On sequence kernels for SVM classification of sets of vectors: 
application to speaker verification

Khalid Daoudi (IRIT-UPS, Toulouse, France)

Major part of the Ph.D. work of Jérôme Louradour
In collaboration with Francis Bach (ARMINES)
within E-TEAM 11

Institut de Recherche en Informatique de Toulouse
Text independent speaker verification

A binary classification task
Determine if a speech sequence has been uttered by a target speaker

Classical approach
- séquence test
- locuteur cible
- imposteurs, "Monde"

Classifieur

PRÉ-TRAITEMENT

ATTRIBUTION DE SCORE

PRISE DE DÉCISION

"locuteur cible" ou "imposteur"

APPRENTISSAGE
Text independent speaker verification

A binary classification task
Determine if a speech sequence has been uttered by a target speaker

Classical approach: UBM-GMM system
- Probabilistic GMM modeling (generative)
Text independent speaker verification

A binary classification task
Determine if a speech sequence has been uttered by a target speaker

Classical approach: UBM-GMM system

- **Motivation**: apply SVM, powerful for binary classification

- Probabilistic GMM modeling (generative)
Support Vector Machines (SVM)

- Good theoretical foundations
- Well mastered learning algorithms
- Good results in static data classification

In speech processing . . .

- Extension to dynamic data is not easy
- Size of training corpus $\Rightarrow$ time & memory consuming
- Bad results of SVM when applied at the frame level
Support Vector Machines (SVM)

- Good theoretical foundations
- Well mastered learning algorithms
- Good results in static data classification

In speech processing . . .
- Extension to dynamic data is not easy
- Size of training corpus $\Rightarrow$ time & memory consuming
- Bad results of SVM when applied at the frame level

- Conception and study of sequence kernels for speaker verification
Outline

1. Sequence kernels
2. FSNS sequence kernels
3. Experimental evaluation of sequence kernels
Basics on kernels

- Similarity measure
- Mercer property: symmetric, positive definite
  \[ k(x, y) = \Phi(x)^\top \Phi(y) \]

\( \Phi \): expansion in a Feature Space \( \mathbb{R}^D \) (dimension \( D \leq +\infty \))
Sequence kernels

Three families of kernels

1. Mutual Information kernels
   Based on *a priori* data distribution

2. Kernels between probability densities
   Sequence $\mapsto$ distribution

3. Combination of vector kernels
   Function of kernel values between inter(intra)-sequence elements
Sequence kernels

Three families of kernels

1. Mutual Information kernels
   Based on \textit{a priori} data distribution

2. Kernels between probability densities
   Sequence $\mapsto$ distribution

3. Combination of vector kernels
   Function of kernel values between inter(intra)-sequence elements

Definition of a sequence

“Variable-length set of acoustic vectors
- same speaker,
- same recording session”
Sequence kernels used in speaker verification

**Mutual Information kernels**

Exploit the a priori (generative) UBM which parameters $\theta_o$ are estimated on non-labeled data

**Fisher kernel** [Jaakkola and Haussler, 1998]

- Approximation of a Mutual Information kernel [Seeger, 2002]
  \[ \kappa(X, Y) = \phi(X)^\top S^{-1} \phi(Y) \]

- $\phi(X) = \nabla_{\theta} \log p(X|\theta)|_{\theta=\theta_o}$ (Fisher expansion)
  
  $S$: second moments of $\phi$ (Fisher information matrix)
Mutual Information kernels

Exploit the a priori (generative) UBM which parameters $\theta_o$ are estimated on non-labeled data

**Fisher kernel** [Jaakkola and Haussler, 1998]

- Approximation of a Mutual Information kernel [Seeger, 2002]
  \[ \kappa(X, Y) = \phi(X)^\top S^{-1} \phi(Y) \]

- $\phi(X) = \nabla_\theta \log p(X|\Theta)|_{\theta=\theta_o}$ (Fisher expansion)
  
  $S$ : second moments of $\phi$ (Fisher information matrix)
Mutual Information kernels

Exploit the a priori (generative) UBM which parameters $\theta_0$ are estimated on non-labeled data

