Lecture 19: Speaker Verification

Mark Hasegawa-Johnson

University of Illinois

ECE 417: Multimedia Signal Processing

Outline

Speaker Verification: Problem Statement

- Given: two test utterances, $x_1[n]$ and $x_2[n]$
- Decide: are they from the same speaker or not?

Why it's hard

- Usually, the test speakers were not represented in your training database.
- Usually, you don't know what they are saying.
- Usually, $x_1[n]$ and $x_2[n]$ are saying different things.
- Usually, $x_1[n]$ and $x_2[n]$ are of different lengths.

General Principles

- Convert $x_1[n]$ and $x_2[n]$ to spectrograms.
- Create very high-dimensional supervectors

$$\boldsymbol{s}_1 = \begin{bmatrix} \boldsymbol{\mu}_{1,1} \\ \vdots \\ \boldsymbol{\mu}_{1,K} \end{bmatrix}, \quad \boldsymbol{s}_2 = \begin{bmatrix} \boldsymbol{\mu}_{2,1} \\ \vdots \\ \boldsymbol{\mu}_{2,K} \end{bmatrix},$$

  where $\boldsymbol{\mu}_{j,k}$ is an estimate of the way in which speaker j says phoneme k.
- Create intermediate-dimension embedding vectors $\boldsymbol{w}_j = \boldsymbol{T}^T \boldsymbol{s}_j$, where $\boldsymbol{T}$ is trained to optimize performance on a training database.
- Measure the similarity between $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$.

Outline

788 IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 19, NO. 4, MAY 2011

Front-End Factor Analysis for Speaker Verification

Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet

***Abstract*—This paper presents an extension of our previous work which proposes a new speaker representation for speaker verification. In this modeling, a new low-dimensional speaker- and channel-dependent space is defined using a simple factor analysis. This space is named the total variability space because it models both speaker and channel variabilities. Two speaker verification systems are proposed which use this new representation. The first system is a support vector machine-based system that uses the cosine kernel to estimate the similarity between the input data. The second system directly uses the cosine similarity as the final decision score. We tested three channel compensation techniques in the total variability space, which are within-class covariance normalization (WCCN), linear discriminate analysis (LDA), and nuisance attribute projection (NAP). We found that the best results are obtained when LDA is followed by WCCN. We achieved an equal error rate (EER) of 1.12% and MinDCF of 0.0094 using the cosine distance scoring on the male English trials of the core condition of the NIST 2008 Speaker Recognition Evaluation dataset. We also obtained 4% absolute EER improvement for both-gender trials on the 10 s-10 s condition compared to the classical joint factor analysis scoring.**

***Index Terms*—Cosine distance scoring, joint factor analysis**

At the same time, the application of support vector machines (SVMs) in GMM supervector space [5] yields interesting results, especially when nuisance attribute projection (NAP) is applied to deal with channel effects. In this approach, the kernel used is based on a linear approximation of the Kullback–Leibler (KL) distance between two GMMs. The speaker GMMs mean supervectors were obtained by adapting the universal background model (UBM) mean supervector to speaker frames using maximum *a posteriori* (MAP) adaptation [4].

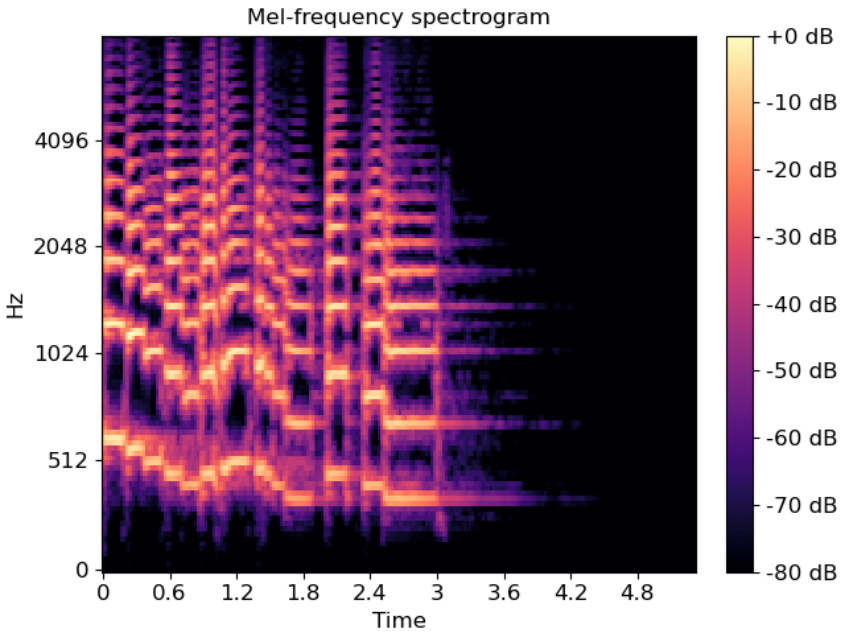
In [6], [7], we proposed a new way of combining JFA and SVMs for speaker verification. It consists in directly using the speaker factors estimated with JFA as input to the SVM. We tested several kernels and the best results were obtained using the cosine kernel [6] when within-class covariance normalization (WCCN) [8] is also used to compensate for residual channel effects in the speaker factor space.

