INTERSPEECH 2016 Tutorial:
Machine Learning for Speaker Recognition

Man-Wai Mak† and Jen-Tzung Chien‡

†The Hong Kong Polytechnic University, Hong Kong
‡National Chiao Tung University, Taiwan

September 8, 2016
Outline

1 Introduction
   1.1. Fundamentals of speaker recognition
   1.2. Feature extraction and scoring
   1.3. Modern speaker recognition approaches

2 Learning Algorithms

3 Learning Models

4 Deep Learning

5 Case Studies

6 Future Direction
Speaker recognition is a technique to recognize the identity of a speaker from a speech utterance.
Speaker identification

- Determine whether unknown speaker matches one of a set known speakers
- One-to-many mapping
- Often assumed that unknown voice must come from a set of known speakers – referred to as close-set identification
- Adding “none of the above” option to closed-set identification gives open-set identification
Speaker verification

- Determine whether unknown speaker matches a **specific** speaker
- One-to-one mapping
- **Close-set** verification: The population of clients is fixed
- **Open-set** verification: New clients can be added without having to redesign the system.
Speaker diarization

- Determine when a speaker change has occurred in speech signal (segmentation)
- Group together speech segments corresponding to the same speaker (clustering)
- Prior speaker information may or may not be available
Input mode

- **Text-dependent**
  - Recognition system knows text spoken by persons
  - Fixed phrases or prompted phrases
  - Used for applications with strong control over user input, e.g., biometric authentication
  - Speech recognition can be used for checking spoken text to improve system performance
  - Sentences typically very short

- **Text-independent**
  - No restriction on the text, typically conversational speech
  - Used for applications with less control over user input, e.g., forensic speaker ID
  - More flexible but recognition is more difficult
  - Speech recognition can be used for extracting high-level features to boost performance
  - Sentences typically very long
Introduction

1. Fundamentals of speaker recognition
2. Feature extraction and scoring
3. Modern speaker recognition approaches

Learning Algorithms

Learning Models

Deep Learning

Case Studies

Future Direction
Feature extraction

- Speech is a time-varying signal conveying multiple layers of information
  - Words
  - Speaker
  - Language
  - Emotion

- Information in speech is observed in the time and frequency domains

```plaintext
Acoustic Features
- Speech is a continuous evolution of the vocal tract
- Need to extract a sequence of spectra or sequence of spectral coefficients
- Use a sliding window - 25 ms window, 10 ms shift

DCT \log|X(\omega)|
MFCC
```

- Produces time-frequency evolution of the spectrum
Feature extraction from speech

- Feature extraction consists in transforming the speech signal to a set of feature vectors. Most of the feature extraction used in speaker recognition systems relies on a cepstral representation of speech.

*Figure*: Modular representation of MFCC feature extractor
Mel-Frequency Cepstrum Coefficients (MFCCs)

\[
S(k) = \sum_{n=0}^{N-1} s(n)e^{-j2\pi nk/N}
\]

\[
freq \quad \log \quad \text{DCT} \quad \text{MFCCs}
\]

\[
H_m(k) \quad \text{Filter Bank}
\]

\[
H_{15}(k) : \text{the 15th triangular filter}
\]

\[
\begin{align*}
\text{MFCC}_n &= c_n = \sum_{m=1}^{M} \cos \left[ n \left( m - \frac{1}{2} \right) \frac{\pi}{M} \right] X(m) = d_n^T \tilde{X}, \quad 0 \leq n \leq P \\
\Rightarrow \text{MFCC vector} &= \begin{bmatrix} c_0 & c_1 & \cdots & c_P \end{bmatrix}^T = D\tilde{X}
\end{align*}
\]

\[
D : \text{DCT Transformation matrix } [P \times M] \\
M : \text{No. of triangular filters in the filter bank, typically 20 \sim 30} \\
P : \text{No. of cepstral coefficients, typically 12} \\
c_0 : \text{Logarithm of energy of the current frame}
\]

\textbf{Figure}: Computing MFCC from one frame of speech
Modeling sequence of features

- For most recognition tasks, we need to model the distribution of feature vector sequences.

- In practice, we often use the Gaussian mixture models (GMMs).
A Gaussian mixture model, namely the universal background model (UBM), is trained to represent the speech of the general population.

\[
p(x|\text{UBM}) = p(x|\Lambda^{\text{ubm}}) = \sum_{c=1}^{C} \pi^{\text{ubm}}_c \mathcal{N}(x|\mu^{\text{ubm}}_c, \Sigma^{\text{ubm}}_c)
\]

The UBM parameters \(\Lambda^{\text{ubm}} = \left\{ \pi^{\text{ubm}}_c, \mu^{\text{ubm}}_c, \Sigma^{\text{ubm}}_c \right\}_{c=1}^{C}\) are estimated by the expectation-maximization algorithm using the speech of many speakers.
Denote the acoustic vectors from a large population as $\mathcal{X} = \{x_t; t = 1, \ldots, T\}$

**Expectation step:**
- Conditional distribution of mixture component $c$:
  \[
  \gamma_t(c) = p(c|x_t) = \frac{\pi_c \mathcal{N}(x_t | \mu_{c^{ubm}}, \Sigma_c^{ubm})}{\sum_{c=1}^C \pi_c^{ubm} \mathcal{N}(x_t | \mu_{c^{ubm}}, \Sigma_c^{ubm})}
  \]

**Maximization step:**
- Mixture weights: $\pi_{c^{ubm}} = \frac{1}{T} \sum_{t=1}^T \gamma_t(c)$
- Mean vectors: $\mu_{c^{ubm}} = \frac{\sum_{t=1}^T \gamma_t(c)x_t}{\sum_{t=1}^T \gamma_t(c)}$
- Covariance matrices: $\Sigma_c^{ubm} = \frac{\sum_{t=1}^T \gamma_t(c)x_t x_t^T}{\sum_{t=1}^T \gamma_t(c)} - \mu_{c^{ubm}}(\mu_{c^{ubm}})^T$
Each target speaker is represented by a Gaussian mixture model:

\[ p(x|\text{Spk } s) = p(x|\Lambda^{(s)}) = \sum_{c=1}^{C} \pi^{(s)}_c \mathcal{N}(x|\mu^{(s)}_c, \Sigma^{(s)}_c) \]

where \( \Lambda^{(s)} = \{\pi^{(s)}_c, \mu^{(s)}_c, \Sigma^{(s)}_c\}_{c=1}^{C} \) are learned by using maximum a posteriori (MAP) adaptation [Reynolds et al., 2000].
The MAP algorithm finds the parameters of target-speaker’s GMM given UBM parameters $\Lambda^{\text{ubm}} = \left\{ \pi_c^{\text{ubm}}, \mu_c^{\text{ubm}}, \Sigma_c^{\text{ubm}} \right\}_{c=1}^C$

First step is the same as EM. Given $T_s$ acoustic vectors $\mathcal{X}^{(s)} = \{x_1, \ldots, x_{T_s}\}$ from speaker $s$, we compute the statistics:

$$n_c = \sum_{t=1}^{T_s} \gamma_t(c) \quad \text{and} \quad E_c(\mathcal{X}^{(s)}) = \frac{1}{n_c} \sum_{t=1}^{T_s} \gamma_t(c) x_t$$

Adapt UBM parameters by

$$\mu_c^{(s)} = \alpha_c E_c(\mathcal{X}^{(s)}) + (1 - \alpha_c) \mu_c^{\text{ubm}}$$

where

$$\alpha_c = \frac{n_c}{n_c + r}$$

and $r$ is called the relevance factor. Typically, $r = 16$. 

Adapt the UBM model to each speaker using the MAP algorithm: 

\[
\mu_c^{(s)} = \alpha_c E_c(\mathcal{X}^{(s)}) + (1 - \alpha_c) \mu_c^{\text{ubm}}
\]

- \( \alpha_c \to 1 \) when \( \mathcal{X}^{(s)} \) comprises lots of data and \( \alpha_c \to 0 \) otherwise.

---

In practice, only the mean vectors will be adapted:
GMM-UBM scoring

Given the acoustic vectors $\mathcal{X}(t)$ from a test speaker and a claimed identity $s$, speaker verification can be formulated as a 2-class hypothesis problem:

- $H_0$: $\mathcal{X}(t)$ comes from the true speaker $s$
- $H_1$: $\mathcal{X}(t)$ comes from an impostor

Verification score is a log-likelihood ratio:

$$S_{LR}(\mathcal{X}(t) \mid \Lambda^{(s)}, \Lambda^{ubm}) = \log p(\mathcal{X}(t) \mid \Lambda^{(s)}) - \log p(\mathcal{X}(t) \mid \Lambda^{ubm})$$
Sources of variability

[Diagram showing sources of variability with axes labeled as inter-speaker variability and nuisance variability. The diagram includes circles and dots to represent target speaker model, test data, and UBM (Universal Background Model).]
How to account for variability

- **GMM-SVM** [Campbell et al., 2006]:
  - Create supervectors from target-speaker GMMs.
  - Then, project the supervectors to a subspace in which inter-speaker variability is maximized and nuisance variability is minimized.
  - Perform SVM classification on the projected subspace.

- **Joint Factor Analysis**:
  - Speaker and session variabilities are represented by latent variables (speaker factors and channel factors) in a factor analysis model.
  - During scoring, session variabilities are accounted for by integrating over the latent variables, e.g., the channel factors as in [Kenny et al., 2007a].

- **I-Vector + PLDA**:
  - Utterances are represented by the posterior means of latent factors, called the i-vectors [Dehak et al., 2011].
  - I-vectors capture both speaker and channel information.
  - During scoring, the unwanted channel variability is removed by LDA projection or by integrating out the latent factors in the PLDA model.

---

2For the relationship between JFA and I-vectors and their derivations, see http://www.eie.polyu.edu.hk/∼mwmak/papers/FA-Ivector.pdf
Performance measures

- For speaker identification

  \[
  \text{Recognition Rate} = \frac{\text{No. of correct recognitions}}{\text{Total no. of trials}}
  \]

- For speaker verification

  \[
  \text{False Rejection Rate (FRR)} = \text{Miss probability} = \frac{\text{No. of true-speakers rejected}}{\text{Total no. of true-speaker trials}}
  \]

  \[
  \text{False Acceptance Rate (FAR)} = \text{False alarm probability} = \frac{\text{No. of impostors accepted}}{\text{Total no. of impostor attempts}}
  \]

- Equal error rate (EER) corresponds to the operating point at which FAR = FRR
Detection error tradeoff (DET) curves are similar to receiver operating characteristic curves but with nonlinear x- and y-axis.
Detection cost functions

- Detection cost function (DCF) is a weighted sum of the FRR ($P_{\text{Miss|Target}}$) and FAR ($P_{\text{FalseAlarm|Nontarget}}$):

$$C_{\text{Det}}(\theta) = C_{\text{Miss}} \times P_{\text{Miss|Target}}(\theta) \times P_{\text{Target}} + C_{\text{FalseAlarm}} \times P_{\text{FalseAlarm|Nontarget}}(\theta) \times (1 - P_{\text{Target}})$$

where $\theta$ is a decision threshold.

- Normalized cost:

$$C_{\text{Norm}} = C_{\text{Det}}(\theta)/C_{\text{Default}}$$

where

$$C_{\text{Default}} = \min\left\{ \begin{align*} C_{\text{Miss}} \times P_{\text{Target}} \\ C_{\text{FalseAlarm}} \times (1 - P_{\text{Target}}) \end{align*} \right\}$$

- NIST 2008 SRE and earlier:

$$C_{\text{Miss}} = 10; \quad C_{\text{FalseAlarm}} = 1; \quad P_{\text{Target}} = 0.01$$

- NIST 2010 SRE:

$$C_{\text{Miss}} = 1; \quad C_{\text{FalseAlarm}} = 1; \quad P_{\text{Target}} = 0.001$$
Detection cost functions

- Detection cost function for NIST 2012 SRE:

\[ C_{\text{Det}}(\theta) = C_{\text{Miss}} \times P_{\text{Miss|Target}}(\theta) \times P_{\text{Target}} + C_{\text{FalseAlarm}} \times (1 - P_{\text{Target}}) \times \]

\[ \left[ P_{\text{FalseAlarm|KnownNontarget}}(\theta) \times P_{\text{Known}} + P_{\text{FalseAlarm|UnknownNontarget}} \times (1 - P_{\text{Known}}) \right] \]

\[ C_{\text{Norm}}(\theta) = \frac{C_{\text{Det}}(\theta)}{C_{\text{Miss}} \times P_{\text{Target}}} \]

- Parameters for core test conditions

\[ C_{\text{Miss}} = 1; \quad C_{\text{FalseAlarm}} = 1; \quad P_{\text{Target1}} = 0.01; \quad P_{\text{Target2}} = 0.001; \]
\[ P_{\text{Known}} = 0.5 \]

- Primary cost:

\[ C_{\text{Primary}} = \frac{C_{\text{Norm1}}(\theta_1) + C_{\text{Norm2}}(\theta_2)}{2} \]
Detection cost functions

Detection cost function for NIST 2016 SRE:

\[ C_{\text{Det}}(\theta) = C_{\text{Miss}} \times P_{\text{Miss|Target}}(\theta) \times P_{\text{Target}} + C_{\text{FalseAlarm}} \times P_{\text{FalseAlarm|Nontarget}}(\theta) \times (1 - P_{\text{Target}}) \]

\[ C_{\text{Norm}}(\theta) = \frac{C_{\text{Det}}(\theta)}{C_{\text{Miss}} \times P_{\text{Target}}} \]

Parameters for core test conditions

\[ C_{\text{Miss}} = 1; \quad C_{\text{FalseAlarm}} = 1; \quad P_{\text{Target1}} = 0.01; \quad P_{\text{Target2}} = 0.005 \]

\[ P_{\text{Target1}} \rightarrow C_{\text{Norm1}}(\theta_1) \quad \text{and} \quad P_{\text{Target2}} \rightarrow C_{\text{Norm2}}(\theta_2) \]

Primary cost:

\[ C_{\text{Primary}} = \frac{C_{\text{Norm1}}(\theta_1) + C_{\text{Norm2}}(\theta_2)}{2} \]
Outline

1 Introduction
   - 1.1. Fundamentals of speaker recognition
   - 1.2. Feature extraction and scoring
   - 1.3. Modern speaker recognition approaches

2 Learning Algorithms

3 Learning Models

4 Deep Learning

5 Case Studies

6 Future Direction
Outline

1 Introduction

2 Learning Algorithms
   2.1 Machine learning
   2.2 EM algorithm
   2.3 Approximate inference
   2.4 Bayesian learning

3 Learning Models

4 Deep Learning

5 Case Studies

6 Future Direction
Outline

1. Introduction

2. Learning Algorithms
   - 2.1. Machine learning
   - 2.2. EM algorithm
   - 2.3. Approximate inference
   - 2.4. Bayesian learning

3. Learning Models

4. Deep Learning

5. Case Studies

6. Future Direction
Model-based method

- **Machine learning** provides a wide range of model-based approaches for speaker recognition
- **Model-based** approach aims to incorporate the physical phenomena, measurements, **uncertainties** and noises in the form of mathematical models
- This approach is developed in a **unified** manner through different algorithms, examples, applications, and **case studies**
- **Main-stream** methods are based on the **statistical** models
- **Latent variable models** in speaker recognition include
  - joint factor analysis (JFA)
  - probabilistic linear discriminant analysis (PLDA)
  - Gaussian mixture model (GMM)
  - mixture of PLDA
Deep **structured/hierarchical** learning
Rapidly developed and widely applied for many applications
Multiple layers of **nonlinear processing units**
High-level abstraction

---

**Run**

**Jump**
### Model-based method vs. neural network

<table>
<thead>
<tr>
<th></th>
<th>Model-based method</th>
<th>Neural network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Top-down</td>
<td>Bottom-up</td>
</tr>
<tr>
<td>Representation</td>
<td>Intuitive</td>
<td>Distributed</td>
</tr>
<tr>
<td>Interpretation</td>
<td>Easy</td>
<td>Harder</td>
</tr>
<tr>
<td>Model-based method</td>
<td>Neural network</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>Semi/unsupervised</td>
<td>Easier</td>
<td>Harder</td>
</tr>
<tr>
<td>Incorp. domain knowl.</td>
<td>Easy</td>
<td>Hard</td>
</tr>
<tr>
<td>Incorp. constraint</td>
<td>Easy</td>
<td>Hard</td>
</tr>
<tr>
<td>Incorp. uncertainty</td>
<td>Easy</td>
<td>Hard</td>
</tr>
</tbody>
</table>
# Model-based method vs. neural network

<table>
<thead>
<tr>
<th></th>
<th>Model-based method</th>
<th>Neural network</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Learning</strong></td>
<td>Many algorithms</td>
<td>Back-propagation</td>
</tr>
<tr>
<td><strong>Inference/decode</strong></td>
<td><strong>Harder</strong></td>
<td><strong>Easier</strong></td>
</tr>
<tr>
<td><strong>Evaluation on</strong></td>
<td>ELBO</td>
<td><strong>End performance</strong></td>
</tr>
</tbody>
</table>
## Modern machine learning

<table>
<thead>
<tr>
<th>Feature</th>
<th>Model-based method</th>
<th>Neural network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Top-down</td>
<td>Bottom-up</td>
</tr>
<tr>
<td>Representation</td>
<td>Intuitive</td>
<td>Distributed</td>
</tr>
<tr>
<td>Interpretation</td>
<td>Easy</td>
<td>Harder</td>
</tr>
<tr>
<td>Semi/unsupervised</td>
<td>Easier</td>
<td>Harder</td>
</tr>
<tr>
<td>Incorp. domain knowl.</td>
<td>Easy</td>
<td>Hard</td>
</tr>
<tr>
<td>Incorp. constraint</td>
<td>Easy</td>
<td>Hard</td>
</tr>
<tr>
<td>Incorp. uncertainty</td>
<td>Easy</td>
<td>Hard</td>
</tr>
<tr>
<td>Learning</td>
<td>Many algorithms</td>
<td>Back-propagation</td>
</tr>
<tr>
<td>Inference/decode</td>
<td>Harder</td>
<td>Easier</td>
</tr>
<tr>
<td>Evaluation on</td>
<td>ELBO</td>
<td>End performance</td>
</tr>
</tbody>
</table>
Outline

1 Introduction

2 Learning Algorithms
   2.1 Machine learning
   2.2 EM algorithm
   2.3 Approximate inference
   2.4 Bayesian learning

3 Learning Models

4 Deep Learning

5 Case Studies

6 Future Direction
Parameter estimation

Assume we have a collection of acoustic frames $X = \{x_t\}_{t=1}^T$ for estimation of model parameters $\theta$.

