IRIT Speaker Verification System NIST SRE 2005 Submission

Jérôme Louradour

Institut de Recherche en Informatique de Toulouse
CNRS
INPT
UPS
UT1

May 25, 2005

Introduction

Applying SVM to a speaker verification task :

- Vector-level approach

$$score(\text{sequence } X = x_t) = \frac{1}{T} \sum_t score(x_t)$$

$\ominus$ Poor performance (overlap, noise...)
$\ominus$ Lots of memory needed, intractable training algorithm

Introduction

Applying SVM to a speaker verification task :

- Vector-level approach

$$\mathit{score}(\text{sequence } X = x_t) = \frac{1}{T}\sum_t \mathit{score}(x_t)$$

  $\ominus$ Poor performance (overlap, noise...)

  $\ominus$ Lots of memory needed, intractable training algorithm

- Sequence-level kernels
  "Quantifying the similarity between sequences"
  Based on generative/informative models (*e.g.* GMM)
  $\ominus$ Perform roughly as GMM, while less efficient
  - KL divergence between GMM estimated on the 2 sequences
    $\ominus$ Problems for estimating robustly models parameters
  - Fisher kernel (score space)
    $\ominus$ Computationnally heavy

Introduction

Applying SVM to a speaker verification task :

- Sequence-level kernels
  "Quantifying the similarity between sequences"
  Based on *"training on a sequence and testing on another one"*
  - GLDS kernel

$$\kappa_{GLDS} \equiv \begin{array}{c}\text{Mapping explicitly sequences} \\ \text{to a (high-dimensional) feature space} \\ + \text{ dot product}\end{array}$$

$\oplus$ Powerful, efficient
$\ominus$ Limited choice for the feature space

Problem formulation

Training corpus $(\tau_n, s_n)_{n=1\dots N}$:
- Real vectors $\tau_n \in \Re^d$
- Desired outputs $s_n \in \{0, 1\}$.

$$\left\{ \begin{array}{rcl} s_n & = & 0 \text{ for fixed set of background data } U = (u_t)_{t=1\dots T_U} \\ s_n & = & 1 \text{ for sequence } A = (a_t)_{t=1\dots T_A} \text{ by target speaker} \\ U \cup A & = & (\tau_n) \end{array} \right. \tag{1}$$

Finding a discriminant function $f : \Re^d \rightarrow \Re$:

$$\min_{f \in \mathcal{H}} \sum_{n=1}^{N} L\left(f(\tau_n), s_n\right) \tag{2}$$

- L : loss function
- $\mathcal{H}$: searching space for f

Problem reformulation

Un important subclass of such problems are genrated by a kernel K, which generates $\mathcal{H}_K$, a *Reproducing Kernel Hilbert Space* (RKHS).

- [Wahba90] showed that the solution to (2) in $\mathcal{H}_K$ has the form :

$$f(x) = \sum_{n=1}^{N} \omega_n K(\tau_n, x) \tag{3}$$

Problem reformulation

Un important subclass of such problems are genrated by a kernel K, which generates $\mathcal{H}_K$, a *Reproducing Kernel Hilbert Space* (RKHS).

- [Wahba90] showed that the solution to (2) in $\mathcal{H}_K$ has the form :

$$f(x) = \sum_{n=1}^{N} \omega_n K(\tau_n, x) \tag{3}$$

- But... N may be too large! Trick : Expressing useful info using a limited amount of “representers” $C = (c_m)_{m=1\ldots M}$ ($M \ll N$).

Problem reformulation

Un important subclass of such problems are genrated by a kernel K, which generates $\mathcal{H}_K$, a *Reproducing Kernel Hilbert Space* (RKHS).

