Computer Vision Laboratory - Tsukuba University

Monthly report - DEC. 2015

Tony Duong

January 12, 2016

Contents

Introduction

In this monthly report, I will show the work done during the month of December 2015.

After developing an application able to detect faces and to classify them using the Constrained and the Orthogonal Mutual Subspace Methods, I dug deeper in my studies of the subspace methods to use the Kernel Subspace Methods. The application is now also able to capture images from a camera (KINECT) and to use the frames as reference images or input images to be classified.

I will explain in this report in details the different methods I learnt and how my application has been improved.

Part 1 : Preparation

Introduction

After understanding the main subspace methods, I studied the kernel methods which are have a greater classification ability. Before that, I had to understand Kernel Principal Component Analysis (KPCA) which is the base of the Kernel Orthogonal Mutual Subspace Methods (KOMSM) and the Kernel Constrained Mutual Subspace Methods (KCMSM).

Kernel PCA

Kernel PCA is a dimension reduction method which is closely related to PCA. The difference between them is that KPCA is able to handle non-linear data whereas PCA can only with linear data.

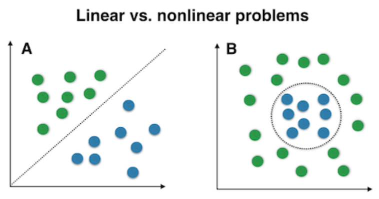

Figure 1 - Example of linear data (left) and non-linear data (right)

The idea is even if in the original space, the cluster of points are scattered in a non-linear pattern, they might form a linear pattern in a higher dimensional feature space. The image below show an example of a mapping from original space to a feature space.

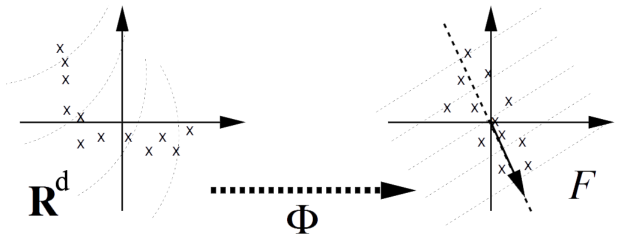

Figure 2 - Original data and data projected after application of kPCA

The original PCA performs linear separation directly in the original space. KPCA, on the other hand, embeds data into high dimensional space (called feature space) first before performing linear separation on the mapped data (which are scattered this time in a linear way). The data is then projected onto low dimension subspace, spanned by eigenvectors of the auto-correlation matrix. But knowing the the mapping Φ nor the feature space is not necessary. A "kernel trick" is going to be used.

I'm going to explain in what way the kernel trick is useful. The normal way would be to map the original d-dimensional features into a larger k-dimensional feature space by creating nonlinear combinations of the original features. For example, if x consists of 2 features:

$x = [x_1\ x_2]\ x \in R^d$

After mapping, x' would be obtained as a result.

$x' = [x_1\ x_2\ x_1 x_2\ x_1^2\ ...]\ x' \in R^k\ k >> d$

But the obvious downside of this method is the computation of the dot product become more and more expensive with increasing sample sizes and number of dimensions. That is why "kernel trick" is useful because it allows the mapping of data onto infinite dimensions with a single function. Many popular kernels such as polynomial kernels exist but in my case, I use the Gaussian kernel which is defined as

$$k(\boldsymbol{x}, \boldsymbol{y}) = e^{\frac{-||\boldsymbol{x}-\boldsymbol{y}||^2}{2\sigma^2}},$$

By using this function between all the data, we obtain a square symmetric matric K where $K_{ij} = k(x_i, x_j)$. After that, the low-dimensional subspace where the data will be projected onto will be the one spanned by the eigenvectors corresponding to the highest eigenvalues of the kernel matrix K.

Kernel Constrained Mutual Subspace Method

In order to understand this method, I read the following paper "3D Object Recognition using KCMSM". After understanding KPCA, KCMSM was easy to understand since the method is based on PCA.

KCMSM is an advanced form of MSM(as well as CMSM and OMSM). The non kernel MSM methods works well when the data distribution of each class can be represented by a linear subspace with no overlap of the distributions. But it is ineffective with highly non-linear structures. KCMSM can both handle non-linear structures of the Kernel method and have the classification ability of CMSM.

