A SEQUENTIAL APPROACH FOR MULTI-CLASS DISCRIMINANT ANALYSIS WITH KERNELS

Fahed ABDALLAH, Cédric RICHARD, Régis LENGELLE

Laboratoire LM2S, Université de Technologie de Troyes
B.P. 2060, F-10010 Troyes Cedex, FRANCE
fahed.abdallah@utt.fr, cedric.richard@utt.fr, regis.lengelle@utt.fr

ABSTRACT

Linear discriminant analysis (LDA) is a standard statistical tool for data analysis. Recently, a method called Generalized discriminant analysis (GDA) has been developed to deal with nonlinear discriminant analysis using kernel functions. Difficulties for GDA method can arise both in the form of computational complexity and storage requirements. In this paper, we present a sequential algorithm for GDA avoiding these problems when one deals with large numbers of datapoints.

1. INTRODUCTION

Fisher linear discriminant analysis (LDA) is a classical multivariate technique both for dimension reduction and classification. It is based on a projection of the data vectors $\{\boldsymbol{x}_i\}_{i=1,\ldots,n}$ that belong to c different classes into a $(c-1)$ dimensional space in such a way that the quotient between the inter-classes inertia and the intra-classes inertia is maximized [3]. The method consists on an eigenvalue resolution which leads to the so-called canonical variates [2] that contain the whole class specific information in the $(c-1)$ dimensional space. This strategy allows low dimensional representations by using the first variates corresponding to the largest eigenvalues indicating that the major part of the information in the data is conserved. It can also be used as a multi-class classification technique by partitioning the projected space into regions that can be defined as the class membership. The LDA method for classification leads to linear decision boundaries and hence, the method fails for nonlinear problems. Several attempts have been made to incorporate nonlinearity into the original algorithm [2].

In the last few years there have been very significant developments in classification algorithms based on kernels. Kernel-based methods are categorized into nonlinear transformation techniques for representation and for classification. Support Vector Machines (SVMs) were introduced and first applied as alternatives to multilayer neural networks [4]. The high generalization ability provided by these learning machines has inspired recent works in discriminant analysis and feature extraction. Recently, a powerful method for obtaining a nonlinear extension of the LDA method has been proposed and referred to as the Generalized Discriminant Analysis (GDA) [1]. An equivalent approach to the GDA can be found in [2]. The main idea is to map the data into a convenient higher dimensional feature space $\mathcal{F}$ and then to perform the LDA algorithm in $\mathcal{F}$ instead of the original input space. Fortunately, the LDA algorithm can be reformulated into dot product form in $\mathcal{F}$ and there is a highly effective trick to compute scalar products in some feature spaces using kernel functions which satisfy the Mercer condition [4]. Hence, the GDA method can be expressed efficiently as a linear algebraic formula in the transformed space using kernel operators. Nevertheless, this formulation implies that we have to manipulate the Gram matrix $\boldsymbol{K}$ of size (n,n) containing all dot products in $\mathcal{F}$. This can cause problems when the number of patterns is very large. Our goal in this paper is to present a sequential method based on a gradient descent strategy avoiding the need of inverting or even storing the kernel matrix $\boldsymbol{K}$. The presented method allows us to calculate a discriminating axe in every stage after manipulating the elements of the matrix $\boldsymbol{K}$ in a way to remove the contribution of the discriminant axes calculated in preceding stages.

In this paper, we first introduce scatter matrices which are very useful for designing separability criteria. In Section 3 we give a brief review of the GDA which is an efficient extension of the LDA after mapping the data into the high dimensional feature space $\mathcal{F}$. In Section 4, the sequential GDA is reformulated with a gradient descent procedure which avoids the manipulation of the kernel matrix $\boldsymbol{K}$ of size (n,n). Experiments showing the validity of our approach are proposed in Section 5. Brief conclusions are provided in Section 6.

