Minimizing System Correlation in SVM Training

Luciana Ferrer

1 Description of the problem

We will consider a binary classification task for which two separate classifiers are available. Each classifier may use different input features and different modeling techniques. In a setup like this, the final decision is made based on a combination of the outputs generated by both classifiers with the hope that the final performance will be better than the performance of the two individual classifiers. This, nevertheless, is not necessarily the case. In the extreme, if both classifiers were generating exactly the same output for each sample, the combined classifier could never have a better performance than the individual ones, independently of the combination procedure used. Intuitively, what we wish is to have two classifiers for which the within class correlation is small. This way, both classifiers contribute independent information leading to a better final decision.

In this work we will study the case in which one of the classifiers is given to us (we can consider this system as a black box which simply gives us a value for each sample) and the other one is an SVM which we need to train. Our goal is to modify the training criteria for the SVM so that the score resulting from this system is as little correlated as possible to the scores from the black box system.

2 Anti-correlation Kernel

In this section we will derive the optimization problem we need to solve in order to achieve the combined goal of minimizing the error of the SVM system (which we will call S) while also minimizing the correlation of this system with the original black box system (which we will call B). Given a training set $T = \{(x^{(i)}, y^{(i)}); i = 1, ..., m\}$, the standard SVM problem is:

$$\begin{array}{lll} \text{minimize}_{w,b,\epsilon} & J(w,\epsilon) = \frac{1}{2} w^t w + C \sum_i \epsilon_i & \\ \text{subject to} & y^{(i)}(w^t x^{(i)} + b) \geq 1 - \epsilon_i & i = 0, ..., m \\ & \epsilon_i \geq 0 & i = 0, ..., m \end{array} \tag{1}$$

We want to modify the objective function by adding a term $\lambda\rho^2$ in the objective function, where λ is a tunable parameter and ρ is the within-class correlation between system S and system B. Given the scores $\{b^{(i)}; i = 1, ..., m\}$ from system B for the training set T, we can compute the within-class correlation between the scores produced by the SVM and these scores the following way:

$$\rho^2 = \frac{\text{cov}(B, S|Y)^2}{\text{var}(B|Y)\text{var}(S|Y)} \tag{2}$$

where $\text{cov}(B, S|Y)$, $\text{var}(B|Y)$ and $\text{var}(S|Y)$ are the within-class covariance and variances. These can be approximated by the within-class sample covariance and variances in the training set T. The within-class sample covariance can be calculated as,

$$\text{cov}(B, S|Y) \approx \frac{1}{m} \sum_{\alpha=1,-1} \sum_i I(y^{(i)} = \alpha)(b^{(i)} - \bar{b}_\alpha)(s^{(i)} - \bar{s}_\alpha) \tag{3}$$

where $\bar{b}_\alpha$ and $\bar{s}_\alpha$ are the sample means for each class $\alpha = 1, -1$. The value $s^{(i)}$ is the output of the SVM, *i.e.* , $s^{(i)} = w^t x^{(i)} + b$. Replacing this into (3) we get,

$$\text{cov}(B, S|Y) \approx w^t K \tag{4}$$

where,

$$K = \frac{1}{m} \sum_{\alpha=1,-1} \sum_{i} I(y^{(i)} = \alpha)(b^{(i)} - \bar{b}_\alpha)(x^{(i)} - \bar{x}) \tag{5}$$

where $\bar{x}$ is the vector of feature means. K is simply the vector of within-class covariances between each input feature and the scores from system B. Similarly, we can compute $\text{var}(S|Y)$ and get $w^t M w$ where M is the within-class sample covariance matrix of the training set T. Calling $v = \text{var}(B|Y)$, we can write ρ^2 as:

$$\rho^2 = \frac{w^t K K^t w}{v \ w^t M w} \tag{6}$$

The new objective function then is $J(w, \epsilon) = \frac{1}{2} w^t w + \frac{1}{2}\lambda \frac{w^t K K^t w}{v \ w^t M w} + C \sum_i \epsilon_i$. This function is not convex. On the other hand, if instead of trying to minimize the within-class correlation we try to minimize the within-class covariance, we get $J(w, \epsilon) = \frac{1}{2} w^t (I + \lambda K K^t) w + C \sum_i \epsilon_i = \frac{1}{2} w^t A w + C \sum_i \epsilon_i$, where $A = I + \lambda K K^t$ is a symmetric positive semidefinite matrix. Now, by doing a change of variable $\tilde{w} = Bw$, with $A = B^t B$ (*i.e.* , B is a matrix square root of A), we can write the new optimization problem as:

