A Smoothing Kernel for Spatially Related Features and Its Application to Speaker Verification

Luciana Ferrer[1,2]*, Kemal Sönmez*[2]*, Elizabeth Shriberg*[2]

[1]Department of Electrical Engineering, Stanford University, Stanford, CA, USA
[2]Speech Technology and Research Laboratory, SRI International, Menlo Park, CA, USA
lferrer@stanford.edu, kemal,ees@speech.sri.com

10.21437/Interspeech.2007-137

Abstract

Most commonly used kernels are invariant to permutations of the feature vector components. This characteristic may make machine learning methods that use such kernels suboptimal in cases where the feature vector has an underlying structure. In this paper we will consider one such case, where the features are spatially related. We show a way to modify the objective function of the support vector machine (SVM) optimization problem to account for this structure. The new optimization problem can be implemented as a standard SVM using a particular smoothing kernel. Results are shown on a speaker verification task using prosodic features that are transformed using a particular implementation of the Fisher score. The proposed method leads to improvements of as much as 15% in equal error rate (EER).
Index Terms: Support Vector Machines, Kernels, Smoothing, Speaker Recognition, Speaker Verification.

1. Introduction

Structural Risk Minimization (SRM) has proven to be a powerful framework for controlling model complexity while building powerful discriminative detectors in high dimensions. The most popular application of SRM, the SVMs, produce superior performance in many problems. SVMs are able to handle large feature vectors, and result in very flexible architectures due to the use of kernel functions [1]. The most widely used kernels, the linear, polynomial, and radial basis functions, are all invariant to permutations of the components of the input vectors. This is perfectly suitable for many problems in which the particular ordering of the components of the input vector is not significant. However, there exist problems in which the feature vectors have an underlying structure. The knowledge of this structure can potentially be of help if it is exploited in an effective way.

In this paper, we derive a new kernel targeted for spatially related features. We consider feature vectors whose components are measurements taken over regions of an underlying space $\mathcal{F}$. The kernel is derived by imposing a new regularization term in the SVM objective function, related to the smoothness of the SVM weight vector, where smoothness is measured using neighborhood relationships in the extraction space $\mathcal{F}$. Common examples of spatially structured feature vectors are those corresponding to histograms as used, for example, in image recognition. Components of a histogram feature vector are related to each other by the distances between the corresponding bins. Even though the kernel is derived using the SVM formulation, it can potentially be used in any kernel method.

As a particular case in which this method can be applied, we consider a classification task where each data sample is a sequence of varying length. These variable-length sequences are transformed into fixed-length vectors via a particular implementation of the Fisher score [2] in which each component of the transformed vector is related to a certain Gaussian in a Gaussian mixture model (GMM) [3, 4]. This transform can be understood as a *soft* histogram where the bins are replaced by Gaussians and the frequencies of the bins are replaced by posteriors. As in the case of histogram features, the resulting vector has a spatial structure and the smoothing kernel can be applied.

We present results on a speaker verification task using syllable-level prosodic features as input. Most features used in speaker recognition are sequential in nature, and a significant amount of work has been done in applying and developing kernels for this task (e.g., [5, 6]). We show that the proposed smoothing procedure gives as much as 15% EER improvement on this task, resulting in our currently best performing classifier for these features.

2. Support Vector Machines

Consider a training set with m samples, $S = \{(x_i, y_i) \in \mathcal{R}^d \times \{-1, +1\}; i = 1, ..., m\}$, where x_i are the features and y_i the class corresponding to sample i. Our goal is to find a function $f(x) = w^T x + b$, such that $\text{sign}(f(x))$ is the predicted class for feature vector x. The standard support vector machine (SVM) formulation for classification is given by (see, e.g., [1]):

$$\begin{array}{lll} \text{minimize} & J(w,\epsilon) = \frac{1}{2} w^T w + C \sum_{i=1}^{m} \epsilon_i & \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 - \epsilon_i & i = 0, ..., m \\ & \epsilon_i \geq 0 & i = 0, ..., m \end{array} \quad (1)$$

Minimizing the norm of the weight vector is equivalent to maximizing the margin between the samples and the hyperplane. The *slack* variables ϵ_i allow for some samples to be at a distance smaller than the margin to the separating hyperplane or even on the wrong side. The parameter C controls the trade-off between the size of the margin and the total amount of error. By deriving the dual form of the optimization problem above we find that input vectors appear only as inner products with each other. Hence, each inner product between input features can be replaced with a function $K(x_i, x_j) = \phi(x_i)^t \phi(x_j)$ called the kernel function, where $\phi(x)$ is a transform of the input features.

