Face and digit classification with CKFD

Aristotle University of Thessaloniki

Karalias Nikolaos

November 2015

Abstract

In this assignment, we are going to be using the complete Kernel Fisher Discriminant(CKFD) framework introduced by Yang et al. for the purposes of digit and face classification on the databases of MNIST and Olivetti respectively.

1 Introduction

Many prevalent algorithms in statistics and machine learning are most often linear which can be insufficient when dealing with data that has complex non-linear structure. A potential way to alleviate this would be to project the data on to a higher dimensional feature space and execute our linear methods there. Unfortunately, performing this kind of procedure can have a very high computational cost and is therefore impractical in most cases. A potential solution that will allow us to gain some of the benefits of this kind of mapping is the kernel trick. Essentially, it is known from functional analysis that applying a nonlinear kernel(e.g a gaussian) on a dot product in the input space, corresponds to a dot product in a higher dimensional Hilbert space, if a suitable kernel that satisfies certain conditions is selected.[5]

This has led to the formulation of many known algorithms in terms of dot products for the purpose of then utilizing the kernel trick; popular algorithms like SVMs and Kernel PCA which we mentioned earlier. [4]

The main focus here is going to be the CKFD method developed in [1] and its application on digit and face recognition. It is a method that combines the principles of Kernel Principal Component Analysis[2] and Kernel Fisher Discriminant Analysis[3].

2 Preliminaries

We will introduce the main tools that will be required for the CKFD framework.

2.1 Kernel PCA

Given a set of observations $\mathbf{x}_k,\ k = 1, 2, \dots, M, \quad \mathbf{x}_k \in \mathbb{R}^N$ that are guaranteed to be centered, i.e

$$\sum_{k=1}^{M} \mathbf{x}_k = 0, \tag{1}$$

PCA projects the data along the axes of maximal variance, diagonalizing the sample covariance matrix which is defined as

$$\mathbf{C} = \frac{1}{M} \sum_{j=1}^{M} \mathbf{x}_j \mathbf{x}_j^{\mathbf{T}}. \tag{2}$$

This is achieved by solving

$$\lambda \mathbf{v} = \mathbf{C}\mathbf{v}, \tag{3}$$

where the eigenvectors $\mathbf{v}$ corresponding to the largest eigenvalues λ are the the principal components and directions of maximal variance.

We can write

$$\mathbf{C}\mathbf{v} = \frac{1}{M} \sum_{j=1}^{M} (\mathbf{x}_j \cdot \mathbf{v}) \mathbf{x}_j \tag{4}$$

and because all solutions of (3) for $\lambda \geq 0$ must lie in the span of $\mathbf{x}_k$, (3) can be written as

$$\lambda(\mathbf{x}_k \cdot \mathbf{v}) = (\mathbf{x}_k \cdot \mathbf{C}\mathbf{v}), \quad k = 1, 2, \ldots, M. \tag{5}$$

For Kernel PCA the same process can be executed but this time in a higher dimensional feature space $\mathcal{H}$ which is related to the input space by the mapping

$$\Phi : \mathbb{R}^N \to \mathcal{H}, \quad x \to \Phi(x). \tag{6}$$

Equivalently the sample covariance matrix on the feature space is defined as

$$\tilde{\mathbf{C}} = \frac{1}{M}\sum_{j=1}^{M} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^{\mathbf{T}}. \tag{7}$$

Then (5) in the feature space corresponds to

$$\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{u}) = (\Phi(\mathbf{x}_k) \cdot \tilde{\mathbf{C}}\mathbf{u}), \quad k = 1, 2, \ldots, M. \tag{8}$$

where $\mathbf{u}$ are the eigenvectors in the feature space. Finally it can be shown that

$$\mathbf{u} = \sum_{i}^{M} \alpha_i \Phi(\mathbf{x}_i). \tag{9}$$

Now, we define the $M \times M$ **Gram** matrix by

$$\mathbf{K}_{ij} := \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j). \tag{10}$$

By (8) and (9) we can get to

$$M\lambda\boldsymbol{\alpha} = \boldsymbol{K}\boldsymbol{\alpha} \tag{11}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_M]^{\mathbf{T}}$.

