Fast Factorization-Free Kernel Learning for Unlabeled Chunk Data Streams

Yi Wang[1,2], **Nan Xue**[1,2], **Xin Fan**[1,2*], **Jiebo Luo**[3],
Risheng Liu[1,2], **Bin Chen**[1,2], **Haojie Li**[1,2] and **Zhongxuan Luo**[1,2]
[1] DUT-RU International School of Information Science and Engineering,
Dalian University of Technology, P. R. China
[2] Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, P. R. China
[3] Department of Computer Science, University of Rochester, USA
{dlutwangyi, xin.fan, rsliu, hjli, zxluo}@dlut.edu.cn, xnfiona@gmail.com, jluo@cs.rochester.edu

Abstract

Data stream analysis aims at extracting discriminative information for classification from continuously incoming samples. It is extremely challenging to detect novel data while updating the model in an efficient and stable fashion, especially for the chunk data. This paper proposes a fast factorization-free kernel learning method to unify novelty detection and incremental learning for unlabeled chunk data streams in one framework. The proposed method constructs a joint reproducing kernel Hilbert space from known class centers by solving a linear system in kernel space. Naturally, unlabeled data can be detected and classified among multi-classes by a single decision model. And projecting samples into the discriminative feature space turns out to be the product of two small-sized kernel matrices without needing such time-consuming factorization like QR-decomposition or singular value decomposition. Moreover, the insertion of a novel class can be treated as the addition of a new orthogonal basis to the existing feature space, resulting in fast and stable updating schemes. Both theoretical analysis and experimental validation on real-world datasets demonstrate that the proposed methods learn chunk data streams with significantly lower computational costs and comparable or superior accuracy than the state of the art.

1 Introduction

One of the challenging aspects of data analytics nowadays is dealing with streaming and fast-moving input data, e.g., digital monitor images, social media feedbacks, marketing and financial data. In real applications, these data might come from several terminals without labels, simultaneously, forming chunk data streams. Particularly, a chunk of data may contain some new examples belong to the seen/known classes and/or unseen/novel classes. Therefore, we have to detect novelties first, and then incorporate all new information into the existing model. Although, novelty detection and incremental learning for data streams have received increasing attention in the past few years [Wang *et al.*, 2016; Faria *et al.*, 2016; Lu *et al.*, 2017], efficient and stable solutions to simultaneously detect novelties and learn chunk data streams are still rare.

Recently, kernel null-space based discriminant analysis (KNDA) [Bodesheim *et al.*, 2013] and incremental KNDA (IKNDA) [Liu *et al.*, 2017] have been reported that their performances outperform classifiers SVDD [Tax and Duin, 2004] and GP-Mean/GP-Var [Kemmler *et al.*, 2010], and deep approaches (e.g., Alexnet) [Krizhevsky *et al.*, 2012] in multi-classification and novelty detection. KNDA learns a decision model by removing the intra-class variances in a reproducing kernel Hilbert null (RKHN) space, which provides elegant discrimination for highly complex and non-linear distribution data. Unfortunately, the eigen-decomposition of a large kernel matrix makes KNDA hard to scale up. IKNDA provides an updating scheme to renew the null space based on singular value decomposition (SVD). However, its incremental updating mechanism is not efficient for on-line learning tasks on large datasets. Besides, kernel methods also suffer from high memory cost of kernel matrix on large and/or high-dimensional datasets.

In this work, we put forward an efficient kernel learning method to unify the novelty detection and incremental learning for unlabeled chunk data streams in one framework. Specifically, our contributions are twofold:

- We propose an efficient factorization-free batch kernel discriminant analysis (FKDA). FKDA constructs a k-dimensional RKH space from k known class centers by solving a linear system [Chu *et al.*, 2015] in kernel space. The mapping (projection) to the feature space turns out to be the production of two small kernel matrices, thus the computational complexity of FKDA is significantly lower than other kernel DA methods, especially for large-scale scenarios. Moreover, each class is represented by a single point with a deterministic novelty threshold in RKH space, that also caters for multi-class novelty detection.

*Corresponding Author

- We develop a fast and stable incremental scheme for FKDA, called IFKDA. In IFKDA, inserting a novel class to the learned model acts by appending a new orthogonal basis to the original feature space without disturbing the distribution of known classes, which makes the evolution more robust and stable. Moreover, IFKDA is very efficient in on-line learning scenarios, requiring only a small portion of the key kernel matrices to be recomputed.

2 Overview of Discriminative Analysis (DA)

In this section, we give a short overview of linear discriminative analysis and kernel discriminative analysis.

Given a data matrix with k clusters, $\boldsymbol{X} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_k\} \in \mathbb{R}^{d\times n}$, where $\boldsymbol{X}_i \in \mathbb{R}^{d\times n_i}$ and $n = \sum_{i=1}^{k} n_i$. The widely known Foley-Sammon transform [Chen *et al.*, 2000] estimates a low-dimensional subspace to discriminative features by maximizing the following criterion:

$$\boldsymbol{G}^* := \arg\max_{\boldsymbol{G}} trace((\boldsymbol{G}^T\boldsymbol{S}_t\boldsymbol{G})^{-1}(\boldsymbol{G}^T\boldsymbol{S}_b\boldsymbol{G})), \quad (1)$$

where $\boldsymbol{G} \in \mathbb{R}^{d\times l}$ is a transformation matrix which projects data in d-dimensional space to l-dimensional subspace, $\boldsymbol{S}_b$ is between-class scatter matrix, and $\boldsymbol{S}_t$ is total scatter matrix ($\boldsymbol{S}_t = \boldsymbol{S}_b + \boldsymbol{S}_w$, where $\boldsymbol{S}_w$ is within-class scatter matrix).

