Incremental and compressible kernel null discriminant analysis

Franck Dufrenois

Version of Record: https://www.sciencedirect.com/science/article/pii/S0031320322001236
Manuscript_fa62d96a5aa2bd0d114b9caa2150f0f7

Incremental and compressible kernel null discriminant analysis for multi-class and one-class learning

F. Dufrenois[a]

[a]*Laboratoire d'Informatique Signal et Image de la Côte d'Opale, 50, rue Ferdinand Buisson, 62228 Calais Cedex*

Abstract

In big data era, learning methods are facing with large and growing multidimensional data bases, most of the time being delivered in flow. In this context, kernel learning methods must solve two fundamental issues: proposing an appropriate formulation with regard to the nature of the data and solving the problem related with the ever-increasing kernel matrix. In this paper, we propose an incremental implementation for computing the solutions of the kernel null discriminant analysis (KNDA), originally introduced in batch mode by Bodesheim et al. in [3]. We show both theoretically and experimentally that our incremental scheme guarantees accurate solutions when compared to state of the art. Additionaly, we introduce a compression strategy based on the properties of the null space of KNDA in order to remove the redundant information in the learning data. This new result allows for the incremental KNDA to be used for large scale scenarios while remaining in acceptable learning time. Numerous experiments both in multi-class problems and one-class problems shows the effectiveness of the proposed strategy.

Keywords: Incremental kernel Discriminant Analysis, null space, compression mechanism, multi-class learning, one-class learning.

1. Introduction

Linear Discriminant Analysis (LDA) and its kernel extension (KDA) are efficient dimensionality reduction tools allowing the classification to multiple categories in datasets. There are used in numerous areas as diverse as speech and music classification [1], video classification [36], outlier detection [34], supervised novelty detection [12, 3], etc... Their attractivity relies on the fact that both are formulated as a Rayleigh quotient whose the solution is easily obtained by solving an equivalent generalized eigenvalue problem (GEP). KDA generalizes LDA to nonlinear data sets by using the famous kernel trick. Specifically, KDA is based on the construction of a kernel matrix and the KDA's solution results from the eigendecomposition of this kernel matrix. With database volume that increases more and more, the storage requirement and the computational cost involved by KDA make impractical its use with real modern

Email address: Dufrenois@lisic.univ-littoral.fr (F. Dufrenois)

data sets. Additionally, KDA is not suited for handling streaming data sets, i.e where data are injected sequentially one at a time or by packets (chunk of data). In this context, two issues must be solved: first, an efficient computation for eigen-decomposition of the kernel matrix must be addressed and secondly, a strategy of compression to avoid the ever-increasing size of the kernel matrix. Concerning the first issue, several works focused on incremental formulation of KDA. Recently, Cai et al. in [4] propose to cast kernel discriminant analysis into a regression framework by using spectral graph analysis. Based on the Cholesky decomposition of the kernel Gram matrix, a recursive implementation of KDA is derived. In this way, this new formulation called SRKDA avoids eigenvector computation and facilitates an incremental formulation of KDA. With the same objective, Gkalelis et al. propose an incremental accelerated kernel discriminant analysis (IAKDA) in [15]. Compared to SRKDA, IAKDA does not require any data normalization which reduces significantly its computational cost and round-off errors. IAKDA is based on a matrix factorization and a simultaneous diagonalization framework. Dimensionality reduction is incrementally obtained by solving a linear system using the block recursive Cholesky framework. However, the linear system is based on the kernel Gram matrix whose the size continuously increases in online setting. This method needs an increasing storage burden as the data arrive. To counter this negative effect, Wang et al. propose an efficient factorization-free Kernel discriminant analysis (IFKDA) [40]. Based on the work of Chu et al. [10], the authors consider that categories can be summarized by theirs centers and use them to construct a k-dimensional Reproducing Kernel Hilbert space (RKH) by solving a simple linear system. The size of the problem is then reduced and the mapping in this RKH space is obtained from the product of two small-sized kernel matrices. However, if effectively FKDA runs extremely fast, our experimental evaluation suggests that considering only class centers as discriminative information is not enough to reach competitive classification performance. Lastly, Liu et al. in [29] introduce an incremental version of the kernel null discriminant analysis (KNDA), initially introduced in batch version by Boldesheim [3]. Based on the null space theory, KNDA build a subspace where within-class scatter vanishes and between-class scatter remains positive. This property allows to represent the entire class by a single point making classification easier. The performance recorded by this approach both in terms of classification and learning time seems very promising.

Aside from Wang's algorithm, most of the previous methods are not suited for online scenario notably because they do not manage the problem of the ever-increasing kernel matrix. This problem is central in most of kernel machine algorithms where the size of the kernel matrix grows with the data. For the purpose of developing a suited LDA for online applications, WonHei et al. propose in [27] an online sketch LDA based on the frequent direction algorithm [28]. The principle of sketch LDA consists in maintaining a low rank sketch matrix of principal components which captures main data variations. Projecting data onto this reduced set of basis vectors leads to perform dimension reduction. As a consequence, the size of the between-class and intra-class scatters remains constant at every stage of the learning process reducing at the same time the computational burden involved by the underlying GEP. However,

this method assumes that the data distribution is Gaussian making it potentially inefficient for nonlinear data. The concept of sketching matrix is also used in [9] to solve the small sample size problem related to discriminant analysis (DA). Considering that solving DA is equivalent to perfom ridge regression on the class membership matrix [44], Chowdhury et al. develop a randomized iterative LDA that guarantees highly accurate solutions. Different random projection and sampling strategies are applied in this work. One difficulty of this kind of method relies on the choice of the sketch size. Again, this method is based on linear DA and can not deal with nonlinear data sets. In [41], a similar work is presented in which a fast Fisher DA is developped. Random projection is used to accelerate regularized Fisher DA and random Fourier features are used to approximate KDA. Based on the integral representation of the kernel function [35, 33], Random Fourier features approximate each entry of the kernel matrix by inner product based on a low dimensional randomized Fourier feature map of the input data. The benefit of this method is strategic since it allows to train linear learning machines on these transformed data and therefore profits from massive time-savings. However, several drawbacks are highlighted in this work. First, the method is not suited for online applications since the number of data must be known in advance and the FDA' solution is computed in bach mode. Second, it is pointed out that the number of the Fourier features greatly influences the classification accuracy [31]. A better accuracy is obtained when the dimension of random feature increases but at the expense of a loss of training time. Its selection must optimize the couple training time/accuracy and this issue remains an open issue.

The goal of this paper is to answer both of these issues by proposing an incremental and compressible Kernel Null Discriminant Analysis. Our work is closely related to the Liu's algorithm [29]. Our contribution compared to the liu's method relies on two points: First, we propose an exact incremental scheme which meets the null LDA constraints and guarantees at each learning steps $c-1$ null projection directions where c being the number of classes. We show theoretically that the Liu's algorithm generates supplementary null projection directions at each updating steps. As a consequence, the dimension of the null space increases over time and the classification performance decreases during the learning step. Secondly, the proposed method, however, remains limited to moderate sized problems because there are faced with an ever-increasing kernel Gram matrix when dealing with streaming data. To remedy this situation, we show that the null space constraint of KNDA can be used as an efficient criterion of compression by identifying redundancy in the training data. The compression rate of the proposed method is monitored by an user-defined compression factor. The combined action of these two steps offers a significant gain in terms of time complexity and memory burden while offering comparable classification performance and learning time with state of the art incremental learning methods. Experimental evaluations both on multi-class and one class problems confirm the effectiveness of the proposed method.

The rest of the paper is organized as follows: in section 2, we introduce notation and preliminary definitions. Section 3 briefly recalls the main mathematical definitions of linear discriminant analysis (LDA) and null LDA. Section 4

covers several points: first we develop an exact incremental implementation of LDA and show the main differences with the Liu's scheme. Secondly, we present a kernel extension of our scheme and lastly, we give the initial settings for one-class learning. In section 5, we introduce a compression mechanism to deal with large streaming data sets. Section 6 details the experimental results. Finally, a conclusion ends the paper in section 7.

2. Notations and definitions

Let $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$ be a $d \times n$ input training data matrix where n and d denote the sample size and the number of features, respectively. Consider a supervised learning scenario composed of c classes, i.e. X will be accompagnied with a n label vector $\mathcal{L}_x = (l_1, l_2, ..., l_n)$ where each component $l_i \in \{1, ..., c\}$. we will note $\mathcal{L}_x^k$ the subset of labels in $\mathcal{L}_x$ belonging to the k^{th} class and $n^k = card(\mathcal{L}_x^k)$. We will note that $n = \sum_{k=1}^{c} n^k$. Now, let us introduce three centralizing operators:

$$W_n = I_n - L_n, \quad B_n = L_n - M_n \quad \text{and} \quad T_n = I_n - M_n \tag{1}$$

where I_n is the $n \times n$ identity matrix, $L_n = \left(L_n^k\right)_{k=1...c}$ is a $n \times n$ block diagonal matrix where each block L_n^k is a $n^k \times n^k$ matrix with all terms equal to $1/n^k$ and M_n is a $n \times n$ matrix with all terms equal to $1/n$. We can derive now the main ingredients of discriminant analysis (DA) in the input space $\mathcal{X}$ defined by the following scatter matrices associated to X:

$$\begin{cases} S_W^x = & XW_nW_n^\top X^\top & = X_W X_W^\top \\ S_B^x = & XB_nB_n^\top X^\top & = X_B X_B^\top \\ S_T^x = & XT_nT_n^\top X^\top & = X_T X_T^\top \end{cases} \tag{2}$$

denoting the $\boldsymbol{W}$*ithin-class scatter* matrix, $\boldsymbol{B}$*etween-class* scatter matrix and the $\boldsymbol{T}$*otal scatter* matrix, respectively. It is easy to check that a simple relation links these scatter matrices as follows: $S_T^x = S_W^x + S_B^x$. For simplicity, we assume here that the n training vectors are linearly independent implying that the ranks of the scatter matrices are given by: $rank(S_W^x) = n - c$, $rank(S_B^x) = c - 1$ and $rank(S_T^x) = n - 1$, with $rank(S_T^x) = rank(S_W^x) + rank(S_B^x)$.

3. Null Linear discriminant analysis

Linear discriminant analysis is a subspace learning method which aims to find a subspace basis $\Pi_x = (\boldsymbol{\pi}_{x1}, \boldsymbol{\pi}_{x2}, ..., \boldsymbol{\pi}_{xh})$ $(\in \mathbb{R}^{d \times h})$ of h independent column vectors which both maximizes the between-class scatter S_B^x and minimizes the intra-class scatter S_W^x through the Fisher discriminant criterion:

$$\Pi_x = \underset{\Pi_x^\top \Pi_x = I_n}{argmax} \quad trace\left((\Pi_x^\top S_W^x \Pi_x)^{-1} \Pi_x^\top S_B^x \Pi_x\right) \tag{3}$$

The solution of (3) is equivalent to solve the following generalized eigenvalue problem (GEP): $S_B^x \Pi_x = \boldsymbol{\lambda}_x S_W^x \Pi_x$. Since the rank of S_B^x is $c - 1$, there are $c - 1$ eigenvectors $(\boldsymbol{\pi}_{x1}, \boldsymbol{\pi}_{x2}, ..., \boldsymbol{\pi}_{x(c-1)})$ corresponding to the $c - 1$ largest nonzero eigenvalues $\left(\lambda_{x1}, \lambda_{x2}, ..., \lambda_{x(c-1)}\right)$. Thus, the discriminative features of LDA are computed by: $\boldsymbol{y}_i = \Pi_x^\top \boldsymbol{x}_i$,

for $i = 1, ..., n$. When the dimensionality d is very large compared to the sample size n, the scatter matrices S_W^x and S_T^x become singular and the conventional LDA defined by Eq.3 cannot be used. This situation is widespread for many modern applications such as face recognition where the dimensionality of the data is very large, most of the time larger than the sample size. This is commonly referenced as the small sample size (SSS) problem in the litterature. The *null* linear discriminant analysis has been introduced by Chen et al. in order to overcome the singularity problem of scatter matrices [6]. Following the same goal, other methods have been developed to solve it such as the regularized LDA [44], FisherFace LDA [2], Direct LDA [43], orthogonal LDA [20], QR/GSVD LDA [32][16]. The null LDA showed very attractive and highly competitive performance with respect to the previously mentioned methods [37][11][3]. Mathematically, Null LDA is equivalent to determine a set of directions $\boldsymbol{\pi}$ on which the within-class scatter vanishes ($\boldsymbol{\pi}_x^\top S_W^x \boldsymbol{\pi}_x = 0$) and the between-class scatter remains positive ($\boldsymbol{\pi}_x^\top S_B^x \boldsymbol{\pi}_x > 0$). This can be translated by the following optimization criterion [6] [17]

$$\Pi_x = \underset{\Pi_x^\top S_W^x \Pi_x = 0, \Pi_x^\top \Pi_x = I}{argmax} \; trace\left(\Pi_x^\top S_B^x \Pi_x\right) \tag{4}$$