The SVM problem can also be expressed as a maximization of $l(w, b) = -\frac{1}{2} w^T w - C \sum_{i=1}^{m} h(y_i(w^T x_i + b))$, where $h(z)$ is the *hinge loss*, given by $(1-z)u(1-z)$ with $u(z)$ the Heaviside step function. This can be interpreted as maximizing a log-posterior probability for the parameters w and b given the data. The first term of l corresponds to a Gaussian prior $\mathcal{N}(0, I)$ on the weight vector w (*i.e.*, weights are assumed a priori to be independent). A flat prior is assumed on the bias term b. The

second term corresponds to the log-likelihood of the data for an appropriately defined likelihood function [7]. Hence, we can interpret the SVM optimization as maximum a posteriori (MAP) estimation of the parameters w and b given the training data.

The above setup corresponds to a classification problem. The regression problem can also be posed as a convex optimization problem by choosing an appropriate distance measure [1, 8] with the objective function given by the sum of the square norm of the weight vector and an error term, as in the classification case. The dual of this problem again takes a form in which features appear only in inner products with other features. Furthermore, the interpretation of the SVM as a MAP estimation still holds given an appropriate choice of likelihood function [7]. Hence, even though the development in Section 3 will be done considering a classification problem for simplicity, the method described and the interpretations given can be equally applied to SVM regression problems.

3. Smoothing Kernel

Assume now that the feature vectors $x \in \mathcal{R}^d$ have some kind of spatial structure, i.e., that each component k in this vectors is somehow related to measurements taken on or around a certain point m_k in an underlying feature space $\mathcal{F}$, which should be a metric space. As mentioned earlier, each component of x could, for example, correspond to the frequency of a certain bin in a histogram. In this case, the m_k's could be the centers of the bins. We will classify the feature vectors x_i with an SVM, which means that our output will be given by $f(x_i) = w^T x_i + b = \sum_k w_k x_i^k + b$. It is natural to think that the SVM weights that multiply components of x_i coming from nearby regions in $\mathcal{F}$ should not vary widely from each other. That is, the importance of the features, as measured by the magnitude of the weight applied to them, cannot differ too much for nearby regions.

3.1. Reformulating the SVM problem

We wish to modify the objective function in (1) by adding a regularization term $\lambda\delta(w)$, where $\lambda \geq 0$ is a tunable parameter and δ is some function of the weight vector that takes small values when the vector is *smooth* (measuring smoothness in the $\mathcal{F}$ domain) and large values when it is not. In all the following we will assume that we can express $\delta(w)$ as a quadratic form $w^T A w$. The new objective function then is $J(w, \epsilon) = \frac{1}{2} w^T w + \frac{1}{2}\lambda w^T A w + C\sum_i \epsilon_i$. As long as we choose A to be positive semidefinite, $J(w, \epsilon)$ will be convex. Furthermore, since $\delta(w)$ is a quadratic form, we can assume A to be symmetric without loss of generality. In the next section we will see one natural way of defining a positive semidefinite matrix A that achieves the desired goal.

