Bottleneck ANN: dealing with small amount of data in shift-MLLR adaptation

Zbyněk Zajíc, Lukáš Machlica, Luděk Müller
Department of Cybernetics, Faculty of Applied Sciences
University of West Bohemia, Plzeň, Czech Republic
Email: {zzajic, machlica, muller}@kky.zcu.cz

Abstract—**The aim of this work is to propose a refinement of the shift-MLLR (shift Maximum Likelihood Linear Regression) adaptation of an acoustics model in the case of limited amount of adaptation data, which can lead to ill-conditioned transformations matrices. We try to suppress the influence of badly estimated transformation parameters utilizing the bottleneck Artificial Neural Network (ANN). The ill-conditioned shift-MLLR transformation is propagated through a bottleneck ANN (suitably trained beforehand), and the output of the net is used as the new refined transformation. To train the ANN the well and the badly conditioned shift-MLLR transformations are used as outputs and inputs of ANN, respectively.**

Keywords: ASR, Adaptation, shift-MLLR, ANN, bottleneck

I. Introduction

A speaker adaptation of an acoustic model in the task of the Automatic Speech Recognition (ASR) is a standard approach how to improve the performance of the speech recognition. The most used adaptation techniques are methods based on a linear transformation, where the number of free parameters to be estimated significantly decreases via clustering of similar parameters in comparison with other adaptation methods. However, the number of free parameters of the transformation is still high to be estimated properly when dealing with extremely small data sets.

Various solutions to avoid this problem have been proposed, e.g. lowering the number of free parameters by using diagonal or block diagonal transformation matrices (1) or finding transformation matrices as a linear combination of basis matrices (2). Another solution is performing a proper initialization of transformation matrices (3), (4). In this work we try to incorporate an Artificial Neural Network (ANN) in order to refine the poor estimates of the shift-MLLR transformation. The adaptation approach is described in Section II.

Our idea is based on the principle of information reduction. The ill-conditioned adaptation matrix estimated on a dataset containing small amount of adaptation data is propagated through a bottleneck neural network (a special type of ANN used for dimensionality reduction, see the Section III), hence it is transformed in a non-linear fashion in order to map it to a robust estimate (reduce the influence of bad estimates of parameters, see the Section III-A). To train the bottleneck ANN at first shift-MLLR transformations are estimated on sufficiently large datasets, and subsequently these transformations are used as inputs and outputs of the bottleneck ANN (see the Section III-B). Tests and their results are given in Section IV, they are performed on SpeechDat-East Corpus.

II. Adaptation based on linear transformation

These adaptation techniques adjust the Speaker Independent (SI) model so that the probability of adaptation data would be maximized. Let the SI model be represented by a Hidden Markov Model (HMM) with output probabilities described by Gaussian Mixture Models (GMMs) with mean $\boldsymbol{\mu}_{jm}$, covariance matrix $\boldsymbol{C}_{jm}$ and weight ω_{jm} of each mixture component $m = 1,\dots,M$, and of each state $j = 1,\dots,J$ of HMM. The m^{th} mixture component's posterior of the j^{th} state of the HMM given an acoustics feature vector $\boldsymbol{o}_t$ is

$$\gamma_{jm}(t) = \frac{\omega_{jm} p(\boldsymbol{o}_t|jm)}{\sum_{m=1}^{M} \omega_{jm} p(\boldsymbol{o}_t|jm)}. \tag{1}$$

One of the most popular adaptation techniques in cases of small amount of adaptation data are methods based on Linear Transformations (LT), e.g. the Maximum Likelihood Linear Regression (MLLR) (1), which transform the mean of output probabilities of HMM $\boldsymbol{\mu}_{jm}$ by an affine transformation

$$\bar{\boldsymbol{\mu}}_{jm} = \boldsymbol{A}_{(n)}\boldsymbol{\mu}_{jm} + \boldsymbol{b}_{(n)}, \tag{2}$$