Such Π_x is called the *null space* of LDA and in this case, the constraint $\Pi_x^\top S_W^x \Pi_x = 0$ leads the LDA's criterion to the best separability. Originally, Null LDA is solved in two stages: first, the training data set is projected on the null space of the intra-class scatter S_W^x and an eigendecomposition of the transformed between-class scatter S_B^x is performed which represent approximately a computational complexity of $O(d^2 n)$. Several batch implementations of the Null LDA have been proposed in the litterature to reduce this complexity [42] [37][11]. In this paper, we will consider the Guo's implementation [42]. In the case of an undersampled problem ($n \leq d$) and under the assumption that the training data vectors are linearly independent, Guo et al. show in [42] that there are $c - 1$ null projection directions. Since all scatter matrices are positives and $S_T^x = S_W^x + S_B^x$, the constraint $\boldsymbol{\pi}_x^\top S_B^x \boldsymbol{\pi}_x > 0$ is equivalent to $\boldsymbol{\pi}_x^\top S_T^x \boldsymbol{\pi}_x > 0$. Then we can formulate the main properties of the null LDA as the following theorem:

Theorem 1. *Let $\mathcal{N}_{S_T^x} = \left\{\boldsymbol{v} | S_T^x \boldsymbol{v} = 0, \boldsymbol{v} \in \mathbb{R}^d\right\}$ and $\mathcal{N}_{S_W^x} = \left\{\boldsymbol{v} | S_W^x \boldsymbol{v} = 0, \boldsymbol{v} \in \mathbb{R}^d\right\}$ be the null spaces of S_T^x and S_W^x, respectively, and let $\mathcal{N}_{S_T^x}^\perp$ and $\mathcal{N}_{S_W^x}^\perp$ be their orthogonal complements. Let $\boldsymbol{\pi}_x$ a projection direction that verifies both*

$$\begin{cases} \boldsymbol{\pi}_x^\top S_T^x \boldsymbol{\pi}_x > 0 & (a) \\ \boldsymbol{\pi}_x^\top S_W^x \boldsymbol{\pi}_x = 0 & (b) \end{cases} \tag{5}$$

then the following statements are verified [42]:

$$\begin{cases} \boldsymbol{\pi}_x \in \left(\mathcal{N}_{S_W^x} \cap \mathcal{N}_{S_T^x}^\perp\right) & (a) \\ dim(\mathcal{N}_{S_W^x} \cap \mathcal{N}_{S_T^x}^\perp) = c - 1 & (b) \end{cases} \tag{6}$$

The null space matrix $\Pi_x = (\boldsymbol{\pi}_{x1}, \boldsymbol{\pi}_{x2}, ..., \boldsymbol{\pi}_{x(c-1)})$ $(\in \mathbb{R}^{d\times c-1})$ *verifying conditions of Eq.5 is represented by both a set of orthonormal bases* $U_x = (\boldsymbol{u}_{x1}, \boldsymbol{u}_{x2}, ..., \boldsymbol{u}_{xn})$ $(\in \mathbb{R}^{d\times n})$ *of* $\mathcal{N}_T^{\perp}$ *and a matrix* $\boldsymbol{\Omega_x} = \left(\boldsymbol{\beta}_{x1}, \boldsymbol{\beta}_{x2}, ..., \boldsymbol{\beta}_{x(c-1)}\right)$ $(\in \mathbb{R}^{n\times c-1})$ *such as*

$$\Pi_x = U_x \Omega_x \tag{7}$$

Proofs of statements in Eq. 6 can be found in [42]. Guo et al. have shown that $\mathcal{N}_{S_T^x}^{\perp}$ is exactly defined by the space spanned by the mean adjusted data set X_T and represented by the orthonormal basis U_x. U_x can be derived from the singular value decomposition of $X_T = U_x \Sigma_x V_x^{\top}$ where U_x and V_x are the left and right orthonormal matrices of singular vectors, respectively and Σ_x is the diagonal matrix of singular values. Given U_x, Ω_x (Eq.7) is then computed by solving the following eigenvalue problem:

$$\left(U_x^T S_W^x U_x\right) \Omega_x = 0 \tag{8}$$

which amounts to determine the null space of the modified within-class scatter matrix $U_x^T S_W^x U_x$. Since $S_W^x = X_W X_W^{\top}$, solving Eq.8 is equivalent to find $\mathcal{N}_{X_W^T U_x}$,i.e:

$$X_W^T U_x \Omega_x = 0 \tag{9}$$

Eq.9 suggests two comments: first, the inner products induced in $X_W^T U_x$ can be easily kernelized and next, an incremental formulation of Eq. 9 needs both to update the eigenbasis U_x extracted from X_T and the within-scatter matrix S_W^x via X_W. In the next section, we present an incremental extension of Eq.9.

4. Incremental null discriminant analysis (INDA)

4.1. Proposed scheme

First, in order to describe the sequential nature of the data we will define $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$ the *old* data matrix already processed, $Y = (\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_l) = (\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, ..., \boldsymbol{x}_{n+l})$ the *newly* collected data matrix and $Z = (X, Y)$ the *updated* data matrix which defines X supplemented by Y. In the same way as X, the centralizing operators defined in Eq.1 will apply to Y and Z with the corresponding size in index, for instance $Y_T = YT_l$ and so on. We will note $\boldsymbol{\mu}_x = X\frac{\boldsymbol{e}_n}{n}$ the sample mean of X where e_n denotes a column vector of n ones. The same notation will be applied to Y and Z.

Now, considering that $X_T = XT_n$ has been already processed, i.e. the triplet $(U_x, \Sigma_x, \Omega_x)$ has been computed at the previous step, the goal of INDA is to compute a new triplet $(U_z, \Sigma_z, \Omega_z)$ associated to $Z_T = XT_{n+l}$ resulting from

the resolution of these two consecutive steps:

$$\begin{array}{ccll} (U_z, \Sigma_z) & \overset{IPCA}{\leftarrow} & \{(U_x, \Sigma_x), X_T, Y_T\} & (a) \\ \Omega_z & \overset{Eq.8}{\leftarrow} & \{U_z, Z_W\} & (b) \end{array} \tag{10}$$

- Step (a) of (10) is solved by using the incremental scheme for computing PCA as described in [7] and whose the main steps are summarized in the sequel. First, consider that the updated total scatter matrix $S_T^z = Z_T(Z_T)^\top$can be judiciously developped as follows [19]:

$$S_T^z = S_T^x + S_T^y + \triangle_T \triangle_T^\top \tag{11}$$

where $\triangle_T = \left(\frac{ln}{n+l}\right)^{1/2} (\boldsymbol{\mu}_x - \boldsymbol{\mu}_y)$ is a corrective term taking into account the change in mean resulting from the added data matrix Y. From the right part of Eq.11, S_T^z can be re-written in a more compact form as $S_T^z = (X_T\ Z_\Psi)\,(X_T\ Z_\Psi)^\top$ where Ψ is a new centralizing operator defined by ([19]):

$$\Psi = \begin{pmatrix} 0 & \rho \boldsymbol{e}_n \\ T_l & -\rho \boldsymbol{e}_l \end{pmatrix} \tag{12}$$

where $\rho = \sqrt{\frac{ln}{(n+l)}}$. Now, let $(U_x^r, \Sigma_x^r, V_x^r)$ be a rank-r factorization of X_T, we can derive a factorization of the updated data matrix Z_T as follows [25] :

$$Z_T = (X_T\ Z_\Psi) = (U_x^r\ J) \begin{pmatrix} \Sigma_x^r & P \\ 0 & R \end{pmatrix} \begin{pmatrix} V_x^r & 0 \\ 0 & I \end{pmatrix}^\top \tag{13}$$

where $P = (U_x^r)^\top Z_\Psi$ and (J, R) are the elements of the QR decomposition of $P^\perp = (Z_\Psi - U_x^r P)$, the orthogonal complement of P. Now considering the SVD of $\begin{pmatrix} \Sigma_x^r & P \\ 0 & R \end{pmatrix} = U_1 \Sigma_1 V_1$, we can deduce the elements of the factorization of Z_T by:

$$\begin{cases} U_z = & (U_x^r, J)\, U_1 & (a) \\ \Sigma_z = & \Sigma_1 & (b) \\ V_z = & \begin{pmatrix} V_x^r & 0 \\ 0 & I \end{pmatrix} V_1 & (c) \end{cases} \tag{14}$$

The *rank-r* singular value factorization of $Z_T = U_z^r \Sigma_z^r V_z^r$ can be recovered by considering only the column vectors of U_1 and V_1 associated to the r largest eigenvalues of Σ_1, i.e : $U_1^r = U_1(:, 1:r)$, $\Sigma_1^r = \Sigma_1(1:r, 1:r)$ and $V_1^r = V_1(:, 1:r)$.

- Step (b) of (10) amounts to solve Eq.9 on the updated within-class scatter matrix, i.e $Z_W^\top U_z \Omega_z = \mathbf{0}$, which can be

factorized by $((X\,Y)W_{n+l})^\top\,(U_x^r\;J)\,U_1\Omega_z = \mathbf{0}$. Considering that $U_1^T U_1 = I$ and after some developments, Ω_z is the solution of the following eigenvalue problem:

$$\begin{pmatrix} D_0^\top & D_3^\top \\ D_1^\top & D_2^\top \end{pmatrix} \Omega_1 = 0 \qquad \text{with} \quad \Omega_z^r = (U_1^r)^\top\,\Omega_1 \tag{15}$$

where

$$\begin{aligned} D_0 &= (U_x^r)^\top X W_{11}, & D_1 &= (U_x^r)^\top Y W_{12} \\ D_3 &= J^\top X W_{21}, & D_2 &= J^\top Y W_{22} \end{aligned} \tag{16}$$

and W_{11}, W_{12}, W_{21} and W_{22} are the subparts of the updated centralizing operator $W_{n+l} = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$. However, a deeper analysis must be addressed on the following issue: what value should be given to r? As we can see previously, the parameter r controls the level of factorization of Z_T via the truncation of its spectrum Σ_z^r. For high dimensional data sets, it can be useful to extract only a small percentage of the total variability needed to describe the data. Beside, Chin et al. in [8] use this strategy to reduce the computational cost of their incremental kernel PCA. We might be tempted at first to do the same thing. But, reader must note that if a truncation is used it changes the rank of Ω_1 and consequently the dimension of the null space resulting from Eq.15. Indeed, let us define $D = \begin{pmatrix} D_0 & D_1 \\ D_3 & D_2 \end{pmatrix}$ and since $D = Z_W^\top U_z$, it results that $rank(D) \leq min\,(rank(W_{n+l}), rank(Z), rank(U_z))$ with $rank(Z) = n + l$, $rank(W_{n+l}) = n + l - c$ and $rank(U_z) = n + l - 1$ (under linear independence assumption). Since $c > 1$, $rank(D) = rank(W_{n+l})$. From the rank-nullity theorem and since $U_z \in \mathbb{R}^{d\times(n+l-1)}$, we deduce that the dimension of the null space of D, i.e $dim(\mathcal{N}_D)$ verifies

$$dim(\mathcal{N}_D) = n + l - 1 - rank(D) = c - 1 \tag{17}$$

which fullfils the property (b) of Eq.6. Now, consider a large truncation such as $r < n + l - c$, then $rank(D) = rank(U_z^r) = r$ and consequently $dim(\mathcal{N}_D) = 0$. In conclusion, the totality of the spectrum of Z_T (non null eigenvalues) must be maintained during the learning process. In the sequel, the parameter r will be then removed.