2. SCATTER MATRICES FOR SEPARABILITY CRITERIA

Recall that we consider a multi-class classification problem with d-dimensional patterns $\{\boldsymbol{x}_i\}_{i=1,\ldots,n}$ belonging to c different classes,$\{\mathcal{C}_i\}_{i=1,\ldots,c}$. Let n_l be the number of patterns of the class l thus $\sum_{l=1}^{c} n_l = n$. For convenience and intelligibility, we show hereafter all scatter matrices in the feature space $\mathcal{F}$.
Every kernel-based algorithm starts with the following idea: via a nonlinear mapping

$$\begin{array}{rccc} \boldsymbol{\phi}: & \mathbb{R}^d & \longrightarrow & \mathcal{F} \\ & \boldsymbol{x} & \longmapsto & \boldsymbol{\phi}(\boldsymbol{x}), \end{array}$$

the patterns are mapped into a high dimensional feature space $\mathcal{F}$. LDA can then be performed in $\mathcal{F}$ on the set $\{(\boldsymbol{\phi}(\boldsymbol{x}_i)\}_{i=1,\ldots,n}$. This reflects the notion that performing a nonlinear data transformation into some specific high dimensional feature spaces increases the probability of having linearly separable classes within the transformed space.

The between-class scatter matrix is the covariance matrix of the class centers in $\mathcal{F}$ and is defined as

$$\boldsymbol{B} = \frac{1}{n}\sum_{l=1}^{c} n_l(\boldsymbol{m}_l^{\boldsymbol{\phi}} - \boldsymbol{m}^{\boldsymbol{\phi}})(\boldsymbol{m}_l^{\boldsymbol{\phi}} - \boldsymbol{m}^{\boldsymbol{\phi}})^t, \qquad (1)$$

where

$$\boldsymbol{m}^{\phi} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\phi}(\boldsymbol{x}_i)$$

is the mean calculated over all the data in $\mathcal{F}$ and

$$\boldsymbol{m}_l^{\phi} = \frac{1}{n_l}\sum_{\boldsymbol{x}\in\mathcal{C}_l}\boldsymbol{\phi}(\boldsymbol{x})$$

is the mean in $\mathcal{F}$ of the data belonging to the class $\mathcal{C}_l$. The within-class scatter matrix in $\mathcal{F}$ is the scatter of all samples around their respective class means

$$\boldsymbol{V} = \frac{1}{n}\sum_{l=1}^{c}\sum_{\boldsymbol{x}\in\mathcal{C}_l}(\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{m}_l^{\phi})(\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{m}_l^{\phi})^t. \quad (2)$$

The mixture scatter matrix in $\mathcal{F}$ is the covariance matrix of all samples regardless of their class assignments. It is defined as

$$\boldsymbol{S} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{m}^{\phi})(\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{m}^{\phi})^t. \quad (3)$$

Note that $\boldsymbol{B}$ represents the inter-classes inertia, $\boldsymbol{V}$ corresponds to the intra-classes inertia and $\boldsymbol{S}$ is the total inertia of the data into $\mathcal{F}$. The three matrix are related by the MANOVA equation $S = V + B$. We show in the next section how to use these matrices in order to get a judicious criterion of separability between different classes.

3. GDA METHOD IN FEATURE SPACE

The projection of a pattern $\boldsymbol{\phi}(\boldsymbol{x})$ from the feature space $\mathcal{F}$ to a $(c-1)$-dimensional space is performed by $(c-1)$ discriminant functions

$$z(i) = \boldsymbol{w}_i^t\boldsymbol{\phi}(\boldsymbol{x}) \qquad i = 1,\ldots,c-1. \quad (4)$$

This can be reduced as a single matrix equation

$$\boldsymbol{z} = \boldsymbol{W}^t\boldsymbol{\phi}(\boldsymbol{x}), \quad (5)$$

where $\boldsymbol{z}$ is of components $z(i)$ and $\boldsymbol{W}$ is a $(D, c-1)$ matrix having $\boldsymbol{w}_i$ as columns with $i \in \{1,\ldots,c-1\}$. D represents the dimension of the feature space $\mathcal{F}$ that can be infinite.

The GDA method consists in finding the transformation matrix $\boldsymbol{W}$ that in some sense maximizes the ratio of the between-class scatter to the within-class scatter. A judicious criterion function is the ratio [1, 3, 5]

$$J(\boldsymbol{W}) = \frac{|\boldsymbol{W}^t\boldsymbol{B}\boldsymbol{W}|}{|\boldsymbol{W}^t\boldsymbol{V}\boldsymbol{W}|} \quad (6)$$

where $|\boldsymbol{X}|$ indicates the determinant of a matrix $\boldsymbol{X}$. The columns of an optimal $\boldsymbol{W}$ are the generalized eigenvectors that correspond to the largest eigenvalues in[1]

$$\boldsymbol{B}\boldsymbol{w}_i = \lambda_i\boldsymbol{S}\boldsymbol{w}_i. \quad (7)$$