Recently [6], we carried out an experiment which proves that channel factors estimated using JFA, which are supposed to model only channel effects, also contain information about

Identity Vectors (i-vectors) Overview

1. Convert waveform to MFCC vectors $\boldsymbol{x}_t$
2. Gaussian mixture model soft-assigns each $\boldsymbol{x}_t$ to a phoneme.
3. Baum-Welch computes the *differences* between this speaker's phonemes and the typical phonemes.
4. PCA reduces the supervector to an i-vector.
5. Measure similarity between two i-vectors using cosine similarity.

Mel Frequency Cepstral Coefficients



- Compute STFT
- Convert to mel spectrogram by nonlinearly resampling the frequency axis, to get semi-logarithmic frequencies
- Convert from mel spectrogram to MFCC: approximate the PCA of each column using a discrete cosine transform

Gaussian Mixture Model

A Gaussian mixture model is like an HMM, but simpler:

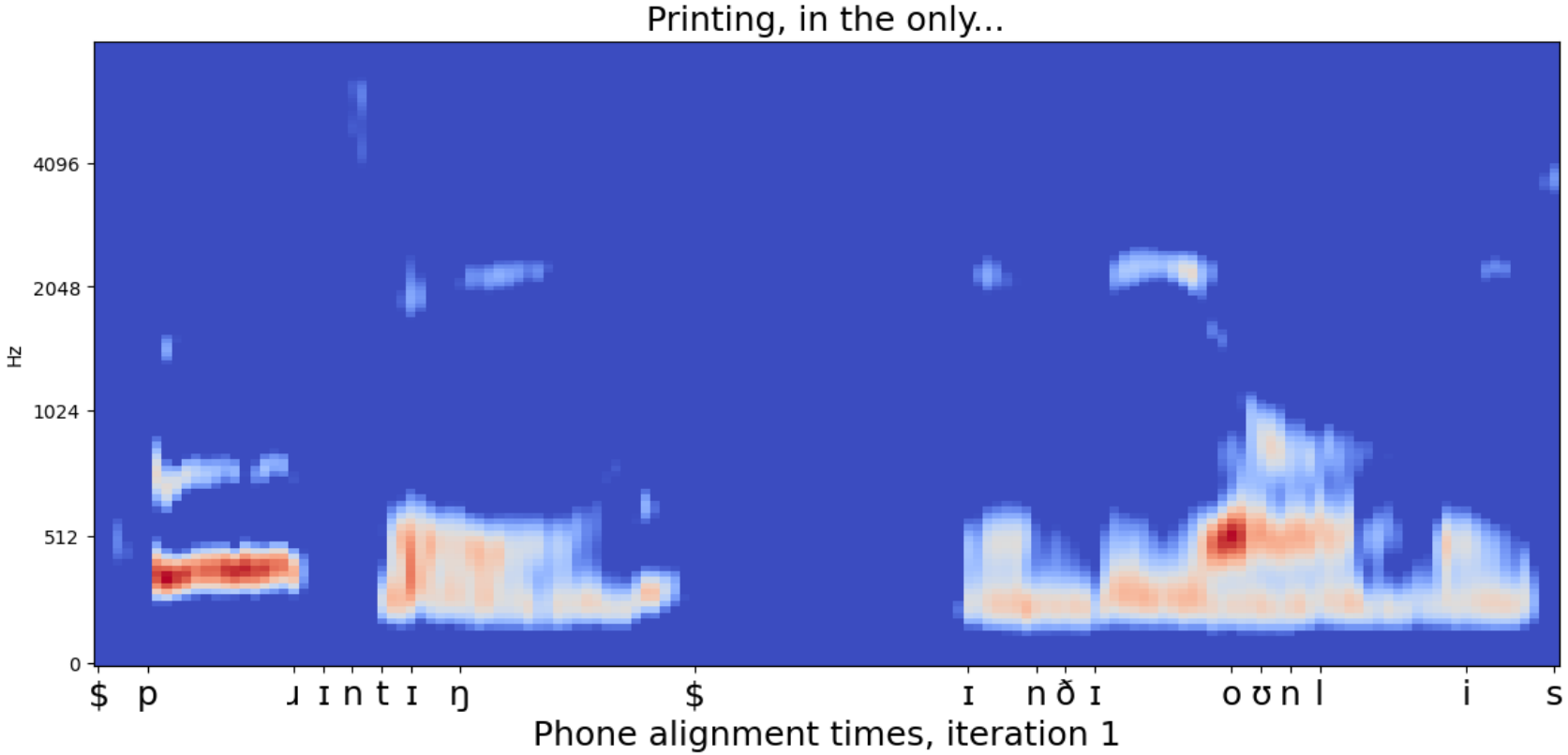
$$\Pr\{q_t = i\} = a_i$$
$$\Pr\{\boldsymbol{x}_t | q_t = i\} = \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