4.2. Comparison with the Liu's scheme [29]

In [29], Liu et al. developed an incremental kernel null discriminant analysis which has strong similarities with the proposed method. But, we show in this part that the authors in [29] introduce some simplifications making their algorithm inexact.

The first simplification concerns the construction of the centralizing operator W_{n+l} when solving Eq.15. Indeed, the authors consider the old data set X and the newly added data Y as independent data sets. Under this assumption, W_{n+l} can be partitioned as a block diagonal matrix such as

$$W_{n+l} = \begin{pmatrix} W_n & 0 \\ 0 & W_l \end{pmatrix} \tag{18}$$

If this simplification seems to have a marginal impact, however it introduces a change in the value of $rank(W_{n+l})$. Without loss of generality, consider that X and Y contains c identical classes, then since W_n and W_l are not correlated, we obtain $rank(W_{n+l}) = rank(W_n) + rank(W_l) = n + l - 2c$ and then $dim(\mathcal{N}_D) = 2c - 1$. This results immediately in zeroing D_3 in Eq.16. Indeed, considering the partitionning (18), D_3 now writes $D_3 = J^\top X W_n$ instead of $J^\top(XW_{11} + YW_{21})$ and since the new basis J computed from Z_Ψ (Eq. 12) is orthogonal to the basis U_x generated by X_T then we have

$$J^\top X_T = J^\top X T_n = 0 \tag{19}$$

Reader must note that the operators T_n and W_n verify $T_n W_n = W_n$, and post-multiplying Eq.19 by W_n gives $J^\top X W_n = J^\top X_W = 0$ which implies $D_3 = 0$ in Eq.15. This result is valid only if the operators T and W are generated from the same data set. But in order to compute an exact Ω_z, the elements D_0, D_1, D_2, and D_3 of Eq.15 need to update X and Y with the updated intra class scatter W_{n+l}. In this case, $D_3 = 0$ is no longer verified.

Another simplification in [29] is to consider $U_1 = I$ in Eq.15. This implies that the authors choose to overlook the eigendecomposition of the central matrix $\begin{pmatrix} \Sigma_z^r & P \\ 0 & R \end{pmatrix}$ of Eq.13. However, to derive the new basis U_z, U_x and J must be weighted by U_1 (see Eq. 14). If this simplification seems more cost effective from a computational point of view, a danger of drift may appear, i.e errors accumulate across the update steps causing the discrepancy between the incremental solutions and the ground truth solution (computed by batch methods). Moreover, the condition $\Pi_x^\top S_W^x \Pi_x = 0$ which is the main constraint of NKDA is no longer verified.

Now since $D_3 = 0$ and $U_1 = I$, Liu et al. simplify the eigenproblem defined in Eq.15. Considering that Ω_z can be partitioned in $\begin{pmatrix} \Omega_{z1} \\ \Omega_{z2} \end{pmatrix}$ such as $D_0^\top \Omega_{z1} = 0$. The latter equality means that Ω_{z1} must lie in Ω_x which spans the null space of $D_0^\top$ since the problem $D_0^\top \Omega_x = 0$ has been solved previously. Therefore Ω_{z1} can be represented by linear combinations of Ω_x : $\Omega_{z1} = \Omega_x \alpha$ and the problem (15) can be simplified as:

$$D^\top \Omega_z = \begin{pmatrix} D_1^\top \Omega_x & D_2^\top \end{pmatrix} \begin{pmatrix} \alpha \\ \Omega_{z2} \end{pmatrix} = 0 \tag{20}$$

which is also an eigenproblem but smaller than (15). When α and Ω_{z2} are computed, the solution Ω_z is easily recovered by $\Omega_z = \begin{pmatrix} \Omega_x \alpha \\ \beta_{z2} \end{pmatrix}$. Now, let us analyse the behavior of the Liu's algorithm over two consecutive learning steps. Let $t = 0$ be the beginning of the learning process, and consider that the first data set $Z^{(t=0)} = X$ is composed of n data vectors and $c^{(t=0)} = a$ categories. Then $\Omega_z^{(t=0)}$ is computed by batch method by solving Eq.9 and $\Omega_z^{(0)} \in \mathbb{R}^{(n-1)\times(c^{(0)}-1)}$. The next step consists in solving Eq.20. Consider that the new data set Y has b categories with certain categories might be already present in X. Since D_1 and D_2 share the same centralizing operator W_l, it is easy to show that $rank(D) = rank(W_l) = l - b$ in Eq.20. We have $D_1^\top \in \mathbb{R}^{l\times(c^{(0)}-1)}$ and $D_2^\top \in \mathbb{R}^{l\times l}$, then we can conclude that the dimension of the null space generated by Eq. 20 is $dim\,(\mathcal{N}_D)^{(k=1)} = l + c^{(0)} - 1 - (l - b) = c^{(0)} + b - 1$. Thus, let $c^{(1)} = c^{(0)} + b$, $\Omega_z^{(k=1)}$ will have $c^{(1)} - 1$ columns. In other words, the common categories between X and Y will be considered as different categories. This result shows that the dimension of the null space in the Liu's scheme increases over time according the number of classes in each data chunk. In other words, the Liu's scheme considers that at each updating step, new classes are injected in the data set Y.

4.3. Introducing the kernel trick

In order to gain in flexibility, the incremental scheme for null LDA as presented in Eq.15 can be formulated in the kernel induced feature space. In this framework, let us define $\boldsymbol{\varphi} : \mathbb{R}^d \to \mathcal{F}$ a nonlinear mapping from the input space $\mathbb{R}^d$ to a high-dimensional or infinite-dimensional feature space $\mathcal{F}$ and $\Phi = (\varphi(\boldsymbol{x}_1), \varphi(\boldsymbol{x}_2), ..., \varphi(\boldsymbol{x}_n))$ the sequence of images of the input data matrix X in $\mathcal{F}$. In pratice the mapping φ is advantageously replaced by a positive definite kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that encodes the inner product in $\mathcal{F}$, instead. Thus, we can construct a kernel matrix K from for every pair of data points in X which is equivalent to the product $\Phi_x^T \Phi_x$ and where each component is defined as follows: $K(i, j) = \varphi(\boldsymbol{x}_i)^\top . \varphi(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Considering the notation introduced in section 4.1, we will define $K_x = \Phi_x^\top \Phi_x$, $K_y = \Phi_y^\top \Phi_y$ and $K_z = \Phi_z^\top \Phi_z$ the kernel Gram matrices build from X, Y and Z, respectively. Moreover, we will note $K_{xy} = \Phi_x^\top \Phi_y$ the kernel Gram matrix build from X and Y and $K_x(\boldsymbol{y}) = \Phi_x^\top \varphi(\boldsymbol{y})$ the kernel Gram vector build from X and a data $\boldsymbol{y}$. The kernel extension of our IDA as defined in Eq.10 needs first to compute the eigenspace (U_x, Σ_x) of the mapped data matrix $\Phi_x T_n$. However, as $\Phi_x T_n$ is a high dimensional matrix, (U_x, Σ_x) cannot be computed directly by SVD. From kernel SVD [38], we compute instead the SVD of the corresponding kernel Gram matrix $T_n K_x T_n = Q \triangle Q^\top$ which gives the following equalities:

$$\begin{aligned} U_x &= \Phi_x \Gamma \qquad (a) \\ \Sigma_x &= \triangle^{1/2} \qquad (b) \end{aligned} \tag{21}$$

where $\Gamma = T_n Q \triangle^{-1/2}$. Of course, only the non null eigen components of $\triangle$ and Q must be considered. The second step is to compute the couple (J, R) in Eq.13 resulting from the QR decomposition of $P_\perp$ as defined in section 4.1.

First, the matrix P (Eq. 13) is easily reformulated via the kernel trick as $P = \Gamma^\top \Phi_x^\top \Phi_z \Psi = \Gamma^\top K_{xz} \Psi$ and second, recalling that $P_\perp = \Phi_z \Psi - U_x P$ (in the feature space) and after some developments, it expresses as

$$P_\perp = \Phi_z \Theta \tag{22}$$

with $\Theta = \begin{pmatrix} \Psi_{1:n,:} - \Gamma\Gamma^\top K_{xz}\Psi \\ \Psi_{n+1:n+l,:} \end{pmatrix}$. Eq. 22 shows clearly that $P_\perp$ inherits of the high dimensionnality of Φ_z making it impossible at this stage to compute a QR decomposition. Fortunately, as previously, this problem can be overcome by computing SVD of the outer product $P_\perp^\top P_\perp$ which introduces explicitly the kernel matrix K_z. Indeed $P_\perp^\top P_\perp = \Theta^\top K_z \Theta$ and let $\Xi\Lambda\Xi^\top$ be the eigendecomposition of $\Theta^\top K_z \Theta$, the components J and R of $P_\perp$ can be easily recovered in the same spirit as in Eq. 21, which gives:

$$\begin{cases} J = & \Phi_z \Upsilon & (a) \\ R = & \Lambda^{1/2}\Xi^\top & (b) \end{cases} \tag{23}$$

where $\Upsilon = \Theta\Xi\Lambda^{-1/2}$. Reader must note that $rank(P_\perp^\top P_\perp) = rank(P_\perp)$ and $P_\perp$ is always rank deficient by at least one due to centering of Φ_z with the centralizing operator Ψ (Eq.12). Then, when computing the eigendecomposition of $\Theta^\top K_z \Theta$, we retain only eigenvectors corresponding to non zero eigenvalues. Then each part of the matrix $\begin{pmatrix} \Sigma_x & P \\ 0 & R \end{pmatrix}$ defined in Eq.13 are recovered and diagonalized as $U_1\Sigma_1 V_1$. The updated matrix of left singular vectors U_z is now given by

$$U_z = (\Phi_x \Gamma, \Phi_z \Upsilon)\, U_1 \tag{24}$$

Now, a kernel formulation of each element of Eq.15 can be then deduced as

$$\begin{aligned} D_0 &= \Gamma^\top K_x W_{11}, & D_1 &= \Gamma^\top K_{xy} W_{21} \\ D_3 &= \Upsilon^\top K_{zx} W_{12}, & D_2 &= \Upsilon^\top K_{zy} W_{22} \end{aligned} \tag{25}$$

4.4. Computational cost

The previous analysis clearly shows that Liu's scheme shows a reduced computational burden compared to the exact method. The efficiency of Liu's algorithm relies on two simplifications: first, the eigendecomposition of $\begin{pmatrix} \Sigma_x & P \\ 0 & R \end{pmatrix}$ corresponding to a complexity of $O((n+l)^3)$ is avoided and secondly, the component D_3 is set to 0 in Eq.15 by considering W_{n+l} partitioned as in Eq. 18. By noting that the first subpart of the new solution Ω_z spans the null space of D_0, then D_0 is also removed from the computation leading to solve the simplified eigenproblem of Eq. 20. In this

case, the time complexity reduces to a cost of $O(\left(\sum_{i=0}^{k} c^{(i)} - 1 + l\right) l^2)$ for the Liu's scheme instead of $O((n+l)^3)$ for the exact solution. These favorable results should not however conceal the fact that this algorithm does not solve exactly the conditions of the null LDA. We will see in the experimental section the impact of these simplifications on the classification performance of the Liu's algorithm.

4.5. One-class classification

KDA as well as NKDA are not explicitly formulated for solving a one class problem because they try to find $c-1$ discriminative projections. Then, the minimal solution of KDA is one discriminative projection for a two class problem. In one class setting, only one class is avalaible and in order to cope with the lack of a second class, a common strategy is to artificially generate representative counter examples. Based on [**? ?**], we propose to separate the target or reference class from the origin of the kernel feature space which is used here as an "artificial" counter example. Beside, the one class support vector machine (SVM) formulation is based on this principle. In the incremental setting, the initial kernel Gram matrix K at time $t=0$ is supplemented with one column of zeros and one row of zeros and when $t>0$ the kernel Gram matrix K_{xy} is supplemented with one row of zeros, as follows :

$$K_x^{oc} = \begin{pmatrix} 0 & \mathbf{0} \\ \mathbf{0} & K_x \end{pmatrix} \quad , K_{xy}^{oc} = \begin{pmatrix} \mathbf{0} \\ K_{xy} \end{pmatrix} \tag{26}$$

Of course, this artificial counter example is flagged with a negative label.