**Fisher kernel** [Jaakkola and Haussler, 1998]

- Approximation of a Mutual Information kernel [Seeger, 2002]

$$\kappa(X, Y) = \phi(X)^\top S^{-1} \phi(Y)$$

- $\phi(X) = \nabla_\theta \log p(X|\theta)|_{\theta=\theta_0}$ (Fisher expansion)

$S$: second moments of $\phi$ (Fisher information matrix)

Distance between Fisher expansions
**Kernels between probability densities**

$$X, Y \xrightarrow{\text{Generative learning}} p_X, p_Y$$

**probability product kernels** [Jebara and Kondor, 2003]

$$\kappa(X, Y) = \int p_X(z)^q p_Y(z)^q \, dz$$

- Analytic form a GMM with degree $q = 1$ [Lyu, 2005]
- “spherical” normalization for more robustness:

$$\hat{\kappa}(X, Y) = \frac{\kappa(X, Y)}{\sqrt{\kappa(X, X)\kappa(Y, Y)}}$$

**Exponential embedding of divergences**

- Analogous of the Gaussian kernel:

$$\kappa(X, Y) = e^{-\frac{D(p_X, p_Y)^2}{2\rho^2}}$$

- $D$ : Distance between GMMs, approximation of KL divergence [Do, 2003]

y “supervectors GMM” Approach [Campbell et al., 2006]
Combination of vector kernels

Sequences of vectors
\[ X = \{ x_t \mid t = 1 \cdots T_x \} \]
\[ Y = \{ y_{t'} \mid t' = 1 \cdots T_y \} \]

Similarity between vectors
\[ k(x, y) = \phi(x) \top \phi(y) \]

Mercer kernel

Similarity between sets of vectors
\[ \kappa(X, Y) \]

function of \( k(x_t, y_{t'}) \), \( k(x_t, x_{t'}) \), \( k(y_t, y_{t'}) \), \ldots

Simple example:
\[ \kappa(X, Y) = \frac{1}{T_x T_y} \sum_{t=1}^{T_x} \sum_{t'=1}^{T_y} k(x_t, y_{t'}) \]

complexity \( O(T^2) \) for each kernel computation
Combination of vector kernels

**GLDS kernel** [Campbell, 2001]

\[
\kappa(X, Y) = \left( \frac{1}{T_X} \sum_t \phi_q(x_t) \right)^\top S_B^{-1} \left( \frac{1}{T_Y} \sum_{t'} \phi_q(y_{t'}) \right)
\]

- \( \phi_q \): Polynomial expansion \((\mathbb{R}^d \rightarrow \mathbb{R}^D)\)
- \( D = \frac{(q+d)!}{q!d!} \) monomes of degree \( \leq q \)
- \( S_B \): Matrix of second moments of \( \phi_q \) estimated on \( B \)

- **Normalization by** \( S_B \):
  - Kernel \( \sim \) scoring on \( X \) a discr. model learned on \( Y \)
  - Same amplitude for each component of the *Feature Space*

- **Explicit expansion of data**:
  - high efficiency during test (linear SVM model)
  - Impossible to use expansions \( \phi_q \) of high or infinite dimension
    (in practice, maxi degree \( q = 3 \))
Combination of vector kernels

**GLDS kernel** [Campbell, 2001]

\[
\kappa(X, Y) = \left( \frac{1}{T_X} \sum_t \Phi_q(x_t) \right)^\top S_B^{-1} \left( \frac{1}{T_Y} \sum_t' \Phi_q(y_{t'}) \right)
\]

\(\Phi_q\): Polynomial expansion \((\mathbb{R}^d \rightarrow \mathbb{R}^D)\)

\[
D = \frac{(q+d)!}{q!d!}
\]

monomes of degree \(\leq q\)

\(S_B\): Matrix of second moments of \(\Phi_q\) estimated on \(B\)

- **Normalization by** \(S_B\):
  - Kernel \(\sim\) scoring on \(X\) a discr. model learned on \(Y\)
  - Same amplitude for each component of the *Feature Space*

- **Explicit expansion of data**:
  - high efficiency during test (linear SVM model)
  - Impossible to use expansions \(\Phi_q\) of high or infinite dimension
    (in practice, maxi degree \(q = 3\))
Outline

