Combining Deep Speaker Specific Representations with GMM-SVM for Speaker Verification

Ryan Price, Sangeeta Biswas, Koichi Shinoda

Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan
ryan@ks.cs.titech.ac.jp, sangeeta@ks.cs.titech.ac.jp, shinoda@cs.titech.ac.jp

Abstract
This study combines a Gaussian mixture model support vector machine (GMM-SVM) system with a nonlinear feature transformation, discriminatively trained to extract speaker specific features from MFCCs. Separation of the speaker information component and non-speaker related information in the speech signal is accomplished using a regularized siamese deep network (RSDN). RSDN learns a hidden representation that well characterizes speaker information by training a subset of the hidden units using pairs of speech segments. MFCC features are input to a trained RSDN and a subset of hidden layer outputs are used as new input features in a GMM-SVM system. We demonstrate the potential of this approach for text-independent speaker verification by applying it to a subset of the NIST SRE 2006 1conv4w-1conv4w task. The hybrid RSDN GMM-SVM system achieves about 5% relative improvement over the baseline GMM-SVM system.

Index Terms: speaker verification, neural networks, feature extraction, GMM-SVM

1. Introduction

Effective speaker verification systems are built from good input features, speaker models, and decision-making mechanisms. GMM-SVM with mel-frequency cepstral coefficients (MFCC) input features are one of the most prevalent approaches for speaker verification [1, 2, 3]. In the GMM-SVM approach, MAP adapted means of the mixture components are stacked to form supervectors. The supervectors are input to SVMs that model the boundary between a speaker and a set of imposters, rather than modeling their probability distributions. One downside to the generative approach of GMM speaker modeling is the lack of the ability to extract speaker specific information by discriminative means. Furthermore, the widely used MFCC input features in these systems are not designed with the objective of maximizing speaker verification performance. Discriminatively trained features may be better suited to the problem of speaker verification.

Recent studies (e.g. [4, 5, 6]) have shown that regularized siamese deep networks (RSDN) show promise as an effective method for extracting speaker specific information from a spectral representation. However, they have not yet been successfully paired with a robust approach to speaker modeling and decision-making for speaker verification. We address this by combining a GMM-SVM system with speaker specific input features extracted from a discriminatively trained RSDN. We demonstrate the potential of this approach by applying it to a subset of the NIST SRE 2006 task [7]. The main contributions of this paper are the application of RSDN to standard benchmark datasets for speaker verification. We demonstrate that RDSN extracted features still offer favorable results compared to MFCC features even when longer test utterances are used and when the network is trained on data with different channel effects and handset types. This work is the first time features extracted from a deep, discriminatively trained network were combined with a GMM-SVM system for speaker verification as far as we know.

In the next section we review some related studies on neural network feature extraction for speaker recognition. Then the RSDN architecture we use for extracting speaker specific features from MFCC inputs is described. Section 4 describes the hybrid RSDN GMM-SVM system for speaker verification, and Section 5 provides the details of our experimental setup and presents our results. We finish the paper with some conclusions and possibilities for future work.

2. Related Studies

Much research has been done on extracting speaker discriminative features from cepstral features using multi-layer perceptrons (MLP) (e.g. [8, 9, 10, 11]). A common approach was to train an MLP to discriminate between a set of selected speakers, with a window of several frames of MFCC features input to the neural network. After sufficient training of the MLP, weights are fixed and features are extracted from a hidden layer, usually a bottleneck layer, and are used as inputs to train another classifier for speaker identification or speaker verification. Although these neural networks typically achieve low speaker classification accuracy in the task they are trained on, the speaker-discriminative feature set they generate can result in significant improvement, especially when scores are combined with MFCC based systems.

In [9], hidden layer activations from MLPs trained to distinguish between a subset of speakers selected through a speaker clustering process were used as input features for a SVM speaker recognition system. Stoll et al. [9] also examined the use of features generated by an MLP that was trained to distinguish between phones as input to a GMM speaker recognition system. [11, 12] focused on extracting features that were robust to mismatched training and testing conditions of speaker verification systems by training MLPs with data for the same speakers under different conditions. Morris et al. [10] dealt with the question of which subset of speakers would be most effective for MLP training when data is available for a large number of speakers. In [13] separate Neural Predictive Coding models (a nonlinear extension of Linear Predictive Coding) were trained to process features for individual speakers, rather than carrying out speech feature extraction in the same way for all speakers.