5. Compressible NKDA

At this stage, our incremental NKDA uses all training samples in the learning stage. But for very large scale datasets, our algorithm is faced two problems: a memory overhead to store the entire kernel Gram matrix and a growing computational cost corresponding to its eigendecomposition. This drawback is a major obstacle to deal with applications requiring online operations. However, when dealing with large data sets, it stand to reason that after a certain learning time, some new training samples would not contain much new information. In this case, these new data samples are considered as redundant and there is no gain to include them. Thus, the definition of a pertinent redundancy measure is a critical issue because it improves training times while having a minimal impact of the classification accuracy. In this perspective, we show in the sequel that the null space of S_W, i.e. $\mathcal{N}_{S_W}$ offers the conditions to define an efficient criterion of redundancy in the training data.

5.1. Null space based compression

Recall that the goal of null LDA is to find a mapping $\mathit{\Pi}_x = \left(\boldsymbol{\pi}_{x1}, \boldsymbol{\pi}_{x2}, ..., \boldsymbol{\pi}_{x(c-1)}\right)$ which in particular verifies

$$\boldsymbol{\pi}_{xk}^{\top} S_W^x \boldsymbol{\pi}_{xk} = 0 \quad \text{for } k = 1...c-1 \tag{27}$$

From the definition of S_W^x (Eq. 2), Eq. 27 can be simplified by $\boldsymbol{\pi}_{xk}^\top X W_n = \mathbf{0_n}$ which gives

$$\boldsymbol{\pi}_{xk}^\top (\boldsymbol{x} - X\frac{\boldsymbol{e}_n^m}{n^m}) = 0 \tag{28}$$

for a single sample $\boldsymbol{x}$ in X from the m^{th} class, where $\boldsymbol{e}_n^m$ is a binary indicator vector associated to the m^{th} class of length n. A kernel formulation of Eq.28 can be deduced by considering that $\boldsymbol{\pi}_{xk} = \Phi_x \Gamma \boldsymbol{\beta}_{xk}$ and using the kernel trick. Then from Eq.28, we define $r_k^m(\boldsymbol{x}/X)$ a measure of membership of $\boldsymbol{x}$ to the k^{th} direction of the null space generated by the old data matrix Φ_x such as

$$r_k^m(\boldsymbol{x}/X) = \boldsymbol{\beta}_{xk}^\top \Gamma^\top K_x(\boldsymbol{x}) - o_k^m(X) = 0 \tag{29}$$

where $o_k^m(X) = \boldsymbol{\beta}_{xk}^\top \Gamma^\top K_x \frac{\boldsymbol{e}_n^m}{n^m}$ denotes the m^{th} class center projected onto the k^{th} direction of the null space. Let us define $\boldsymbol{r}^m(\boldsymbol{x}/X) = \left(r_1^m(\boldsymbol{x}/X), r_2^m(\boldsymbol{x}/X), ..., r_{c-1}^m(\boldsymbol{x}/X)\right)$, then since Eq.29 is verified for all directions of the null space, we deduce that the l_2-norm $\left\|\boldsymbol{r}^m(\boldsymbol{x}/X)\right\|_2 = 0$ which is verified for each data $\boldsymbol{x}$ in X. Now, given a new data $\boldsymbol{y}$ collected in Y, the goal is to derive $\boldsymbol{r}^m(\boldsymbol{y}/X)$, i.e. the level of membership of $\boldsymbol{y}$ to the null space generated by X in the feature space. It seems reasonable to suppose that as the learning is progressing the greater the chance that new data show intrinsic similarities with the previously learned data. In other word, $\boldsymbol{r}^m(\boldsymbol{y}/X)$ decreases as the size of X increases and can be considered as a redundancy measure in the kernel induced feature space between $\boldsymbol{y}$ and the learned data set X. In the following, we propose a compression mechanism based on this assumption. Before all, recall that the updated intra-class scatter S_W^z can be recursively obtained by

$$S_W^z = S_W^x + S_W^y + \triangle_W \triangle_W^\top \tag{30}$$

where $\triangle_W$ is a corrective term, taking into account the common classes between X and Y. Let $\boldsymbol{\mu}_x^m$ and $\boldsymbol{\mu}_y^m$ be the sample means of the m^{th} class from the sets X and Y, respectively, then $\triangle_W \triangle_W^\top$ is defined by $\triangle_W \triangle_W^\top = \sum_{m \in \mathcal{L}_x \cap \mathcal{L}_y} \triangle_{W_m} \triangle_{W_m}^\top$, where $\triangle_{W_m} = \left(\frac{n^m l^m}{n^m + l^m}\right)^{1/2} \left(\boldsymbol{\mu}_x^m - \boldsymbol{\mu}_y^m\right)$. Our objective is to measure the redundancy level between Y and X with respect to the null space generated by X, i.e. by computing for each null direction $\boldsymbol{\pi}_{xk}$

$$\pi_{xk}^\top S_W^z \boldsymbol{\pi}_{xk} \tag{31}$$

Since $\boldsymbol{\pi}_{xk}^\top S_W^x \boldsymbol{\pi}_{xk} = 0$ and from Eq. 30, Eq.31 becomes

$$\pi_{xk}^\top D D^\top \boldsymbol{\pi}_{xk} \tag{32}$$

with $D = (Y, \triangle_W)$. An overall measure of redundancy between Y and X can be deduced from Eq.32 by

$$R(Y/X) = \frac{trace\left(\Pi_x^\top DD^\top \Pi_x\right)}{l} \tag{33}$$

Now, given $\boldsymbol{y} \in Y$ in the m^{th} class, from Eq.32 the redundancy between $\boldsymbol{y}$ and X computed on the k^{th} component of the null space is easily deduced by

$$r_k^m(\boldsymbol{y}/X) = \boldsymbol{\beta}_{xk}^\top \Gamma^\top K_x(y) - o_k^m(Y/X) \tag{34}$$

where $o_k^m(Y/X)$ denotes the change of m^{th} class center when Y is added to X on the k^{th} direction of the null space and defined by

$$o_k^m(Y/X) = \boldsymbol{\beta}_{xk}^\top \Gamma^\top \left((1-\alpha_m)K_{xy}\frac{\boldsymbol{e}_l^m}{l^m} + \alpha_m K_x \frac{\boldsymbol{e}_n^m}{n^m}\right) \tag{35}$$

with $\alpha_m = \left\{ \begin{array}{cc} \left(\frac{l^m n^m}{l^n + n^m}\right)^{1/2} & \text{if} \quad m \in \mathcal{L}_x \\ 0 & \text{otherwise} \end{array} \right\}$. Thus a global measure of redundancy between $\boldsymbol{y}$ and X can be defined by $\|\boldsymbol{r}^m(\boldsymbol{y}/X)\|_2$ with $\boldsymbol{r}^m(\boldsymbol{y}/X) = \left(r_1^m(\boldsymbol{y}/X), r_2^m(\boldsymbol{y}/X), ..., r_{c-1}^m(\boldsymbol{y}/X)\right)$ is a redundancy vector bringing together the redundancy of $\boldsymbol{y}$ recorded on each direction of the null space of X. The average redundancy of the m^{th} class beetwen the new data chunk Y and the old data X is then easily deduced

$$\overline{r}^m(Y/X) = \frac{1}{l_m} \sum_{\boldsymbol{y} \in \mathcal{L}_y^m} \|\boldsymbol{r}^m(\boldsymbol{y}/X)\|_2 \tag{36}$$

It seems obvious that more the learning is progressing, more the average redundancies for each class tend to decrease meaning that similarities grow up between old and new data. This trend leads us to define a sparsity criterion which uses thresholding on the redundancies computed for each new collected data. More precisely, given $\left\{\overline{r}^j(Y/X^{(t=0)})\right\}_{j=1}^c$ an initial set of average redundancies computed for each class at step $t = 0$, and consider a new data $\boldsymbol{y}$ collected in the m^{th} class a the t^{th} learning step, then $\boldsymbol{y}$ will be removed from the learning process if its redundancy $\|\boldsymbol{r}^m(\boldsymbol{y}/X)\|_2$ is lower than a specified positive threshold ν $(0 < \nu < 1)$ such that

$$\frac{\left\|\boldsymbol{r}^m(\boldsymbol{y}/X^{(t>0)})\right\|_2}{\overline{r}^m(Y/X^{(0)})} < \nu \tag{37}$$

The initial set of average redundancies $\left\{\overline{r}^j(Y/X^{(0)})\right\}_{j=1}^c$ for each class has been computed during the cross validation step. Since the average redundancy per class is tending to decrease, the compression starts when a new data records a redundancy with X lower than a certain threshold which is proportional to the average redundancy of its class computed at the first step. The parameter ν allows the user to regulate the compression rate during the learning

process and we will study its impact on the classification in the experimental section. If it is reasonable to assume that $\|\boldsymbol{r}^m(\boldsymbol{y}/X)\|_2$ is decreasing when the learning grows up, the following section tries to provide some mathematical evidence.

5.2. Redundancy measure: property

The proposed proof assumes that the measure of redundancy has the same behavior for each class and consequently, we will omit the exponent m denoting the class number in the sequel. In order to prove that the l_2-norm $\|\boldsymbol{r}(\boldsymbol{y}/X)\|_2$ (Eq.34) is a mesure of redundancy which decreases as the learning is progressing, we consider the following scenario: let us define $\boldsymbol{x}, \boldsymbol{y}_0$ and $\boldsymbol{y}$ three data vectors verifying

1. $\boldsymbol{x} \in X, \boldsymbol{x} = \underset{\boldsymbol{x}_i \in X}{min} \|\boldsymbol{x}_i - \boldsymbol{y}\|_2$
2. $\boldsymbol{y}_0, \boldsymbol{y} \in Y, \|\boldsymbol{x} - \boldsymbol{y}\|_2 \geq \|\boldsymbol{y_0} - \boldsymbol{y}\|_2$

The first condition means that $\boldsymbol{x}$, a learned data in X, is the nearest to $\boldsymbol{y}$ among all $\boldsymbol{x}_i \in X$ and the second condition means that the intermediate new data $\boldsymbol{y}_0$ is nearest to $\boldsymbol{y}$ than $\boldsymbol{x}$ to $\boldsymbol{y}$. This scenario is a snapshot at time t of the learning process. If we consider that $\boldsymbol{y}_0$ will be the next data learned at time $t+1$, then we want to prove that

$$\left\|\boldsymbol{r}(\boldsymbol{y}/X^{(t)})\right\|_2 \geq \left\|\boldsymbol{r}(\boldsymbol{y}/X^{(t+1)})\right\|_2 \tag{38}$$

with $X^{(t+1)} = X^{(t)} \cup \{\boldsymbol{y}_0\}$. Or in other words, the redundancy level between $\boldsymbol{y}$ and X increases when a new data close to $\boldsymbol{y}$ is included in the learning process. First, in the feature space the inequality between distances (condition 2) remains true , i.e.

