Q}{\partial \mathbf{U}^2} \Big|_{\mathbf{U}=0} = -\sum_{m,t} \gamma_m(t) (\mathbf{h}_t \mathbf{h}_t^T \otimes \boldsymbol{\Sigma}_m^{-1}) \quad (11)$$

using matrix calculus [18] and \otimes is the Kronecker product.

The term (11) can be written in detail as

$$\begin{aligned} &\sum_{m,t} \gamma_m(t) (\mathbf{h}_t \mathbf{h}_t^T \otimes \boldsymbol{\Sigma}_m^{-1}) \\ &= \sum_{m,t} \gamma_m(t) \begin{pmatrix} h_{t1} h_{t1} \frac{1}{\sigma_{m1}^2} & \dots & 0 & \dots & h_{t1} h_{tK} \frac{1}{\sigma_{m1}^2} & \dots & 0 \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ 0 & \dots & h_{t1} h_{t1} \frac{1}{\sigma_{m2}^2} & \dots & 0 & \dots & h_{t1} h_{tK} \frac{1}{\sigma_{m2}^2} \\ h_{t2} h_{t1} \frac{1}{\sigma_{m1}^2} & \dots & 0 & \dots & h_{t2} h_{tK} \frac{1}{\sigma_{m1}^2} & \dots & 0 \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ 0 & \dots & h_{t2} h_{t1} \frac{1}{\sigma_{m2}^2} & \dots & 0 & \dots & h_{t2} h_{tK} \frac{1}{\sigma_{m2}^2} \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ h_{tK} h_{t1} \frac{1}{\sigma_{m1}^2} & \dots & 0 & \dots & h_{tK} h_{tK} \frac{1}{\sigma_{m1}^2} & \dots & 0 \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ 0 & \dots & h_{tK} h_{t1} \frac{1}{\sigma_{m2}^2} & \dots & 0 & \dots & h_{tK} h_{tK} \frac{1}{\sigma_{m2}^2} \end{pmatrix} \end{aligned} \quad (12)$$

where h_{ti} denotes the i -th element in the hidden layer output \mathbf{h}_t . The memory for evaluating this term is $O(DK \times DK)$. For a setup with $D=39$ and $K=2000$, the cost is 6.09×10^9 floating points, which corresponds to 48 G byte of memory. A typical computer cannot hold such large memory.

Notice that the above terms (11) and (14) are full but sparse matrices because $\boldsymbol{\Sigma}_m$ is diagonal. Fortunately a memory and CPU efficient algorithm can be derived by using the permutation matrix [18]. The intuition behind the algorithm is re-arranging the above full but sparse matrices in (14) to be block-diagonal. A permutation matrix $\mathbf{T}_{m,n} \in \mathbb{R}^{mn \times mn}$ is a matrix operator composed of 0s and 1s, with a single 1 on each row and column. It has a number of special properties. Particularly, $\mathbf{T}_{m,n}^{-1} = \mathbf{T}_{n,m} = \mathbf{T}_{m,n}^T$ and $\mathbf{T}_{m,n} \text{vec}(A) = \text{vec}(A^T)$.

Note that

$$\mathbf{h}_t \mathbf{h}_t^T \otimes \boldsymbol{\Sigma}_m^{-1} = \mathbf{T}_{K,D} \boldsymbol{\Sigma}_m^{-1} \otimes \mathbf{h}_t \mathbf{h}_t^T \mathbf{T}_{D,K}. \quad (13)$$

Substituting this to (11), we have

$$\frac{\partial^2 Q}{\partial \mathbf{U}^2} \Big|_{\mathbf{U}=0} = -\mathbf{T}_{K,D} \left(\sum_{m,t} \gamma_m(t) \boldsymbol{\Sigma}_m^{-1} \otimes \mathbf{h}_t \mathbf{h}_t^T \right) \mathbf{T}_{D,K} \quad (14)$$

By applying (14) to (10) we get

$$\text{vec}(\mathbf{U}) = \mathbf{T}_{K,D} \left(\sum_{mt} \gamma_m(t) \boldsymbol{\Sigma}_m^{-1} \otimes \mathbf{h}_t \mathbf{h}_t^T \right)^{-1} \mathbf{T}_{D,K} \text{vec} \left(\frac{\partial Q}{\partial \mathbf{U}} \right) \Big|_{\mathbf{U}=\mathbf{0}}$$

After some manipulations detailed in Appendix A.1, we get the closed-form solution at d -th row of \mathbf{U} , $\mathbf{U}_{d,:}$, for $d = \{1, \dots, D\}$ as

$$\text{vec}(\mathbf{U}_{d,:}) = \left(\sum_{mt} \gamma_m(t) \frac{1}{\sigma_{md}^2} \mathbf{h}_t \mathbf{h}_t^T \right)^{-1} \text{vec} \left(\frac{\partial Q}{\partial \mathbf{U}_{d,:}} \right) \Big|_{\mathbf{U}=\mathbf{0}}, \quad d = \{1, \dots, D\} \quad (15)$$

where σ_{md}^2 is the (d,d) -th element of the diagonal covariance matrix $\boldsymbol{\Sigma}_m$.