$$\begin{array}{lll} \text{minimize}_{\tilde{w},b,\epsilon} & J(\tilde{w}, \epsilon) = \frac{1}{2} \tilde{w}^t \tilde{w} + C \sum_i \epsilon_i & \\ \text{subject to} & y^{(i)}(\tilde{w}^t z^{(i)} + b) \geq 1 - \epsilon_i & i = 0, ..., m \\ & \epsilon_i \geq 0 & i = 0, ..., m \end{array} \tag{7}$$

where $z^{(i)} = (B^{-1})^t x^{(i)}$. This is simply another SVM problem with kernel $K(x, y) = x^t A^{-1} y$. The matrix A^{-1} can be computed extremely efficiently using the matrix inversion lemma by which $A^{-1} = I - \frac{\lambda}{1 + \lambda K^t K} K K^t$, thus,

$$K(x, y) = x^t y - \frac{\lambda}{1 + \lambda K^t K} x^t K y^t K \tag{8}$$

We can give an intuitive interpretation to this kernel. When λ is small this kernel is close to the linear kernel. When λ grows to infinity the kernel substract the projection of the points x and y into the vector K from the linear kernel. The resulting value of the kernel will be small if x and y are both aligned with K. Since the SVM will only make an effort to separate points which give a high kernel value, this means we are considering vectors which direction is close to that of K to be unimportant and, in consequence, emphasizing the importance of the vectors which direction is different from that of K.

3 Simulation

In order to test this idea we created a toy problem. We generated data for two classes with model $x = Cy + m_\alpha$, where the y_i are generated independently with a normal distribution with zero mean and unit variance, C is a random matrix intended to create correlation between the features and m_α is a vector of zeros for one class and a vector of ones for the other class. We then took half of the features and trained an SVM, which served as system B. The remaining features were used to train system S with varying values of λ. We created two separate sets, one for training and one for testing.

Figure 1 shows the scatter plot of scores (on the training data) for both systems with $\lambda = 0$ and $\lambda = 10^7$. We can see that for the large value of λ, the within class correlations have been reduced (eventhough we are actually minimizing the covariance and not the correlation). We can also see from this picture that the separation of the two classes is better for the larger λ which implies that the performance of the combination should be better in this case. Figure 2 confirms this observation. In this figure we see the error rates for system S, system B and the combined system and the correlation between system S and system B as a function of the value of λ for the test data. The error for system B (blue line) does not depend on λ. The error for system S (red line) grows with the value of λ since we are tradding off poorer performance in

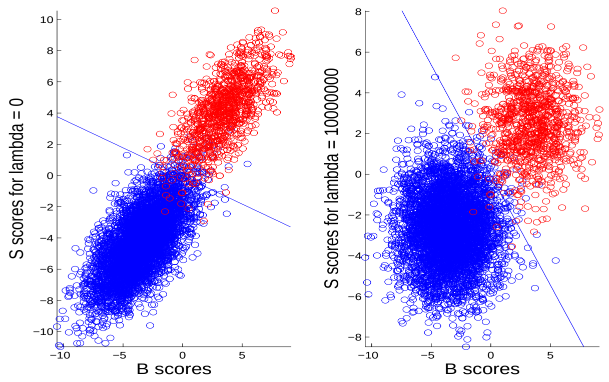

Figure 1: Scores from system B versus scores from system S for two values of λ

exchange for lower correlation with system B. The black line shows that, in fact, we achieve lower correlations (reaching a value of zero) for higher values of λ. Finally, the green line shows the performance measure we care about, the performance of the combined system. The combination is performed using another SVM which is trained on the training set with the scores from the two SVM systems, B and S, for each value of λ. The combination performance improves for higher values of λ, from 2.02% to 1.32%. This is a 35% relative improvement. Similar results were found by changing the random seed, the number of features, the number of training and test samples, etc.