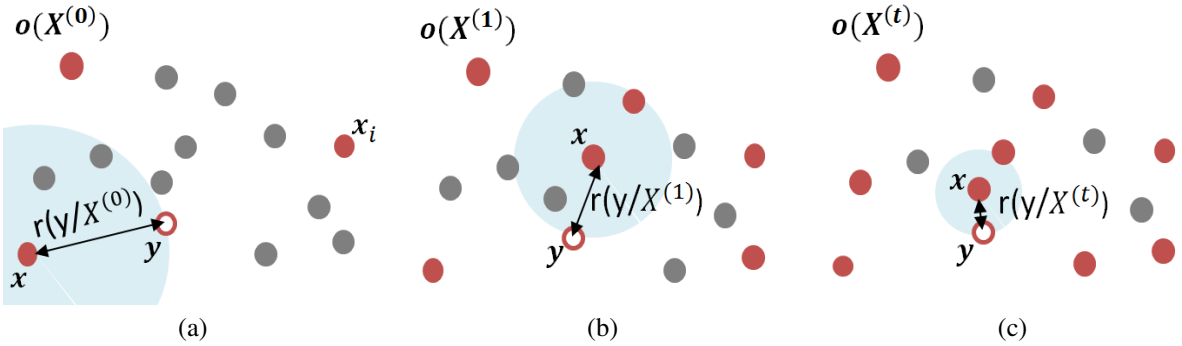
$$\|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})\|_2 \geq \|\varphi(\boldsymbol{y_0}) - \varphi(\boldsymbol{y})\|_2 \tag{39}$$

Let $\boldsymbol{\pi}_{x_k} = \Phi_x \Gamma \boldsymbol{\beta}_{xk}$, the k^{th} direction of the null space Π_x computed at time t, multipling both sides of Eq.39 by $\boldsymbol{\pi}_{k_i}$ and considering that the operator norm verifies $\|A\|\,\|\boldsymbol{v}\| \geq \|A\boldsymbol{v}\|$ (where A is a matrix and $\boldsymbol{v}$ is a vector), Eq.39 can be written as

$$\left\|\boldsymbol{\pi}_{x_k}^\top(\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y}))\right\|_2 \geq \left\|\boldsymbol{\pi}_{x_k}^\top(\varphi(\boldsymbol{y_0}) - \varphi(\boldsymbol{y}))\right\|_2 \tag{40}$$

Using the kernel trick, Eq.40 expresses as

$$\left\|\boldsymbol{\beta}_{xk}^\top \Gamma^\top K_x(\boldsymbol{x}) - \boldsymbol{\beta}_{xk}^\top \Gamma^\top K_x(\boldsymbol{y})\right\|_2 \geq \left\|\boldsymbol{\beta}_{xk}^\top \Gamma^\top K_x(\boldsymbol{y}_0) - \boldsymbol{\beta}_{xk}^\top \Gamma^\top K_x(\boldsymbol{y})\right\|_2 \tag{41}$$



(a) (b) (c)

Figure 1: Illustration of the decreasing trend of $\|\boldsymbol{r}\left(\boldsymbol{y}/X^{(t)}\right)\|$.

Let us define $o_k(X^{(t)})$ the class center projected onto the k^{th} component of the null space at time t and integrating it in both sides of Eq.41, Eq.41 simplifies as

$$\left\|r_k\left(\boldsymbol{y}/X^{(t)}\right)\right\|_2 \geq \left\|r_k\left(\boldsymbol{y}_0/X^{(t)}\right) - r_k\left(\boldsymbol{y}/X^{(t)}\right)\right\|_2 \tag{42}$$

Now, consider that y_0 is learned, i.e. $X^{(t+1)} = X^{(t)} \cup \{\boldsymbol{y}_0\}$, the old class center $o_k(X^{(t)})$ is updated and becomes $o_k(X^{(t+1)})$. The immediat consequence is that $r_k\left(\boldsymbol{y}_0/X^{(t+1)}\right) = 0$ and Eq.42 verifies now

$$\left\|r_k\left(\boldsymbol{y}/X^{(t)}\right)\right\|_2 \geq \left\|r_k\left(\boldsymbol{y}/X^{(t+1)}\right)\right\|_2 \tag{43}$$

This result holds for each direction $k = \{1, ..., c-1\}$ of the null space and Eq.38 is verified. Figure 1 explains graphically the behavior of $\|\boldsymbol{r}\left(\boldsymbol{y}/X^{(t)}\right)\|$, the redundancy measure of a new data $\boldsymbol{y}$ (data circled in red in Figures 1) during the 1^{th}(Fig. 1.a), 2^{th} (Fig. 1.b) and t^{th} (Fig. 1.c) learning step. Data filled in red denote the old data already learned and data filled in grey represent the remaining data. Data already learned are projected in the same vector $\boldsymbol{o}(X^{(t)}) = \left(\boldsymbol{o}_1(X^{(t)}), \boldsymbol{o}_2(X^{(t)}), ..., \boldsymbol{o}_{c-1}(X^{(t)})\right)$ which defines the class center in the null space at time t. As the learning is progressing, more data are learned, or in other words they are represented by the same reference point in the null space. As a consequence, the distance measured by $\|\boldsymbol{r}\left(\boldsymbol{y}/X^{(t)}\right)\|$ between $\boldsymbol{y}$ and $\boldsymbol{o}(X^{(t)})$ decreases irremediably (see Figure 1.c).

6. Experiments

6.1. Data sets and algorithms

Table I provides a detailed description of the datasets which will be used to evaluate our method. These data sets involve several areas such as image recognition, document recognition and URL adresses. Most of them have several categories and will be used for the multi-class learning experiments. One-class learning experiments will be conducted by selecting only one target category among others. Online learning will be simulated as follows: the majority of the

Table 1: Data description

Data sets	Types	Classes	Dim	Train	Test	Source
MNIST	g.l. images (28×28)	10	784	60000	10000	[24]
CIFAR-10	color images (32×32)	10	3072	50000	10000	[22]
YaleFaceB	g.l. image (64×64)	38	4096	1900	514	[14]
Ayahoo	concept images	12	4096	988	1249	[13]
RCV1 (Reuters)	documents	65	18933	2347	1249	[26]
URL	URL addresses	2	3231961	10137	400	[30]

Table 2: Algorithm description

Algorithms	Authors	Learning type	Parameters	Incr.	Comp.
ICNKDA	proposed method	MC/OC	2	$\surd$	$\surd$
INKDA	Liu et al. [29]	MC/OC	1	$\surd$	$\times$
SRKDA	Cai et al [4]	MC	2	$\surd$	$\times$
AKDA	Gkalelis et al [15]	MC	2	$\surd$	$\times$
FKDA	Wang et al [40]	MC	2	$\surd$	$\surd$
SoDA	Weihong et al [27]	MC	1	$\surd$	$\surd$
IOC-SVM	Laskov et al [23]	OC	2	$\surd$	$\times$
ISVDD	Jiang et al [18]	OC	2	$\surd$	$\surd$
IOC-GPR	Kemmler et al [21]	OC	2	$\surd$	$\times$

data set is stored in hard drive and a chunk of data will be extracted and stored in the random access memory at each learning step. Table II presents the different methods which will be used as a basis for comparison. All these methods work incrementally or recursively (see column *Incr.* in Table II) but few of them have a compression mechanism (see column *Comp.* in Table II). All these methods have been implemented in matlab and experiments are conducted on a Workstation with Intel-7 2.5 GHz, 8G RAM, 64-bit Windows system.

6.2. Evaluation and comparison

6.2.1. Comparative study with the Liu's scheme

In this experiment, we compare specifically the two schemes on several points: a) the accuracy in approximating batch KNDA solution. b) the behavior with respect to the batchsize l. c) the empirical time complexity. Several performance measures will be considered to analyse the behavior of each method: the area under the curve (AUC) to measure the classification performance, the null direction error (NDE) which measures the l2-norm of the difference between the score vectors returned by the batch method and the incremental method, the training time (TIME) and the dimension of the null space (DIM). The MNIST data set will be used to conduct the evaluation. Two learning scenarios will be analysed: Multi-class learning (MCL) and one-class learning (OCL). MCL will be studied from 100 training images per classes while OCL will be based on 1000 training images of the digit 4, the other digits will be considered as anomalies. The test data set will be composed of 100 images of each class. In both case, the Gaussian kernel of scale parameter σ is choosen to conduct the experiments and a cross validation step is used to select the values of σ

Table 3: Multi-class learning: comparative study.

methods	l	AUC	NDE	Times (sec)	DIM
exact-$\sigma = 4$	10	99.54	0	63.30	9
	30	99.54	0	23.86	9
	50	99.54	0	15.70	9
Liu-$\sigma = 6$	10	90.30	7.22	9.11	615
	30	95.54	4.50	2.28	313
	50	98.05	3.26	1.23	197

Table 4: One-class learning: comparative study.

methods	l	AUC	NDE	Times (sec)	DIM
exact- $\sigma = 3$	10	95.11	0	51.5	1
	30	95.11	0	18.7	1
	50	95.11	0	11.6	1
Liu- $\sigma = 15$	10	88.80	1.58	3.90	96
	30	91.45	0.8	1.22	33
	50	93.74	0.56	0.82	20

corresponding to the best AUC.

Table III and Table IV summarize the results obtained from the exact method and the Liu's method. First in both scenario, we observe that the exact method converges towards the ground truth null direction projections (returned by the batch method) since its *null direction error* (NDE) reaches 0 at the end of the learning (4-th column of Table III and IV). Conversely, the Liu 'scheme systematically shows a non-zero NDE confirming that is a wrong approximation of the batch KNDA while the exact method reaches the expected solution. Moreover, results recorded by the exact method are independent of the batchsize l while it is not the case for the Liu's method. Concerning the classification performance, we remark that the exact method systematically outperforms the Liu's scheme. The AUC value (3-th column in Table III and IV) recorded by the exact method remains constant no matter what the batchsize value while the AUC value returned by the Liu's scheme is very sensitive to the batchsize value. As demonstrated previously, the dimension of the null space increases at every updating step for the Liu's scheme and its growth rate depends on the batchsize l. Conversely, the exact method verifies the main condition of KNDA (6-th column of Table III and IV). Lastly, as expected the Liu's scheme shows a faster training time that the exact method (5-th column of Table III and IV) but at the expense of the classification performance.

Figure 2 confirms graphically this analysis by illustrating the behavior of the two algorithms with respect to the dimension of the null space (DIM, first row of the figure 2) and the null direction error (NDE, second row of the figure 2) and for different values of the batchsize l. As we can observe on Figure 2.a, the value of DIM computed by the

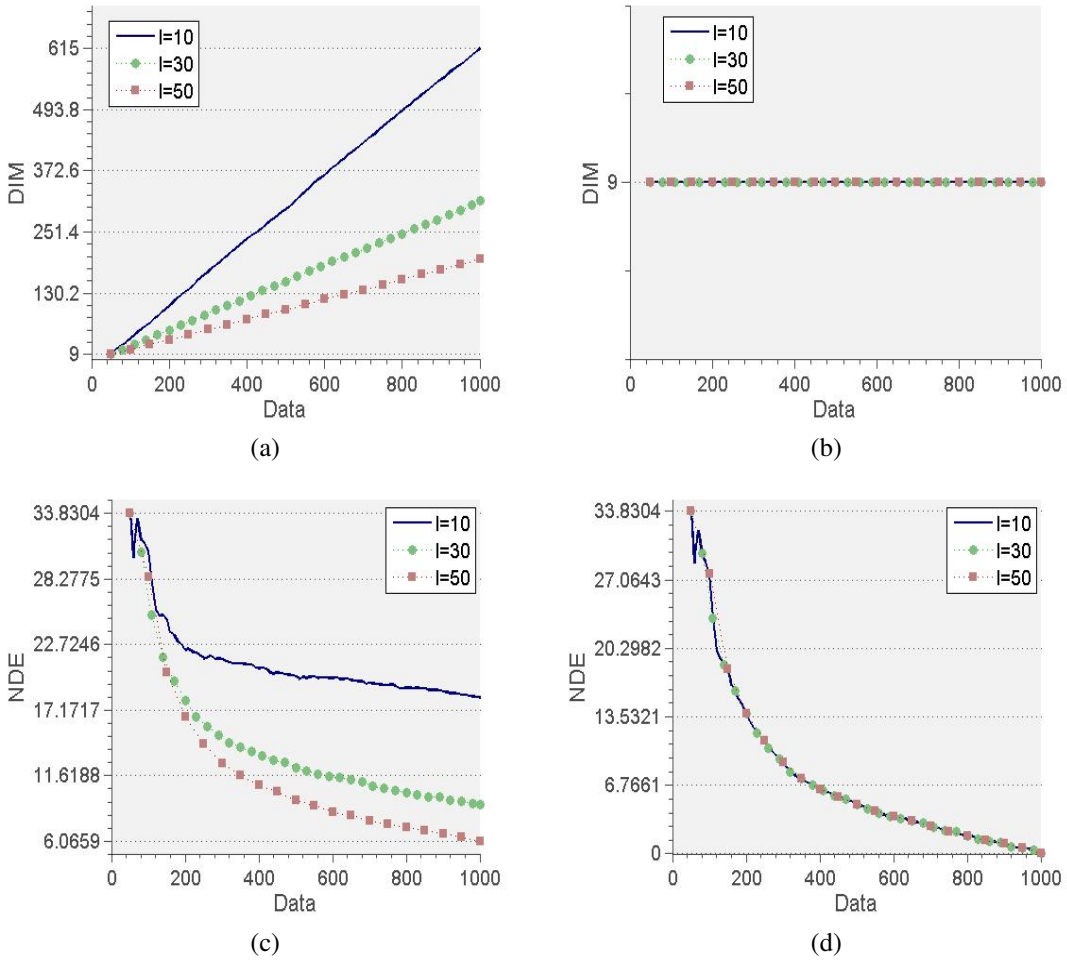


Figure 2: Our method vs Liu's method. The first row shows the evolution of the dimension of the null space (left:Liu's scheme, right: our scheme). The second row shows the evolution of the null direction error (left: Liu's scheme, right:our scheme).

