SIMPLIFICATION AND OPTIMIZATION OF I-VECTOR EXTRACTION

Ondrej Glembek, Lukas Burget, Pavel Matejka, Martin Karafiat, Patrick Kenny

ICASSP Prague
May 25th 2011
• An i-vector is an **information-rich, low-dimensional fixed-length** vector representing a voiceprint of an arbitrarily long sequence of speech frames. [Dehak 2010]

• We like these little units, because they turn the speaker verification task to pattern recognition problem.
Subspace modeling

Simplification and optimization of i-vector extraction

Prague, May 25 2011

\[ s = m + Tw \]

\[ \mathbf{s} = \begin{bmatrix} \mu^{(1)} \\ \mu^{(2)} \\ \mu^{(3)} \end{bmatrix} \]

\[ \mathbf{n}^T = \begin{bmatrix} n_1^{(1)} \\ n_2^{(2)} \\ n_3^{(3)} \end{bmatrix} \]
\[ s = m + Tw \]

- Matrix \( T \) describes the directions of the highest variability of \( s \)
- Vector \( w \) is a hidden variable and as such, we can impose a prior distribution on it
- Getting adaptation data \( \mathcal{X} \), the posterior of \( w \) can be computed
- What we call an **i-vector** is the mean of this posterior \( w_{\mathcal{X}} \)

\[
p(w) = \mathcal{N}(w; 0, 1) \quad p(w|\mathcal{X}) = \mathcal{N}(w; w_{\mathcal{X}}, L_{\mathcal{X}}^{-1})
\]
\[ N^{(c)}_\chi = \sum_t \gamma^{(c)}_t \]

\[ f^{(c)}_\chi = \sum_t \gamma^{(c)}_t o_t \]

\[ f^{(c)}_\chi \leftarrow f^{(c)}_\chi - N^{(c)} \mathbf{m}^{(c)} \]

\[ \mathbf{m}^{(c)} \leftarrow \mathbf{0} \]

\[ f^{(c)}_\chi \leftarrow \sum^{(c)} - \frac{1}{2} f^{(c)}_\chi \]

\[ T^{(c)} \leftarrow \sum^{(c)} - \frac{1}{2} T^{(c)} \]

\[ \sum^{(c)} \leftarrow \mathbf{I} \]
- With such data structures, the i-vector is computed as

\[ w_{\chi} = L_{\chi}^{-1} T f_{\chi} \]

\[ L_{\chi} = I + \sum_{c=1}^{C} N_{\chi}^{(c)} T^{(c)}' T^{(c)} \]

\[ O(C F M + C M^2 + M^3) \quad O(C F M + C M^2) \]

computational complexity  
memory complexity

C ... # of GMM components  
F ... Feature dimensionality  
M ... Subspace dimensionality
Motivation for simplifications

- Port the application to small-scale devices
- Prepare the framework for discriminative training

\[
\begin{align*}
    w_x &= L_x^{-1} T' f_x \\
    L_x &= I + \sum_{c=1}^{C} N_x^{(c)} T^{(c)'} T^{(c)}
\end{align*}
\]
Simplification 1

\[ \mathbf{L}_\mathbf{x} = \mathbf{I} + \sum_{c=1}^{C} N_{\mathbf{x}}^{(c)} \mathbf{T}^{(c)'} \mathbf{T}^{(c)} \]

- Simplifying 0\(^{th}\) order stats:

\[ \bar{N}_{\mathbf{x}}^{(c)} = \omega^{(c)} N_{\mathbf{x}} \]

\[ \mathbf{W} = \sum_{c=1}^{C} \omega^{(c)} \mathbf{T}^{(c)'} \mathbf{T}^{(c)} \]

\[ \bar{\mathbf{L}}_{\mathbf{x}} = \mathbf{I} + N_{\mathbf{x}} \mathbf{W} \]

\[ O(CFM + CM^2 + M^3) \quad O(CFM + CM^2) \]

computational complexity  

memory complexity
Simplification 2

\[ \mathbf{L}_\mathcal{X} = \mathbf{I} + \sum_{c=1}^{C} N^{(c)} \mathbf{T}^{(c)'} \mathbf{T}^{(c)} \]

- Let us assume, that there exist a linear transformation \( \mathbf{G} \) that would diagonalize all \( \mathbf{T}^{(c)'} \mathbf{T}^{(c)} \).

\[ \hat{\mathbf{L}}_\mathcal{X} = \mathbf{G}' \mathbf{G} + \sum_{c=1}^{C} N^{(c)} \mathbf{G}' \mathbf{T}^{(c)'} \mathbf{T}^{(c)} \mathbf{G} \]

\[ \mathbf{L}_\mathcal{X} = \mathbf{G}^{-1}' \hat{\mathbf{L}}_\mathcal{X} \mathbf{G}^{-1} \]

- Also, \( \hat{\mathbf{L}}_\mathcal{X} \) is diagonal, so the inversion is trivial, which can be implemented effectively.
• Effectively written

\[ \hat{L}_x = \text{Diag} \left( \text{diag}(G'G) + Vn_x \right) \]

\[ \hat{w}_x = G\hat{L}_x^{-1}G'T'f_x \]

• where \( V \) packs diagonal matrices into column vectors:

\[ \text{diag}(G'T^{(c)'}T^{(c)}G) \]

\[ O(CFM + CM^2) \]

\[ O(CFM + CM^2 + M^2) \]

computational complexity

\[ O(CFM + M^2 + CM) \]

memory complexity
• The simplest approach to estimate the orthogonalization matrix $G$ is PCA.