- **Maximum likelihood (ML) estimation**
  
  $$
  \theta_{\text{ML}} = \arg \max_{\theta} p(X|\theta)
  $$

- **Maximum a posteriori (MAP) estimation**
  
  $$
  \theta_{\text{MAP}} = \arg \max_{\theta} p(\theta|X) = \arg \max_{\theta} p(X|\theta)p(\theta)
  $$

  where $p(\theta)$ denotes the prior distribution of $\theta$. 
Expectation-maximization algorithm

- Likelihood function for observations $x$ in latent variable model with latent variable $z$
  \[ p(x|\theta) = \sum_z p(x, z|\theta) \]

- **Expectation (E) step**: calculate an auxiliary function
  \[ Q(\theta, \theta^{old}) = \mathbb{E}_z[\log p(x, z|\theta)|x, \theta^{old}] \]

- **Maximization (M) step**: find a new estimate $\theta^{new}$ via
  \[ \theta^{new} = \arg \max_{\lambda} Q(\theta, \theta^{old}) \]

- EM algorithm [Dempster et al., 1977] for ML can be extended for MAP
Lower bound & KL divergence

- Introduce an **approximate** or **variational** distribution \( q(z) \) and adopt the **Jensen’s inequality** for convex function \(-\log(\cdot)\) to obtain

  \[
  \log p(x|\theta) = \log \sum_z \frac{p(x, z|\theta)}{q(z)} q(z) = \log \mathbb{E}_q \left[ \frac{p(x, z|\theta)}{q(z)} \right] \\
  \geq \mathbb{E}_q \left[ \log \frac{p(x, z|\theta)}{q(z)} \right] \triangleq \mathcal{L}(q, \theta)
  \]

  \[
  \sum_z q(z) \log p(x|\theta) - \mathcal{L}(q, \theta) = -\sum_z q(z) \log \left\{ \frac{p(z|x, \theta)}{q(z)} \right\} \triangleq \text{KL}(q\|p)
  \]

**Evidence Decomposition**

\[
\log p(x|\theta) = \text{KL}(q\|p) + \mathcal{L}(q, \theta)
\]
Maximum Likelihood

$$\text{KL}(q \| p) = -\mathbb{E}_q[\log p(z|x, \theta)] - \mathbb{H}_q[z]$$

$$\mathcal{L}(q, \theta) = \mathbb{E}_q[\log p(x, z|\theta)] + \mathbb{H}_q[z]$$

Maximizing $p(x|\theta)$ is equivalent to first setting $\text{KL}(q \| p) = 0$ or approximating (E-step)

$$q(z) = p(z|x, \theta^{old})$$

then maximizing the resulting lower bound (M-step)

$$\mathcal{L}(q, \theta) \triangleq Q(\theta, \theta^{old}) + \text{const}$$

where $Q(\theta, \theta^{old}) \triangleq \mathbb{E}_q[\log p(x, z|\theta)|x, \theta^{old}]$ is a concave function
EM algorithm

\[ KL(q \| p) \]

\[ \mathcal{L}(q, \theta) \]

\[ \log p(x | \theta) \]

0
EM algorithm: E-step

\[ \text{KL}(q \| p) = 0 \]

\[ \mathcal{L}(q, \theta^{\text{old}}) \]

\[ \log p(\mathbf{x} | \theta^{\text{old}}) \]
EM algorithm: M-step

\[
L(q; \theta^{\text{new}}) = \log p(x|\theta^{\text{new}})
\]

\[
KL(q\|p) \leq \log p(x|\theta^{\text{new}})
\]
EM algorithm: lower bound

\[
\log p(x|\theta) \geq \mathcal{L}(q, \theta^{\text{old}}) \geq \mathcal{L}(q, \theta^{\text{new}})
\]
Outline

1 Introduction

2 Learning Algorithms
   2.1 Machine learning
   2.2 EM algorithm
   2.3 Approximate inference
   2.4 Bayesian learning

3 Learning Models

4 Deep Learning

5 Case Studies

6 Future Direction
Why approximate inference?

- There are a number of latent variables in model-based speaker recognition
  - i-vectors
  - common factors
  - variability matrix
  - mixture labels
  - channel, speaker and noise information

- **Posterior** distribution of latent variables should be analytical and factorizable

- **Evolution of inference algorithms**
  - maximum likelihood
  - maximum \textit{a posteriori}
  - variational Bayesian
  - Gibbs sampling
Posterior distribution

$$p(z|x) = \frac{p(x|z)p(z)}{\int p(x|z)p(z)dz}$$

- Latent variables and parameters $z = \{z_1, \ldots, z_m\}$ are coupled
Approximate posterior

Find an **approximate** distribution \( q(z) \) that is **factorizable** and maximally similar to the **true** posterior \( p(z|x) \).
Variational Bayesian inference

\[ q(z_1:m | \nu_1:m) = \prod_{j=1}^{m} q(z_j | \nu_j) \]

Variational calculus

functional

\[ \mathcal{L}(q) : q \mapsto \mathcal{L}(q) \]

Optimization problem

\[ \max_q \mathcal{L}(q) \quad \text{s.t.} \quad \int_z q(dz) = 1 \]

\[ p(\mathbf{x}) = \text{KL}(q\|p) + \mathcal{L}(q) \]

where \( \text{KL}(q\|p) = -\mathbb{E}_q[\ln p(z|\mathbf{x})] - \mathbb{H}_q[z] \)

\[ \mathcal{L}(q) = \mathbb{E}_q[\ln p(\mathbf{x}, z)] + \mathbb{H}_q[z] \]

(Evidence Lower BOund, ELBO)
Estimation for variational distribution

$$\max_{q(z)} \mathbb{E}_q[\log p(x, z)] + \mathbb{H}_q[z]$$

s.t. $$\int q(dz) = 1$$

$$\hat{q}(z_j|\nu_j) = \frac{\exp(\mathbb{E}_{i\neq j}[\log p(x, z|\nu)])}{\int \exp(\mathbb{E}_{i\neq j}[\log p(x, z|\nu)]) dz_j}$$
Variational Bayesian (VB) inference is implemented via a **doubly-looped** algorithm.

**VB-EM algorithm**

- **VB-E step**: calculate the **variational distribution** $q(z)$ in inner loop
  
  $$\hat{q}(z) = \arg\max_{q(z)} \mathcal{L}(q, \theta)$$

- **VB-M step**: calculate the **model parameter** $\theta$ in outer loop
  
  $$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\hat{q}, \theta)$$

- **Convex optimization** is performed

- VB-EM steps **converge** by a number of iterations
Gibbs sampling algorithm

Initialize $z^{(1)}$, where $z = z_{1:m}$

for $\tau \leftarrow 1$ to $T - 1$ do

    for $j \leftarrow 1$ to $m$ do

        Sample $z_j^{(\tau + 1)} \sim p(z_j|z_{1:(j-1)}, z_{j+1:m})$

    end for

end for
Gibbs sampling

Two dimensional Gaussian mixture model with two mixture components
Gibbs sampling

\[ z_j \sim p(\cdot \mid z_{-j}, \mathbf{x}) \]

Randomly assign mixture component for each sample \( j \)
Gibbs sampling

$z_j \sim p(\cdot | z_{-j}, \mathbf{x})$

Extract one sample and compute the conditional distribution
Gibbs sampling

$$z_j \sim p(\cdot | z_{-j}, \mathbf{x})$$

Sample a mixture component from the conditional distribution
Gibbs sampling

\[ z_j \sim p(\cdot \mid z_{-j}, \mathbf{x}) \]

Extract one sample and compute the conditional distribution
Sample a mixture component from the conditional distribution
Gibbs sampling

$z_j \sim p(\cdot \mid z_{-j}, \mathbf{x})$

Extract one sample and compute the conditional distribution
Sample a mixture component from the conditional distribution

\[ z_j \sim p(\cdot \mid z_{-j}, \mathbf{x}) \]
Gibbs sampling

Extract one sample and compute the conditional distribution

\[ z_j \sim p( \cdot \mid z_{-j}, \mathbf{x} ) \]
Sample a mixture component from the conditional distribution
Gibbs sampling

\[ z_j \sim p(\cdot | z_{-j}, \mathbf{x}) \]

Extract one sample and compute the conditional distribution
Gibbs sampling

Sample a mixture component from the conditional distribution

\[ z_j \sim p(\cdot \mid z_{-j}, \mathbf{x}) \]
Gibbs sampling

\[ z_j \sim p(\cdot | z_{-j}, x) \]

Extract one sample and compute the conditional distribution
Gibbs sampling

Sample a mixture component from the conditional distribution

\[ z_j \sim p(\cdot \mid z_{-j}, \mathbf{x}) \]
Gibbs sampling

$z_j \sim p(\cdot \mid z_{-j}, x)$

Extract one sample and compute the conditional distribution
Gibbs sampling

\[ z_j \sim p( \cdot | z_{-j}, \mathbf{x} ) \]

Sample a mixture component from the conditional distribution
Gibbs sampling

Extract one sample and compute the conditional distribution

\[ z_j \sim p(\cdot | z_{-j}, \mathbf{x}) \]
Sample a mixture component from the conditional distribution

\[ z_j \sim p(\cdot \mid z_{-j}, \boldsymbol{x}) \]
Gibbs sampling

Finally obtain an appropriate clustering result
Variational Bayes

- deterministic approximation
- find an analytical proxy $q(z)$ that is maximally similar to $p(z|x)$
- inspect distribution statistics
- never generate exact results
- fast
- often hard work to derive
- convergence guarantees
- need a specific parametric form

Gibbs sampling

- stochastic approximation
- design an algorithm that draws samples $z^{(1)}, \ldots, z^{(\tau)}$ from $p(z|x)$
- inspect sample statistics
- asymptotically exact
- computationally expensive
- tricky engineering concerns
- no convergence guarantees
- no need parametric form
Outline

1 Introduction

2 Learning Algorithms
   - 2.1. Machine learning
   - 2.2. EM algorithm
   - 2.3. Approximate inference
   - 2.4. Bayesian learning

3 Learning Models

4 Deep Learning

5 Case Studies

6 Future Direction
Challenges in model-based approach

Thomas Bayes (1701-1761)

- We are facing the challenges of big data
- An enormous amount of multimedia data is available in internet which contains speech, text, image, music, video, social networks and any specialized technical data
- The collected data are usually noisy, non-labeled, non-aligned, mismatched, and ill-posed
- Probabilistic models may be improperly-assumed, over-estimated, or under-estimated
Uncertainty modeling

- We need tools for **modeling**, **analyzing**, **searching**, **recognizing** and **understanding** real-world data.
- Our modeling tools should:
  - faithfully represent **uncertainty** in model structure and its parameters.
  - reflect **noise** condition in observed data.
  - be automated and **adaptive**.
  - assure **robustness**.
  - be **scalable** for large data sets.
- Uncertainty can be properly expressed by **prior distribution** or **process**.
Regularization refers to a process of introducing additional information in order to solve the **ill-posed** problem or to prevent **overfitting**.

**Occam’s razor** is imposed to deal with the issue of **model selection**.

**Scalable modeling**
Bayesian speaker recognition

- Real-world speaker recognition
  - unsupervised learning
  - number of factors is unknown
  - very short enrollment utterance
  - high inter/intra speaker variabilities
  - variabilities from channel and noise

- Why Bayesian? [Watanabe and Chien, 2015]
  - exploration for latent variables
  - model regularization
  - uncertainty modeling
  - approximate Bayesian inference
  - better prediction
Outline

1 Introduction

2 Learning Algorithms

3 Learning Models
   3.1. GMM-UBM system
   3.2. Joint factor analysis
   3.3. Probabilistic linear discriminant analysis
   3.4. Support vector machine
   3.5. Restricted Boltzmann machine

4 Deep Learning

5 Case Studies

6 Future Direction
Outline

1. Introduction

2. Learning Algorithms

3. Learning Models
   - 3.1. GMM-UBM system
   - 3.2. Joint factor analysis
   - 3.3. Probabilistic linear discriminant analysis
   - 3.4. Support vector machine
   - 3.5. Restricted Boltzmann machine

4. Deep Learning

5. Case Studies

6. Future Direction
GMM distribution from three Gaussians
Gaussian mixture model

- Gaussian mixture model (GMM) is a weighted sum of Gaussians

\[
p(x|\theta) = \sum_{i=1}^{M} \pi_i b_i(x)
\]

\[
\theta = \{\pi_i, u_i, \Sigma_i\}
\]

- \(\pi_i\): mixture weight
- \(u_i\): mixture mean vector
- \(\Sigma_i\): mixture covariance matrix

\[
b_i(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(x - u_i)^\top \Sigma_i^{-1}(x - u_i)\right)
\]

- Mixture component \(z_i\) is a latent variable which is either zero or one
In ML estimation, we need to
- compute the likelihood of a sequence of features given a GMM
- estimate the parameters of GMM given a set of feature vectors

Assuming independence between features in a sequence, we have

$$p(X|\theta) = p(x_1, \ldots, x_T|\theta) = \prod_{t=1}^{T} p(x_t|\theta)$$

ML estimation is performed by

$$\theta_{ML} = \arg \max_{\theta} p(X|\theta) = \arg \max_{\theta} \sum_{t=1}^{T} \log \left[ \sum_{i=1}^{M} \pi_i b_i(x_t) \right]$$
Parameter estimation

- **E-step** is to calculate the auxiliary function

\[
Q(\theta, \theta^{\text{old}}) = \mathbb{E}_z[\log p(X, z|\theta)|X, \theta^{\text{old}}]
\]

\[
= \sum_{t=1}^{T} \sum_{i=1}^{M} p(z_{ti} = 1|X, \theta^{\text{old}}) \log p(x_t, z_{ti} = 1|\theta)
\]

- **ML estimates** are obtained via M-step as

\[
\pi_{i}^{\text{new}} = \frac{T_i}{T}
\]

\[
\mu_{i}^{\text{new}} = \frac{1}{T_i} \sum_{t=1}^{T} \gamma(z_{ti}) x_t = \frac{1}{T_i} \mathbb{E}_i[x]
\]

\[
\Sigma_{i}^{\text{new}} = \frac{1}{T_i} \sum_{t=1}^{T} \gamma(z_{ti})(x_t - \mu_i)(x_t - \mu_i)^\top = \frac{1}{T_i} \mathbb{E}_i[xx^\top] - \mu_i\mu_i^\top
\]

where \( T_i = \sum_t \gamma(z_{ti}) \) and \( \gamma(z_{ti}) = p(z_{ti} = 1|X, \theta^{\text{old}}) \)
**E-step**

