COGS118B Final Project

Multiview Spectral Clustering on Picture Recognition

Xiaofei Teng
A16163740
xteng@ucsd.edu

Abstract

This project works on clustering pictures of handwritten numerals using the spectral clustering algorithm, both single-view and multi-view version. In the single-view spectral clustering, I used the Ng, Jordan, Weiss spectral clustering (Ng et al., 2001) as taught in class. Its perform varies for different features of the patterns that the patterns are clustered by, as well as different numerals that are being clustered. In hope of an improving clustering result, I applied the two-view clustering algorithm (de Sa et al., 2005). In general, this two-view algorithm slightly improves the result, but there are also certain constraints to it.

Introduction

As we learned in this course, spectral clustering is an unsupervised machine learning algorithm that could be used the classify data based on the similarity between each data-pairs. It can also be used to solve clustering problems that are undoable through Kmeans. An implementation of the algorithm is to cluster pictures according to their certain features. However, in real-world settings, one feature could be insufficient for the algorithm to perform a good clustering on certain datasets that is similar in that particular feature (for example, handwritten numeral "6" and handwritten numeral "9" is very similar on their Fourier coefficients). Therefore, it provides me with a strong motivation to apply multi-view clustering algorithm in this project and to see if adding additional features to the algorithm could improve the clustering results.

The dataset I used for this project is from the UCI Machine Learning Repository. It consists of features of handwritten numerals ('0'-'9') extracted from Dutch utility maps. There are 2000 patterns in total, which consists of 200 patterns for each numeral in the order from 0 to 9. The digits are represented in six different features in six different sub-datasets:

1. mfeat-fou: 76 Fourier coefficients of the character shapes;
2. mfeat-fac: 216 profile correlations;
3. mfeat-kar: 64 Karhunen-Love coefficients;
4. mfeat-pix: 240 pixel averages in 2 x 3 windows;
5. mfeat-zer: 47 Zernike moments;
6. mfeat-mor: 6 morphological features.

I did some background research on what these six features are and how they could represent a picture (or digit).

Fourier coefficients are the coefficients of a Fourier series, which is a summation of harmonically related sinusoidal function. One can think of the coefficients as a description of how the shape of the function changes. In general, a Fourier series with infinite terms can converge to any real functions, thus can represent any shape. In this dataset, we are only using the first 76 Fourier coefficients. Notice that the Fourier Coefficients are roughly "relative", to the shape, which makes it extremely insensitive to rotations. Thus, one could expect that spectral clustering based purely on this feature would perform poorly on differentiating numeral '6' and '9'.

Profile correlations represent edge crosses of all possible pairwise combinations of linear independent entries. It works similar to how our visual cortex recognize pictures through frequency difference, but only on a very basic level.

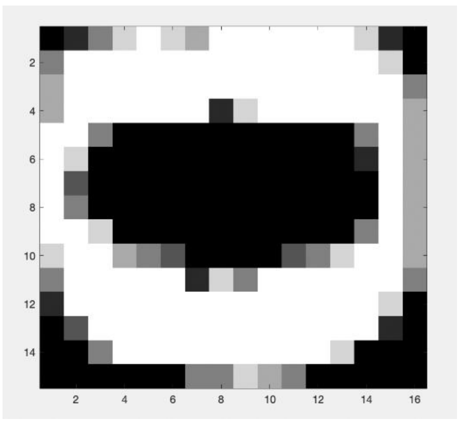
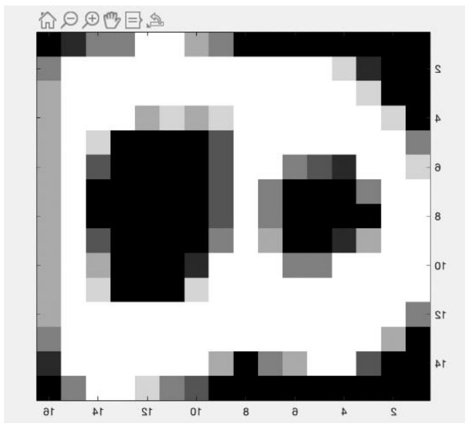
Karhunen-Love coefficients comes from Kosambo-Karhunen-Loeve Theorem, which is a representation of a stochastic process as an infinite linear combination of orthogonal functions, analogous to a Fourier series representation of a function on a bounded interval. This, theoretically, could represent any shape with infinite terms as well. Here we are just using the first 64 coefficients from the series. Because of its similarity to Fourier coefficients, I predict that it will have a hard time distinguishing '6' and '9'.

Pixels are just what we usually use for representing pictures. In this dataset, mfeat-pix is the only dataset that I can reconstruct the original pattern using the entries of the dataset.

Zernike polynomials are sets of orthogonal polynomials defined on the unit disk. Zernike moments are simply the projection of the image function onto these orthogonal basis functions. One can theoretically reconstruct any digital images from Zernike moments. Because of its similarity to Fourier coefficients, I predict that it will have a hard time distinguishing '6' and '9'.