By observing (7), we can conclude that deriving the GDA solutions may be a computationally intractable problem since we have to work in $\mathcal{F}$ which may be a very high, or even infinitely, dimensional space. However, by using the theory of reproducing kernels [4, 6, 7, 8, 9], such a problem can be solved without explicitly mapping the data to the feature space $\mathcal{F}$. Hence, any vector $\boldsymbol{w} \in \mathcal{F}$ of $\boldsymbol{W}$ must lie in the span of all training samples in $\mathcal{F}$. Therefore $\boldsymbol{w}$ can be written as follows:

$$\boldsymbol{w} = \sum_{i=1}^{n}\boldsymbol{\alpha}(i)\,\boldsymbol{\phi}(\boldsymbol{x}_i) \quad (8)$$

where the $\boldsymbol{\alpha}(i)$'s denote the components of a dual vector $\boldsymbol{\alpha}$ of size n. Note that the largest eigenvalue of (7) leads to the maximum quotient of the inertia [1]

$$\lambda = \frac{\boldsymbol{w}^t\boldsymbol{B}\boldsymbol{w}}{\boldsymbol{w}^t\boldsymbol{S}\boldsymbol{w}}. \quad (9)$$

It can be shown that (9) is equivalent to [1]

$$\lambda = \frac{\boldsymbol{\alpha}^t\boldsymbol{K}\boldsymbol{L}\boldsymbol{K}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^t\boldsymbol{K}\boldsymbol{K}\boldsymbol{\alpha}} \quad (10)$$

where $\boldsymbol{K}$ is the Gram matrix whose components correspond to the inner product of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in $\mathcal{F}$, $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i)^t\boldsymbol{\phi}(\boldsymbol{x}_j)$. Here, κ can be any kernel that satisfies the Mercer condition. $\boldsymbol{L}$ is a (n,n) block diagonal matrix

$$\boldsymbol{L} = (\boldsymbol{L}_l)_{l=1,\ldots,c}$$

where $\boldsymbol{L}_l$ is a (n_l, n_l) matrix with all terms equal to $\frac{1}{n_l}$. The resolution of the eigenvector system (10) requires an eigenvectors decomposition of the matrix $\boldsymbol{K}$, $\boldsymbol{K} = \boldsymbol{P}\boldsymbol{\Gamma}\boldsymbol{P}^t$. It can be shown that the solution $\boldsymbol{\alpha}$ is of the form [1]

$$\boldsymbol{\alpha} = \boldsymbol{P}\boldsymbol{\Gamma}^{-1}\boldsymbol{\beta} \quad (11)$$

with $\boldsymbol{\beta}$'s can be obtained by maximizing λ in

$$\lambda\boldsymbol{\beta} = \boldsymbol{P}^t\boldsymbol{L}\boldsymbol{P}\boldsymbol{\beta}. \quad (12)$$

Note that the coefficients $\boldsymbol{\alpha}$ should be divided by $\sqrt{\boldsymbol{\alpha}^t\boldsymbol{K}\boldsymbol{\alpha}}$ in order to get a normalized $\boldsymbol{w}$ as $\boldsymbol{w}^t\boldsymbol{w} = 1$.
The i-th component of the projected pattern $\boldsymbol{\phi}(\boldsymbol{x})$ on a vector $\boldsymbol{w}_i$ is given by using (4) and (8):

$$z(i) = \sum_{j=1}^{n}\boldsymbol{\alpha}_i(j)\kappa(\boldsymbol{x},\boldsymbol{x}_j) = \boldsymbol{\alpha}_i^t\widetilde{\boldsymbol{\kappa}}(\boldsymbol{x}) \quad (13)$$

where $\boldsymbol{\alpha}_i$ is the dual vector corresponding to $\boldsymbol{w}_i$ and the vector $\widetilde{\boldsymbol{\kappa}}(\boldsymbol{x}) = (\kappa(\boldsymbol{x},\boldsymbol{x}_1)\ \ldots\ \kappa(\boldsymbol{x},\boldsymbol{x}_n))^t$. Memory and complexity problems can arise for the GDA method when we deal with large number of patterns since we have to perform an eigenvectors decomposition of $\boldsymbol{K}$. In the next section, we present the sequential GDA which is less memory costing than the previous method since we do not need to manipulate the (n,n) matrix $\boldsymbol{K}$.

