SIDEKIT – Python Toolkit for Speaker Recognition

Presented by:

Kong Aik LEE

Institute for Infocomm Research (I^2R)
A*STAR, Singapore

Overview

- Recap and overview
- Training a TVM using SIDEKIT
  - FeatureExtractor, FeatureServer, StatServer
  - Whitening
  - Minimum divergence (MD) re-estimation
- Probabilistic LDA (PLDA)
  - Tying across frame
  - Simplified PLDA
  - Speaker adaptation
- Interesting research topics

SIDEKIT – Python Toolkit for Speaker Recognition

RECAP AND OVERVIEW

I-vector extraction (1/3)

- **Compression** process – an i-vector is a fixed-length low-dimensional representation of a variable-length speech utterance [Dehak et al, 2011].

- **MAP estimate** – posterior mean of the latent variable $\mathbf{h}$ in a multi-Gaussian factor analysis model (i.e., total variability model)

$$\phi = \arg\max_{\mathbf{h}} \left[\prod_{c=1}^{C} \prod_{t=1}^{N_c} \mathcal{N}\left(o_t \mid \boldsymbol{\mu}_c + \mathbf{W}_c \mathbf{h}, \boldsymbol{\Phi}_c\right) \right] p(\mathbf{h})$$

i-vector

$$\phi = \underbrace{\left(\sum_{c=1}^{C} N_c \mathbf{W}_c^{\mathrm{T}} \boldsymbol{\Phi}_c^{-1} \mathbf{W}_c + I\right)}_{\mathbf{L}^{-1}} \sum_{c=1}^{C} \mathbf{W}_c^{\mathrm{T}} \boldsymbol{\Phi}_c^{-1} \underbrace{\left(\sum_{t=1}^{T} \gamma_t \left(o_t - \boldsymbol{\mu}_c\right)\right)}$$

Zero-order statistics

First-order statistics (centred)

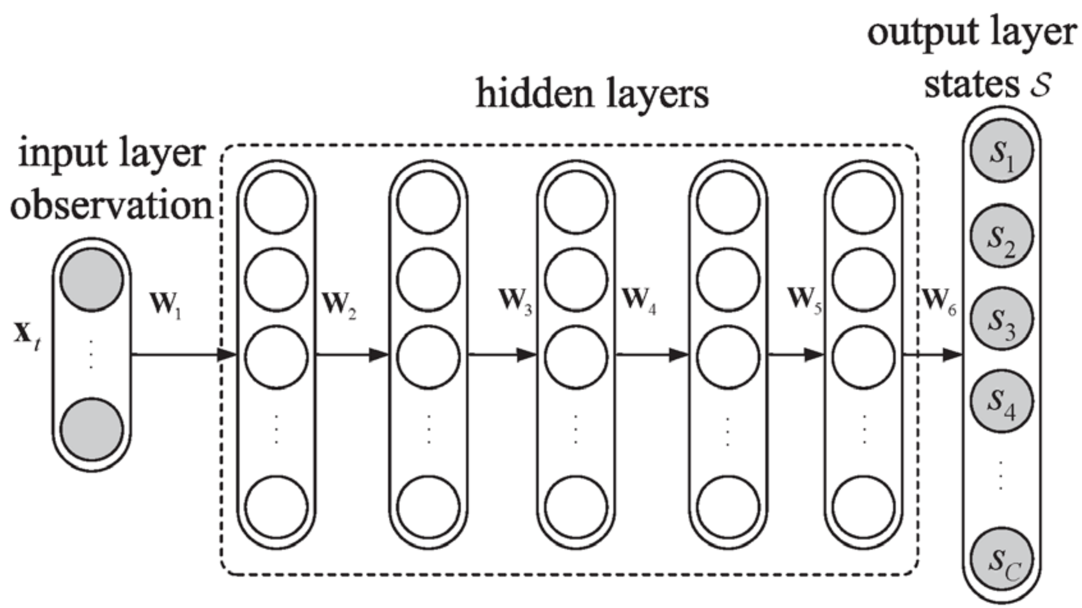
$$N_c = \sum_{t=1}^{T} \underbrace{\gamma_t}(c)$$

$$\gamma_t(c) = \frac{\omega_c \mathcal{N}\left(o_t \mid \boldsymbol{\mu}_c, \boldsymbol{\Phi}_c\right)}{\sum_{j=1}^{C} \omega_j \mathcal{N}\left(o_t \mid \boldsymbol{\mu}_j, \boldsymbol{\Phi}_j\right)}$$

Alignment of frame $\mathbf{o}_t$ to Gaussian c

I-vector extraction (2/3)

- Frame alignment with *deep neural network* (DNN)
  - NN trained for phone state classification was used to align the frames.
  - Each output node of the DNN is trained to estimate the posterior probability of tied-states given the acoustic observations.



I-vector extraction (3/3)

- Vector notation

$$\phi_r = \left(\sum_{c=1}^{C} N_{r,c}\mathbf{W}_c^{\mathrm{T}}\boldsymbol{\Phi}_c^{-1}\mathbf{W}_c + \mathbf{I}\right)^{-1}\left(\sum_{c=1}^{C}\mathbf{W}_c^{\mathrm{T}}\boldsymbol{\Phi}_c^{-1}\sum_{t=1}^{N_{r,c}}\left(o_t - \boldsymbol{\mu}_c\right)\right)$$

$$\phi_r = \underbrace{\left(\mathbf{I} + \mathbf{T}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{N}_r\mathbf{T}\right)^{-1}}_{\mathbf{L}_r^{-1}} \cdot \mathbf{T}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\left(\mathbf{F}_r - \mathbf{N}_r\mathbf{m}_o\right)$$

- $\mathbf{F}_r$ is a $CD \times 1$ vector obtained by stacking the first-order statistics from all components. $\mathbf{N}_r$ is a $CD \times CD$ diagonal matrix consisting of diagonal blocks $N_{r,c} \times \mathbf{I}$

$$\mathbf{F}_r = \begin{bmatrix} \mathbf{F}_r(1) \\ \mathbf{F}_r(2) \\ \vdots \\ \mathbf{F}_r(C) \end{bmatrix} \qquad \mathbf{N}_r = \begin{bmatrix} N_{r,1} \times \mathbf{I} & 0 & \cdots & 0 \\ 0 & N_{r,2} \times \mathbf{I} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & N_{r,C} \times \mathbf{I} \end{bmatrix}$$

Total variability model

- The **total variability matrix $\mathbf{T}$** is obtained by stacking up the loading matrices from each Gaussian:

UBM mean vectors and covariance matrices

Total variability matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_C \end{bmatrix} \qquad \mathbf{m}_o = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_C \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Phi}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{\Phi}_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \boldsymbol{\Phi}_C \end{bmatrix}$$

- Total variability model

$$p(o_t \mid c) = \mathcal{N}\left(o_t \mid \boldsymbol{\mu}_c + \mathbf{W}_c \mathbf{h}_r, \boldsymbol{\Phi}_c\right) \quad \text{for} \quad c = 1, 2, \ldots, C$$

$$\boldsymbol{\mu}_{r,c} = \boldsymbol{\mu}_c + \mathbf{W}_c \mathbf{h}_r \quad \text{for} \quad c = 1, 2, \ldots, C$$

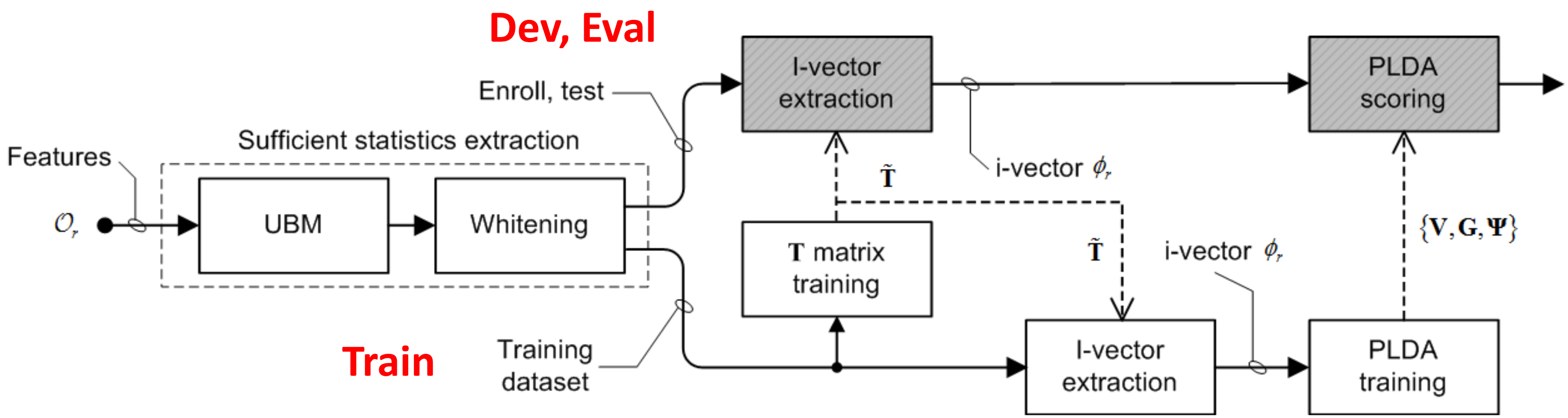
mean supervector

$$\boxed{\mathbf{m}_r = \mathbf{m}_o + \mathbf{T}\mathbf{h}_r}$$

Session index, *r*

I-vector PLDA pipeline

- I-vector extraction followed by PLDA scoring



[K. A. Lee and H. Li, Interspeech 2017]

- PLDA scoring

Test Enroll

$$s(\phi_t, \phi_e) = \frac{p(\phi_t, \phi_e)}{p(\phi_t)\, p(\phi_e)} = \frac{p(\phi_t \mid \phi_e)\, p(\phi_e)}{p(\phi_t)\, p(\phi_e)} = \frac{p(\phi_t \mid \phi_e)}{p(\phi_t)}$$

SIDEKIT – Python Toolkit for Speaker Recognition

TRAINING A TVM USING SIDEKIT

SIDEKIT

- End-to-end python implementation from feature extraction, scoring, calibration (Python BOSARIS)
- Main classes: FeatureExtractor, FeatureServer, StatServer
- Reference:

  A. Larcher, K. A. Lee, and S. Meignier, "An extensible speaker identificaition SIDEKIT in Python," in *Proc. IEEE ICASSP*, 2016, pp. 5095 – 5099.