The posterior probability of phoneme i at time t is therefore:

$$\gamma_t(i) = \Pr\{q_t = i | \boldsymbol{x}_t\} = \frac{\Pr\{q_t = i, \boldsymbol{x}_t\}}{\sum_k \Pr\{q_t = k, \boldsymbol{x}_t\}} = \frac{a_i \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_k a_k \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

Soft Phoneme Segmentation using GMM

The image below shows hard phoneme alignment. A GMM computes soft phoneme alignment, $\gamma_t(i) = \Pr\{q_t = i|\boldsymbol{x}_t\}$.



GMM Supervector

A GMM supervector lists, for each phoneme, the **differences** between the way in which this person produces the phoneme and the way in which a typical person would produce the phoneme:

$$\boldsymbol{s} = \begin{bmatrix} \frac{\sum_t \gamma_t(1)(\boldsymbol{x}_t - \boldsymbol{\mu}_1)}{\sum_t \gamma_t(1)} \\ \vdots \\ \frac{\sum_t \gamma_t(N)(\boldsymbol{x}_t - \boldsymbol{\mu}_N)}{\sum_t \gamma_t(N)} \end{bmatrix}$$

The dimension of this vector is very large. Typically each $\boldsymbol{x}_t$ is a 40-dimensional MFCC vector, and typically there are 2048 Gaussians, for a total $\text{len}(\boldsymbol{s}) = 2048 \times 40 = 81920$.

i-vector

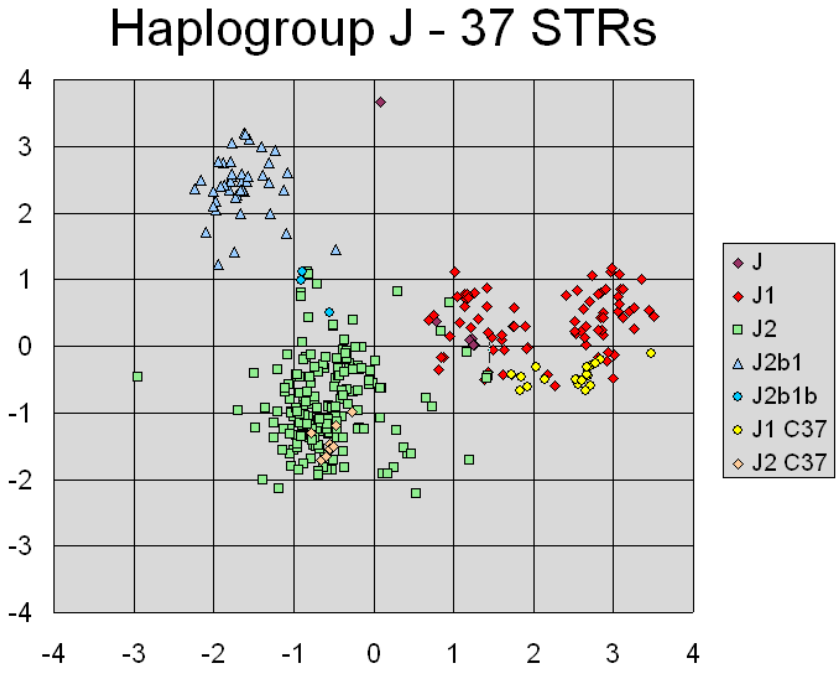
- The i-vector is a principal components analysis of the GMM supervector:

$$\boldsymbol{w} = \boldsymbol{G}^{-1}\boldsymbol{T}^{T}\boldsymbol{s}$$

  where the columns of $\boldsymbol{T} \in \Re^{81920\times 600}$ are the eigenvectors of the covariance of $\boldsymbol{s}$, and $\boldsymbol{G}$ is a diagonal normalization matrix related to the eigenvalues.
- The dimension of $\boldsymbol{w}$ is chosen to be intermediate between the dimension of $\boldsymbol{s}$ (81920) and the dimension of $\boldsymbol{x}_t$ (40). Typically, $\text{len}(\boldsymbol{w}) \approx 600$. Although i-vector originally meant "identity vector," people sometimes also gloss it as "intermediate vector."

