

# Homework 2

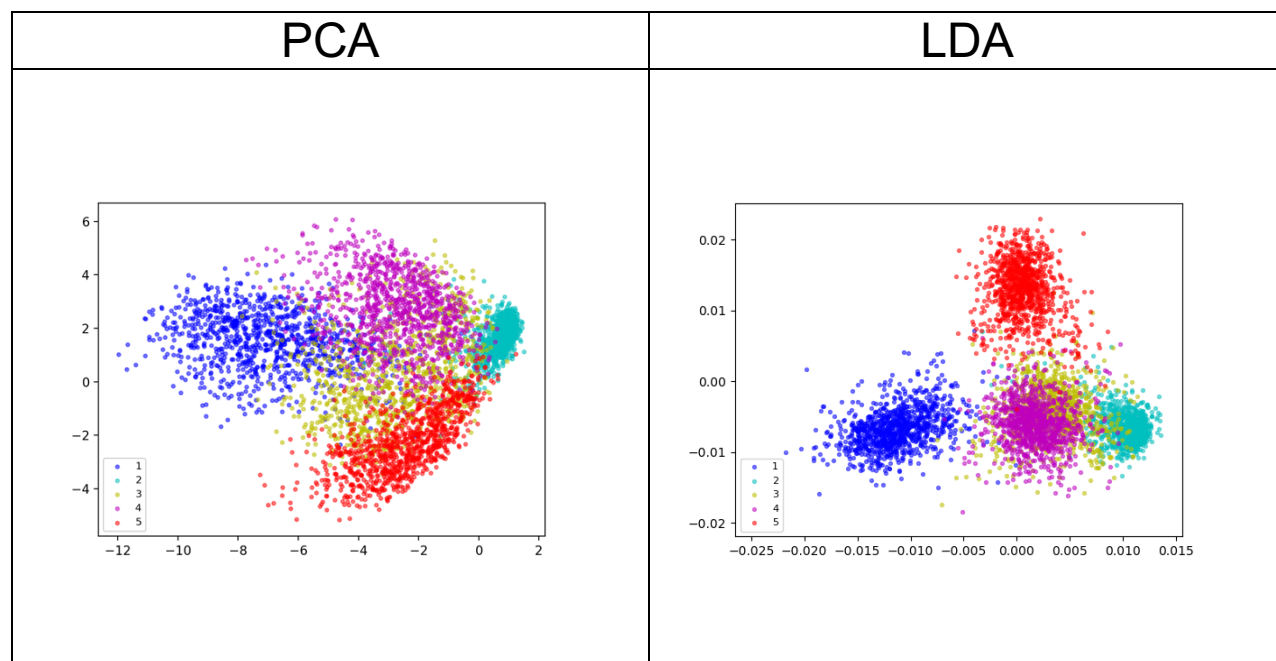
0556623

Chieh Yu, Chen

陳傑宇

## 1. Visualization

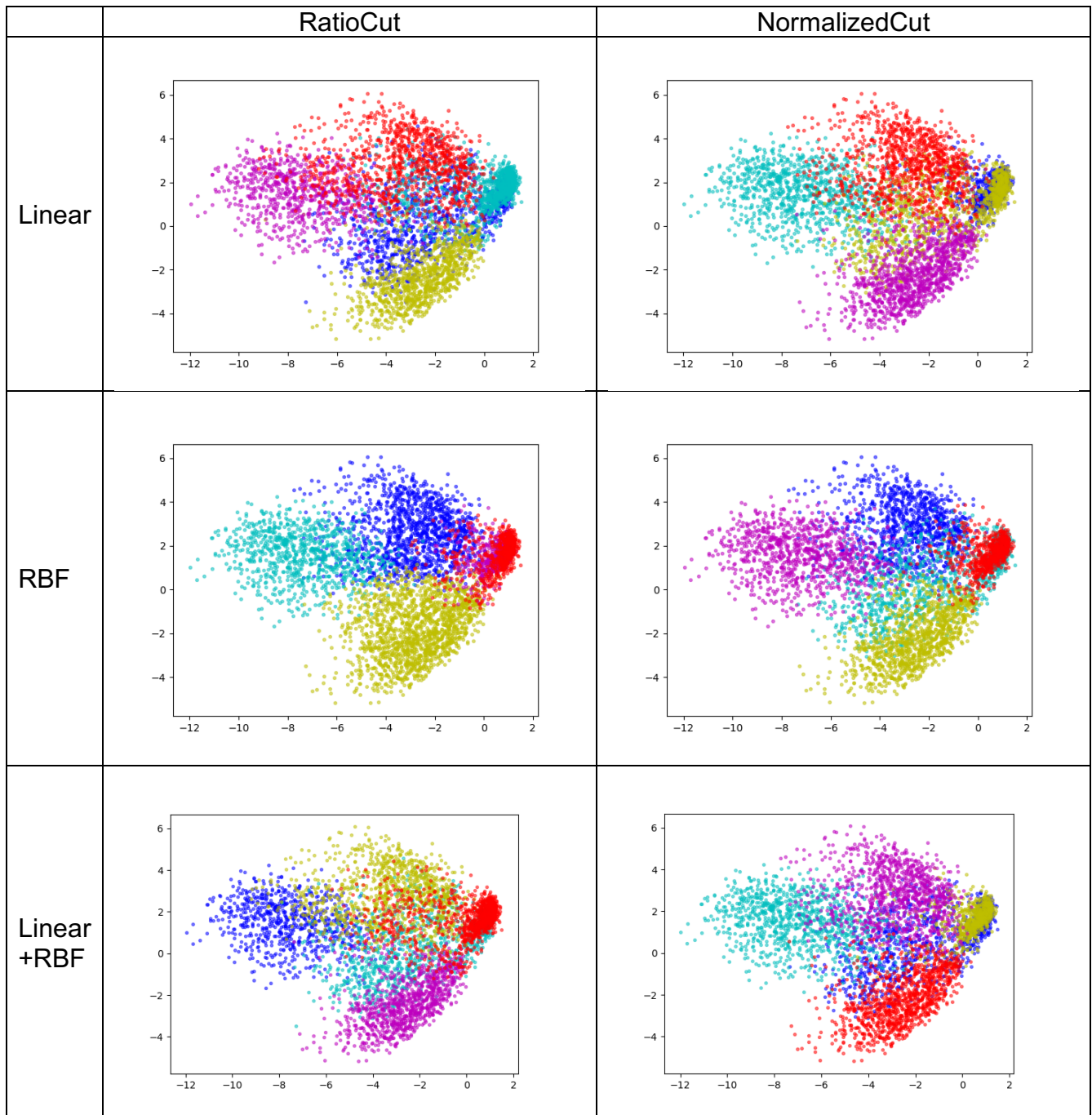
- a. Use PCA to project all your data X\_train.csv onto 2D space . Use LDA to project all your data X\_train.csv onto 2D space.



- Observation and discussion :

I visualize all training data with different colors and show their corresponding labels on left-bottom corner. Visually, I