This corresponds to re-arranging the term (14) into the following block diagonal matrix:

$$\sum_{mt} \gamma_m(t) \begin{pmatrix} h_{t1} h_{t1} \frac{1}{\sigma_{m1}^2} & \dots & h_{t1} h_{tK} \frac{1}{\sigma_{m1}^2} & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ h_{tK} h_{t1} \frac{1}{\sigma_{m1}^2} & \dots & h_{tK} h_{tK} \frac{1}{\sigma_{m1}^2} & \dots & 0 & \dots & 0 \\ 0 & \dots & 0 & \dots & \ddots & \dots & 0 \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & h_{t1} h_{t1} \frac{1}{\sigma_{md}^2} & \dots & h_{t1} h_{tK} \frac{1}{\sigma_{md}^2} \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & h_{tK} h_{t1} \frac{1}{\sigma_{md}^2} & \dots & h_{tK} h_{tK} \frac{1}{\sigma_{md}^2} \end{pmatrix} \quad (16)$$

Keeping only the nonzero block diagonal matrices, now the total memory cost is $O(D \times K \times K)$. In terms of the computational cost, the matrix inversion $(\partial^2 Q / \partial \mathbf{U}^2)^{-1}$ in (10) is on a matrix of dimension $KD \times KD$. Using the new algorithm, the matrix inversion $(\sum_{mt} \gamma_m(t) (1/\sigma_{md}^2) \mathbf{h}_t \mathbf{h}_t^T)^{-1}$ in (15) is on block diagonal matrices with dimension of $K \times K$. Notice that the computational cost of a typical matrix inversion algorithm such as Gauss-Jordan elimination [19] is $O(n^3)$ for a matrix of dimension $n \times n$. Therefore, the new algorithm has the computational cost of $O(D \times K^3)$ for the matrix inversion whereas the matrix inversion in (10) costs $O(D^3 \times K^3)$ operations.

3.2. Maximum likelihood estimation on observed features

To optimize for the observed features we cannot drop the Jacobian term in (4) and need to solve it using an iterative Gauss-Newton procedure

$$\mathbf{U} \leftarrow \tilde{\mathbf{U}} - \eta \left(\frac{\partial Q^2}{\partial \tilde{\mathbf{U}}} \right)^{-1} \frac{\partial Q}{\partial \tilde{\mathbf{U}}} \Big|_{\mathbf{U}=\tilde{\mathbf{U}}} \quad (17)$$

This procedure starts from some initial estimates, such as $\mathbf{U} = \mathbf{0}$, until a certain number of iterations is reached.

We keep the Hessian term from (11) unchanged, but include a first order differential term from (8). Applying permutation matrices as shown in Appendix A.2, we can rewrite (8) as

$$\text{vec}(\mathbf{J}_t^{-T})^T [\alpha (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T \otimes \mathbf{I}_{D \times D}] = \alpha (\text{vec}(\mathbf{J}_t^{-1})^T \mathbf{I}_{D \times D} \otimes (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T)^T \quad (18)$$

which is a vector of dimension $1 \times DK$. Its d -th subvector with dimension K is $\alpha (\mathbf{J}_t^{-1})_{:,d}^T (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T$.

Applying permutation matrices similarly as in the above section, we can write (17) as follows:

$$\text{vec}(\mathbf{U}_{d,:}) = \text{vec}(\tilde{\mathbf{U}}_{d,:}) + \eta \left(\sum_{mt} \gamma_m(t) \frac{1}{\sigma_{md}^2} \mathbf{h}_t \mathbf{h}_t^T \right)^{-1} \times \text{vec} \left(\frac{\partial Q}{\partial \tilde{\mathbf{U}}_{d,:}} \right) \Big|_{\mathbf{U}=\tilde{\mathbf{U}}}, \quad d = \{1, \dots, D\} \quad (19)$$

To obtain $\text{vec}(\partial Q / \partial \mathbf{U}_{d,:})$, we substitute (18) into (7); i.e.

$$\text{vec} \left(\frac{\partial Q}{\partial \mathbf{U}_{d,:}} \right) = - \sum_{mt} \gamma_m(t) (\boldsymbol{\Sigma}_m^{-1} (\mathbf{U} \mathbf{h}_t - \boldsymbol{\rho}_{mt}) \mathbf{h}_t^T)_{:,d} + \sum_{mt} \gamma_m(t) \alpha (\mathbf{J}_t^{-1})_{:,d}^T (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T \quad (20)$$

Note that here we assume that Hessian $\partial^2 Q / \partial \mathbf{U}^2$ is unchanged once it is estimated from the point $\mathbf{U} = \mathbf{0}$ to speed up computation. The Gauss-Newton estimate can be initialized from either $\mathbf{U} = \mathbf{0}$ or the closed-form solution (15). The latter provides a better initial estimate and requires fewer iterations to estimate \mathbf{U} .

