Conceiving a new sequence kernel and applying it to SVM speaker verification

Jérôme Louradour, Khalid Daoudi

IRIT, CNRS UMR 5505 - 118, rte de Narbonne, 31062 Toulouse, FRANCE
phone: (+33/0)5 61 55 72 01 - fax: (+33/0)5 61 55 62 58
email: {louradou,daoudi}@irit.fr - web: www.irit.fr/recherches/SAMOVA/

Abstract

In this paper, we propose a new approach for sequence classification which is based on the framework of Reproducing Kernel Hilbert Spaces (RKHS). We introduce the theoretical material which leads to the formulation of an original sequence kernel, that we implement in a SVM scheme. Experiments are carried out on a speaker verification task using NIST SRE data. They show that our new sequence kernel significantly outperforms the generative approach with Gaussian Mixture Models (GMM). They also show that it generally outperforms the powerful Generalized Linear Discriminant Sequence (GLDS) kernel, while offering more efficiency and flexibility (than GLDS).

1. Introduction

In order to apply SVM to a speaker verification task, the simplest approach may be to learn vector-level frontiers and to score on a sequence using the mean of SVM’s vectors outputs, as it was done in [1]. But in practice, such a frame-level discriminative approach achieves poor performances because there is an important overlap between different classes in the input space. In addition, the implementation may be awkward as it is common to collect a large amount of background data to feed the SVM with impostor instances. The training algorithm becomes thus intractable unless a data clustering is used, and the number of support vectors to keep in memory for testing increases with the number of training utterances.

Moreover, in speaker verification, the goal is to minimize classification errors on sequences, not on speech frames. Thus, a sequence-based learning approach seems more appropriate. Several recent studies dealt with the conception of sequence kernels [2, 3], the challenge being to overcome the length variability which leads to the formulation of an original sequence kernel, named Generalized Linear Discriminant Sequence (GLDS) Kernel, is derived as a simple dot product in this feature space. This approach seems to be very powerful as it significantly outperforms the GMM classifier (in all experiments we carried out).

In this paper we use the same philosophy as in the GLDS approach, but our mapping is defined by evaluating kernel functions using some vectors which are representative of the training data. Actually, these “representers” generate a Reproducing Kernel Hilbert Subspace (RKHS) in which classification is performed. By doing so, we derive a new sequence kernel which is more efficient and more flexible than the GLDS kernel, while it leads to similar or better performance (than GLDS) in speaker verification.

2. Conception of the new Sequence Kernel

2.1. Training on a sequence in a RKHS

2.1.1. Problem formulation

Consider we are given a training corpus \((\tau_n, s_n)_{n=1}^{N}\) with real vectors \(\tau_n \in \mathbb{R}^d\) and desired outputs \(s_n \in \{0, 1\}\). Given a target speaker \(sp_A\), \(s_n = 0\) for a fixed set universal of background data \(U = (\tau_i)_{i=1}^{T_u}\) and \(s_n = 1\) for all vectors \(A = (\tau_i)_{i=1}^{T_a}\) produced by \(sp_A\), \(U \cup A = (\tau_n)\). The process of finding a discriminant function \(f : \mathbb{R}^d \rightarrow \mathbb{R}\) on this corpus can be written as:

\[
\min_{f \in \mathcal{H}} \sum_{n=1}^{N} L(f(\tau_n), s_n) \quad (1)
\]

where \(L\) is a loss function, and \(\mathcal{H}\) is the space of functions in which the search for \(f\) is performed.

An important subclass of problems of the form (1) is generated by a positive definite kernel \(K\), and the corresponding space of functions \(\mathcal{H}_K\) is called Reproducing Kernel Hilbert Space (RKHS). If \(K\) has an (possibly infinite) eigen-expansion \(K(x,y) = \sum \gamma_n \phi_n(x) \phi_n(y)\) then each element of \(\mathcal{H}_K\) has an expansion \(f(x) = \sum \omega_n \phi_n(x)\). In [7], it is shown that the solution to (1) has a finite-dimensional form:

\[
f(x) = \sum_{n=1}^{N} \omega_n K(\tau_n, x) \quad (2)
\]