We can rewrite J as $\frac{1}{2} w^T (I + \lambda A) w + C\sum_i \epsilon_i$. Now, by the change of variable $\tilde{w} = Bw$, with $B^T B = I + \lambda A$ (B is a matrix square root of $I + \lambda A$ and can be chosen to be real and symmetric since $I + \lambda A$ is real, positive definite and symmetric), we can write the new optimization problem as

$$\begin{array}{lll} \text{minimize} & J(\tilde{w}, \epsilon) = \dfrac{1}{2}\tilde{w}^T \tilde{w} + C\displaystyle\sum_{i=1}^{m} \epsilon_i & \\ \text{subject to} & y_i(\tilde{w}^T \tilde{x}_i + b) \geq 1 - \epsilon_i & i = 0, ..., m \\ & \epsilon_i \geq 0 & i = 0, ..., m \end{array} \tag{2}$$

where $\tilde{x}_i = B^{-T} x_i$ (B is invertible since $B^T B = I + \lambda A$ is positive definite). Comparing this with (1) we see that we have obtained a new SVM problem where the features are now the $\tilde{x}_i$s. We can then implement this as an SVM problem on the original features using a kernel given by

$$K(x_i, x_j) = x_i^T B^{-1} B^{-T} x_j = x_i^T (I + \lambda A)^{-1} x_j. \tag{3}$$

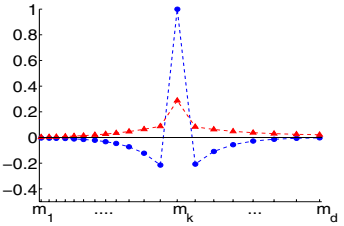


Figure 1: Row k of the matrices M (round markers) and B^{-T} (triangular markers), plotted against the m_l.

We can interpret the optimization problem (2) probabilistically as we did for the standard SVM problem in Section 2. The problem (2) can be formulated as maximizing $-\frac{1}{2} w^T (I + \lambda A) w - C\sum_{i=1}^{m} l(y_i(w^T x_i + b))$. This means that the prior distribution on the weights is now $\mathcal{N}(0, (I + \lambda A)^{-1})$, as opposed to $\mathcal{N}(0, I)$. Therefore, we are now imposing a correlation structure between the weights instead of assuming their independence.

3.2. The A matrix

Our aim is to design a matrix A such that $w^T A w$ is small for smooth weight vectors and large otherwise. We will measure the smoothness in the original feature space $\mathcal{F}$ by considering the distance between the m_k's to which each of the components of the vector x are related. We define the matrix A to be $M^T M$ (which results in A being positive semidefinite for any choice of M) where

$$M_{k,l} = \begin{cases} 1 & \text{if } k = l \\ -\gamma_k e^{\alpha_k d(m_l, m_k)} & \text{otherwise} \end{cases}$$

where d is a distance defined in $\mathcal{F}$; α_k is determined such that if $l(k, s)$ is the index of the sth closest point to m_k then $M_{k,l(k,1)} = 100 M_{k,l(k,n)}$; and γ_k is determined so that $\sum_{l \neq k} M_{k,l} = -1$. This way, Mw resembles a discrete derivative of w since its kth component is given by $(Mw)_k = w_k - \sum_{l \neq k} a_l w_l$, with $a_l = -M_{k,l} \geq 0$, for $l \neq k$, and $\sum_{l \neq k} a_l = 1$. The value of n determines how many w's are considered in the derivative. We choose n to be such that the sum of the mixture weights for the closest n points to m_k is larger than p, where p is a tunable parameter. This way, the derivative is computed over more points if we are in a low density region and over fewer points if we are in a high density region.

Figure 1 shows a simple example where $\mathcal{F} = \mathcal{R}$ and d is the Euclidean distance. The figure shows row k of the matrices M and B^{-T} as a function of the m_l. Interestingly, the rows of B^{-T} (which is the matrix that multiplies our input features x) resemble the impulse response of a smoothing filter. Asking for smoothness on the SVM weights results in a smoothing transformation on the input features. Nevertheless, this is a particular smoothing transform that optimizes the trade-off between classification error and regularization. In fact, all other smoothing procedures we have tried on the features have failed to give any improvements.

