An extension of the Chernoff-based transformation matrix estimation method for on-line learning in Bayesian binary hypothesis tests

F.D. LORENZO-GARCÍA, J.L. NAVARRO-MESA, A.G. RAVELO-GARCÍA
Departamento de Señales y Comunicaciones. Departamento de Telemática
Universidad de Las Palmas de Gran Canaria
Campus Universitario de Tafira. Las Palmas de Gran Canaria.
SPAIN

Abstract: - In a previous paper [8] we have proposed a method to improve the classification between two classes in a new transformed space using the Chernoff similarity measure. The key idea is to estimate a transformation matrix such that the overlap between the pdf associated to the competing classes is minimum thus leading to a minimization of the classification error. Starting from a surrogate cost function we review the previous method from the consideration that in many practical applications the (online) learning examples come in a sample-by-sample manner instead of a batch manner. Then we propose a new formulation of the learning algorithm in on-line mode and we derive the corresponding formulation. We arrive to iterative formulations of the estimation processes. The classes are modeled by a Gaussian mixture model with a varying number of components and we investigate the new method for several dimensionalities of the transformed subspace. The experiments are carried out over a database of speech with and without pathology and we show that the performance of the on-line approach compares favorably with respect to the batch mode and outperforms some reference methods.

Key-Words: - Learning algorithm, Chernoff bound, transformation matrix, cost function, Gaussian mixture models

1 Introduction

The problem of discrimination between classes is well known and has been studied in several papers. The main problem is that the classification scores degrade significantly when the classes are highly confusable, that is, the corresponding probability density functions (pdf) are highly overlapped. Our main objective is to tackle this problem in order to achieve low confusability in binary hypothesis tests. Various discriminative methods have been proposed in the literature to deal with the problem of confusability. In [2] an average divergence measure is used as criterion for finding a transformation matrix which maps the original features into a subspace with a better discriminative ability. This and other approaches [2,3,5] use subspace projections by maximizing an appropriately chosen class-separability criterion. We have previously worked on this idea in [6,8] relying on the Chernoff bound as a measure of similarity between competing classes in binary hypothesis tests where each class is modeled by means of Gaussian Mixture Models. The paper presented here is an incremental work based on [6,8]. We offer a new vision of the learning algorithms in terms of the way in which samples are incorporated in the learning process.

The starting concept of our approach to the design and estimation of transformation matrices is the Chernoff bound. It is, in fact, an upper bound of the probability of error. Based on this concept, we start by defining a surrogate cost function whose maximization let to approach a minimization of the classification error.

On-line and batch learning algorithms are used in the estimation process of the transformation matrix. In the on-line learning, the whole sample is not used in batch at each step of the algorithm, but only the latest observation vector x^t is used, while in batch learning the entire training set is used at every step of the iteration to form the expectation. On-line learning algorithms are typically more efficient, and simple to implement, however common learning problems fit more naturally in the batch learning setting. Batch learning algorithm is probably the most common supervised machine-learning setting and it has represented the first algorithm in our methodology. This subject will be developed in section 2.

A selected set of Gaussians from each class participates in the process and the transformation matrix is expected to separate the Gaussians in the new subspace. For this purpose we develop an iterative method based on the steepest ascent method

that aims at finding maximum discrimination between classes. Different initializations were studied in [10] and here we use one of them based on the maximization of the Bhattacharyya distance [5]. This subject and other optimization details will be developed in section 3.

Once defined the cost function and the maximization method we centre our experiments in pathological speech classification (section 4). The vectors in the original space are formed by Mel-warped log-filterbank energies (MFE) features. The main objective of our experiments is to study the on-line algorithm in terms of the dimensionality in the transformed space and compare its performance with the batch version. Our reference transformation matrix is the one based on the discrete cosine transform which is classic in the speech recognition literature.