Morphological features are features of the outward shape or appearance of the original pattern. We do not know how it is measures in this specific dataset.

Using the viewcolumn.m script given by HW4 on the transpose of mfeat-pix dataset (because in this dataset the pattern in stored as rows), and set the output picture to be 15*16, we can reconstruct the original pattern. Here are some examples of the reconstructed patterns:

This is the numeral '0' and the numeral '8'. Note that all the patterns are consist of digits rotated 90 degrees clockwise.

Method

First, I applied the single-view Ng, Jordan Weiss spectral clustering (Ng et al., 2001) on all six features to get a more specific understanding of hoe different feature affect the result of the clustering. For each feature, I ran a test on the first 50 patterns of ‘0’ and the first 50 patterns of ‘1’ as a starter, then ran a more general test on the 50 patterns of every single numeral. I plotted the similarity matrix W, the Laplacian, the top eigenvectors X, and the normalized top eigenvectors Y respectively, and from the result of this part, I tried to adjust sigma square accordingly (because the datasets are all of different dimensions and the entries are of different magnitudes, different sigma squares should be applied to different features).

Then, with the acquired sigma square from previous step, I ran the single-view clustering algorithm specifically on digit pairs that could be hard for the algorithm to differentiate, namely ‘1’ and ‘7’, ‘2’ and ‘3’, ‘6’ and ‘9’, and plotted the results respected to different digit pairs and different features.

For the next step, I implemented the two-view spectral clustering algorithm (de Sa et al., 2005). Similar to the single-view, it builds a similarity relationship between each data point pairwise, but instead of simply fitting one feature into the Gaussian similarity function, this time we multiply distances in both features to the power of natural exponential, thus the similarity function of the two-view clustering algorithm is:

$$
\begin{aligned}
w_{ij} &= \sum_{p} e^{-\frac{||(x_i^{(1)} - x_k^{(1)})||^2}{2\sigma_1^2}} e^{-\frac{||(x_j^{(2)} - x_k^{(2)})||^2}{2\sigma_2^2}} \\
&= [A_{v1} \times A_{v2}]_{ij}
\end{aligned}
$$

We could simply get the two-view similarity matrix W by applying a cross-product of the similarity matrix in each of the single-view version. Then we enlarge the matrix to be a larger 2N*2N matrix with zero matrices on the diagonal, meaning that within a feature, the distance is 0, and W and W’ on the upper-right and lower-left corner, representing the distance between two features:

$$
A_{sM-D} = \begin{bmatrix} 0_{p\times p} & W \\ W' & 0_{p\times p} \end{bmatrix}
$$

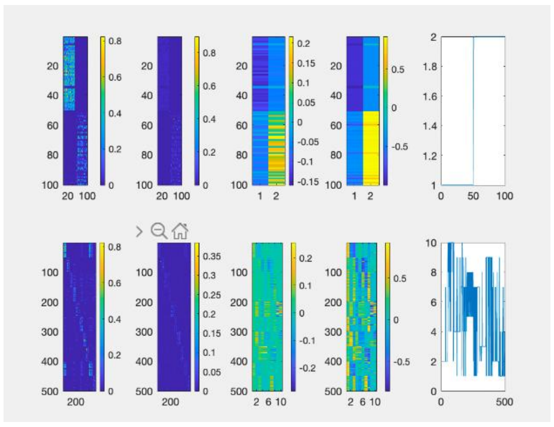
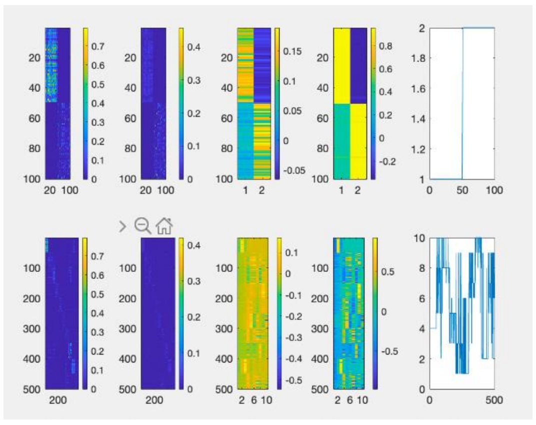
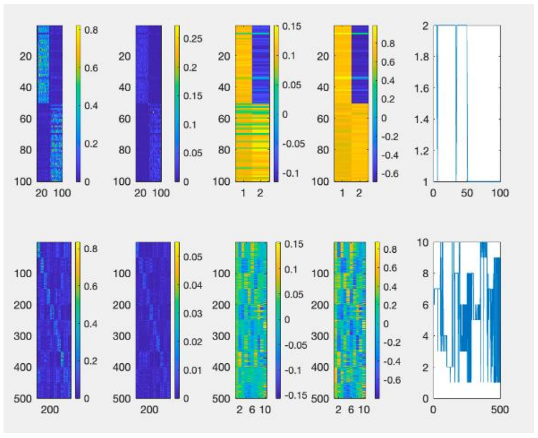
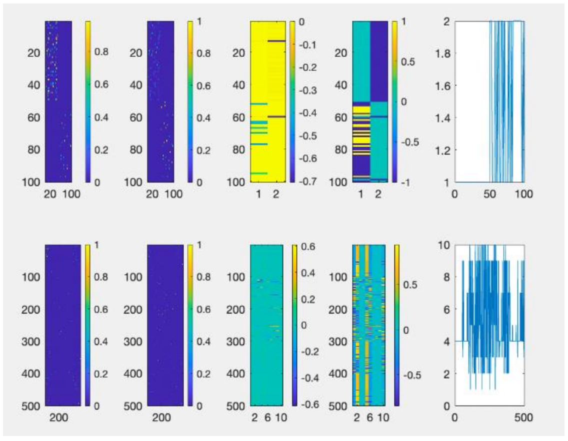
This matrix AsM-D could then be used as the Laplacian and feed into later procedures of the Ng, Jordan, Weiss spectral clustering algorithm. Notice that with a 2N*2N Laplacian, X and Y, which contains the top eigenvectors, would also double in size, namely 2N*2K, where N is the number of patterns in each feature and K is the total number of clusters. The first N rows of X (and Y as well) are the eigenvectors with respect to the first view, and the second N rows vice versa. We have to add weight to both halves that add up to 1 before performing Kmeans on Y. For example, if we determine that both views are equally important, then the formal Y should be 0.5*Y(1:N,:)+0.5*Y(N+1:2N,:), while if we decide that one view should be more significant than the other, we should then tilt the weight towards that particular view accordingly.

