A Feature Space Transformation Method for Personalization using Generalized I-Vector Clustering

Kaisheng Yao, Yifan Gong, and Chaojun Liu
Microsoft Corporation, Redmond, 98052, WA, USA
{kaisheng.yao, ygong, chaojunl}@microsoft.com

Abstract
We present a feature space transformation method for personalization. This method includes a generalization of i-vector based clustering that allows parameter tying of sub-loading matrices. This method trains i-vector parameters from the utterances of a mobile device, uncovering a low dimension space for clustering variability given the device. We show through empirical results impact of parameters of the generalized i-vector method. We conducted recognition experiments on an internal large vocabulary voice search system for gaming. The method achieved significant reductions of word error rates by 28%, compared to a per utterance adaptation system.

Index Terms: speech recognition, personalization, adaptation, i-vector

1. Introduction
Automatic speech recognition (ASR) is an important component for latest mobile applications for example web search, speech understanding, and games. In these applications, a mobile device is used by a single user in a particular environment. However, it might be used by other people, for example other family members. It could also be used in different acoustic environments, for example cars, streets or family rooms, where large acoustic variations exist.

ASR is known to be sensitive to mismatches of training and testing. Personalization is an idea to reduce such mismatch by adapting the ASR associated to a device to a particular user. Usually, personalization uses methods such as speaker adaptation, feature transformation, language model adaptation, etc. An effective method in a personalized device needs to be robust to the speaker and environment variability.

Recently, i-vector based method [1] has been proposed for acoustic model training [2, 3] and speaker adaptation [4]. In the i-vector framework, a Gaussian mixture model (GMM) is trained with $K$ Gaussian components. Denote the $k$-th Gaussian component mean vector as $\mu_k(0) \in \mathbb{R}^D$. Given observation sequence $x_t, t = 1, \ldots, T$ of an utterance $s$, mean vectors of this GMM are adapted. Using a super-vector $\mu(s) = [\mu_1(s)^T \cdots \mu_K(s)^T]^T$ to represent a concatenation of these mean vectors, we relate this super-vector to the super-vector of the original GMM mean vectors, $\mu(0) = [\mu_1(0)^T \cdots \mu_K(0)^T]^T$ as follows:

$$\mu(s) = \mu(0) + Tw(s) \tag{1}$$

where $T \in \mathbb{R}^{KD \times d}$ is a loading matrix. $w(s) \in \mathbb{R}^d$ is the i-vector of this utterance and it has a priori distribution of $\mathcal{N}(:, 0, I)$. $d$ is the i-vector dimension.

This paper introduces a new method for personalization using i-vector based feature space transformation. We first develop a generalization of i-vector extraction that includes tying of sub-loading matrices and residual noise. The generalization allows us to study impacts of i-vector parameters on likelihood improvements and clustering. It may also lead to small size of i-vector parameters because of tying. Another novelty of the new method is a way that i-vectors are trained. Different from other i-vector based methods [2, 3, 4], the new method trains i-vector parameters only on utterances from a device. This produces small number of i-vector parameters. Importantly, the extracted i-vectors only represent environment and speaker variability in a particular device.

This paper is organized as follows: we describe feature space transform for personalization in section 2. Section 3 presents the new i-vector training and extraction method. Section 4.1 shows impact of parameters on log-likelihood scores and clustering. Recognition results from a large vocabulary ASR experiment are in Sec. 4.2.

2. Feature space transformation for personalization
We assume that environment and speaker variability can be compensated via constrained maximum likelihood linear regression (CMLLR) feature transform $[5, 6]$

$$\hat{x}_t = Ax_t + b = W \xi_t \tag{2}$$

where $W = [b; A]$. $b$ is a bias vector and $A$ is a rotation/scaling matrix. $\xi_t = [x_t^T]^T$ is the augmented observation vector at time $t$.

However, for personalization, it is reasonable to assume that each utterance has a particular environment and
speaker variability. The above transform could be cluster-dependent. The cluster-dependent transform $W^{(c)}$ is estimated by maximizing a cluster-dependent auxiliary score

$$Q(W^{(c)}) = \beta^{(c)} \log |A^{(c)}| + \text{tr}(K^{(c)}T W^{(c)}) - \frac{1}{2} \sum_i r_i^{(c)}G_i^{(c)}r_i^{(c)}$$

where the column vector $r_i^{(c)}$ corresponds to the $i$-th row of $W^{(c)}$. $\beta^{(c)}$, $K^{(c)}$, and $G_i^{(c)}$ are respectively cluster-dependent count, first order and second order statistics for CMLLR [5].