2 Chernoff-based cost function and the learning algorithms

The probability of error is a good measure of the performance of a decision rule [4]. Let $q_i(x)=P_i p_i(x)/p(x)$ be the a posteriori probability of class i given a pattern x. Then, the conditional error given x, due to the Bayes decision rule is either $q_1(x)$ or $q_2(x)$ whichever smaller and the Bayes error can be estimated through the following expression.

$$\begin{aligned}\varepsilon &= \int \min\left[q_1(x), q_2(x)\right] p(x)dx \\ &= \int \min\left[P_1 p_1(x), P_2 p_2(x)\right]\ dx\end{aligned} \qquad (1)$$

where $p_i(x)=p(x/w_i)$ $\{i=1,2\}$ and $P_i=P(w_i)$ are the conditional probability density and the a priori probability of the i class, respectively.

An upper bound of the second integrand in (1) can be obtained by stating the fact that $min[a,b] \le a^{\alpha} b^{1-\alpha}$, where $0 \le \alpha \le 1$. Then, using this inequality the total error can be bounded by

$$\overline{D} = \max_{0\le\alpha\le 1} \left\{ -Ln\left(\int P_1^{\alpha} p_1^{\alpha}(X) P_2^{1-\alpha} p_2^{1-\alpha}(X) dx \right) \right\} \qquad (2)$$

This expression is called the Chernoff bound and it can also be seen as a measure of similarity between two pdf. For example, each pdf may define the probability of pertaining to a given class, and therefore measures how similar or different the two classes are. Obviating the maximization in (2) with respect to α and without loss of generality it is assumed to be constant hereafter and equal to $1/2$ which results in the well known Bhattacharyya distance. The important fact for our purposes is that the larger the distance $\overline{D}$ is between the two distributions, the smaller the probability of misclassification between classes. In fact, the expression in (2) is often used to obtain an upper bound on the probability of misclassification such that the bigger the distance the smaller that probability [9].

The maximization of the Chernoff distance has been used in [8] to find a transformation matrix. The objective was to obtain an improved discrimination capability to reach a minimized classification error.

Taking into account that in practical applications the samples to be classified are discrete and finite, a more suitable form of expression (2) is as follows. In a general supervised training, let $Y=\{(x^1,y^1),\ldots,(x^N,y^N)\}$ be a finite set of training instances, where each instance x^t corresponds to a label $y^t=\{1,2\}$. Let $A\ (k,x,m)$ be a linear matrix which maps an original observation x^t into a transformed one as $v^t=A^T x^t$, where x^t is a k-dimensional vector, v^t is an m-dimensional vector and $m \le k$. In [6,8] we have suggested that the convex nature of (2) due to the application $-Ln[.]$ over the integral is not appropriate to obtain the transformation matrix. Instead, we proposed a surrogate cost function which is based on the Chernoff distance as follows:

$$D = \max_{A} \tanh\left\{ -S \cdot Ln\left[\sum_{t=1}^{N} p_1^{\alpha}(v^t) p_2^{1-\alpha}(v^t) \right] \right\} \qquad (3)$$

where for convenience the a priori probabilities have been included inside the expression of the conditional probabilities, the integral has been substituted by a summation because the training set is finite, the hyperbolic tangent has been introduced for convenience and $S>0$ is a constant factor that controls the dynamic range of the argument. Note that the effect of $tanh(.)$ is that when the pdf's of the two classes tend to overlap both $\overline{D}$ and D tend to zero but when the overlap decreases then D tends to one. The resulting function is concave increasing and facilitates the search for minimum confusability between classes by searching for the maximum of (3). This search will be done with a stochastic gradient ascent method [10] as follows. In constrained optimization problems, the cost functions use to be complicated and it is difficult to arrive to closed solutions. The cost function (3) has the form J(A) = f{g(A,v)}. In our work the function $f\{.\}$ takes the form of $tanh(-SLn(\Sigma(.)))$ and function $g(A,v) = p_1(v)^{\alpha} p_2(v)^{1-\alpha}$. The steepest ascent learning rule becomes