4. Application to Sequential Data

We will consider the following setup as an example of a case in which the feature vectors present an underlying spatial structure. Consider a classification problem in which the samples are

variable-length sequences (as is the case in most speech problems). These sequences are realizations of an underlying random process whose characteristics depend on the class of the sample. Specifically, each of our samples (f_i, y_i) consists of a sequence of feature vectors $f_i = \{f_i^t \in \mathcal{R}^p; t = 1, \ldots, N_i\}$ which corresponds to a single realization of the underlying random process corresponding to the class of the sample, y_i. We wish to classify these samples using support vector machines. To do this we need to define a transformation (which in turn defines a kernel function) of the input features that turns the variable-length sequences into fixed-length vectors.

We further assume that the features $\{f_i^j\}$ for sample i are generated independently for each j with the same distribution. The empirical distribution for each sample will be parameterized using the vector quantization (VQ) method described in [3]. Briefly, the feature space $\mathcal{F} = \mathcal{R}^p$ is divided into clusters based on some held-out data. A Gaussian model is computed for each cluster k by calculating the mean μ_k and the variance Σ_k of the data assigned to the cluster. A GMM is then formed by those Gaussians, with mixture weights c_k given by the proportion of samples assigned to each cluster. The transform is finally composed by the posterior probabilities for each of the Gaussians given the sample's data. Explicitly, the kth component of the transform corresponding to the ith sample is given by

$$x_i^k = \frac{1}{N_i} \sum_{j=1}^{N_i} \Pr(G = k | f_i^j) = \frac{1}{N_i} \sum_{j=1}^{N_i} \frac{g_k(f_i^j) c_k}{\sum_l g_l(f_i^j) c_l} \quad (4)$$

where G is the random variable corresponding to the Gaussian index and g_k is the Gaussian distribution corresponding to index k. These features can be considered a *soft* version of a multidimensional histogram, where each cell is replaced by a Gaussian and the frequencies of the cells are replaced by the posteriors of the Gaussians.

For these features, the most natural way to define the point m_k that represents the region over which the feature x^k is extracted is to take $m_k = \mu_k$, the mean of the Gaussian for which x^k is the posterior. Using these values and the Euclidean distance we can define the matrix A as in Section 3.2.

In [2], Jaakkola and Haussler introduced a kernel specially designed for sequential features, called the Fisher kernel. The kernel is defined as $K(f_i, f_j) = U_{f_i}^T I^{-1} U_{f_j}$, where $U_f = \nabla_\theta \log P(f|\theta)$ is the Fisher score, with θ being the parameters of some generative model for the features f and $I = E_f[U_f U_f^T]$ the Fisher information matrix. Since the Fisher score U_f has expectation equal to zero, the Fisher information matrix is simply its covariance matrix. Each element of U_f is the gradient of the log-likelihood with respect to a parameter and it describes how that parameter contributes to the process of generating a certain sample.

The method described above is one instance of the Fisher kernel where the generative model $P(f|\theta)$ is given by the VQ trained GMM. It can be shown that, in this setup, if we define θ_k such that $c_k = \theta_k / \sum_k \theta_k$, then the U_{f_i}'s are equal to the x_i's as defined in [4], up to a shift and a scale factor on each dimension. If we normalize the features x to have zero mean and unit variance, then our kernel $K(f_i, f_j) = x_i^T x_j$ is identical to the Fisher kernel if we assume I to be diagonal. Interestingly though, we have found that more sophisticated transformations, which aim to equalize the distribution of the components of x, work significantly better than simple subtraction of the mean and division by standard deviation (which only equalizes the first and second moments).

5. Experiments

Experiments were conducted on a text-independent speaker verification task. The goal is, given a speech sample and a claimed speaker identity, to decide whether the claim is true or false. This is a binary classification task for which SVMs were repeatedly found to outperform other classification methods. Here, we focus on a system that uses prosodic features extracted over automatically extracted syllable regions. The length of the sequence f for each sample is then the number of syllables in the output of an automatic speech recognizer. For each syllable, a set of 140 features is extracted, containing information about the pitch, energy, and duration characteristics of the syllable. We call these features Syllable NERFs (nonuniform extraction region features). For a description of the features see [9].