1. Sequence kernels
2. FSNS sequence kernels
3. Experimental evaluation of sequence kernels
FSNS sequence kernels

Definition

Extension of the GLDS kernel: FSNS kernels

FSNS kernels (Feature Space Normalized Sequence kernels)

\[
\kappa(X, Y) = \left( \frac{1}{T_X} \sum_t \phi(x_t) \right)^\top (S_B + \varepsilon I)^{-1} \left( \frac{1}{T_Y} \sum_t' \phi(y_{t'}) \right)
\]

- \( \phi \): any Mercer expansion
- \( S_B \): second moments matrix of \( \phi \)
- Regularization \( \varepsilon > 0 \): necessary if \( \phi \) is of high dimension

Objectif:

- avoid to compute \( \phi \)
- rewrite using the Mercer kernel \( k = \phi^\top \phi \)
Extension of the GLDS kernel: FSNS kernels

**FSNS kernels (Feature Space Normalized Sequence kernels)**

\[
\kappa(X, Y) = \left( \frac{1}{T_X} \sum_t \Phi(x_t) \right)^\top \left( \mathbf{S}_B + \varepsilon \mathbf{I} \right)^{-1} \left( \frac{1}{T_Y} \sum_{t'} \Phi(y_{t'}) \right)
\]

- \( \Phi \): any Mercer expansion
- \( \mathbf{S}_B \): second moments matrix of \( \Phi \)
- Regularization \( \varepsilon > 0 \): necessary if \( \Phi \) is of high dimension

**Objectif:**
- Avoid to compute \( \Phi \)
- Rewrite using the Mercer kernel \( k = \Phi^\top \Phi \)

**FSMS kernels (Feature Space Mahalanobis Sequence kernels)**

\[
\kappa(X, Y) = \left( \frac{1}{T_X} \sum_t \Phi(x_t) \right)^\top \left( \mathbf{\Sigma}_B + \varepsilon \mathbf{I} \right)^{-1} \left( \frac{1}{T_Y} \sum_{t'} \Phi(y_{t'}) \right)
\]

- \( \mathbf{\Sigma}_B \): covariance matrix of \( \Phi \)
- SVM are invariant to translations in the Feature Space
  - \( \Rightarrow \) same as FSNS with centring of \( \Phi \)
- Kernel \( \sim \) KL divergence between Gaussians in the Feature Space
Dual form of FSNS kernels

Training (Background) data: \( B = \{ b_i \mid i = 1 \cdots N \} \)

- **Gram matrix**
  \[
  K = \begin{bmatrix}
  k(b_1,b_1) & \cdots & k(b_1,b_N) \\
  \vdots & \ddots & \vdots \\
  k(b_N,b_1) & \cdots & k(b_N,b_N)
  \end{bmatrix}
  \]

- **Empirical map**
  \[
  \psi_B(x) = \begin{bmatrix}
  k(b_1,x) \\
  \vdots \\
  k(b_N,x)
  \end{bmatrix}
  \]

**Proposition** (without regularization, \( \varepsilon = 0 \))

\[
\kappa(X,Y) = \left( \frac{1}{T_X} \sum_t \Phi(x_t) \right)^\top \mathbf{S}_B^{-1} \left( \frac{1}{T_Y} \sum_{t'} \Phi(y_{t'}) \right)
\]

\[
= \left( \frac{1}{T_X} \sum_t \psi_B(x_t) \right)^\top \left( \frac{1}{N} K^2 \right)^{-1} \left( \frac{1}{T_Y} \sum_{t'} \psi_B(y_{t'}) \right)
\]
Dual form of FSNS kernels

Training (Background) data: \( B = \{ b_i \mid i = 1 \cdots N \} \)

**Gram matrix**

\[
K = \begin{bmatrix}
k(b_1, b_1) & \cdots & k(b_1, b_N) \\
\vdots & \ddots & \vdots \\
k(b_N, b_1) & \cdots & k(b_N, b_N)
\end{bmatrix}
\]

**Empirical map**

\[
\psi_B(x) = \begin{bmatrix}
k(b_1, x) \\
\vdots \\
k(b_N, x)
\end{bmatrix}
\]