A common issue of discriminative learning is the small-size-sample (SSS) problem that n is far less than d, such that $\boldsymbol{S}_w$ is singular in general. One popular extension replaces the inverse in the classical model by pseudo-inverse, resulting in the following optimization model [Huang *et al.*, 2002],

$$\boldsymbol{G}^* := \arg\max_{\boldsymbol{G}} trace((\boldsymbol{G}^T\boldsymbol{S}_t\boldsymbol{G})^{\dagger}(\boldsymbol{G}^T\boldsymbol{S}_b\boldsymbol{G}), \quad (2)$$

where $(\cdot)^{\dagger}$ denotes the pseudo-inverse of $(\cdot)$. And any transfer matrix $\boldsymbol{G}$ with the form

$$trace((\boldsymbol{G}^T\boldsymbol{S}_t\boldsymbol{G})^{\dagger}(\boldsymbol{G}^T\boldsymbol{S}_b\boldsymbol{G})) = k-1, \quad (3)$$

should be an optimal solution to (2) [Chu *et al.*, 2015].

In [Chu *et al.*, 2015], Chu *et al.* proposed a more efficient way for SSS problem by solving a linear system in (4) via the economic QR-factorization of $\boldsymbol{X}$.

$$\boldsymbol{X}^T\boldsymbol{G} = \boldsymbol{E}, \quad (4)$$

where $\boldsymbol{G} \in \mathbb{R}^{d\times k}$, and $\boldsymbol{E}$ is a n-by-k matrix and the i-th column is $(0, ..., e_i, ..., 0)^T$ where $e_i = [1...1]^T \in \mathbb{R}^{n_i\times 1}$.

Assuming that training samples are linearly independent, the solution of (4) with the minimal 2-norm is

$$\boldsymbol{G} = \boldsymbol{Q}\boldsymbol{R}^{-T}\boldsymbol{E}. \quad (5)$$

If considering kernel tricks for DA, we should further map $\boldsymbol{X}$ into a feature space $\boldsymbol{H}$ with an implicit mapping function $\boldsymbol{\Phi} : x_j^i \to \phi(x_j^i)$, where $\phi(x_j^i) \in \boldsymbol{H}$ and x_j^i denote the j-th item of the i-th class. And we have

$$\boldsymbol{X}^{\Phi} = \boldsymbol{\Phi}(\boldsymbol{X}) = [\phi(x_1), \ldots, \phi(x_n)]. \quad (6)$$

Accordingly, the criterion in (2) extends to

$$\boldsymbol{G}^{*\Phi} := \arg\max_{\boldsymbol{G}^{\Phi}} trace(((\boldsymbol{G}^{\Phi})^T\boldsymbol{S}_t^{\Phi}\boldsymbol{G}^{\Phi})^{\dagger}(\boldsymbol{G}^{\Phi})^T\boldsymbol{S}_b^{\Phi}\boldsymbol{G}^{\Phi}). \quad (7)$$

And any $\boldsymbol{G}^{\Phi}$ with the form

$$trace(((\boldsymbol{G}^{\Phi})^T\boldsymbol{S}_t{}^{\Phi}\boldsymbol{G}^{\Phi})^{\dagger}(\boldsymbol{G}^{\Phi})^T\boldsymbol{S}_b{}^{\Phi}\boldsymbol{G}^{\Phi}) = k-1, \quad (8)$$

is an optimal solution of the optimization problem (7).

3 Factorization-free KDA (FKDA)

In this section, a fast factorization-free batch kernel DA method (FKDA) is developed. We first induce the linear system in (4) into the kernel space.

Theorem 1: Assuming $\boldsymbol{X}^{\boldsymbol{\Phi}}$ are linearly independent, and the linear system in (9) is solvable, then the solution to the optimization problem (7) is given by $\boldsymbol{G}^{\Phi}$.

$$(\boldsymbol{X}^{\Phi})^T\boldsymbol{G}^{\Phi} = \boldsymbol{E}. \quad (9)$$

Proof: See the Appendix.

Considering the advantages of using class centers as input in storage and computation as in AKDA/QR [Xiong *et al.*, 2005], we introduce them into (9) and have

$$(\boldsymbol{C}^{\Phi})^T\boldsymbol{G}^{\Phi} = \boldsymbol{M}^T(\boldsymbol{X}^{\Phi})^T\boldsymbol{G}^{\Phi} = \boldsymbol{M}^T\boldsymbol{E}, \quad (10)$$

where $\boldsymbol{C}^{\Phi} = \boldsymbol{X}^{\Phi}\boldsymbol{M}$ is a global centroid matrix in the kernel space, and $\boldsymbol{M}$ is a n-by-k matrix with the i-th column is $(0, ..., e_i/n_i, ..., 0)^T$. Since

$$\begin{aligned}\boldsymbol{M}^T\boldsymbol{E} &= \begin{bmatrix} \frac{1}{n_1}e_1{}^T & & \\ & \ddots & \\ & & \frac{1}{n_k}e_k{}^T \end{bmatrix}\begin{bmatrix} e_1 & & \\ & \ddots & \\ & & e_k \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{n_1}e_1{}^Te_1 & & \\ & \ddots & \\ & & \frac{1}{n_k}e_k{}^Te_k \end{bmatrix} = \boldsymbol{I},\end{aligned} \quad (11)$$

where $\boldsymbol{I} \in \mathbb{R}^{k\times k}$ is an identity matrix, then

$$(\boldsymbol{C}^{\Phi})^T\boldsymbol{G}^{\Phi} = \boldsymbol{I}. \quad (12)$$

A key observation for most widely used Gaussian kernel function: $exp(-\|x-y\|^2/\sigma)$, is that for relatively large σ, the center of each class in the original space will be projected very close to the center of each class in the kernel space [Xiong *et al.*, 2005]. So we adopt this idea to compute the centroid matrix $\boldsymbol{C}_o$ in the original space first, then we approximate the kernel centroid matrix $\boldsymbol{C}^{\Phi}$ by

$$\boldsymbol{C}^{\Phi} \approx [\phi(c_1), \ldots, \phi(c_k)](\equiv \boldsymbol{C}_o^{\Phi}), \quad (13)$$

where c_i denotes the center vector of class $\boldsymbol{X}_i$ in the original space. Therefore, we have

$$(\boldsymbol{C}_{\boldsymbol{o}}^{\boldsymbol{\Phi}})^T\boldsymbol{G}^{\boldsymbol{\Phi}} = \boldsymbol{I}. \quad (14)$$