The prosodic features, f_i, for each sample are transformed to fixed length vectors (referred to as x_i) using the method described in Section 4. Since the dimension of the vectors is large (140) and the length of the sample sequences is usually small (also in the hundreds), we cannot train a single GMM for the complete feature vector. Instead the strategy described in [3] is used. Briefly, each prosodic feature is transformed separately. A GMM is obtained for each feature and for sequences of features for two and three consecutive syllables or pauses. We call these unigram, bigram and trigram models, respectively. This allows us to model temporal dependencies, which we would ignore otherwise. The transforms corresponding to each feature and each sequence are concatenated together into the final vector x_i. For each feature and each of its sequences, a matrix M is computed. The λ used for each case depends linearly on the size of the corresponding GMM. The matrix $I + \lambda A$ is then formed as a block diagonal matrix, with a block for each individual GMM. Parameters were chosen so that the total number of features for each N-gram length is approximately equal: 11,000 for unigrams, 13,000 for bigrams, 14,000 for trigrams, and 38,000 for the complete system.

Experiments were conducted using data from the NIST speaker recognition evaluation (SRE) from 2005 and 2006. The data from 2005 were used for parameter tuning. Each speaker verification trial consists of a test sample and a speaker model. The samples are one side of a telephone conversation with approximately 2.5 minutes of speech. We consider the 1- and 8-side training conditions in which we are given 1 or 8 conversation sides to train the speaker model. Each of these conversations corresponds to one positive example when training the SVM model for the speaker. The data used as negative examples for the SVM training are taken from 2003 and 2004 NIST evaluations along with some FISHER data, resulting in a total of 2122 conversation sides. The SRE2005 and SRE2006 tasks contain 25,887 and 24,004 trials for the 1-side training condition and 17,216 and 15,105 trials for the 8-side training condition, respectively. The average number of syllables per conversation side is around 600. Trials involving a test or train conversation side with fewer than 60 syllables were removed from the original list prepared by NIST, resulting in the number of trials mentioned above. The data used to obtain the GMMs for each sequence were drawn from data from the 2003 and 2004 evaluations along with some FISHER data, yielding a total of 2456 conversation sides (2 sides from each of 1228 speakers) with little overlap with the negative example data.

The performance measures used in this paper are the equal error rate (EER), and NIST's minimum detection cost function (DCF), which is defined as the Bayesian risk with probability of target equal to 0.01, cost of false alarm equal to 1, and cost of

Method	none	mean/std	Gaussian	uniform
Not smoothed	14.89	13.78	13.82	13.33
Smoothed	14.77	12.38	12.30	11.97

Table 1: Comparison of EER for different normalization methods on SRE2005 1-side

miss equal to 10. SVMlight [10] is used to perform regression on the class labels with a cost for errors on positive samples equal to the ratio of negative samples to positive samples.

As mentioned earlier, the resulting vectors x_i are normalized on a per-component basis using the statistics obtained on the set of negative examples. We tried three different types of normalization: subtraction of mean and division by standard deviation (mean/std), conversion to normal, and conversion to uniform. The last two methods correspond to transformations designed to turn the distribution of each feature into a standard Gaussian and uniform distributions, respectively. Table 1 shows the results on SRE2005 for the different normalization methods, with and without smoothing. We see that performing no normalization is suboptimal. We can understand this by considering the interpretation of the SVM as MAP estimation with a prior on the weights with equal variance on all components. This means that large weights are penalized equally for all components. This assumption implies that we expect features with smaller ranges to be less important, since, a priori, they would have a smaller contribution in $f(x)$. The fact that normalization is necessary in our experiments implies that features with smaller range are as important as features with large range, that is, regions with low probability are as important as regions with high probability. Interestingly, although not yet understood, uniform normalization outperforms the other two normalization methods. Furthermore, smoothing gives an improvement only on normalized features. This means that we can assume smoothness on the weights only after the ranges of the input features have been normalized.