In many real-world applications (such as speaker recognition), the classifier (2) may be intractable because of the huge amount of training vectors \(\tau_n\). To make it tractable, our strategy is to express (useful) information contained in the training data using a limited amount of representative vectors that we call representers.
2.1.2. Efficient sequence discriminant model

Assume a set of vectors $C = (c_m)_{m=1...M}$ ($M \ll N$) was determined to be “representative” of the training data, with some clustering method. We then search $f \in \mathcal{H}_K$ in the form:

$$f(x) = \sum_{m=1}^{M} \omega_m K(c_m, x)$$ (3)

This amounts to solve the problem (1) in the RKH Subspace generated by basis functions $K(c_m,.)$. Such functions are known as the representatives of evaluation at $c_m$ in $\mathcal{H}_K$, because $\langle K(c_m), K(x,.) \rangle_{\mathcal{H}_K} = K(c_m, x)$.

In practice, the prior probabilities of the training set and impostor training set are usually imbalanced (because much more impostor data is available). In order to compensate for this, we weight the criterion to minimize in (1), which becomes under a squared-error loss function:

$$\min_{\omega_1,...,\omega_M} \left[ \frac{1}{T_A} \sum_{t=1}^{T_A} \left( 1 - \sum_{m=1}^{M} \omega_m K(c_m, a_t) \right)^2 \right] + \frac{1}{T_I} \sum_{t=1}^{T_I} \left( \sum_{m=1}^{M} \omega_m K(c_m, a_t) \right)^2$$ (4)

This problem can be written in the matrix form:

$$\min_{\omega} (D s - D K_{r \times C} \omega)^T (D s - D K_{r \times C} \omega)$$ (5)

where $\omega = [\omega_1, ..., \omega_M]^T$, $D = \frac{1}{N} \text{diag}(\sqrt{\sum_{t=1}^{T_A} s_1^2}, ..., \sqrt{\sum_{t=1}^{T_A} s_T^2}, ..., \sqrt{\sum_{t=1}^{T_I} s_{T_I}^2})$, and $K_{r \times C} = \left( K(c_m, r_t) \right)_{(n,m) \in \{1,...,N\} \times \{1,...,M\}}$ (6)

Solving this problem using the method of normal equations yields a sequence discriminant model:

$$\hat{\omega}_A = (K_{r \times C}^T D^2 K_{r \times C})^{-1} (D K_{r \times C})^T D s$$

$$= 1/2 R_c^{-1} \overline{\varphi_c}(A)$$

where

$$\overline{\varphi_c}(A) = \frac{1}{T_A} \sum_{t=1}^{T_A} \varphi_c(a_t)$$

$$R_c = K_{r \times C}^T D^2 K_{r \times C}$$

Noting that $K_{r \times C} = [\varphi_c(\tau_1), ..., \varphi_c(\tau_K)]^T$, the matrix $R_c$ is thus the empirical expectation $E \left[ \varphi_c(x) \varphi_c(x)^T \right]$. Modulo the means, it is similar to a covariance matrix for the vector mappings $\varphi_c$, estimated on the set $(\tau_i)$, giving the same importance to the subsets $(a_n)$ and $(u_n)$. This matrix can be approximated by being estimated only on the representers $C$:

$$R_c \approx K_{c}^T I_M K_{c} = \frac{1}{M} K_{c}$$ (7)

with $K_{c} = \left( K(c_m, c_n) \right)_{(m,n) \in \{1,...,M\}^2}$. This simplification will permit us to conceive a sequence kernel which is independent of the target speaker. Such property is welcome for efficiency and also system stability.

The discriminant function for a sequence $A$ is finally:

$$f_A(x) = \frac{M}{2} \langle K_{c} \overline{\varphi_c}(A), \varphi_c(x) \rangle$$ (8)

Each component of this model, which can be seen as a basis function, is indexed by a prototype $c_m$. If more flexibility is desired in a particular region, then that region needs to be represented by more basis functions of the form $K(c_m,.)$. By this way, we can control the complexity of the representation.