3.3. A tandem scheme

The proposed nonlinear bias compensation approach can be applied on top of the affine fMLLR [10]

$$\hat{\mathbf{x}}_t = \mathbf{A} \mathbf{x}_t + \mathbf{b} = \boldsymbol{\Psi} \boldsymbol{\xi}_t \quad (21)$$

to further improve the performance, where $\mathbf{A} \in R^{D \times D}$ is the transformation that does rotation on the input \mathbf{x}_t at time t , $\mathbf{b} \in R^D$ is the bias to the input, and $\boldsymbol{\Psi} = [\mathbf{A}; \mathbf{b}] \in R^{D \times (D+1)}$ is the combined affine transformation. $\boldsymbol{\xi}_t = [\mathbf{x}_t^T \mathbf{1}]^T$ is the augmented observation vector at time t . This tandem scheme can take advantage of neighboring frames and compensate for remaining nonlinear mismatches after the affine fMLLR.

3.4. Computational costs

We list the computational costs to implement our algorithms in Table 1. The forward propagation process in each method estimates values such as $\mathbf{U} \mathbf{h}_t$. The backward propagation process re-estimates neural network parameters \mathbf{U} and \mathbf{W} . We use N to denote the number of observation frames. As described in Section 3.1, solution in (15) is not an iterative algorithm. Therefore it costs $O(N \times L \times D \times K)$ for the forward process. The cost for solution (15) in the backward process to estimate the neural network parameters is $O(D \times K^3)$. Our second solution (19) using Eq. (20) needs to compute, at each time frame t , an inversion matrix \mathbf{J}_t^{-1} . Since inversion operations costs $O(D^3)$, which dominates the computational costs at each time frame, the cost to compute solution (19) takes $O(N \times D^3)$. Considering the solution (19) may take J iterations to converge, the total computational cost for (19) is therefore $O(J \times N \times D^3)$, which is larger than the cost for solution (15). In the forward process of solution (19), the cost to compute $\mathbf{U} \mathbf{h}_t$ is $O(J \times N \times L \times D \times K)$ which is higher than the forward-pass cost in solution (15) because solution (19) takes J iterations.

Other methods in [13] use iterative forward and backward propagation algorithms to estimate both the lower- and upper-layer weights, each costs $O(J \times N \times L \times D \times K)$ operations to converge. According to Table 1, the cost of forward process in [13] is higher than the costs for solution (15). Moreover, since L is usually larger than D and K is approximately the same as D , the cost for backward propagation in [13] is also higher than our solutions (15) and (19). It is worth mentioning that the number of iterations J in our solutions might be smaller than that used in [13] because our solutions use Gauss-Newton method, which converges faster than

Table 1

The computational costs to implement our algorithms. D is the feature dimension, L is the context length, K is the hidden layer size, N is the number of observation frames, and J is the number of iterations.

Method	Forward prop.	Backward prop.
Solution (15) in Section 3.1	$O(N \times L \times D \times K)$	$O(D \times K^3)$
Solution (19) in Section 3.2	$O(J \times N \times L \times D \times K)$	$O(J \times N \times D^3)$
Method in [13]	$O(J \times N \times L \times D \times K)$	$O(J \times N \times L \times D \times K)$

the gradient descent method used in [13]. Specifically, the solution (15) is a closed-form solution and converges in one iteration.

3.5. Other implementation considerations

Note that we have $\gamma_m(t)$, the posterior probability of Gaussian m at time t , in the above algorithm. In practice, we use Viterbi alignment at the Gaussian level, so $\gamma_m(t)$ is either 1 or 0.

For completeness, we list in Appendix A.3 auxiliary functions for fMLLR [10]. To debug our implementations, we use the scores of these auxiliary functions to compare against the scores by the proposed algorithm.

To replicate results, the randomization of \mathbf{W} starts from a fixed seed. The random values for the lower layer weights \mathbf{W} are in the range of -2.0 to 2.0 .

Inputs vectors need to be normalized to be zero mean and unit variance in each input feature dimension. This improves convergence of the neural network learning.

The step size η is important for Gauss-Newton algorithm. The step size starts from a small value, usually 1×10^{-2} . If one iteration does not increase auxiliary score, the current estimate is backed to its previous estimate and the step size η is halved for the next iteration.

For numerical issues, the inversion of Jacobian in (8) may be implemented using singular value decomposition (SVD).

4. Experiments

4.1. System descriptions

We conduct automatic speech recognition (ASR) experiments on an internal Xbox voice search data set. The scenario supports distant talking voice search (of music catalog, games, movies, etc.) using a microphone array. The training set of the speaker-independent (SI) GMM-HMM system contains 40 h of Xbox voice search commands. The evaluation was conducted on data from 25 speakers. For each speaker 200 utterances are used for adaptation and 100 utterances are used for testing. The total number of test sentences is 2500 with 10,563 words. There are no overlap among training, adaptation and test sets. We use Kaldi [20] to train a weighted finite state transducer (WFST)-based recognition system [21]. The features are 13-dimension Mel filter-bank cepstral coefficients (MFCC) with delta and delta-delta coefficients. Per-device cepstral mean subtraction is applied. The SI acoustic model has 7.5 k Gaussian components trained with the standard maximum likelihood estimation (MLE) [1] procedure. We compare the proposed approaches with the baseline system without adaptation and the system using fMLLR [10].