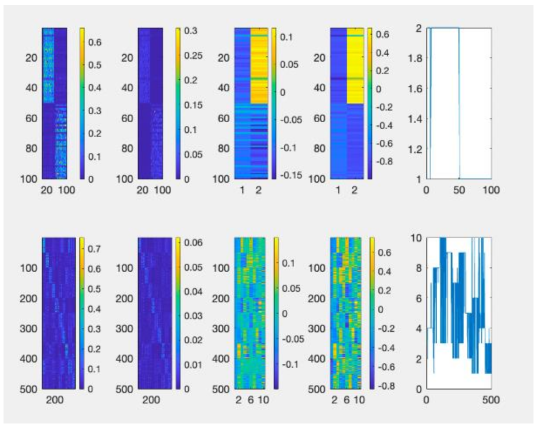
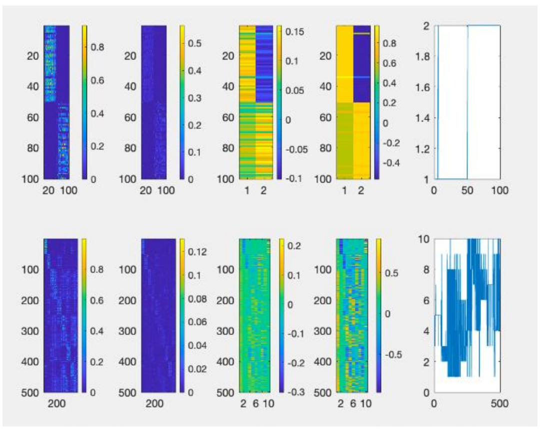
Then I performed the two-view clustering algorithm on the numeral pairs I am interested in, namely '1'&'7', '6'&'9' with the same sigma squares respective to each feature, and manipulated the weight of the two views accordingly to see if there is an improvement in the clustering result for potentially bad results from the single-view clustering algorithm.

Result

(All the results below are in the format of similarity matrix, Laplacian, X, Y, K means result of Y, from left to right).

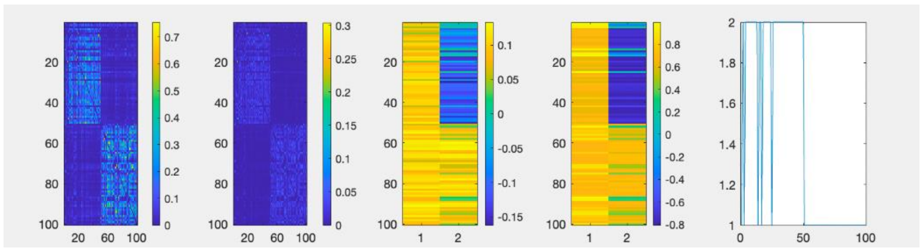
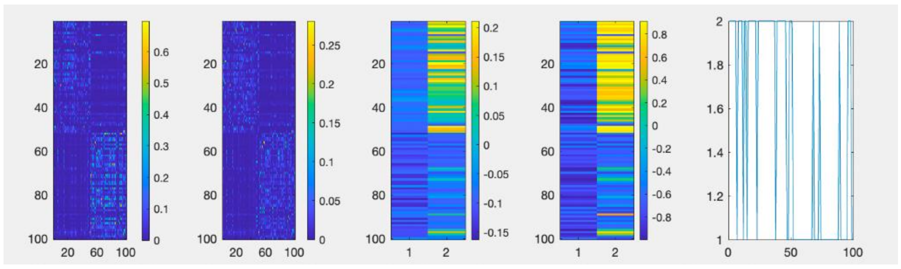
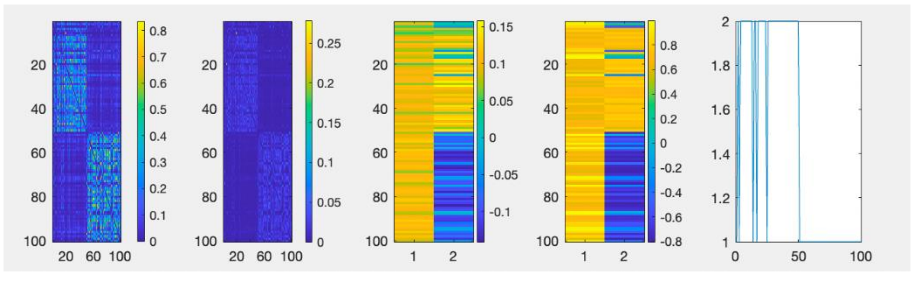
Below are the results of the setup which I used to decide proper sigma square for each of the six features:

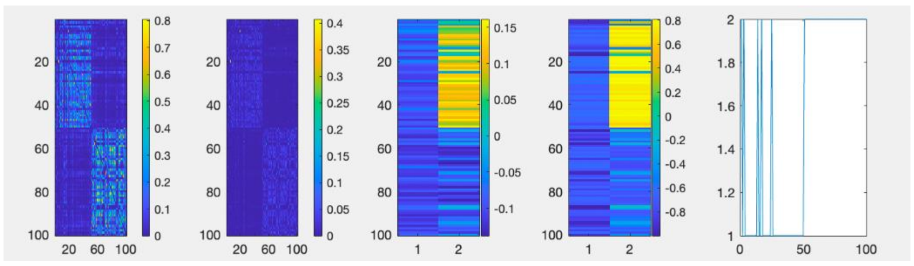





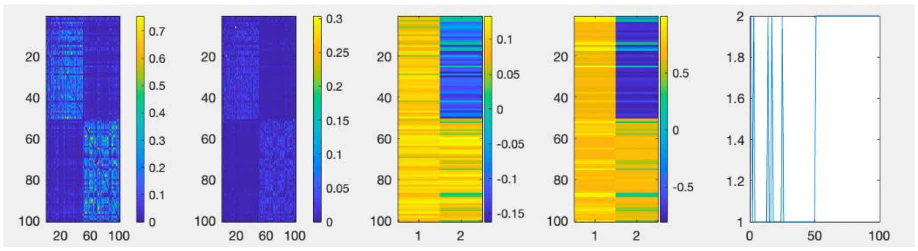



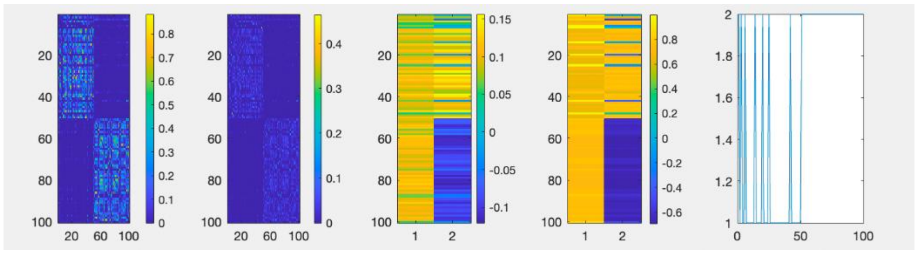
The order is mfeat-fac, mfeat-fou, mfeat-kar, mfeat-mor, mfeat-pix, mfeat-zer, from left to right and from up to below. From the upper-left two plots of each result, we can see that the sigma square is proper for comparing ‘0’ and ‘1’ (I phrase it like this because it is possible that there shou be different sigma squares for different numeral comparison even within the same sub-dataset, or the same feature, but for convenience of the project, I made all the sigma squares the same for the same feature). The sigma squares I are 200,000 for mfeat-fac. 0.12 for mfeat-fou, 240 for mfeat-kar, 200 for mfeat-mor, 750 for mfeat-pix, and 55,000 for mfeat-zer. From the first row of each plot, we see that except for mfeat-mor, all the features did pretty good job on differentiating between ‘0’ and ‘1’. From the second line of each plot, we see that none of the six features did a great job in distinguishing all ten digits.

Below are the results of comparing ‘1’ and ‘7’ of the six features using the above sigma squares, in the order of mfeat-fac, mfeat-fou, mfeat-kar, mfeat-mor, mfeat-pix, mfeat-zer.