Probabilistically align samples to each mixture

\[
p(z_1 = 1|X) \\
p(z_2 = 1|X) \\
p(z_3 = 1|X) \\
p(z_i = 1|X) = \frac{\pi_i b_i(x)}{\sum_{j=1}^{M} \pi_j b_j(x)}
\]

Accumulate sufficient statistics

\[
T_i = \sum_{t=1}^{T} p(z_{ti} = 1|X) \\
E_i(x) = \sum_{t=1}^{T} p(z_{ti} = 1|X)x_t \\
E_i(xx^T) = \sum_{t=1}^{T} p(z_{ti} = 1|X)x_t x_t^T
\]
Update model parameters

\[ \pi_i^{\text{new}} = \frac{T_i}{T} \]
\[ \mu_i^{\text{new}} = \frac{1}{T_i} \mathbb{E}_i[x] \]
\[ \Sigma_i^{\text{new}} = \frac{1}{T_i} \mathbb{E}_i[xx^T] - \mu_i \mu_i^T \]
Speaker verification

- Realization of log likelihood ratio test from signal detection theory

\[ S_{LR}(X|\theta^{\text{target}}, \theta^{\text{ubm}}) = \log(X|\theta^{\text{target}}) - \log(X|\theta^{\text{ubm}}) \]

- GMMs are used for both target and background models
  - target model trained using enrollment speech
  - universal background model trained using speech from many speakers
Target model & UBM

- **Target model** is adapted from **universal background model (UBM)**
  - good with limited target training data
- **Maximum a posteriori (MAP)** adaptation
  - align target training vectors to UBM
  - accumulate sufficient statistics
  - update target model parameters with smoothing to UBM parameters
- Adaptation for those parameters of **seen** acoustic events
  - sparse regions of feature space filled in by UBM parameters
- **Side benefits**
  - keep correspondence between target and UBM mixtures
  - allow for **fast scoring** when using many target models (top-M scoring)
Maximum a posteriori adaptation

- **Prior density** for GMM mean vector $\mu = \{\mu_i\}$ is introduced
  
  $p(\mu_i) = \mathcal{N}(\mu_i | \mu_i^{ubm}, \sigma^2 I)$

- **MAP estimation** [Gauvain and Lee, 1994] is performed by using the enrollment data $X_s = \{x\}^T_{t=1}$ from a target speaker $s$
  
  - **E-step** is to calculate
    
    $$Q(\mu_i, \mu_i^{old}) = \sum_{t=1}^{T_s} \gamma(z_{ti}) \log p(x_t | z_{ti} = 1, \mu_i) + \log p(\mu_i | \mu_i^{ubm}, \sigma^2 I)$$

  - **M-step** is to maximize $Q(\mu_i, \mu_i^{old})$ to find
    
    $$\mu_i^{new} = \frac{\sum_t \gamma(z_{ti}) x_t}{\sum_t \gamma(z_{ti}) + r} + \frac{r \mu_i^{ubm}}{\sum_t \gamma(z_{ti}) + r} = \alpha_i \mathbb{E}_i[x] + (1 - \alpha_i) \mu_i^{ubm}$$

  where $\alpha_i = \frac{\sum_t \gamma(z_{ti})}{\sum_t \gamma(z_{ti}) + r}$
UBM is based on GMM and trained by using EM algorithm

Speaker GMM is established by adjusting UBM by using MAP adaptation
Speaker recognition procedure

Enrollment
- Feature extractor
- Modeling
- GMM-UBM

Test utterance
- Feature extractor
- Scoring
- Results
Outline

1. Introduction

2. Learning Algorithms

3. Learning Models
   - 3.1. GMM-UBM system
   - 3.2. Joint factor analysis
   - 3.3. Probabilistic linear discriminant analysis
   - 3.4. Support vector machine
   - 3.5. Restricted Boltzmann machine

4. Deep Learning

5. Case Studies

6. Future Direction
Joint factor analysis

- Factor analysis is a statistical method which is used to describe the variability among the observed variables in terms of potentially lower number of unobserved variables called factors.
- Factor analysis is a latent variable model for feature extraction.
- Joint factor analysis (JFA) was the initial paradigm for speaker recognition.

\[
u = m + Vy + Ux + Dz
\]
A supervector for a speaker should be decomposable into speaker independent, speaker dependent, channel dependent, and residual components.

Each component is represented by low-dimensional factors, which operate along the principal dimensions of the corresponding component.

Speaker dependent component, known as the eigenvoice, and the corresponding factors.

\[
V \cdot y = \begin{bmatrix} v_1 & v_2 & \cdots & v_N \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}
\]

Eigenvoice matrix

Low dimensional eigenvoice factors

Each speaker factor controls an eigendimension of the eigenvoice matrix.
GMM supervector $u$ for a speaker can be decomposed as

$$u = m + Vy + Ux + Dz$$

where

- $m$ is a speaker-independent supervector from UBM
- $V$ is the eigenvoice matrix
- $y \sim \mathcal{N}(0, I)$ is the speaker factor vector
- $U$ is the eigenchannel matrix
- $x \sim \mathcal{N}(0, I)$ is the channel factor vector
- $D$ is the residual matrix, and is diagonal
- $z \sim \mathcal{N}(0, I)$ is the speaker-specific residual factor vector
For a 512-mixture GMM-UBM system, the dimensions of each JFA component are typically as follows:

- $V$: 20,000 by 300 (300 eigenvoices)
- $y$: 300 by 1 (300 speaker factors)
- $U$: 20,000 by 100 (100 eigenchannels)
- $x$: 100 by 1 (100 channel factors)
- $D$: 20,000 by 20,000 (20,000 residuals)
- $z$: 20,000 by 1 (20,000 speaker-specific residuals)

These dimensions have been empirically determined to produce the best results. Bayesian model selection can help. Judge by the marginal likelihood over latent component under different dimensions.
Training procedure

- We train the JFA matrices in the following order [Kenny et al., 2007a]
  1. Train the eigenvoice matrix $V$, assuming that $U$ and $D$ are zero
  2. Train the eigenchannel matrix $U$ given the estimate of $V$, assuming that $D$ is zero
  3. Train the residual matrix $D$ given the estimates of $V$ and $U$

- Using these matrices, we compute $y$ for speaker, $x$ for channel, and $z$ for residual factors

- We compute the final score by using these matrices and factors
Subspaces $U$ and $V$ are not completely independent.

A combined total variability space was used [Dehak et al., 2011]

$$u = m + Vy + Ux + Dz$$

$$u = m + Tw$$

Intermediate/identity vector (i-vector)
Training total variability space

- Rank of $\mathbf{T}$ is set prior to training
- $\mathbf{T}$ and $\mathbf{w}$ are latent variables
- EM algorithm is used
- Training total variability matrix $\mathbf{T}$ is similar to training $\mathbf{V}$ except that training $\mathbf{T}$ is performed by using all utterances from a given speaker but as produced by different speakers
- Random initialization for $\mathbf{T}$
- Each $\mathbf{o}_t$ has dimension $D$. Number of Gaussian components is $M$. Dimension of supervector is $M \cdot D$
- UBM diagonal covariance matrix $\mathbf{\Sigma}$ ($MD \times MD$) is introduced to model the residual variability not captured by $\mathbf{T}$
Sufficient statistics

- **$0^{th}$ order statistics** $N_c(u) = \sum_t \gamma_c(o_t)$ of an utterance $u$
- **$1^{st}$ order statistics** $F_c(u) = \sum_t \gamma_c(o_t) o_t$
- **$2^{nd}$ order statistics** $S_c(u) = \text{diag}\left(\sum_t \gamma_c(o_t) o_t o_t^\top\right)$ where

  \[
  \gamma_c(o_t) = p(c|o_t, \theta_{ubm}) = \frac{\pi_c p(o_t|m_c, \Sigma_c)}{\sum_{j=1}^M \pi_i p(o_t|m_j, \Sigma_j)}
  \]

- **Centralized $1^{st}$ and $2^{nd}$ order statistics**

  \[
  \tilde{F}_c(u) = \sum_{t=1}^T \gamma_c(o_t)(o_t - m_c)
  \]

  \[
  \tilde{S}_c(u) = \text{diag}\left(\sum_{t=1}^T \gamma_c(o_t)(o_t - m_c)(o_t - m_c)^\top\right)
  \]

  where $m_c$ is the subvector corresponding to mixture component $c$
EM algorithm

- Sufficient statistics

\[
N(u) = \begin{bmatrix}
N_1(u) \cdot I_{D \times D} & 0 & \cdots & 0 \\
0 & N_2(u) \cdot I_{D \times D} & 0 & \cdots \\
\vdots & 0 & \ddots & 0 \\
0 & \cdots & 0 & N_M(u) \cdot I_{D \times D}
\end{bmatrix}
\]

\[
\tilde{F}(u) = \begin{bmatrix}
\tilde{F}_1(u) \\
\tilde{F}_2(u) \\
\vdots \\
\tilde{F}_M(u)
\end{bmatrix}
\]

- EM algorithm [Kenny et al., 2005]
  - Initialize \(m, \Sigma\) and \(T\)
  - E-step: for each utterance \(u\), calculate the parameters of the posterior distribution of \(w(u)\) using the current estimates of \(m, \Sigma, T\)
  - M-step: update \(T\) and \(\Sigma\) by solving a set of linear equations in which \(w(u)\)'s play the role of explanatory variables
  - Iterate until data likelihood given the estimated parameters converges
E-step: posterior distribution of $w(u)$

- For each utterance $u$, we calculate the matrix

$$L(u) = I + T^\top \Sigma^{-1} N(u) T$$

- Posterior distribution of $w(u)$ conditioned on the acoustic observations of an utterance $u$ is Gaussian with mean

$$\mathbb{E}[w(u)] = L^{-1}(u) T^\top \Sigma^{-1} \tilde{F}(u)$$

and covariance matrix

$$\text{Cov}(w(u), w(u)) = L^{-1}(u)$$

- Variational Bayesian JFA was developed for speaker verification [Zhao and Dong, 2012]
I-vectors from JFA model are used in linear discriminant analysis (LDA)

\[ u = m + Tw \quad \text{Factor analysis} \]

\[ \mathcal{W} = Aw \quad \text{Linear discriminant analysis} \]

Both methods used to reduce the dimensionality of speaker model

\( A \) is chosen such that within-speaker variability \( S_w \) is minimized and between-speaker variability \( S_b \) is maximized within the space

\( A \) is found by eigenvalue method via maximizing

\[ \mathcal{J}(A) = \text{Tr}\{S_w^{-1}S_b\} \]
1 Introduction

2 Learning Algorithms

3 Learning Models
   3.1. GMM-UBM system
   3.2. Joint factor analysis
   3.3. Probabilistic linear discriminant analysis
   3.4. Support vector machine
   3.5. Restricted Boltzmann machine

4 Deep Learning

5 Case Studies

6 Future Direction
Assuming a factor analysis model of the i-vectors of the form

\[ w = u + Fh + \varepsilon \]

- \( w \) is the i-vector, \( u \) is the mean of i-vectors, and \( h \sim \mathcal{N}(0, I) \) is the latent factors.
- First compute the maximum likelihood estimate of the factor loading matrix \( F \), also known as the eigenvoice subspace.
- Full covariance of residual noise \( \varepsilon \) explains the variability not captured through the latent variables.

**PLDA**

Under Gaussian assumption, this model is known in face recognition as PLDA [Prince and Elder, 2007]
Gaussian PLDA

- Assume that there are low dimensional, normally distributed hidden variables $x_1$ and $x_{2r}$ such that

$$D_r = m + \underbrace{U_1 x_1}_{S} + \underbrace{U_2 x_{2r} + \varepsilon_r}_{C_r}$$

- Residual $\varepsilon_r$ is normally distributed with mean 0 and precision matrix $\Lambda$
- $m$ is the center of acoustic space and $x_1$ is the speaker factors
- Columns of $U_1$ are the *eigenvoices*

$$\text{Cov}(S, S) = U_1 U_1^\top$$

- $x_{2r}$ varies from one recording to another (channel factors)
- Columns of $U_2$ are the *eigenchannels*

$$\text{Cov}(C_r, C_r) = \Lambda^{-1} + U_2 U_2^\top$$
Including $x_{2r}$ enables the decomposition of speaker and channel factors.

$x_{2r}$ can *always* be eliminated at recognition time.

Between-speaker covariance matrix $\text{Cov}(S, S)$ & within-speaker covariance matrix $\text{Cov}(C_r, C_r)$

These matrices cannot be treated as full rank.
Given two i-vectors $D_1$ and $D_2$, we would like to perform the hypothesis test

$H_1$: the speakers are the same

$H_0$: the speakers are different

Likelihood ratio is calculated by

$$\frac{p(D_1, D_2 | H_1)}{p(D_1 | H_0)p(D_2 | H_0)}$$

Likelihood ratio for any type of speaker recognition or speaker clustering problem

The evidence integral should be calculated

$$\int p(D, z)dz$$
Model assumption

Assume that

- we have succeeded in estimating the model parameters \( \theta = \{ \mathbf{m}, U_1, U_2, \Lambda \} \)
- given a collection \( D = (D_1, \ldots, D_R) \) of i-vectors associated with a speaker, we have figured out how to evaluate the marginal likelihood or the evidence

\[
p(D) = \int p(D, z) dz = \int p(D|z)p(z)dz
\]

- \( z = \{ \mathbf{x}_1, \mathbf{x}_{2r} \} \) is the hidden variables associated with the speaker

We show how to do speaker recognition in this situation and how both problems are tackled by using variational Bayes to approximate the posterior distribution \( p(z|D) \).
Variational approximation

- Evidence $p(D)$ can be evaluated exactly in the Gaussian case but this involves inverting the large sparse block matrices.
- If $q(z)$ is any distribution on $z$, variational lower bound is yielded as

$$L \triangleq \mathbb{E}_q \left[ \log \frac{p(D, z)}{q(z)} \right]$$

where $\log p(D) \geq L$ with equality iff $q(z) = p(z|D)$.