$$A^{(k+1)} = A^{(k)} + \gamma_k \left.\frac{\partial J(A)}{\partial A}\right|_{A=A^{(k)}} \quad (4)$$

for *{k=1,2,..., N_k}* where *k* is the iteration index, N_k is the number of iterations, $\gamma_k=\gamma_0$ *(1-k/(N_k-1)* is a step size that depends on the step and γ_0 is a small initialization constant that has been set to 0,01 in the experiments. The gradient matrix $A^{(k+1)}$ is computed at the point $A^{(k)}$. Substituting equation (3) in (4) and making the derivatives and some mathematical arrangements we have:

$$\frac{\partial J(A)}{\partial A} = S\left[1-\tanh^2\left(-SLn\left(\sum_{t=1}^{N} p_1^{\alpha}(v^t)\,p_2^{1-\alpha}(v^t)\right)\right)\right] \left[\frac{\partial}{\partial A}\left(-Ln\left(\sum_{t=1}^{N} p_1^{\alpha}(v^t)\,p_2^{1-\alpha}(v^t)\right)\right)\right] \quad (5)$$

Apart from a constant factor, the expression *Σ(g(A,v))* in (5) takes the form of a kind of expectation which is approximated by the sample mean of this function over the sample $v^1,\ldots,v^N$ set. This kind of learning algorithm, where the entire training set is used at every step of the iteration to form the expectation, is called batch learning. In principle, it is easy to form the gradient and Hessian of the cost functions because first and second derivatives with respect to the elements of *A* can be taken inside the expectation with respect to *v*. It suffices that function *g(A,v)* is twice differentiable with respect to the elements of *A* for this operation to be allowed.

However, in many practical situations it may sometimes be tedious to compute the mean values or samples averages of the appropriate functions at each iteration step. This is especially problematic if the number of samples is not fixed but new observations keep on coming in the course of the iteration. The statistics of the samples may also be (slowly) varying, and the algorithm should be able to track this. In the learning paradigm called on-line learning, the whole sample is not used in batch at each step of the algorithm, but only the latest observation vector v^t is used. In this case, the expectation has to be dropped from (4) and (5) and the on-line learning rule becomes

$$A^{(k+1)} = A^{(k)} + \gamma_k \left.\frac{\partial g(A)}{\partial A}\right|_{A=A^{(k)}} \quad (6)$$

where now each sample v^k is considered in the iteration (e.g., *{k=1,2,..., N}*), *N* is the sample size and $\gamma_k=\gamma_0$ is a constant step size that has been set to *1/N* in our experiments. The gradient matrix $A^{(k+1)}$ is computed at the point $A^{(k)}$. Taking the expression of *g(A,x)* and making its derivative with respect to *A* and some mathematical arrangements we have:

$$\frac{\partial g(A)}{\partial A} = S\left[1-\tanh^2\left(-SLn\left(p_1^{\alpha}(v^t)\,p_2^{1-\alpha}(v^t)\right)\right)\right] \left[\frac{\partial}{\partial A}\left(-Ln\left(p_1^{\alpha}(v^t)\,p_2^{1-\alpha}(v^t)\right)\right)\right] \quad (7)$$

This leads to fluctuating directions of instantaneous gradients on subsequent iteration steps (*{1,2,..., N_k}*), but the average direction in which the algorithm proceeds is still roughly the direction of the steepest descent of the batch cost function. The factor γ_k in (4) plays an important role in controlling these fluctuations. We set $\gamma_k=min(10^{-4}, 1/N)$ where *1/N* controls the contribution of each sample and 10^{-4} prevents from a high constant when $N\leq 10^{-4}$ Generally, stochastic gradients algorithms converge much slower than the respective steepest ascent algorithms. This is compensated by their often very low computational cost. The computational load at one step of the iteration is considerably reduced. This is because the value of the function *∂/∂A[g(A,x)]* has to be computed only once, for vector x^t.