4 Application to Speaker Verification

Speaker verification is the task of deciding whether a speech sample was produced by a certain target speaker or not. It is a binary classification task where the two classes are *true-speaker* and *impostor*. We are considering a speaker verification system composed by two subsystems briefly described below: the UBM-GMM and the MLLR-SVM. The goal is to use the method explained above to train the MLLR-SVM subsystem taking into account the fact that the final system will be a combination of both these systems.

4.1 Universal Background Model GMM (UBM-GMM) system

The UBM-GMM system is a generative classifier. The input features are given by the Mel Frequency Cepstral Coefficients (MFCC), which serve as a description of the vocal tract characteristics of the speaker. Each class (impostor and true-speaker) is modeled using a Gaussian mixture model (GMM). The impostor GMM, usually called background GMM, is trained using data from a large set of held-out speakers. The true-speaker models for each speaker (called target models) are trained by adapting the background GMM to the speaker's training data. The score is finally computed as the logarithm of the ratio between the likelihoods of both models given the test data [1].

4.2 Maximum Likelihood Linear Regression SVM (MLLR-SVM) system

State-of-the-art speech recognition systems use methods to adapt the models for the acoustic units to the speaker of the utterance being recognized. One strategy for performing this adaptation is to compute a linear transformation of the means of the Gaussians of the acoustic models such that the transformed models maximize the likelihood of the utterance. The parameters of this MLLR transform encode a description of the speaker specific characteristics of the acoustic models, i.e., they describe how the speaker differs from the average population of speakers used to train the original acoustic models. Thus, these parameters can be used to train the SVM target models for speaker verification systems [2, 3]. The MLLR-SVM system is currently one of the best performing speaker verification systems.

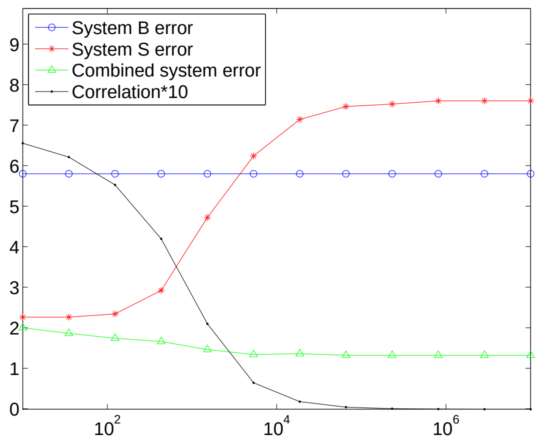

Figure 2: Error and correlation as a function of λ

4.3 Application of the proposed method to the speaker verification problem

In speaker verification we need to train one model (SVM) for each target speaker. In this work we will consider the case in which around 2.5 minutes of a telephone conversation are available for training and another 2.5 minutes are available for testing. The MLLR-SVM system converts each sample into one long vector of dimension 6240 containing the parameters of the MLLR transforms for each class of acoustic models. This implies that when training a target SVM we have only one vector labeled *true-speaker*. On the other hand, we can use all the held-out conversation used for training the UBM model in the UBM-GMM system as samples for the *impostor* class[1].

To implement the proposed method to train a target model we need to estimate the vector K of within-class covariances between the UBM-GMM system and each of the MLLR features. Since only one sample is available for the true-speaker class, we will estimate vector K as the covariances for the impostor class only (i.e., in Equation 3 the first sum is done only over $\alpha = -1$). To estimate this covariance we need to have the output of the UBM-GMM system corresponding to the speaker for some significant amount of impostor samples. For this, we select a set of 184 held-out utterances from different speakers (which are never target speakers) and obtain scores from the UBM-GMM system (using the adapted model corresponding to the target speaker). These samples are used to obtain an estimation of the vector K according to Equation 3. This procedure has to be done separately for each target speaker. This implies that a different kernel is used to train (and test) each target SVM.

4.4 Results

We applied this method to the data from the 2005 speaker recognition evaluation organized by NIST. The correlation vectors are estimated using speakers from the 2004 evaluation. The resulting MLLR-SVM systems for different values of λ are combined with the UBM-GMM sytem using a linear perceptron. Figure 4.4 shows the curve of false rejection versus false acceptances for five different systems[2]: The UBM-GMM system, two MLLR-SVM systems, one for $\lambda = 0$ and one for $\lambda = 10$ (results do not change for larger values of λ) and the two corresponding combined systems. We can see that a significant improvement (of around 5% on most operating points) on the combined system is achieved by using the proposed kernel.