where $\boldsymbol{W}_{(n)} = [\boldsymbol{A}_{(n)}, \boldsymbol{b}_{(n)}]$ is the transformation matrix composed from the matrix $\boldsymbol{A}_{(n)}$ and the bias $\boldsymbol{b}_{(n)}$, and $\bar{\boldsymbol{\mu}}_{jm}$ is the adapted mean. The advantage of MLLR is that it can adapt more HMM components at once (in this work only means are adapted) using the same transformation matrix. For this purpose similar model components are clustered into clusters $K_n, n = 1,\dots,N$, hence the number of parameters to be estimated decreases. Denoting d the dimension of acoustics features, the number of free parameters for adaptation is $D = N \times (d^2 + d)$. The clusters can be obtained using a regression tree (5). It is a binary tree where the root (first) node contains all the means of all the GMM mixture components contained in the HMM. The splitting of nodes is based upon the euclidean distance of GMM means belonging to a given node – two new child nodes are formed so that the distance of centroids computed from GMM means in these nodes is maximized. Once a tree is constructed and adaptation data are avilable an occupation of each node $\upsilon_k = \sum_{t=1}^{T} \sum_m \gamma_{jm}(t)$ in the

978-1-4673-2197-6/12/$31.00 ©2012 IEEE

tree is computed. The index m ranges over all the indexes of GMM means contained in the node k, and T is the number of feature vectors used for adaptation. A threshold θ_{th} has to be specified, the transformation matrix $\boldsymbol{W}$ is computed only for the deepest nodes for which the condition $\upsilon_k > \theta_{th}$ still holds. Thus, means in such a node will be transformed by the same transformation matrix, for detail see (5). Note that the number of clusters N depends on the amount of given adaptation data and on the value of the threshold θ_{th}. If θ_{th} is high and only a low amount of adaptation data is available than only one (global) transformation will be used (same transformation matrix for all the GMM means). Hence, N determines the depth of the regression tree and it is the upper bound on the number of clusters that could be acquired if enough data would be available.

Another popular method based on linear transformations is called shift-MLLR (6), where only the bias vector $\boldsymbol{b}_{(n)}$ is utilized (the matrix $\boldsymbol{A}_{(n)}$ is assumed to be the identity matrix)

$$\bar{\boldsymbol{\mu}}_{jm} = \boldsymbol{\mu}_{jm} + \boldsymbol{b}_{(n)}. \qquad (3)$$

Thus, the number of free parameters further significantly decreases ($D = N{\cdot}d$). To estimate the bias $\boldsymbol{b}_{(n)}$ Maximum Likelihood (ML) criterion is used. The auxiliary function, which is maximized has the form

$$\begin{aligned} Q(\lambda,\bar{\lambda}) = -\frac{1}{2}\sum_{t=1}^{T}\sum_{jm\in K_n}\gamma_{jm}(t)(\log|\boldsymbol{C}_{jm}|+ \\ +(\boldsymbol{o}(t)-\bar{\boldsymbol{\mu}}_{jm})^T\boldsymbol{C}_{jm}^{-1}(\boldsymbol{o}(t)-\bar{\boldsymbol{\mu}}_{jm})). \end{aligned} \qquad (4)$$

The bias $\boldsymbol{b}_{(n)}$ is then given by

$$\boldsymbol{b}_{(n)} = \boldsymbol{S}\sum_{t=1}^{T}\sum_{jm\in K_n}\gamma_{jm}(t)[\boldsymbol{C}_{jm}^{-1}(\boldsymbol{o}(t)-\boldsymbol{\mu}_{jm})], \qquad (5)$$

where

$$\boldsymbol{S} = (\sum_{t=1}^{T}\sum_{jm\in K_n}\gamma_{jm}(t)\boldsymbol{C}_{jm}^{-1})^{-1}. \qquad (6)$$

In the case, where the number of clusters N is set using a regression tree (5), it is possible to choose smaller threshold θ_{th} for shift-MLLR than for the MLLR adaptation. The reason is that the number of free parameters for shift-MLLR is much lower since the matrix $\boldsymbol{A}$ has not to be estimated.

III. Information reduction via bottleneck

Bottleneck ANN is frequently used in order to reduce the dimensionality of feature vectors (7). The bottleneck strategy consists in distributing the D dimensional input data through a hidden layer with number of neurons smaller than the dimension. However, the number of neurons in the output layer is same as the dimension of input vectors, for details see Figure 1.