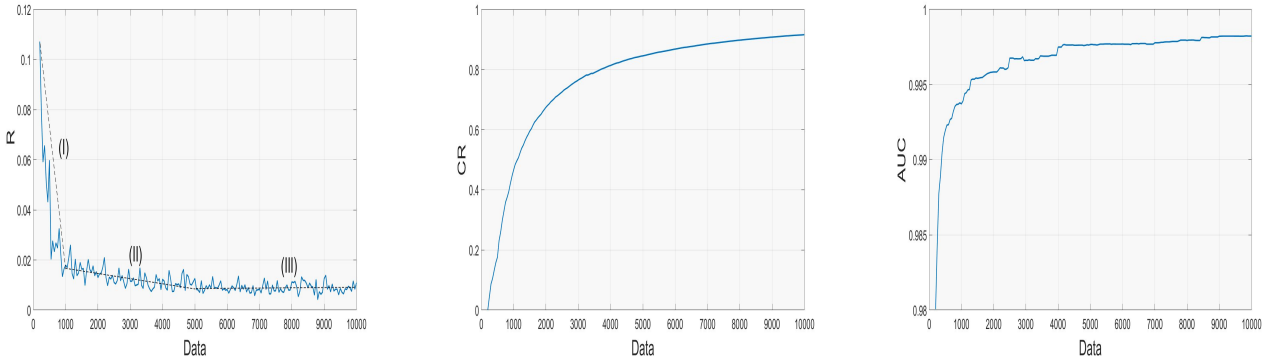
Liu's scheme increases at each updating step confirming the result of Eq.**??** while on Figure 2.b it remains constant for the exact method verifying the rank of S_B, i.e. $C-1$. Figure 2.c shows that, whatever the batchsize value l, the Liu's scheme does not converge to the true solution (NDE>0) while the exact method converges towards zero NDE (Figure 2.d).

6.2.2. *Influence of the compression rate*

In this part, we analyse the impact of the compression strategy with respect to the classification performance (AUC) and the training time (Times). The MNIST data set is used for the experiment. We maintain the experimental setup as described in the previous section (see tables III and IV). The compression parameter ν varies in $[0, 0.15, 0.25, 0.35, 0.45]$ where $\nu = 0$ means "no compression". Table V summarizes the results of this experiment. From a general point of view, these results show that the redundancy level is high in the MNIST data set. Indeed, for $\nu = 0.45$, corresponding to a high compression rate, we record a 0.6% decrease in AUC for the multi-class problem and a 0.3% decrease in

Table 5: Influence of the compression parameter ν.

Learn. mode	ν	AUC	Times (sec)	CR
OCL	0	95.11	11.60	0
	0.15	95.23	1.80	62.0
	0.25	95.62	1.04	72.5
	0.35	95.44	0.68	79.4
	0.45	94.52	0.30	85.1
	Liu's sch.	93.74	0.82	0
MCL	0	99.54	15.78	0
	0.15	99.52	6.62	29.5.0
	0.25	99.45	3.98	46.9
	0.35	99.29	2.07	61.3
	0.45	98.97	1.20	71.9
	Liu's sch.	98.05	1.23	0



(a) (b) (c)

Figure 3: Illustration of the change in $R(Y/X)$ (Eq.33) during the multi-class experiment (MNIST data set). Changes in the parameter R(Y/X) (a), the compression rate CR (b) and the AUC value (b).

AUC for the one-class problem. In the same way, the training time are drastically reduced: a decrease by a factor of ~15 for the multi-class problem and ~40 for the one-class problem.

6.2.3. Global redundancy: analysis of $R(Y/X)$

Figure 3.a illustrates the evolution of the parameter $R(Y/X)$ (Eq.33) computed by the proposed method for a compression rate $\nu = 0.3$. In this experiment we use 10000 training images of the MNIST data base (multi-class learning). Clearly, we observe that $R(Y/X)$ is subjected to three gradients of decrease: first, R initiates a significant decrease over the first 1000 images (I), next a slower decrease up the 5000^{th} image and lastly reaches a constant level over the remaining samples. This variation means that redundancy is very low during the initial learning stages, the dictionnary is being filled and next redundancy increases and finally stabilizes. Figure 3.b shows the variation of the compression rate, CR, during the learning step. Like $R(Y/X)$, the variation of CR shows 3 slopes of increase. At the end of the learning, most of the data are removed reducing the increase of CR. In this experiment the compression rate reaches

91.6%. Lastly, Figure 3.c shows the variation of AUC during the learning process and reader can observe that the AUC value increases monotonically reaching 99.82 at the end of the learning stage.

6.2.4. Multi-class learning and comparative study

We propose here to extend our analysis by comparing our method with recent incremental extensions of kernel discriminant analysis (see Table II) such as spectral regression KDA (ISRKDA, [4]), accelerated KDA (IAKDA, [15]), factorization-free KDA (IFKDA, [40]) and null KDA (INKDA, [29]). We will name our method ICNKDA, acronym of incremental and compressible null KDA. A Gaussian Kernel of bandwidth σ is used for all the methods. A regularization parameter β is needed for ISRKDA and IAKDA to solve the singularity problem associated to the inversion of the kernel matrix. We tune σ and β to achieve best testing performance for the proposed methods which are reported in the 3-th and 4-th columns of table VI. Moreover, the behavior of ICNKDA will be analysed without compression ($\nu = 0$) and with compression ($\nu = 0.35$). Since most of the competitors are not equipped with a compression mechanism, we will limit the size of the training sample to 1000. Performance of the methods will be studied according to the AUC values in percent and the training times in secondes. Reader must note that the training time used to build and update the kernel matrix is not reported because this is a common step for all the methods, only the training time used to update the model will be recorded.

From Table VI, several comments can be pointed out:

a) For all the studied data sets, ICNKDA without compression reaches better or equivalent classification performances than concurents. As expected, training times consumed by the proposed approach without compression are by far the worst. The use of a compression strategy in ICNKDA with an appropriate choice of the compression parameter ν provides a classication level and training time truly competitive with others competitors. In particular, we observe that the training times for ICNKDA$_{0.35}$ are faster than those of ISRKDA and IAKDA. It shoud be noted that values in brackets in the column *AUC* correspond to the compression rate recorded at the end of the learning process of ICNKDA$_{0.35}$.

b) Despite very good training times, INKDA performs less well and these results confirm a loss of accuracy during the learning process. An illustration of this trend is observed on Figure 4 with the YALE-B data set by comparing the change in AUC for both methods. Three values of the kernel parameter σ are used in this experiment. Whatever the value of the kernel width, after a period of growth, the Liu's scheme records a noticeable performance loss (dashed lines in Figure 4) while our method shows a continuous increase of the AUC value over the learning (solid lines in Figure 4). This example is not a particular case that might be explained by a wrong choice of the kernel parameter but it reflects a general trend of the Liu's scheme. Indeed, despite a cross validation step to select the best value of σ, the Liu's scheme is not optimal to compute incrementally the KNDA's solution.

Table 6: Multi-class learning: comparative results

Data	Methods	σ	β	Times (sec)	AUC
$MNIST_{10}$	ICNKDA$_0$	4	-	15.78	99.54
	ICNKDA$_{0.35}$	4	-	2.07	99.29(CR=61.3)
	INKDA	4	-	1.23	98.05
	ISRKDA	2	1e-3	4.30	99.6
	IAKDA	2	2	8.07	99.25
	IFKDA	2	-	0.44	96.15
$CIFAR_{10}$	ICNKDA$_0$	8	-	15.26	77.94
	ICNKDA$_{0.35}$	8	-	2.83	75.01(CR=56.0)
	INKDA	8	-	1.01	65.27
	ISRKDA	8	1e-3	4.47	75.10
	IAKDA	8	10	7.13	77.53
	IFKDA	8	-	0.38	64.63
$YALE-B_{38}$	ICNKDA$_0$	10	-	16.48	98.89
	ICNKDA$_{0.35}$	10	-	3.33	98.24(CR=45.7)
	INKDA	20	-	4.08	81.48
	ISRKDA	10	1e-3	5.03	99.16
	IAKDA	10	2	6.72	96.06
	IFKDA	10	-	2.92	60.2
$AYAHOO_{'12'}$	ICNKDA$_0$	2	-	14.26	98.27
	ICNKDA$_{0.35}$	2	-	3.88	97.39(CR=41.4)
	INKDA	1	-	2.55	97.25
	ISRKDA	2	1e-3	4.06	97.78
	IAKDA	2	2	7.06	98.35
	IFKDA	2	-	0.46	95.48

c) The factorization mechanism used by IFKDA provides the best training times among the competitors, but using only the class centers as discriminative information seem to be not enough to perform as well as the other competitors.

6.2.5. One-class learning and comparative study

To complete our study, we propose here to compare our approach with different incremental one-class learning methods. In this experiment, the learning process is based on a single target class and the others classes are considered as anomalies. The competitors include the incremental one class svm (IOCSVM, [5, 23]), incremental support vector data description (ISVDD, [39, 18]), incremental one class gaussian process regression (IOCGPR, [21]) and the incremental kernel null discriminant analysis in its one class version (INKDA, [29]).

For all the methods a Gaussian kernel of scale σ is used. Additional parameters are needed in the experiments. The solutions obtained by IOCSVM and ISVDD are influenced by a regularization parameter $C = \frac{1}{N\nu}$ with ν is related to the outlier ratio in the data set. If we consider that no outliers affect the training data set, C can be set to 1 which corresponds to a very small outlier ratio ($\nu \sim \frac{1}{N}$). Concerning IOCGPR, we use the predictive variance criterion for

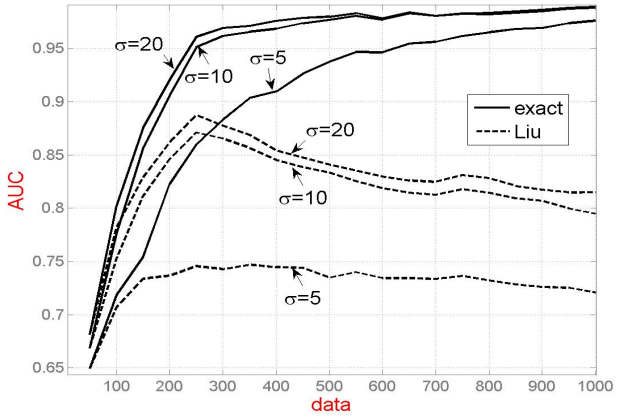


Figure 4: Multi-class learning: evolution of the AUC values for the YALEB data set (see text).

Table 7: one-class learning: comparative study.

Data	Methods	σ	Times (sec)	CR	AUC
$MNIST_{'4'}$	ICNKDA$_0$	3	11.60	0	95.12
	ICNKDA$_{0.35}$	3	0.68	79.4	95.44
	ISVDD	4	1.69	93.9	95.44
	IOCSVM	8	6.31	0	95.45
	IOCGPR	6	3.90	0	94.36
	INKDA	15	0.82	0	93.74
$CIFAR_{'deer'}$	ICNKDA$_0$	2	11.61	0	76.64
	ICNKDA$_{0.35}$	2	2.12	59.8	75.31
	ISVDD	4	88.1	38.1	75.28
	IOCSVM	4	16.5	0	75.28
	IOCGPR	4	3.91	0	75.26
	INKDA	20	2.44	0	72.39
$RCV1_{'earn'}$	ICNKDA$_0$	5	6.36	0	86.64
	ICNKDA$_{0.35}$	3	0.67	85.5	83.38
	ISVDD	15	1.84	97.9	86.06
	IOCSVM	15	1.17	0	86.10
	IOCGPR	10	1.32	0	86.09
	INKDA	20	0.76	0	81.73
$URL_{'normal'}$	ICNKDA$_0$	4	17.86	0	82.4
	ICNKDA$_{0.35}$	2	11.88	61.1	83.73
	ISVDD	4	248.36	93.9	81.2
	IOCSVM	4	115.88	0	81.2
	IOCGPR	4	5.74	0	81.9
	INKDA	1	9.09	0	75.2

deriving an OCC score but we replace the histogram intersection kernel with the radial basis function (see [21]). The result of IOCGPR is dependent with the variance of the additive-noise σ_n which is set to 0.01 in all the experiments. Lastly, both INKDA and ICNKDA use the strategy presented in section 4.5 to build the one-class kernel Gram matrix. As in the multi-class experiment, the performance of ICNKDA will be studied without ($\nu = 0$) and with compression ($\nu = 0.35$).