Similarity

- The i-vector is a **PCA** of the **differences** between the test speaker's productions of phonemes versus the typical production. How do we evaluate similarity in this space?
- The problem: covariances are diagonal in this space, but not uniform (example at right is chemical analysis, but speaker verification looks similar).



https://commons.wikimedia.org/wiki/File:PCA_of_Haplogroup_J_using_37_STRs.png

Similarity Measures

The original i-vector paper tested two similarity measures:

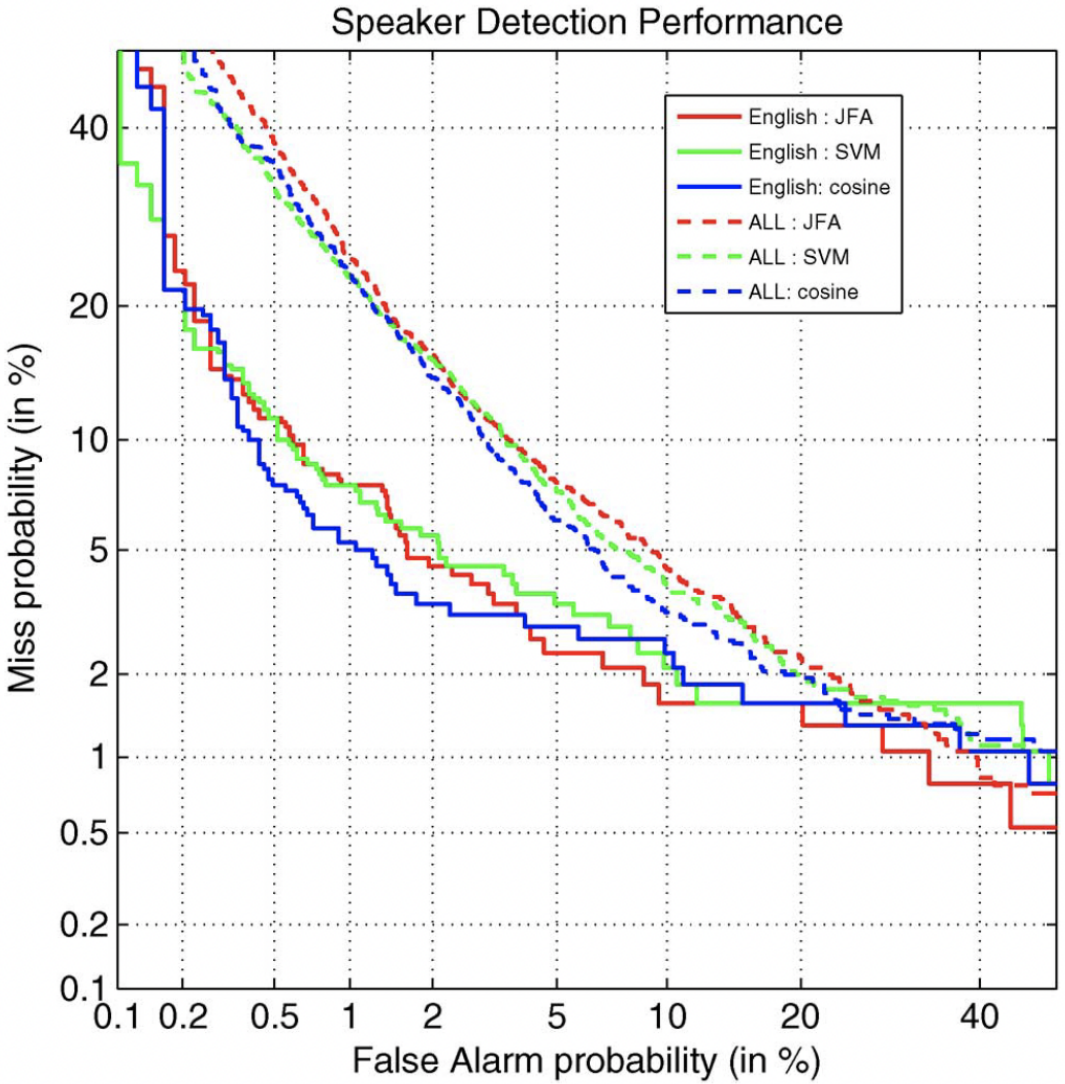
- Directly threshold the cosine similarity:

$$y = \begin{cases} 1 & \frac{\boldsymbol{w}_1^T \boldsymbol{w}_2}{\|\boldsymbol{w}_1\| \|\boldsymbol{w}_2\|} > \theta \\ 0 & \text{otherwise} \end{cases}$$

- Train a binary SVM using the cosine similarity as the kernel

Both of these measures, applied to i-vectors, were tested against joint factor analysis (JFA), the previous state of the art.

i-vector: Results



Outline

X-VECTORS: ROBUST DNN EMBEDDINGS FOR SPEAKER RECOGNITION

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, Sanjeev Khudanpur

Center for Language and Speech Processing & Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA

ABSTRACT

In this paper, we use data augmentation to improve performance of deep neural network (DNN) embeddings for speaker recognition. The DNN, which is trained to discriminate between speakers, maps variable-length utterances to fixed-dimensional embeddings that we call x-vectors. Prior studies have found that embeddings leverage large-scale training datasets better than i-vectors. However, it can be challenging to collect substantial quantities of labeled data for training. We use data augmentation, consisting of added noise and reverberation, as an inexpensive method to multiply the amount of training data and improve robustness. The x-vectors are compared with i-vector baselines on Speakers in the Wild and NIST SRE 2016 Cantonese. We find that while augmentation is beneficial in the PLDA classifier, it is not helpful in the i-vector extractor. However, the x-vector DNN effectively exploits data augmentation, due to its supervised training. As a result, the x-vectors achieve superior performance on the evaluation datasets.

Alternatively, neural networks can be directly optimized to discriminate between speakers. This has potential to produce powerful, compact systems [13], that only require speaker labels to train. In early systems, neural networks are trained to separate speakers, and frame-level representations are extracted from the network and used as features for Gaussian speaker models [14, 15, 16]. Heigold et al., introduced an end-to-end system, trained on the phrase "OK Google," that jointly learns an embedding along with a similarity metric to compare pairs of embeddings [13]. Snyder et al., adapted this approach to a text-independent application and inserted a temporal pooling layer into the network to handle variable-length segments [17]. The work in [1] split the end-to-end approach into two parts: a DNN to produce embeddings and a separately trained classifier to compare them. This facilitates the use of all the accumulated backend technology developed over the years for i-vectors, such as length-normalization, PLDA scoring, and domain adaptation techniques.

DNN embedding performance appears to be highly scalable with

x-vector: Acoustic features

Acoustic features are mel spectrograms with 40 mel bands, then passed through a 5-layer 1d CNN:

$$h_i^{(l)}[n] = g\left(\sum_j \sum_m w_{i,j}^{(l)}[m] h_j^{(l-1)}[n-m]\right)$$

The five layers have 512, 512, 512, 512, and 1500 channels, respectively.

x-vector: Supervector

- The last layer of the CNN has 1500-dimensional vectors $\boldsymbol{h}^{(5)}[n] = [h_1^{(5)}[n], \ldots, h_{1500}^{(5)}[n]]^T$. That's large enough to permit different sections of the vector to represent mean shift of different phonemes.
- The supervector is a concatenation of the utterance mean and variance, $\boldsymbol{s} = \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{v} \end{bmatrix}$, where

$$\boldsymbol{m} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{h}^{(5)}[n], \quad \boldsymbol{v} = \frac{1}{N-1}\sum_{n=1}^{N} \left(\boldsymbol{h}^{(5)}[n] - \boldsymbol{m}\right)^2$$

x-vector: Intermediate vector

- The x-vector itself is a linear compression of $\boldsymbol{s}$ down to 512 dimensions:

$$\boldsymbol{w} = \boldsymbol{T}^T \boldsymbol{s},$$

  where $\boldsymbol{T} \in \Re^{512 \times 3000}$ is a learned projection matrix.
- The matrix $\boldsymbol{T}$ is trained so that a further two-layer neural net, applied to $\boldsymbol{x}$, classifies all of the training speakers with minimum cross-entropy.
- After training, the 2-layer network is discarded. It's not needed, because none of the test speakers are in the training database anyway.

Similarity: Probabilistic Linear Discriminant Analysis (PLDA)

Probabilistic linear discriminant analysis (PLDA) assumes that we a training set of N speakers (not including the test speaker!). The i^{th} training speaker is represented by some x-vectors $\boldsymbol{w}_{i,1}, \ldots, \boldsymbol{w}_{i,m}$, whose mean is

$$\boldsymbol{v}_i = \frac{1}{m} \sum_{j=1}^{m} \boldsymbol{w}_{i,j}$$

PLDA models $\boldsymbol{v}_i$ and $\boldsymbol{w}_{i,j}$ as jointly Gaussian:

$$\boldsymbol{v}_i \sim \mathcal{N}(\boldsymbol{v}|\boldsymbol{0}, \boldsymbol{A}\boldsymbol{\Psi}\boldsymbol{A}^T)$$
$$\boldsymbol{w}_{i,j} \sim \mathcal{N}(\boldsymbol{w}|\boldsymbol{v}_i, \boldsymbol{A}\boldsymbol{A}^T),$$

The matrices $\boldsymbol{A}$ and $\boldsymbol{\Psi}$ are a generalization of PCA called linear discriminant analysis.