Then we can write the projection of a test vector $\mathbf{x}$ on the the k^{th} eigenvector $\mathbf{u}^k$ as

$$\mathbf{u}^k \cdot \Phi(\mathbf{x}) = \sum_{i=1}^{M} \alpha_i^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})) \tag{12}$$

A little caveat must be pointed out here. The data in the feature space can not be centered since we do not have them in explicit form. Luckily, we can express the centered Gram matrix as

$$\tilde{\mathbf{K}} = \mathbf{HKH} \tag{13}$$

where we define

$$\mathbf{H} = \mathbf{I} - \frac{1}{M}\mathbf{1}_M \mathbf{1}_M^{\mathbf{T}} \tag{14}$$

as the centering matrix [6]. $\mathbf{1}_M$ is just an M-dimensional vector of ones. To conclude, we can perform Kernel PCA following these simple steps:

1. Compute the Gram matrix $\mathbf{K}$.
2. Compute the centralized Gram matrix using (13).
3. Compute the solutions to (11), replacing $\mathbf{K}$ with $\tilde{\mathbf{K}}$.
4. Compute the centralized Gram matrix $M \times N$ $\tilde{\mathbf{K}}_\star$ for the test data using:

$$\tilde{\mathbf{K}}_\star = \mathbf{K}_\star - \mathbf{O}_N\mathbf{K}_\star - \mathbf{K}_\star\mathbf{O}_M + \mathbf{O}_N\mathbf{K}_\star\mathbf{O}_M \tag{15}$$

   where

$$\mathbf{O}_L = \frac{1}{L}\mathbf{1}_L\mathbf{1}_L^{\mathbf{T}} \tag{16}$$

5. The projected test data will be:

$$\mathbf{Y} = \tilde{\mathbf{K}}_\star(\mathbf{A}\boldsymbol{\Lambda}^{-1/2}). \tag{17}$$

where $\mathbf{A}$ is the eigenvector matrix and $\boldsymbol{\Lambda}$ the diagonal eigenvalue matrix.

2.2 KFD

The construction for KFD is similar to KPCA. For c known pattern classes the between class scatter operator and the within class scatter operators in feature space are defined as:

$$\mathbf{S}_b^\Phi = \frac{1}{M}\sum_{i=1}^{c} l_i(\mathbf{m}_i^\Phi - \mathbf{m}_0^\Phi)(\mathbf{m}_i^\Phi - \mathbf{m}_0^\Phi)^{\mathbf{T}}. \tag{18}$$

$$\mathbf{S}_w^\Phi = \frac{1}{M}\sum_{i=1}^{c}\sum_{j=1}^{l_i} l_i(\Phi(\mathbf{x}_{ij}) - \mathbf{m}_0^\Phi)(\Phi(\mathbf{x}_{ij}) - \mathbf{m}_0^\Phi)^{\mathbf{T}}. \tag{19}$$

The Fisher linear discriminant in feature space is given by the vector $\boldsymbol{\phi}$ which maximizes

$$J(\boldsymbol{\phi}) = \frac{\boldsymbol{\phi}^{\mathbf{T}}\mathbf{S}_b^\Phi\boldsymbol{\phi}}{\boldsymbol{\phi}^{\mathbf{T}}\mathbf{S}_w^\Phi\boldsymbol{\phi}}, \quad \boldsymbol{\phi} \neq \mathbf{0}. \tag{20}$$

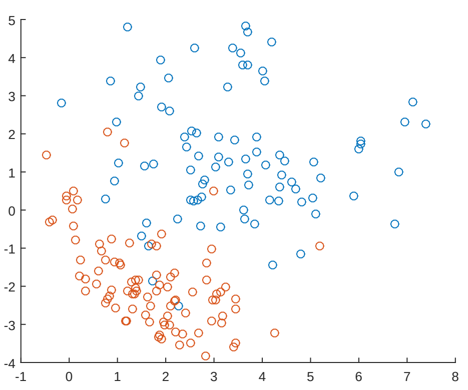
3 The CKFD algorithm

The CKFD algorithm can be summarized in the following steps:

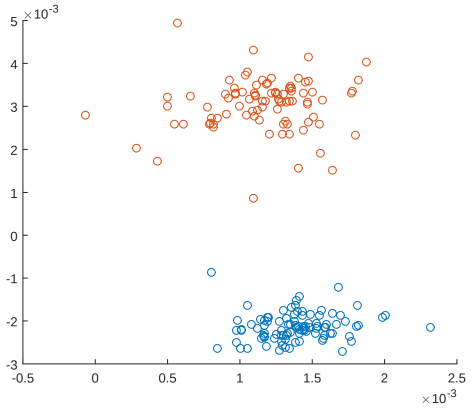
1. Transform the input data into the KPCA feature space $\mathbb{R}^m$, where $m = rank(\tilde{\mathbf{K}})$ using the steps specified in 2.1.
2. Compute $\mathbf{S}_b$ and $\mathbf{S}_w$ in feature space and calculate $\mathbf{S}'_w s$ orthornormal eigenvectors $\mathbf{a_1}, \ldots, \mathbf{a}_m$.
3. Let $\mathbf{P}_1 = (\mathbf{a}_1, \ldots, \mathbf{a}_q)$, $q = rank(\mathbf{S}_w)$. Define $\tilde{\mathbf{S}}_b = \mathbf{P}_1^{\mathbf{T}}\mathbf{S}_b\mathbf{P}_1$ and $\tilde{\mathbf{S}}_w = \mathbf{P}^{\mathbf{T}}\mathbf{S}_w\mathbf{P}_1$. Calculate the generalized eigenvectors $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d]^{\mathbf{T}}$ of

$$\tilde{\mathbf{S}}_b\boldsymbol{\xi} = \lambda\tilde{\mathbf{S}}_w\boldsymbol{\xi}. \tag{21}$$

   The regular discriminant feature vector is $\mathbf{z}^1 = \mathbf{U}^{\mathbf{T}}\mathbf{P}_1^{\mathbf{T}}\mathbf{y}$.



(a) KPCA



(b) CKFD:Regular

Figure 1: Distribution of digits 3 and 8.

4. Let $\mathbf{P}_2 = (\mathbf{a}_{q+1}, \ldots, \mathbf{a}_m)$. Define $\tilde{\mathbf{S}}_b = \mathbf{P}_2^{\mathbf{T}} \mathbf{S}_b \mathbf{P}_2$ and calculate $\tilde{\mathbf{S}}_b$'s orthonormal eigenvectors $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d]^{\mathbf{T}}$. The irregular discriminant feature vector is $\mathbf{z}^2 = \mathbf{V}^{\mathbf{T}} \mathbf{P}_2^{\mathbf{T}} \mathbf{y}$.

5. Perform NN or MD with for a test point $\mathbf{x}$ using the following distance metric:

$$\tilde{g}(\mathbf{z}, \mathbf{z_i}) = \theta \frac{||\mathbf{z}^1 - \mathbf{z}_i^1||}{\sum_{j=1}^{M} ||\mathbf{z}^1 - \mathbf{z}_j^1||} + \frac{||\mathbf{z}^2 - \mathbf{z}_i^2||}{\sum_{j=1}^{M} ||\mathbf{z}^2 - \mathbf{z}_j^2||}. \quad (22)$$