- [Wahba90] showed that the solution to (2) in $\mathcal{H}_K$ has the form :

$$f(x) = \sum_{n=1}^{N} \omega_n K(\tau_n, x) \tag{3}$$

- But... N may be too large! Trick : Expressing useful info using a limited amount of “representers” $C = (c_m)_{m=1\ldots M}$ $(M \ll N)$.
- We then search $f \in \mathcal{H}_K$ in the form :

$$f(x) = \sum_{m=1}^{M} \omega_m K(c_m, x) \tag{4}$$

Solve (2) in the RKHSubspace generated by basis functions $K(c_m, .)$

Efficient sequence discriminant model

In practice, the prior probabilities $P(sp_A) \ll P(impostor)$
So we weight the criterion to minimize in (2), which becomes :

$$\min_{\omega_1,\ldots,\omega_M} \left[\frac{1}{T_A} \sum_{t=1}^{T_A} \left(1 - \sum_{m=1}^{M} \omega_m K(c_m, a_t) \right)^2 + \frac{1}{T_U} \sum_{t=1}^{T_U} \left(\sum_{m=1}^{M} \omega_m K(c_m, u_t) \right)^2 \right]$$

The solution is a sequence discriminant model :

$$\hat{\omega}_A = 1/2 \ \mathbf{R}_\tau{}^{-1} \overline{\varphi_c}(A)$$

$$\text{where} \begin{cases} \varphi_c(x) = \left[K(x, c_1), \ldots, K(x, c_M) \right]^\top & \overline{\varphi_c}(A) = \frac{1}{T_A} \sum_{t=1}^{T_A} \varphi_c(a_t) \\ \mathbf{K}_c = \left(K\left(c_m, c_n\right) \right)_{(n,m) \in \{1 \ldots M\}^2} & \mathbf{R_c} \approx \frac{1}{M} \mathbf{K}_c^\top \mathbf{K}_c \end{cases}$$

Estimating similarity between two sequences

Estimating the correlation matrix R only on representers gives The discriminant function for a sequence A :

$$f_A(x) = \frac{M}{2}\langle \mathbf{K}_c{}^{-2}\overline{\varphi_C}(A), \varphi_c(x)\rangle \tag{5}$$

Extending this *measure of similarity* to a sequence $B = (b_t)_{t=1\ldots T_B}$ can be done by computing the average, which gives our kernel expression :

$$\boxed{\kappa_{RKHS}(A, B) = \langle\overline{\Phi_C}(A), \overline{\Phi_C}(B)\rangle} \tag{6}$$

where we define the *sequence mapping* :

$$\overline{\Phi}_C(x_1, \ldots x_T) = \mathbf{K}_c{}^{-1}\overline{\varphi_C}(x_1, \ldots, x_T) \tag{7}$$

Advantages of the new sequence kernel / mapping

- Freedom on choice of feature space geometry via function K.
  No limitation because of dimension problems.
- Control on representation complexity via representers C.
  If more flexiblity is desired in a particular region of the input space, that region needs to be represented by more sample vectors.
- Efficiency : much faster than UBM-GMM approach (fast scoring).
- Easy to implement. Only need to store :
  - M d-dimensional Representers (from a simple VQ)
  - $M \times M$ matrix $\mathbf{K}_c$ (which requires to map only few representers)
  - Some M-dimensional impostor mappings $\left(\overline{\Phi}_C(X_1), ..., \overline{\Phi}_C(X_{N_{imp}})\right)$
  - $2 \times M$ normalization parameters (μ, σ)

On the notion of sequence

In the case of speaker verification,

$$\text{sequence} = \left\{ \begin{array}{l} \text{a set of vector produced by a same speaker} \\ \text{under the same conditions} \\ \text{(handset, channel, language,...)} \end{array} \right\} \quad (8)$$

- Experiments show that even when only one training sequence is available per target speaker, there is no point in splitting this sequence.

On the notion of sequence

In the case of speaker verification,

$$\text{sequence} = \left\{ \begin{array}{l} \text{a set of vector produced by a same speaker} \\ \text{under the same conditions} \\ \text{(handset, channel, language,...)} \end{array} \right\} \tag{8}$$

- Experiments show that even when only one training sequence is available per target speaker, there is no point in splitting this sequence.
- On the opposite, if several sequences from different recordings are available, they should not be combined
  (to preserve information about recording session variability).

Implementation of the sequence mapping

Our mapping is entirely defined by the choice of K and C.