Table 2 shows the results for SRE2006. The first and last lines (VQ and VQS) correspond to the system described in this paper, without and with smoothing kernel, respectively, using uniform normalization. The second and third lines show a comparison with two alternative systems. The VQB method uses two GMMs for each GMM in the VQ system, a large one and a smaller one, keeping the total number of Gaussians equal to that of the VQ system. The smaller GMM results in more robust features, which compensate in part for the noisiness of the larger GMM. EM refers to a system for which the original GMMs are obtained using the EM algorithm instead of simple VQ. Using EM results in a GMM where Gaussians overlap each other significantly, resulting in more robust but less sensitive features (for more details on this, see [3]). Applying the smoothing kernel to the EM system does not lead to improvement, arguably because there is less noise present in these features. We see that when only one positive sample is available VQS performs better than the other three methods, although the difference between EM and VQS is not significant (significance is measured here by a McNemar test at level 0.05). When more positive examples are available, smoothing still helps over the simple VQ system, although the difference is not significant.

Figure 2 shows the EER for each N-gram length separately and the overall system. Clearly, the improvement achieved from smoothing is relatively larger for shorter N-grams, being 15% for unigrams on both training conditions. We can see that, for N-gram lengths of 1 and 2, VQS significantly outperforms all three other systems in both conditions. For N-gram length equal to 3 there is a significant improvement only over the simple VQ.

Method	1-side		8-sides	
	EER	DCF	EER	DCF
VQ	13.65	0.601	4.91	0.241
VQB	13.28	0.575	4.91	0.228
EM	12.25	0.553	4.85	0.224
VQS	12.09	0.544	4.79	0.214

Table 2: System comparison for SRE2006

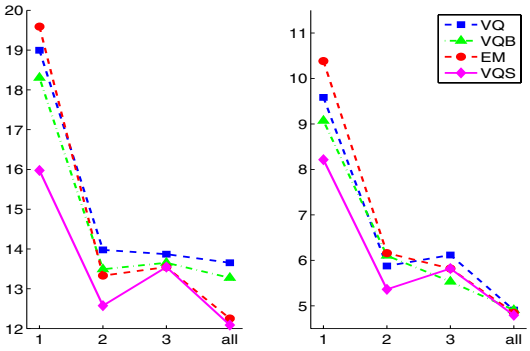


Figure 2: EER for each N-gram length and the overall system for SRE2006, for 1-side (left) and 8-side (right) training.

6. Conclusions

We presented a smoothing kernel derived from adding a regularization term to the SVM objective function for the case in which the input features are spatially related. We show results in a speaker verification task, where the original syllable-level prosodic features are transformed into a fixed-length vector using a particular implementation of the Fisher kernel. Results show that the smoothing kernel leads to 10% improvement in the overall system and 15% in the unigram-only system.

7. Acknowledgments

We thank the compression group at Stanford University for helpful discussions. This work was supported by NSF IIS-0544682. The views herein are those of the authors and do not necessarily represent the views of the funding agency.

8. References

[1] V. Vapnik, *The nature of statistical learning theory*, Springer, 1999.

[2] T. S. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances in Neural Information Processing Systems*, 1998, vol. 11, pp. 487–493.

[3] L. Ferrer, E. Shriberg, S. Kajarekar, and K. Sönmez, "Parameterization of prosodic feature distributions for SVM modeling in speaker recognition," in *Proc. ICASSP*, Honolulu, Apr. 2007.

[4] N. Scheffer and J. F. Bonastre, "UBM-driven discriminative approach for speaker verification," in *Proc. Odyssey-06*, Puerto Rico, USA, June 2006, pp. 1–7.

[5] V. Wan and S. Renals, "Speaker verification using sequence discriminant support vector machines," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 2, pp. 203–210, 2005.

[6] W. Campbell, D. Sturim, and D. Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 308–311, May 2006.

[7] P. Sollich, "Probabilistic interpretation and bayesian methods for support vector machines," in *Proc. ICANN*, Edinburgh, Sept. 1999, pp. 91–96.

[8] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *NeuroCOLT2 Technical Report NC2-TR-1998-030*, 1998.

[9] E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A. Stolcke, "Modeling prosodic feature sequences for speaker recognition," *Speech Communication*, vol. 46, no. 3-4, pp. 455–472, 2005.

[10] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. 1999, MIT Press.