**Machine Learning Homework 7**

**K-means & spectral clustering**

**數據所碩一葉詠富310554031**

**Github:** [**https://github.com/frankye1000/NYCU-MachineLearning/tree/master/HW7**](https://github.com/frankye1000/NYCU-MachineLearning/tree/master/HW7)

I. Kernel Eigenfaces

* Part1: Use PCA and LDA to show the first 25 eigenfaces and fisherfaces, and randomly pick 10 images to show their reconstruction. (please refer to the lecture slides).
* First 25 eigenfaces & fisherfaces

|  |  |
| --- | --- |
| eigenfaces | fisherfaces |
|  |  |

* Random pick 10 images recostruction

|  |  |
| --- | --- |
| eigenfaces | fisherfaces |
|  |  |

* Part2: Use PCA and LDA to do face recognition, and compute the performance. You should use k nearest neighbor to classify which subject the testing image belongs to.
* I try num\_dim:3~135, k: 3~9 to get the best performance.

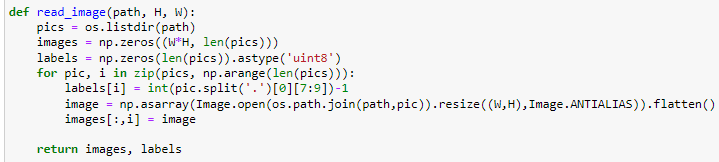
|  |  |
| --- | --- |
| PCA | LDA |
| **num\_dim:18, k:7, acc:90%** | **num\_dim:28, k:9, acc:60%** |

* Part3: Use kernel PCA and kernel LDA to do face recognition, and compute the performance. (You can choose whatever kernel you want, but you should try different kernels in your implementation.) Then compare the difference between simple LDA/PCA and kernel LDA/PCA, and the difference between different kernels.
* In order to compare the difference between the simple LDA and the kernel LDA, I did not change the parameters num\_dim & k.
* We can find the kernel PCA have a little worse performance than simple PCA. The RBF kernel is better than polynomial kernel.
* We can find the kernel LDA have the worse performance than simple LDA. The RBF kernel is better than polynomial kernel.

|  |  |  |
| --- | --- | --- |
| kernel | kernel PCA | kernel LDA |
| RBF | num\_dim:18, k:7, acc:86.67% | num\_dim:28, k:9, acc:56.67% |
| polynomial | num\_dim:18, k:7, acc:83.33% | num\_dim:28, k:9, acc:46.67% |

**Code:**

* 1. Read image.

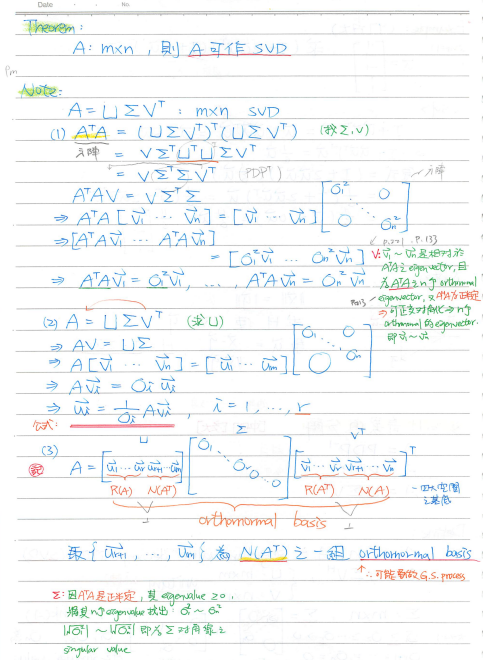


* 1. Print face & reconstruction result.



* 1. PCA

I use the concept of SVD(U@Sigma@V.T) to calculate PCA.



I use five steps to get PCA.

step1: around center.

step2: X.T@X to caculate sigma&V.

step3: sort eigenvalue big -> small.

step4: delete eigenvalue<0

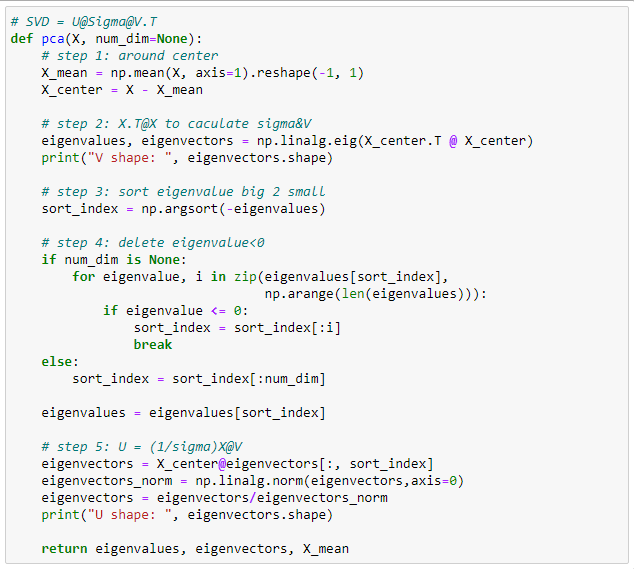
step5: U = (1/sigma)X@V

We can get:

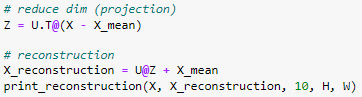
X shape:(45045, 135)

V shape:(135, 135)

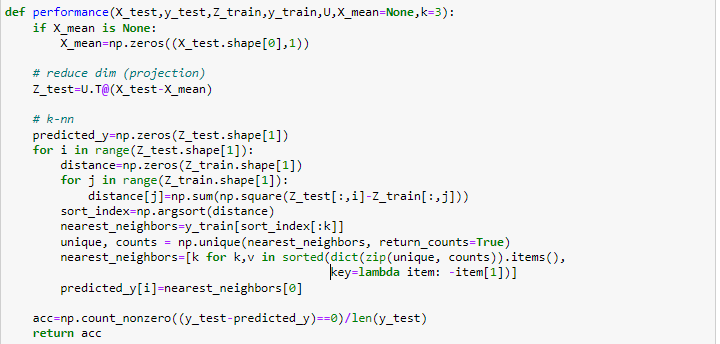
U shape:(45045, num\_dim) # can set num\_dim



Reconstruction and print result.



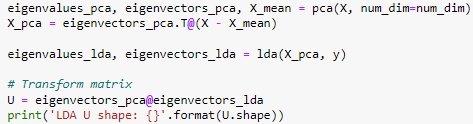
Use KNN to compare the performance.

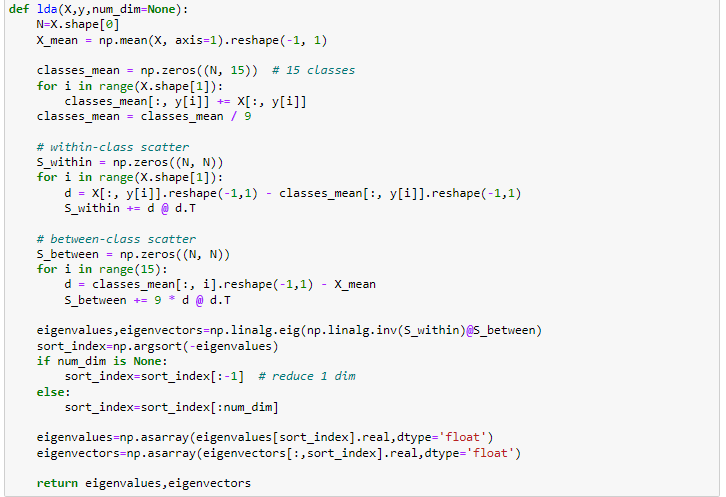


* 1. LDA

Because of the singularity of the within-class scatter matrix. First I use pca to reduce dimension to 28, then use lda to reduce dimension to 27. It can get the best performance acc:60%.

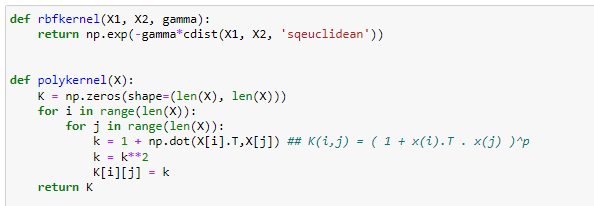
The transformation matrix U is eigenvector of PCA @ eigenvector of LDA.





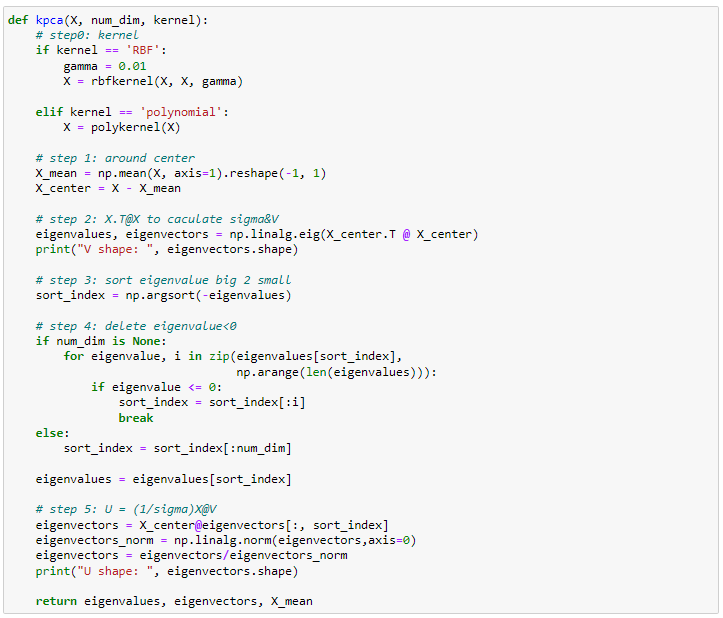
* 1. Kernel

Define rbf kernel & polynomial kernel.