- Tutorial page:

  http://www-lium.univ-lemans.fr/sidekit/overview/index.html

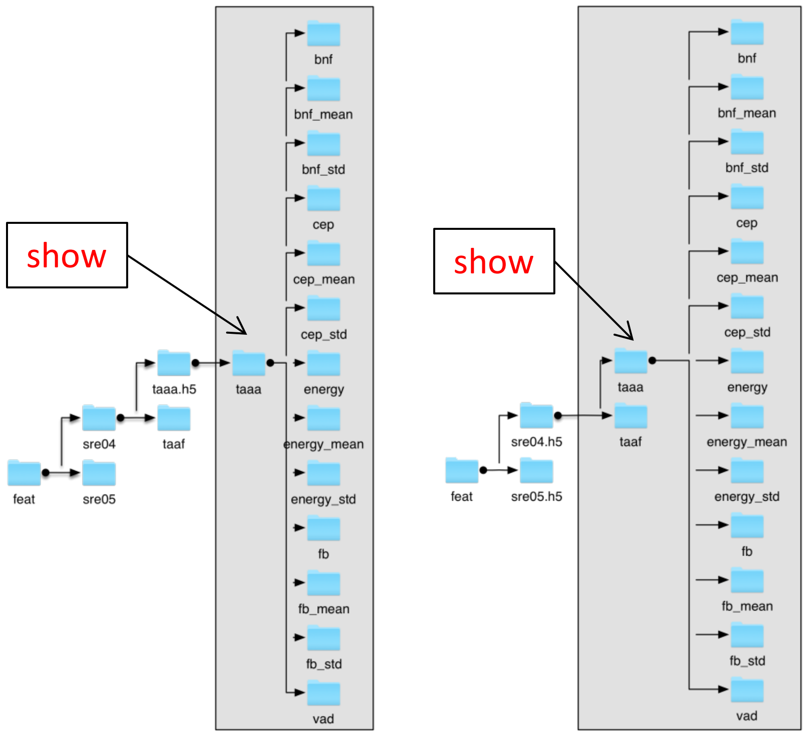
- PIP (with Anaconda and virtual environment)

```
$ pip install sidekit
```

Feature Extractor (1/2)

- `FeatureExtractor` – a Python class providing functionalities to
  - Extract features (`MFCC`, `LFCC`) from audio files (`sph`, `wav`, `raw`)
  - Energy-based voice activity detection (VAD)
  - Features (e.g., MFCC, VAD, log energy etc.) are stored as datasets in hdf5 format

- Hierarchical architecture of hdf5 allows
  - One hdf5 feature file per audio recording
  - One hdf5 feature file for a collection of audio recordings



Feature Extractor (2/2)

- Example: perform feature extraction given a list

```
print("Create Feature extractor")
extractor = sidekit.FeaturesExtractor(audio_filename_structure=None,
                                      feature_filename_structure=None,
                                      sampling_frequency=None,
                                      lower_frequency=200,
                                      higher_frequency=3800,
                                      filter_bank="log",
                                      filter_bank_size=24,
                                      window_size=0.025,
                                      shift=0.01,
                                      ceps_number=20,
                                      vad="snr",
                                      snr=40,
                                      pre_emphasis=0.97,
                                      save_param=["vad", "energy", "cep", "fb"],
                                      keep_all_features=True)
```

Initiate a FeatureExtractor with **attributes** given as arguments

```
print("Start extracting features")
extractor.save_list(show_list=show_list,
                    channel_list=channel_list,
                    audio_file_list=audio_list,
                    feature_file_list=feature_list,
                    num_thread=nbThread)
```

Extract feature from channel a or b from a audio file and save all features to a hdf5 file

```
show_list[0] => 'sre04/tghy'
channel_list[0] => 0
audio_list[0] => '/media/kalee/SRE04/train/data/tghy.sph'
feature_list[0] => '/disk1/ProjectPy/sidekit3/mfcc_24/sre04/tghy.h5'
```

Feature Server (1/2)

- `FeatureServer` – a Python class providing functionality to load features from files and post-process the features:
  - Normalization (CMS, CMVN, RASTA, Short-Term Gaussianization)
  - Temporal context (delta, delta-delta)
  - Feature selection
- Initiate a FeatureServer

Load and concatenate ceptral coefficients (first 13 from 0 to 12) and log energy

```
# Define a FeaturesServer for loading the acoustic features
fs = sidekit.FeaturesServer(feature_filename_structure="{dir}/{{}}.{ext}".format(dir=
feature_dir, ext=feature_extension),
                            dataset_list=["energy", "cep", "vad"],
                            mask="[0-12]",
                            feat_norm="cmvn",
                            keep_all_features=False,
                            delta=True,
                            double_delta=True,
                            rasta=True,
                            context=None)
```

'h5'

Perform RASTA, append delta and delta-delta

Remove silence frames based on the VAD labels

Feature Server (2/2)

- Example: Using FeatureServer to load features for UBM training

```
# Define a FeaturesServer for loading the acoustic features
fs = sidekit.FeaturesServer(feature_filename_structure="{dir}/{{}}.{ext}".format(dir=
```

```
# Train UBM using EM algorithm and write it to disk. Save current model
# to disk at the end of each EM iteration
print('Train the UBM by EM')
nbThread = min(max(multiprocessing.cpu_count()-1, 1), nbThread_max)
ubm = sidekit.Mixture()
llk = ubm.EM_split(fs, ubm_list, distrib_nb, num_thread=nbThread, save_partial='{}/ubm'.
format(ubm_dir))
```

List of feature files to train UBM

```
sre06/noil_b
sre06/pbgy_a
sre05/jbwz_a
      .
      .
```

Feature filename and the dataset name in the h5 file.

Stat Server (1/2)

- `StatServer` – a Python class providing functionality to extract, store, process, train total variability model (or PLDA) and extract i-vector.

- Extract sufficient statistics

```
back_stat = sidekit.StatServer(back_idmap, ubm)
back_stat.accumulate_stat(ubm=ubm, feature_server=fs, seg_indices=range(back_stat.segset.shape[0]), num_thre
back_stat.write('data/stat_back_{}.h5'.format(distrib_nb))
```

- Train TVM (with sufficient statistics whitening and MD re-estimation)

```
tv_stat = sidekit.StatServer.read_subset('data/stat_back_{}.h5'.format(distrib_nb), tv_idmap)
tv_mean, tv, _, __, tv_sigma = tv_stat.factor_analysis(rank_f = rank_TV,
                                                        rank_g = 0,
                                                        rank_h = None,
                                                        re_estimate_residual = False,
                                                        it_nb = (tv_iteration,0,0),
                                                        min_div = True,
                                                        ubm = ubm,
                                                        batch_size = 100,
                                                        num_thread = nbThread,
                                                        save_partial = "data/TV_{}".format(distrib_nb))
sidekit.sidekit_io.write_tv_hdf5((tv, tv_mean, tv_sigma), "data/TV_{}".format(distrib_nb))
```

Stat Server (2/2)

- Extract sufficient statistics

```
enroll_stat = sidekit.StatServer(enroll_idmap, ubm)
enroll_stat.accumulate_stat(ubm=ubm, feature_server=fs, seg_indices=range(enroll_stat.segset.shape[0]) ,num_
enroll_stat.write('data/stat_sre10_core-core_enroll_{}.h5'.format(distrib_nb))

test_stat = sidekit.StatServer(test_idmap, ubm)
test_stat.accumulate_stat(ubm=ubm, feature_server=fs, seg_indices=range(test_stat.segset.shape[0]), num_thre
test_stat.write('data/stat_sre10_core-core_test_{}.h5'.format(distrib_nb))

back_idmap = plda_all_idmap.merge(tv_idmap)
back_stat = sidekit.StatServer(back_idmap, ubm)
back_stat.accumulate_stat(ubm=ubm, feature_server=fs, seg_indices=range(back_stat.segset.shape[0]), num_thre
back_stat.write('data/stat_back_{}.h5'.format(distrib_nb))
```