4 Experiments

The experiments were performed on two datasetets; the handwritten digits of MNIST and the Olivetti faces. In both cases 10-fold cross validation was performed. For MNIST 1000 observations were used in each fold. 900 for training and 100 for testing. For the Olivetti faces, there were 360 training observations and 40 test ones in each fold. CKFD with Nearest Neighbor(NN) as well as Nearest Centroid(MD) is compared to nearest centroid and NN using simple and kernel PCA to extract features from the data. In the case of MNIST, the CKFD algorithm is a clear winner over the other two with values for the fusion coefficient around 0.8 producing the best results. Increasing the sigma usually required the increase of the fusion coefficient, otherwise the success rate dropped. For Kernel PCA the maximum amount of dimensions from the feature space was maintained. Dropping dimensions had a negative impact on the classification score. In general we can notice that KPCA does not perform very well even against regular PCA. In figure 1, the digits 3 and 8 are projected on the first two axes corresponding to the heighest eigevalues. We can see the difference in distribution between the digits from KPCA and CKFD:reguarl. The regular CKFD features are way better separated in the first two components. An important advantage of CKFD over regular PCA is that with just 2k samples CKFD could reach classification rates around 96%. For regular PCA+NN to achieve similar results, significantly more samples are required. Therefore CKFD seems to combine the best of both worlds, requiring less data since the number of features grows with the sample size, but at the same time achieving higher success rates. And on these smaller sets the computational overhead from the extra eigendecompositions is manageable.

Table 1: Classification results for the methods on the MNIST handwritten digits.

Method	Classification Rate	(Training + Decision) Time	Kernel	Parameter	Fusion Coefficient
PCA+NN	89.5	0.27029s	-	-	-
PCA+MD	79.5	0.27029s	-	-	-
KPCA+NN	87.9	1.9044s	Gaussian	$\sigma = 5$	-
KPCA+MD	82.5	1.9044s	Gaussian	$\sigma = 5$	-
CKFD + NN	93.2	2.4729s	Gaussian	$\sigma = 5$	$\theta = 0.8$
CKFD + MD	92.6	2.4729	Gaussian	$\sigma = 5$	$\theta = 0.8$

Table 2: Classification results for the methods on the Olivetti faces.

Method	Classification Rate	(Training + Decision) Time	Kernel	Parameter	Fusion Coefficient
PCA+NN	93.7500	0.29125s	-	-	-
PCA+MD	86	0.29125s	-	-	-
KPCA+NN	93.5000	0.76282s	Gaussian	$\sigma = 40$	-
KPCA+MD	86	0.76282s	Gaussian	$\sigma = 40$	-
CKFD + NN	94	0.82828s	Gaussian	$\sigma = 35$	$\theta = 0.75$
CKFD + MD	88	0.82828s	Gaussian	$\sigma = 35$	$\theta = 0.75$

5 Instructions

The directory contains a folder with scripts and a folder with the figures. The experiments run in two stages. Prepare the data. This can be done by either running *Olivetti.m* or *prepMNIST.m*. This will set up the required data structures for the chosen dataset. After that just run *crossv.m*. This script has 3 options. By commenting/uncommenting the corresponding line, the feature extraction/classification method can be picked. The 3 options are the ones that have been reported in the results, namely, CKFD+NN/MD, KPCA+NN/MD, PCA+NN/MD.

References

[1] Yang, Jian, et al. "KPCA plus LDA: a complete kernel Fisher discriminant framework for feature extraction and recognition." Pattern Analysis and Machine Intelligence, IEEE Transactions on 27.2 (2005): 230-244.

[2] Schlkopf, Bernhard, Alexander Smola, and Klaus-Robert Mller. "Nonlinear component analysis as a kernel eigenvalue problem." Neural computation 10.5 (1998): 1299-1319.

[3] Scholkopft, Bernhard, and Klaus-Robert Mullert. "Fisher discriminant analysis with kernels." Neural networks for signal processing IX 1 (1999): 1.

[4] Hofmann, Thomas, Bernhard Schlkopf, and Alexander J. Smola. "A review of kernel methods in machine learning." Mac-Planck-Institute Technical Report 156 (2006).

[5] Smola, Alex J., and Bernhard Schlkopf. "A tutorial on support vector regression." Statistics and computing 14.3 (2004): 199-222.

[6] Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.