We use the following method in Sec. 3 to cluster utterances originated from a device.

3. The new clustering method

The new method is a generalization of previous methods [1, 3, 2], in which we introduce parameter tying and derive formulae in consideration of residual noise. Details are described below.

We assume that $x_t$ in a training set follows a distribution that is modeled by a universal background model (UBM). This UBM is represented as a GMM $\mathcal{U} = \{c_kN(\mu_k(0), \Sigma_k(0)) : k = 1, \ldots, K\}$, where $c_k$, $\mu_k(0)$ and $\Sigma_k(0)$ each denote mixture weight, mean vector, and (diagonal) covariance matrix of component $k$. In the context of personalization, we differ from previous methods [2, 3, 4] in that we train UBM only on data from a device in interest. Now we assume that each utterance on a new set has its own distribution as

$$x_t(s) \sim \sum_k c_k N(\beta(t), \Sigma_k(0))$$

That is, observations of the $s$-th utterance have their own mean vectors, though they share the same weights and covariance matrices as those in the UBM $\mathcal{U}$. The mean vector of a component $k$, $\mu_k(s)$ is adapted from its original mean vector $\mu_k(0)$. They are related as follows:

$$\mu_k(s) = \mu_k(0) + T_k w(s) + \epsilon_k(s)$$

where $T_k \in R^{D \times d}$ is the $k$-th sub-loading matrix for Gaussian component $k$. $w(s)$, as described in (1) in Sec. 1, is the i-vector of utterance $s$. The i-vector dimension $D$ is usually smaller than observation feature dimension $d$. The residual noise term $\epsilon_k(s) \in R^D \sim N(\cdot; 0, \Psi_k)$ represents modeling error. $\Psi_k$ is a $D \times D$ diagonal matrix.

Introducing the sub-loading matrix $T_k$ and the residual noise term $\epsilon_k(s)$ allows us to make a generalization as follows: Define $r_k(s)$ as a regression class index of sub-loading matrices $\{T_k : k = 1, \ldots, K\}$. We denote the total number of regression classes as $M$. We then have sub-loading matrices tying as follows

$$\mu_k(s) = \mu_k(0) + T_{r_k(s)} w(s) + \epsilon_k(s)$$

where $T_{r_k(s)} \in R^{D \times d}$ now is a tied sub-loading matrix that is shared with other Gaussian components belonging to the same regression class $r_k$. Notice that $\epsilon_k(s)$ is kept from tying.

The particular tying of sub-loading matrices can be intuitive. For example, a group of Gaussian components may represent silence and the other may represent speech. Therefore, we may choose two regression classes for noise and speech.

The general procedure for training UBM and GMM for personalization is in Algorithm 1. At a high level, we apply UBM to obtain i-vectors of training utterances from a device and then train a GMM on these i-vectors. At test time, we first estimate i-vector of an utterance from the device using its UBM and then assign the utterance to the cluster with the closest centroid in the GMM.

**Input**: training set of a device
**Output**: UBM $\mathcal{U}$ on observations and GMM $\mathcal{G}$ on i-vectors

Train UBM $\mathcal{U}$ on observations from this device

```
foreach utterance $s$ in training do
  estimate $w(s)$ and $\epsilon(s)$ as described in Sec 3.1
end
```

Estimate hyperparameters as described in Sec 3.2
Iterate the above two steps until convergence
Train GMM $\mathcal{G}$ on i-vectors $w(s)$ of the training utterances