- Variational Bayes provides a principled way to find a good approximation $q(z)$ to $p(z|D)$.
- Model parameters $\theta = \{m, U_1, U_2, \Lambda\}$ are estimated by maximizing the evidence lower bound (ELBO) $L$ which is calculated over all of the speakers in a training set.
Bayesian speaker recognition

- **Full Bayesian** avoids the point estimates of model parameters
- **Coupling** of multiple latent variables is tackled in VB [Villalba and Lleida, 2014]
- Uncertainties are compensated for model regularization in speaker recognition
- **Prior densities** $p(U_1)$ and $p(U_2)$ can be flexibly incorporated
- Selection for the number of speaker factors or channel factors
- Manually tuning for unknown variables is avoided
- Analogous to the treatment of the number of mixture components in Bayesian estimation of GMM
- **Bayesian mixture of PLDA** [Mak et al., 2016] was recently developed
Outline

1. Introduction

2. Learning Algorithms

3. Learning Models
   - 3.1. GMM-UBM system
   - 3.2. Joint factor analysis
   - 3.3. Probabilistic linear discriminant analysis
   - 3.4. Support vector machine
   - 3.5. Restricted Boltzmann machine

4. Deep Learning

5. Case Studies

6. Future Direction
What is a good decision boundary?

- Consider a **two-class, linearly separable** classification problem
- Many decision boundaries!
  - perceptron algorithm can be used to find such a boundary
  - different algorithms have been proposed
- Are all decision boundaries equally good?
Large-margin decision boundary

- Decision boundary should be as far away from the data of both classes as possible [Vapnik, 2013]
  - we should maximize the margin \( m \)
  - distance between the origin and the line \( \mathbf{w}^\top \mathbf{x} = k \) is \( \frac{k}{\|\mathbf{w}\|} \)

\[ \mathbf{w}^\top \mathbf{x} + b = -1 \]
\[ \mathbf{w}^\top \mathbf{x} + b = 0 \]
\[ \mathbf{w}^\top \mathbf{x} + b = 1 \]
Finding the decision boundary

• \{x_1, \ldots, x_n\} is the data set and \(y_i \in \{1, -1\}\) is the class label of \(x_i\)
• Decision boundary should classify all points correctly

\[ y_i (w^\top x_i + b) \geq 1, \quad \text{for } i = 1, \ldots, n \]

• Decision boundary can be found by solving the constrained optimization problem

Minimize \(\frac{1}{2}||w||^2\)

subject to \(y_i (w^\top x_i + b) \geq 1, \quad \text{for } i = 1, \ldots, n\)
Constrained optimization

- Original problem

\[
\begin{align*}
\text{Minimize} & \quad \frac{1}{2} \| \mathbf{w} \|^2 \\
\text{subject to} & \quad y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \text{for } i = 1, \ldots, n
\end{align*}
\]

- We introduce the Lagrange multipliers \( \alpha_i \geq 0 \) to form the Lagrangian function

\[
L = \frac{1}{2} \| \mathbf{w} \|^2 - \sum_{i=1}^{n} \alpha_i \left( y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right)
\]

- Setting the gradient of \( L \) w.r.t \( \mathbf{w} \) and \( b \) to zero, we have

\[
\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i
\]

\[
\sum_{i=1}^{n} \alpha_i y_i = 0
\]
New objective function is expressed in terms of $\alpha_i$ only
If we know $w$, we know all $\alpha_i$. If we know all $\alpha_i$, we know $w$
Original problem is known as the primal problem
A quadratic programming objective is formed by

$$
\text{maximize } L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j
$$

subject to $\alpha_i \geq 0$, $\sum_{i=1}^{n} \alpha_i y_i = 0$

Global maximum of $\alpha_i$ can be found
Characteristics of the solution

- Many of $\alpha_i$ are zero
  - $w$ is a linear combination of a small number of data points
  - This sparse representation can be viewed as data compression as in the construction of KNN classifier

- $x_i$ with non-zero $\alpha_i$ are called support vectors
  - decision boundary is determined only by support vectors
  - $t_j, j = 1, \ldots, s$ are the indices of $s$ support vectors

$$w = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} x_{t_j}$$

- For testing with a new data $z$
  - compute $f = w^T z + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} x_{t_j}^T z + b$ and classify $z$ as class 1 if the sum is positive, and class 2 otherwise
Non-separable problem

- We allow error $\xi_i$ in classification. It is based on the output of the discriminant function $\mathbf{w}^\top \mathbf{x} + b$
- $\xi_i$ approximates the number of misclassified samples
If we minimize $\sum_i \xi_i$, $\xi_i$ can be computed by

$$\begin{cases} 
w^\top x_i + b \geq 1 - \xi_i & y_i = 1 \\
w^\top x_i + b \leq -1 + \xi_i & y_i = -1 \\
\xi_i \geq 0 & \forall i
\end{cases}$$

- $\xi_i$ are slack variables in optimization
- $\xi_i = 0$ if there is no error for $x_i$
- $\xi_i$ is an upper bound of the number of errors

We want to minimize $\frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$
- $C$ is a tradeoff parameter between error and margin

Optimization problem becomes

$$\text{Minimize } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$$

subject to $y_i(w^\top x_i + b) \geq 1 - \xi_i, \; \xi_i \geq 0$
Dual problem

- Dual of this new constrained optimization problem is

\[
\text{maximize } L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \! \cdot \! x_j
\]

subject to \( C \geq \alpha_i \geq 0 \), \( \sum_{i=1}^{n} \alpha_i y_i = 0 \)

- \( w \) is recovered as \( w = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} x_{t_j} \)

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound \( C \) on \( \alpha_i \) now

- Once again, a quadratic programming solver can be used to find \( \alpha_i \)
So far, we have only considered large-margin classifier with a linear decision boundary.

How to generalize it to become non-linear?

Key idea: transform $x_i$ to a higher dimensional space to make life easier.

- input space: the points $x_i$ are located
- feature space: the space of $\phi(x_i)$ after transformation

Why transform?

- linear operation in the feature space is equivalent to non-linear operation in input space
- classification can become easier with a proper transformation.
- In the XOR problem, for example, adding a new feature make the problem linearly separable.
Dimensionality in feature space

- Computation in the feature space can be costly because it is highly dimensional
  - feature space is typically infinite-dimensional!
- Kernel trick comes to rescue
Kernel trick

maximize \( L(\alpha) = \sum_{i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j \)

subject to \( C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0 \)

- \( x_i^\top x_j \) is inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations, e.g. angle, distance, can be expressed by inner products
- Define the kernel function \( K \) by

\[
K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)
\]
Kernel function in SVM

- Change all inner products to kernel functions
- For training
  - original

\[
\text{maximum } L(\alpha) = \sum_{i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j \\
\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0
\]

- with kernel function

\[
\text{maximum } L(\alpha) = \sum_{i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\
\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0
\]
Kernel function in SVM

- For testing, the new data \( \mathbf{z} \) is classified as class 1 if \( f \geq 0 \) and as class 2 if \( f < 0 \)
  
  - original

\[
\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}
\]

\[
f = \mathbf{w}^\top \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^\top \mathbf{z} + b
\]

- with kernel function

\[
\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})
\]

\[
f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b
\]
Since the training of SVM only requires the value of \( K(x_i, x_j) \), there is no restriction of the form of \( x_i \) and \( x_j \):

- \( x_i \) can be a sequence or a tree instead of a feature vector

\( K(x_i, x_j) \) is just a similarity measure comparing \( x_i \) and \( x_j \).

Kernel function needs to satisfy the Mercer function, i.e., the function is positive-definite.

For a test object \( z \), the discriminant function essentially is a weighted sum of the similarity between \( z \) and a pre-selected set of objects, also called the support vectors:

\[
f(z) = \sum_{x_i \in S} \alpha_i y_i K(z, x_i) + b
\]

where \( S \) denotes the set of support vectors.
Outline

1. Introduction

2. Learning Algorithms

3. Learning Models
   - 3.1. GMM-UBM system
   - 3.2. Joint factor analysis
   - 3.3. Probabilistic linear discriminant analysis
   - 3.4. Support vector machine
   - 3.5. Restricted Boltzmann machine

4. Deep Learning

5. Case Studies

6. Future Direction
Restricted Boltzmann machine

- Bipartite the undirected graphical model with visible variable \( \mathbf{v} \) and hidden variable \( \mathbf{h} \)
- Building-block for deep belief networks and deep Boltzmann machines
- RBMs are generative models of \( \mathbf{v} \) based on the marginal distribution
- Joint distribution of \( (\mathbf{v}, \mathbf{h}) \) is an exponential family
- Discriminative fine-tuning can be applied
- Variables are typically binary, however no such restriction exists
- Bidirectional graphical model
Graphical model

- No connection between nodes of the same layer (i.e. sparsity)
- Allow fast training (blocked-Gibbs sampling)
- Correlations between nodes in \( v \) are still present in the marginal \( p(v|W) \)
- Hidden variable \( h \) captures the higher level information
Energy-based distribution is defined by

\[ p(v, h|\theta) = Z(\theta)^{-1} p^*(v, h|\theta) \]

where

\[ p^*(v, h|\theta) = \exp\left(\sum_i v_i b_i + \sum_j h_j a_j + \sum_{i,j} v_i h_j W_{ij} - E(v, h)\right) \]

and \( \theta = \{W, b, a\} \)

\( \{b, a\} \) denote the biases and are usually assumed to be zero for compact notation

\( Z(\theta) = \sum_{v,h} p^*(v, h|\theta) \) is the partition function
Consider binary \((v, h)\) with zero biases \(\{b, a\}\)

\[
p(h|v) = \prod_j p(h_j|v) \quad \text{where} \quad p(h_j|v) = \frac{1}{1 + \exp(-\sum_i v_i W_{ij})}
\]

\[
p(v|h) = \prod_i p(v_i|h) \quad \text{where} \quad p(v_i|h) = \frac{1}{1 + \exp(-\sum_j h_j W_{ij})}
\]

Product form is due to the restricted structure
Maximize the likelihood of $\theta$ given $\mathbf{v}^n$

$$p(\mathbf{v}^n) = \prod_n Z^{-1} \sum_h \exp \left( \sum_{ij} v_i^n h_j^n W_{ij} \right)$$

We obtain

$$\frac{\partial \log p(\mathbf{v}^n)}{\partial W_{ij}} = E_{p_{data}}[v_i h_j] - E_{p_{model}}[v_i h_j]$$

where $p_{data}(\mathbf{v}^n, \mathbf{h}^n) = p(\mathbf{h}^n|\mathbf{v}^n)p(\mathbf{v}^n)$

$p(\mathbf{h}^n|\mathbf{v}^n)$ is an easy and exact calculation for RBM

$p(\mathbf{v}^n)$ is an empirical distribution

$$\frac{\partial \log Z(\theta)}{\partial W_{ij}} = E_{p_{model}}[v_i h_j]$$ is hard to compute

Learning using contrastive divergence with mini-batches is performed
Outline

1. Introduction

2. Learning Algorithms

3. Learning Models

4. Deep Learning
   - 4.1. Deep neural network
   - 4.2. Deep belief network
   - 4.3. Stacking auto-encoder
   - 4.4. Variational auto-encoder
   - 4.5. Deep transfer learning

5. Case Studies

6. Future Direction
Outline

1. Introduction

2. Learning Algorithms

3. Learning Models

4. Deep Learning
   - 4.1. Deep neural network
   - 4.2. Deep belief network
   - 4.3. Stacking auto-encoder
   - 4.4. Variational auto-encoder
   - 4.5. Deep transfer learning

5. Case Studies

6. Future Direction
Neural network

Multilayer perceptron
Nonlinear activation

\[ y_{tk} = y_k(x_t, w) = f \left( \sum_{j=0}^{M} w^{(2)}_{jk} f \left( \sum_{i=0}^{D} w^{(1)}_{ij} x_{ti} \right) \right) \]
Deep learning

- **Deep belief networks** (DBN) obtained great results due to good initialization and deep model structure
  - pre-train each layer from bottom up
  - each pair of layers is a restricted Boltzmann machine
  - jointly fine tune all layers using back-propagation

- **Deep neural network** (DNN)
  - discriminative model works for classification tasks
  - empirically works well for image recognition, speech recognition, information retrieval and many others
  - no theoretical guarantee
DBN-DNN training
Why go deep?

- Deep architecture can be representationally efficient
  - fewer computational units for the same function
- Deep representation might allow for a hierarchical representation
  - allows non-local generalization
  - comprehensibility
- Multiple levels of latent variables allow combinatorial sharing of statistical strength
- Deep architecture works well for representation of vision, audio, NLP, music and many other technical data
Different level of abstraction

- **Hierarchical learning**
  - natural progression from *low* level to *high* level structure as seen in natural complexity
  - easier to monitor what is being learnt and to guide the machine to better *subspaces*
  - a good lower level representation can be used in different tasks
Trainable feature hierarchy

- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable feature transform
- Image
  - Pixel $\rightarrow$ edge $\rightarrow$ texton $\rightarrow$ motif $\rightarrow$ part $\rightarrow$ object
- Text
  - Character $\rightarrow$ word $\rightarrow$ word group $\rightarrow$ clause $\rightarrow$ sentence $\rightarrow$ story
- Speech
  - Sample $\rightarrow$ spectral band $\rightarrow$ sound $\rightarrow$ ... $\rightarrow$ phone $\rightarrow$ phoneme $\rightarrow$ word
Deep architecture

- **Feed-forward**: multilayer neural nets, convolutional nets

- **Feed-back**: stacked sparse coding, deconvolutional nets

- **Bi-directional**: deep Boltzmann machines, **stacked auto-encoders**
Training strategy

- **Purely supervised**
  - initialize parameters randomly
  - train in supervised mode
    - typically with SGD, using backprop to compute gradients
  - used in most practical systems for speech and image recognition

- **Unsupervised, layerwise + supervised classifier on top**
  - train each layer unsupervised, one after the other
  - train a supervised classifier on top, keeping the other layers fixed
  - good when very few labeled samples are available

- **Unsupervised, layerwise + global supervised fine-tuning**
  - train each layer unsupervised, one after the other
  - add a classifier layer, and retrain the whole thing supervised
  - good when label set is poor

- **Unsupervised pre-training often uses the regularized auto-encoders**
Generalizable learning

- **Shared representation**
  - multi-task learning
  - unsupervised training

- **Partial feature sharing**
  - mixed mode learning
  - composition of functions

![Diagram showing shared intermediate representation and partial feature sharing through tasks and outputs.](image-url)
Forward & backward passes

- **Forward propagation**
  - sum inputs, produce activation, feed-forward

- **Training: back propagation of error**
  - calculate total error at the top
  - calculate contributions to error at each step going backwards
Deep neural network

- Simple to construct
  - sigmoid nonlinearity for hidden layers
  - softmax for the output layer
- Backpropagation does not work well if randomly initialized
  [Bengio et al., 2007]
  - deep networks trained without unsupervised pretraining perform worse than shallow networks

<table>
<thead>
<tr>
<th>Model</th>
<th>train</th>
<th>valid</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN, unsupervised pre-training</td>
<td>0%</td>
<td>1.2%</td>
<td>1.2%</td>
</tr>
<tr>
<td>Deep net, auto-associator pre-training</td>
<td>0%</td>
<td>1.4%</td>
<td>1.4%</td>
</tr>
<tr>
<td>Deep net, supervised pre-training</td>
<td>0%</td>
<td>1.7%</td>
<td>2.0%</td>
</tr>
<tr>
<td>Deep net, no pre-training</td>
<td>.004%</td>
<td>2.1%</td>
<td>2.4%</td>
</tr>
<tr>
<td>Shallow net, no pre-training</td>
<td>.004%</td>
<td>1.8%</td>
<td>1.9%</td>
</tr>
</tbody>
</table>

(Bengio et al., NIPS 2007)
Problems and solvers with back propagation

- Gradient is progressively getting more dilute
  - below top few layers, correction signal is minimal
- Gets stuck in local minima
  - random initialization: may start out far from good regions
- In usual settings, we can use only labeled data
  - almost all data are unlabeled
  - the brain can learn from unlabeled data
- Use unsupervised learning via greedy layer-wise training
  - allow abstraction to develop naturally from one layer to another
  - help the network initialize with good parameters
- Perform supervised top-down training as final step
  - refine the features in intermediate layers more relevant for the task
Outline

1 Introduction

2 Learning Algorithms

3 Learning Models

4 Deep Learning
   - 4.1. Deep neural network
   - 4.2. Deep belief network
   - 4.3. Stacking auto-encoder
   - 4.4. Variational auto-encoder
   - 4.5. Deep transfer learning

5 Case Studies

6 Future Direction
Deep belief network (DBN) is a probabilistic generative model

Deep architecture with multiple hidden layers

**Unsupervised pre-learning** provides a good initialization
- maximizing the **lower-bound** of the log-likelihood of data

**Supervised fine-tuning**
- generative: up-down algorithm
- discriminative: back propagation
Model structure

Hidden Layers

Visible Layers

\[ p(v, h^1, h^2, \ldots, h^l) = p(v|h^1)p(h^1|h^2) \ldots p(h^{l-2}|h^{l-1})p(h^{l-1}|h^l) \]
Greedy training

- First step:
  - construct an RBM with an input layer $\mathbf{v}$ and a hidden layer $\mathbf{h}$
  - train the RBM
Greedy training

- Second step:
  - Stack another hidden layer on top of the RBM to form a new RBM
  - Fix $W^1$, sample $h^1$ from $q(h^1|v)$ as input. Train $W^2$ as RBM
Greedy training

- Third step:
  - continue to stack layers on top of the network, train it as previous step, with sample sampled from $q(h^2|h^1)$

- And so on...
Deep Boltzmann machine

\[ p(\mathbf{v}) = \sum_{\mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3} \frac{1}{Z} \exp[\mathbf{v}^T \mathbf{W}^1 \mathbf{h} + (\mathbf{h}^1)^T \mathbf{W}^2 \mathbf{h}^2 + (\mathbf{h}^2)^T \mathbf{W}^3 \mathbf{h}^3] \]

- **Undirected connections** between all layers. No connections between the nodes in the same layer
- **High-level** representations are built from unlabeled inputs. Labeled data is used to only slightly fine-tune the model

[Salakhutdinov and Hinton, 2009]
Training procedure

- **Pre-training**
  - initialize from stacked RBMs

- **Generative** fine-tuning
  - positive phase: variational or mean-field approximation
  - negative phase: persistent chain & stochastic approximation

- **Discriminative** fine-tuning
  - back-propagation
Why greedy layer wise training works

- **Regularization** hypothesis
  - pre-training is *constraining* the parameters in a region relevant to unsupervised dataset
  - better *generalization* - representations that better describe unlabeled data are more discriminative for labeled data

- **Optimization** hypothesis
  - unsupervised training initializes lower level parameters near localities of better *minima* than random initialization can
Outline

1 Introduction

2 Learning Algorithms

3 Learning Models

4 Deep Learning
   - 4.1. Deep neural network
   - 4.2. Deep belief network
   - 4.3. Stacking auto-encoder
   - 4.4. Variational auto-encoder
   - 4.5. Deep transfer learning

5 Case Studies

6 Future Direction
Denoising auto-encoder

- **Corrupt** the input, e.g. set 25% of inputs to 0
- **Reconstruct** the uncorrupted input
- Use the uncorrupted encoding as input to next level

[Vincent et al., 2008]
Manifold learning perspective

- Learn a vector field towards higher probability regions
- Minimize the variational lower bound on a generative model
- Correspond to the regularized score matching on an RBM
Stacked denoising auto-encoders
Greedy layer-wise learning

- Start with the lowest level and stack upwards
- Train each layer of auto-encoder using the intermediate codes or features from the layer below
- Top layer can have a different output, e.g. softmax non-linearity, to provide an output for classification
Outline

1 Introduction

2 Learning Algorithms

3 Learning Models

4 Deep Learning
   - 4.1. Deep neural network
   - 4.2. Deep belief network
   - 4.3. Stacking auto-encoder
   - 4.4. Variational auto-encoder
   - 4.5. Deep transfer learning

5 Case Studies

6 Future Direction
Auto-encoder
Variational auto-structure
[Kingma and Welling, 2014]

- Mean-field approach requires analytical solutions $\mathbb{E}_q$, which are intractable in the case of neural network.
- Use neural network and sample the latent variables $z$ from variational posterior.
Variational Bayesian inference aims to find a variational distribution $q(z|x)$ that is maximally close to the original true posterior distribution $p(z|x)$.