can tell LDA results in more compact cluster than PCA. I think this phenomenon is consistent to the algorithm, because the LDA not only maximize the distance between cluster, but also minimize the variance in between cluster.

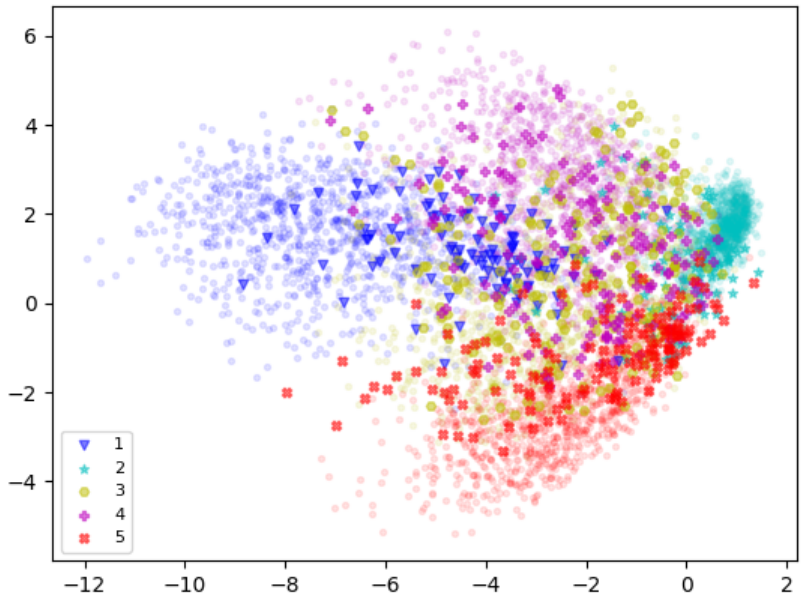
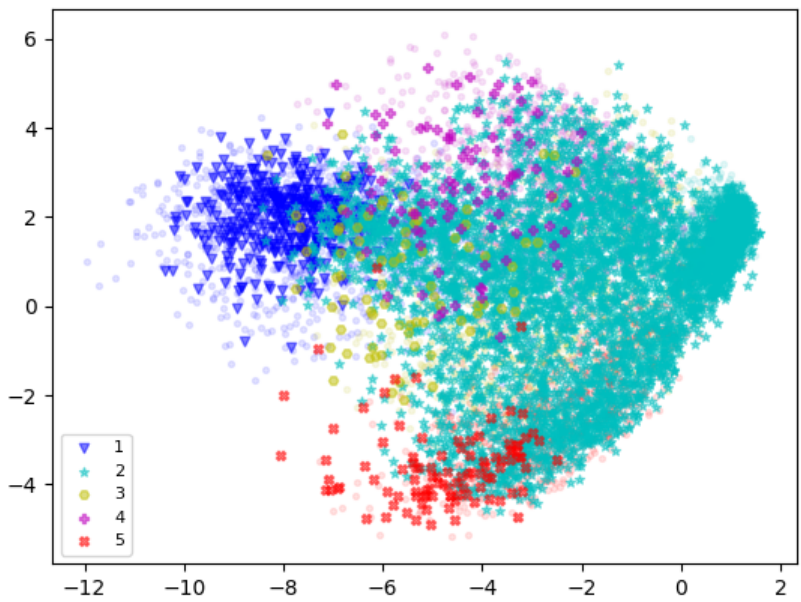
b. Use different colors to show clusters obtained by RatioCut and NormalizedCut in 2D space (PCA projection).