3 Optimization details

Let's particularize for the case in which each class *j={1,2}* probability $p_j(X)$ is characterized by a mixture model of *M* Gaussian components (GMM) with means μ^i_j, covariance matrixes Σ^i_j, weighting factors ω^i_j and *{i=1,…,M}*, $N^i_j(\mu^i_j,\Sigma^i_j)$. The objective is to obtain the matrix *A* such that the cost function is maximized since it is associated to a minimum classification error with the new pdf $N^i_j(A^T\mu^i_j, A^T\Sigma^i_j A)$. To achieve that we start by taking the partial derivatives of (7) with respect to *A*. Applying the chain rule to the square bracketed part we obtain the following derivative for which only a given mixture component from each class is considered

$$\frac{\partial}{\partial A}\left(-Ln(.)\right) = \left(\alpha\cdot B^i_1 A - \alpha\cdot\Sigma^i_1 A\left(A^T\Sigma^i_1 A\right)^{-1}\left(A^T B^i_1 A\right) + \Sigma^i_1 A\left(A^T\Sigma^i_1 A\right)^{-1}\right) + \left((1-\alpha)\cdot B^l_2 A - (1-\alpha)\Sigma^l_2 A\left(A^T\Sigma^l_2 A\right)^{-1}\left(A^T B^l_2 A\right) + \Sigma^l_2 A\left(A^T\Sigma^l_2 A\right)^{-1}\right)$$

where $B^i_j = A^T(x^t - \mu^i_j)(x^t - \mu^i_j)^T A$ and the component pairs *(i,l)* have been taken from class 1 and 2, respectively. We have previously explored this idea in [6, 8]. The key idea is to take the two closest components from each class. The distance between component pairs has been estimated by means of the Bhattacharyya distance.

In the batch learning algorithm the derivatives in the square bracketed part of (5) are:

$$\frac{\partial}{\partial A}\left(-Ln(.)\right)=\sum_{t=1}^{N}\left[\frac{p_1^{\alpha}\left(v^t\right)p_2^{1-\alpha}\left(v^t\right)}{\sum_{t=1}^{N}p_1^{\alpha}\left(v^t\right)p_2^{1-\alpha}\left(v^t\right)}\right]*$$
$$\left(\alpha\cdot B_1^i A-\alpha\cdot\Sigma_1^i A\left(A^T\Sigma_1^i A\right)^{-1}\left(A^T B_1^i A\right)+\Sigma_1^i A\left(A^T\Sigma_1^i A\right)^{-1}\right)+$$
$$\left((1-\alpha)\cdot B_2^l A-(1-\alpha)\Sigma_2^l A\left(A^T\Sigma_2^l A\right)^{-1}\left(A^T B_2^l A\right)+\Sigma_2^l A\left(A^T\Sigma_2^l A\right)^{-1}\right)$$

With all these equations we obtain two estimation expressions. One is the Iterative Chernoff Maximization using batch learning (ICM-B) in (5) and, the other, the Iterative Chernoff Maximization using online learning (ICM-O) in (7).

Now we can make some considerations. The factor (*1 – tanh²[.]*) in (5) and (7) plays an interesting role because it is a reflection of the way in which the cost function progresses to its maximum when a new iteration is performed. Thus, this factor tends to zero making the derivative smaller step by step. The product between the smoothing factor *S* and the step size γ_k acts as a prevention from a fast progress of (3) to a maximum.