Table VII shows the results obtained on 4 data sets: MNIST, CIFAR, RCV1 and URL. The target classes are outlined in the first column. For each experiment, the training data set is composed of 1000 data randomly selected in the target class and the test set is built from at most 100 data samples extracted from each class. Each competitor uses the Gaussian function to build the kernel Gram matrix and its bandwidth σ is selected by cross validation (see table VII).

From Table VII, we can notice that ICNKDA shows similar classification results with recent incremental one-class learning approaches. In most of the cases, by choosing a moderated compression factor, the classification performance remains competitive with other OCL methods. As previously, it is obvious that the compression strategy provides substancial gain in training time. Despite fast training times, we observe that the Liu's scheme (INKDA) provides the worst classification results and confirms the previous comments.

Concerning one-class SVM methods such as ISVDD and IOCSVM, the results are very similar. ISVDD has the advantage of using only existing support vectors and the new data vectors to recompute the boundary while IOCSVM must use all the training data vectors at each updating step. By ignoring interior points of the boundary, ISVDD is intrinsically equipped with a compression mechanism. But, we observe that training time returned by ISVDD is sensitive to the dimensionality of the data. Precisely, training time will be very fast for low dimensional data but will increase drastically for high dimensional data (see the results for the URL data set in Table VII). Indeed, the repeated access to the expanding, shrinking and bookkeeping steps for maintaining the KKT conditions can be time-consuming for high dimensional data sets. Based on the same strategy, IOCSVM is even less competitive thereby confirming the results given in [18].

IOCGPR provides similar classification results that other OCL methods with competitive learning times. However, without compression mechanism of the kernel Gram matrix, its use will be limited to moderate sized data sets.

6.2.6. Large scale data sets

As previously, we conduct two experiments: Large scale multi-class learning and large scale one-class learning.

For the multi-class learning experiment, 50000 gray level images of MNIST and 50000 color images of CIFAR will be used in the training phase. Moreover, we compare our method with IFKDA [40] and SoDA [27]. Among the randomized LDA and KDA [41], SoDA is an online algorithm which recursively updates its solution and does not depend of the knowledge of the size of the training data set. IFKDA is also an online algorithm which does not need

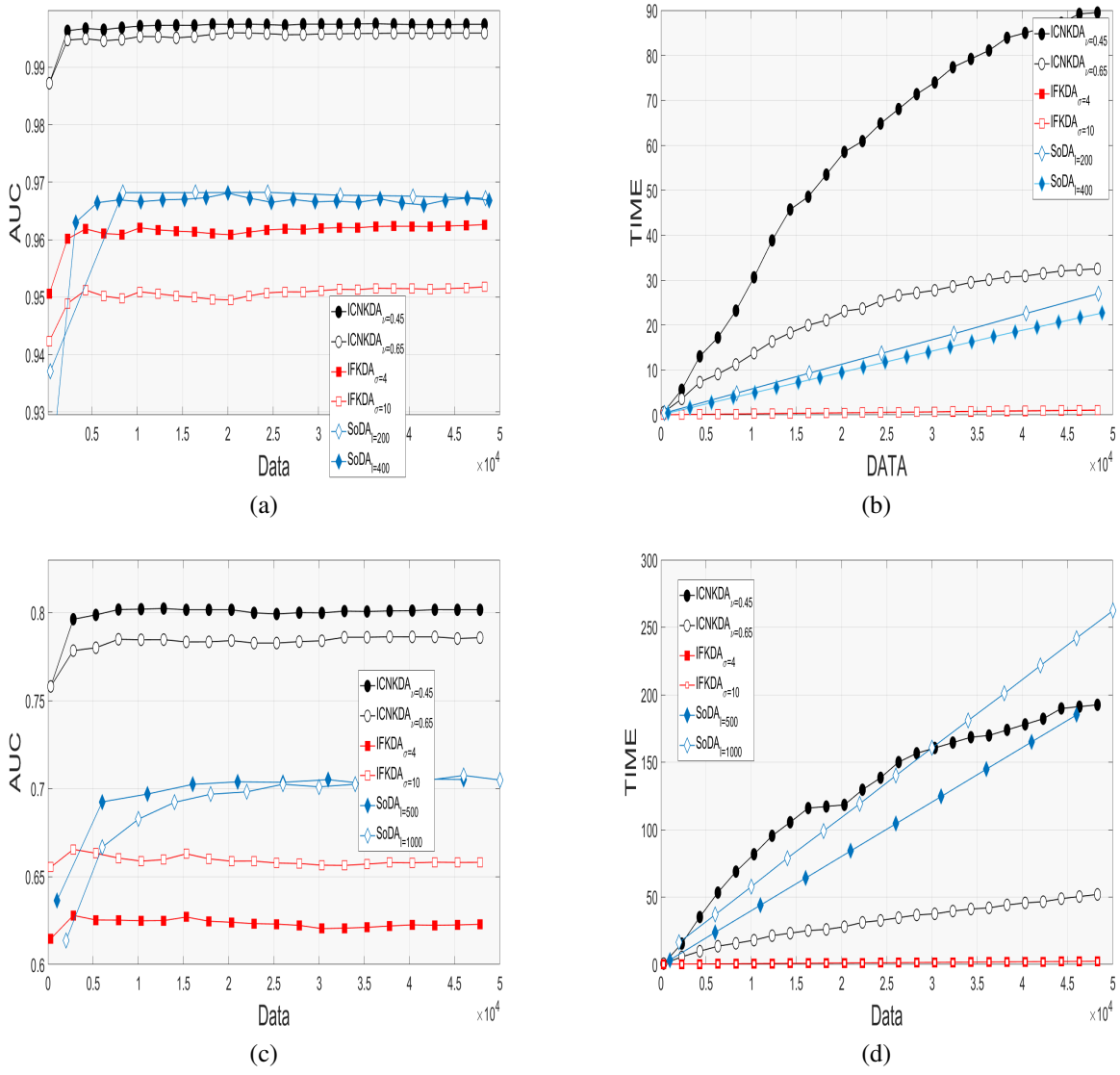


Figure 5: Large scale multi-class learning. First row: evolution of the AUC values (a) and training times (b) recorded by ICNKDA (circles), SoDA (diamonds) and IFKDA (squares) from 50000 MNIST images. Second row: evolution of the AUC values (a) and training times (b) recorded by ICNKDA (circles), SoDA (diamonds) and IFKDA (squares) from 50000 CIFAR-10 images.

to store the kernel Gram matrix of the training data. Performance of SoDA is analysed with respect to the size of the sketch matrix, noticed l, used to capture the main data variations. Different values of l are used in these experiments. We test SoDA with $l = 200, 400$ for MNIST (low dimensional data set) and $l = 500, 1000$ for CIFAR-10 (high dimensional data sets). IFKDA will be studied with respect to two values of its kernel parameter $\sigma = 4, 10$. The proposed method ICNKDA is parametrized by the kernel width σ and the compression factor ν. In this experiment, we select $\sigma = 4$ for MNIST and $\sigma = 8$ for CIFAR-10 and ν will be in the range $\{0.45, 0.65\}$ for both data sets. Figure 5 shows the evolution of the AUC values and the training times recorded by all the methods. In terms of classification performance, Figures 5.a and 5.c show clearly that the proposed method (black curves with filled and empty circles) outperforms all of its competitors on the MNIST (fig. 5.a) and CIFAR-10 (fig. 5.c) data sets. Moreover, as expected, we observe that the compression factor ν influences both the classification performance and the training time, with a different level depending on the complexity (variability) of the data sets. A trade-off between classification performance and learning time should be considered. Off all the methods, IFKDA records by far the best training times but shows the worse classification results (red curves with filled and empty squares). Training times returned by SoDA (blue curves with filled and empty diamonds) can be substantial with the dimensionnality of the data. For instance, SoDA shows the worse training times for the CIFAR-10 data set. Table VIII shows the final results returned by all the methods. We observe that ICNKDA removes a large quantity of redundant information (see the last column of Table VIII) and despite this, ICNKDA performs very well and shows the best results.

For the one-class learning experiment, three large scale data sets are analysed: MNIST, CIFAR-10 and URL. 5842 images of the digit 4 from MNIST are used as target images, 4904 images of deer from CIFAR-10 and 10000 benign URL adresses as target from the URL data set. We compare our method with ISVDD which has the advantage to include implicitly a compression mechanism. Figure 6 presents the evolution of the AUC values and the training times recorded by the two methods. Clearly, we observe that ICNKDA records equivalent classification results with ISVDD for the MNIST data set but performs better for high dimensional data sets such as CIFAR-10 and URL. As previously observed in Table VII, ISVDD is time-consuming for high dimensional data sets while ICNKDA shows faster training times. As expected, the classification performance of ICNKDA is slightly affected when the compression factor changes. Table IX shows the final results obtained by the two methods and reports the following points: first, with equivalent compression rates, the two methods return similar classification results for the MNIST experiment. However, when the dimensionality of the data significantly increases ICNKDA outperforms ISVDD. Indeed, the experiments on the CIFAR-10 and URL data sets seems to confirm this trend.

7. Conclusions

This paper introduces an incremental scheme of KNDA for solving large scale multi-class learning and one-class learning problems. In order to reduce the size of the kernel Gram matrix which is the main drawback of most kernel

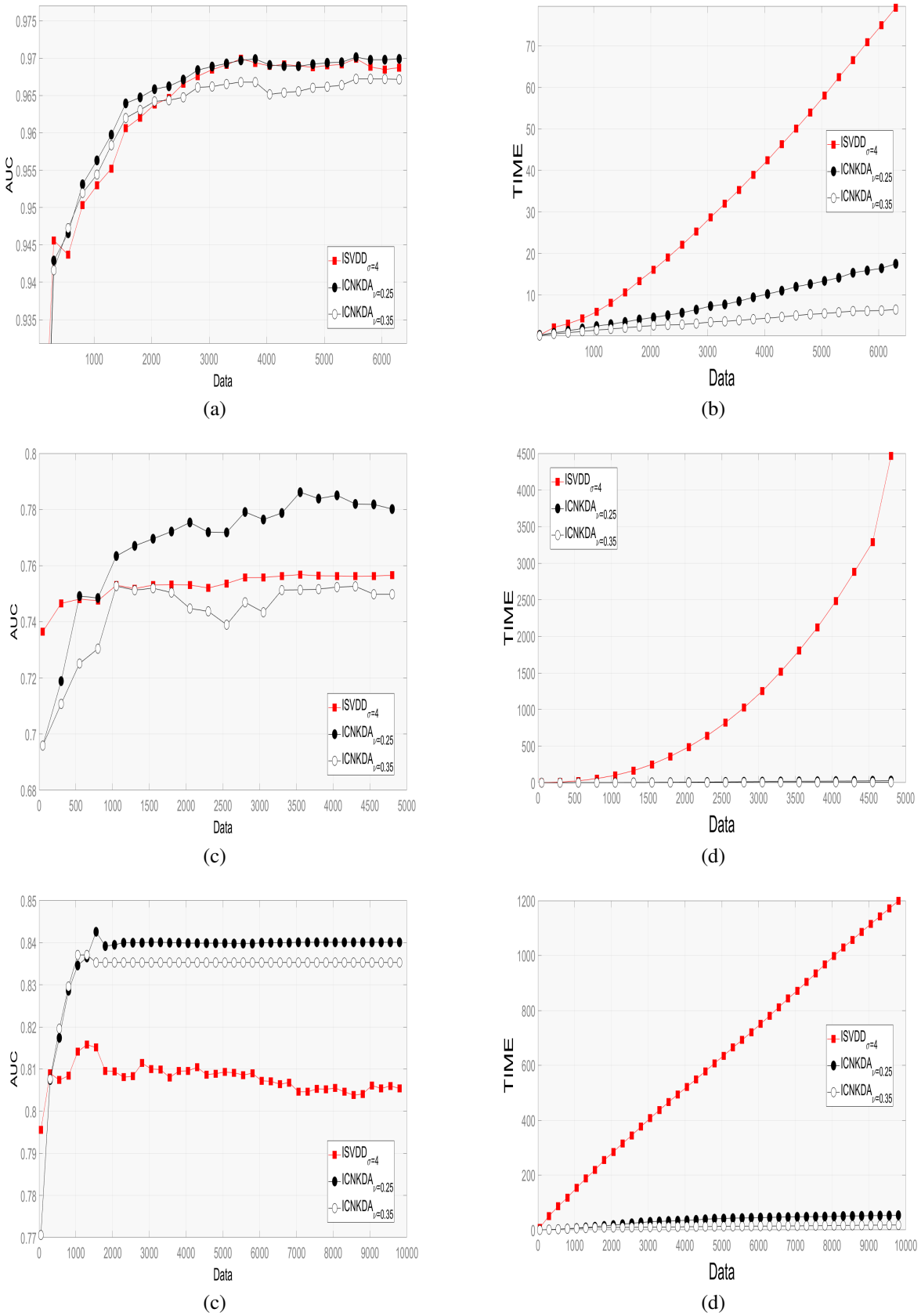


Figure 6: Large scale one-class learning. First row: evolution of the AUC values (a) and training times (b) recorded by ICNKDA (circles) and ISVDD (squares) from MNIST images (target: digit 4) Second row: evolution of the AUC values (a) and training times (b) obtained on CIFAR-10 images (target: deer). Third row: evolution of the AUC values (a) and training times (b) obtained on URL adresses (target: benign adresses)

Table 8: large scale multi-class learning.