- Extract i-vector for Train, Enroll and Test i-vectors

```
enroll_stat = sidekit.StatServer('data/stat_sre10_core-core_enroll_{}.h5'.format(distrib_nb))
enroll_iv = enroll_stat.estimate_hidden(tv_mean, tv_sigma, V=tv, batch_size=100, num_thread=nbThread)[0]
enroll_iv.write('data/iv_sre10_core-core_enroll_{}.h5'.format(distrib_nb))

test_stat = sidekit.StatServer('data/stat_sre10_core-core_test_{}.h5'.format(distrib_nb))
test_iv = test_stat.estimate_hidden(tv_mean, tv_sigma, V=tv, batch_size=100, num_thread=nbThread)[0]
test_iv.write('data/iv_sre10_core-core_test_{}.h5'.format(distrib_nb))

plda_stat = sidekit.StatServer.read_subset('data/stat_back_{}.h5'.format(distrib_nb), plda_all_idmap)
plda_iv = plda_stat.estimate_hidden(tv_mean, tv_sigma, V=tv, batch_size=100, num_thread=nbThread)[0]
plda_iv.write('data/iv_plda_{}.h5'.format(distrib_nb))
```

Whitening (1/5)

- Whitening or sphering [Bishop, 2006] refers to the normalization of data, where $\boldsymbol{\mu}$ and $\boldsymbol{\Phi}$ are sample mean and covariance matrix estimated from data:

$$\tilde{o}_t = \boldsymbol{\Phi}^{-1/2}(o_t - \boldsymbol{\mu})$$

$$\boldsymbol{\mu} = \frac{\mathbf{1}}{\boldsymbol{T}}\sum_{t=1}^{T} o_t, \quad \boldsymbol{\Phi} = \frac{\mathbf{1}}{\boldsymbol{T}}\sum_{t=1}^{T}(o_t - \boldsymbol{\mu})(o_t - \boldsymbol{\mu})^{\mathrm{T}}$$

- The matrix $\boldsymbol{\Phi}^{-1/2}$ is obtained by decomposing $\boldsymbol{\Phi}$ to two parts, for instance, with Cholesky decomposition or eigenvalue decomposition. In both cases, a better choice is to decompose $\boldsymbol{\Phi}$ and then followed by inversion
  - Cholesky decomposition:

$$\boldsymbol{\Phi}^{-\mathbf{1}} = \left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^{-\mathbf{1}} = \left(\mathbf{L}^{\mathrm{T}}\right)^{-\mathbf{1}}\left(\mathbf{L}^{-\mathbf{1}}\right) = \left(\mathbf{L}^{-\mathbf{1}}\right)^{\mathrm{T}}\boxed{\left(\mathbf{L}^{-\mathbf{1}}\right)}$$

$$\tilde{o}_t = \mathbf{L}^{-\mathbf{1}}(o_t - \boldsymbol{\mu})$$

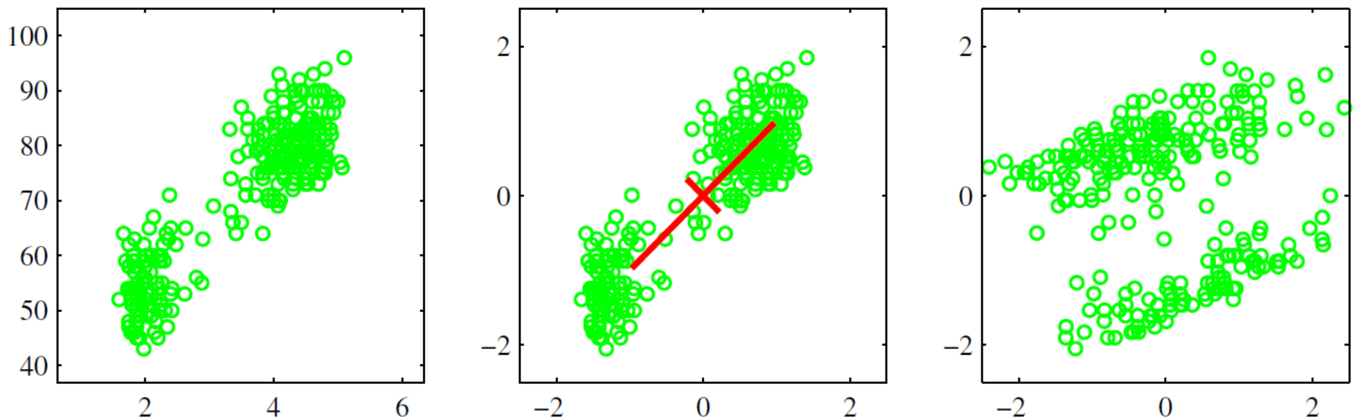
Whitening (2/5)

– Eigenvalue decomposition:

$$\mathbf{\Phi}^{-1} = \left(\mathbf{Q\Lambda Q}^{\mathrm{T}}\right)^{-1} = \mathbf{Q\Lambda}^{-1}\mathbf{Q}^{\mathrm{T}} = \mathbf{Q\Lambda}^{-1/2}\mathbf{\Lambda}^{-1/2}\mathbf{Q}^{\mathrm{T}} = \left(\mathbf{Q\Lambda}^{-1/2}\right)\left(\mathbf{Q\Lambda}^{-1/2}\right)^{\mathrm{T}}$$

$$\tilde{o}_t = \left(\mathbf{Q\Lambda}^{-1/2}\right)^{\mathrm{T}}(o_t - \boldsymbol{\mu}) = \mathbf{\Lambda}^{-1/2}\mathbf{Q}^{\mathrm{T}}(o_t - \boldsymbol{\mu})$$

- Whitening leads to zero mean and unit covariance distribution



The effects of whitening: original data (left), whitening with diagonal covariance matrix (middle), whitening with full covariance matrix (right) [Bishop, 2006]

Whitening (3/5)

- We aim to whiten the acoustic observations with respect to the UBM (multiple Gaussians). This is accomplished by subjecting the first-order statistics to the following transformation.
  - Soft-alignment of frames to component

$$\mathbf{F}_r(c) = \sum_t \gamma_t(c) o_t$$

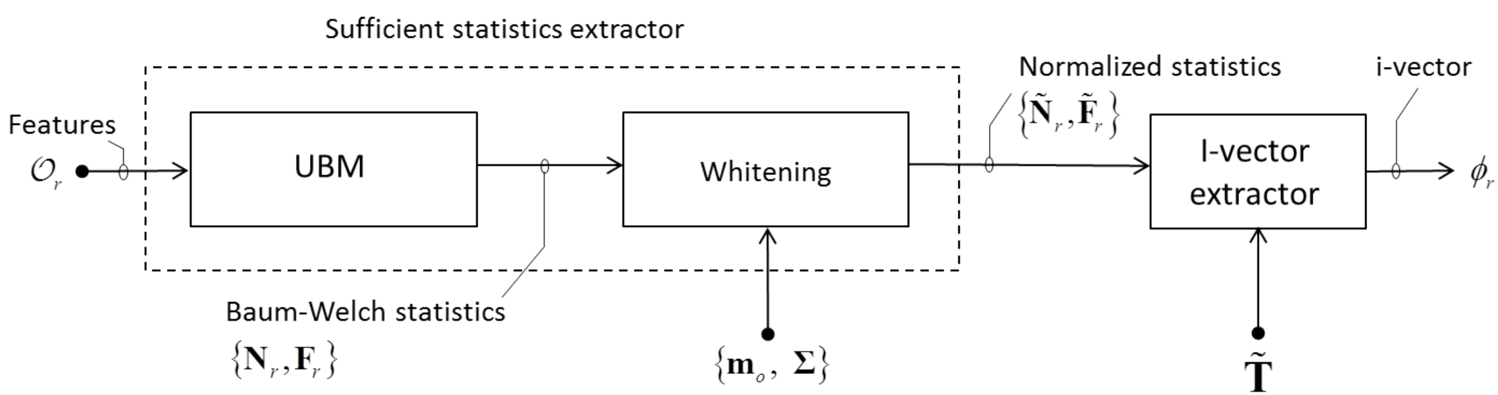
  - Subjecting the first-order statistics to the following transformation

$$\tilde{\mathbf{F}}_r(c) = \boldsymbol{\Phi}_c^{-1/2} \sum_t \gamma_t(c)(o_t - \boldsymbol{\mu}_c) \text{ for } c = 1,2,\dots,C$$

$$\tilde{\mathbf{F}}_r = \boldsymbol{\Sigma}^{-1/2}(\mathbf{F}_r - \mathbf{N}_r \mathbf{m}_o)$$

Whitening (4/5)

- UBM parameters $\{\mathbf{m}_o, \mathbf{\Sigma}\}$ are absorbed as part of sufficient statistics and therefore no longer required in subsequent processing (as shown in the block diagram).
- The implementation becomes the same for diagonal or full covariance matrices.
- Strictly speaking, an i-vector extractor consists both sufficient statistics extraction and i-vector inference.