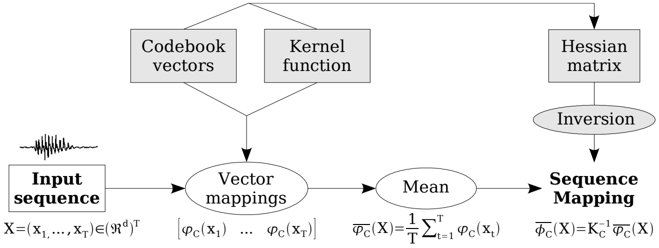

Codebook vectors
Kernel function
Hessian matrix
Inversion
Input sequence
Vector mappings
Mean
Sequence Mapping
$X=(x_1,\dots,x_T)\in(\Re^d)^T$
$[\varphi_C(x_1) \ \dots \ \varphi_C(x_T)]$
$\overline{\varphi_C}(X)=\frac{1}{T}\sum_{t=1}^{T}\varphi_C(x_t)$
$\overline{\Phi_C}(X)=K_C^{-1}\overline{\varphi_C}(X)$

Figure: Implementation of the Sequence Mapping $\overline{\Phi_C}$. Gray boxes indicate preliminary settings for the system.

SVM model learning

- Once the mapping is defined, impostor mappings can be computed in advance

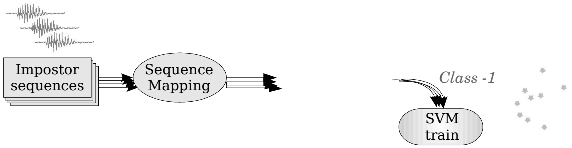


Figure: Block diagram of the SVM training scheme.

SVM model learning

- Once the mapping is defined, impostor mappings can be computed in advance
- Then, one SVM model per target speaker can be trained.

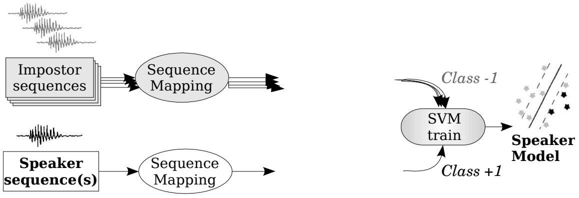


Figure: Block diagram of the SVM training scheme.

SVM model learning

- Once the mapping is defined, impostor mappings can be computed in advance
- Then, one SVM model per target speaker can be trained.
- Note that all mapping component are normalized $\overline{\Phi_C}(X) \mapsto \frac{\overline{\Phi_C}(X)-\mu}{\sigma}$

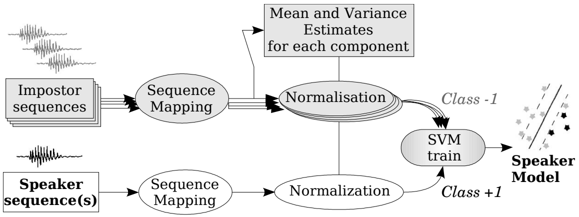


Figure: Block diagram of the SVM training scheme.

SVM testing

For utterance testing, given the linearity of the dot product, the SVM score for a sequence Y has the form (as with the GLDS kernel):

$$score(Y) = \langle \overline{\Phi_C}(Y), \sum_i \alpha_i y_i \overline{\Phi_C}(S_i) \rangle = \langle \overline{\Phi_C}(Y), \Omega_{sp} \rangle \qquad (9)$$

where (α_i) are positive weights, (S_i) are support sequences, $y_i = \pm 1$ and Ω_{sp} is a single vector where all the support sequences are collapsed.

- $\oplus$ Memory saving for models storing
- $\oplus$ Time saving during test.

Front-end Processing

In chronological order :

- 16-ms Hamming windowing, 10-ms step.
  24 channels filter-bank in [340;3400Hz]
  $\Rightarrow$12 MFCC + 12 ΔMFCC + $\Delta\log(\mathrm{E})$
- Speech activity detection : hysteresis threshold on frame energies.
  (bugs in the first version)
- Feature warping on 3 sec sliding windows.
  Such a normalization reduce effects of additive noise and channel convolutive effects.