**Algorithm 1**: Train UBM $\mathcal{U}$ and GMM $\mathcal{G}$ on a device

3.1. I-Vector and residual noise estimation

To estimate i-vector and residual noise for an utterance $s$, we define the following function to be maximized

$$Q(w(s)) \propto -\frac{1}{2} \sum_k \epsilon_k(s)^T \epsilon_k(s) - \frac{1}{2} w(s)^T \psi_k^{-1} w(s)$$

where $\psi_k^{-1} = \frac{1}{2} \sum_k \epsilon_k(s)^T \epsilon_k(s)$

$$\beta_k(s) = \sum_t \gamma_k(t) \epsilon_k(s) \in R^d$$

where $\beta_k(s) = \sum_t \gamma_k(t) \epsilon_k(s)$ is the $k$-th sub-vector of $\epsilon(s)$. $p_k(s)$ and $q_k(s)$ each are defined as follows

$$p_k(s) = \sum_t \gamma_k(t) \epsilon_k(0)$$

$$q_k(s) = \sum_t \gamma_k(t) \epsilon_k(s)$$
Differentiating Eq. (7) w.r.t. $\epsilon_k(s)$ and equating the result to zero, we have

$$\epsilon_k(s) = \Phi_k^{-1} \Sigma_k^{-1} (p_k(s) - \beta_k(s) \epsilon_k(s))$$  \hspace{1cm} (10)

where $\Phi_k = \Psi_k^{-1} + \beta_k(s) \Sigma_k^{-1}$.

We substitute the estimated $\epsilon_k(s)$ to (7) and differentiate it w.r.t. $w(s)$ to obtain its estimate

$$w(s) = \Xi^{-1} \sum_k T_{r(k)}^T \Sigma_k^{-1} (p_k(s) - \beta_k(s) \epsilon_k(s))$$  \hspace{1cm} (11)

where $\Xi = I + \sum_k \beta_k(s) T_{r(k)}^T \Sigma_k^{-1} T_{r(k)}$.

This process is iterated between estimating $\epsilon(s)$ and $w(s)$.

### 3.2. Hyperparameter estimation

To estimate hyperparameters $T$ and $\Psi$, we define the following function to be maximized

$$Q(T, \Psi) = -\frac{1}{2} \sum_{s,k,t} \gamma_k^{(s)}(t)(x_t(s) - \mu_k(s))^T \Sigma_k^{-1} (x_t(s) - \mu_k(s))$$

$$-\frac{1}{2} w(s)^T w(s) - \frac{1}{2} \epsilon(s)^T \epsilon(s) - S \log |\Psi|$$  \hspace{1cm} (12)

where $S$ is the number of training utterances.

We solve the following equation to estimate $T_{r(k)}$ for each regression class $l$

$$\sum_a \sum_{r(k)=l} \beta_k(s) w(s)^T T_{r(k)} = \sum_a w(s) \sum_{r(k)=l} (p_k(s) - \beta_k(s) \epsilon_k(s))^T$$  \hspace{1cm} (13)

Differentiating (12) w.r.t. $\Psi$, we have the following estimate

$$\Psi = \frac{1}{S} \sum_s \epsilon(s) \epsilon(s)^T$$  \hspace{1cm} (14)

Since we assume that residual noise is uncorrelated, we keep only the diagonal elements of the above estimate.

We repeat process of i-vector estimation in Sec. 3.1 and hyperparameter estimation in Sec. 3.2 until convergence.

Notice that $U$ and $G$ are dominated by the loading matrix with its size of $O(K \times D \times d)$. Tying can reduce its size to $O(M \times D \times d)$ and the total number of regression classes $M$ is usually smaller than $K$.

Note that the standard i-vector representation in Eq. (1) is a special case of (6) that has unique regression classes for each sub-loading matrix and ignores residual noise.

### 4. Experiments

#### 4.1. Effects on log-likelihood scores and clustering performances

We first studied impact of i-vector parameters on log-likelihood scores. The i-vector parameters were trained on a single speaker with 70 utterances. Features were 39 dimension MFCCs, delta, and delta-delta. UBM $U$ had 4 Gaussian components. I-vector dimension was 10. In all of the experiments, initial value for elements of $\Psi$ was set to $10^{-5}$. Cosine distance was used for clustering.

Three setups were tested. The standard i-vector in Eq. (1) was used first, denoted as i+l in Figure 1. We then included the estimate of residual noise $\epsilon_k(s)$ and its parameter $\Psi_k$, denoted as i+l+e. We further included sub-loading matrices tying to have 2 tied sub-loading matrices, denoted as i+2l+e.