Previous attempts demonstrate some success at using neural network hidden representations as features for speaker recognition, especially when scores are fused with MFCC based sys-
3. Regularized siamese deep network

3.1. Overview

Unlike standard deep architectures trained with speech input, RSDN learns a speaker specific representation by discriminatively training a subset of the hidden units using pairs of speech segments (Fig. 1). This subset of code layer units (we refer to the middlemost hidden layer as the “code layer”) learns a stable representation of the speech from a given speaker. The representation learned for each speaker should also be different for different speakers. The remaining code layer units act as a regularizer. This is accomplished using 2 types of loss functions described in Subsection 3.3. In addition to speaker verification tasks, RSDN extracted features have been applied to speaker comparison and speaker segmentation [4, 5, 14]. Training the RSDN can be divided into two phases which are described next.

3.2. Pretraining

The pretraining phase serves as means of initializing the weights and biases of a deep autoencoder using unlabeled data, which can then be converted to a siamese network and discriminatively trained using labeled pairs of speech segments. One method for initializing a deep autoencoder is to stack denoising autoencoders [15]. Denoising autoencoders, like classical autoencoders, map an input to a hidden representation, which is then mapped back to a reconstruction of the input. Howver, denoising autoencoders reconstruct the input from a corrupted version of it and allow for an overcomplete hidden layer while still learning a useful intermediate representation. Mean squared error loss between the clean input and the reconstructed input is minimized with respect to the weights and biases.

During the RSDN pretraining phase, unlabeled speech data, in this case MFCC features, are used for training a stacked denoising autoencoder. MFCC features are continuous valued so Gaussian noise in the form of \( \mathcal{N}(0, \sigma) \), where \( \sigma \) is the standard deviation of feature \( i \) estimated from the training data, is added to give a corrupted version of the inputs during layerwise pretraining. After the pretraining phase, the deep autoencoder is duplicated to form a siamese network with two identical subnets (see Figure 1). The weights and biases of the the two halves of the siamese network are shared and any subsequent weight update is applied to both halves, keeping their values the same.

3.3. Discriminative training

During the discriminative training phase, pairs of short speech segments \( (X_1, X_2) \), \( T \) frames in length, coming either from the same speaker (genuine pairs) or from different speakers (imposter pairs), are presented to the network. The contrastive loss function (Eq. (1)) is applied to a subset of the code layer units in order to learn a speaker specific representation and is actually a combination of two cost functions - one minimizes the objective with respect to genuine pairs, and the other minimizes with respect to imposter pairs. A second loss function, reconstruction loss (Eq. (4)), provides a form of weight regularization during discriminative training by requiring the network still be good at reconstructing the input.

The contrastive loss \( L_C \) is defined as:

\[
L_C(X_1, X_2) = I[C_m + C_s] + (1 - I)[\epsilon - \frac{C_m}{\lambda_m} + \epsilon - \frac{C_s}{\lambda_s}] \tag{1}
\]

where

\[
C_m = \|\mu_{S_1} - \mu_{S_2}\|^2 \tag{2}
\]

\[
C_s = \|\Sigma_{S_1} - \Sigma_{S_2}\|^2 \tag{3}
\]

with \( \mu_{S_1}, \mu_{S_2}, \Sigma_{S_1} \) and \( \Sigma_{S_2} \) being the means and covariance matrices of the outputs of speaker specific code layer units corresponding to the segment pair \( (X_1, X_2) \), respectively, and \( ||\cdot||_F \) is the Frobenius norm. \( I = 1 \) for genuine pairs and 0 for imposter pairs, \( \lambda_m \) and \( \lambda_s \) are hyperparameters set to approximately balance the value of the contrastive loss function for genuine pairs and imposter pairs.

The reconstruction loss \( L_R \) is defined as:

\[
L_R(X_1, X_2) = \frac{1}{T} \sum_{t=1}^{T} \|x_{1t} - \hat{x}_{1t}\|^2 + \|x_{2t} - \hat{x}_{2t}\|^2 \tag{4}
\]

The contrastive loss and reconstruction loss are combined in the overall loss function (Eq. (5)) which is minimized during discriminative training.

\[
L(X_1, X_2) = \alpha L_R + (1 - \alpha)L_C \tag{5}
\]

where \( \alpha \) determines the trade-off between \( L_R \) and \( L_C \).

4. Hybrid RSDN GMM-SVM

4.1. RSDN speaker specific feature extractor

Previous studies on RSDN extracted features for speaker verification [5, 6, 14] used RSDN code layer outputs to derive single Gaussian speaker models from short speech segments. Scores for binary classification of a given test trial were calculated using a simplified form of the divergence metric for two normal distributions from [16], or by symmetric Gaussian log likelihood measure [17]. Performance based on equal error rate (EER) and minimum detection cost (MDC) metrics was...
Figure 2: Only the encoder portion of the trained RSDN is retained for efficient feature extraction.

compared with features extracted from autoencoders and convolutional neural networks, as well as a GMM trained with 19 dimensional MFCC features. While that approach offered promising results in the speaker verification tasks studied, we believe that the speaker specific features extracted from the RSDN code layer can be combined with a more robust speaker modeling approach and classifier. We propose a straightforward approach to using RSDN representations along with GMM-SVM to form a hybrid system for speaker verification.