**Proposition**

\[
\kappa(X, Y) = \left( \frac{1}{T_X} \sum_t \Phi(x_t) \right)^\top \left( S_B + \varepsilon I \right)^{-1} \left( \frac{1}{T_Y} \sum_{t'} \Phi(y_{t'}) \right)
\]

\[
= \left( \frac{1}{T_X} \sum_t \psi_B(x_t) \right)^\top \left( \frac{1}{N} K^2 + \varepsilon K \right)^{-1} \left( \frac{1}{T_Y} \sum_{t'} \psi_B(y_{t'}) \right)
\]

- **Hypothesis**: \( \Phi(b_i) \) span the training "\( \Phi(x) \)"
Dual form of FSNS kernels

Training (Background) data: \( B = \{ b_i \mid i = 1 \cdots N \} \)

**Gram matrix**

\[
K = \begin{bmatrix}
k(b_1,b_1) & \cdots & k(b_1,b_N) \\
\vdots & \ddots & \vdots \\
k(b_N,b_1) & \cdots & k(b_N,b_N)
\end{bmatrix}
\]

**Empirical map**

\[
\psi_B(x) = \begin{bmatrix} k(b_1,x) \\
\vdots \\
k(b_N,x) \end{bmatrix}
\]

**Proposition (with centering)**

\[
k(X, Y) = \left( \frac{1}{T_X} \sum_t \phi(x_t) \right)^\top \left( \Sigma_B + \epsilon I \right)^{-1} \left( \frac{1}{T_Y} \sum_{t'} \phi(y_{t'}) \right)
\]

\[
= \left( \frac{1}{T_X} \sum_t \psi_B(x_t) \right)^\top \left( \frac{1}{N} K \Pi K + \epsilon K \right)^{-1} \left( \frac{1}{T_Y} \sum_{t'} \psi_B(y_{t'}) \right)
\]

- **Hypothesis**: the \( \phi(b_i) \) span the training \( \phi(x) \)
- **Centering**: \( \Pi = I - \frac{1}{N} 1 \) (instead of \( I \))

Noyaux de séquences pour la vérification du locuteur par SVM

Khalid Daoudi (IRIT-UPS, Toulouse, France)
Computational complexity

Dot product of normalized expansions

\[ \kappa(X, Y) = \overline{\Phi}(X)^\top M_B \overline{\Phi}(Y) = \left\langle U\overline{\Phi}(X), U\overline{\Phi}(Y) \right\rangle \quad (1) \]

\[ = \overline{\psi_B}(X)^\top R_B \overline{\psi_B}(Y) = \left\langle V\overline{\psi_B}(X), V\overline{\psi_B}(Y) \right\rangle \quad (2) \]

<table>
<thead>
<tr>
<th>Pre-computation of U/V</th>
<th>form (1)</th>
<th>form (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence expansion U\overline{\Phi}/V\overline{\psi_B}</td>
<td>(O(D^3))</td>
<td>(O(N^3))</td>
</tr>
<tr>
<td>Dot product comput.</td>
<td>(O(TD^2))</td>
<td>(O(TN^2))</td>
</tr>
<tr>
<td></td>
<td>(O(D))</td>
<td>(O(N))</td>
</tr>
</tbody>
</table>

\(D\) : dimension of the Feature Space (size of \(\Phi\))
\(N\) : number of background vectors (size of \(\psi\))

+ Possibility to use expansions \(\Phi\) of infinite dim (Gaussian kernel)
- Complexity problem for large databases
Computational complexity

### Dot product of normalized expansions

\[
\kappa(X, Y) = \overline{\Phi}(X)^\top M_B \overline{\Phi}(Y) = \langle U\overline{\Phi}(X), U\overline{\Phi}(Y) \rangle \quad (1)
\]
\[
= \overline{\psi}_B(X)^\top R_B \overline{\psi}_B(Y) = \langle V\overline{\psi}_B(X), V\overline{\psi}_B(Y) \rangle \quad (2)
\]

<table>
<thead>
<tr>
<th>Pre-computation of ( U/V )</th>
<th>( O(D^3) )</th>
<th>( O(N^3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence expansion ( U\overline{\Phi}/V\overline{\psi}_B )</td>
<td>( O(TD^2) )</td>
<td>( O(TN^2) )</td>
</tr>
<tr>
<td>Dot product comput.</td>
<td>( O(D) )</td>
<td>( O(N) )</td>
</tr>
</tbody>
</table>