Assuming that class centers are linearly independent, we can resort to QR-decomposition of $\boldsymbol{C}_o^{\Phi} = \boldsymbol{Q}^{\Phi}\boldsymbol{R}^{\Phi}$ to solve $\boldsymbol{G}^{\Phi}$, where $\boldsymbol{Q}^{\Phi}$ is the column orthogonal and $\boldsymbol{R}^{\Phi}$ is nonsingular. Then we have

$$\boldsymbol{G}^{\Phi} = \boldsymbol{Q}^{\Phi}(\boldsymbol{R}^{\Phi})^{-T}. \quad (15)$$

Denote $\boldsymbol{K}_o = (\boldsymbol{C}_o^{\Phi})^T\boldsymbol{C}_o^{\Phi} \in \mathbb{R}^{k\times k}$.

Since $(\boldsymbol{R}^{\Phi})^T\boldsymbol{R}^{\Phi} = (\boldsymbol{R}^{\Phi})^T(\boldsymbol{Q}^{\Phi})^T\boldsymbol{Q}^{\Phi}\boldsymbol{R}^{\Phi} = (\boldsymbol{C}_o^{\Phi})^T\boldsymbol{C}_o^{\Phi}$, thus

$$\boldsymbol{G}^{\Phi} = \boldsymbol{Q}^{\Phi}\boldsymbol{R}^{\Phi}(\boldsymbol{R}^{\Phi})^{-1}(\boldsymbol{R}^{\Phi})^{-T} = \boldsymbol{C}_o^{\Phi}\boldsymbol{K}_o^{-1}. \quad (16)$$

Result: Given a sample z, we can get its projection in the k-dimension RKH space by $(\boldsymbol{G}^{\Phi})^T\phi(z)$ in (17).

$$(\boldsymbol{G}^{\Phi})^T\phi(z) = (\boldsymbol{C}_o^{\Phi}\boldsymbol{K}_o^{-1})^T\phi(z) = \boldsymbol{K}_o^{-1}\boldsymbol{K}_{cz}, \quad (17)$$

Algorithm 1 FKDA-batch method

Require: Data matrix $\boldsymbol{X}$, nonlinear mapping $\boldsymbol{\Phi}$ and a testing sample z.
Ensure: The projected feature of z.
Stage1:
1: Compute the centroid matrix: $\boldsymbol{C}_o$.
2: Construct the kernel matrix: $\boldsymbol{K}_o = (\boldsymbol{C}_o^{\Phi})^T \boldsymbol{C}_o^{\Phi}$.
Stage2:
3: Construct the kernel vector: $\boldsymbol{K}_{cz} =< \phi(\boldsymbol{C}_o^{\Phi}), \phi(z) >$.
4: Compute $(\boldsymbol{G}^{\Phi})^T z^{\Phi} = \boldsymbol{K}_o^{-1} \boldsymbol{K}_{cz}$.

Method	Space	Time
AKDA/QR	$\boldsymbol{O}(dk + nk + k^2)$	$\boldsymbol{O}(dnk + dn + dk^2)$
SRKDA	$\boldsymbol{O}(n^2 + nk + k^2)$	$\boldsymbol{O}(dn^2 + n^3 + n^2k)$
KNDA	$\boldsymbol{O}(n^2 + nk)$	$\boldsymbol{O}(dn^2 + n^3 + n^2k)$
LDA/QR	$\boldsymbol{O}(dn + n^2 + nk)$	$\boldsymbol{O}(dn^2 + dnk)$
KPE	$\boldsymbol{O}(n^2)$	$\boldsymbol{O}(dn^2 + dn + n^3)$
FKDA	$\boldsymbol{O}(dk + k^2)$	$\boldsymbol{O}(dn + dk^2)$

Table 1: Space and time complexities for batch DAs (n is the training sample number, d is the feature dimension, and k is the class number).

where $\boldsymbol{K}_{cz} \in \mathbb{R}^{k \times 1}$ and $\boldsymbol{K}_{cz}(i) =< \phi(c_i), \phi(z) >$.

Obviously, there is no need to really compute QR-decomposition of $\boldsymbol{C}_o^{\Phi}$.

FKDA is summarized as **Algorithm 1**.

Next, we compare the space and time complexities of FKDA with AKDA/QR [Xiong *et al.*, 2005], SRKDA [Cai *et al.*, 2007], KNDA [Bodesheim *et al.*, 2013], KPE [Min *et al.*, 2016], and LDA/QR [Chu *et al.*, 2015] in Table 1. Since FKDA gives an implicit transform matrix $\boldsymbol{G}^{\Phi}$, we add the projecting step to other methods for fairness. We also compare with a linear DA method (i.e., LDA/QR), so the space and time complexities of constructing the kernel matrix are added for all kernel methods in comparisons.

AKDA/QR and FKDA both use centroid matrix as input, so they need less space and time cost in constructing kernel matrix than other kernel DAs. The cost of constructing centroid matrix is as low as $\boldsymbol{O}(dn)$ in our implementation. Although, FKDA has to compute the inverse of $\boldsymbol{K}_o \in \mathbb{R}^{k \times k}$ with the complexity of $\boldsymbol{O}(k^3)$, other kernel DAs also have such computation, e.g., AKDA/QR needs four times matrix inversion with the same scale, though it employs QR-decomposition, and KNDA has to compute the inversion of a matrix with the complexity of $\boldsymbol{O}(n^3)$. In addition, the total computation cost of FKDA, including constructing kernel matrix, is no more than LDA/QR, especially when k is far less than n. Besides, LDA/QR has a demand of $(d >> n)$ of $\boldsymbol{Q}$ matrix of $\boldsymbol{X}$. In contrast, FKDA needs no factorization and can perform well on both $(d >> n > k)$ and $(n >> d > k)$ datasets, that will be demonstrated in Section 5.