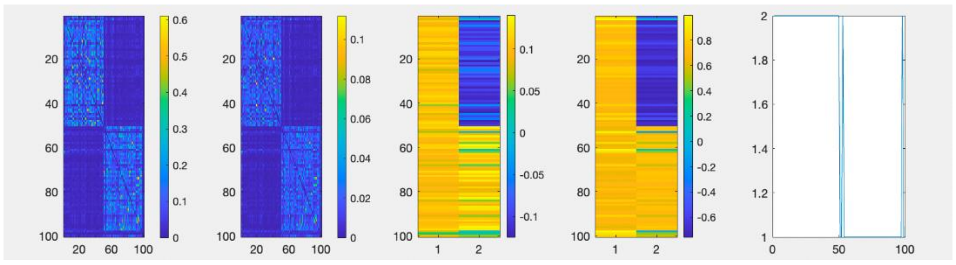




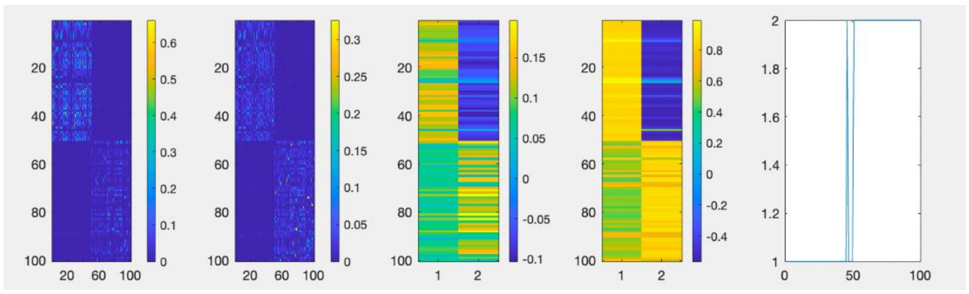


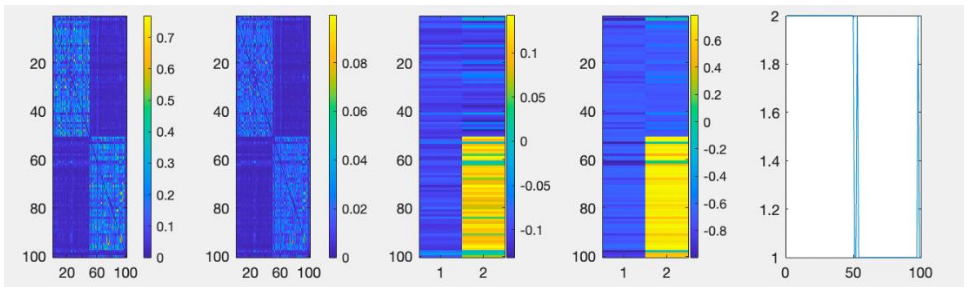


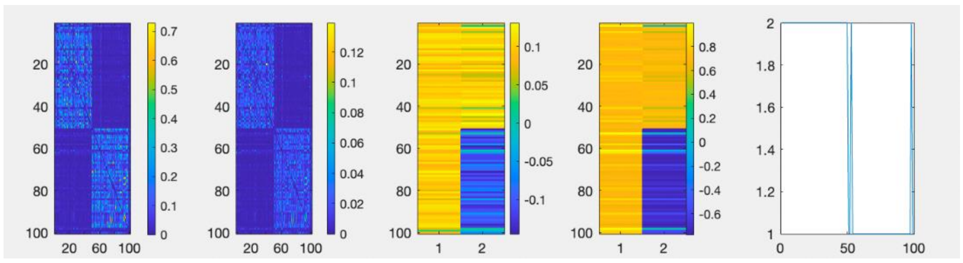


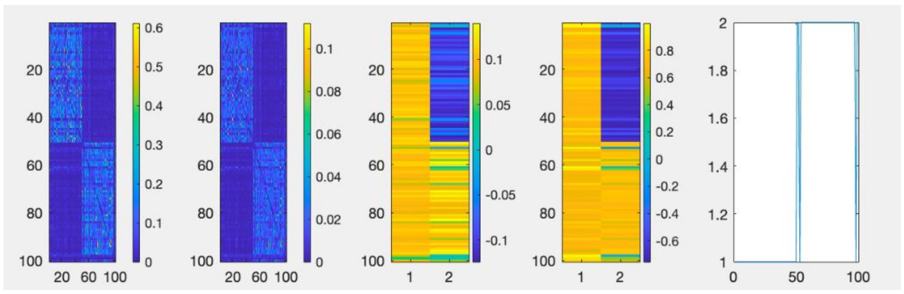
Below are the results of comparing ‘2’ and ‘3’ of the six features using the above sigma squares, in the order of mfeat-fac, mfeat-fou, mfeat-kar, mfeat-mor, mfeat-pix, mfeat-zer.