According to the evidence decomposition, we have

$$p(x) = \mathcal{L}(q) + KL(q||p)$$

$$\mathcal{L}(q) = \mathbb{E}_q[\log p(x, z)] + \mathbb{H}_q[z]$$

$$KL(q||p) = -\mathbb{E}_q[\log p(z|x)] - \mathbb{H}_q[z]$$
Mean field variational inference

- Assume that $q(z|x)$ can be factorized into the product of individual probability distributions

$$q(z|x) = \prod_{n=1}^{N} q(z_n|x_n)$$

- We can perform the coordinate ascent for each factorized variational distributions by

$$\hat{q}(z_j|x_j) \propto \exp(\mathbb{E}_{q(z_{i\neq j})}[\log p(x, z)])$$
Model parameters are learned by maximizing the variational lower bound

\[
\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \text{KL}(q_\phi(z|x) || p_\omega(z)) \\
= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z|x)] \\
\triangleq \mathbb{E}_{q_\phi(z|x)}[f_\Theta(x, z)] \\
\triangleq \mathcal{L}_\Theta
\]

where \( \Theta = \{\theta, \phi, \omega\} \)
Stochastic backpropagation

Objective:

\[ L_\theta = \mathbb{E}_{q_\phi(z|x)}[f_\theta(x, z)] \]

Gradient:

Step 1

sample \( z^{(l)} \) from \( q_\phi(z|x) \)

Step 2

\[ L_\theta \simeq f_\theta(x|z^{(l)}) \]

Step 3

\[ \nabla_\theta L_\theta \simeq \nabla_\theta f_\theta(x, z^{(l)}) \]

- Problem: high variance by directly sampling \( z \) [Rezende et al., 2014]
Stochastic gradient variational Bayes

Objective:
\[ \mathcal{L}_\theta = \mathbb{E}_{q_\phi(z|x)}[f_\theta(x, z)] \]

Gradient:

Step 1: Sample \( \epsilon^{(l)} \) from \( \mathcal{N}(0, I) \)

Step 2: \( z^{(l)} = \mu_z + \sigma_z \odot \epsilon^{(l)} \)

Step 3: \( \mathcal{L}_\theta \simeq f_\theta(x|z^{(l)}) \)

Step 4: \( \nabla_\theta \mathcal{L}_\theta \simeq \nabla_\theta f_\theta(x, z^{(l)}) \)

- Reduce the variance caused by directly sampling \( z \)
Outline

1 Introduction

2 Learning Algorithms

3 Learning Models

4 Deep Learning
   4.1. Deep neural network
   4.2. Deep belief network
   4.3. Stacking auto-encoder
   4.4. Variational auto-encoder
   4.5. Deep transfer learning

5 Case Studies

6 Future Direction
Why transfer learning?

- **Mismatch** between training and test data in speaker recognition always exists
- Traditional machine learning works well under an assumption that training and test data follow the same distribution
  - real-world data may not follow this assumption
- **Feature-based domain adaptation** is a common approach
  - allow knowledge to be transferred across domains through learning a good feature representation
- Co-train for feature representation and speaker recognition without labeling in target domain
Transfer learning

- Let $\mathcal{D} = \{\mathcal{X}, p(X)\}$ denote a domain
  - feature space $\mathcal{X}$
  - marginal probability distribution $p(X)$
  - $X = \{x_1, \cdots, x_T\} \subset \mathcal{X}$

- Let $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$ denote a task
  - label space $\mathcal{Y}$
  - objective predictive function $f(\cdot)$
    can be written as $p(Y|X)$

- Assumptions in transfer learning
  - source and target domains are different $\mathcal{D}_S \neq \mathcal{D}_T$
  - source and target tasks are different $\mathcal{T}_S \neq \mathcal{T}_T$
Multi-task learning

$$\min_{\theta} \ell(D, \theta) + \lambda \Omega(\theta)$$

Joint Learning Task

<table>
<thead>
<tr>
<th>Input</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>{apple, not apple}</td>
</tr>
<tr>
<td>pear</td>
<td>{pear, not pear}</td>
</tr>
</tbody>
</table>

Main Task | Auxiliary Task

Joint Learning Task

Input

apple

pear
Multi-task neural network learning

\[
\min_{\theta} \ell(D, \theta) + \lambda \Omega(\theta)
\]
Semi-supervised learning is conducted under multiple objectives.
Outline

1 Introduction

2 Learning Algorithms

3 Learning Models

4 Deep Learning

5 Case Studies
   5.1. Heavy-Tailed PLDA
   5.2. SNR-Invariant PLDA
   5.3. Mixture of PLDA
   5.4. DNN I-vectors
   5.5. PLDA with RBM

6 Future Direction
Outline

1. Introduction

2. Learning Algorithms

3. Learning Models

4. Deep Learning

5. Case Studies
   - 5.1. Heavy-Tailed PLDA
   - 5.2. SNR-Invariant PLDA
   - 5.3. Mixture of PLDA
   - 5.4. DNN I-vectors
   - 5.5. PLDA with RBM

6. Future Direction
Motivation

- **Motivation of i-vectors:**
  - Insufficiency of joint factor analysis (JFA) in distinguishing between speaker and channel information, as channel factors were shown to contain speaker information.
  - Better to use a two-step approach: (1) use low-dimensional vectors (called i-vectors) that comprise both speaker and channel information to represent utterances; and (2) model the channel and variabilities of the i-vectors during scoring.

- **Motivation of Heavy-tailed PLDA:**
  - JFA assumes that the speaker and channel components follow Gaussian distributions.
  - The Gaussian assumption prohibits large deviations from the mean.
  - But speaker effects (e.g., non-native speakers) and channel effect (gross channel distortion) could cause large deviations.
  - Use heavy-tailed distributions instead of Gaussians for modeling the speaker and channel components in i-vectors [Kenny, 2010].
Assuming that we have $H_i$ i-vectors $\mathcal{X}_i = \{x_{ij}, j = 1, \ldots, H_i\}$ from speaker $i$, the generative model is

$$x_{ij} = m + Vh_i + Gr_{ij} + \epsilon_{ij}$$

where $V$ and $G$ represent the the speaker and channel subspaces, respectively.

In heavy-tailed PLDA,

$$h_i \sim \mathcal{N}(0, u_1^{-1}I) \quad u_1 \sim \mathcal{G}(n_1/2, n_1/2)$$

$$r_{ij} \sim \mathcal{N}(0, u_2^{-1}I) \quad u_2 \sim \mathcal{G}(n_2/2, n_2/2)$$

$$\epsilon_{ij} \sim \mathcal{N}(0, (v_j \Lambda)^{-1}) \quad v_j \sim \mathcal{G}(v_j/2, v_j/2)$$

By integrating out the hyperparameters ($u_1$, $u_2$, and $v_j$), one can show [Eq. 2.161 of Bishop (2006)] that the priors of $h_i$, $r_{ij}$, and $\epsilon_{ij}$ follow Student’s $t$. So, $x_{ij}$ also follows Student’s $t$. 
## Performance on NIST 2008 SRE

- **Telephone speech, without score normalization**

<table>
<thead>
<tr>
<th></th>
<th>Gaussian</th>
<th>heavy-tailed</th>
</tr>
</thead>
<tbody>
<tr>
<td>short2-short3</td>
<td>3.6% / 0.014</td>
<td>2.2% / 0.010</td>
</tr>
<tr>
<td>8conv-short3</td>
<td>3.7% / 0.009</td>
<td>1.3% / 0.005</td>
</tr>
<tr>
<td>10sec-10sec</td>
<td>16.4% / 0.070</td>
<td>10.9% / 0.053</td>
</tr>
</tbody>
</table>

- **Microphone speech, with score normalization**

<table>
<thead>
<tr>
<th></th>
<th>partially heavy-tailed</th>
<th>fully heavy-tailed</th>
</tr>
</thead>
<tbody>
<tr>
<td>det1</td>
<td>3.3% / 0.017</td>
<td>3.4% / 0.017</td>
</tr>
<tr>
<td>det4</td>
<td>2.8% / 0.016</td>
<td>3.1% / 0.018</td>
</tr>
<tr>
<td>det5</td>
<td>4.0% / 0.020</td>
<td>3.8% / 0.020</td>
</tr>
</tbody>
</table>

[Kenny, 2010]
In 2011, [Garcia-Romero and Espy-Wilson, 2011] discovered that Gaussian PLDA performs as good as HT-PLDA provided that i-vectors have been subjected to the following pre-processing steps:

**Whitening + Length normalization**

- These steps have the effect of making the i-vectors more Gaussian.
- As Gaussian PLDA is computationally much simpler than HT-PLDA, the former has been extensively used in speaker verification.
Outline

1 Introduction

2 Learning Algorithms

3 Learning Models

4 Deep Learning

5 Case Studies
   5.1 Heavy-Tailed PLDA
   5.2 SNR-Invariant PLDA
   5.3 Mixture of PLDA
   5.4 DNN I-vectors
   5.5 PLDA with RBM

6 Future Direction
Motivation

- While i-vector extraction followed by PLDA is very effective in addressing channel variability.
- Performance degrades rapidly in the presence of background noise with a wide range of signal-to-noise ratios (SNR).
- Classical approach: Multi-condition training where i-vectors from various background noise level are pooled to train a PLDA model.

![Diagram showing i-vectors with 2 SNR ranges, EM Algorithm, and SNR distribution.](image)
We argue that the variation caused by SNR variability can be modeled by an **SNR subspace** and utterances falling within a narrow SNR range should share the same set of SNR factors.

SNR-specific information were **separated from speaker-specific information** through marginalizing out the SNR factors during scoring.
I-vectors derived from utterances of similar SNR will be mapped to a small region in the SNR subspace.
SNR-Invariant PLDA

- Classical PLDA: \( x_{ij} = m + Vh_i + \epsilon_{ij} \)

- By adding an SNR factor to the conventional PLDA, we have **SNR-invariant PLDA** [Li and Mak, 2015]:

\[
x_{ij}^k = m + Vh_i + Uw_k + \epsilon_{ij}^k, \quad k = 1, \ldots, K
\]

where \( U \) denotes the SNR subspace, \( w_k \) is an SNR factor, and \( h_i \) is the speaker (identity) factor for speaker \( i \).

- Note that it is not the same as PLDA with channel subspace:

\[
x_{ij}^k = m + Vh_i + Gr_{ij} + \epsilon_{ij},
\]

where \( G \) defines the channel subspace and \( r_{ij} \) represents the channel factors.
Generative model:

\[ x_{ij}^k = m + Vh_i + Uw_k + \epsilon_{ij}^k, \quad k=1, \ldots, K \]

- \( h_i \) is speaker factors with prior distribution \( \mathcal{N}(0, I) \)
- \( x_{ij}^k \) is the \( j \)-th i-vector from speaker \( i \) in the \( k \)-th SNR subgroup
- \( V \) is the eigenvoice matrix
- \( U \) defines the SNR subspace
- \( w_k \) is SNR factor with prior distribution \( \mathcal{N}(0, I) \)
- \( \epsilon_{ij}^k \) is a residual term with prior distribution \( \mathcal{N}(0, \Sigma) \); \( \Sigma \) is a full covariance matrix aiming to model the channel variability
SNR-invariant PLDA

- Training utterances are divided into $K$ groups, according to their SNR.
PLDA vs. SNR-invariant PLDA

- Comparing the use of training i-vectors with conventional PLDA

Conventional PLDA

\[ x_{ij} = m + Vh_i + \varepsilon_{ij} \]

SNR-Invariant PLDA

\[ x_{ij}^k = m + Vh_i + Uw_k + \varepsilon_{ij}^k \]
## PLDA vs. SNR-invariant PLDA

Comparing generative models:

<table>
<thead>
<tr>
<th>PLDA</th>
<th>SNR-Invariant PLDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{ij} = m + Vh_i + \epsilon_{ij}$</td>
<td>$x_{ij}^k = m + Vh_i + Uw_k + \epsilon_{ij}^k$</td>
</tr>
<tr>
<td>$x \sim \mathcal{N}(x</td>
<td>m, VV^T + \Sigma)$</td>
</tr>
<tr>
<td>$\theta = {m, V, \Sigma}$</td>
<td>$\theta = {m, V, U, \Sigma}$</td>
</tr>
</tbody>
</table>
The parameters $\theta = \{m, V, U, \Sigma\}$ can be learned from a training set $\mathcal{X}$ using maximum likelihood estimation.

$\mathcal{X} = \{x_{ij}^k; i = 1, \ldots, S; j = 1, \ldots, H_i(k); k = 1, \ldots, K\}$

- $S$: No. of training speakers
- $K$: No. of SNR groups
- $H_i(k)$: No. of utterances from speaker $i$ in the $k$-th SNR group.