4 On-line Learning for Chunk Data Streams

The on-line phase, shown in Figure 1, is composed of three operations: detect novelties, classify new samples and evolute the decision model. We will detail them in this section.

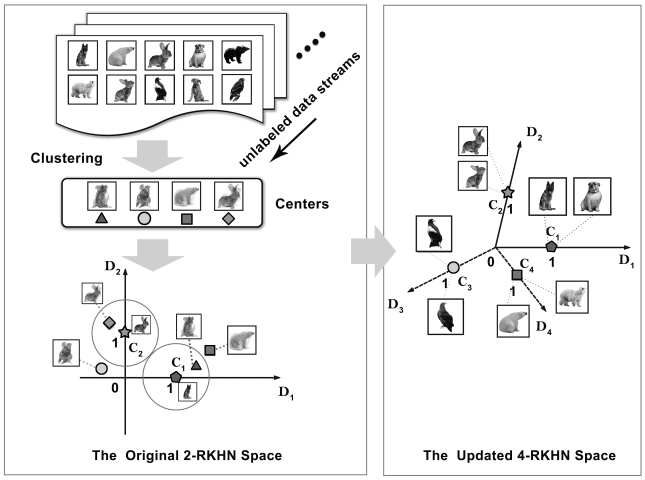


Figure 1: The framework of on-line learning for unlabeled chunk data streams. FKDA learns a k-RHK space from k known classes centers, and the projection of each known class center is fixed in the unit position on each axis, e.g., $\boldsymbol{C}_1$ of $\boldsymbol{X}_1(Dog)$ and $\boldsymbol{C}_2$ of $\boldsymbol{X}_2(Rabit)$. New or/and novel examples can be detected and discriminated in one decision model with a deterministic boundary threshold. The model evolves with the centralized new data. Particularly, inserting novel classes, e.g., $\boldsymbol{X}_3(Eagle)$ and $\boldsymbol{X}_4(PolarBear)$, equals to append new basics, e.g., $\boldsymbol{D}_3$ and $\boldsymbol{D}_4$, perpendicular to the original space.

4.1 Novelty Detection and Classification

Given a chunk of unlabeled data, we assume they are clustered without noises or outliers by clustering algorithms, e.g., CluStream [Aggarwal *et al.*, 2003], for clustering is not the focus of this work. Then we have a set of unlabeled cluster centers as input. And novelty detection and classification are performed in the learned k-RKH space.

To be noted, from the derivation of (14), we have

$$((\boldsymbol{C}_{\boldsymbol{o}}^{\boldsymbol{\Phi}})^T \boldsymbol{G}^{\boldsymbol{\Phi}})^T = (\boldsymbol{G}^{\boldsymbol{\Phi}})^T (\boldsymbol{C}_{\boldsymbol{o}}^{\boldsymbol{\Phi}}) = \boldsymbol{I}. \tag{18}$$

This means k class centers are projected to the unit positions of the k orthogonal axes, respectively, for the value of each volume of $\boldsymbol{I}$ represents the coordinate of the mapping points of known class centers in k-RKH space. For instance, given two known classes as shown in Figure 1, we have a 2-RKH space, where $\boldsymbol{c}_1$ of $\boldsymbol{X}_1(Dog)$ is projected to $\boldsymbol{C}_1(1, 0)$ and $\boldsymbol{c}_2$ of $\boldsymbol{X}_2(Rabit)$ is projected to $\boldsymbol{C}_2(0, 1)$.

Therefore, we can build a hypersphere surrounding each class center as classification boundary. New instances will be classified with respect to their distances to the previously known classes centers. The instances that being projected out of boundaries of all the known classes are classified as novel classes. Furthermore, we set the same value for the thresholds of boundaries of known classes, i.e., the half value of the Euclidean distance between two arbitrary classes. Since the positions of class centers are fixed on the axes, this value is also valid for the new inserted classes, that will be explained in the next subsection.

4.2 Updating Schemes (IFKDA)

The updating task can be easily accomplished by recomputing or appending some parts of the kernel matrices $\boldsymbol{K}_o$ and $\boldsymbol{K}_{cz}$.

We suppose that a chunk of data is composed of some new samples belong to k_a existing classes and some samples belongs to k_b new classes. Then the new kernel matrix $\tilde{\boldsymbol{K}}_{\boldsymbol{o}} \in \mathbb{R}^{(k+k_b)\times(k+k_b)}$ can be updated by (19)

$$\tilde{\boldsymbol{K}}_{\boldsymbol{o}} = \begin{bmatrix} \boldsymbol{K}'_o & \boldsymbol{K}^b_o \\ \boldsymbol{K}^{bT}_o & \boldsymbol{K}^c_o \end{bmatrix}, \tag{19}$$

where $\boldsymbol{K}'_o \in \mathbb{R}^{k\times k}$ is the updated original kernel matrix $\boldsymbol{K}_o$ by recomputing the items relating to those known classes which have new samples, and $\boldsymbol{K}^b_o \in \mathbb{R}^{k\times k_b}$ and $\boldsymbol{K}^c_o \in \mathbb{R}^{k_b\times k_b}$ are the new items introduced by novel classes in (20) and (21), respectively.

$$\boldsymbol{K}^b_o = \begin{bmatrix} < \alpha_1, \alpha_{k+1} >, \ldots, < \alpha_1, \alpha_{k+k_b} > \\ \vdots \\ < \alpha_k, \alpha_{k+1} >, \ldots, < \alpha_k, \alpha_{k+k_b} > \end{bmatrix} \tag{20}$$

$$\boldsymbol{K}^c_o = \begin{bmatrix} < \alpha_{k+1}, \alpha_{k+1} >, \ldots, < \alpha_{k+1}, \alpha_{k+k_b} > \\ \vdots \\ < \alpha_{k+k_b}, \alpha_{k+1} >, \ldots, < \alpha_{k+k_b}, \alpha_{k+k_b} > \end{bmatrix} \tag{21}$$

where α_i represents $\phi(\boldsymbol{c}_i)$.