Whitening (5/5)

- Using the whitened sufficient statistics for training, we obtained a whitened total variability matrix $\tilde{\mathbf{T}}$ (i.e., whitening the sufficient statistics has similar effects in whitening the $\mathbf{T}$ matrix). The original $\mathbf{T}$ matrix could be recovered by inversing the transformation

$$\tilde{\mathbf{T}} = \mathbf{\Sigma}^{-1/2}\mathbf{T} \quad \Rightarrow \quad \mathbf{T} = \mathbf{\Sigma}^{1/2}\tilde{\mathbf{T}}$$

- The i-vectors (more generally the posterior distribution of the latent variable) remains the same

$$\begin{aligned}
\phi_r &= \left(\mathbf{I} + \mathbf{T}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{N}_r\mathbf{T}\right)^{-1} \cdot \mathbf{T}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\left(\mathbf{F}_r - \mathbf{N}_r\mathbf{m}_o\right) \\
&= \left(\mathbf{I} + \mathbf{T}^{\mathrm{T}}\mathbf{\Sigma}^{-1/2}\mathbf{\Sigma}^{-1/2}\mathbf{N}_r\mathbf{T}\right)^{-1} \cdot \mathbf{T}^{\mathrm{T}}\mathbf{\Sigma}^{-1/2}\mathbf{\Sigma}^{-1/2}\left(\mathbf{F}_r - \mathbf{N}_r\mathbf{m}_o\right) \\
&= \left[\mathbf{I} + \left(\mathbf{\Sigma}^{-1/2}\mathbf{T}\right)^{\mathrm{T}}\mathbf{N}_r\left(\mathbf{\Sigma}^{-1/2}\mathbf{T}\right)\right]^{-1} \cdot \left(\mathbf{\Sigma}^{-1/2}\mathbf{T}\right)^{\mathrm{T}}\mathbf{\Sigma}^{-1/2}\left(\mathbf{F}_r - \mathbf{N}_r\mathbf{m}_o\right) \\
&= \underbrace{\left[\mathbf{I} + \tilde{\mathbf{T}}^{\mathrm{T}}\mathbf{N}_r\tilde{\mathbf{T}}\right]^{-1}}_{\mathbf{L}_r^{-1}} \cdot \tilde{\mathbf{T}}^{\mathrm{T}}\tilde{\mathbf{F}}_r
\end{aligned}$$

Minimum divergence re-estimation (1/3)

- The objective of minimum divergence re-estimation is to scale the $\mathbf{T}$ matrix (and shift the global mean vector $\mathbf{m}_o$) in order to force the empirical distribution of the i-vectors conform to the standard Gaussian prior.

- Consider a TVM with single Gaussian and single frame for the purpose of illustration

$$
\begin{aligned}
p(o) &= \int \mathcal{N}\left(o \mid \mathbf{m}_o + \mathbf{Th}, \mathbf{\Phi}\right) \mathcal{N}\left(\mathbf{h} \mid \boldsymbol{\mu}_h, \mathbf{\Sigma}_h\right) d\mathbf{h} \\
&= \mathcal{N}\left(o \mid \mathbf{m}_o + \mathbf{T}\boldsymbol{\mu}_h, \mathbf{T}\mathbf{\Sigma}_h\mathbf{T}^{\mathrm{T}} + \mathbf{\Phi}\right) \\
&= \mathcal{N}\left(o \mid \mathbf{m}_o + \mathbf{T}\boldsymbol{\mu}_h, \mathbf{T}\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)\mathbf{T}^{\mathrm{T}} + \mathbf{\Phi}\right) \\
&= \mathcal{N}\left(o \mid \mathbf{m}_o + \mathbf{T}\left(\mathbf{L}\mathbf{L}^{-1}\right)\boldsymbol{\mu}_h, \left(\mathbf{TL}\right)\left(\mathbf{TL}\right)^{\mathrm{T}} + \mathbf{\Phi}\right) \\
&= \mathcal{N}\left(o \mid \mathbf{m}_o + \widetilde{\mathbf{T}}\mathbf{L}^{-1}\boldsymbol{\mu}_h, \widetilde{\mathbf{T}}\widetilde{\mathbf{T}}^{\mathrm{T}} + \mathbf{\Phi}\right) \\
&= \int \mathcal{N}\left(o \mid \mathbf{m}_o + \widetilde{\mathbf{T}}\mathbf{L}^{-1}\boldsymbol{\mu}_h + \widetilde{\mathbf{T}}\mathbf{h}, \mathbf{\Phi}\right) \mathcal{N}\left(\mathbf{h} \mid 0, \mathbf{I}\right) d\mathbf{h}
\end{aligned}
$$

Minimum divergence re-estimation (2/3)

- MD re-estimation of $\mathbf{T}$ matrix and global mean vector $\mathbf{m}_o$

$$\widetilde{\mathbf{T}} = \mathbf{T}\mathbf{L}$$

$$\tilde{\mathbf{m}}_o = \mathbf{m}_o + \widetilde{\mathbf{T}}\mathbf{L}^{-1}\boldsymbol{\mu}_h = \mathbf{m}_o + \mathbf{T}\boldsymbol{\mu}_h$$

- In general, most implementations rotate and scale only the $\mathbf{T}$ matrix

$$\begin{aligned} p(o) &= \int \mathcal{N}\left(o \mid \mathbf{m}_o + \mathbf{T}\mathbf{h}, \boldsymbol{\Phi}\right) \mathcal{N}\left(\mathbf{h} \mid \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h\right) d\mathbf{h} \\ &= \mathcal{N}\left(o \mid \mathbf{m}_o + \widetilde{\mathbf{T}}\mathbf{L}^{-1}\boldsymbol{\mu}_h, \widetilde{\mathbf{T}}\widetilde{\mathbf{T}}^{\mathrm{T}} + \boldsymbol{\Phi}\right) \\ &= \int \mathcal{N}\left(o \mid \mathbf{m}_o + \widetilde{\mathbf{T}}\mathbf{h}, \boldsymbol{\Phi}\right) \mathcal{N}\left(\mathbf{h} \mid \mathbf{L}^{-1}\boldsymbol{\mu}_h, \mathbf{I}\right) d\mathbf{h} \end{aligned}$$

- MD re-estimation is applied after each EM step. Its role is to get good estimate of the eigenvalues corresponding to the eigenvectors defining the total variability space [Kenny, 2008]

Minimum divergence re-estimation (3/3)

- The empirical distribution $N(\mathbf{h}|\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$ is estimated from i-vectors minimizing the KL divergence criterion, i.e., to find the Gaussian distribution $N(\mathbf{h}|\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$ with minimum KL distances from all the *R* posterior distributions

$$D(\theta_{\mathrm{MD}}) = \sum_{i=1}^{I} E\left\{\log \frac{\mathcal{N}\left(\mathbf{h} \mid \phi_i, \mathbf{L}_i^{-1}\right)}{\mathcal{N}\left(\mathbf{h} \mid \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h\right)}\right\}$$

- Closed-form solution

$$\boldsymbol{\mu}_h = \frac{1}{R}\sum_{r=1}^{R}\phi_r$$

$$\boldsymbol{\Sigma}_h = \left(\frac{1}{R}\sum_{r=1}^{R} L_r^{-1} + \phi_r\phi_r^{\mathrm{T}}\right) - \boldsymbol{\mu}_h\boldsymbol{\mu}_h^{\mathrm{T}}$$

Appendix – MD estimate of prior (1/4)

number of sessions / posterior distributions

$$D(\theta) = \sum_{r=1}^{R} D_{KL}(\mathcal{N}_r \,|\, \mathcal{N}_0)$$

μ_h Σ_h

$$= \sum_{r=1}^{R} E\left\{\log \frac{\mathcal{N}(h|m_r, L_r^{-1})}{\mathcal{N}(h|y, P^{-1})}\right\}, \quad \theta = \{y, P^{-1}\}$$

$$= \sum_{r=1}^{R} \frac{1}{2}\Big[\mathrm{tr}(P L_r^{-1}) + (y-m_r)^T P (y-m_r) - k - \log|L_r^{-1}| + \log|P^{-1}|\Big]$$

Dimensionality of variables

$$= \sum_{r=1}^{R} \frac{1}{2}\Big[\mathrm{tr}(P L_r^{-1}) + (y-m_r)^T P (y-m_r) + \log|P^{-1}| \underbrace{- k - \log|L_r^{-1}|}_{\text{constant for a given dimensionality and dataset}}\Big]$$

Appendix – MD estimate of prior (2/4)

$$D(\theta) = \frac{1}{2}\Big[\sum_{r=1}^{R} tr(L_r^{-1}P) + \underbrace{(y-m_r)^T P (y-m_r)}\Big] \quad \text{—— (1)}$$

$$-\frac{1}{2} R \log|P| + C$$

$$tr\left((y-m_r)(y-m_r)^T P\right)$$

Differentiate (1) wrt y

$$\frac{\partial D(\theta)}{\partial y} = \sum_{r=1}^{R} (y-m_r)^T P \quad \text{—— (3)}$$

Setting (3) to zero

$$\sum_{r=1}^{R} (y-m_r)^T P = 0$$

$$\sum_{r=1}^{R} (y-m_r) = 0$$

$$Ry - \sum_{r=1}^{R} m_r = 0$$

$$y = \frac{1}{R}\sum_{r=1}^{R} m_r \quad \text{—— (4)}$$

Appendix – MD estimate of prior (3/4)

$$D(\theta) = \frac{1}{2}\left[tr(SP) - R \log |P| \right] + C \quad \text{—— (2)}$$

$$S = \sum_{r=1}^{R} L_r^{-1} + (y - m_r)(y - m_r)^T$$

Differentiate (2) w.r.t P

$$\frac{\partial D(\theta)}{\partial P} = \frac{1}{2}(S - RP^{-1}) \quad \text{—— (5)}$$

setting (5) to zero

$$S = RP^{-1}$$

$$P^{-1} = \frac{1}{R} S = \frac{1}{R} \sum_{r=1}^{R} L_r^{-1} + (y - m_r)(y - m_r)^T$$

$$= L^{-1} + \frac{1}{R}\sum_{r=1}^{R} (y - m_r)(y - m_r)^T$$

Appendix – MD estimate of prior (4/4)

$$P^{-1} = \frac{1}{R}\sum_{r=1}^{R} L_r^{-1} + \overbrace{(y - m_r)(y - m_r)^T}^{\text{Emperical term}}$$

$$= \frac{1}{R}\sum_{r=1}^{R} L_r^{-1} + y y^T + m_r m_r^T \quad 2y\, m_r^T$$

$$= \frac{1}{R}\sum_{r=1}^{R} L_r^{-1} + m_r m_r^T + \frac{1}{R}\sum_{r=1}^{R}\left(y y^T - 2y m_r^T\right)$$

$$= \frac{1}{R}\sum_{r=1}^{R} L_r^{-1} + m_r m_r^T + y y^T - 2y y^T$$

$$= \left(\frac{1}{R}\sum_{r=1}^{R} L_r^{-1} + m_r m_r^T\right) - \underbrace{y y^T}_{\mu_h \mu_h^T}$$

SIDEKIT – Python Toolkit for Speaker Recognition

PROBABILISTIC LDA (PLDA)

Channel compensation

- I-vector extraction is a **compression** process, where we compress across the time (variable-length sequence to **fixed-length** vector) and supervector space (high to **low** dimensionality).