Given an initial value $\theta$, we aim to find a new estimate $\hat{\theta}$ that maximizes the auxiliary function:

$$Q(\hat{\theta} | \theta) = \mathbb{E}_{h,w} \left[ \sum_{ikj} \ln \left( p(x_{ij}^k | h_i, w_k, \hat{\theta}) p(h_i, w_k) \right) \bigg| \mathcal{X}, \theta \right]$$

$$= \mathbb{E}_{h,w} \left[ \sum_{ikj} \left( \ln \mathcal{N}(x_{ij}^k | m + Vh_i + Uw_k, \Sigma) + \ln p(h_i, w_k) \right) \bigg| \mathcal{X}, \theta \right]$$
We show 3 ways to compute the posteriors:

1. Computing the posterior of $h_i$ and $w_k$ separately.
2. Computing the posterior $h_i$ while fixing $w_k$ using the Gauss-Seidel method.
3. Computing the joint posterior of $h_i$ and $w_k$ using variational Bayes.
Method 1: Computing posteriors separately

Given i-vectors $x_{ij}^k$, the posterior density of $h_i$ has the form:

$$p(h_i|x_{ij}^k, \theta) \propto p(x_{ij}^k|h_i, \theta)p(h_i)$$

$$= \int p(x_{ij}^k, w_k|h_i, \theta)p(h_i)dw_k$$

$$= \int p(x_{ij}^k|h_i, w_k, \theta)p(w_k)p(h_i)dw_k$$

$$= \int \mathcal{N}(x_{ij}^k|m + Vh_i + Uw_k, \Sigma)\mathcal{N}(w_k|0, I)\mathcal{N}(h_i|0, I)dw_k$$

$$= \mathcal{N}(x_{ij}^k|m + Vh_i, \Phi)\mathcal{N}(h_i|0, I)$$

$$\propto \exp\left\{h_i^T V^T \Phi^{-1}(x_{ij}^k - m) - \frac{1}{2}h_i^T(I + V^T \Phi^{-1}V)h_i\right\}$$

where $\Phi = UU^T + \Sigma$. 
Method 1: Computing posteriors separately

- If all of the i-vectors of speaker \(i\), say \(X_i\), are given,

\[
p(h_i|x_{ij}^k \forall j \text{ and } k, \theta) \propto \prod_{k=1}^K \prod_{j=1}^{H_i(k)} p(x_{ij}^k|h_i, \theta)p(h_i)
\]

\[
\propto \exp\left\{ h_i^T V^T \Phi^{-1} \sum_{k=1}^K \sum_{j=1}^{H_i(k)} (x_{ij}^k - m) - \frac{1}{2} h_i^T \left( I + \sum_{k=1}^K H_i(k)V^T \Phi^{-1} V \right) h_i \right\}
\]

- This is a Gaussian with mean and 2nd-order (uncentralized) moment

\[
\langle h_i | x_i \rangle = \left( I + \sum_{k=1}^K H_i(k)V^T \Phi^{-1} V \right)^{-1} V^T \Phi^{-1} \sum_{k=1}^K \sum_{j=1}^{H_i(k)} (x_{ij}^k - m)
\]

\[
\langle h_i h_i^T | x_i \rangle = \left( I + \sum_{k=1}^K H_i(k)V^T \Phi^{-1} V \right)^{-1} + \langle h_i | x_i \rangle \langle h_i | x_i \rangle^T
\]

\[
\mathcal{N}(h | \mu_h, C_h) \propto \exp \left\{ -\frac{1}{2} (h - \mu_h)^T C_h^{-1} (h - \mu_h) \right\} \propto \exp \left\{ h^T C_h^{-1} \mu_h - \frac{1}{2} h^T C_h^{-1} h \right\}
\]
Method 1: Computing posteriors separately

Similarly, to compute the posterior expectations of $w_k$, we marginalize over $h_i$’s. Thus, the posterior density of $w_k$ is

$$p(w_k|x_{ij}^k, \theta) \propto \int p(x_{ij}^k|h_i, w_k, \theta)p(h_i)p(w_k)dh_i$$

$$= \int \mathcal{N}(x_{ij}^k|m + Vh_i + Up_k, \Sigma)\mathcal{N}(h_i|0, I)\mathcal{N}(w_k|0, I)dh_i$$

$$= \mathcal{N}(x_{ij}^k|m + Uw_k, \Psi)\mathcal{N}(w_k|0, I)$$

$$\propto \exp \left\{ w_k^TU^T\Psi^{-1}(x_{ij}^k - m) - \frac{1}{2}w_k(I + U^T\Psi^{-1}U)w_k \right\}$$
Method 1: Computing posteriors separately

Given all of the i-vectors ($\mathcal{X}^k$) from the $k$-th SNR group, we can compute the posterior expectations as follows:

$$\langle w_k | \mathcal{X}^k \rangle = \left( I + \sum_{i=1}^{S} H_i(k) U^T \Psi^{-1} U \right)^{-1} U^T \Psi^{-1} \sum_{i=1}^{S} \sum_{j=1}^{S} (x_{ij}^k - m)$$

$$\langle w_k w_k^T | \mathcal{X}^k \rangle = \left( I + \sum_{i=1}^{S} H_i(k) U^T \Psi^{-1} U \right)^{-1} + \langle w_k | \mathcal{X}^k \rangle \langle w_k | \mathcal{X}^k \rangle^T$$

(2)

where $\Psi = VV^T + \Sigma$
Method 2: Computing posteriors by Gauss-Seidel method

- Another approach to computing $p(h_i|\mathcal{X}_i)$ is to assume that $w_k$’s are fixed for all $k = 1, \ldots, K$.
- This is called the Gauss-Seidel method [Barrett et al., 1994]
- We fix $w_k$ to its posterior mean: $w_k^* \equiv \langle w_k | \mathcal{X}^k \rangle$
- The posterior density of $h_i$ becomes:

$$p(h_i|\mathcal{X}_i, w_k^*, \theta) \propto \prod_{k=1}^{K} \prod_{j=1}^{H_i(k)} p(x_{ij}^k|h_i, w_k^*, \theta) p(h_i)$$

$$= \prod_{k=1}^{K} \prod_{j=1}^{H_i(k)} \mathcal{N}(x_{ij}^k|m + V h_i + U w_k^*, \Sigma) \mathcal{N}(h_i|0, I)$$

$$\propto \exp \left\{ h_i^T V^T \Sigma^{-1} \sum_{k=1}^{K} \sum_{j=1}^{H_i(k)} (x_{ij}^k - m - U w_k^*) - \frac{1}{2} h_i^T \left( I + \sum_{k=1}^{K} H_i(k) V^T \Sigma^{-1} V \right) h_i \right\}$$
Comparing this posterior density with a standard Gaussian, we have

\[
\langle h_i | \mathcal{X}_i \rangle = \left( L_i^{(1)} \right)^{-1} V^T \Sigma^{-1} \sum_{k=1}^{K} \sum_{j=1}^{H_i(k)} (x_{ij}^k - m - Uw_k^*)
\]

where \( L_i^{(1)} \equiv I + \sum_{k=1}^{K} H_i(k) V^T \Sigma^{-1} V \)

Note that these formulations is similar to the JFA model estimation in [Vogt and Sridharan, 2008].
Apply the same approach to computing the posterior density of $\mathbf{w}_k$, we have

$$
\langle \mathbf{w}_k | \mathbf{x}^k \rangle = \left( \mathbf{L}_k^{(2)} \right)^{-1} \mathbf{U}^T \mathbf{\Sigma}^{-1} \sum_{i=1}^{S} \sum_{j=1}^{S} (\mathbf{x}_{ij}^k - \mathbf{m} - \mathbf{V}_i \mathbf{h}_i^*)
$$

where $\mathbf{L}_k^{(2)} = \mathbf{I} + \sum_{i=1}^{S} \mathbf{H}_i(\mathbf{k}) \mathbf{U}^T \mathbf{\Sigma}^{-1} \mathbf{U}$ and $\mathbf{h}_i^* \equiv \langle \mathbf{h}_i | \mathbf{x}_i \rangle$

$$
\langle \mathbf{w}_k \mathbf{w}_k^T | \mathbf{x}^k \rangle = \left( \mathbf{L}_k^{(2)} \right)^{-1} + \langle \mathbf{w}_k | \mathbf{x}^k \rangle \langle \mathbf{w}_k | \mathbf{x}^k \rangle^T
$$
Method 3: Computing posteriors by variational Bayes

- Denote $\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_K]$ and $\mathbf{h} = [\mathbf{h}_1, \ldots, \mathbf{h}_S]$

- In variational Bayes [Bishop, 2006, Kenny, 2010], we factorize the joint posterior as follows:

$$
\ln p(\mathbf{h}, \mathbf{w} | \mathcal{X}) \approx \ln q(\mathbf{h}) + \ln q(\mathbf{w}) = \sum_{i=1}^{S} \ln q(\mathbf{h}_i) + \sum_{k=1}^{K} \ln q(\mathbf{w}_k)
$$

where

$$
\ln q(\mathbf{h}) = \mathbb{E}_{\mathbf{w}} \{ \ln p(\mathbf{h}, \mathbf{w}, \mathcal{X}) \} + \text{const}
$$

$$
\ln q(\mathbf{w}) = \mathbb{E}_{\mathbf{h}} \{ \ln p(\mathbf{h}, \mathbf{w}, \mathcal{X}) \} + \text{const}
$$

where $\mathbb{E}_{\mathbf{w}}$ means taking expectation with respect to $\mathbf{w}$. 
Method 3: Computing posteriors by variational Bayes

Consider \( \ln q(h) \):

\[
\ln q(h) = \mathbb{E}_w \{ \ln p(h, w, x') \} + \text{const}
\]

\[
= \langle \ln p(x' | h, w) \rangle_w + \langle \ln p(h, w) \rangle_w + \text{const}
\]

\[
= \sum_{ijr} \langle \ln \mathcal{N}(x'_{ij} | m + Vh_i + Uw_r, \Sigma) \rangle_{w_r}
\]

\[
+ \sum_i \langle \ln \mathcal{N}(h_i | 0, I) \rangle_w + \sum_r \langle \ln \mathcal{N}(w_r | 0, I) \rangle_w + \text{const}
\]

\[
= -\frac{1}{2} \sum_{ijr} (x'_{ij} - m - Vh_i - Uw_r^*)^T \Sigma^{-1} (x'_{ij} - m - Vh_i - Uw_r^*)
\]

\[
- \frac{1}{2} \sum_i h_i^T h_i + \text{const}
\]

\[
= \sum_i \left[ h_i^T V^T \Sigma^{-1} \sum_{jr} (x'_{ij} - m - Uw_r^*) - \frac{1}{2} h_i^T \left( I + \sum_{jr} V^T \Sigma^{-1} V \right) h_i^T \right] + \text{const}
\]

\[
q(h_i) \text{ a Gaussian with mean and precision identical to Eq. 3:}
\]

\[
\langle h_i | x'_{i} \rangle = \left( L_i^{(1)} \right)^{-1} V^T \Sigma^{-1} \sum_{jr} (x'_{ij} - m - Uw_r^*)
\]

\[
L_i^{(1)} = I + \sum_{jr} V^T \Sigma^{-1} V
\]
Method 3: Computing posteriors by variational Bayes

\[ \ln q(w) = \langle \ln p(X|h, w) \rangle_h + \langle \ln p(h, w) \rangle_h + \text{const} \]

\[ = \sum_{ijk} \langle \ln \mathcal{N}(x_{ij}^k|m + Vh_i + Uw_k, \Sigma) \rangle_{h_i} \]

\[ + \sum_i \langle \ln \mathcal{N}(h_i|0, I) \rangle_{h_i} + \sum_k \langle \ln \mathcal{N}(w_k|0, I) \rangle_h + \text{const} \]

\[ = -\frac{1}{2} \sum_{ijk} (x_{ij}^k - m - Vh_i^* - Uw_k)^T \Sigma^{-1} (x_{ij}^k - m - Vh_i^* - Uw_k) \]

\[ - \frac{1}{2} \sum_k w_k^T w_k + \text{const} \]

\[ = \sum_k \left[ w_k^T U^T \Sigma^{-1} \sum_{ij} (x_{ij}^k - m - Vh_i^*) - \frac{1}{2} w_k^T \left( I + \sum_{ij} U^T \Sigma^{-1} U \right) w_k \right] + \text{const} \]

\[ q(w_k) \text{ is a Gaussian with mean and precision identical to Eq. 4:} \]

\[ \langle w_k|X^k \rangle = \left( L_k^{(2)} \right)^{-1} U^T \Sigma^{-1} \sum_{ij} (x_{ij}^k - m - Vh_i^*) \]

\[ L_k^{(2)} = I + \sum_{ij} U^T \Sigma^{-1} U \] (7)

Note: \[ \langle \ln \mathcal{N}(h_i|0, I) \rangle_h \] is the differential entropy of normal distribution and is independent of \( w_k \), see [Norwich, 1993](Ch 8).
Computing posterior moment

- The exact posterior moment \( \langle w_k h_i^T | \mathcal{X} \rangle \) will be complicated because \( h_i \) and \( w_k \) are correlated in the posterior.
- If Gauss-Seidel’s method is used, we may approximate the posterior moments by (Kenny 2010, p.6)

\[
\langle w_k h_i^T | \mathcal{X} \rangle \approx \langle w_k | \mathcal{X}^k \rangle (h_i^*)^T
\]
\[
\langle h_i w_k^T | \mathcal{X} \rangle \approx \langle h_i | \mathcal{X}_i \rangle (w_k^*)^T
\]

where \( h_i^* \) and \( w_k^* \) are the most up-to-date posterior means in the EM iterations.
- Alternatively, we may compute the exact joint posterior.³ But it will be computationally intensive.

A better approach is to use variational Bayes:

\[ p(h_i, w_k | \mathcal{X}) \approx q(h_i)q(w_k) \quad (8) \]

Note that as both \( q(h_i) \) and \( q(w_k) \) are Gaussians. Based on the law of total expectation,\(^4\) the factorization in Eq. 8 gives

\[
\langle w_k h_i^T | \mathcal{X} \rangle \approx \langle w_k | \mathcal{X}^k \rangle \langle h_i | \mathcal{X}_i \rangle^T
\]
\[
\langle h_i w_k^T | \mathcal{X} \rangle \approx \langle h_i | \mathcal{X}_i \rangle \langle w_k | \mathcal{X}^k \rangle^T
\]

\(^4\)https://en.wikipedia.org/wiki/Product_distribution
In the M-step, we maximize the auxiliary function:

\[
Q(\theta) = \mathbb{E}_{h,w} \left\{ \sum_{ijk} \ln \mathcal{N} \left( x_{ij}^k | m + Vh_i + Uw_k, \Sigma \right) p(h_i, w_k) \bigg| \mathcal{X}, \theta \right\}
\]

\[
= \sum_{ijk} \mathbb{E}_{h,w} \left\{ -\frac{1}{2} \log |\Sigma| - \frac{1}{2} \left( x_{ij}^k - m - Vh_i - Uw_k \right)^T \Sigma^{-1} \right. \\
\left. \times \left( x_{ij}^k - m - Vh_i - Uw_k \right) + \ln p(h_i, w_k) \bigg| \mathcal{X}, \theta \right\}
\]

As \( p(h_i, w_k) \) is independent of the model parameters \( V, U, \) and \( \Sigma \), they could be taken out of \( Q(\theta) \) in the M-step [Prince and Elder, 2007].
Maximization Step

- Differentiating $Q(\theta)$ with respect to $V$, $U$, and $\Sigma$ and set the results to 0, we obtain

$$V = \left\{ \sum_{ijk} \left[ (x_{ij}^k - m) \langle h_i | x_i \rangle - U \langle w_k h_i^T | x_i \rangle \right] \right\} \left[ \sum_{ijk} \langle h_i h_i^T | x_i \rangle \right]^{-1}$$

$$U = \left\{ \sum_{ijk} \left[ (x_{ij}^k - m) \langle w_k | x_i^k \rangle - V \langle h_i w_k^T | x_i \rangle \right] \right\} \left[ \sum_{ijk} \langle w_k w_k^T | x_i \rangle \right]^{-1}$$

$$\Sigma = \frac{1}{N} \sum_{ijk} \left[ (x_{ij}^k - m)(x_{ij}^k - m)^T \right.$$  

$$- V \langle h_i | x_i \rangle (x_{ij}^k - m)^T - U \langle w_k | x_i^k \rangle (x_{ij}^k - m)^T \left] \right.$$
Likelihood Ratio Scores

- Given target-speaker's i-vector $x_s$ and test-speaker's i-vector $x_t$
- If $x_s$ and $x_t$ are from the same speaker, they should share the same speaker factor $h$:

$$
\begin{bmatrix}
  x_s \\
  x_t
\end{bmatrix} =
\begin{bmatrix}
  m \\
  m
\end{bmatrix} +
\begin{bmatrix}
  V & U & 0 \\
  V & 0 & U
\end{bmatrix}
\begin{bmatrix}
  h \\
  w_s \\
  w_t
\end{bmatrix} +
\begin{bmatrix}
  \epsilon_s \\
  \epsilon_t
\end{bmatrix}
$$

$$
\implies \hat{x}_{st} = \hat{m} + \hat{A}\hat{z}_{st} + \hat{\epsilon}_{st}.
$$

- Same-speaker likelihood:

$$
p(\hat{x}_{st}|\text{same-speaker}) = \int p(\hat{x}_{st}|\hat{z}_{st})p(\hat{z}_{st})d\hat{z}_{st}
$$

$$
= \int \mathcal{N}(\hat{x}_{st}|\hat{m} + \hat{A}\hat{z}_{st}, \hat{\Sigma})\mathcal{N}(\hat{z}_{st}|0, I)d\hat{z}_{st}
$$

$$
= \mathcal{N}(\hat{x}_{st}|\hat{m}, \hat{A}\hat{A}^T + \hat{\Sigma})
$$

$$
= \mathcal{N}
\left(
\begin{bmatrix}
  x_s \\
  x_t
\end{bmatrix},
\begin{bmatrix}
  m \\
  m
\end{bmatrix},
\begin{bmatrix}
  \Sigma_{tot} & \Sigma_{ac} \\
  \Sigma_{ac} & \Sigma_{tot}
\end{bmatrix}
\right)
$$

where $\hat{\Sigma} = \text{diag}\{\Sigma, \Sigma\}$, $\Sigma_{tot} = \mathbf{V}\mathbf{V}^T + \mathbf{U}\mathbf{U}^T + \Sigma$ and $\Sigma_{ac} = \mathbf{V}\mathbf{V}^T$. 
Likelihood Ratio Scores