Data	Methods	Times (sec)	AUC	CR
$MNIST_{10}$	ICNKDA$_{\nu=0.45}$	90.6	99.74	98.07
	ICNKDA$_{\nu=0.65}$	33.07	99.60	98.56
	IFKDA$_{\sigma=4}$	1.09	96.26	-
	IFKDA$_{\sigma=10}$	1.14	95.18	-
	SoDA$_{l=200}$	27.87	96.81	-
	SoDA$_{l=400}$	23.26	96.71	-
$CIFAR_{10}$	ICNKDA$_{\nu=0.45}$	193.72	80.16	97.26
	ICNKDA$_{\nu=0.65}$	52.71	78.57	98.17
	IFKDA$_{\sigma=4}$	2.48	62.28	-
	IFKDA$_{\sigma=10}$	2.47	65.82	-
	SoDA$_{l=200}$	402.90	69.20	-
	SoDA$_{l=400}$	248.30	70.40	-

Table 9: large scale one-class learning.

Data	Methods	Times (sec)	AUC	CR
$MNIST_{'4'}$	ICNKDA$_{\nu=0.25}$	17.76	97.01	92.97
	ICNKDA$_{\nu=0.35}$	6.55	96.75	95.33
	ISVDD$_{\sigma=4}$	80.42	96.94	95.36
$CIFAR_{'deer'}$	ICNKDA$_{\nu=0.25}$	29.95	78.1	83.91
	ICNKDA$_{\nu=0.35}$	7.60	75.1	90.96
	ISVDD$_{\sigma=4}$	4643	75.7	56.10
$URL_{'1'}$	ICNKDA$_{\nu=0.25}$	54.18	84.1	91.77
	ICNKDA$_{\nu=0.35}$	18.54	83.5	95.96
	ISVDD$_{\sigma=4}$	1222	80.5	97.16

learning methods, we propose an efficient compression mechanism based on the main condition of the null KDA. The proposed criterion removes redundancy in the data and the overlapping rate is monitored by a compression factor. The combined action of this two steps allows to reduce significantly the learning time while maintaining a classification level close to the result obtained by the exact method (without compression).

Another contribution is an exact updating scheme for computing the solution of kernel discriminant analysis compared to the one proposed by Liu et al. in [29] which does not respect the main conditions of KNDA. As the consequence, the Liu's scheme produces more null projection directions than theoretically expected and we observe experimentally a loss of classification performance during the learning process.

The proposed method is compared with recent incremental kernel learning methods both on multi-class and one-class problems. The results obtained show that our exact incremental KNDA equipped with a compression strategy offers the capacity to deal with large scale streaming data sets.

References

[1] Alexandre-Cortizo, E., Rosa-Zurera, M., Lopez-Ferreras, F., 2005. Application of fisher linear discriminant analysisto speech/music classification. EUROCON 2005 - The International Conference on "Computer as a Tool" 2, 1666–1669.

[2] Belhumeur, P., Hespanha, J., Kriegman, D., 1997. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence 19, 711–720.

[3] Bodesheim, P., Freytag, A., Rodner, E., Kemmler, M., Denzler, J., 2013. Kernel null space methods for novelty detection, in: Computer Vision and Pattern Recognition, pp. 3374–3381.

[4] Cai, D., He, X., Han, J., 2011. Speed up kernel discriminant analysis. The VLDB Journal 20, 21–33.

[5] Cauwenberghs, G., Poggio, T., 2000. Incremental and decremental support vector machine learning, in: Proceedings of the 13th International Conference on Neural Information Processing Systems, MIT Press, Cambridge, MA, USA. pp. 388–394.

[6] Chen, F., Liao, H., Ko, M., Lin, J., Yu, G., 2001. A new lda-based face recognition system which can solve the small sample size problem, in: Pattern Recognition, pp. 2067–2070.

[7] Chin, T., Suter, D., 2007. Incremental kernel principal component analysis. IEEE Transactions on Image Processing 16, 1662–1674.

[8] Chin, T.J., Suter, D., 2007. Incremental kernel principal component analysis. IEEE Trans. Image Processing 16, 1662–1674.

[9] Chowdhury, A., Yang, J., Drineas, P., 2020. Randomized iterative algorithms for fisher discriminant analysis, in: Proceedings of The 35th Uncertainty in Artificial Intelligence Conference, pp. 239–249.

[10] Chu, D., Liao, L., Ng, M.K., Wang, X., 2015. Incremental linear discriminant analysis: A fast algorithm and comparisons. IEEE Transactions on Neural Networks and Learning Systems 26, 2716–2735.

[11] Chu, D., Thye, G.S., 2010. A new and fast implementation for null space based linear discriminant analysis. Pattern Recognition 43, 1373–1379. URL: https://doi.org/10.1016/j.patcog.2009.10.004, doi:10.1016/j.patcog.2009.10.004.

[12] Dufrenois, F., Noyer, J., 2016. One class proximal support vector machines. Pattern Recognition 52, 96–112.

[13] Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.A., 2009. Describing objects by their attributes., in: CVPR, IEEE Computer Society. pp. 1778–1785. URL: https://vision.cs.uiuc.edu/attributes/.

[14] Georghiades, A., Belhumeur, P., Kriegman, D., 2001. From few to many: Illumination cone models for face recognition under variable lighting and pose. IEEE Trans. Pattern Anal. Mach. Intelligence 23, 643–660. URL: http://vision.ucsd.edu/ leekc/ExtYaleDatabase/ExtYaleB.html.

[15] Gkalelis, N., Mezaris, V., 2017. Incremental accelerated kernel discriminant analysis. MM '17: Proceedings of the 25th ACM international conference on Multimedia , 1575–1583.

[16] Howland, P., Park, H., 2004. Generalizing discriminant analysis using the generalized singular value decomposition. IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 995–1006. doi:10.1109/TPAMI.2004.46.

[17] Huang, R., Liu, Q., Lu, H., Ma, S., 2002. Solving the small sample size problem of lda. Proceedings of the 16th International Conference on Pattern Recognition 3, 29–32.

[18] Jiang, H., Wang, H., Hu, W., Kakde, D., Chaudhuri, A., 2017. Fast incremental svdd learning algorithm with the gaussian kernel. arXiv e-prints .

[19] J.Lim, D.Ross, Lin, R., Yang, M., 2004. Incremental learning for visual tracking. In Advances in NIPS , 793 – 800.

[20] J.Ye, T.Xiong, 2006. Null space versus orthogonal linear discriminant analysis. Proc. Int. Conf. Machine Learning , 1073–1080.

[21] Kemmler, M., Rodner, E., Wacker, E., Denzler, J., 2013. One-class classification with gaussian processes. Pattern Recogn. 46, 3507–3518.

[22] Krizhevsky, A., Hinton, G., 2009. Learning multiple layers of features from tiny images URL: https://www.cs.toronto.edu/ kriz/cifar.html.

[23] Laskov, P., Gehl, C., Krüger, S., Müller, K., 2006. Incremental support vector learning: Analysis, implementation and applications. J. Mach. Learn. Res. 7, 1909–1936.

[24] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition , 2278–2324URL: http://yann.lecun.com/exdb/mnist/.

[25] Levey, A., Lindenbaum, M., 2000. Sequential karhunen-loeve basis extraction and its application to images. IEEE Transaction on Imgage Processing 9, 1371–1374.

[26] Lewis, D.D., Yang, Y., Rose, T.G., Li, F., 2004. Rcv1: A new benchmark collection for text categorization research. J. Mach. Learn. Res. 5, 361–397. URL: http://www.daviddlewis.com/resources/testcollections/reuters21578.

[27] Li, W., Zhong, Z., Zheng, W., 2017. One-pass person re-identification by sketch online discriminant analysis. CoRR abs/1711.03368. URL: http://arxiv.org/abs/1711.03368.

[28] Liberty, E., 2013. Simple and deterministic matrix sketching, in: The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013, ACM. pp. 581–588.

[29] Liu, J., Lian, Z., Wang, Y., Xiao, J., 2017. Incremental kernel null space discriminant analysis for novelty detection. CVPR , 4123–4131.

[30] Ma, J., Saul, L.K., Savage, S., Voelker, G.M., 2009. Identifying suspicious urls: An application of large-scale online learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 681–688. URL: http://www.sysnet.ucsd.edu/projects/url/.

[31] Musco, C., Musco, C., 2017. Recursive sampling for the nyström method. Advances in Neural Information Processing Systems 2017-December, 3834–3846. 31st Annual Conference on Neural Information Processing Systems, NIPS 2017 ; Conference date: 04-12-2017 Through 09-12-2017.

[32] Park, H., Drake, B., Lee, S., Park, C., 2007. Fast linear discriminant analysis using qr decomposition and regularization. Technical report GT-CSE-07-21 .

[33] Rahimi, A., Recht, B., 2007. Random features for large-scale kernel machines, in: Proceedings of the 20th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA. pp. 1177–1184.

[34] Roth, V., 2006. Kernel fisher discriminant for outlier detection. Neural Computation 18, 942–960.

[35] Rudin, W., 1962. Fourier analysis on groups.

[36] S. Pang, Ozawa, S., Kasabov, N., 2005. Incremental linear discriminant analysis for classification of data streams. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 35, 905–914.

[37] Sharma, A., Paliwal, K.K., 2012. A new perspective to null linear discriminant analysis method and its fast implementation using random matrix multiplication with scatter matrices. Pattern Recognition 45, 2205–2213.

[38] Shawe-Taylor, J., Cristianini, N., 2004. Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA.

[39] Tax, D., Duin, R., 1999. Support vector domain description. Pattern Recognition Letters 20, 1191–1199.

[40] Wang, Y., Xue, N., Fan, X., Luo, J., Liu, R., Chen, B., Li, H., Luo, Z., 2018. Fast factorization-free kernel learning for unlabeled chunk data streams. IJCAI , 2833–2839.

[41] Ye, H., Li, Y., Chen, C., Zhang, Z., 2017. Fast fisher discriminant analysis with randomized algorithms. Pattern Recognition 72, 82–92.

[42] Y.F.Guo, L.Wu, H.Lu, Z.Feng, X.Xue, 2006. Null foley-sammon transform. Pattern Recognition 39, 2248–2251.

[43] Yu, H., Yang, J., 2001. A direct lda algorithm for high-dimensional data - with application to face recognition. Pattern Recognition 34, 2067–2070.

[44] Zhang, Z., Dai, G., Xu, C., Jordan, M.I., 2010. Regularized discriminant analysis, ridge regression and beyond. J. Mach. Learn. Res. 11, 2199–2228.