- An i-vector contains
  - Speaker/language characteristic
  - other information originated from the transmission channel, recording devices, and acoustic environment (**channel** or **session** variability)

- Channel (or session) compensation – to deal with the **mismatch** between enrolment and test utterances.

- Main stream compensation techniques include (implemented in SIDEKIT)
  - Linear discriminant analysis (LDA) [Bishop, 2006]
  - Within-class covariance normalization [Hatch et al, 2006]
  - **Probabilistic LDA** (PLDA) [Prince and Elder, 2007]

I-vector/PLDA using SIDEKIT

- PLDA scoring

```
# Normalize i-vector using Spherical Nuisance Normalization (similar to
length-normalization) and compute scores using PLDA

meanSN, CovSN = plda_iv.estimate_spectral_norm_stat1(1, 'sphNorm')
plda_iv.spectral_norm_stat1(meanSN[:1], CovSN[:1])
enroll_iv.spectral_norm_stat1(meanSN[:1], CovSN[:1])
test_iv.spectral_norm_stat1(meanSN[:1], CovSN[:1])

plda_mean, plda_F, plda_G, plda_H, plda_Sigma = plda_iv.factor_analysis(rank_f=plda_rk,
                                                                      rank_g=0,
                                                                      rank_h=None,
                                                                      re_estimate_residual=
                                                                      True,
                                                                      it_nb=(10,0,0),
                                                                      min_div=True,
                                                                      ubm=None,
                                                                      batch_size=1000,
                                                                      num_thread=nbThread)

scores_plda = sidekit.iv_scoring.PLDA_scoring(enroll_iv, test_iv, test_ndx, plda_mean, plda_F
, plda_G, plda_Sigma, full_model=False)
```

Whitening + unit norm

NOT used in simplified PLDA

Channel variability is model with full residual covariance matrix for simplified PLDA

Cosine scoring

- LDA followed by cosine scoring

```
# LDA (reduce the dimension of i-vectors to 150 dimensions) followed by cosine scoring.
LDA = plda_iv.get_lda_matrix_stat1(150)
plda_iv.rotate_stat1(LDA)
enroll_iv.rotate_stat1(LDA)
test_iv.rotate_stat1(LDA)
scores_cos_lda = sidekit.iv_scoring.cosine_scoring(enroll_iv, test_iv, test_ndx, wccn=None)
```

- WCCN followed by cosine scoring

```
# WCCN followed by cosine scoring.
wccn = plda_iv.get_wccn_choleski_stat1()
scores_cos_wccn = sidekit.iv_scoring.cosine_scoring(enroll_iv, test_iv, test_ndx, wccn=wccn)
```

PLDA definition

- PLDA explains the observed data (i.e., i-vectors) in terms of **speaker** identity and **channel** effects.

- Let ϕ_{ij} be an i-vector extracted from the j-th session of speaker i, the generative explanation for the i-vector is

Number of speakers

Number of sessions per speaker

$$\boxed{\phi_{ij} = \boldsymbol{\mu} + \mathbf{F}\mathbf{h}_i + \mathbf{G}\mathbf{w}_{ij} + \boldsymbol{\varepsilon}_{ij} \text{ for } i = 1, 2, \ldots, I \text{ and } j = 1, 2, \ldots, J}$$

  – The **speaker** variable $\mathbf{h}_i$ quantifies the observed deviations from the mean $\boldsymbol{\mu}$ due to the changes of speaker.

  – The **channel** variables $\mathbf{w}_{ij}$ quantifies the observed deviations from the mean $\boldsymbol{\mu}$ due to the different sessions of the same speaker

  – The **residual** noise term $\boldsymbol{\varepsilon}_{ij} \sim N(0, \boldsymbol{\Phi})$ described the residual variation (note: this corresponds to the UBM covariance matrices in the total variability model)

Probabilistic representation

- In probabilistic term, PLDA is expressed as

$$p\left(\phi_{ij} \mid \mathbf{h}_i, \mathbf{w}_{ij}\right) = \mathcal{N}\left(\phi_{ij} \mid \boldsymbol{\mu} + \mathbf{F}\mathbf{h}_i + \mathbf{G}\mathbf{w}_{ij}, \boldsymbol{\Phi}\right)$$

Conditional distribution

$$\text{Priors} \left\{ \begin{aligned} p\left(\mathbf{h}_i\right) &= \mathcal{N}\left(\mathbf{h}_i \mid 0, \mathbf{I}\right) \\ p\left(\mathbf{w}_{ij}\right) &= \mathcal{N}\left(\mathbf{w}_{ij} \mid 0, \mathbf{I}\right) \end{aligned} \right.$$

- Given the **prior** and **conditional** distributions, we integrate out the speaker and channel latent variables to obtain the **marginal** distribution.

$$\begin{aligned} p\left(\phi_{ij}\right) &= \int p\left(\phi_{ij} \mid \mathbf{h}_i, \mathbf{w}_{ij}\right) p\left(\mathbf{h}_i\right) p\left(\mathbf{w}_{ij}\right) d\mathbf{h}_i \mathbf{w}_{ij} \\ &= \mathcal{N}\left(\phi_{ij} \mid \boldsymbol{\mu}, \boxed{\mathbf{F}\mathbf{F}^{\mathrm{T}} + \mathbf{G}\mathbf{G}^{\mathrm{T}} + \boldsymbol{\Phi}}\right) \end{aligned}$$

Structured covariance matrix

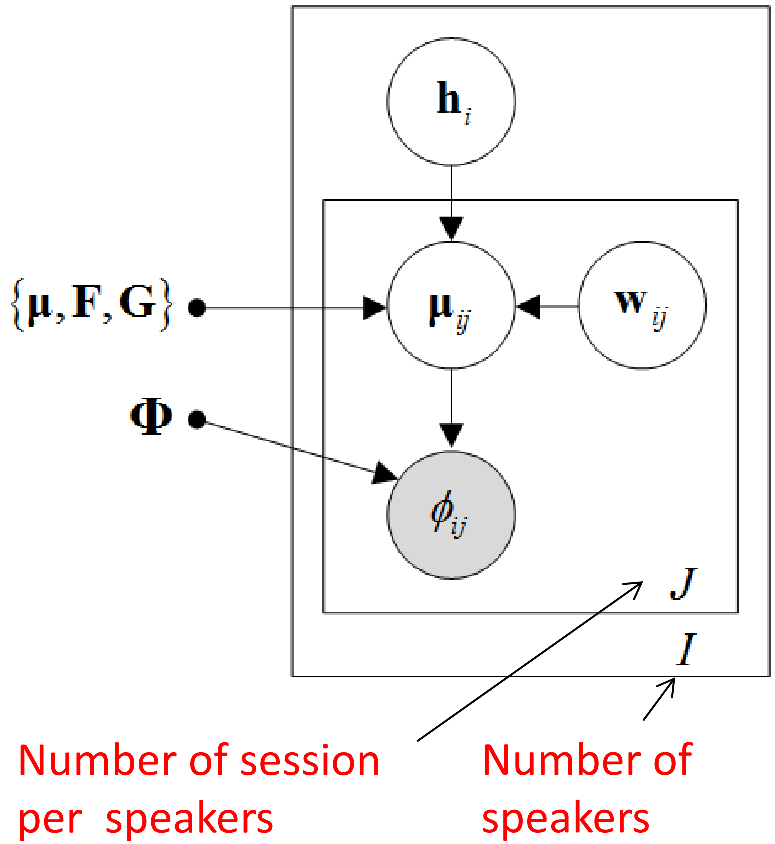
Between speaker variation

Within speaker variation

- PLDA is a Gaussian density with **structured covariance matrix**, similar to factor analysis, with additional subspaces $\mathbf{F}$ and $\mathbf{G}$.