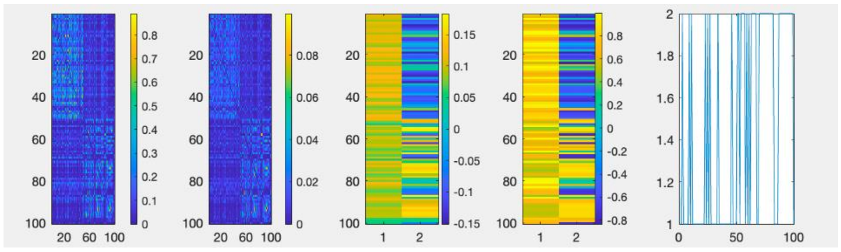




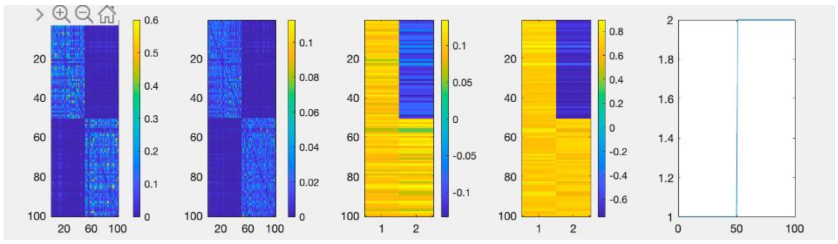


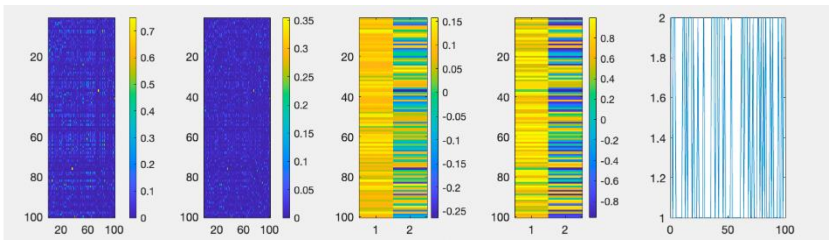


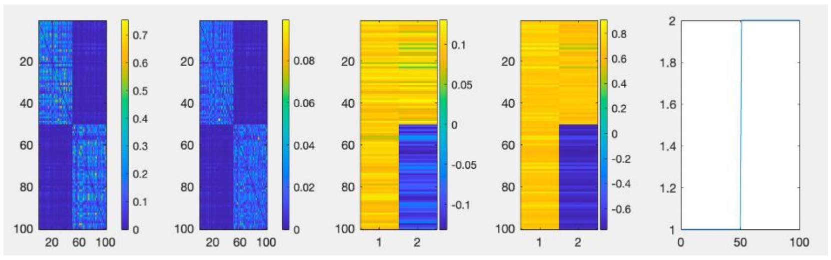


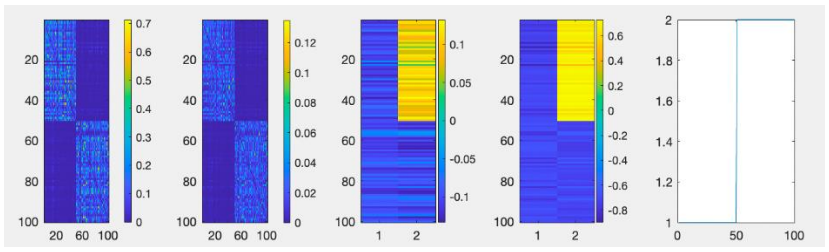


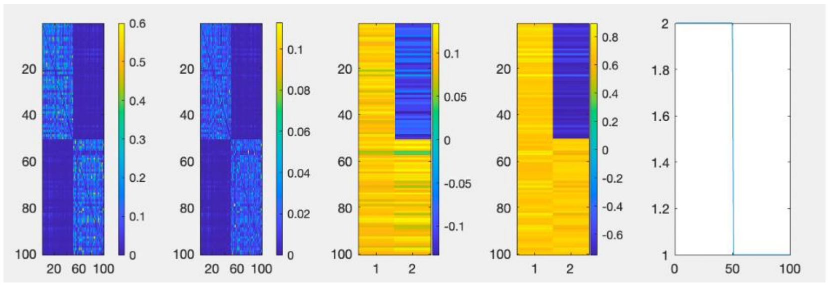
Below are the results of comparing '6' and '9' of the six features using the above sigma squares, in the order of mfeat-fac, mfeat-fou, mfeat-kar, mfeat-mor, mfeat-pix, mfeat-zer.