- If \( \mathbf{x}_s \) and \( \mathbf{x}_t \) are from different speakers, they should have their own speaker factor \((\mathbf{h}_s, \mathbf{h}_t)\):

\[
\begin{bmatrix}
\mathbf{x}_s \\
\mathbf{x}_t
\end{bmatrix} =
\begin{bmatrix}
\mathbf{m} \\
\mathbf{m}
\end{bmatrix} +
\begin{bmatrix}
\mathbf{V} & 0 & \mathbf{U} & 0 \\
0 & \mathbf{V} & 0 & \mathbf{U}
\end{bmatrix}
\begin{bmatrix}
\mathbf{h}_s \\
\mathbf{h}_t \\
\mathbf{w}_s \\
\mathbf{w}_t
\end{bmatrix} +
\begin{bmatrix}
\epsilon_s \\
\epsilon_t
\end{bmatrix}
\]

\[\implies \hat{\mathbf{x}}_{st} = \hat{\mathbf{m}} + \tilde{\mathbf{A}}\tilde{\mathbf{z}}_{st} + \hat{\epsilon}_{st}\]

- Different-speaker likelihood:

\[
p(\hat{\mathbf{x}}_{st} | \text{diff-speaker}) = \int p(\hat{\mathbf{x}}_{st} | \tilde{\mathbf{z}}_{st})p(\tilde{\mathbf{z}}_{st})d\tilde{\mathbf{z}}_{st}
\]

\[= \int \mathcal{N}(\hat{\mathbf{x}}_{st} | \hat{\mathbf{m}} + \tilde{\mathbf{A}}\tilde{\mathbf{z}}_{st}, \hat{\Sigma})\mathcal{N}(\tilde{\mathbf{z}}_{st} | \mathbf{0}, \mathbf{I})d\tilde{\mathbf{z}}_{st}
\]

\[= \mathcal{N}(\hat{\mathbf{x}}_{st} | \hat{\mathbf{m}}, \tilde{\mathbf{A}}\tilde{\mathbf{A}}^\top + \hat{\Sigma})
\]

\[= \mathcal{N}\left(\begin{bmatrix}
\mathbf{x}_s \\
\mathbf{x}_t
\end{bmatrix} | \begin{bmatrix}
\mathbf{m} \\
\mathbf{m}
\end{bmatrix}, \begin{bmatrix}
\Sigma_{tot} & \mathbf{0} \\
\mathbf{0} & \Sigma_{tot}
\end{bmatrix}\right)\]
Likelihood Ratio Scores

- Log-likelihood ratio score (assuming i-vectors have been mean subtracted, $x \leftarrow x - m$)

$$S_{LR}(x_s, x_t) = \log \frac{p(x_s, x_t | \text{Same-speaker})}{p(x_s, x_t | \text{Diff-speaker})}$$

$$= \log \frac{\mathcal{N} \left( \begin{bmatrix} x_s \\ x_t \end{bmatrix} | \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma_{tot} & \Sigma_{ac} \\ \Sigma_{ac} & \Sigma_{tot} \end{bmatrix} \right)}{\mathcal{N} \left( \begin{bmatrix} x_s \\ x_t \end{bmatrix} | \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma_{tot} & 0 \\ 0 & \Sigma_{tot} \end{bmatrix} \right)}$$

$$= \frac{1}{2} \left[ x_s^T Q x_s + 2 x_s^T P x_t + x_t^T Q x_t \right] + \text{const}$$

where

$$Q = \Sigma_{tot}^{-1} - \left( \Sigma_{tot} - \Sigma_{ac} \Sigma_{tot}^{-1} \Sigma_{ac} \right)^{-1}$$

$$P = \Sigma_{tot}^{-1} \Sigma_{ac} \left( \Sigma_{tot} - \Sigma_{ac} \Sigma_{tot}^{-1} \Sigma_{ac} \right)^{-1}$$
Likelihood Ratio Scores

- The LLR in Eq. 9 assumes that the SNR of both target-speaker’s utterance and test utterance are unknown.
- If both SNRs ($\ell_s, \ell_t$) are known, we may compute the score as follows:

$$S_{LR}(x_s, x_t | \ell_s, \ell_t) = \log \frac{p(x_s, x_t | \text{Same-speaker}, \ell_s, \ell_t)}{p(x_s, x_t | \text{Diff-speaker}, \ell_s, \ell_t)}$$
Likelihood Ratio Scores

- Given i-vector $x$ and utterance SNR $\ell$, the likelihood of $x$ is

$$p(x|\ell) = \int_h \int_w p(x|h, w, \ell) p(h, w|\ell) dh dw$$

$$= \int_h \int_w p(x|h, w, \ell) p(h|w, \ell) p(w|\ell) dh dw$$

$$= \int_h \int_w p(x|h, w, \ell) p(h) dh p(w|\ell) dw$$

where we have assumed that $h$ is a priori independent of $w$ and $\ell$.

- Note that if $\ell \in k$-th SNR group, we have $w = w_k^* \equiv \langle w_k | X^k \rangle$

$$p(x|\ell \in k\text{-th SNR group}) = \int_h p(x|h, w_k^*) p(h) dh$$

$$= \int_h \mathcal{N}(x|m + Vh + Uw_k^*, \Sigma) \mathcal{N}(h|0, I) dh$$

$$= \mathcal{N}(x|m + Uw_k^*, VV^T + \Sigma)$$
Likelihood Ratio Scores

\[ S_{LR}(x_s, x_t | l_s, l_t) = \log \frac{p(x_s, x_t | \text{Same-speaker}, l_s, l_t)}{p(x_s, x_t | \text{Diff-speaker}, l_s, l_t)} \]

\[ = \log \frac{\mathcal{N} \left( \begin{bmatrix} x_s \\ x_t \end{bmatrix} \bigg| \begin{bmatrix} m + Uw_{ks}^* \\ m + Uw_{kt}^* \end{bmatrix}, \begin{bmatrix} \psi & \Sigma_{ac} \\ \Sigma_{ac} & \psi \end{bmatrix} \right)}{\mathcal{N} \left( \begin{bmatrix} x_s \\ x_t \end{bmatrix} \bigg| \begin{bmatrix} m + Uw_{ks}^* \\ m + Uw_{kt}^* \end{bmatrix}, \begin{bmatrix} \psi & 0 \\ 0 & \psi \end{bmatrix} \right)} \]

\[ = \frac{1}{2} [\bar{x}_s^T Q \bar{x}_s + 2\bar{x}_s^T P \bar{x}_t + \bar{x}_t^T Q \bar{x}_t] + \text{const} \]

where

\[ \bar{x}_s = x_s - m - Uw_{ks}^* \]
\[ \bar{x}_t = x_t - m - Uw_{kt}^* \]
\[ Q = \psi^{-1} - (\psi - \Sigma_{ac} \psi^{-1} \Sigma_{ac})^{-1} \]
\[ P = \psi^{-1} \Sigma_{ac} (\psi - \Sigma_{ac} \psi^{-1} \Sigma_{ac})^{-1} \]
\[ \psi = VV^T + \Sigma; \quad \Sigma_{ac} = VV^T \]
Scoring in JFA is based on the **sequential mode** [Kenny et al., 2007b]:

\[
S_{\text{JFA-LR}}(O_s, O_t) = \frac{P_{\Lambda(s)}(O_t)}{P_{\Lambda}(O_t)}
\]

where \(\Lambda(s)\) denotes the adapted speaker model based on enrollment speech \(O_s\) from speaker \(s\).

- Computing \(P_{\Lambda(s)}(O_t)\) requires the posterior density of speaker factors \(y(s)\) and \(z(s)\) in Kenny 2007], which are posteriorly correlated.
- The scoring function in Eq. 9 is based on the **batch mode**.
- Batch mode is similar to speaker comparison in which no model adaptation is performed. So, the posterior correlation between speaker factors and SNR factors does not occur in Eq. 9.
Scoring based on sequential mode

- The batch-mode scoring (Eq. 10) requires inverting a big matrix if the target speaker has a large number of enrollment utterances.
- The sequential-mode scoring can mitigate this problem.
- For notational simplicity, we assume that the target speaker only have one enrollment utterance with i-vector $x_s$:

$$S_{LR}(x_s, x_t|\ell_s, \ell_t) = \frac{p(x_s, x_t|\ell_s, \ell_t)}{p(x_s|\ell_s)p(x_t|\ell_t)} = \frac{p(x_t|x_s, \ell_s, \ell_t)p(x_s|\ell_s)}{p(x_s|\ell_s)p(x_t|\ell_t)} = \frac{p(x_t|x_s, \ell_s, \ell_t)}{p(x_t|\ell_t)}$$
Scoring based on sequential mode

- For simplicity, we omit $\ell_s$ and $\ell_t$ from now on.

$$p(x_t|x_s) = \int \int p(x_t|h, w)p(h, w|x_s) dh dw$$

- As $h$ and $w$ are posteriorly dependent, we use variational Bayes to approximate the joint posterior:

$$p(x_t|x_s) \approx \int \int p(x_t|h, w)q(h)q(w) dh dw$$

$$= \int_h \int_w \mathcal{N}(x_t|m + Vh + Uw, \Sigma) \mathcal{N}(h|\mu_{hs}, \Sigma_{hs}) \mathcal{N}(w|\mu_{ws}, \Sigma_{ws}) dh dw$$

(11)

where $\mu_{hs}$, $\Sigma_{hs}$, $\mu_{ws}$, and $\Sigma_{ws}$ are posterior means and posterior covariances.
Scoring based on sequential mode

- Eq. 11 is a convolution of Gaussians

\[ p(x_t|x_s) \approx \int_h \int_w N(x_t|m + Vh + Uw, \Sigma)N(h|\mu_{h_s}, \Sigma_{h_s})dhN(w|\mu_{w_s}, \Sigma_{w_s})dw \]

\[ = \int_w N(x_t|m + V\mu_{h_s} + Uw, V\Sigma_{h_s}V^T + \Sigma)N(w|\mu_{w_s}, \Sigma_{w_s})dw \]

\[ = N(x_t|m + V\mu_{h_s} + U\mu_{w_s}, V\Sigma_{h_s}V^T + U\Sigma_{w_s}U^T + \Sigma) \]

- If \( \ell_s \) falls on the \( k \)-th SNR group, we may replace \( \mu_{w_s} \) by \( w^*_k \equiv \langle w_k|x^{'k} \rangle \) and assume that \( \Sigma_{w^*_k} \to 0 \):

\[ p(x_t|x_s) = N(x_t|m + V\mu_{h_s} + Uw^*_k, V\Sigma_{h_s}V^T + \Sigma) \]

- \( p(x_t) \) is a marginal density

\[ p(x_t) = \int p(x_t|h, w)p(h, w)dhdw \]

\[ = \int N(x_t|m + Vh + Uw, \Sigma)N(h|0, I)N(w|0, I)dhdw \]

\[ = N(x_t|m, VV^T + UU^T + \Sigma) \]
Scoring based on sequential mode

The posteriors means and covariances can be obtained from Eq. 6 and Eq. 7 by considering a single utterance from target-speaker $s$:

$$\mu_{h_s} = \langle h_s | x_s \rangle = \Sigma_{h_s} V^T \Sigma^{-1} (x_s - m - U \mu_{w_s})$$

$$\mu_{w_s} = \langle w_s | x_s \rangle = \Sigma_{w_s} U^T \Sigma^{-1} (x_s - m - V \mu_{h_s})$$

$$\Sigma_{h_s} = (I + V^T \Sigma^{-1} V)^{-1}$$

$$\Sigma_{w_s} = (I + U^T \Sigma^{-1} U)^{-1}$$

Note that $\mu_{h_s}$ and $\mu_{w_s}$ depend on each other, meaning that they should be found iteratively.
Experiments on SRE12

- **Evaluation dataset**: Common evaluation condition 1 and 4 of NIST SRE 2012 core set.
- **Parameterization**: 19 MFCCs together with energy plus their 1st and 2nd derivatives $\Rightarrow$ 60-Dim acoustic vectors
- **UBM**: Gender-dependent, mic+tel, 1024 mixtures
- **Total Variability Matrix**: Gender-dependent, mic+tel, 500 total factors
- **I-Vector Preprocessing**: Whitening by WCCN then length normalization followed by non-parametric feature analysis (NFA)\(^5\) and WCCN (500-dim $\rightarrow$ 200-dim)

---

Prepare training speech files

- Original tel speech files
  - 15dB FaNT
  - 14dB FaNT
  - 6dB FaNT

Noisy speech files

Selection

- SNR ≤ 4dB
- 4dB < SNR ≤ 5dB
- 13dB < SNR ≤ 14dB
- 14dB < SNR

Table:

<table>
<thead>
<tr>
<th>SNR Range</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNR ≤ 4dB</td>
<td>Interval 1</td>
</tr>
<tr>
<td>4dB &lt; SNR ≤ 5dB</td>
<td>Interval 2</td>
</tr>
<tr>
<td>13dB &lt; SNR ≤ 14dB</td>
<td>Interval 11</td>
</tr>
<tr>
<td>14dB &lt; SNR</td>
<td>Interval 12</td>
</tr>
</tbody>
</table>

Training Utterances

Union

Original utts (mic+tel)
SNR distributions

- SNR Distribution of training and test utterances in CC4
Performance on SRE12

### Mixture of PLDA (Mak, Interspeech14)

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>EER(%)</td>
<td>minDCF</td>
</tr>
<tr>
<td>PLDA</td>
<td></td>
<td>5.42</td>
<td>0.371</td>
</tr>
<tr>
<td>mPLDA</td>
<td></td>
<td>5.28</td>
<td>0.415</td>
</tr>
<tr>
<td>SNR-Invariant PLDA</td>
<td>3</td>
<td>5.42</td>
<td>0.382</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5.28</td>
<td>0.381</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5.29</td>
<td>0.388</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>5.56</td>
<td>0.384</td>
</tr>
</tbody>
</table>

**No. of SNR Groups**

**No. of SNR factors (dim of $\mathbf{W}_k$)**
## Performance on SRE12

### Mixture of PLDA (Mak, Interspeech14)

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K</td>
<td>Q</td>
<td>EER(%)</td>
</tr>
<tr>
<td>PLDA</td>
<td>-</td>
<td>-</td>
<td>3.13</td>
</tr>
<tr>
<td>mPLDA</td>
<td>-</td>
<td>-</td>
<td>2.88</td>
</tr>
<tr>
<td>SNR-Invariant PLDA</td>
<td>3</td>
<td>40</td>
<td>2.72</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>40</td>
<td>2.67</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>40</td>
<td><strong>2.63</strong></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>30</td>
<td>2.70</td>
</tr>
</tbody>
</table>

- **No. of SNR Groups**
- **No. of SNR factors (dim of $w_k$)**
Performance on SRE12

![Graph showing performance comparison between Conventional PLDA and SNR-Invariant PLDA](image)

- **Conventional PLDA**
- **SNR-Invariant PLDA**

**CC4, Female**
Mixture of PLDA: Motivation

- Conventional i-vector/PLDA systems use a single PLDA model to handle all SNR conditions
Mixture of PLDA: Motivation

- We argue that a PLDA model should focus on a small range of SNR.
Distribution of SNR

Each SNR region is handled by a PLDA Model
Proposed solution

- The full spectrum of SNRs is handled by a mixture of PLDA in which the posteriors of the indicator variables depend on the utterance’s SNR.
- Verification scores depend not only on the same-speaker and different-speaker likelihoods but also on the posterior probabilities of SNR.
Mixture of PLDA [Mak et al., 2016]

- Model parameters:

\[ \theta = \{ \pi, \mu, \sigma, m, V, \Sigma \} \]
\[ = \{ \pi_k, \mu_k, \sigma_k, m_k, V_k, \Sigma_k \}_{k=1}^K \]

Modeling SNR Speaker subspaces

- Generative model:

\[ x_{ij} \sim \sum_{k=1}^K P(y_{ijk} = 1|\ell_{ij}) \mathcal{N}(x_{ij}|m_k, V_k V_k^T + \Sigma_k) \]

where

\[ P(y_{ijk} = 1|\ell_{ij}) = \frac{\pi_k \mathcal{N}(\ell_{ij}|\mu_k, \sigma^2_k)}{\sum_{k'=1}^K \pi_{k'} \mathcal{N}(\ell_{ij}|\mu_{k'}, \sigma^2_{k'})} \]

and \( \ell_{ij} \) is the SNR of the utterance \( j \) from speaker \( i \).
Mixture of PLDA

Graphical model:

\[ \theta = \{ \pi_k, \mu_k, \sigma_k, \mathbf{m}_k, \mathbf{V}_k, \Sigma_k \}_{k=1}^K \]