4. SEQUENTIAL GDA METHOD

It can be shown straightforwardly that (10) is equivalent to the following quotient

$$\lambda = \frac{\sum_{l=1}^{c} n_l(\boldsymbol{\alpha}^t\boldsymbol{\mu}_{ll})^2}{\boldsymbol{\alpha}^t\boldsymbol{N}\boldsymbol{\alpha}} \quad (14)$$

where the (n,n) matrix $\boldsymbol{N} = \boldsymbol{K}\boldsymbol{K}^t - n\boldsymbol{\mu}\boldsymbol{\mu}^t$, $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\widetilde{\boldsymbol{\kappa}}(\boldsymbol{x}_i)$ and $\boldsymbol{\mu}_{ll} = \boldsymbol{\mu} - \boldsymbol{\mu}_l$ with $\boldsymbol{\mu}_l = \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{C}_l}\widetilde{\boldsymbol{\kappa}}(\boldsymbol{x})$. In the aim of using

[1] $\boldsymbol{B}\boldsymbol{w}_i = \rho_i\boldsymbol{V}\boldsymbol{w}_i$ and $\boldsymbol{B}\boldsymbol{w}_i = \lambda_i\boldsymbol{S}\boldsymbol{w}_i$ are equivalent eigenvalue equations with identical solutions $\boldsymbol{w}_i$. See [2] for a demonstration.

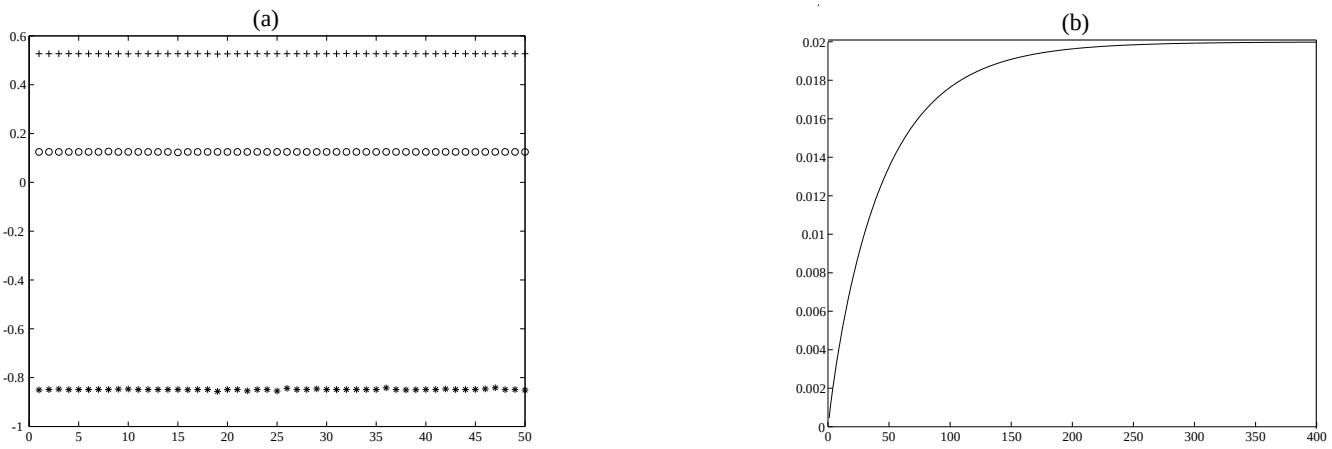

Fig. 1. (a) Represents the projection of each class (50 elements) of Iris data on the first axe of the sequential algorithm. A gaussian kernel was used with $\sigma = 1$. (b) Gives the value of the criterion in (14) as a function of number of iterations.

a gradient descent strategy, we shall now take the inverse of (14) as an objective function to be minimized[2]

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^t \boldsymbol{N} \boldsymbol{\alpha}}{\sum_{l=1}^{c} n_l (\boldsymbol{\alpha}^t \boldsymbol{\mu}_{ll})^2}. \tag{15}$$