Now, I'm going to explain in short how it works. As in CMSM, a generalized difference subspace (GDS) is used, where all the patterns are projected into before computing the similarity.

I'm going to explain briefly how the GDS is generated in the CMSM method. So, the GDS represents the difference among multiple subspaces. It is spanned by the eigenvectors corresponding to the smallest eigenvalues of the matrix

$G = \sum_{i=1}^{k} P_i$

where P_i correspond to the projection matrices $P_i = \sum_{j=1}^{N} \Phi_j^i \Phi_j^i$, Φ_j^i is the j-th orthonormal basis vector of the i-th class subspace.

In the case of KCMSM, we cannot calculate explicitly the Kernel Generalized Difference Subspace(KGDS). The kernel trick is used in order to solve this problem. Indeed, the projection of the mapped pattern onto $\Phi(x)$ onto the orthonormal basis vector d_i^{Φ} can be calculated from an input pattern (coming from the original space) and all the other training patterns.

$$\begin{aligned}(\phi(\mathbf{x}) \cdot \mathbf{d}_i^{\phi}) &= \sum_{j=1}^{r \times N} \sum_{s=1}^{m} \mathrm{b}_{ij} \mathrm{a}_{\eta(j)s}^{\zeta(j)} (\phi(\mathbf{x}_s^{\zeta(j)}) \cdot \phi(\mathbf{x})) \\ &= \sum_{j=1}^{r \times N} \sum_{s=1}^{m} \mathrm{b}_{ij} \mathrm{a}_{\eta(j)s}^{\zeta(j)} k(\mathbf{x}_s^{\zeta(j)}, \mathbf{x})\end{aligned}$$

where b is an eigenvector of the matrix D (shown below) and a is an eigenvector of the kernel matrix K.

$$\begin{aligned}&\mathbf{D}\mathbf{b} = \beta\mathbf{b} \\ &\mathrm{D}_{ij} = (\mathbf{E}[i] \cdot \mathbf{E}[j]), \quad (i, j = 1, \ldots, r \times N)\end{aligned}$$

$$\mathbf{E} = [\mathbf{e}_1^1, \ldots, \mathbf{e}_N^1, \ldots, \mathbf{e}_1^r, \ldots, \mathbf{e}_N^r].$$

e_i^j is the i-th basis vector of the j-th class.

Now that we know how to project our data into the KGDS, what we have to do left is to calculate the similarity between the projected subspaces using MSM.

Kernel Orthogonal Mutual Subspace Method

As explained in the previous report, OMSM consists in orthogonalizing the class subspaces in advance. That way, the classification ability will be higher.

KOMSM, as the KCMSM, uses the kernel method to be effective in case of non-linear distribution. To apply the orthogonalization, a whitening transformation matrix is necessary. Thus, the main task we need to do is constructing this whitening transformation matrix for orthogonalizing the linear subspaces in the feature space. But as in the case of KCMSM, knowing explicitly O is not possible as the mapping function and the feature space are not known. But with the help of kernel trick, the application of the kernel whitening matrix O to the mapped vector $\Phi(x)$ can be calculated from an input vector x and all the other training vectors.

$$(\phi(\mathbf{x}) \cdot \mathbf{O}_{\phi_i}) = \sum_{j=1}^{r\times d}\sum_{s=1}^{l} \frac{\mathrm{b}_{ij}}{\sqrt{\beta_i}} \mathrm{a}^{\zeta(j)}_{\eta(j)s} (\phi(\mathbf{x}) \cdot \phi(\mathbf{x}^{\zeta(j)}_s))$$
$$= \sum_{j=1}^{r\times d}\sum_{s=1}^{l} \frac{\mathrm{b}_{ij}}{\sqrt{\beta_i}} \mathrm{a}^{\zeta(j)}_{\eta(j)s} k(\mathbf{x}, \mathbf{x}^{\zeta(j)}_s) \ .$$

where b is an eigenvector and β is an eigenvalue of the matrix D (shown below) and a is an eigenvector of the kernel matrix K.

$$\mathbf{D}\mathbf{b} = \beta\mathbf{b}$$
$$\mathrm{D}_{ij} = (\mathbf{E}[i] \cdot \mathbf{E}[j]), \quad (i, j = 1, \ldots, r\times N)$$

$$\mathbf{E} = [\mathbf{e}^1_1, \ldots, \mathbf{e}^1_N, \ldots, \mathbf{e}^r_1, \ldots, \mathbf{e}^r_N].$$

e^j_i is the i-th basis vector of the j-th class.