The time complexity for computing $\tilde{\boldsymbol{K}}_o$, $\boldsymbol{K}^b_o$ and $\boldsymbol{K}^c_o$ are $\boldsymbol{O}(dkk_a)$, $\boldsymbol{O}(dkk_b)$, and $\boldsymbol{O}(dk^2_b)$, respectively.

Similarly, we update $\boldsymbol{K}_{cz}$ in two parts by

$$\tilde{\boldsymbol{K}}_{cz} = [\boldsymbol{K}'_{cz}, \boldsymbol{K}^b_{cz}]^T, \tag{22}$$

where $\boldsymbol{K}'_{cz}$ is the updated kernel vector after inserting labeled new samples by (23), and $\boldsymbol{K}^b_{cz}$ is the kernel vector after appending new classes by (24), where β denotes $\phi(z)$.

$$\boldsymbol{K}'_{cz} = [< \alpha_1, \beta >, \cdots, < \tilde{\boldsymbol{\alpha}}_{\boldsymbol{i}}, \boldsymbol{\beta} >, \cdots, < \alpha_k, \beta >]^T \tag{23}$$

$$\boldsymbol{K}^b_{cz} = [< \alpha_{k+1}, \beta >, \cdots, < \alpha_{k+k_b}, \beta >]^T \tag{24}$$

As we can see, the updating of known-class centers by new arriving samples does not change the rank of the kernel matrix, so as to the dimension of feature space. And the positions of known-class centers do not change under the constraint of (18). While inserting a novel class is equal to increasing one rank of the kernel matrix, resulting in appending a new basis perpendicular to the original space, see the demonstration in Figure 1. And the center of a new class is also projected into the unit position of the new axis, therefore its boundary threshold can be set to the same value as the knowns classes. Both of these features make IFKDA more stable and robust for classification, that will be verified by experiments on real datasets in Section 5.

We then compare space and time complexities for updating chunk data for IFKDA, ILDA/SSS [Kim *et al.*, 2011], and ILDA/QR [Chu *et al.*, 2015] in Table 2. In both respects, the updating costs of IFKDA are minimal, especially when k is far less than n ($k << n$) the advantages are more obvious. Besides, IKNDA only has the algorithm for learning novel classes, with $\boldsymbol{O}(ns + s^2)$ and $\boldsymbol{O}(d(n+s)s + n^2s + ns^2)$ in space and time complexities (where s is the sample number in a new class), respectively, for the insertion of one novel class. In contrast, IFKDA only needs extra space of $(d + k + 1)$, and has $\boldsymbol{O}(dk + d)$ in time complexity in this case.

Method	Space	Time
ILDA/SSS	$\boldsymbol{O}(dh + dk_b)$	$\boldsymbol{O}(d\tilde{n}^2 + \tilde{n}^3)$
ILDA/QR	$\boldsymbol{O}(dh + \tilde{k}h + h^2)$	$\boldsymbol{O}(dnh + d\tilde{k}h + dh^2)$
IFKDA	$\boldsymbol{O}(dk_b + \tilde{k}k_b)$	$\boldsymbol{O}(dh + dkf + dk_b^2)$

Table 2: The space and time complexities of incremental DAs for learning chunk data. (h is the sample number in a chunk, k_b is the new class number in a chunk, $\tilde{n} = n + h$, $\tilde{k} = k + k_b$, and f is the total class number in a chunk.)

Dataset	d	k_t	n_i	k_a/k_b	n_{c_i}
AR	2000	100	6	5/4	4
MNIST-F	784	5	1000	1/1	50
AWA	4096	40	56	5/2	18
Caltech256	4096	200	37	5/3	12

Table 3: The settings of datasets. (d: the dimension of data, k_t: the training class number, n_i: the sample number in each training class, k_a: the known class number in a chunk, k_b: the novel class number in a chunk, and n_{c_i}: the sample number of each class in a chunk)

5 Experimental Results and Analysis

In this section, we evaluate the performances of the proposed methods on four publicly-available datasets: AR [Kim *et al.*, 2011], AWA [Lampert *et al.*, 2014], Caltech256 [Griffin *et al.*, 2007] and MNIST-Fashion (MINST-F for short) [Xiao *et al.*, 2017]. For AWA and Caltech256, we take the outputs from the 7-th full connected layer of very deep 19-layer CNN as features (4096 dims). The kernel function: $exp(-\|x-y\|^2/\sigma)$ is used for all kernel DAs. Since FKDA uses approximate centers to construct kernel matrix as AKDA/QR, σ should be assigned a large value. Experiments show that choosing ($\sigma = d$) for FKDA(IFKDA) and AKDA/QR produce good overall results, we thus use this value for them in all the experiments. All methods are implemented in MATLAB and ran on an Intel (R) Core (TM) $i7$ PC with 3.40 GHz CPU and 8 GB RAM.

5.1 Multi-class Novelty Detection for Chunks

The settings of initial batch training stage and on-line incoming chunk data are in Table 3. Table 4 tabulates the results for FKDA and KNDA for detecting five unlabeled chunks with 20-fold cross-validation.

As we can see, although KNDA ($\sigma = 1$) produces good overall results on AWA and Caltech256, and KNDA ($\sigma = d$) performs better than KNDA ($\sigma = 1$) on AR and MNIST-F, the overall misclassification rates (denoted by Err) of FKDA ($\sigma = d$) are far less than those of KNDA on all datasets.