2.2. Estimating similarity between two sequences

Suppose a function $f_A$ was learned from a sequence $A$ using (8). Then, extending this measure of similarity to a sequence $B = (b_t)_{t=1,...,T_B}$ can be done by computing the average:

$$\text{sim}_A(B) = \frac{1}{T_B} \sum_{t=1}^{T_B} f_A(b_t)$$

$$= \frac{M}{T_B} \left( \langle K_{c} \overline{\varphi_c}(B), K_{c} \overline{\varphi_c}(A) \rangle \right)$$

Skipping the scaling factor $M/2$, this leads to our new symmetric sequence kernel:

$$\kappa_{\text{RKH}S}(A, B) = \langle \overline{\varphi_c}(A), \overline{\varphi_c}(B) \rangle$$ (9)

where we define the sequence mapping:

$$\overline{\varphi_c}(x_1, ..., x_T) = K_{c} \overline{\varphi_c}(x_1, ..., x_T)$$ (10)

Now we stretch the advantages of our sequence kernel as compared to the GLDS kernel. First, in the case of GLDS kernel, if $d$ is the dimension of the input space, and $k$ the maximal polynomial order, the dimension of the mapping is $M = (d+1)^k$. In our application, $d = 25$, and $M$ becomes too large when $k > 3$. Therefore, the approach is limited to polynomial expansion with orders equal or lower than three, which restricts the choice of similarity measure, and consequently the geometrical properties of the classification system. In our case, we have more freedom on choice of the feature space geometry via the function $K$ and the representers $C$.

Second, in case of GLDS kernel, the correlation matrix computation requires an expensive mapping of all background population (a diagonal approximation must be used when $k \geq 3$). In our case, the computation of $K_{c}$ requires to map only (relatively) few representers of the training data.

3. SVM architecture

In this section, we describe how to use our sequence kernel in a SVM scheme for a speaker verification task. We define a sequence as a set of vector produced by a same speaker under the same external conditions (microphone type, mood...). Our experiments have shown that even when there is only one training sequence 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.

Once $K$ and $C$ are chosen, the impostor sequences mappings can be computed from a background corpus. Then, one SVM model per target speaker can be trained (e.g. using [8]).

A common precaution for SVM, which are not invariant to linear transformations, is to standardize each component of the mapping with the normalization:

$$\overline{\varphi_c}(X) \rightarrow \overline{\varphi_c}(X) - \frac{\mu}{\sigma}$$ (11)

where $\mu$ and $\sigma$ are the mean and standard deviation estimates on background mappings $\overline{\varphi_c}(X)$. 
For utterance testing, given the linearity of the dot product, the SVM score for a sequence \( Y \) has the form:

\[
score(Y) = \langle \hat{f}_C(Y), \sum_i \alpha_i \hat{f}_C(S_i) \rangle = \langle \hat{f}_C(Y), \Omega_{sp} \rangle
\]

(12)

where \( \alpha_i \) are positive weights, \( y_i = \pm 1 \), \( (S_i) \) are support sequences and \( \Omega_{sp} \) is a single vector where all support sequences are collapsed. This allows memory saving for models storing and time saving during test.

Finally, the binary decision is taken by comparing the sequence score (12) to a threshold fixed on development data.

4. Experimental setup

To extract acoustic vectors from a speech sequence, 12 MFCC and their first order time derivatives are extracted on 16ms window, at a 10ms frame rate. The derivative of the energy logarithm is also added. Then, a speech activity detector discards silence frames. Finally, the 25-dimensional input vectors are warped over 3 sec windows [9].

For the background corpus, that is used to determine the representers and to feed the SVM with impostor sequences, we used the NIST 2001 SRE database.

In Sec.5 and Sec.6, we use NIST 2003 and 2004 SRE database [10] as development and test data. These databases involve telephone conversations with microphone type mismatch.

5. Technical choices for our new kernel

5.1. Choice of representers

If we strictly follow the theory in 2.1.2, the representers \( C \) should be extracted from both background and target speaker data. With such a strategy, the verification for a given speaker requires a specific mapping, that makes the system awkward. Besides, representers should ideally represent a “universal” speaker. Thus we can assume that ignoring the “few” target speaker sequences would have little effect on the discrimination capability. We follow this heuristic, i.e. we use only background data to compute the representers. By doing so, our sequence kernel is independent of the target speaker, which makes the system highly efficient and stable.