Since we are only interested in using the RSDN as a nonlinear feature extractor, after pretraining and discriminative training the RSDN as described in Section 3, the siamese network is no longer needed and we keep only the encoder portion of the first half of the siamese network, allowing for more efficient feature extraction (Fig. 2).

4.2. Combining with GMM-SVM

The remaining nonlinear feature extractor (Fig. 2) is used to extract speaker specific features from MFCC inputs for each utterance in the universal background model (UBM), enrollment and evaluation data. GMM-SVM training follows the typical GMM-SVM training procedure [1] starting with using the extracted features to derive a GMM UBM. RSDN code layer outputs are then extracted for all frames in the enrollment and evaluation data. Next, GMM supervectors are created on a per utterance basis using MAP adaptation of the means with a relevance factor of 1. GMM supervectors extracted from the utterances used to train the UBM are used as imposter examples to train an SVM model with a linear kernel for each target speaker in the enrollment set. Finally, scores are calculated for each target and nontarget trial using the target speaker’s SVM model and the supervector extracted from the test utterance.

5. Experiment

In this section we describe the experimental settings, including the preprocessing of the raw inputs and also the network training. We then describe the copora used in our experiments. Finally, we present the results of a speaker verification task and demonstrate the effectiveness of the hybrid system compared to a well performing GMM-SVM baseline with MFCC features.

5.1. Experimental setup

We use the following procedure for extracting MFCC features for use as input features. Silence was removed using an energy based VAD. The speech signal was pre-emphasized by applying the first order difference equation \( s_n = s_n - 0.95s_{n-1} \). A 25 msec Hamming window and 10 ms frame rate are used to extract a 19 dimensional MFCC vector (zero order coefficient is excluded) from 24 filterbank channels. Cepstral mean normalization is applied to account for long-term spectral effects caused by channel differences.

Details of RSDN training are as follows. All frames in the training data were used for pretraining. For discriminative training, we created approximately 6000 pairs of segments \((X_1, X_2)\) that are 500 frames in length by splitting up the training data into a set, \( S \), of non-overlapping segments and randomly select another a segment, \( X_2 \), for each segment \( X_1 \in S \), with \( X_2 \neq X_1 \). We ensured the data set is “balanced” by generating approximately the same number of genuine pairs as imposter pairs.

Mini-batch sizes were 100 frames and 500 frames for pretraining and discriminative training, respectively. Note that Eq. (1) is defined using 1st and 2nd order statistics of code layer outputs generated from a speech segment that is \( T \) frames in length and it is thus necessary to use a mini-batch size of \( T \) for discriminative training. We used a network with 5 hidden layers having sizes of 100, 100, 200, 100, and 100 hidden units, respectively. We used 100 speaker specific units in the code layer, which is half the number of hidden units in that layer, as this was found to be advantageous in [14, 4]. The network was pretrained for 40, 20, 20 epochs at learning rate of 0.01 before discriminative training with a learning rate of 0.001. We set the trade-off parameter \( \alpha \) to 0.2. Early stopping and learning rate annealing were used to prevent overfitting by checking EER on the development data, and monitoring the contrastive and reconstruction losses. GPU implementation of the RSDN was done in Python using the Theano library [18].

5.2. Text-independent speaker verification

We evaluate the performance of the hybrid RSDN GMM-SVM system on subsets of NIST SRE 2004 and 2006 [7]. The 242 male speakers from NIST SRE 2004 1-side were selected for pretraining and discriminative training of the RSDN. Utterances from 50 randomly selected male speakers who did not appear in the training data were taken from NIST SRE 2004 8-side training files and used for RSDN development. The same utterances from NIST SRE 2004 1-side that we used for RSDN training were used to train the UBM. For evaluation, we randomly selected 100 male speakers from the NIST SRE 2006 1conv4w, 1conv4w task and used all trials associated with those speakers. In total, 453 genuine trials and 6057 imposter trials were used for evaluation.

Previous studies showing the effectiveness of RSDN extracted features for speaker verification [5, 6] have focused on using short test utterances (5s or less) and weren’t trained on data with channel and environmental mismatch. In contrast, the training and test utterances used in this experiment for both our hybrid system and baseline GMM-SVM are considerably longer in length (average of 2 minutes after silence removal) and are trained on data with different channel effects and handset types. Most of the segments are in English (but include non-native speakers) and are recorded over a telephone line, but different languages, types of telephone handsets and transmission channels are included.