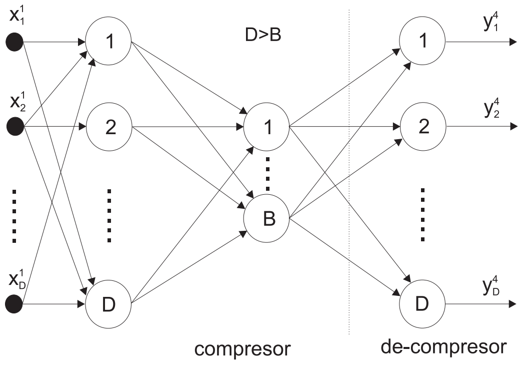

Fig. 1. Bottleneck ANN

The training of bottleneck ANN is supervised, the output for each input is supplied by a teacher. The task of the hidden layer with B neurons is to reduce the dimension of the input space to a smaller dimension B, while requesting the best possible match with the output vector, for details see (8). The principle of the bottleneck can be divided into two parts: compression and decompression of information. The ANN is then used for the task of data compression. In the task of speech recognition, the bottlenecks were presented e.g. in the work (9) or (10).

A. Refinement of adaptation

Our goal is not to find the low dimensional representation of the input data (this is the case of the dimensionality reduction task), but to filter out the useless and inaccurate information contained in the adaptation matrices. Estimating the adaptation matrices with small amount of adaptation data (the insufficient information for the estimation of all free adaptation parameters) can lead to the improperly adjusted adaptation parameters. These parameters contained the bad information for speaker adaptation and can cause the poor recognition rates.

The input to the ANN is a vector, not a matrix. Hence in order to cope with matrices (e.g. with MLLR transformation matrix $\boldsymbol{W} = [\boldsymbol{A}, \boldsymbol{b}]$) a vector $\boldsymbol{w} = \mathrm{vec}(\boldsymbol{W})$ has to be formed, where the operator vec concatenates the rows of a matrix so that a high dimensional vector – supervector – is formed. Unfortunately, since the matrix $\boldsymbol{W}$ has to be decomposed to a vector, treated like a vector and again reassembled to a matrix, the properties of the linear space generated by the matrix $\boldsymbol{W}$ can be spoiled in a large extent and the matrix multiplication $\boldsymbol{A}_{(n)}\boldsymbol{\mu}_{jm} + \boldsymbol{b}_{(n)}$ in (2) can lead to poor recognition.

The same problem is solved in method (2), where the new adaptation matrix is constructed as a linear combination of some basis matrices $\boldsymbol{w_e} = \mathrm{vec}(\boldsymbol{W_e})$, but the final matrix $\boldsymbol{W}_{out}$ is found using the ML criterion.

To avoid this problem, we turn to the shift-MLLR, where only bias vectors $\boldsymbol{b}$ have to be processed. These are then used as the input to the ANN.

B. Bottleneck for shift-MLLR

The proposed approach of the shift-MLLR refinement has the following steps:

- **Format of data**: $\boldsymbol{w}_s = [\boldsymbol{b}^{\mathrm{T}}_{s(1)}, \dots, \boldsymbol{b}^{\mathrm{T}}_{s(N)}]^{\mathrm{T}}$ is the s^{th} speaker's input vector – all the transformation vectors $\boldsymbol{b}_{s(n)}, n = 1, \dots, N$ from all clusters K_n are concatenated into one supervector. The number of clusters N is fixed. Dimension of the supervector is $D = N \cdot d$.
- **Training**: Input supervectors $\boldsymbol{w}^{\mathrm{train}}_s$ from the training dataset are estimated for each speaker s utilizing MLLR-shift adaptation using only one sentence from this given speaker (to simulate a limited amount of adaptation data). Output supervectors $\boldsymbol{w}^{train-out}_s$ (the information from the teacher - supervised training) are composed from transformations estimated on all available training data from speaker s. Finally, pairs $(\boldsymbol{w}^{\mathrm{train}}_s, \boldsymbol{w}^{train-out}_s), s = 1, \dots, S$ are continuously introduced one-by-one to the bottleneck ANN in order to train ANN paramaters. Hence, a non-linear transformation is trained, the ANN learns the relation between ill- and well-conditioned estimates of the shift-MLLR biases. The bottleneck should remove the inconsistency between input and output.
- **Testing**: After a speaker model was adapted, the supervector $\boldsymbol{w}^{\text{test}}$ is constructed. This supervector $\boldsymbol{w}^{\text{test}}$ is distributed through the bottleneck ANN so that the output supervector $\boldsymbol{w}^{\text{test-out}}$ is obtained. The output supervector $\boldsymbol{w}^{\text{test-out}} = [\boldsymbol{b}^{\text{test-out}}_{(1)}, \dots, \boldsymbol{b}^{\text{test-out}}_{(N)}]$ (refined transformation) is than decomposed and utilized to adapt the speaker independent model (the previous adaptation is left out).