4.2. Effects of hidden layer and context window size

4.2.1. Hidden layer size

We first conducted experiments to investigate how the hidden layer size K affects the performance using a development set. Notice that there are methods [22,23] to automatically decide hidden layer size. The method in [22] selects the hidden layer size by reducing its regression error to a predefined threshold. The method in [23] proposes a bound of the generalization error. The method selects the hidden layer size by maximizing the coverage of unseen samples with a constraint or by thresholding on the bound. For speech recognition task, the most direct measure of error is the word error rate. However, to our knowledge, there is not yet a method to directly relate word error rate to neural network size.

We therefore empirically investigated effects of the hidden layer size to speech recognition performance. In these experiments we used the solution (15) (i.e., optimize for the transformed features) to estimate the upper layer weights \mathbf{U} . Fig. 2 summarizes the experimental results under the condition that the input dimension is set to 351 (or $L=9$ frames) and α is set to 0.6. It can be observed that we achieved the lowest WER of 25.73% when 39 hidden units were used. Additional experiments using different input layer size indicate that the lowest WERs were typically obtained when 20–40 hidden units were used. This suggests that the optimal hidden layer size is not sensitive to the input layer size. The relatively small optimal hidden layer size is attributed to the small adaptation set used.

4.2.2. Context window size

Other experiments on neural network speech recognition [2,24] suggest using 9–11 frames of context window. Notice that 9–11 frames correspond to 90–110 ms, approximately the average duration of a phone in speech signal. We therefore conducted experiments to cross validate the above recommendation using the same development set. We evaluated the impact of the context window size L to the word error rate (WER) when the hidden layer size K is fixed using the same procedure described above. Table 2 summarizes our results. From Table 2 we observe that lowest WERs can be obtained with $L=9$ context windows if the hidden layer size is large enough to take advantage of it.

Similar experiments were conducted using the solution (19) (i.e., optimize for the observed data), which was initialized from the closed-form solution (15) and was run for 10 iterations. In Table 3 WER results with context window size L set to 1, 9, and 11

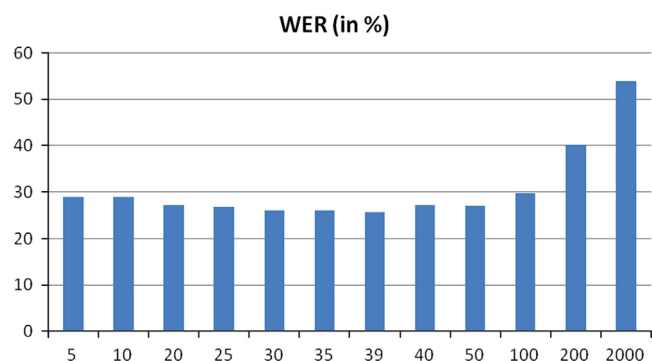


Fig. 2. The effect of hidden layer size on word error rate (WER) with solution (15). The horizontal axis is the hidden layer size.

Table 2

The effect of context window size on word error rate (WER) with solution (15).

Context window size	$K=20$	$K=39$
1	28.80	31.91
3	27.09	28.66
9	27.22	25.73
11	28.08	25.91

Table 3

Effect of context window size and hidden layer size on WERs with solution (19).

Context window size	$K=20$	$K=39$
1	28.66	31.49
9	27.06	25.68
11	28.41	25.73

and hidden layer size K set to 20 and 39 are reported. The step size η was, respectively, set to 1×10^{-2} for context window $L=1$, 5×10^{-3} for context window $L \in \{9, 11\}$ with $K=20$, and 1×10^{-3} for context window $L \in \{9, 11\}$ with $K=39$. In all these cases, using 9-frame context provides the best WER. These experiments cross validate the recommendation of using 9–11 frames for neural network speech recognition [2,24].

Remark 1. It is worthwhile to point out that most previously developed methods [4,6,5,10,13] cannot take advantage of the neighboring frames, which, indicated by Tables 2 and 3, to have positive effect. [17] also benefits from the neighboring frames but uses linear model.

Remark 2. Notice that the closed-form solution (15) converges in one iteration and has similar performances as the iterative solution (19), which takes 10 iterations. Other existing methods [13] use the gradient descent method, therefore is even slower than the Gauss–Newton method used in the iterative solution (19). We measured the real time factor (RTF) on one speaker with the setup $K=39$ and $L=9$ that has the lowest WER. The solution (15) has RTF of 0.22 and is real time. The solution (19) has RTF of 1.21. The closed-form solution (15) therefore is desirable in conditions such as fast adaptation.