5.3. Results and analysis

5.3.1. Comparison with MFCC based GMM-SVM

We use EER as an evaluation metric. The number of GMM mixture components was varied from 32 to 256 and the EER for both systems is shown in Table 1. The best performing hybrid system achieved 12.58% EER with 64 mixture components. This amount to about a 5% relative reduction in EER and demonstrates that combining speaker specific features extracted from
Table 1: EER(%) for MFCC based GMM-SVM and Hybrid RSDN GMM-SVM.

<table>
<thead>
<tr>
<th>System</th>
<th>Mix Comp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32</td>
</tr>
<tr>
<td>GMM-SVM (MFCC)</td>
<td>14.77</td>
</tr>
<tr>
<td>Hybrid RSDN GMM-SVM</td>
<td>12.59</td>
</tr>
</tbody>
</table>

RSDN with a GMM-SVM system can be an effective approach.

It should be noted that there are several differences in the GMM-SVM system evaluated here compared to those evaluated by others who have also used NIST’04 for background training data and NIST’06 for evaluation, such as [3], were all of NIST’04 was used (we use a subset) and concepts like score normalization, nuisance attribute projection (NAP), and factor analysis were also applied.

The dimensionality of the extracted speaker specific feature vectors used to train the hybrid system is about 5 times greater than that of the MFCC feature vectors used to train the baseline GMM-SVM. Given that the number of training vectors remains the same for both systems, it is not surprising that the best performing hybrid system has fewer mixture components relative to the baseline GMM-SVM. We observe that while the performance of the MFCC based GMM-SVM system remains relatively unchanged for 64, 128, and 256 mixture components, the performance of the hybrid system declines abruptly when the number of mixture components is increased beyond 64, probably due to overfitting.

5.3.2. Score fusion

Several of the neural network feature extraction methods discussed in Section 2 showed improvement using score fusion with MFCC based systems, even when the extracted features performed poorly alone. For simplicity, we have taken a linear combination of the scores from the 64 mixture component hybrid RSDN GMM-SVM and the 128 mixture component MFCC GMM-SVM, giving scores from the 2 systems equal weighting. The Detection Error Tradeoff (DET) curves for the fused scores, hybrid RSDN GMM-SVM, and MFCC based GMM-SVM are plotted in Figure 3. The 3 curves follow similar trends but the hybrid RSDN GMM-SVM and score fusion consistently outperform the MFCC based GMM-SVM. Score fusion of the hybrid RSDN GMM-SVM and MFCC based GMM-SVM provides some further reduction in EER. Score fusion achieves an EER of 12.36%, which is a 6.6% relative improvement over the MFCC based GMM-SVM.

5.3.3. Alternative code layer outputs

In [8] the net input values to MLP hidden and output layers were used as features since many of the values were close to 0 or 1 after applying the nonlinearity and data with a very peaked distribution is not well suited for use as input features. We also tried extracting the net input values to the speaker specific code layer units prior to applying the nonlinearity and trained a 64 mixture component GMM-SVM system but found that it did not give better performance than the system trained on the nonlinear activations. We examined the values of these extracted features after applying the sigmoid nonlinearity and found that they were not typically dominated by values near 0 or 1. While exploring the training characteristics of RSDN in early experiments, we noticed that when the too many of the outputs are outside the “linear” regime of the sigmoid, their performance as features for RSDN may also improve performance. We have used only a single frame of acoustic features as input to the RSDN but believe the performance would likely benefit from inputing a window of several frames to the RSDN, rather than one frame at a time.

6. Conclusions and future work

We have demonstrated the novel combination of RSDN extracted features and a GMM-SVM system for text-independent speaker verification. This hybrid system outperformed the baseline GMM-SVM system based on MFCC input features for the subset of the NIST SRE 2006 1conv4w-1conv4w task studied. These are much longer utterances than previous studies of RSDN extracted features and include different channel effects and handset types. This straightforward approach for a hybrid RSDN GMM-SVM system yielded a modest performance gain of 5% relative reduction in EER. Score fusion offers further improvement, giving a 6.6% relative reduction in EER.

The hybrid RSDN GMM-SVM system has a considerable number of parameters, many of which would benefit from a more thorough tuning than was done here. Often delta and delta-delta features are appended to the MFCC coefficients and feature post-processing methods for increased robustness, such as feature warping [19], are applied for speaker verification. More work needs to be done to determine the optimal raw input features for RSDN feature extraction and performance should be compared to features that are commonly used in speaker verification approaches today. Application of channel compensation techniques, such as NAP [20], and score normalization [21] may also improve performance. We have used only a single frame of acoustic features as input to the RSDN but believe the performance would likely benefit from inputing a window of several frames to the RSDN, rather than one frame at a time.
7. References