IV. Experiments

A. SpeechDat-East (SD-E) Corpus

For experiment purposes we used the Czech part of the SpeechDat-East corpus (see (12)). In order to extract the features Mel-frequency cepstral coefficients (MFCCs) were utilized, 11 dimensional feature vectors were extracted each 10 ms utilizing a 32 ms hamming window, Cepstral Mean Normalization (CMN) was applied, and Δ, Δ^2 coefficients were added.

A 3 state HMM based on triphones with 2105 states total and 8 GMM mixture components with diagonal covariances in each of the states was trained on 700 speakers with 50 sentences for each speaker (cca 4 sec. on a sentence).

To test the systems performance different 200 speakers from SD-E were used with 50 sentences for each speaker, however a maximum of 12 sentences was used for the adaptation. A language model based on trigrams was used for the recognition process (13). The vocabulary consisted of 7000 words.

B. Adaptation Setup

In our experiments we used unsupervised shift-MLLR adaptation with one global transformation $\boldsymbol{b}$ for each speaker and with 64 transformations $\boldsymbol{b}_{(n)}, n = 1, \dots, 64$ depending on classes in the regression tree. The bottleneck ANN was trained with IRPROP training algorithm (11) and ANN with 3 layers was used. For the global and shift-MLLR adaptation with 64 transformations the number of neurons in each layer of ANN was 33, 10, 33 and 2112, 100, 2112, respectively. The topology of ANN can be seen in Figure 1, in hidden layers the sigmoid activation function was utilized and linear function was used in the output layer. In the training of ANN we utilized 700 speakers (used in the training phase of HMM) from the SD-E corpus. Input vectors – shift-MLLR biases – used in the training phase of ANN were estimated on one and two adaptation sentences from all the 700 speakers to prepare for cases of small amount of adaptation data. Next, for each speaker 20 input vectors were collected based on different sentences. Each output vector (shift-MLLR bias) was estimated on all available data from one speaker (cca 50 sentences). Note that to all the 20 input vectors of one speaker the same output vector was assigned. The task of ANN is to find the relation between poorly and well estimated transformations.

In the testing phase at first shift-MLLR adaptation was performed, the supervector was formed and propagated through ANN, the output vector was decomposed to individual biases – refined transformation. Finally, the acoustic SI model was adapted by the refined transformation.

C. Results

The proposed method was tested on varying number of adaptation sentences. Figure 2 depicts results (Accuracy (Acc) in %) of ASR using shift-MLLR adaptation with global transformation and with 64 transformations (denoted as shift-MLLR-global and shift-MLLR-64, respectively), and results of refined shift-MLLR adaptation utilizing bottleneck ANN (denoted as ANN-shift-MLLR-global and ANN-shift-MLLR-64, respectively). Also accuracies of MLLR adaptation assuming only global transformation (MLLR-global) and the unadapted SI model (baseline) are presented. The selected results are shown in Table I, the accuracy of SI model is 68.75%. Note that we chose a global transformation for MLLR, because of insuffiecient amount of adaptation data.

No.Sentences	MLLR-global	shiftMLLR-64	ANN-shiftMLLR-64
1	14.04	69.84	70.81
2	56.36	70.56	71.19
3	66.74	70.79	71.17
4	69.58	70.94	71.19
5	70.23	71.16	71.21
6	70.74	71.31	71.53
8	72.30	71.76	71.15
10	72.30	72.14	71.26
12	72.33	71.54	71.25

TABLE I
The results (Acc)[%] of speech recognition utilizing unadapted SI model, MLLR adaptation, shift-MLLR adaptation, and refined ANN-shift-MLLR adaptation for different number of adaptation sentences.