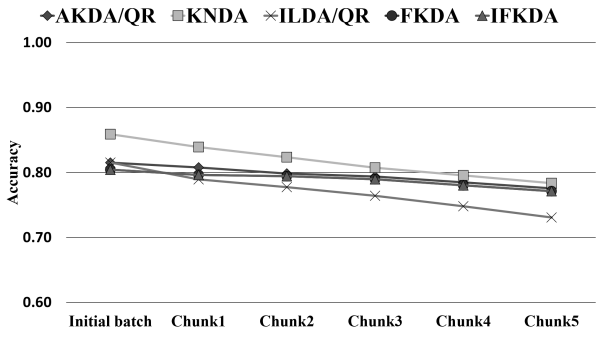
5.2 Incremental Learning

We first compare the performances of IFKDA and ILDA/QR with batch methods AKDA/QR, KNDA ($\sigma = 1$), and FKDA in learning chunk data streams on AWA and Caltech256.

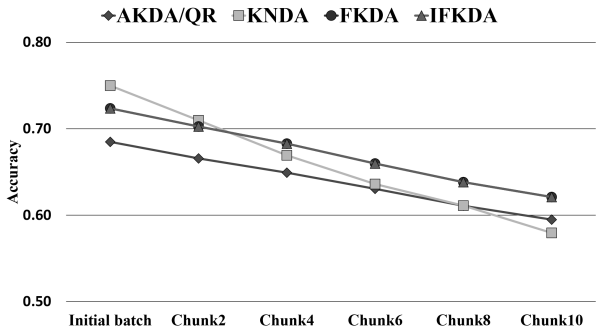
The settings of experimental data are the same as those in Table 3. The testing data are randomly selected from the rest samples of training classes and the novel classes with

Method	Dataset	Err	F_a	F_b	F_c
KNDA	AR	77.8	50.0	50.0	50.0
($\sigma = 1$)	MNIST-F	82.0	80.0	20.0	64.0
	AWA	**12.9**	40.0	1.2	0.8
	Caltech256	**43.0**	81.3	20.0	0.0
KNDA	AR	**48.4**	75.5	12.0	14.8
($\sigma = d$)	MNIST-F	**28.0**	56.0	0.0	0.0
	AWA	29.7	0.0	41.6	0.0
	Caltech256	58.8	45.3	66.8	0.0
FKDA	AR	**20.7**	0.5	36.8	0.0
($\sigma = d$)	MNIST-F	**22.0**	44.0	0.0	0.0
	AWA	**5.7**	17.0	1.2	0.0
	Caltech256	**31.0**	68.0	8.8	0.0

Table 4: The performance of novelty detection for chunk data streams. $Err = (F_n + F_p + F_e)/N_c$ is the overall percentage for misclassification, where F_n is the number of instances belonging to novel classes but misclassified into known classes, F_p is the number of instance belonging to known classes but misclassified into novel classes, and F_e is the number instance belonging to some known classes but misclassified into other known classes in a chunk. N_c is the total instance number in a chunk. N_o is the total instance number of novel classes in a chunk. $F_a = F_n/N_o$: % of novel-class instances falsely identified to known classes. $F_b = F_p/(N_c - N_o)$: % of known-class instances falsely identified to novel classes; $F_c = F_e/(N_c - N_o)$: % of known-class instances falsely identified to other known classes.



(a) AWA



(b) Caltech256

Figure 2: The classification accuracy of inserting chunk data streams.

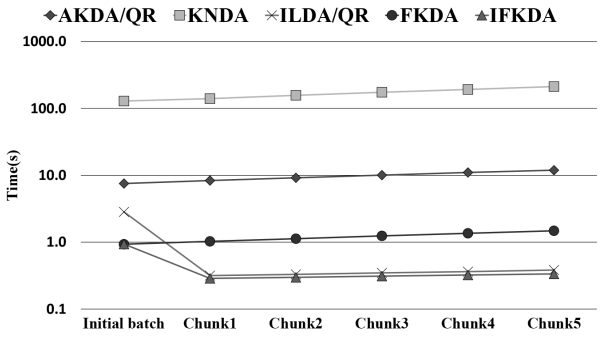
the same scale of chunk data. The results of classification accuracy and CPU time on two datasets are shown in Figure 2 and Figure 3 with 20-fold-cross validation, respectively. We have the following observations:

1) IFKDA is the exact incremental version of FKDA, giving the same accuracies to FKDA in all tests. The accuracies of IFKDA and FKDA are comparable to the best algorithm. All kernel DAs are more accurate than ILDA/QR.

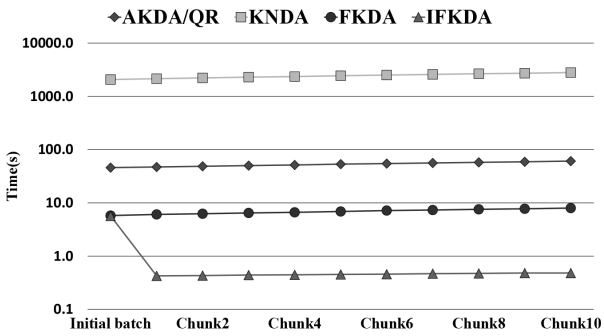
2) Both IFKDA and ILDA/QR are far faster than batch methods, which verifies the efficiency of incremental methods over the batch ones. IFKDA performs about 5-10 times faster than FKDA, hundreds of times faster than AKDA, and thousands of times faster than KNDA on both datasets. Although ILDA/QR runs similar as IFKDA on AWA ($d = 4096$ and $n = 2240$), but it can not work on Caltech256 (with $d = 4096$ and $n = 7400$), due to the limitation of ($d >> n$) for $\boldsymbol{Q}$ matrix of $\boldsymbol{X}$ in its algorithm.

3) FKDA spends less time than other batch methods in the training stage on both datasets. FKDA even has five times speed up over LDA/QR (the batch version of ILDA/QR, used in the initial training stage) on AWA. FKDA is dozens of times faster than AKDA/QR on AWA with comparable accuracy, and hundreds of times faster than AKDA/QR on Caltech256 with higher accuracy, even though they share the common on using centers as the input.