- \( \mathbf{x}_{ij} \): i-vector of the j-th utterance from the i-th speaker
- \( \ell_{ij} \): SNR of the j-th utterance from the i-th speaker

\[ \boldsymbol{\pi} = \{ \pi_k \}_{k=1}^K \]

\[ \mathbf{V} = \{ \mathbf{V}_k \}_{k=1}^K \]

For modeling SNR of utts.
For modeling SNR-dependent i-vectors
Graphical models:

**PLDA**

- \( z_i \)
- \( x_{ij} \)
- \( H_i \)
- \( V \)
- \( N \)

**Mixture of PLDA**

- \( m \)
- \( \Sigma \)
- \( x_{ij} \)
- \( z_i \)
- \( y_{ijk} \)
- \( K \)
- \( \pi \)
- \( V \)
- \( \mu, \sigma \)
- \( \ell_{ij} \)
- \( H_i \)
- \( N \)
Experiments on SRE12

- **Evaluation dataset**: Common evaluation condition 1 and 4 of NIST SRE 2012 core set.
- **Parameterization**: 19 MFCCs together with energy plus their 1st and 2nd derivatives $\rightarrow$ 60-Dim acoustic vectors
- **UBM**: Gender-dependent, mic+tel, 1024 mixtures
- **Total Variability Matrix**: Gender-dependent, mic+tel, 500 total factors
- **I-Vector Preprocessing**: Whitening by WCCN then length normalization followed by LDA and WCCN (500-dim $\rightarrow$ 200-dim)
Experiments on SRE12

- In NIST 2012 SRE, training utterances from telephone channels are clean, but some of the test utterances are noisy.
- We used the FaNT tool to add babble noise to the clean training utterances.
DET Performance

- Train on tel+mic speech and test on noisy tel speech (CC4).
I-vector Cluster Alignment

Without using SNR

With SNR as guidance
Outline

1 Introduction
2 Learning Algorithms
3 Learning Models
4 Deep Learning
5 Case Studies
   5.1 Heavy-Tailed PLDA
   5.2 SNR-Invariant PLDA
   5.3 Mixture of PLDA
   5.4 DNN I-vectors
   5.5 PLDA with RBM
6 Future Direction
DNN for speaker recognition

- **Main idea**: replacing the universal background model (UBM) with a phonetically-aware DNN for computing the frame posterior probabilities.
- The most successful application of DNN to speaker recognition [Lei et al., 2014, Ferrer et al., 2016, Richardson et al., 2015]

DNN I-vector extraction

UBM I-vectors

- Factor analysis model for UBM i-vectors:
  \[
  \mu_c = \mu_c^{(b)} + T_c w \\
  c = 1, \ldots, C
  \]

- Given the MFCC vectors of an utterance \( O = \{o_1, \ldots, o_T\} \), its i-vector is the posterior mean of \( w \)

  \[
  x \equiv \langle w|O \rangle = L^{-1} \sum_{c=1}^{C} T_c^T \left( \Sigma_c^{(b)} \right)^{-1} \sum_{t=1}^{T} \gamma_c(o_t)(o_t - \mu_c^{(b)})
  \]

where

\[
L = I + \sum_{c=1}^{C} \sum_{t=1}^{T} \gamma_c(o_t) T_c^T \left( \Sigma_c^{(b)} \right)^{-1} T_c
\]

\[
\gamma_c(o_t) \equiv \Pr(\text{Mixture} = c|o_t) = \frac{\lambda_c^{(b)} \mathcal{N}(o_t|\mu_c^{(b)}, \Sigma_c^{(b)})}{\sum_{j=1}^{C} \lambda_j^{(b)} \mathcal{N}(o_t|\mu_j^{(b)}, \Sigma_j^{(b)})}
\]
DNN I-vectors

- Replace $\gamma_c(o_t)$ by DNN output, $\gamma_c^{\text{DNN}}(a_t)$
- The DNN is trained to produce posterior probabilities of senones, given multiple frames of acoustic features, $a_t$, as input.

- Acoustic features for speech recognition in the DNN are not necessarily the same as the features for the i-vector extractor.
Given MFCC or bottleneck (BN) feature vectors $\mathcal{O} = \{o_1, \ldots, o_T\}$, the DNN i-vector is:

$$x \equiv \langle w | \mathcal{O} \rangle = L^{-1} \sum_{c=1}^{C} T_c^T \left( \Sigma_c^{DNN} \right)^{-1} \sum_{t=1}^{T} \gamma_c^{DNN} (a_t) (o_t - \mu_c^{DNN})$$

where

$$\mu_c^{DNN} = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T_i} \gamma_c^{DNN} (a_{it}) o_{it}}{\sum_{i=1}^{N} \sum_{t=1}^{T_i} \gamma_c^{DNN} (a_{it})}$$

$$\Sigma_c^{DNN} = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T_i} \gamma_c^{DNN} (a_{it}) o_{it} o_{it}^T}{\sum_{i=1}^{N} \sum_{t=1}^{T_i} \gamma_c^{DNN} (a_{it})} - \mu_c^{DNN} \left( \mu_c^{DNN} \right)^T$$

---

\[\text{For a full set of formulae, see}\]
Denoising deep classifier [Tan et al., 2016]
DNN I-vectors from denoising deep classifier

\[ \text{MFCC} \rightarrow \text{Input Layer} \]
\[ \text{Hidden Layer 1} \rightarrow w_1 + \varepsilon_1' \]
\[ \text{Hidden Layer 2} \rightarrow w_2 + \varepsilon_2' \]
\[ \text{Hidden Layer 3} \rightarrow w_2^T + \varepsilon_3' \]
\[ \text{Hidden Layer 4} \rightarrow w_1^T + \varepsilon_4' \]
\[ \text{Hidden Layer 5} \rightarrow w_3 + \varepsilon_5 \]
\[ \text{Hidden Layer 5} \rightarrow w_5 + \varepsilon_7 \]

\[ \text{BN Layer} \]

\[ \text{Senones} \rightarrow \text{BN Posterior} \]

\[ \text{PCA Whitening} \]

\[ \text{I-Vector Extractor} \rightarrow \text{Senone I-Vector} \]

\[ \text{1st Order Sufficient Statistics} \]
\[ \text{0th Order Sufficient Statistics} \]

\[ \text{Speech} \]

\[ \text{Denoising Deep Classifier} \]
Performance on NIST 2012 SRE

- Performance on CC4 with test utterances contaminated with babble noise.

<table>
<thead>
<tr>
<th>Acoustic Features</th>
<th>Postiors from</th>
<th>15dB</th>
<th></th>
<th>6dB</th>
<th></th>
<th>0dB</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>EER</td>
<td>minDCF</td>
<td>EER</td>
<td>minDCF</td>
<td>EER</td>
<td>minDCF</td>
</tr>
<tr>
<td>MFCC</td>
<td>GMM (1024 mixtures)</td>
<td>3.366</td>
<td>0.322</td>
<td>3.243</td>
<td>0.353</td>
<td>5.353</td>
<td>0.631</td>
</tr>
<tr>
<td>MFCC</td>
<td>GMM (2048 mixtures)</td>
<td>4.215</td>
<td>0.352</td>
<td>3.819</td>
<td>0.379</td>
<td>5.332</td>
<td>0.646</td>
</tr>
<tr>
<td>BN Features</td>
<td>GMM (1024 mixtures)</td>
<td>3.269</td>
<td>0.263</td>
<td>3.493</td>
<td>0.368</td>
<td>4.608</td>
<td>0.551</td>
</tr>
<tr>
<td>BN Features</td>
<td>DNN (2000 senones)</td>
<td><strong>2.448</strong></td>
<td><strong>0.236</strong></td>
<td><strong>2.774</strong></td>
<td><strong>0.311</strong></td>
<td><strong>4.503</strong></td>
<td><strong>0.544</strong></td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. Learning Algorithms
3. Learning Models
4. Deep Learning
5. Case Studies
   - 5.1. Heavy-Tailed PLDA
   - 5.2. SNR-Invariant PLDA
   - 5.3. Mixture of PLDA
   - 5.4. DNN I-vectors
   - 5.5. PLDA with RBM
6. Future Direction
PLDA–RBM [Stafylakis et al., 2012]

**Main idea:**
- Use i-vectors as input to the Gaussian visible layer of an RBM
- Divide RBM weights into two parts: speaker and channel
- Consider RBM weights as analogue to PLDA’s loading matrices
- Divide the Gaussian hidden layer into two parts: speaker and channel
- Hidden nodes are considered as latent factors
PLDA vs. PLDA–RBM

- **PLDA (omitting global mean):**
  \[ \mathbf{v} = \mathbf{Vs} + \mathbf{Uc} + \epsilon \]
  where \( \mathbf{v} \) is an i-vector, \( \mathbf{s} \) and \( \mathbf{c} \) are speaker and channel factors.

- **RBM-PLDA:**
  \[ \mathbf{v}_n = \sigma_v \left[ \mathbf{W}_s \frac{\mathbf{s}_{st}}{\sigma_s} + \mathbf{W}_c \frac{\mathbf{c}_{st}}{\sigma_c} \right] \]
  where \( \mathbf{v}_n \) is the expected value of visible layer in the negative phase of CD-1 sampling, \( \mathbf{s}_{st} \) and \( \mathbf{c}_{st} \) are the states of Gaussian hidden nodes of the RBM.
Given two i-vectors \( \mathbf{v}_i, i = 1, 2 \), compute \( \mathbf{s}_i = \mathbf{W}_s^T \mathbf{v}_i \).

The log-likelihood ratio is

\[
LLR = -\frac{1}{2}(\mathbf{s}_1 - \mathbf{s}_2)^T(\mathbf{s}_1 - \mathbf{s}_2) + \text{const}
\]

If \( \|\mathbf{s}_i\| = 1 \), the model is similar to cosine distance scoring.
Results on NIST 2010 SRE

- NIST’10, female, core-extended:
Outline

1 Introduction

2 Learning Algorithms

3 Learning Models

4 Deep Learning

5 Case Studies

6 Future Direction
   6.1. Domain adaptation
   6.2. Short-utterance speaker verification
   6.3. Text-dependent speaker recognition
Outline

1. Introduction
2. Learning Algorithms
3. Learning Models
4. Deep Learning
5. Case Studies
6. Future Direction
   6.1. Domain adaptation
   6.2. Short-utterance speaker verification
   6.3. Text-dependent speaker recognition
Domain adaptation aims to adapt a system from a resource-rich domain to a resource-limited domain.

Domain mismatch: systems perform very well in the domain (environment) for which they are trained; however, their performance suffers when the users use the systems in other domain.

Domain Adaptation Challenge (DAC13) was introduced in 2013.

The effect of domain mismatch on PLDA parameters is more pronounced.

**Popular approaches:**
- Bayesian adaptation of PLDA models [Villalba and Lleida, 2014]
- Unsupervised clustering of i-vectors for adapting covariance matrices of PLDA models [Shum et al., 2014, Garcia-Romero et al., 2014]
- Inter-dataset variability compensation [Aronowitz, 2014, Kanagasundaram et al., 2015]
Bayesian Adaptation of PLDA (Villalba and Lleida, 2014)

- Use a generative model to generate out-of-domain labelled ($\theta_{d_j}$) data and in-domain (target) unlabelled ($\theta_j$) data.
- Unknown labels ($\theta_j$) are modelled as latent variables

Generative model:

$$\phi_j = \mu + \mathbf{V} \mathbf{y}_i + \epsilon_j$$

Joint posterior of the latent variables:

$$P(\mathbf{Y}, \mathbf{Y_d}, \theta, \pi_\theta, \mu, \mathbf{V}, \mathbf{W}, \alpha | \Phi, \Phi_d) \approx q(\mathbf{Y}, \mathbf{Y_d}) q(\theta) q(\pi_\theta) \prod_{r=1}^{d} q(\tilde{\mathbf{v}}_r) q(\mathbf{W}) q(\alpha)$$
Unsupervised clustering of i-vectors (Shum et al., 2014)

Step 1 Use the within-class and between-class covariance matrices of out-of-domain data to compute a pairwise affinity matrix on unlabelled in-domain data.

Step 2 Use the affinity matrix to obtain hypothesized speaker clusters of in-domain data.

Step 3 Linearly interpolate the in-domain and out-of-domain covariance matrices to obtained the adapted matrices:

\[
\Sigma_{\text{adapt}} = \alpha_{\text{wc}} \Sigma_{\text{in}} + (1 - \alpha_{\text{wc}}) \Sigma_{\text{out}}
\]

\[
\Phi_{\text{adapt}} = \alpha_{\text{ac}} \Phi_{\text{in}} + (1 - \alpha_{\text{ac}}) \Phi_{\text{out}}
\]
IDVC aims at explicitly modeling dataset shift variability in the i-vector space and compensating it as a pre-processing cleanup step. No need to use in-domain data to adapt the PLDA model.

**Step 1** Divide the development data into different subsets according to their sources, e.g., different LDC distributions of Switchboard.

**Step 2** Estimate an inter-dataset variability subspace with the largest variability across the mean i-vectors of these subsets.

**Step 3** Remove the variability of i-vectors in this subspace via nuisance attribute projection (NAP).
6 Future Direction
- 6.1. Domain adaptation
- 6.2. Short-utterance speaker verification
- 6.3. Text-dependent speaker recognition
Performance of i-vector/PLDA systems degrades rapidly when the systems are presented with short utterances or utterances with varying durations.

These systems consider long and short utterances as equally reliable.

However, the i-vectors of short utterances have much bigger posterior covariances.

**Popular approaches:**

- Uncertainty Propagation: Propagate the posterior covariances of i-vectors to PLDA [Kenny et al., 2013]
- Full posterior distribution PLDA [Cumani et al., 2014]
In i-vector extraction, besides the posterior mean of the latent variable (i-vector), we also have the posterior covariance matrix, which reflects the uncertainty of the i-vector estimate.

\[
\mathbf{\omega} = \text{cov}(\eta, \eta) \sum_{c=1}^{C} \mathbf{T}_c^T \mathbf{\Sigma}_c^{-1} \tilde{\mathbf{f}}_c
\]

\[
\text{cov}(\eta, \eta) = \mathbf{L}^{-1} = \left( \mathbf{I} + \sum_{c=1}^{C} N_c \mathbf{T}_c^T \mathbf{\Sigma}_c^{-1} \mathbf{T}_c \right)^{-1}
\]

\(\mathbf{L}\) is the precision matrix of the posterior density
\(N_c\) is zero-order sufficient statistics with respect to UBM
\(\tilde{\mathbf{f}}_c\) is first-order sufficient statistics with respect to UBM
Uncertainty propagation (Kenny et al., 2013)

\[ w_{ij} = m + Vh_i + U_{ij}z_{ij} + \epsilon_{ij} \]

- \( U_{ij} \) is the Cholesky decomposition of the posterior covariance matrix \((L_{ij}^{-1})\) of the \( j \)-th i-vector by the \( i \)-th speaker.
- The intra-speaker covariance matrix becomes:
  \[
  \text{cov}(w_{ij}, w_{ij}|h_i) = U_{ij}U_{ij}^T + \Sigma
  \]
  where \( U_{ij}U_{ij}^T \) changes from utterance to utterance, thus reflecting the reliability of the i-vector \( w_{ij} \).
- Scoring is computationally expensive. See [Lin and Mak, 2016] for a fast scoring algorithm.
Outline

1. Introduction
2. Learning Algorithms
3. Learning Models
4. Deep Learning
5. Case Studies
6. Future Direction
   - 6.1. Domain adaptation
   - 6.2. Short-utterance speaker verification
   - 6.3. Text-dependent speaker recognition
Text-dependent SV using short utterances

- Very difficult for short-utterance text-dependent SV
- Text-dependent tasks with the uncertainty propagation version of i-vector/PLDA were unsatisfactory [Stafylakis et al., 2013].
- It is more natural to use HMMs rather than GMMs for text-dependent tasks. But HMMs require local hidden variables, which are difficult to handle because of data fragmentation.

**Emergent approaches:**
- Content matching [Scheffer and Lei, 2014]
- y-vector and JFA backend [Kenny et al., 2015b]
- l-vector backend [Kenny et al., 2015a]
- Hidden supprervector backend [Kenny et al., 2016]
- Use DNN/RNN to extract utterance-level features [Heigold et al., 2016, Bhattacharya et al., 2016, Zeinali et al., 2016]


Contact information

- **Man-Wai Mak**
  The Hong Kong Polytechnic University
  man.wai.mak@polyu.edu.hk
  http://www.eie.polyu.edu.hk/~mwmak/

- **Jen-Tzung Chien**
  National Chiao Tung University
  jtchien@nctu.edu.tw
  http://chien.cm.nctu.edu.tw