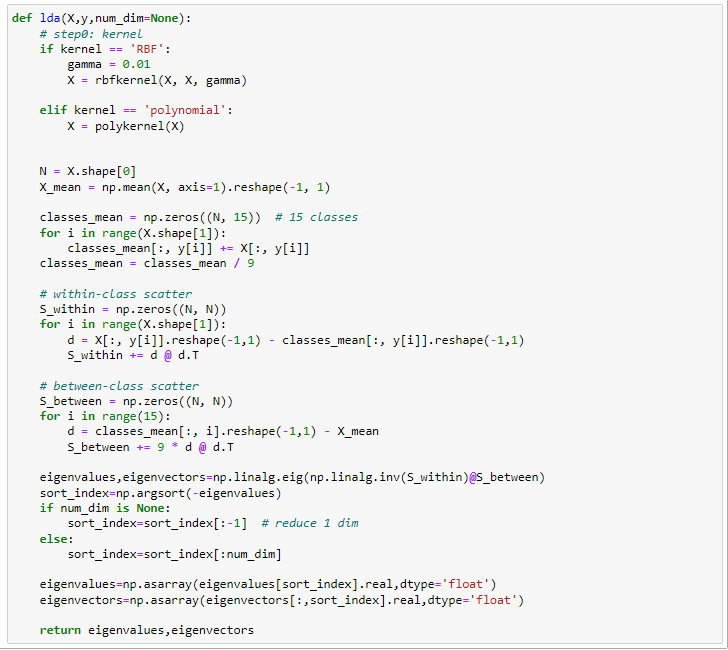
* 1. Kernel PCA

Through the kernel PCA, we perform a non-linear correspondence function to transform the data into a higher-dimensional space, and in this high-dimensional space, use standard PCA, and then project the data back to a lower-dimensional subspace.



* 1. Kernel LDA

Also through the kernel LDA, we perform a non-linear correspondence function to transform the data into a higher-dimensional space, and in this high-dimensional space, use standard LDA, and then project the data back to a lower-dimensional subspace.



II. t-SNE

* Part1: Try to modify the code a little bit and make it back to symmetric SNE. You need to first understand how to implement t-SNE and find out the specific code piece to modify. You have to explain the difference between symmetric SNE and t-SNE in the report (e.g. point out the crowded problem of symmetric SNE).
* t-SNE & symmetric SNE is different in  **& gradient**. t-SNE uses t- distribution. symmetric SNE uses Gaussian distribution.
* We can find , so we modify num.

|  |  |
| --- | --- |
| t-SNE | symmetric SNE |
|  |  |
|  |  |
|  |  |
|  |  |

* The **crowded problem** means that the clusters are gathered together and cannot be distinguished(fig.1).
* The distance relationship in high-dimensional data cannot be completely retained in low-dimensional space.
* Part2: Visualize the embedding of both t-SNE and symmetric SNE. Details of the visualization:
  + Project all your data onto 2D space and mark the data points into different colors respectively. The color of the data points depends on the label.
  + Use videos or GIF images to show the optimize procedure.

perplexity=20

|  |  |
| --- | --- |
| t-SNE | symmetric SNE |
| (fig.1) | (fig.2) |
| HW7\tsne\_python\GIF\tsne.gif | HW7\tsne\_python\GIF\ssne.gif |

* Part3: Visualize the distribution of pairwise similarities in both high-dimensional space and low-dimensional space, based on both t-SNE and symmetric SNE.
* The range of ssne in low-dim is much smaller than that of tsne, so ssne has a **crowded problem**.

|  |  |  |
| --- | --- | --- |
|  | t-SNE | symmetric SNE |
| High-dim |  |  |
| Low-dim |  |  |

* Part4: Try to play with different perplexity values. Observe the change in visualization and explain it in the report.
* We observe a tendency towards clearer shapes as the perplexity value increases.

|  |  |  |
| --- | --- | --- |
| perplexity | t-SNE | symmetric SNE |
| 10 |  |  |
| 20 |  |  |
| 40 |  |  |

III. reference

[1] <https://www.youtube.com/watch?v=g-Hb26agBFg&ab_channel=Serrano.Academy>

[2] <https://kknews.cc/code/a6olrln.html>

[3] <https://www.uj5u.com/houduan/294222.html>

[4] <https://kknews.cc/code/o2gbm45.html>

[5] <https://arbu00.blogspot.com/2017/02/7-kernel-pca.html>