Curves similar to the ones presented in Section 3 could be shown here. We do not show them due to lack of space, but we want to point out that eventhough the correlation between both systems goes down when λ increases, the value never reaches 0. For $\lambda = 0$ the average correlation for all speakers is 0.57. For large values of λ the correlation decreases to 0.43. On the other hand, the inner product between the vector

[1]In order to compensate for this very unbalanced number of samples for the classes we use a weight of 500 for the slack variable corresponding to the positive sample and a weight of 1 for the slack variables corresponding to the negative samples in the objective function of the SVM

[2]Here we use the complete curve that is obtained by sweeping a threshold on the scores to show that improvements are obtained at any value of the threshold. Speaker recognition systems are usually used on the left upper corner of this plot, where false acceptances are very low.

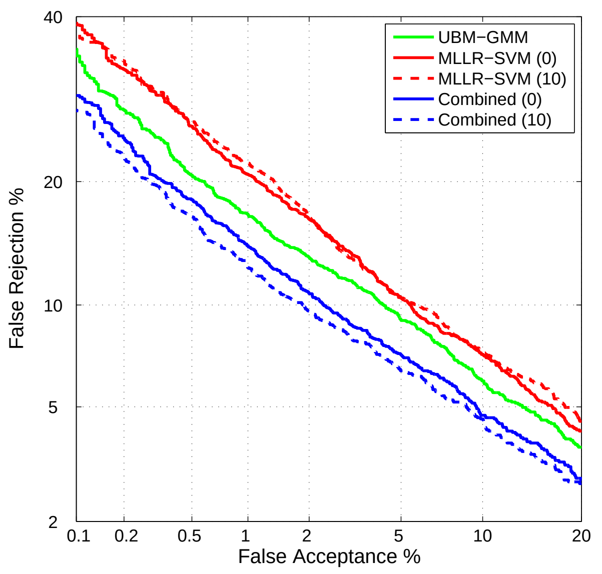

Figure 3: False rejection versus false acceptances for the UBM-GMM, two MLLR-SVM systems and the corresponding combinations

K and the resulting weight vector w does reach 0, which is what we expect this kernel to achieve. This indicates that the SVM is finding the right weight vectors for the estimated K vectors, but that these vectors that are estimated from the held-out data are not predicting the correlations between the features and the UBM-GMM system accurately. This might indicate that we should use more impostor samples to estimate K, that we need better matched data (2004 and 2005 databases where somewhat different data collections) or that we need to include true-speaker samples in the estimation of K. In any case, this is an issue that needs further investigation.

There are two ways in which the results presented in this section are somewhat optimistic. First, the optimal value of lambda was chosen to optimize the test performance; and second, the combiner was trained on the test data. We believe that both are relatively minor problems. In the case of the choice of lambda, we saw in the simulations that the performance of the combination converges to a certain value as lambda grows. This same effect is seen in the speaker verification performance. Thus, the exact choice of lambda is not important, as long as the value is large enough. The problem of the combiner being trained on the test data is somewhat more important, but since the combiner is simply a perceptron with only three parameters and the number of training samples is large (26270 samples, 10% of which are true-speaker samples) the possibility of overfitting the parameters to the test data is very small. For these reasons, we believe that when we test the system on new unseen data we will obtain results similar to the ones presented above.

4.5 Conclusions

The presented method shows to be effective in finding SVMs that lead to improved performance for the overall combined system with respect to the performance obtained by training the SVMs to optimize the performance of the subsystem by itself.

References

[1] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn, “Speaker verification using adapted Gaussian mixture models,” *Digital Signal Processing*, vol. 10, pp. 19–41, 2000.

[2] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman, “MLLR transforms as features in speaker recognition,” in *Proceedings of the 9th European Conference on Speech Communication and Technology*, Lisbon, Sept. 2005, pp. 2425–2428.

[3] A. Stolcke, L. Ferrer, and S. Kajarekar, “Improvements in MLLR-transform-based speaker recognition,” in *Proc. IEEE Odyssey 2006 Speaker and Language Recognition Workshop*, San Juan, Puerto Rico, June 2006.