We achieved good performances when representers are estimated using a classical Vectorial Quantization (VQ). In this case, we use the background corpus without exploiting information about sequencing (all in a same bag). No difference in performance was observed when using an EM algorithm and taking GMM means as representers.

We also tried another approach that would take into account the notion of sequencing. It aims at retaining vectors that lie in speaker discriminative regions. To do so, a frame-level SVM frontier is learned on each background sequence \((x_i)_{i=1,..,T}\), using the kernel \( K \). Universal background vectors \( \bar{x}_n \) serve as impostor samples. Then, we obtain as many vector-level models as development sequences, with the form:

\[
o(x) = \sum_{i=1}^T \alpha_i K(x, x_i) - \sum_n \bar{\alpha}_n K(x, \bar{x}_n) + b
\]

(13)

Finally we retain as representers, from the gathering of these models, support vectors corresponding to the \( M \) highest Lagrange multipliers \( \alpha_i > 0 \). As a matter of fact, these coefficients reflect the importance of training vectors. This approach, though totally different, gives roughly the same classification performance than when choosing representers with a VQ.

Fig.1a shows a comparison between VQ centroids, GMM means and high weighted support vectors, for \( M = 2048 \) and a 7-degree polynomial kernel \( K \). Performances are similar.

5.2. Choice of \( M \) and \( K \)

Without a component normalization (11) for SVM inputs, we observed a significant instability of the system when varying \( M \). But with the normalization, we observed that increasing the number \( M \) of representers improves performance.

The same behavior occurs when varying the parameters of the kernel \( K \). We experimented radial basis and polynomial kernels, defined respectively by:

\[
K_{rbf}(x, y) = e^{-\gamma(x-y,x-y)}
\]

\[
K_{poly}(x, y) = (1 + (x, y))^k
\]

Polynomials give the best performance, but RBF also perform well. However, without normalization, instability is observed when varying the degree \( k \) of \( K_{poly} \) (Fig.2).

6. Comparison with GMM and GLDS

6.1. Reference systems

In this section, we compare our system with two well-performing state-of-the-art systems.

The first one, that we presented at NIST 2004 SRE, is an UBM-GMM system [11]. In this system, two gender-dependent background GMM are estimated and each target speaker GMM
is derived from the appropriate background model, by adapting mean vectors with a MAP criterion. During the decision phase, a sequence score is computed as the mean log-likelihood ratio.

The second one uses the GLDS kernel described in [6]. The maximal polynomial order is set to 3, therefore the size of the mapping is \( (25+3)! \approx 3276 \).

In our system, we use \( M = 2048 \) representers (extracted from a VQ of background data) and a 7-degree polynomial kernel \( (K_{poly} \text{ with } k = 7) \). For fair comparison, exactly the same development and test data where used for all systems.

6.2. Results

Results on NIST 2003 and 2004 SRE common tasks are shown on Fig.3 and Fig.4. The performance degradation is due to the quite higher mismatch between train and test data in the case of 2004. The Equal Error Rate (EER) occurs when the decision threshold is set so that the false rejection (FR) rate is approximately equal to the false acceptance (FA) rate. The Detection Cost Function (DCF) was defined by NIST as a weighted sum of these two rates: \( DCF = (0.1 \times P_{FR}) + (0.9 \times P_{FA}) \). This goodness criterion, to be minimized, can be measured once the decision threshold was determined on development data. Then binary decisions can be compared with desired answers.

These results show that our system (as well as the GLDS one) significantly outperforms the UBM-GMM classifier, even after T-normalization of scores. Moreover, it generally outperforms the GLDS system, while using lower-dimensional feature space (\( M = 2048 \) instead of 3276). This indicates that our approach (which is similar in spirit to the GLDS one) is very promising and deserves further investigation.

7. Conclusion

A new sequence kernel was introduced and applied to a speaker verification task. All the experiments we carried out show that the new system significantly outperforms the classical UBM-GMM classifier. Moreover, it outperforms the powerful GDLS method while using lower-dimensional feature space and offering more flexibility.

The new designed kernel also offers good perspectives for further improvements (for instance, how optimally choose representers), and can be used in other classification tasks involving variable-length sequences.

8. References