4.3. Main results

Table 4 compares our proposed bias compensation approaches with the SI system and fMLLR with bias compensation [10]. In these experiments we used 9-frames of input (351 input units), 20 hidden units and 39 output units. The α was set to 0.6 for the proposed approaches. The step size η was set to 5×10^{-3} and 1×10^{-3} for hidden layer size K set to 20 and 39, respectively.

From Table 4 we observe that fMLLR with bias compensation reduced WERs from 31.52%, achieved by the SI system, to 29.49%. This reduction is statistically significant with significance level of 1.3×10^{-3} . With the hidden layer size K set to 20, using our proposed approaches we can further reduce the WER to 27.22% and 27.06%, respectively, with and without dropping the Jacobian term. Compared to the fMLLR with bias compensation, our approaches cut errors by 7.7% and 8.2%. These reductions are significant, each with a significance level of 2.5×10^{-4} and 8.7×10^{-5} .

Note that the method with closed-form solution (15) without considering the Jacobian term achieved the similar performance as that with Gauss–Newton method (19). This provides a significant gain in efficiency since the closed-form solution is much faster than the Gauss–Newton method. In fact, as shown in the table, WER can be further reduced to 25.73% using the closed-form solution (15) if 39, instead of 20, hidden units are used. This translates to a relative WER reduction of 13.0% over the fMLLR with bias compensation system and is statistically significant with significance level of 9.5×10^{-10} [25]. The method with Jacobian term further reduced the WER to 25.68%, with the significance level of 5.6×10^{-10} .

Table 4

Compare the proposed approaches with SI model and the fMLLR with bias compensation.

	WERs (%)
SI model (w/o adaptation)	31.52
fMLLR with bias compensation	29.49
Proposed approach with closed-form solution ($K=20$)	27.22
Proposed approach with Gauss–Newton method ($K=20$)	27.06
Proposed approach with closed-form solution ($K=39$)	25.73
Proposed approach with Gauss–Newton method ($K=39$)	25.68

Table 5

Compare the tandem scheme and the affine fMLLR.

	WERs (%)
Affine fMLLR	24.03
Tandem scheme with closed-form solution	22.80
Tandem scheme with Gauss–Newton method	22.68

The tandem scheme proposed in Section 3.3 applies both bias and rotation compensation. Table 5 compares the tandem scheme with the closed-form and the Gauss–Newton method with affine fMLLR. In these experiments a SHLNN with 9-frame input feature, 39 hidden units and 39 output units were used. The step size η was set to 1×10^{-3} . As shown in this table, our proposed approaches reduced WER to 22.80% and 22.68%, respectively, with and without dropping the Jacobian term, from 24.03% achieved by affine fMLLR. These results correspond to 5.1% and 5.6% relative WER reductions with significance level of 3.5×10^{-2} and 2.0×10^{-2} , respectively. Compared to the SI model, our proposed methods cut WERs significantly by more than 27% with significance level of zero.

5. Conclusions and discussions

In this paper, we have introduced a novel approach to train single-hidden-layer neural networks to reduce mismatch between training and testing for speaker adaptation of GMM–HMMs. The neural network may be considered as a case of the recently proposed extreme learning machine which uses randomly generated lower-layer weights and linear output units. To estimate upper layer weights, we have developed efficient closed-form solution and Gauss–Newton methods. Our proposed approach enables compensating for a time dependent bias and takes advantages of neighboring observation frames. On a large vocabulary speech recognition task, we show that the proposed approach can reduce word error rates by more than 27% over a speaker-independent system. Our proposed nonlinear adaptation approach, which uses a closed-form solution, is also significantly faster than prior arts [13] that use gradient-based methods such as back-propagation [26].

To speed up the adaptation process we have used randomly generated lower-layer weights similar to the ELM. However, trade-off may be made between the adaptation speed and the power of the model. For example, instead of using random values we may learn the lower-layer weights as well, possibly by exploiting the constraint imposed by the closed-form relationship between the weights of upper- and lower-layers. Alternatively we may stack one ELM on top of another [16,15] via input–output concatenation to increase the modeling power, which has been shown to improve phone recognition accuracy over ELM [27,28]. We believe that these extensions may further decrease the recognition error rate, at the cost of increased adaptation time.