Graphical model representation

- A PLDA is a **factor analysis** model with additional **tying across observations**
  - I-vectors are the observed variables
  - I-vectors of the same speaker share the speaker variable $\mathbf{h}_i$ (i.e., tying of latent variable across i-vectors of the same speaker)
  - Each i-vector is characterized by a channel variable $\mathbf{w}_{ij}$
  - Separate speaker and channel subspaces $\{\mathbf{F}, \mathbf{G}\}$ to tease apart the contributions of speaker and channel information (TVM has one single subspace)



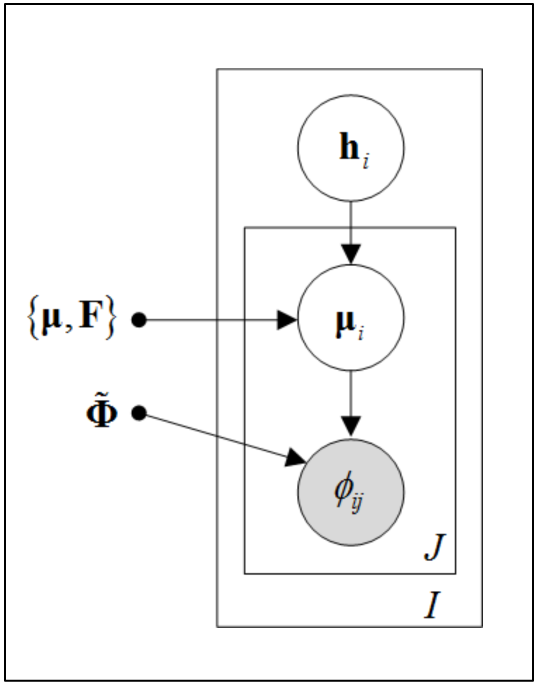
Simplified PLDA

- The general PLDA is a model for Gaussian distribution with a structured covariance matrix

$$\boxed{\phi_{ij} = \boldsymbol{\mu} + \mathbf{F}\mathbf{h}_i + \mathbf{G}\mathbf{w}_{ij} + \boldsymbol{\varepsilon}_{ij} \text{ where } \boldsymbol{\varepsilon}_{ij} \sim \mathcal{N}(0, \boldsymbol{\Phi})}$$

$$p(\phi_{ij} \mid \mathbf{h}_i, \mathbf{w}_{ij}) = \mathcal{N}(\phi_{ij} \mid \boldsymbol{\mu} + \mathbf{F}\mathbf{h}_i + \mathbf{G}\mathbf{w}_{ij}, \boldsymbol{\Phi})$$

$$p(\phi_{ij}) = \mathcal{N}(\phi_{ij} \mid \boldsymbol{\mu}, \mathbf{F}\mathbf{F}^{\mathrm{T}} + \mathbf{G}\mathbf{G}^{\mathrm{T}} + \boldsymbol{\Phi})$$



- The simplified PLDA is obtained by integrate out $\mathbf{w}$

$$\boxed{\phi_{ij} = \boldsymbol{\mu} + \mathbf{F}\mathbf{h}_i + \tilde{\boldsymbol{\varepsilon}}_{ij} \text{ where } \tilde{\boldsymbol{\varepsilon}}_{ij} \sim \mathcal{N}(0, \tilde{\boldsymbol{\Phi}})}$$

$$p(\phi_{ij} \mid \mathbf{h}_i) = \mathcal{N}(\phi_{ij} \mid \boldsymbol{\mu} + \mathbf{F}\mathbf{h}_i, \tilde{\boldsymbol{\Phi}})$$

$$p(\phi_{ij}) = \mathcal{N}(\phi_{ij} \mid \boldsymbol{\mu}, \mathbf{F}\mathbf{F}^{\mathrm{T}} + \boxed{\underbrace{\mathbf{G}\mathbf{G}^{\mathrm{T}} + \boldsymbol{\Phi}}_{\tilde{\boldsymbol{\Phi}}}})$$

Full covariance matrix

Posterior inference and EM

- Given all the J i-vectors from speaker i, the posterior distribution of $\mathbf{h}_i$ is a normal distribution

$$p\left(\mathbf{h}_i \mid \phi_{i,1}, \phi_{i,2}, \ldots, \phi_{i,J}\right) = \mathcal{N}\left(\mathbf{h}_i \mid \mathbf{m}_h, \mathbf{L}_h^{-1}\right)$$

Posterior mean

Posterior covariance

$$\mathbf{m}_h = \underbrace{\left(J\mathbf{F}^{\mathrm{T}}\tilde{\mathbf{\Phi}}^{-1}\mathbf{F} + I\right)^{-1}}_{\mathbf{L}_h^{-1}} \cdot \mathbf{F}^{\mathrm{T}}\tilde{\mathbf{\Phi}}^{-1}\left(\sum_{j=1}^{J}\phi_{ij} - \boldsymbol{\mu}\right)$$

- The parameters of the simplified PLDA model $\theta = \{\mathbf{F}, \tilde{\mathbf{\Phi}}\}$ are learned via EM algorithm
  - E-step computes the posterior distribution for each speaker variables $\mathbf{h}_i$
  - M-step updates the using the relations

Zero-order statistics

$$\mathbf{F} = \left(\sum_{i=1}^{I} \underbrace{\sum_{j=1}^{J}\left(\phi_{ij} - \boldsymbol{\mu}\right)}\mathrm{E}\left[\mathbf{h}_i^{\mathrm{T}}\right]\right)\left(\sum_{i=1}^{I} J\mathrm{E}\left[\mathbf{h}_i\mathbf{h}_i^{\mathrm{T}}\right]\right)^{-1}$$

First-order statistics

$$\tilde{\mathbf{\Phi}} = \sum_{i=1}^{I}\sum_{j=1}^{J}\left(\phi_{ij} - \boldsymbol{\mu}\right)\left(\phi_{ij} - \boldsymbol{\mu}\right)^{\mathrm{T}} - \mathbf{F}\mathrm{E}\left[\mathbf{h}_i\right]\left(\phi_{ij} - \boldsymbol{\mu}\right)^{\mathrm{T}}$$

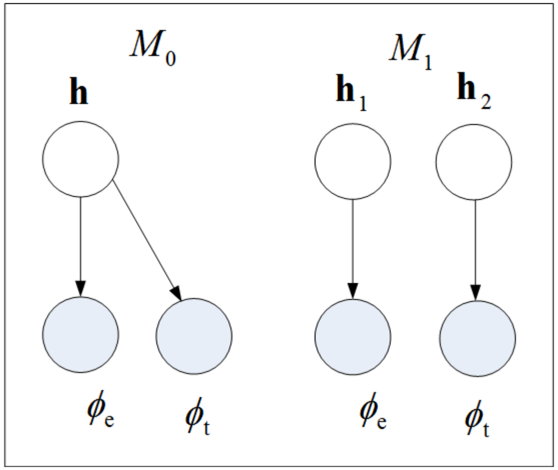
PLDA scoring

- A commonly used scoring method is based on the likelihood ratio test between two hypotheses – whether **the enrollment and test i-vectors, $\boldsymbol{\phi_e}$ and $\boldsymbol{\phi_t}$, are from the same speaker or different speakers**.

- The verification score is given by the log-likelihood ratio between the two models $\{M_0, M_1\}$

$$l(\phi_e, \phi_t) = \log \frac{p(\phi_e, \phi_t \mid M_0)}{p(\phi_e, \phi_t \mid M_1)}$$

$$= \log \frac{\mathcal{N}\left(\begin{bmatrix} \phi_e \\ \phi_t \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \end{bmatrix}, \begin{bmatrix} \mathbf{F}\mathbf{F}^{\mathrm{T}} + \tilde{\boldsymbol{\Phi}} & \mathbf{F}\mathbf{F}^{\mathrm{T}} \\ \mathbf{F}\mathbf{F}^{\mathrm{T}} & \mathbf{F}\mathbf{F}^{\mathrm{T}} + \tilde{\boldsymbol{\Phi}} \end{bmatrix} \right)}{\mathcal{N}\left(\phi_e \mid \boldsymbol{\mu}, \mathbf{F}\mathbf{F}^{\mathrm{T}} + \tilde{\boldsymbol{\Phi}}\right) \mathcal{N}\left(\phi_e \mid \boldsymbol{\mu}, \mathbf{F}\mathbf{F}^{\mathrm{T}} + \tilde{\boldsymbol{\Phi}}\right)}$$



- We don’t build “speaker model”. Speaker verification scores are computed by comparing the enrollment and test i-vectors through PLDA model.

Appendix: joint distribution $p(\phi_\mathrm{e}, \phi_\mathrm{t})$

- Construct the composite equation

$$\underbrace{\begin{bmatrix} \phi_\mathrm{e} \\ \phi_\mathrm{t} \end{bmatrix}}_{\phi'} = \underbrace{\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \end{bmatrix}}_{\boldsymbol{\mu}'} + \underbrace{\begin{bmatrix} \mathbf{F} \\ \mathbf{F} \end{bmatrix}}_{\mathbf{F}'} \mathbf{h} + \underbrace{\begin{bmatrix} \boldsymbol{\varepsilon}_\mathrm{e} \\ \boldsymbol{\varepsilon}_\mathrm{t} \end{bmatrix}}_{\boldsymbol{\varepsilon}'}$$

$$\phi' = \boldsymbol{\mu}' + \mathbf{F}'\mathbf{h} + \boldsymbol{\varepsilon}'$$