\( D \) : dimension of the *Feature Space* (size of \( \Phi \))

\( N \) : number of background vectors (size of \( \psi \))

- Possibility to use expansions \( \Phi \) of infinite dim (Gaussian kernel)
- Complexity problem for large databases
- Objective : find an appropriate approximation
**Kernel Approximation**

**Goal**

1. Reduce the size of the empirical map $\psi$
2. Keep a maximum of information

**ICD : Incomplete Cholesky Decomposition** [Fine and Scheinberg, 2001]

- Selection of a sub-population of background vectors:
  
  \[ C = \{ b_{p_1}, \ldots, b_{p_i}, \ldots, b_{p_m} \} \subset B \]
  
  \[ I = \{ p_1, \ldots, p_i, \ldots, p_m \} \subset \{1\ldots N\} \]

- Low-rank approximation of the Gram matrix:
  
  \[ K \approx L_I = K(:, I)K(I, I)^{-1}K(:, I)^\top \]

  \[ \min_I \text{tr}(K - L_I) \equiv \min_C \sum \| \phi(b_i) - \Phi_C(b_i) \|^2 \]

  + CPU and memory
Kernel Approximation

**Goal**

1. Reduce the size of the empirical map $\psi$
2. Keep a maximum of information

**ICD : Incomplete Cholesky Decomposition**

Training data
Codebook

ICD, Gaussian kernel
Approximate Form of FSNS kernels

**Proposition**

**ICD**

\[
\kappa(X, Y) = \overline{\psi}_B(X)^\top R_B \overline{\psi}_B(Y) \approx \overline{\psi}_C(X)^\top R_{B \times C} \overline{\psi}_C(Y)
\]

- Expansion of size \( m \ll N \):
  \[
  \overline{\psi}_C(X) = \frac{1}{T_X} \sum_t \begin{bmatrix} k(b_{p1}, x_t) \\ \vdots \\ k(b_{pm}, x_t) \end{bmatrix}
  \]
  \[
  R_{B \times C} = \left( \frac{1}{N} K(:, I) \prod K(:, I)^\top + \varepsilon K(I, I) \right)^{-1}
  \]

- **Complexity**

<table>
<thead>
<tr>
<th>Expansion norm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(TD^2) )</td>
</tr>
<tr>
<td>( O(TN^2) )</td>
</tr>
<tr>
<td>( O(Tm^2) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dot product</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(D) )</td>
</tr>
<tr>
<td>( O(N) )</td>
</tr>
<tr>
<td>( O(m) )</td>
</tr>
</tbody>
</table>
Outline

1. Sequence kernels
2. FSNS sequence kernels
3. Experimental evaluation of sequence kernels
Data

Speech corpus: NIST Speaker Recognition Evaluation

Development: NIST 2003 and 2004
- Background corpus (1/2)
- Validation corpus (2/2)
  - Hyper-parameters of kernels
  - Parameter $C$ (SVM learning)
  - Decision threshold

Evaluation: NIST 2005
~ 18000 tests, 400 target speakers

\[ \text{DCF} = \tau_{\text{FR}} P_{\text{loc}} \text{FR}\% + \tau_{\text{FA}} P_{\text{imp}} \text{FA}\% \]
**Data**

**Speech corpus:** NIST Speaker Recognition Evaluation

**Development:** NIST 2003 and 2004
- Background corpus (1/2)
- Validation corpus (2/2)
  - Hyper-parameters of kernels
  - Parameter $C$ (SVM learning)
  - Decision threshold

**Evaluation:** NIST 2005
- $\sim 18000$ tests, 400 target speakers

$$DCF = \tau_{FR} P_{loc} FR\% + \tau_{FA} P_{imp} FA\%$$

**Preprocessing**

<table>
<thead>
<tr>
<th>SVM</th>
<th>GMM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>acoustic vectors</strong></td>
<td>$MFCC + \Delta MFCC + \Delta \log E$</td>
</tr>
<tr>
<td><strong>silence suppression</strong></td>
<td>clustering of the energy</td>
</tr>
<tr>
<td><strong>normalization</strong></td>
<td>centring-reduction</td>
</tr>
</tbody>
</table>
Development: Centring & Regularization?