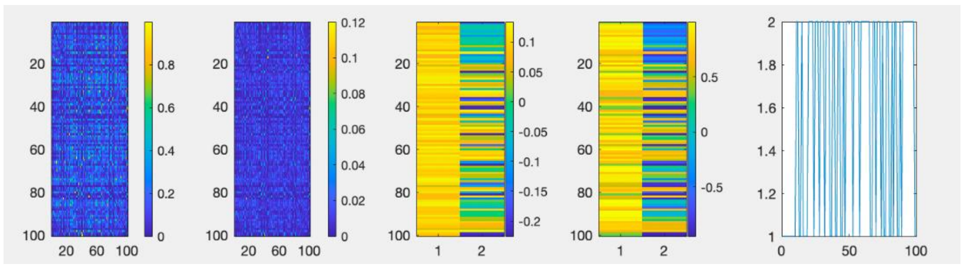




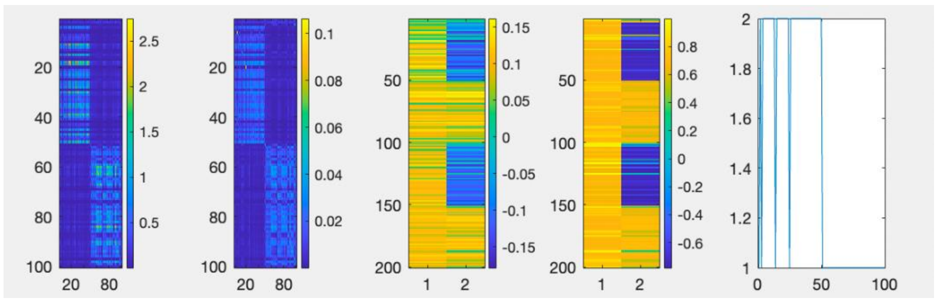


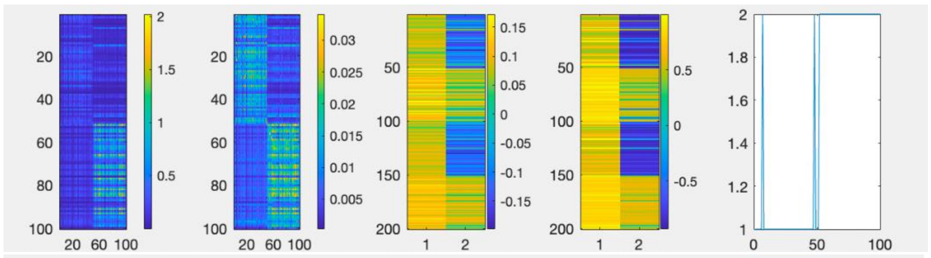


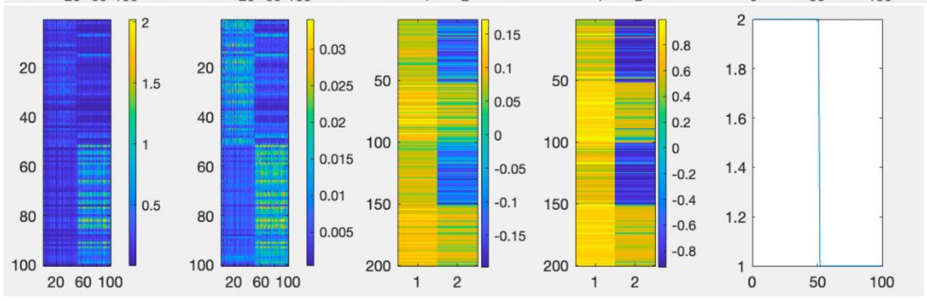




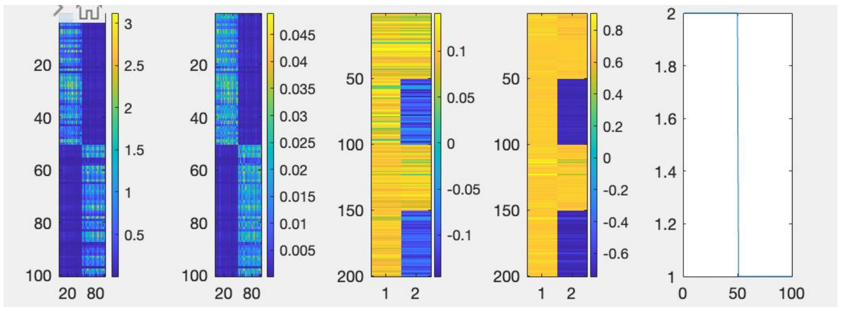
Below are the results of comparing ‘1’ and ‘7’ with two-view clustering, in the order of fac-pix with weight 0.5, 0.5; fou-pix with weight 0.5, 0.5; fou-pix with weight 0.3,0.7.