The gradient of (15) is given by the following expression:

$$\nabla_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) = \frac{2}{(\sum_{l=1}^{c} n_l (\boldsymbol{\alpha}^t \boldsymbol{\mu}_{ll})^2)^2} \times \left[\boldsymbol{N}\boldsymbol{\alpha} \sum_{l=1}^{c} n_l (\boldsymbol{\alpha}^t \boldsymbol{\mu}_{ll})^2 - \sum_{l=1}^{c} n_l (\boldsymbol{\alpha}^t \boldsymbol{\mu}_{ll}) \boldsymbol{\mu}_{ll} (\boldsymbol{\alpha}^t \boldsymbol{N} \boldsymbol{\alpha}) \right]. \tag{16}$$

From (15), it is obvious that the norm $\|\boldsymbol{\alpha}\|$ of $\boldsymbol{\alpha}$ is irrelevant. This implies that we can keep in (16) $\sum_{l=1}^{c} n_l (\boldsymbol{\alpha}^t \boldsymbol{\mu}_{ll})^2 = \boldsymbol{\alpha}^t \sum_{l=1}^{c} n_l \boldsymbol{\mu}_{ll} \boldsymbol{\mu}_{ll}^t \boldsymbol{\alpha} = \boldsymbol{\alpha}^t \Sigma \boldsymbol{\alpha} = 1$. Here $\Sigma = \boldsymbol{A}\boldsymbol{A}^t$ where $\boldsymbol{A} = [\sqrt{n_1}\boldsymbol{\mu}_{11}, \ldots, \sqrt{n_c}\boldsymbol{\mu}_{cc}]$. This can be done by dividing $\boldsymbol{\alpha}$ by $\|\boldsymbol{\alpha}^t \boldsymbol{M} \boldsymbol{D}^{\frac{1}{2}}\|$ where $\boldsymbol{M}$ and $\boldsymbol{D}$ are respectively the matrices containing the eigenvectors and eigenvalues of Σ. Here Σ is a (n, n) matrix. Thus it may arises in the eigenvector decomposition the same problems of storage and complexity calculation as with the standard GDA. We solve it by observing that Σ has a maximum rank c. Eigenvectors of Σ are the same as those of the (c, c) matrix $\boldsymbol{A}^t \boldsymbol{A}$. Finally, we get the expression of the update $\Delta\boldsymbol{\alpha}$ where the factor of 2 has been ignored:

$$\Delta\boldsymbol{\alpha} = \boldsymbol{N}\boldsymbol{\alpha} - (\boldsymbol{\alpha}^t \boldsymbol{N} \boldsymbol{\alpha}) \sum_{l=1}^{c} n_l (\boldsymbol{\alpha}^t \boldsymbol{\mu}_{ll}) \boldsymbol{\mu}_{ll}. \tag{17}$$

This can be written using only vectors calculation

$$\Delta\boldsymbol{\alpha} = \sum_{i=1}^{n} y(i) \left[\widetilde{\boldsymbol{\kappa}}(\boldsymbol{x}_i) - y(i) \sum_{l=1}^{c} n_l (\boldsymbol{\alpha}^t \boldsymbol{\mu}_{ll}) \boldsymbol{\mu}_{ll} \right] + n \boldsymbol{\mu}^t \boldsymbol{\alpha} \left[\boldsymbol{\mu}^t \boldsymbol{\alpha} \sum_{l=1}^{c} n_l (\boldsymbol{\alpha}^t \boldsymbol{\mu}_{ll}) \boldsymbol{\mu}_{ll} - \boldsymbol{\mu} \right], \tag{18}$$

[2]the reason for using the inverse of (14) will be clear in the following.

where $y(i) = \boldsymbol{\alpha}^t \tilde{\boldsymbol{\kappa}}(\boldsymbol{x}_i)$. The sequential algorithm can be then described as follows:

1. initialize $\boldsymbol{\alpha}$
2. calculate $\boldsymbol{\mu}_l$, $\boldsymbol{\mu}_{ll}$ and $\boldsymbol{\mu}$
3. calculate the eigenvectors and eigenvalues of Σ from $\boldsymbol{A}^t \boldsymbol{A}$ and normalize $\boldsymbol{\alpha}$ as $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} / \|\boldsymbol{\alpha}^t \boldsymbol{M} \boldsymbol{D}^{\frac{1}{2}}\|$, and next calculate $\Delta\boldsymbol{\alpha}$ from (18)
4. update $\boldsymbol{\alpha}$: $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \eta \, \Delta\boldsymbol{\alpha}$, with $\eta > 0$ the learning step size
5. if finish, then exit; otherwise, go to 3.