From Fig 1, we observed that introducing residual noise does not change much of the log-likelihood. On the other hand, introducing tying, as expected, reduced log-likelihood. In this case, 4 Gaussians in UBM were split into 2 classes, each with 2 Gaussians. Their log-likelihood scores versus iteration numbers are listed in Table 1 for details.

We trained 4 Gaussian components GMM on i-vectors for the three setups and observed no differences in their clustering results.

![Figure 1: Log-likelihood scores versus iteration numbers. Details listed in Table 1.](image)

<table>
<thead>
<tr>
<th>setup</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>i+l</td>
<td>2.58</td>
<td>2.83</td>
<td>2.88</td>
<td>2.90</td>
<td>2.91</td>
<td></td>
</tr>
<tr>
<td>i+l+e</td>
<td>2.58</td>
<td>2.83</td>
<td>2.88</td>
<td>2.90</td>
<td>2.91</td>
<td></td>
</tr>
<tr>
<td>i+2l+e</td>
<td>2.19</td>
<td>2.41</td>
<td>2.49</td>
<td>2.52</td>
<td>2.53</td>
<td></td>
</tr>
</tbody>
</table>

#### 4.2. Recognition results

We evaluated effectiveness of the method on an internal data set. The SR task was voice search using Xbox.
kinect device. The scenario supports distant talking voice search (of music catalog, games, movies, etc.) using a microphone array. The data has 20 hours of Xbox voice search commands for training and 2 hours for testing. Each device in the training and testing had 70 utterances. We used Kaldi [7] to train ASR system. The features were 39-dimention MFCC, delta, and delta-delta coefficients. Per-device cepstral mean subtraction was applied. CMLLR had full transformations. Gender independent acoustic model had a maximum of 15k Gaussian components, first trained on maximum likelihood estimation (MLE) and then followed by maximum mutual information (MMI) and boosted MMI (bMMI) trainings [8]. bMMI used boosting factor of 0.1. The language model was trained with text from web queries and catalogue.

Test set had 25 devices. Each device had 70 utterances, the i-vector feature dimension was empirically set to 10. The clustering by the new method resulted in 2 to 4 components if we set the minimum count per cluster to 10. Unsupervised CMLLR in Sec. 2 was applied in the following four cases. In the per utterance, or “per utt” case, CMLLR was applied per utterance without using statistics from other utterances in a device. In the case of per device, which corresponded to one cluster per device, CMLLR statistics were accumulated on all utterances from a device. CMLLR was then applied to those utterances from the device. We denote this case as “per dev 1cls”. In the case of per device 2 clusters, or “per dev 2cls”, utterances from each device were classified into 2 clusters. The case of per device 4 clusters, or “per dev 4cls”, had 4 clusters per device. CMLLR was then estimated for each cluster.

WERs are reported in Table 2. We observed that

1. In MLE system, “per dev 1cls” reduced WERs in comparison to “per utt” by 15%. “per dev 2cls” and “per dev 4cls” each further reduced WERs by 7% and 10%.

2. In MMI system, “per dev 1cls” reduced WERs in comparison to “per utt” by 17%. “per dev 2cls” and “per dev 4cls” each further reduced WERs by 10% and 4%.

3. In bMMI system, “per dev 1cls” reduced WERs in comparison to “per utt” by 16%. “per dev 2cls” and “per dev 4cls” each further reduced WERs by 12% and 5%.

4. On MLE system, the largest word error rate reduction (WERR) was achieved using 4 clusters. On MMI systems, the largest WERRs were achieved using 2 clusters.

Overall, the new i-vector based method improved performances over per device and per utterance adaptation. Results also seemed to show that, with better acoustic modeling using MMI and bMMI, the optimal number of clusters could be reduced. The optimal number of clusters was reduced to 2 with MMI and bMMI models, from 4 with MLE acoustic models.

5. Conclusions

We have presented a SR acoustic personalization method for mobile devices. This method uses a generalized i-vector clustering method to train i-vector parameters on utterances from a device and to classify test utterances from the same device. On a large vocabulary voice search task on Xbox kinect device, this approach significantly reduced WERs by 28% relative, in comparison to per utterance adaptation. We plan to extend this method to subspace fMLLR [6], which may train cluster-specific basis.

6. References