Corpora

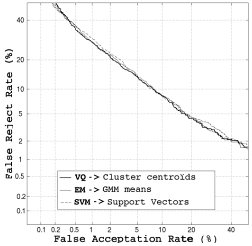
- *Background corpus :*
  NIST 2001 and 2003 SRE database
  $\longrightarrow$ choice of the representers (only on 2001)
  $\longrightarrow$ feed SVM with impostor sequences

- *Development corpus :*
  NIST 2004 SRE database
  $\longrightarrow$ parameters tuning
  $\longrightarrow$ score threshold for min DCF

Choice of the representers

Previous experiments (generating random vectors with different distribution) showed that representers have to be representative of real-word data. We estimate them from a background corpus.

- We achieved good performances with a Vectorial Quantization (VQ).
- No difference when using an EM algorithm.
- We tried a SVM strategy to extract representers from speaker discriminant regions, but performance remains roughly the same.



Number M of representers

Extracting 1024 representers achieves good performance, while after 2048, results roughly do not chang (good news).

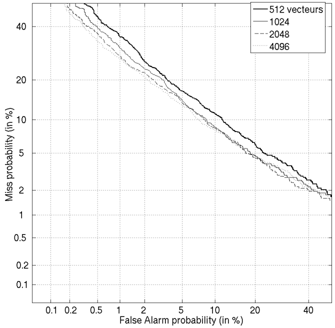


Figure: Comparison of different size of codebook after a VQ of the background corpus

Kernel function K

We experimented radial basis and polynomial kernels :

$$\begin{aligned} K_{rbf}(x,y) &= e^{-\gamma\langle x-y,x-y\rangle} \\ K_{poly}(x,y) &= (1+\langle x,y\rangle)^k \end{aligned} \tag{10}$$

Polynomials give the best performance, but RBF also perform well.
Pptimal degree for K_{poly} seems to be $k = 7$.

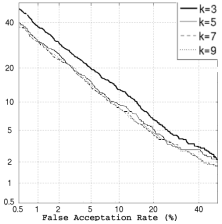
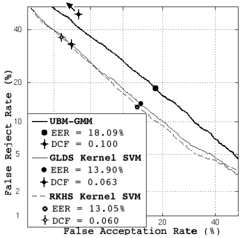


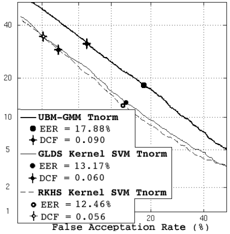
Figure: DET plot showing different polynomial degree k.

Results on NIST 2004 SRE

Using exactly the same data for all systems...



(a) without score Tnorm



(b) with score Tnorm

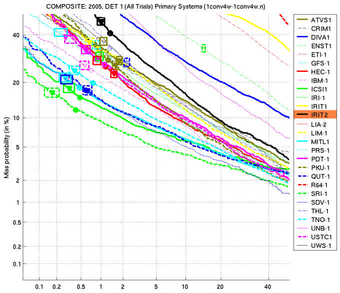
Figure: DET plot showing performance on NIST'04 SRE

NB : as many problems were encountered in this year evaluation, no time for performing T-norm.

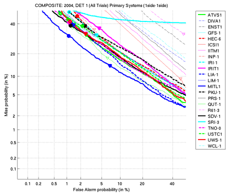
Results on NIST 2005 SRE

- A first version with several bugs : disaster.
- Debugged version finally performs as well as expected.
- But not very competitive as other participants have also made progress

Perhaps because they used last year data more intelligently (I did not use nist SRE 2004 data for any model learning neither any normalization).



NIST SRE 2005 Results



Past Year Results

Conclusion

Other participants took advantage of the similarity between this year and last year evaluation corpus. I did not really, as i used 2004 data only to confirm my kernel choice and choose the decision thershold.

I should evaluate my system with the following improvement :

- Training 2005 models using 2004 background data.
- Performing T-norm.

For the future, next research will concern similar kernel methods exploiting high-level feature which are known to be useful (and robust) clues for speaker recognition.