Similarity: Probabilistic Linear Discriminant Analysis (PLDA)

PLDA then computes the similarity between two test vectors, $\boldsymbol{u}_1 = \boldsymbol{A}^T \boldsymbol{w}_1$ and $\boldsymbol{u}_2 = \boldsymbol{A}^T \boldsymbol{w}_2$, as

$$R(\boldsymbol{w}_1, \boldsymbol{w}_2) = \frac{\text{likelihood(same)}}{\text{likelihood(different)}}$$
$$= \frac{\int \Pr(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{v}) d\boldsymbol{v}}{\left(\int \Pr(\boldsymbol{u}_1, \boldsymbol{v}_1) d\boldsymbol{v}_1\right) \left(\int \Pr(\boldsymbol{u}_2, \boldsymbol{v}_2) d\boldsymbol{v}_2\right)}$$

If we set $\bar{\boldsymbol{u}} = \frac{1}{2}(\boldsymbol{u}_1 + \boldsymbol{u}_2)$, then we can write:

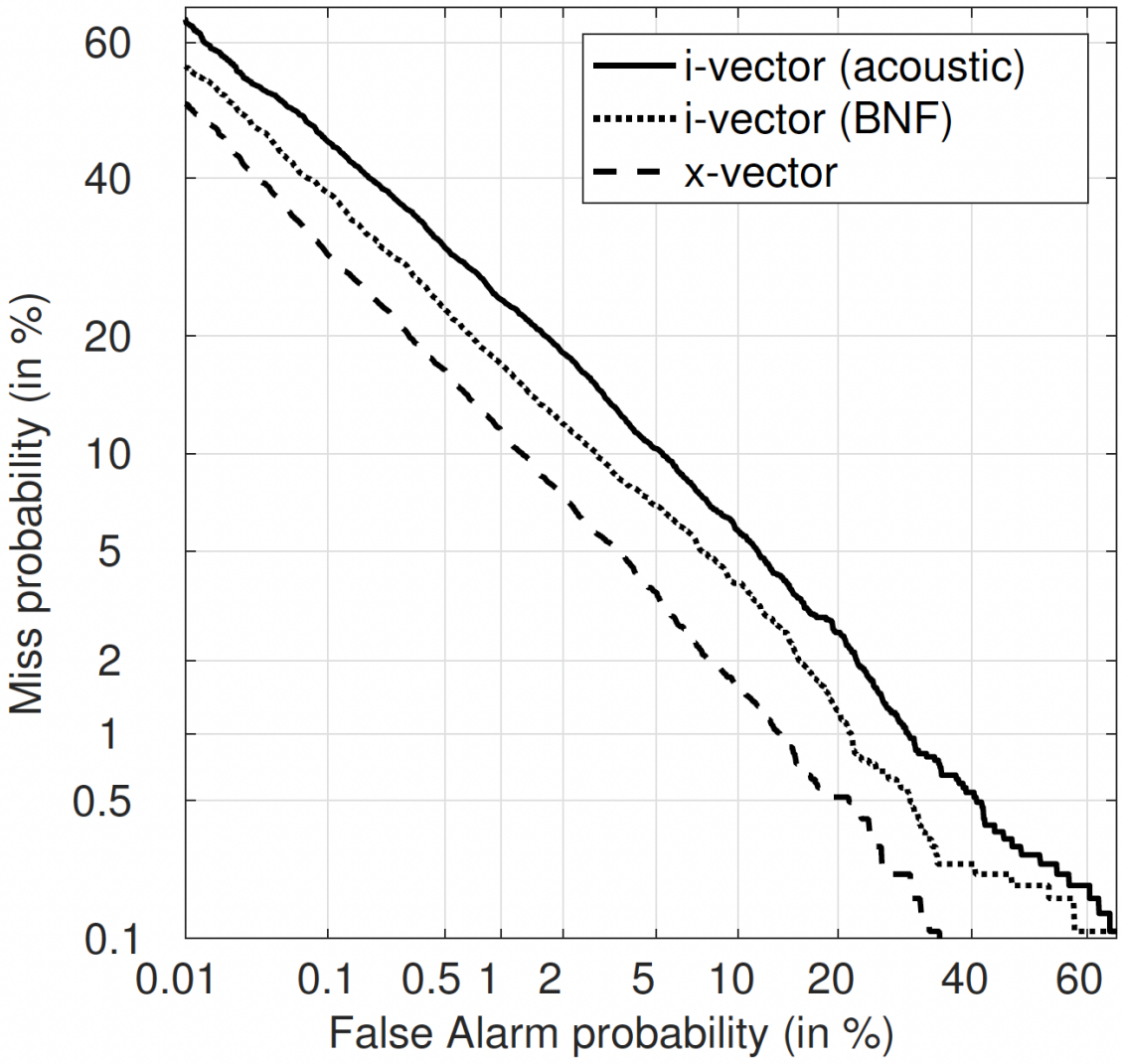
$$R(\boldsymbol{w}_1, \boldsymbol{w}_2) \propto \frac{\mathcal{N}\left(\bar{\boldsymbol{u}} | \boldsymbol{0}, \Psi + \frac{1}{2}\boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{u}_1 | \bar{\boldsymbol{u}}, \boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{u}_2 | \bar{\boldsymbol{u}}, \boldsymbol{I}\right)}{\mathcal{N}\left(\boldsymbol{u}_1 | \Psi + \boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{u}_2 | \Psi + \boldsymbol{I}\right)},$$