Now after applying this transformation to all the subspaces, what we have to do left is to calculate the similarity between the orthogonalized subspaces using MSM.

Part 2 : Development

Introduction

Some features have been added to the previous face recognition application. The Orthogonal Mutual Subspace Method has been replaced by its kernel counterpart so the classification ability is higher. Not only that, but the application now integrates a camera (I'm using the Microsoft KINECT) that can capture images or create a subspace from frames in real-time.

Training phase

In the previous report, I explained that the training part of the application can be done either from images (the entire process from face detection to generation of the reference subspaces would be done) or from a matlab file "data.m" which contains all the variables necessary for the recognition part (this file is generated each time the training from images is done).

This month, I made use of a camera that could create a subspace from real-time captured images. The button "Start acquisition" launches the process. First, the face detection algorithm is performed on the frame captured. If a face is detected, then a cropping is done and and we append that image on its vector form to a matrix that will be the image matrix. If no face is detected, then just continue to perform the face detection algorithm. At the end, we should get as a result a matrix in which its column vectors are the face images.

This process will stop when the number of face images will reach a determined number of images (set by the user). The user can give also give the name of the person to that subspace as shown in the picture below.

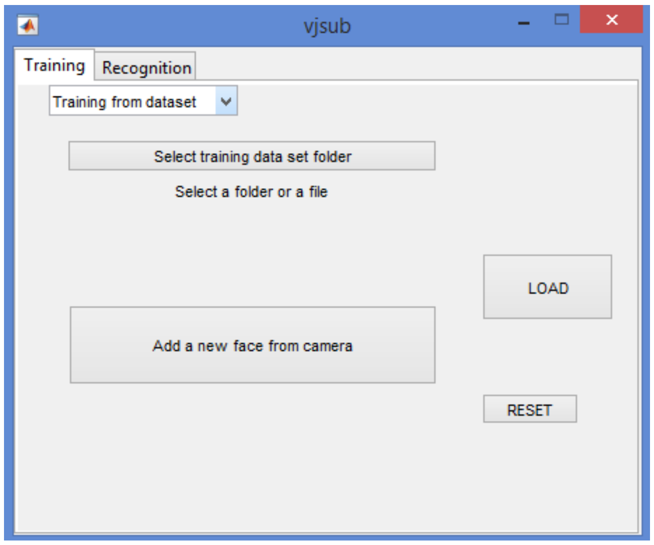
vjsub
Training
Recognition
Training from dataset
Select training data set folder
Select a folder or a file
LOAD
Add a new face from camera
RESET

Figure 3 - GUI

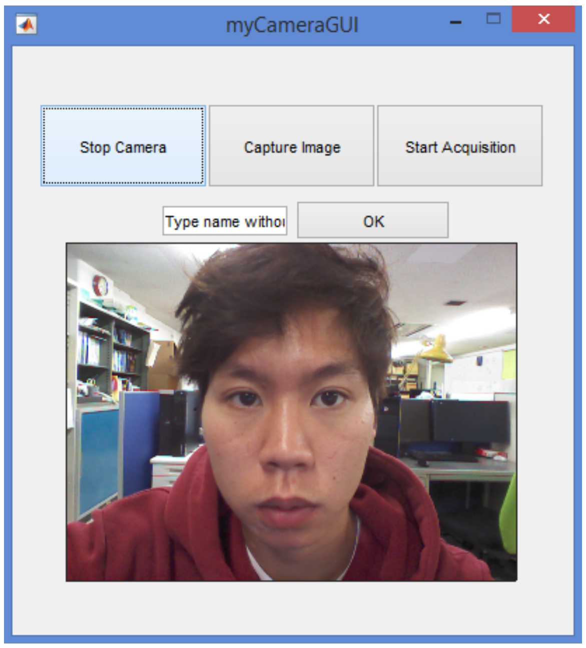

Figure 3 - Camera interface

One problem is that the face detection algorithm can not actually be performed on each frame (it takes around 0.5 seconds for each image). To decrease this time at its most, I have done some modifications such as decreasing the size of the images (24*32 pixels images).