In order to calculate a second vector $\boldsymbol{\alpha}_2$, we should first eliminate the contribution on the data of the first $\boldsymbol{\alpha}_1$ calculated by maximizing (14). This can be done by observing that in (7), when $\boldsymbol{S}$ is of full rank, the two axes $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ corresponding to the first two eigenvectors of $\boldsymbol{S}^{-1}\boldsymbol{B}$ verify the equation $\boldsymbol{w}_1^t \boldsymbol{B} \boldsymbol{w}_2 = 0$. Thus in $\mathcal{F}$, we should replace $\boldsymbol{\phi}(\boldsymbol{x}_i)$ using the rule

$$\boldsymbol{\phi}(\boldsymbol{x}_i)^{\text{new}} \leftarrow \boldsymbol{\phi}(\boldsymbol{x}_i) - \frac{(\boldsymbol{B}\boldsymbol{w}_1)(\boldsymbol{B}\boldsymbol{w}_1)^t \boldsymbol{\phi}(\boldsymbol{x}_i)}{\|\boldsymbol{B}\boldsymbol{w}_1\|^2} \tag{19}$$

for all the data in the training set. Note that calculating (19) may be a computationally intractable problem. However, observing that our algorithm is formulated using only dot products, we only need to compute the dot products

$$\kappa^{\text{new}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{\phi}(\boldsymbol{x}_i)^{\text{new}})^t \boldsymbol{\phi}(\boldsymbol{x}_j)^{\text{new}}. \tag{20}$$

Developing (20) we obtain the expression of the dot product as

$$\begin{aligned} \kappa^{\text{new}}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{\boldsymbol{\phi}(\boldsymbol{x}_j)(\boldsymbol{B}\boldsymbol{w}_1)(\boldsymbol{B}\boldsymbol{w}_1)^t \boldsymbol{\phi}(\boldsymbol{x}_j)}{\|\boldsymbol{B}\boldsymbol{w}_1\|^2} \\ &= \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{M(\boldsymbol{x}_i, \boldsymbol{x}_j)}{L} \end{aligned} \tag{21}$$

where $M(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is given by

$$M(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{n^2} \sum_{l,f}^{c} \sum_{a}^{n_l} \sum_{b}^{n_f} \kappa(\boldsymbol{x}_a, \boldsymbol{x}_i) \kappa(\boldsymbol{x}_b, \boldsymbol{x}_j) \Gamma(l) \Gamma(f), \tag{22}$$

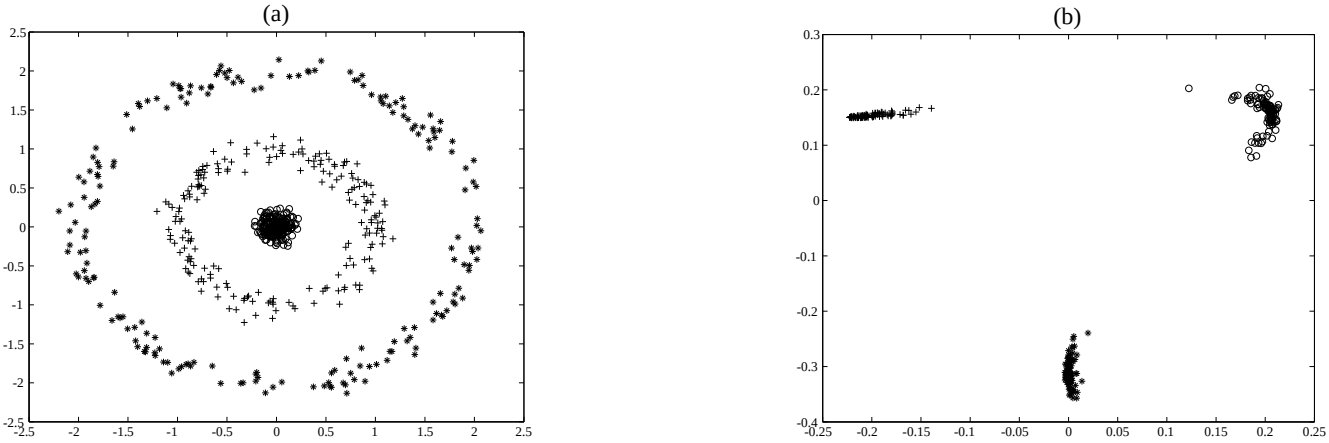

Fig. 2. (a) synthetic data consisting of three classes. The first is represented by $\circ$, the second by $+$ and the third by $\star$. (b) Gives the projection of the whole examples on the first two axes using the sequential approach. A gaussian kernel was used with $\sigma = 1$.