Similarity: Probabilistic Linear Discriminant Analysis (PLDA)

Two test utterances are then judged to have come from the same speaker if their PLDA score is above a threshold:

$$y = \begin{cases} 1 & R(\boldsymbol{w}_1, \boldsymbol{w}_2) > \theta \\ 0 & \text{otherwise} \end{cases}$$

x-vector: Results



Outline

GENERALIZED END-TO-END LOSS FOR SPEAKER VERIFICATION

Li Wan *Quan Wang* *Alan Papir* *Ignacio Lopez Moreno*

Google Inc., USA

{liwan, quanw, papir, elnota} @google.com

ABSTRACT

In this paper, we propose a new loss function called generalized end-to-end (GE2E) loss, which makes the training of speaker verification models more efficient than our previous tuple-based end-to-end (TE2E) loss function. Unlike TE2E, the GE2E loss function updates the network in a way that emphasizes examples that are difficult to verify at each step of the training process. Additionally, the GE2E loss does not require an initial stage of example selection. With these properties, our model with the new loss function decreases speaker verification EER by more than 10%, while reducing the training time by 60% at the same time. We also introduce the MultiReader technique, which allows us to do domain adaptation — training a more accurate model that supports multiple keywords (*i.e.*, "OK Google" and "Hey Google") as well as multiple dialects.

Index Terms— Speaker verification, end-to-end loss, Multi-

1.2. Tuple-Based End-to-End Loss

In our previous work [13], we proposed the tuple-based end-to-end (TE2E) model, which simulates the two-stage process of runtime enrollment and verification during training. In our experiments, the TE2E model combined with LSTM [14] achieved the best performance at the time. For each training step, a tuple of one evaluation utterance $\mathbf{x}_{j\sim}$ and M enrollment utterances $\mathbf{x}_{km}$ (for $m = 1, \ldots, M$) is fed into our LSTM network: $\{\mathbf{x}_{j\sim}, (\mathbf{x}_{k1}, \ldots, \mathbf{x}_{kM})\}$, where $\mathbf{x}$ represents the features (log-mel-filterbank energies) from a fixed-length segment, j and k represent the speakers of the utterances, and j may or may not equal k. The tuple includes a single utterance from speaker j and M different utterance from speaker k. We call a tuple positive if $\mathbf{x}_{j\sim}$ and the M enrollment utterances are from the same speaker, *i.e.*, $j = k$, and negative otherwise. We generate positive and negative tuples alternatively.