As we expected ordinary MLLR with only a global transformation matrix $\boldsymbol{W} = [\boldsymbol{A}, \boldsymbol{b}]$ completely failed in the case of small amount of adaptation data (1-4 sentences, which is approx. 4-16s of speech without reference transcription). On

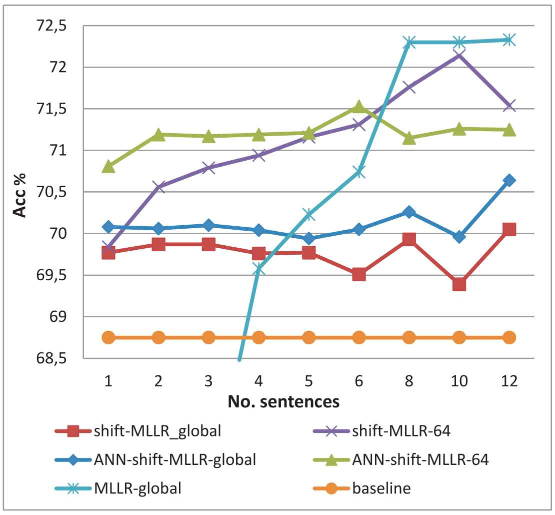

Fig. 2. Accuracy (Acc)[%] of speech recognition utilizing unadapted SI model and different types of adaptation dependence on the number of adaptation sentences.

the other hand the shift-MLLR approach with only a bias vector used as a transformation ($\boldsymbol{W} = [\boldsymbol{b}]$) seems to be a good choice for the adaptation with a few adaptation data. For all different amounts of adaptation data the performance of the recognition system is not spoiled, as was the case for MLLR adaptation. The proposed approach – shift-MLLR refined by ANN (ANN-shift-MLLR) – improves the accuracy for small amounts of adaptation data (1-4 sentences) in comparison to shift-MLLR, however no additional improvement is acquired when more adaptation data are available. The reason is that ANN was tuned for cases with small amounts of adaptation data (1-2 sentences).

V. Conclusions

Presented experiments proved the increase of the accuracy of the speech recognition utilizing MLLR based adaptation, especially the involvement of the shift-MLLR refined by ANN in the task of extremely small data sets. The bottleneck obviously suppresses the influence of the ill-conditioned parameters of the adaptation. In future work we would like to extend proposed methods based on ANN also to MLLR with full transformation matrix.

VI. Acknowledgments

This research was supported by the Technology Agency of the Czech Republic, project No. TA01030476 and by the grant of the University of West Bohemia, project No. SGS-2010-054.

References

M.J.F. Gales, “Maximum likelihood linear transformations for HMM-based speech recognition”, Computer Speech and Language, vol. 12, pp. 75–98, 1997.

D. Povey, and K. Yao, “A Basis Representation of Constrained MLLR Transforms for Robust Adaptation”, Computer Speech & Language, vol. 26, pp. 35–51, 2012.

Y. Li, H. Erdogan, T. Gao, and E. Marcheret, “Incremental on-line feature space MLLR adaptation for telephony speech recognition”, 7th International Conference on Spoken Language Processing, pp. 1417–1420, 2002.

Z. Zajíc, L. Machlica, and L. Müller, “Initialization of fMLLR with Sufficient Statistics from Similar Speakers”, Lecture Notes in Computer Science, vol. 6836, pp. 187–194, 2011.

M. J. F. Gales, “The Generation and use of Regression class Trees for MLLR Adaptation”, Techreport Cambridge University Engineering Department, 1996.

D. Giuliani, and F. Brugnara, “Acoustic model adaptation with multiple supervisions”, TC-STAR Workshop on Speech-to-Speech Translation, pp. 151-154, 2006.

E. Parviainen, “Dimension Reduction for Regression with Bottleneck Neural Networks”, Lecture Notes in Computer Science, vol. 6283, pp. 37–44 , 2010.

Ch. M. Bishop, “Neural Networks for Pattern Recognition”, Oxford University Press, USA, 1996.

F. Grézl, M. Karafiát, and L. Burget, “Investigation into bottle-neck features for meeting speech recognition”, Interspeech, vol. 9, pp. 2947-2950 , 2009.

J. Zelinka, J. Trmal, and L. Müller, “Low-dimensional Space Transforms of Posteriors in Speech Recognition”, Interspeech, vol. 2010, pp. 1193–1196 , 2010.

Ch. Igel, and M. Hsken, “Improving the Rprop Learning Algorithm”, Second International Symposium on Neural Computation, pp. 115–121 , 2000.

P. Pollak, et al., “SpeechDat(E) - Eastern European Telephone Speech Databases”, XLDB - Very Large Telephone Speech Databases (ELRA), 2000.

A. Pražák, J. Psutka, J. Hoidekr, et al., “Automatic online subtitling of the Czech parliament meetings”, Lecture Notes in Artificial Intelligence, vol. 4188, pp. 501–508, 2006.