We then compare the classification accuracy and CPU time of IFKDA with IKNDA ($\sigma = 1$) in learning novel classes on Caltech256. AKDA/QR and FKDA are also taken as the baseline. The settings of batch training stage are the same as Table 3. And in on-line stage, fifteen novel classes (each has n_{c_i} samples) are inserted one by one. The testing samples are

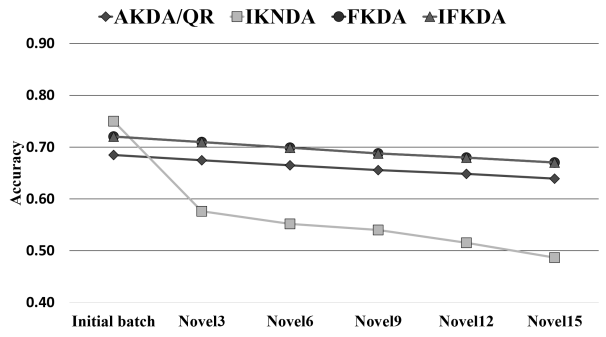


(a) AWA

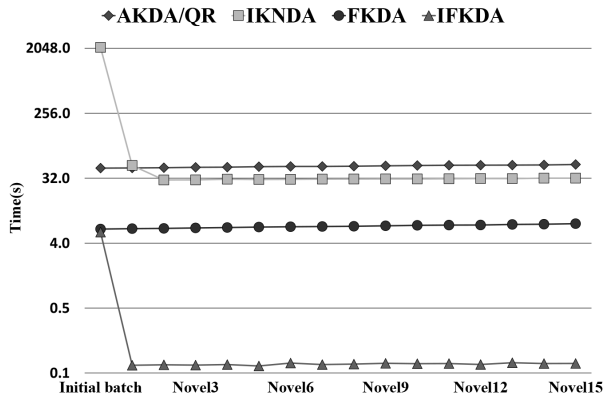


(b) Caltech256

Figure 3: The CPU time of inserting chunk data streams.



(a) Classification Accuracy



(b) CPU Time

Figure 4: The classification accuracy and CPU time of adding novel classes one by one on Caltech256.

randomly selected form the rest of novel classes with the same scale of inserting data. From the results in Figure 4, we have the following observations:

1) As to accuracy, all algorithms drop with the injection of new classes sequentially, but IFKDA is the most stable with the highest accuracy at the end of insertion.

2)As to CPU time, IFKDA is hundreds of times faster than IKNDA and AKDA/QR, even FKDA is faster than IKNDA.

Furthermore, we test the stability of IFKDA and IKNDA in *known-class recognition* after inserting fifteen novel classes on Caltech256. The testing samples are selected from the rest of initially known classes. The recognition accuracy of IFKDA is 72.35% in the batch training stage, then drops to 71.10% after the final insertion, while the accuracy of IKNDA is 75.01% in the batch training stage, then drops to 49.83% after the final insertion. This result further verifies the stability and robustness of IFKDA.

6 Conclusions

We propose a fast factorization-free kernel method (FKDA) by constructing an approximate kernel centroid matrix to solve a linear system in kernel space. Then the incremental FKDA (IFKDA) runs extremely fast by updating part of matrices. Both theoretical analysis and empirical experiments on several benchmarks validate the superiority of FKDA and IFKDA against the state-of-the-art methods. By taking advantage of our proposed algorithms, the kernel methods may be more widely applied especially when handling incremental learning problems for chunk data streams.

Acknowledgments

This research has been supported by the National Key Research and Development Program of China (Grant No. 2017YFB1103704), and the National Natural Science Foundation of China (Grant No. 61733002 and No.61572096).

A Proof of Theorem 1

The within class scatter in kernel space is

$$\boldsymbol{S}_w{}^{\Phi} = \boldsymbol{X}^{\Phi}\left(\boldsymbol{I} - \begin{bmatrix} \varepsilon_1 & & \\ & \ddots & \\ & & \varepsilon_k \end{bmatrix}\right)(\boldsymbol{X}^{\Phi})^T, \tag{25}$$

where $\varepsilon = \frac{1}{n}ee^T$, $\varepsilon_i = \frac{1}{n_i}e_i{}^T$, $e = [1...1]^T \in \mathbb{R}^{n\times 1}$ and $e_i = [1...1]^T \in \mathbb{R}^{n_i\times 1}$. Thus

$$\begin{aligned}
&(\boldsymbol{G}^{\Phi})^T\boldsymbol{S}_w{}^{\Phi}\boldsymbol{G}^{\Phi} \\
&= (\boldsymbol{G}^{\Phi})^T\boldsymbol{X}^{\Phi}\left(\boldsymbol{I} - \begin{bmatrix} \varepsilon_1 & & \\ & \ddots & \\ & & \varepsilon_k \end{bmatrix}\right)(\boldsymbol{X}^{\Phi})^T\boldsymbol{G}^{\Phi} \\
&= \boldsymbol{E}^T\left(\boldsymbol{I} - \begin{bmatrix} \varepsilon_1 & & \\ & \ddots & \\ & & \varepsilon_k \end{bmatrix}\right)\boldsymbol{E} \\
&= \boldsymbol{E}^T\boldsymbol{E} - \boldsymbol{E}^T\begin{bmatrix} \frac{e_1}{\sqrt{n_1}} & & \\ & \ddots & \\ & & \frac{e_k}{\sqrt{n_k}} \end{bmatrix} \\
&\times \begin{bmatrix} \frac{e_1}{\sqrt{n_1}} & & \\ & \ddots & \\ & & \frac{e_k}{\sqrt{n_k}} \end{bmatrix}^T\boldsymbol{E} \\
&= \begin{bmatrix} n_1 & & \\ & \ddots & \\ & & n_k \end{bmatrix} - \begin{bmatrix} \sqrt{n_1} & & \\ & \ddots & \\ & & \sqrt{n_k} \end{bmatrix} \\
&\times \begin{bmatrix} \sqrt{n_1} & & \\ & \ddots & \\ & & \sqrt{n_k} \end{bmatrix}^T = 0.
\end{aligned} \tag{26}$$