and

$$L = \frac{1}{n^2} \sum_{l,f}^{c} \sum_{a}^{n_l} \sum_{b}^{n_f} \kappa(\boldsymbol{x}_a, \boldsymbol{x}_b) \Gamma(l) \Gamma(f). \qquad (23)$$

Here $\Gamma(r) = \left[\frac{1}{n_r} \sum_{k=1}^{n_r} y(k) - \frac{1}{n} \sum_{p=1}^{n} y(p) \right]$ for a class r. Then, the same algorithm described at the beginning of this section can be applied with $\boldsymbol{K}^{\text{new}}$. If we need to calculate a third axe, we can apply (21) on the elements of $\boldsymbol{K}^{\text{new}}$ instead of the elements of $\boldsymbol{K}$. It is obvious that the complexity of the method increases with the number of axes to be found. Indeed, our approach is computationally efficient to compute the first axe. However, obtaining the second, the third, ..., axes requires to compute the elements of $\boldsymbol{K}^{\text{new}}$ which increases the complexity of the method.

5. EXPERIMENTS

The Iris data consist of 150 4-dimension examples of three classes [1] (each class consists on 50 examples). One class is linearly separable from two other non-linearly separable classes. Figure 1 (a) shows the projection of the three classes on the first axe, that was obtained with the sequential GDA. A gaussian kernel was used with width $\sigma = 1$. The first axe seems to be sufficient to separate the data. Figure 1 (b) gives the value of the criterion in (14) as a function of number of iterations. The step size η was set to 0.02. Next, we consider in Figure 2 (a) three non-linearly separable synthetic classes consisting of samples uniformaly located upon three circles with the same center but with different radius. Each class contains 200 2-dimension samples. The first class is represented by $\circ$, the second by $+$ and the third by $\star$. Even if the first axe is sufficient to separate the classes, we give in Figure 2 (b) the projection of the whole examples on the two first axes using our sequential approach. A gaussian kernel was used with $\sigma = 1$. The step size η was set to 0.04. Note that the second axe allows to separate only classes 1 and 2 from class 3.

6. CONCLUSION

In this paper, we have presented a sequential approach to calculate nonlinear features based on the GDA method proposed by [1]. The importance in the proposed algorithm for sequential GDA is that it does not need the inversion or even the storage of the Gram matrix of size (n, n). However, the weakness of our approach is that the complexity increases with the number of axes to be found.

7. REFERENCES

[1] G. Baudat, F. Anouar. "Generalized discriminant analysis using a kernel approach," in *Neural Computation*, vol. 12, no. 10, pp. 2385–2404, 2000.

[2] V. Roth and V. Steinhage. "Nonlinear discriminant analysis using kernel functions," in *Advances in Neural Information Processing Systems*, S.A. Solla, T.K. Leen, and K.-R. Mller, editors, vol. 12, pp. 568–574, MIT Press, 2000.

[3] K. Fukunaga. *Statistical Pattern Recognition*, San Diego: Academic Press, 1990.

[4] V. Vapnik. *The Nature of Statistical Learning Theory*. New York: Springer Verlag, 1995.

[5] R. O. Duda, P. E. Hart, D. G. Stork. *Pattern Classification*, New York: Wiley and Sons, 2001.

[6] S. Mika, G. Rätsch, J. Weston, B. Schölkopf and K. R. Müller. "Fisher discrimiant analysis with kernels," in *Advances in Neural networks for signal processing* , Y. H. Hu, J. Larsen, E. Wilson, S. Douglas, editors, pp. 41–48, 1999.

[7] K. R. Müller, S. Mika, Rätsch, K. Tsuda and B. Schölkopf. "An introduction to kernel-based learning algorithms," *IEEE Neural Networks*, vol. 12, no. 2, pp. 181–201, May 2001.

[8] S. Saitoh. *Theory of Reproducing Kernels and its Applications*. Longman Scientific & Technical, 1988.

[9] B. Schölkopf, C. Burges and A. Smola. *Advances in Kernel Methods-Support Vector Learning*, MIT Press, 1999.