Now we have to pay attention to the way in which the initialization matrix A^0 is made. We will use the Bhattacharyya distance as defined in [5,4] which is coherent with the Chernoff distance when $\alpha=1/2$. Then, A^0 can be estimated as follows. Let $U=[u_1,\ldots,u_n]$ be the eigenvector matrix, and let $\Lambda=diag(\lambda_1,\ldots,\lambda_n)$ be the eigenvalue matrix of $\Sigma_2^{-1}\Sigma_1$ where the super indexes identifying the components from each class mixture have been omitted. Hence in the original space the Bhattacharyya distance can be written as

$$B(1,2)=\sum_{i=1}^{n}\left[\frac{1}{4}\frac{\left\{u_i^T\left(\mu_2-\mu_1\right)\right\}^2}{1+\lambda_i}+\frac{1}{4}\left\{\ln\left(\lambda_i+\frac{1}{\lambda_i}+2\right)-\ln 4\right\}\right]$$

Since we want to maximize the B(1,2) distance, the columns of the matrix $A^{(0)}$ can be initialized with the *m* eigenvector of $\Sigma_2^{-1}\Sigma_1$ corresponding to de *m* largest terms in last equation.

4 Experiments and results

For the classification experiments we have used a database of normal speech (class NS) and pathologic speech (class PS). The database is composed of 54 speakers without pathology and 608 speakers with pathology from the Disordered Voice Database Model 4337. Recordings consisted in sustained vowel /ah/ with varying duration depending on the speaker. A half of the speakers in the database were used for training and half for testing. The original static features were MFE obtained by applying *m=20* triangular filters to the magnitude spectrum and the dimensionality in the transformed space vary from *k=11* to *k=18* in order to explore a wide range of dimensions. The speech waveforms are sampled at 25 kHz or 50 kHz depending on the recording. The signals are blocked into 30 ms. frames with 20 ms. of overlap between adjacent frames. The number of training frames is 38128 and the same for testing. Each frame is passed through pre-emphasis filter (*a=0,95*) and a Hamming window. Then, a 2048 point FFT is applied to the frame to produce a 1024-point power spectrum. The power spectrum is estimated in the maximum power band. This band is taken from 0 to 4 kHz. The output power in a set of sub bands is estimated by combining a weighted sum of frequency components, shaped by the triangular filter, to obtain the output. Logarithms of the *m=20* outputs are then calculated arriving at 20 MFE's feature vectors per frame.

This paper is not devoted to the estimation of the optimum number of mixture components. For our experimental purposes we use what we consider a low *(M=3)* and a moderate high (*M=7*) number of components in this particular application. For comparison purposes the reference features are the MFE in the original space and MFCC in the transformed space. In all experiments the initialization method is the one explained in section 3. We have experimentally found that a good choice of the control factor in (3) is *S=1/32*.

A speaker is classified as follows. Once the transformation matrix is estimated it is used to transform the feature vectors in the original space to the transformed one. Then, new GMM are estimated for each class. In the classification process the whole set of feature vectors of a given speaker are used to estimate an accumulated probability of pertaining to a given class. The class with the highest accumulated probability is the one we assign to that speaker. In table 1 and 2 we show the global classification scores for the four methods mentioned in the previous sections. Different dimensionalities *m* in the transformed space are considered. As we can see one (*M=3*) or both (*M=7*) versions of the methods we propose, ICM-B and ICM-O, outperform the MFE and MFCC in the most experiments. Despite that ICM-O performs worst than the ICM-B for *M=7* its performance is better than the reference methods.

It is interesting to observe that when *M=7* the MFCC transformation does not improve the classification scores in comparison with MFE in contrast to what one could expect from the literature on speech recognition. The scores obtained from the other transformations are really improved. Both ICM-B and ICM-O give the best results. For *M=7* we obtain an improvement of 35,79% in the classification

error over the method MFE and MFCC while for *M=3* using ICM-B the improve of classification error is 43,45% over MFE and 13,24% over MFCC.

m\Method	MFE	MFCC	ICM-O	ICM-B
11	93,05%	93,35%	92,75%	**94,26%**
12	93,05%	92,75%	94,56%	**95,77%**
13	93,05%	93,96%	92,75%	**94,56%**
14	93,05%	94,86%	92,75%	**95,17%**
15	93,05%	94,26%	92,75%	**94,26%**
16	93,05%	95,47%	93,05%	**95,77%**
17	93,055	94,86%	93,65%	**96,07%**
18	93,05%	95,47%	93,65%	**94,86%**