Appendix A

A.1. Derivation of an efficient closed-form solution

Since Σ_m^{-1} is a diagonal matrix, $\Sigma_m^{-1} \otimes \mathbf{h}_t \mathbf{h}_t^T$ is a block diagonal matrix. We therefore have

$$\begin{aligned} & \text{vec}(\mathbf{U}) \\ &= \mathbf{T}_{K,D} \left(\begin{array}{ccc} \sum_m \gamma_m(t) \sigma_{m1}^{-1} \mathbf{h}_t \mathbf{h}_t^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_m \gamma_m(t) \sigma_{mD}^{-1} \mathbf{h}_t \mathbf{h}_t^T \end{array} \right)^{-1} \end{aligned}$$

$$\begin{aligned}
& \times \mathbf{T}_{D,K} \text{vec} \left(\frac{\partial Q}{\partial \mathbf{U}} \right) \Big|_{\mathbf{U}=\mathbf{0}} \\
& = \mathbf{T}_{K,D} \left(\begin{array}{ccc} \sum_{mt} \gamma_m(t) \sigma_{m1}^{-1} \mathbf{h}_t \mathbf{h}_t^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_{mt} \gamma_m(t) \sigma_{mD}^{-1} \mathbf{h}_t \mathbf{h}_t^T \end{array} \right)^{-1} \\
& \times \text{vec} \left(\left(\frac{\partial Q}{\partial \mathbf{U}} \right)^T \right) \Big|_{\mathbf{U}=\mathbf{0}} \quad (22)
\end{aligned}$$

Multiplying the above equation with $\mathbf{T}_{K,D}^{-1}$ on both sides, we get

$$\begin{aligned}
& \mathbf{T}_{K,D}^{-1} \text{vec}(\mathbf{U}) \\
& = \left(\begin{array}{ccc} \sum_{mt} \gamma_m(t) \sigma_{m1}^{-1} \mathbf{h}_t \mathbf{h}_t^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_{mt} \gamma_m(t) \sigma_{mD}^{-1} \mathbf{h}_t \mathbf{h}_t^T \end{array} \right)^{-1} \\
& \times \text{vec} \left(\left(\frac{\partial Q}{\partial \mathbf{U}} \right)^T \right) \Big|_{\mathbf{U}=\mathbf{0}} \\
& = \mathbf{T}_{D,K} \text{vec}(\mathbf{U}) \\
& = \text{vec}(\mathbf{U}^T) \quad (23)
\end{aligned}$$

We therefore have

$$\begin{aligned}
& \text{vec}(\mathbf{U}^T) \\
& = \left(\begin{array}{ccc} \sum_{mt} \gamma_m(t) \sigma_{m1}^{-1} \mathbf{h}_t \mathbf{h}_t^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_{mt} \gamma_m(t) \sigma_{mD}^{-1} \mathbf{h}_t \mathbf{h}_t^T \end{array} \right)^{-1} \\
& \times \text{vec} \left(\left(\frac{\partial Q}{\partial \mathbf{U}} \right)^T \right) \Big|_{\mathbf{U}=\mathbf{0}} \\
& = \left(\begin{array}{ccc} \left(\sum_{mt} \gamma_m(t) \sigma_{m1}^{-1} \mathbf{h}_t \mathbf{h}_t^T \right)^{-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \left(\sum_{mt} \gamma_m(t) \sigma_{mD}^{-1} \mathbf{h}_t \mathbf{h}_t^T \right)^{-1} \end{array} \right) \\
& \times \text{vec} \left(\left(\frac{\partial Q}{\partial \mathbf{U}} \right)^T \right) \Big|_{\mathbf{U}=\mathbf{0}} \quad (24)
\end{aligned}$$

This indicates that each row of \mathbf{U} , which is a vector of dimension K , can be updated independently from other rows as in (15).

A.2. Derivation of an efficient Gauss-Newton algorithm when considering Jacobian term

Using permutation matrix, we rewrite the right hand side of (8) as

$$\begin{aligned}
& \text{vec}(\mathbf{J}_t^{-T})^T [\alpha (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T \otimes \mathbf{I}_{D \times D}] \\
& = \text{vec}(\mathbf{J}_t^{-T})^T [\alpha \mathbf{T}_{DD} \mathbf{I}_{D \times D} \otimes (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T] \mathbf{T}_{DK} \\
& = \alpha (\mathbf{T}_{DD}^T \text{vec}(\mathbf{J}_t^{-T}))^T \mathbf{I}_{D \times D} \otimes (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T \mathbf{T}_{DK} \\
& = \alpha \text{vec}(\mathbf{J}_t^{-1})^T \mathbf{I}_{D \times D} \otimes (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T \mathbf{T}_{DK} \\
& = \alpha (\mathbf{T}_{DK}^T (\text{vec}(\mathbf{J}_t^{-1})^T \mathbf{I}_{D \times D} \otimes (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T))^T \\
& = \alpha (\mathbf{T}_{KD} (\text{vec}(\mathbf{J}_t^{-1})^T \mathbf{I}_{D \times D} \otimes (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T))^T \\
& = \alpha (\text{vec}(\mathbf{J}_t^{-1})^T \mathbf{I}_{D \times D} \otimes (\mathbf{H}_t (\mathbf{I}_{K \times K} - \mathbf{H}_t) \mathbf{W}_c)^T)^T \quad (25)
\end{aligned}$$

where \mathbf{T}_{DK} is a permutation matrix. The above is a vector of dimension $1 \times DK$.