- Use the composite terms in the PLDA marginal distribution

$$\begin{aligned} p(\phi') &= \int \mathcal{N}(\phi' \mid \boldsymbol{\mu}' + \mathbf{F}'\mathbf{h}, \boldsymbol{\Phi}') \mathcal{N}(\mathbf{h} \mid 0, \mathbf{I}) d\mathbf{h} \\ &= \mathcal{N}\left(\phi' \mid \boldsymbol{\mu}', \mathbf{F}'\mathbf{F}'^\mathrm{T} + \boldsymbol{\Phi}'\right) \end{aligned}$$

$$\begin{aligned} \mathbf{F}'\mathbf{F}'^\mathrm{T} + \boldsymbol{\Phi}' &= \begin{bmatrix} \mathbf{F} \\ \mathbf{F} \end{bmatrix} \begin{bmatrix} \mathbf{F}^\mathrm{T}, & \mathbf{F}^\mathrm{T} \end{bmatrix} + \begin{bmatrix} \tilde{\boldsymbol{\Phi}} & 0 \\ 0 & \tilde{\boldsymbol{\Phi}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{F}\mathbf{F}^\mathrm{T} + \tilde{\boldsymbol{\Phi}} & \mathbf{F}\mathbf{F}^\mathrm{T} \\ \mathbf{F}\mathbf{F}^\mathrm{T} & \mathbf{F}\mathbf{F}^\mathrm{T} + \tilde{\boldsymbol{\Phi}} \end{bmatrix} \end{aligned}$$

Speaker adaptation (1/2)

- Re-write the log-likelihood ratio as follows

$$l(\phi_\mathrm{e},\phi_\mathrm{t}) = \log \frac{p(\phi_\mathrm{e},\phi_\mathrm{t} \mid M_0)}{p(\phi_\mathrm{e},\phi_\mathrm{t} \mid M_1)} = \log \frac{p(\phi_\mathrm{t} \mid \phi_\mathrm{e})\, \cancel{p(\phi_\mathrm{e})}}{p(\phi_\mathrm{t})\, \cancel{p(\phi_\mathrm{e})}} = \log \frac{p(\phi_\mathrm{t} \mid \phi_\mathrm{e})}{p(\phi_\mathrm{t})}$$

- The numerator and denominator could be interpreted as **speaker-dependent and universal PLDA models**

$$\begin{aligned} p(\phi_\mathrm{t} \mid \phi_\mathrm{e}) &= \int p(\phi_\mathrm{t} \mid \mathbf{h})\, p(\mathbf{h} \mid \phi_\mathrm{e})\, d\mathbf{h} \\ &= \mathcal{N}\Big(\phi_\mathrm{t} \mid \underbrace{\boldsymbol{\mu} + \mathbf{F}\mathbf{m}_h}_{\text{Adapted mean vector}}, \underbrace{\mathbf{F}\mathbf{L}_h^{-1}\mathbf{F}^\mathrm{T} + \tilde{\boldsymbol{\Phi}}}_{\text{Adapted covariance matrix}}\Big) \end{aligned}$$

$$\begin{aligned} p(\phi_\mathrm{t}) &= \int p(\phi_\mathrm{t} \mid \mathbf{h})\, p(\mathbf{h})\, d\mathbf{h} \\ &= \mathcal{N}\left(\phi_\mathrm{t} \mid \boldsymbol{\mu}, \mathbf{F}\mathbf{F}^\mathrm{T} + \tilde{\boldsymbol{\Phi}}\right) \end{aligned}$$

- The verification score is given by the log-likelihood ratio between the speaker-dependent PLDA model and the universal PLDA model, in a way much similar to the idea of *Universal Background Model*.

Speaker adaptation (2/2)

- The parameters $\{\mathbf{m}_h, \mathbf{L}_h^{-1}\}$ are the posterior mean and covariance of the latent speaker variable estimated using the enrollment i-vector ϕ_{e}.

$$p(\mathbf{h} | \phi_{\mathrm{e}}) = \mathcal{N}(\mathbf{h} | \mathbf{m}_h, \mathbf{L}_h^{-1})$$

$$\mathbf{m}_h = \mathbf{L}_h^{-1} \cdot \mathbf{F}^{\mathrm{T}} \tilde{\mathbf{\Phi}}^{-1} (\phi_{\mathrm{e}} - \boldsymbol{\mu})$$

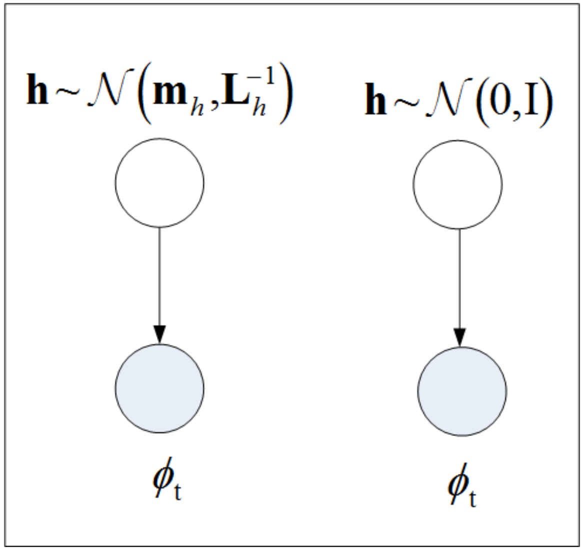
$$\mathbf{L}_h^{-1} = \left(\mathbf{F}^{\mathrm{T}} \tilde{\mathbf{\Phi}}^{-1} \mathbf{F} + I\right)^{-1}$$

- Extension to multi-session enrollment (by-the-book solution)

$$\mathbf{m}_h = \mathbf{L}_h^{-1} \cdot \mathbf{F}^{\mathrm{T}} \tilde{\mathbf{\Phi}}^{-1} \left(\sum_{j=1}^{J} \phi_{ij} - \boldsymbol{\mu}\right)$$

$$\mathbf{L}_h^{-1} = \left(J\mathbf{F}^{\mathrm{T}} \tilde{\mathbf{\Phi}}^{-1} \mathbf{F} + I\right)^{-1} \longleftarrow$$

The covariance shrinks for larger *J*



Graphical model illustrating the model adaptation scoring approach

L. Chen, K. A. Lee et al, "Minimum divergence estimation of speaker prior in multi-session PLDA scoring," in *Proc. IEEE ICASSP*, 2014, pp. 4035 – 4039.

Minimum divergence estimate of speaker prior (1/2)

- For each enrollment session from the speaker i, we compute the mean and covariance of the posterior distribution:

$$\mathbf{m}_{ij} = \mathbf{L}^{-1}\mathbf{F}^{\mathrm{T}}\tilde{\mathbf{\Phi}}^{-1}\left(\phi_{ij} - \boldsymbol{\mu}\right)$$

$$\mathbf{L}^{-1} = \left[\mathbf{F}^{\mathrm{T}}\tilde{\mathbf{\Phi}}^{-1}\mathbf{F} + \mathbf{I}\right]^{-1} \longleftarrow$$

The same for all sessions

- We seek for a Gaussian distribution (i.e., the speaker prior) $N\left(\mathbf{h}|\mathbf{y}_i, \mathbf{P}_i^{-1}\right)$ that best represents the J posterior distributions [Chen et al, ICASSP 2014] minimizing the KL divergence between the prior from the J posteriors:

$$D\left(\theta_{\mathrm{MD}}\right) = \sum_{j=1}^{J} E\left\{\log\frac{\mathcal{N}\left(\mathbf{h}\,|\,\mathbf{m}_{ij}, \mathbf{L}^{-1}\right)}{\mathcal{N}\left(\mathbf{h}\,|\,\mathbf{y}_i, \mathbf{P}_i^{-1}\right)}\right\}$$

$$\mathcal{N}\left(\mathbf{h}|\mathbf{y}_i, \mathbf{P}_i^{-1}\right) \Rightarrow \mathbf{y}_i = \frac{1}{R}\sum_{r=1}^{R}\mathbf{m}_{ij},\ \mathbf{P}_i^{-1} = \mathbf{L}^{-1} + \frac{1}{J}\cdot\sum_{j=1}^{J}\left(\mathbf{m}_{ij} - \mathbf{y}_i\right)\left(\mathbf{m}_{ij} - \mathbf{y}_i\right)^{\mathbf{T}}$$

Minimum divergence estimate of speaker prior (2/2)

- NIST SRE'12 (Core task, CC2): one to over a hundred training segments per speaker, probably with content overlap among different segments for the same speaker.
  - MFCC 57, UBM 512, i-vector 400
  - By-the-book approach does not perform better than the other two approaches.
  - Results on SRE'12 show a clear benefit of MinDiv

Table2 Comparison of three speaker adaptation approaches on CC2 of NIST SRE'12 core task.