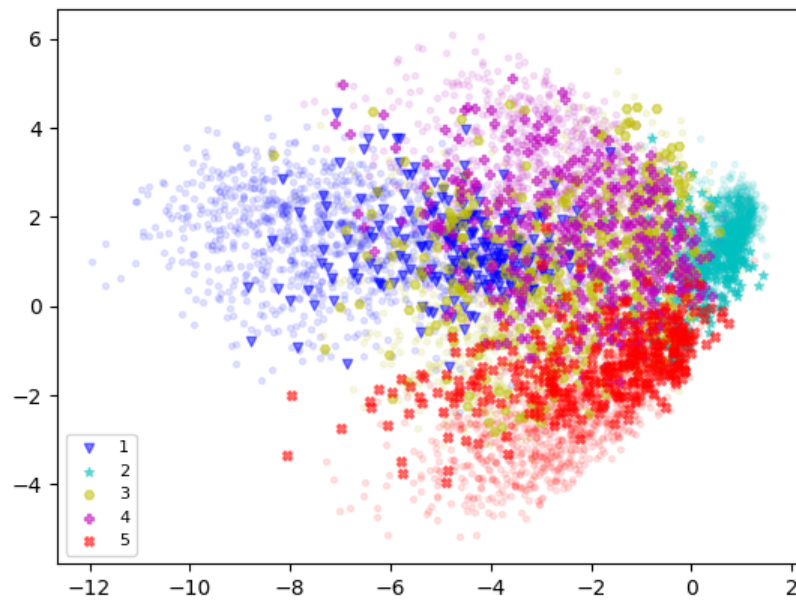
- Observation and discussion :

Visually, RatioCut and NormalizedCut perform about the same quality with linear kernel, but NormalizedCut is better than RatioCut with RBF kernel, we can see RatioCut mis-classify some data resulting in only few purple points in the figure.

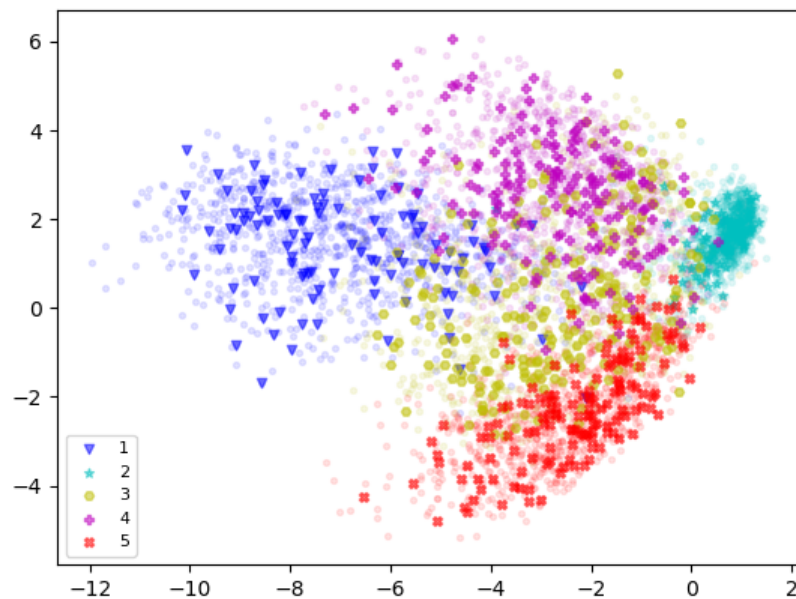
- c. With all the data samples are shown by “dots”, the “support vectors” that you obtained from homework#1 should be shown with different symbols, e.g. square, triangle, cross. 4 figures in total (linear, polynomial, RBF, linear+RBF)