A.3. Auxiliary scores by fMLLR

fMLLRs can be implemented either as a bias transformation or an affine transformation. In the case of bias transformation, its

auxiliary score is

$$Q = -\frac{1}{2} \sum_{mt} (\mathbf{b} - \rho_{mt})^T \Sigma_m^{-1} (\mathbf{b} - \rho_{mt}) \quad (26)$$

where \mathbf{b} is a global bias to be added to the input observation \mathbf{x}_t . In the case of affine transformation, its auxiliary score is as follows:

$$Q = \beta \log(|\mathbf{A}|) - \frac{1}{2} \sum_j (\mathbf{w}_j \mathbf{G}_j \mathbf{w}_j^T - 2 \mathbf{w}_j \mathbf{k}_j) - \frac{1}{2} \sum_{mt} \gamma_m(t) \boldsymbol{\mu}_m^T \Sigma_m^{-1} \boldsymbol{\mu}_m \quad (27)$$

where \mathbf{w}_j is the j -th row of the transformation $[\mathbf{A}; \mathbf{b}]$. \mathbf{G}_j and \mathbf{k}_j are the second and first order statistics for fMLLR estimations [10].

References

- [1] L.R. Rabiner, A tutorial on Hidden Markov Models and selected applications in speech recognition, *Proc. IEEE* 77 (February) (1989) 257–286.
- [2] G.E. Dahl, D. Yu, L. Deng, A. Acero, Context-dependent pre-trained deep neural networks for large vocabulary speech recognition, *IEEE Trans. Audio Speech Lang. Process.* 20 (1) (2012) 30–42.
- [3] X. Huang, A. Acero, H.-W. Hong, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice Hall, 2001.
- [4] M. Gales, S. Young, S.J. Young, Robust continuous speech recognition using parallel model combination, *IEEE Trans. Speech Audio Process.* 4 (1996) 352–359.
- [5] Y. Gong, A method of joint compensation of additive and convolutive distortions for speaker-independent speech recognition, *IEEE Trans. Speech Audio Process.* 13 (5) (2005) 975–983.
- [6] L. Deng, J. Droppo, A. Acero, Enhancement of log-spectra of speech using a phase-sensitive model of the acoustic environment, *IEEE Trans. Speech Audio Process.* 12 (13) (2004) 133–143.
- [7] L. Deng, A. Acero, L. Jiang, J. Droppo, X. Huang, High-performance robust speech recognition using stereo training data, in: *Proceedings of the ICASSP*, 2001, pp. 301–304.
- [8] M. Afify, X. Cui, Y. Gao, Stereo-based stochastic mapping for robust speech recognition, *IEEE Trans. Audio Speech Lang. Process.* 17 (7) (2009) 1325–1334.
- [9] A. Sankar, C.-H. Lee, A maximum-likelihood approach to stochastic matching for robust speech recognition, *IEEE Trans. Speech Audio Process.* 4 (1996) 190–202.
- [10] M.J.F. Gales, Maximum likelihood linear transformations for HMM-based speech recognition, *Comput. Speech Lang.* 12 (1998) 75–98.
- [11] K. Yao, L. Netsch, V. Viswanathan, Speaker-independent name recognition using improved compensation and acoustic modeling methods for mobile applications, in: *Proceedings of the ICASSP*, 2006, pp. 173–176.
- [12] J. Li, L. Deng, D. Yu, Y. Gong, A. Acero, High-performance HMM adaptation with joint compensation of additive and convolutive distortions via vector Taylor series, in: *ASRU*, 2007, pp. 67–70.
- [13] A.C. Surendran, C.-H. Lee, M. Rahim, Nonlinear compensation for stochastic matching, *IEEE Trans. Audio Speech Lang. Process.* 7 (6) (1999) 643–655.
- [14] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, *Neurocomputing* 70 (1) (2006) 489–501.
- [15] D. Yu, L. Deng, Accelerated parallelizable neural network learning algorithm for speech recognition, in: *INTERSPEECH*, 2011, pp. 2281–2284.
- [16] L. Deng, D. Yu, Deep convex net: a scalable architecture for speech pattern classification, in: *INTERSPEECH*, 2011, pp. 2285–2288.
- [17] J. Huang, K. Visweswariah, P. Olsen, V. Goel, Front-end feature transforms with context filtering for speaker adaptation, in: *ICASSP*, 2011, pp. 4440–4443.
- [18] P. Fackler, *Notes on Matrix Calculus*, North Carolina State University, 2005.
- [19] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Gauss-Jordan elimination*, in: *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, 2 edition, Chapter 2.1, Cambridge University Press, 1992, pp. 27–32.
- [20] D. Povey, A. Ghoshal, The Kaldi speech recognition toolkit, in: *IEEE ASRU*, 2011.
- [21] M. Mohri, F. Pereira, M. Riley, Weighted finite-state transducers in speech recognition, *Comput. Speech Lang.* 16 (1) (2002) 69–88.
- [22] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, *IEEE Trans. Neural Networks* 20 (2009) 1352–1357.
- [23] D.S. Yeung, W.W. Ng, D. Wang, E.C.C. Tsang, X.-Z. Wang, Localized generalization error model and its application to architecture selection for radial basis function neural network, *IEEE Trans. Neural Networks* 18 (2007) 1294–1305.
- [24] A.-R. Mohamed, D. Yu, L. Deng, Investigation of full-sequence training of deep belief networks for speech recognition, in: *INTERSPEECH*, 2010, pp. 2846–2849.
- [25] L. Gillick, S. Cox, Some statistical issues in the comparison of speech recognition algorithms, in: *ICASSP*, 1989, pp. 532–535.
- [26] A.E. Bryson, Y.-C. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*, Blaisdell Publishing Company or Xerox College Publishing, 1969, p. 481.
- [27] L. Deng, D. Yu, J. Platt, Scalable stacking and learning for building deep architectures, in: *ICASSP*, 2012, pp. 2133–2136.