Since $\boldsymbol{S}_t^{\Phi} = \boldsymbol{S}_b^{\Phi} + \boldsymbol{S}_w^{\Phi}$, therefore,

$$(\boldsymbol{G}^{\Phi})^T\boldsymbol{S}_t{}^{\Phi}\boldsymbol{G}^{\Phi} = (\boldsymbol{G}^{\Phi})^T\boldsymbol{S}_b{}^{\Phi}\boldsymbol{G}^{\Phi}. \tag{27}$$

And

$$\boldsymbol{S}_b{}^{\Phi} = \boldsymbol{X}^{\Phi}\left(\begin{bmatrix} \varepsilon_1 & & \\ & \ddots & \\ & & \varepsilon_k \end{bmatrix} - \varepsilon\right)(\boldsymbol{X}^{\Phi})^T \tag{28}$$

We have the following result:

$$
\begin{aligned}
&(\boldsymbol{G}^{\Phi})^T \boldsymbol{S}_b{}^{\Phi} \boldsymbol{G}^{\phi} \\
&= (\boldsymbol{G}^{\Phi})^T \boldsymbol{X}^{\Phi} \left(\begin{bmatrix} \varepsilon_1 & & \\ & \ddots & \\ & & \varepsilon_k \end{bmatrix} - \varepsilon \right) (\boldsymbol{X}^{\Phi})^T \boldsymbol{G}^{\Phi} \\
&= \boldsymbol{E}^T \left(\begin{bmatrix} \varepsilon_1 & & \\ & \ddots & \\ & & \varepsilon_k \end{bmatrix} - \varepsilon \right) \boldsymbol{E} \\
&= \begin{bmatrix} n_1 & & \\ & \ddots & \\ & & n_k \end{bmatrix} - \frac{1}{n} \begin{bmatrix} n_1 \\ \vdots \\ n_k \end{bmatrix} \begin{bmatrix} n_1 & \cdots & n_k \end{bmatrix}.
\end{aligned} \tag{29}
$$

Thus, $rank((\boldsymbol{G}^{\Phi})^T \boldsymbol{S}_b^{\Phi} \boldsymbol{G}^{\Phi}) = k - 1$, which together with (27) yields

$$trace(((\boldsymbol{G}^{\Phi})^T \boldsymbol{S}_t{}^{\Phi} \boldsymbol{G}^{\Phi})^{\dagger} (\boldsymbol{G}^{\Phi})^T \boldsymbol{S}_b{}^{\Phi} \boldsymbol{G}^{\Phi}) = k - 1, \tag{30}$$

so $\boldsymbol{G}^{\Phi}$ is a solution of (7).

References

[Aggarwal *et al.*, 2003] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases*, volume 29, pages 81–92. VLDB Endowment, 2003.

[Bodesheim *et al.*, 2013] Paul Bodesheim, Alexander Freytag, Erik Rodner, Michael Kemmler, and Joachim Denzler. Kernel null space methods for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-13)*, pages 3374–3381, 2013.

[Cai *et al.*, 2007] Deng Cai, Xiaofei He, and Jiawei Han. Efficient kernel discriminant analysis via spectral regression. In *Proceedings of Seventh IEEE International Conference on Data Mining (ICDM -07)*, pages 427–432. IEEE, 2007.

[Chen *et al.*, 2000] Li-Fen Chen, Hong-Yuan Mark Liao, Ming-Tat Ko, Ja-Chen Lin, and Gwo-Jong Yu. A new lda-based face recognition system which can solve the small sample size problem. *Pattern recognition*, 33(10):1713–1726, 2000.

[Chu *et al.*, 2015] Delin Chu, Li-Zhi Liao, Michael Kwok-Po Ng, and Xiaoyan Wang. Incremental linear discriminant analysis: a fast algorithm and comparisons. *IEEE transactions on neural networks and learning systems*, 26(11):2716–2735, 2015.

[Faria *et al.*, 2016] Elaine R Faria, Isabel JCR Gonçalves, André CPLF de Carvalho, and João Gama. Novelty detection in data streams. *Artificial Intelligence Review*, 45(2):235–269, 2016.

[Griffin *et al.*, 2007] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[Huang *et al.*, 2002] Rui Huang, Qingshan Liu, Hanqing Lu, and Songde Ma. Solving the small sample size problem of lda. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 3, pages 29–32. IEEE, 2002.

[Kemmler *et al.*, 2010] Michael Kemmler, Erik Rodner, and Joachim Denzler. One-class classification with gaussian processes. In *Asian Conference on Computer Vision (ACCV-10)*, pages 489–500. Springer, 2010.

[Kim *et al.*, 2011] Tae-Kyun Kim, Björn Stenger, Josef Kittler, and Roberto Cipolla. Incremental linear discriminant analysis using sufficient spanning sets and its applications. *International Journal of Computer Vision*, 91(2):216–232, 2011.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Lampert *et al.*, 2014] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014.

[Liu *et al.*, 2017] Juncheng Liu, Zhouhui Lian, Yi Wang, and Jianguo Xiao. Incremental kernel null space discriminant analysis for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-17)*, pages 792–800, 2017.

[Lu *et al.*, 2017] Yang Lu, Yiu-ming Cheyng, and Yuan Yan Tang. Dynamic weighted majority for incremental learning of imbalanced data streams with concept drift. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 2393–2399, 2017.

[Min *et al.*, 2016] Hwang-Ki Min, Yuxi Hou, Sangwoo Park, and Iickho Song. A computationally efficient scheme for feature extraction with kernel discriminant analysis. *Pattern Recognition*, 50:45–55, 2016.

[Tax and Duin, 2004] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[Wang *et al.*, 2016] Shuo Wang, Leandro L Minku, and Xin Yao. Dealing with multiple classes in online class imbalance learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 2118–2124, 2016.

[Xiao *et al.*, 2017] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[Xiong *et al.*, 2005] Tao Xiong, Jieping Ye, Qi Li, Ravi Janardan, and Vladimir Cherkassky. Efficient kernel discriminant analysis via qr decomposition. In *Advances in neural information processing systems*, pages 1529–1536, 2005.