Table 1. Classification scores for M=3

m\Method	MFE	MFCC	ICM-O	ICM-B
11	95,77%	94,56%	**96,07%**	**96,07%**
12	95,77%	94,56%	95,77%	**96,37%**
13	95,77%	95,17%	96,07%	**96,37%**
14	95,77%	94,56%	**97,28%**	96,68%
15	95,77%	95,47%	96,68%	**97,28%**
16	95,77%	95,47%	**96,37%**	**96,37%**
17	95,77%	95,47%	96,07%	**96,68%**
18	95,77%	95,77%	96,07%	**96,37%**

Table 2. Classification scores for *M=7*

In table 3 we can see the confusion matrix of which methods with the best classification score in tables 1 and 2.

m	M	Pathology	NS	PS
17	3	NS	92,59%	7,41%
17	3	PS	3,62%	96,38%
14	7	NS	62,96%	37,04%
14	7	PS	0,33%	99,67%

Table 3. Confusion Matrices

5 Conclusion

We have formulated two forms of a cost function which is based on the Chernoff distance and we have given the formulae for iterative estimations in batch and on-line modes. These formulae have been particularized for Gaussian mixtures to model the competing classes in a transformed space. The simplified version with one Gaussian per class is computationally less demanding than the original one while give good classification scores. As expected, the online learning mode performs worst than the batch mode. However, the performance with pathological speech suggests that our online learning mode is well established. This and other related subjects (e.g., convergence) will be a matter of future work in binary and M-ary hypothesis testing with a special emphasis in extending the formulation to hidden Markov models.

References:

[1] L.R. Bahl, P.F. Brown, P.V. Souza, & R.L. Mercer, "Maximum mutual information estimation of hidden Markov models for speech recognition". Proc ICASSP'86, pp 49-52, 1986.

[2] P.C. Loizou, & A.S. Spanias, "Improved speech recognition using a subspace projection approach". IEEE Transactions on Speech and Audio Processing, Volume: 7, Issue: 3, pp 343–345, 1999.

[3] R. Haeb-Umbach & H. Ney, "Linear discriminant analysis for improved large vocabulary continuous speech recognition". Proc ICASSP'92, pp 13-16, 1992.

[4] Fukunaga, K, "Introduction to Statistical Pattern Recognition". Academic Press, Inc. 2nd Edition, 1990.

[5] P.C. Loizou, & A.S. Spanias, "High-performance alphabet recognition". IEEE Transactions on Speech and Audio Processing, Volume: 4, pp 430-445, 1996.

[6] F.D. Lorenzo-García, J.L.Navarro-Mesa, A. Ravelo-García, S. Martín-González, "New minimum classification error and maximum classification accuracy criterions for Bayesians binary hypothesis testing using linear matrix transformations in GMM". Seventh IMA-IEE International Conference on Mathematics in Signal Processing, pp 98-101, 2006.

[7] M Loog & R. Duin. "Linear Dimensionality Reduction via Heterocedastic Extension of LDA: The Chernoff Criterion". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 38, Nº 6, June 2004, pp 732-739.

[8] F.D. Lorenzo-García, A.G Ravelo-García, J.L. Navarro-Mesa, S.I. Martín-González, P.J. Quintana-Morales, E. Hernández-Pérez. "A Chernoff-based approach to the Estimation of Transformation Matrices for Binary Hypothesis Testing". IEEE International Conference on Acoustics, Speech, and Signal Processing – ICASSP, pp 753 – 756, 2006

[9] T. Kailath, "The Divergence and Bhattacharyya Distance Measures in Signal Selection", IEEE Transactions on Communication Technology, Vol. 15, No 1, Febrero 1967

[10] Hyvärinen K, Karhunen J, Oja E. "Independent Component Analysis". John Wiley & Sons, Inc. 1990.