- [28] B. Hutchinson, L. Deng, D. Yu, A deep architecture with bilinear modeling of hidden representations: applications to phonetic recognition, in: ICASSP, 2012, pp. 4805–4808.



Kaisheng Yao received his Ph.D. degree on Communication and Information Systems from Tsinghua University, China, in 2000. From 2000 to 2002, he worked as an invited researcher at Advanced Telecommunication Research Lab in Japan. From 2002 to 2004, he was a post-doc researcher at University of California at San Diego. From 2004 to 2008, he was with Texas Instruments. Since 2008, he has been with Microsoft. His research and development interests include speech, language, and image processing, machine learning, data mining, pattern recognition and signal processing. He has published 50 papers in these areas and is the inventor/coinventor of more than 20 granted/pending patents.



Dong Yu joined Microsoft Corporation in 1998 and Microsoft Speech Research Group in 2002, where he is currently a senior researcher. His research interests include speech processing, robust speech recognition, discriminative training, and machine learning. His most recent work focuses on deep learning and its applications to large vocabulary speech recognition. He has published over 100 papers in these areas and is the inventor/coinventor of more than 40 granted/pending patents.

He is a senior member of IEEE. He is currently serving as a member of the IEEE Speech and Language Processing Technical Committee (2013) and an associate

Q3 editor of IEEE transactions on audio, speech, and language processing (2011). He has served as an associate editor of IEEE signal processing magazine (2008–2011) and the lead guest editor of IEEE transactions on audio, speech, and language processing – special issue on deep learning for speech and language processing (2010–2011).

Q4



Li Deng received the Bachelor degree from the University of Science and Technology of China, and received the Master and Ph.D. degrees from the University of Wisconsin, Madison. He joined the Department of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada in 1989 as an Assistant Professor, where he became a Full Professor with tenure in 1996. In 1999, he joined Microsoft Research, Redmond, WA as a Senior Researcher, where he is currently a Principal Researcher. Since 2000, he has also been an Affiliate Full Professor and graduate committee member in the Department of Electrical Engineering at University of Washington, Seattle. Prior

to MSR, he also worked or taught at Massachusetts Institute of Technology, ATR Interpreting Telecom. Research Lab. (Kyoto, Japan), and HKUST. His current (and past) research activities include deep learning and machine intelligence, deep neural networks for speech and related information processing, automatic speech and speaker recognition, spoken language identification and understanding, speech-to-speech translation, machine translation, language modeling, information retrieval, neural information processing, dynamic systems, machine learning and optimization, parallel and distributed computing, graphical models, audio and acoustic signal processing, image analysis and recognition, compressive sensing, statistical signal processing, digital communication, human speech production and perception, acoustic phonetics, auditory speech processing, auditory physiology and modeling, noise robust speech processing, speech synthesis and enhancement, multimedia signal processing, and multimodal human-computer interaction. In these areas, he has published over 300 refereed papers in leading journals and conferences and 3 books, and has given keynotes, tutorials, and distinguished lectures worldwide. His recent technical work (since 2009) on industry-scale deep learning with colleagues and collaborators and his leadership in this emerging area have created significant impact on speech recognition, signal processing, and related applications with high practical value.

He has been granted over 60 US or international patents in acoustics/audio, speech/language technology, machine learning (deep learning), and related fields. He received numerous awards/honors bestowed by IEEE, ISCA, ASA, Microsoft, and other organizations. He is a Fellow of the Acoustical Society of America, a Fellow of the IEEE, and a Fellow of ISCA. He served on the Board of Governors of the IEEE Signal Processing Society (2008–2010). More recently, he served as Editor-in-Chief for the IEEE Signal Processing Magazine (2009–2011), which, according to the Thomson Reuters Journal Citation Report released on June 2010 and 2011, ranks first in both years among all IEEE publications (127 in total) and all publications within the Electrical and Electronics Engineering Category worldwide (247 in total) in terms of its impact factor: 4.9 and 6.0, and for which he received the 2011 IEEE SPS Meritorious Service Award. He currently serves as an Editor-in-Chief for the IEEE Transactions on Audio, Speech, and Language Processing 2012–2014.