	EER (%)	minDCF10	minDCF12	
By-the-book	6.8953	0.6015	0.5394	Male
Mean	3.9395	0.4765	0.4065	
MinDiv	3.5746	0.4238	0.3624	
By-the-book	6.4646	0.6338	0.5621	Female
Mean	3.2145	0.5382	0.4440	
MinDiv	3.0597	0.5235	0.4292	

Interesting Research Topics

RAPID I-VECTOR EXTRACTION

Subspace *ortho*normalizing prior

- Consider the following informative prior

$$\mathbf{x} \sim \mathcal{N}\left(0, \boldsymbol{\Sigma}_{\mathrm{p}}\right) \quad \text{with} \quad \boldsymbol{\Sigma}_{\mathrm{p}} = \left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1}$$

- I-vector extraction

$$\phi = \mathbf{L}^{-1} \cdot \mathbf{T}^{\mathrm{T}}\tilde{\mathbf{F}} \qquad \mathbf{L}^{-1} = \left[\mathbf{T}^{\mathrm{T}}\mathbf{T} + \mathbf{T}^{\mathrm{T}}\mathbf{N}\mathbf{T}\right]^{-1}$$

$$\begin{aligned}
\phi &= \left[\mathbf{T}^{\mathrm{T}}\mathbf{T} + \mathbf{T}^{\mathrm{T}}\mathbf{N}\mathbf{T}\right]^{-1} \cdot \mathbf{T}^{\mathrm{T}}\tilde{\mathbf{F}} \\
&= \left(\left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)\left[\mathbf{I} + \left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1}\mathbf{T}^{\mathrm{T}}\mathbf{N}\mathbf{T}\right]\right)^{-1} \cdot \mathbf{T}^{\mathrm{T}}\tilde{\mathbf{F}} \\
&= \left[\mathbf{I} + \left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1}\mathbf{T}^{\mathrm{T}}\mathbf{N}\mathbf{T}\right]^{-1}\left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1} \cdot \mathbf{T}^{\mathrm{T}}\tilde{\mathbf{F}}
\end{aligned}$$

Solving the matrix inversion (cont'd)

- Let $\mathbf{A} = (\mathbf{I} + \mathbf{N})$

$$\left[\mathbf{I}+\mathbf{N}\cdot\mathbf{T}\left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1}\mathbf{T}^{\mathrm{T}}\right]^{-1} = \left(\mathbf{I}+\mathbf{N}-\mathbf{N}\mathbf{U}_2\mathbf{U}_2^{\mathrm{T}}\right)^{-1} = \left(\mathbf{A}-\mathbf{N}\mathbf{U}_2\mathbf{U}_2^{\mathrm{T}}\right)^{-1}$$

- Using the matrix inversion lemma

$$\left(\mathbf{A}-\mathbf{N}\mathbf{U}_2\mathbf{U}_2^{\mathrm{T}}\right)^{-1} = \mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{N}\left(\mathbf{I}-\mathbf{U}_2\mathbf{U}_2^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{N}\right)^{-1}\mathbf{U}_2\mathbf{U}_2^{\mathrm{T}}\mathbf{A}^{-1}$$

- Using again the matrix inversion identity $(\mathbf{I}+\mathbf{P}\mathbf{Q})^{-1}\mathbf{P} = \mathbf{P}(\mathbf{I}+\mathbf{Q}\mathbf{P})^{-1}$

$$\left[\mathbf{I}+\mathbf{N}\cdot\mathbf{T}\left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1}\mathbf{T}^{\mathrm{T}}\right]^{-1} = \mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{N}\mathbf{U}_2\mathbf{U}_2^{\mathrm{T}}\left(\mathbf{I}-\mathbf{A}^{-1}\mathbf{N}\mathbf{U}_2\mathbf{U}_2^{\mathrm{T}}\right)^{-1}\mathbf{A}^{-1}$$

Rapid computation of i-vector

- Uniform occupancy assumption:

$$\boxed{\mathbf{A}^{-1}\mathbf{N} = (\mathbf{I}+\mathbf{N})^{-1}\mathbf{N} \approx \alpha\mathbf{I}} \quad \text{for} \quad \boxed{0 \le \alpha < 1}$$

- Or equivalently

$$\frac{N_c}{1+N_c} \approx \alpha \ \forall c$$

- The matrix inversion can be simplified as

$$\left[\mathbf{I}+\mathbf{N}\cdot\mathbf{T}\left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1}\mathbf{T}^{\mathrm{T}}\right]^{-1} \approx \mathbf{A}^{-1} + \alpha\cdot\mathbf{U}_2\mathbf{U}_2^{\mathrm{T}}\left(\mathbf{I}-\mathbf{A}^{-1}\mathbf{N}\mathbf{U}_2\mathbf{U}_2^{\mathrm{T}}\right)^{-1}\mathbf{A}^{-1}$$

- Since $\mathbf{T} \perp \mathbf{U}_2$, the second term diminishes

$$\phi = \left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1}\mathbf{T}^{\mathrm{T}}\boxed{\left[\mathbf{I}+\mathbf{N}\mathbf{T}\left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1}\mathbf{T}^{\mathrm{T}}\right]^{-1}}\tilde{\mathbf{F}} \approx \left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1}\mathbf{T}^{\mathrm{T}}\boxed{(\mathbf{I}+\mathbf{N})^{-1}}\tilde{\mathbf{F}}$$

Computational complexity and memory cost

	Complexity	Memory cost	Time ratio
Baseline (slow)	$O(CFM^2 + M^3)$	$O(CFM)$	106.44
Baseline (fast)	$O(CFM + CM^2 + M^3)$	$O(CFM + CM^2)$	11.99
Proposed (exact)	$O(CFM + CM^2 + M^3)$	$O(CFM + CM^2)$	12.65
Proposed (fast)	$O(CFM)$	$O(CFM)$	1

Baseline (slow) $$\phi = \left(\mathbf{I} + \sum_c N_c \mathbf{T}_c^{\mathrm{T}} \mathbf{T}_c\right)^{-1} \cdot \mathbf{T}^{\mathrm{T}} \tilde{\mathbf{F}}$$

Baseline (fast) $$\phi = \left(\mathbf{I} + \sum_c N_c \mathbf{A}_c\right)^{-1} \cdot \mathbf{T}^{\mathrm{T}} \tilde{\mathbf{F}}$$

Proposed (exact) $$\phi = \left(\sum_c (N_c + 1) \mathbf{T}_c^{\mathrm{T}} \mathbf{T}_c\right)^{-1} \cdot \mathbf{T}^{\mathrm{T}} \tilde{\mathbf{F}}$$

Proposed (fast) $$\phi = \left(\mathbf{T}^{\mathrm{T}} \mathbf{T}\right)^{-1} \mathbf{T}^{\mathrm{T}} \left(\mathbf{I} + \mathbf{N}\right)^{-1} \tilde{\mathbf{F}}$$

SRE'10 core-extended (female)

- For the **tel-tel CC 5**, the relative degradation is **10.04%** in EER and **4.54%** in min DCF.

EER **MinDCF10**

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9
baseline	**2.1986**	**3.8313**	4.3591	2.8421	3.3182	5.3826	6.3412	2.6505	2.1288
	0.3657	**0.6088**	0.6553	0.4593	**0.5179**	0.8506	0.7698	**0.5199**	0.2393
proposed (exact)	2.3014	3.8716	4.3417	**2.7644**	**3.2932**	**5.3464**	**6.3280**	**2.6176**	1.9896
	0.3680	0.6091	**0.6468**	**0.4581**	0.5192	0.8550	0.7565	0.5259	**0.2212**
proposed (fast)	2.4699	4.4485	**4.1780**	2.8356	3.6514	5.9624	7.2228	2.9741	1.8361
	0.3681	0.6507	0.7142	0.4963	0.5414	**0.8349**	**0.7331**	0.5786	0.2881
$(\mathbf{I}+\mathbf{N})^{-1}\,\widetilde{\mathbf{F}}$	2.4144	4.5298	4.9959	3.2292	3.8124	6.2883	6.8657	2.8703	**1.7414**
	0.4247	0.7267	0.7856	0.5281	0.6106	0.9183	0.8013	0.5915	0.2994
$(\mathbf{I}+\mathbf{N})^{-1}\,\mathbf{F}$	3.5880	6.7725	6.2079	4.4367	4.6302	7.3215	9.7463	3.8837	2.2043
	0.5502	0.8591	0.8916	0.6586	0.6537	0.9457	0.8589	0.6860	0.3936

L. Xu, K. A. Lee, H. Li, and Z. Yang, "," in *Proc. Odyssey 2016: The Speaker and Language Recognition Workshop*, 2016, pp. 47 – 52.

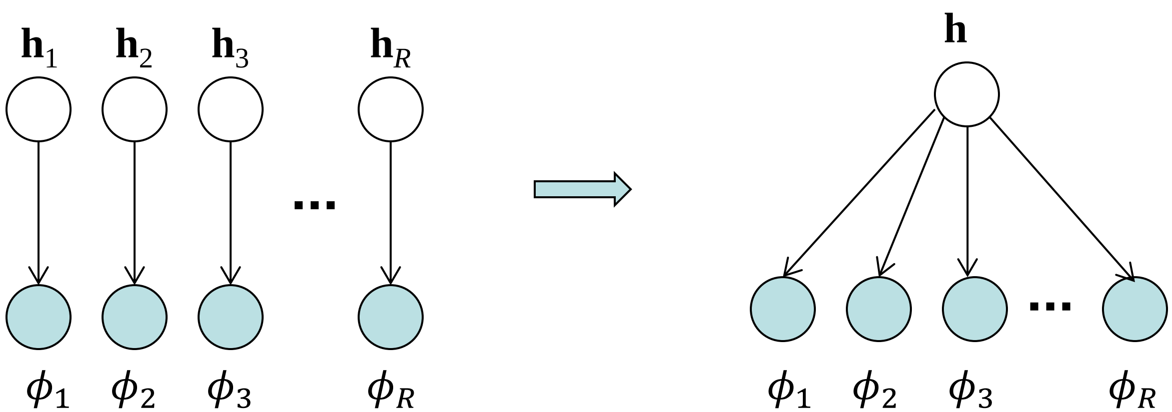
Kong Aik LEE
*Scientist, Institute for Infocomm Research, A*STAR, Singapore*

THANK YOU

Learning Diary

A. Sufficient statistics are pre-whitened prior to i-vector extraction. This pre-whitening step does not change the i-vector as similar transformation is absorbed by the T matrix. Proof this.

B. A probabilistic LDA (PLDA) model is an extension to the classical factor analysis model by tying of observed variables



Explain the rationale of tying multiple observed variables (in the context of i-vector PLDA speaker recognition system) to a single latent variable.