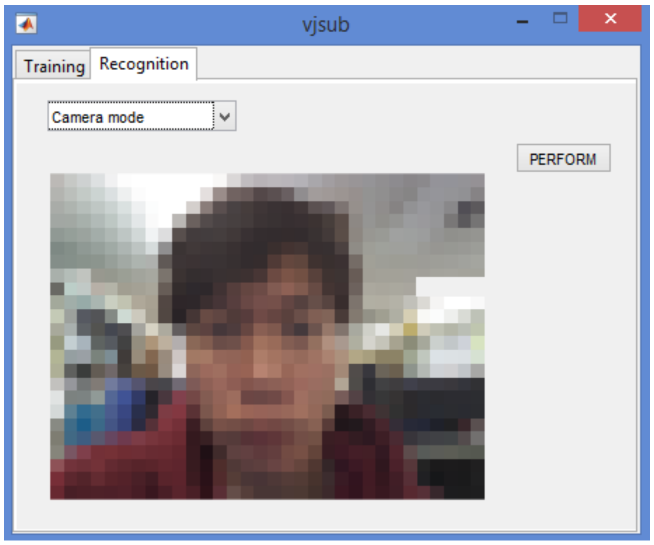

Figure 4 - resized image

Recognition phase

As for the recognition phase, the application is now able to capture an image and use it as a input image. Kernel Orthogonal Mutual Subspace Method is used as the classification method for my application. This method can deal with non-linear face images that can be caused by illumination or difference of angles between images. According to this paper "The Kernel Orthogonal Mutual Subspace Method and its Application to 3D Object" Recognition, its recognition rate is the highest among the subspace methods. I did not test it on a large dataset, but the results seems satisfactory with the small samples that I used.

The same camera interface is used for both recognition phase and training phase.

Part 3 : Difficulties

As expected, after understanding the Kernel PCA method, understanding the main point of the Kernel subspace methods was not so difficult, even though I could not understand the meaning of the mathematical formulas. For example, I know what is the purpose of the whitening matrix which is used for the orthogonalization of the subspaces but I cannot understand how the calculation done to construct this matrix allows it to be useful for orthogonalization. I think, during my internship, that this is one of the biggest problems I have encountered because of my lack of mathematical knowledge.

One other problem is that I don't know if my way of doing is right since I've been working alone even if I sometimes ask help for some problems. I plan to do a short presentation in a meeting to get reviews, comment about my work. I will present my work, what algorithms do I use for it etc.

Part 4 : Conclusion

During this month, I've been closer to my goal. I could use the camera to be part of my application. I learned advanced subspace methods, Kernel Orthogonal Subspace Method and Kernel Constrained Subspace Method. I think they are really useful for my general knowledge about computer vision. The next step now is to test my application on a bigger database to check its performance.

References

[1] FUKUI Kazuhiro, YAMAGUCHI Osamu, STENGER BJORN, A framework for 3D object recognition using the kernel constrained mutual subspace method (2006),
$http : //mi.eng.cam.ac.uk/\ bdrs2/papers/fukui_{a}ccv06.pdf$

[2] YANG Ming-Hsuan, Kernel Eigenfaces vs Kernel Fisherfaces: Face recognition using kernel methods (2002)

[3] YANG Ming-Hsuan, Face recognition using kernel methods (2001),
$http : //www.face - rec.org/algorithms/kernel/nips01.pdf$

[4] OLSSON Daniel, Applications and Implementation of Kernel Principal Component Analysis to Specific Data Sets Master's Thesis Report (2011),
$http : //www.diva - portal.org/smash/get/diva2 : 402792/fulltext01.pdf$

[5] Bernhard SCHOLKOPF, Alexander SMOLA, Klaus-Robert MULLER, Nonlinear Component Analysis as a Kernel Eigenvalue Problem Technical Report (1996),
$http : //www.face - rec.org/algorithms/Kernel/kernelPCA_{s}cholkopf.pdf$

[6] FUKUI Kazuhiro, YAMAGUCHI Osamu, The Kernel Orthogonal Mutual Subspace Method and its Application to 3D Object Recognition (2007),
$http : //www.cvlab.cs.tsukuba.ac.jp/\ kfukui/english/epapers/accv2007.pdf$

[7] OHKAWA Yasuhiro, FUKUI Kazuhiro, Hand Shape Recognition based on Kernel Orthogonal Mutual Subspace Method (2009), $http : //cvlab.cs.tsukuba.ac.jp/\ ohkawa/papers/MVA2009_{P}APER.pdf$