\[
\kappa(X, Y) = \overline{\psi}_C(X)^\top \left[ \frac{1}{N} K(:, I)^\top P K(:, I) + \varepsilon K(I, I) \right]^{-1} \overline{\psi}_C(Y)
\]

<table>
<thead>
<tr>
<th>Centring</th>
<th>Regul.</th>
<th>EER(%)</th>
<th>DCF_{min}</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>(P = \Pi) (\varepsilon = 10^{-4})</td>
<td>15.83</td>
<td>63.5</td>
</tr>
<tr>
<td>--</td>
<td>(P = \Pi) (\varepsilon = 10^{-6})</td>
<td>14.11</td>
<td>53.1</td>
</tr>
<tr>
<td>—</td>
<td>(P = \Pi) ((\varepsilon = 0))</td>
<td>10.62</td>
<td>49.3</td>
</tr>
<tr>
<td>—</td>
<td>((P = I)) ((\varepsilon = 0))</td>
<td>10.95</td>
<td>49.9</td>
</tr>
</tbody>
</table>

- No significant gain with centering
- Regularization degrades
Experimental evaluation of sequence kernels

**FSNS kernels**

**Developement**: Choice and tuning of the vector kernel

\[
\kappa(X, Y) = \overline{\psi_C}(X)^\top \left[ \frac{1}{N} K(:, I)^\top P K(:, I) + \varepsilon K(I, I) \right]^{-1} \overline{\psi_C}(Y)
\]

<table>
<thead>
<tr>
<th>(\rho)</th>
<th>EER(%)</th>
<th>DCF_{\text{min}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>2(\rho_0)</td>
<td>15.7</td>
<td>60.2</td>
</tr>
<tr>
<td>(\rho_0/2)</td>
<td>11.90</td>
<td>50.5</td>
</tr>
<tr>
<td>(\rho_0)</td>
<td>10.95</td>
<td>49.9</td>
</tr>
</tbody>
</table>

- Best results are obtained with a Gaussian kernel \(k(x, y) = e^{-\frac{\|x - y\|^2}{2\rho^2}}\).
- ... using a good spread parameter \(\rho \approx \rho_0 = \sqrt{d\bar{\sigma}}\).
Experimental evaluation of sequence kernels

FSNS kernels

Developement: Size of the codebook

\[
\kappa(X, Y) = \overline{\psi_C(X)^\top \left[ \frac{1}{N} \mathbf{K}(:, I)^\top \mathbf{P} \mathbf{K}(:, I) + \varepsilon \mathbf{K}(I, I) \right]^{-1} \overline{\psi_C(Y)}}
\]

<table>
<thead>
<tr>
<th>Codebook size</th>
<th>EER(%)</th>
<th>DCF_{min}</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m = 300 )</td>
<td>15.70</td>
<td>65.0</td>
</tr>
<tr>
<td>( m = 600 )</td>
<td>14.38</td>
<td>61.2</td>
</tr>
<tr>
<td>( m = 1250 )</td>
<td>13.59</td>
<td>57.4</td>
</tr>
<tr>
<td>( m = 2500 )</td>
<td>12.52</td>
<td>53.5</td>
</tr>
<tr>
<td>( m = 5000 )</td>
<td>10.95</td>
<td>49.9</td>
</tr>
<tr>
<td>( m = 8000 )</td>
<td>10.17</td>
<td>48.9</td>
</tr>
</tbody>
</table>

Augmenting la codebook size improve the performances

... until a certain point (\( m \sim 5000 \))
Experimental evaluation of sequence kernels

**FSNS kernels**

**Evaluation**: FSNS kernels vs. Classical approaches

<table>
<thead>
<tr>
<th>Validation corpus</th>
<th>EER(%)</th>
<th>DCF$_{min}$</th>
<th>Evaluation corpus</th>
<th>EER</th>
<th>DCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) GLDS SVM</td>
<td>12.80</td>
<td>51.9</td>
<td>(1) GLDS SVM</td>
<td>12.54</td>
<td>48.8</td>
</tr>
<tr>
<td>(2) FSNS SVM</td>
<td>10.55</td>
<td>47.5</td>
<td>(2) FSNS SVM</td>
<td>11.91</td>
<td>41.6</td>
</tr>
<tr>
<td>(3) UBM-GMM</td>
<td>11.48</td>
<td>49.1</td>
<td>(3) UBM-GMM</td>
<td>12.06</td>
<td>40.6</td>
</tr>
</tbody>
</table>