For each input tuple, we compute the L2 normalized response

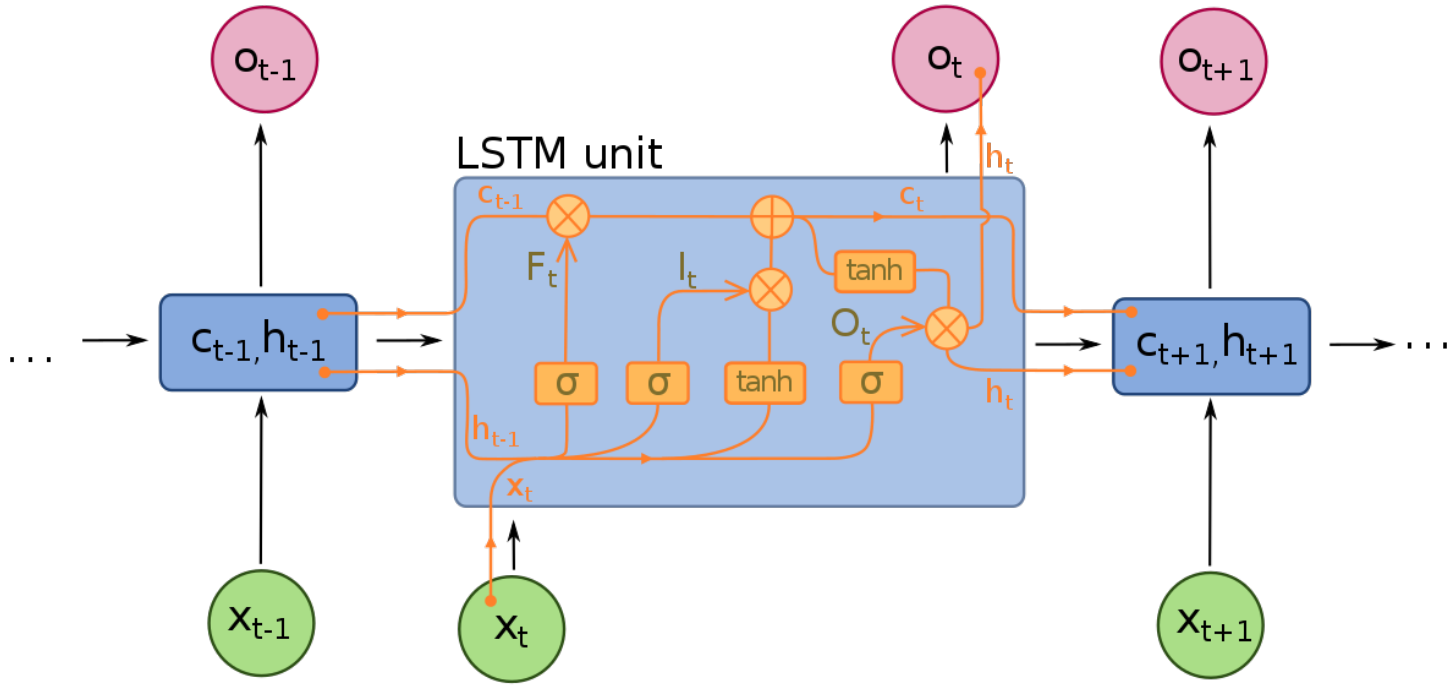
d-vector: Overview

- Acoustic features: 40 Mel filterbank coefficients
- Supervector = 768d state vector of a unidirectional LSTM after the end of the utterance
- Intermediate vector (d-vector) = 256d linear projection of the supervector
- Similarity score: cosine similarity

$$y = \begin{cases} 1 & \frac{\boldsymbol{w}_1^T \boldsymbol{w}_2}{\|\boldsymbol{w}_1\| \|\boldsymbol{w}_2\|} > \theta \\ 0 & \text{otherwise} \end{cases}$$

Supervector = 768d state vector of a unidirectional LSTM

Run the following system over a length-T entire utterance:



https://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg

. . . then $\boldsymbol{h}_T$ is the supervector, and $\boldsymbol{o}_T$ is the 256d d-vector.

End-to-end training

Test speakers are not in the training set! Nevertheless, end-to-end training is possible if your training set has a large enough number of speakers. In that case you can use:

$$\mathcal{L} = \sum_{j=1}^{\#\text{ speakers}} \sum_{i=1}^{\#\text{ utterances}} \log \left(\frac{e^{\cos(\boldsymbol{w}_{j,i}, \bar{\boldsymbol{w}}_j)}}{\sum_{k=1}^{\#\text{ speakers}} e^{\cos(\boldsymbol{w}_{j,i}, \bar{\boldsymbol{w}}_k)}} \right)$$

$$\cos(\boldsymbol{w}_{j,i}, \bar{\boldsymbol{w}}_k) = \frac{\boldsymbol{w}_{j,i}^T \bar{\boldsymbol{w}}_k}{\|\boldsymbol{w}_{j,i}\| \|\bar{\boldsymbol{w}}_k\|},$$

where $\bar{\boldsymbol{w}}_k$ is the average d-vector computed from utterances of speaker k.

End-to-end training

Test speakers are not in the training set! Nevertheless, end-to-end training is possible if your training set has a large enough number of speakers:

- i-vector was trained using 1,748 speakers
- x-vector was trained using 2,600 speakers, but only beat i-vector if it used data augmentation during training
- d-vector was trained using 648,000 speakers

d-vector: Results

Table 1. MultiReader vs. directly mixing multiple data sources.

Test data (Enroll → Verify)	Mixed data EER (%)	MultiReader EER (%)
OK Google → OK Google	1.16	0.82
OK Google → Hey Google	4.47	2.99
Hey Google → OK Google	3.30	2.30
Hey Google → Hey Google	1.69	1.15

Table 2. Text-dependent speaker verification EER.

Model Architecture	Embed Size	Loss	Multi Reader	Average EER (%)
(512,) [13]	128	TE2E	No	3.30
			Yes	2.78
$(128, 64) \times 3$	64	TE2E	No	3.55
			Yes	2.67
$(128, 64) \times 3$	64	GE2E	No	3.10
			Yes	2.38

Outline

Comparison of Systems

System	Supervector	Projection	Similarity	Training Speakers
i-vector	Mean shift of GMM centroids	PCA	cos	1,748
x-vector	Mean and variance of 1500d CNN output	learned	PLDA	2,600
d-vector	Last state of LSTM	learned	cos	648,000