• Inspired by other fields, we also tried Heteroscedastic Linear discriminant analysis (HLDA) [Kumar 1997, Gales 1999].

Obviously, we want the transform to rotate the space 45 degrees, but the average within-class covariance matrix would be diagonal $\Rightarrow$ PCA fails.
i-vector extractor training

- i-vector extractor $T$ training procedure comprises collecting the following accumulators

\[
C = \sum_i f_i w'_i
\]

\[
A^{(c)} = \sum_i N^{(c)}_i \left( L_i^{-1} + w_i w'_i \right)
\]

- The update is the given as

\[
T^{(c)} = CA^{(c)}^{-1}
\]

- We see that the simplifications can be applied even in training
Experimental setup

• Features
  MFCC 19+E, short time cepstral mean and variance normalization over 300 frames, $\Delta + \Delta \Delta$

• Training set
  SWII, phase 2 and 3, SW cellular, NIST2004-6, Fisher English 1,2

• Test set
  NIST SRE 2010
  Extended core condition 5 – tel-tel, female only

• Performance set
  • MATLAB, single core, Intel Xeon CPU X5670, 2.93GHz
  • 50 randomly picked utterances from MIXER corpus

• UBM
  Diagonal covariance, 2048 component UBM
• Summary in numbers:
  \[ C = 2048 \]
  \[ F = 60 \]
  \[ M = 400 \]

• \( M = 400 \) has been chosen as a tradeoff between performance and technical conditions.

• With simplification 1, we can afford to use \( M = 800 \) with the same memory consumption
Results

Simplification and optimization of i-vector extraction

Prague, May 25 2011
## Results – Comparison of speed

<table>
<thead>
<tr>
<th></th>
<th>absolute [sec]</th>
<th>relative to 400 baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 baseline</td>
<td>13.70</td>
<td>100.00%</td>
</tr>
<tr>
<td>400 simple 1</td>
<td>1.01</td>
<td>7.37%</td>
</tr>
<tr>
<td>400 simple 2</td>
<td>0.54</td>
<td>3.94%</td>
</tr>
<tr>
<td>800 baseline</td>
<td>65.75</td>
<td>480.00%</td>
</tr>
<tr>
<td>800 simple 1</td>
<td>3.64</td>
<td>26.57%</td>
</tr>
<tr>
<td>800 simple 2</td>
<td>1.11</td>
<td>8.10%</td>
</tr>
</tbody>
</table>

**BASELINE** \(O(CFM + CM^2 + M^3)\)

**SIMPLE 1** \(O(CFM + M^3)\)

**SIMPLE 2** \(O(CFM)\)
<table>
<thead>
<tr>
<th></th>
<th>constant</th>
<th>algorithm specific</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 baseline</td>
<td>422.96</td>
<td>2,500.00</td>
<td>2,923.00</td>
</tr>
<tr>
<td>400 simple 1</td>
<td></td>
<td>1.22</td>
<td>424.18</td>
</tr>
<tr>
<td>400 simple 2</td>
<td></td>
<td>7.47</td>
<td>430.43</td>
</tr>
<tr>
<td>800 baseline</td>
<td>802.84</td>
<td>10,000.00</td>
<td>10,802.84</td>
</tr>
<tr>
<td>800 simple 1</td>
<td></td>
<td>4.88</td>
<td>807.83</td>
</tr>
<tr>
<td>800 simple 2</td>
<td></td>
<td>17.38</td>
<td>820.23</td>
</tr>
</tbody>
</table>

**BASELINE** \( O(CFM + CM^2) \)

**SIMPLE 1** \( O(CFM + M^2) \)

**SIMPLE 2** \( O(CFM + M^2 + CM) \)
<table>
<thead>
<tr>
<th></th>
<th>$\text{DCF}_{\text{new}}$</th>
<th>$\text{DCF}_{\text{old}}$</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 baseline</td>
<td>0.5460</td>
<td>0.1722</td>
<td>3.40</td>
</tr>
<tr>
<td>400 simple 1</td>
<td>0.5376</td>
<td>0.1729</td>
<td>3.42</td>
</tr>
</tbody>
</table>
• We managed to significantly simplify the state-of-the-art technique in terms of **speed and memory** with a sacrifice of slight degradation in recognition performance

• We have simplified the formulas so that they are easily differentiable and usable for numerical optimizations for **discriminative training**

• We managed to fit the i-vector based SRE system into a real **cell-phone application** (EC-sponsored MOBIO project)
Thank you