Kernel	Visualization
Linear	 <p>A scatter plot for the Linear kernel. The x-axis ranges from -12 to 2, and the y-axis ranges from -4 to 6. There are five classes of data points: Class 1 (blue downward triangles), Class 2 (cyan stars), Class 3 (yellow circles), Class 4 (magenta crosses), and Class 5 (red squares). The points are distributed in a way that shows some overlap between classes, but the support vectors are clearly marked with their respective symbols.</p>
Polynomial	 <p>A scatter plot for the Polynomial kernel. The x-axis ranges from -12 to 2, and the y-axis ranges from -4 to 6. The same five classes of data points are shown as in the Linear plot. The distribution of points is different due to the polynomial transformation, with Class 2 (cyan stars) appearing more concentrated on the right side of the plot. Support vectors are again highlighted with their respective symbols.</p>

RBF



Linear+RBF



- Observation and discussion :

I visualize all support vectors with different symbols according to their labels, we can see support vectors distributed uniformly in the pca subspace, and visually the linear+RBF kernel outperform others.

## 2. My Implementation Details

- a. <data\_loader.py>  
as in homework 1, the code transfer data from .csv into .json format.
- b. <pca.py>
  - In function `pca_fit_n_transform()`, my algorithm is shown as follows:
    - i. Computing the Covariance Matrix
    - ii. Computing eigenvectors and corresponding eigenvalues
    - iii. Sorting the eigenvectors by decreasing eigenvalues
    - iv. Choosing k eigenvectors with the largest eigenvalues
    - v. Transforming the samples onto the new subspace
  - `pca_plot()` is for visualization of spectral clustering.
  - `pca_plot_with_svm()` is for visualization of SVM, drawing different symbols for support vectors.
- c. <lda.py>
  - In function `lda_fit_n_transform()`, my algorithm is shown as follows:
    - i. Calculate means for each class
    - ii. Compute the Covariance Matrix between classes
    - iii. Compute the Covariance Matrix within classes
    - iv. Computing eigenvectors and corresponding eigenvalues
    - v. Sorting the eigenvectors by decreasing eigenvalues
    - vi. Choosing k eigenvectors with the largest eigenvalues
    - vii. Transforming the samples onto the new subspace
- d. <spectral\_slustering.py>
  - In function `spectral_clustering()`, my algorithm is shown as follows:
    - i. construct similarity graph A (affinity matrix) by kernel function.
    - ii. compute Laplacian Cut
      1. mode==NCUT:  $L = D^{**(-1/2)} A D^{**(-1/2)}$
      2. mode==RCUT:  $L = D - A$
    - iii. compute the first k eigenvector of Laplacian Cut, U contains k eigenvectors
      1. mode==NCUT: get T by normalizing all rows of U to norm 1
      2. mode==RCUT: return U
    - iv. cluster all points with k-means algorithm in k-dims (k eigenvectors) space.
  - In function `main()`, I enumerate 6 combinations of pca projection between 2 laplacian cuts and 3 kernel functions, and save those figures as image files.
- e. <plot\_svm\_support\_vector.py>
  - In function `test_kernel()`:

- I deal with three kernels in this function including linear kernel, polynomial kernel, and RBF kernel.
- First get the support vectors through libsvm api. I feed the support vectors into model again to get their predicted result for better visualization. Then transform support vector from sparse format to dense array. Finally, I can use function `pca_plot_with_svm()` to visualize training data and support vectors in PCA subspace.
- In function `precomputed_kernel()`:
  - I deal with the customized kernel (RBF+Linear kernel).
  - Alike HW1 precompute the gram matrix at first. Then get the support vectors in similarity format through libsvm api, and use the indices to fetch the support vectors from training data. Finally, I can use function `pca_plot_with_svm()` to visualize training data and support vectors in PCA subspace.

## Reference

1. [http://sebastianraschka.com/Articles/2014\\_pca\\_step\\_by\\_step.html](http://sebastianraschka.com/Articles/2014_pca_step_by_step.html)
2. <http://goelhardik.github.io/2016/10/04/fishers-lda/>
3. [https://en.wikipedia.org/wiki/Spectral\\_clustering](https://en.wikipedia.org/wiki/Spectral_clustering)