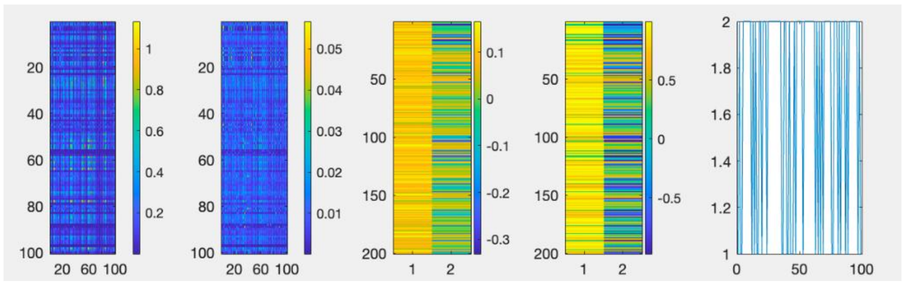


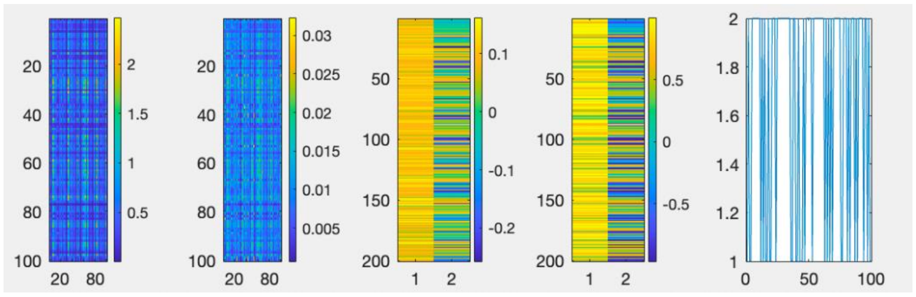




Below are the results of comparing '6' and '9' with two-view clustering, in the order of fac-kar with weight 0.5, 0.5; fou-zer with weight 0.5, 0.5; fac-fou with weight 0.5,0.5.







More tests are done on different comparing different feature pairs, different numeral pairs, and different weight attributed to different views; however, for the simplicity of this report, I will not include all the plots in this section, though some of the test results may be mentioned in the discussion section.

Discussion

For the '1', '7' pair of single-view clustering, all features except for mfeat-fou did a pretty good job on recognizing '7' as '7' while recognizing some '1's as '7' at the time. Mfeat-fou did an awful job at distinguishing '1' and '7', probably because of the relative relationship for Fourier series made it hard to recognize digits which shape is similar rotation-wise.

For the '2', '2' pair of single-view clustering, so my surprise, all features did a very good job on distinguishing '2' and '3', this is why I chose to exclude this numeral pair from the two-view clustering part, because I predict that every feature pair could do well based on the performance of the features in single-view clustering.

For the '6', '9' pair of single-view clustering, we can see the result differs in an extreme fashion: fac, kar, mor, and pix did perfect in distinguishing '6' and '9', while fou and zer did terrible (almost random clustering) on the same task. From the nature of digit 6 and 9 are rotations of each other, I confirm the inference I made at the beginning of the report that Fourier series and Zernike moments are not sensitive to rotations, but Karhunen-Love coefficients did a good job in recognizing rotations, which is unexpected.

If we combine features that yield similar results on a specific numeral pair, the two-view clustering will also yield similar results to both single-view clustering results. This is confirmed by testing on perfect pairs on '6' and '9' (fac and kar), awful pairs on '6' and '9' (fou and zer), and similar pairs on '1' and '7' (fac and pix).

Another interesting discovery is that if one of the two features did really bad on a specific numeral pair (which means it yields result almost completely random), it will affect the result of two-view clustering, no matter how well the other feature performed on its own. The result of the fac-fou pair on numeral '6' and '9' gives best demonstration on this. Notice that once the similarity matrix is generated using these two features, the result will be completely random (like the fou result on '6' and '9'), regardless of the weight you put on each view. The plot included in the result part used weight of 0.5 and 0.5, but I have also tested on other weight distributions including extreme weights of 0,1 and 1,0 (those plots I did not put in the result part), and the final result all resemble that of the complete randomness.

However, there are examples where each of the two features have an average performance, while the combination of the two features significantly improved the result. The mfeat-fou dataset actually did a pretty bad job on distinguishing '1' and '7', but when combined with mfeat-pix and performed the two-view clustering, the result is significantly improved (even significantly better than mfeat-pix alone). Also, with a little change in the weight distribution, namely from 0.5,0.5 to 0.3,0.7, it yielded the perfect result in differentiating '1' and '7'. I also did the test including more patterns of each numeral, and the final optimal weight distribution lies somewhere near 0.27,0.73.

One weakness to the test I did on these datasets and the Multiview clustering algorithm is that, as I mentioned in the result part, that different sigma square could exist for different numeral pairs within the same feature. Thus, it is very logical to imagine that different sigma square pairs exist for different numeral pairs with different feature pairs. However, it would be extremely redundant work if one has to adjust the sigma squares manually every time he tested the algorithm on different numeral pairs and different feature pairs. As far as I know, there are ways to automatically determine the sigma square for single-view spectral clustering, so building upon that, the next possible research direction could be looking for an algorithm that could automatically determine the sigma square pairs for two-view clustering, or even the sigma square vector for higher-view spectral clustering.

Reference

1. Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). *On spectral clustering: Analysis and an algorithm.* Advances in Neural Information Processing Systems 14.
2. de Sa, V.R. (2005). *Spectral Clustering with Two Vie*ws. ICML (International Conference on Machine Learning) Workshop on Learning with Multiple Views. Bonn, Germany.
3. UCI Machine Learning Repository: Multiple Features Data Set