- FSNS kernel : better performances than GLDS kernel
- FSNS-SVM competitive with UBM-GMM
Evaluation: Perspectives of fusion

Validation corpus

<table>
<thead>
<tr>
<th></th>
<th>EER (%)</th>
<th>DCF min</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) GLDS SVM</td>
<td>12.80</td>
<td>51.9</td>
</tr>
<tr>
<td>(2) FSNS SVM</td>
<td>10.55</td>
<td>47.5</td>
</tr>
<tr>
<td>(3) UBM-GMM</td>
<td>11.48</td>
<td>49.1</td>
</tr>
<tr>
<td>(1+2) fusion</td>
<td>no improvment</td>
<td></td>
</tr>
<tr>
<td>(1+3) fusion</td>
<td>10.32</td>
<td>45.0</td>
</tr>
<tr>
<td>(2+3) fusion</td>
<td>9.50</td>
<td>43.5</td>
</tr>
</tbody>
</table>

Evaluation corpus

<table>
<thead>
<tr>
<th></th>
<th>EER</th>
<th>DCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) GLDS SVM</td>
<td>12.54</td>
<td>48.8</td>
</tr>
<tr>
<td>(2) FSNS SVM</td>
<td>11.91</td>
<td>41.6</td>
</tr>
<tr>
<td>(3) UBM-GMM</td>
<td>12.06</td>
<td>40.6</td>
</tr>
<tr>
<td>(1+2) fusion</td>
<td>no improvment</td>
<td></td>
</tr>
<tr>
<td>(1+3) fusion</td>
<td>10.54</td>
<td>38.8</td>
</tr>
<tr>
<td>(2+3) fusion</td>
<td>9.71</td>
<td>37.0</td>
</tr>
</tbody>
</table>

Gain in discriminative / generative fusion
Experimental evaluation of sequence kernels

Evaluation: All sequence kernels

<table>
<thead>
<tr>
<th></th>
<th>EER</th>
<th>DCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability product</td>
<td>13.92</td>
<td>52.1</td>
</tr>
<tr>
<td>GLDS kernel</td>
<td>12.54</td>
<td>48.8</td>
</tr>
<tr>
<td>Fisher kernel</td>
<td>11.90</td>
<td>44.0</td>
</tr>
<tr>
<td>FSNS kernel</td>
<td>11.91</td>
<td>41.6</td>
</tr>
<tr>
<td>UBM-GMM (ref)</td>
<td>12.06</td>
<td>40.6</td>
</tr>
<tr>
<td>Supervectors GMM</td>
<td><strong>10.40</strong></td>
<td><strong>37.7</strong></td>
</tr>
</tbody>
</table>

Best results: exploiting GMM in SVM
## Evaluation: Fusion

<table>
<thead>
<tr>
<th>Kernel Type</th>
<th>EER</th>
<th>DCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) FSNS kernel</td>
<td>11.91</td>
<td>41.6</td>
</tr>
<tr>
<td>(2) UBM-GMM</td>
<td>12.06</td>
<td>40.6</td>
</tr>
<tr>
<td>(3) Supervectors GMM</td>
<td>10.40</td>
<td>37.7</td>
</tr>
</tbody>
</table>

- (2+3) fusion: no improvement
- (1+2) fusion: **9.71** | 37.0
- (1+3) fusion: **10.28** | **36.1**

**Gain in GMM / SVM fusion**
Conclusions and future work

Theoretical and experimental exploration of kernels between variable-length sets

- A lot of possible kernels
- A novel kernel (FSNS)
- Experimental comparison on NIST SRE
- Best performance: fusion FSNS/GMM supervectors

Several ways to improve SVM for speaker verification

- Model adaptation
- String kernels for high-level features (prosody...)
- How to handle SVM scores?