Duttaabhinivesh Devathi
MATH 444
Assignment 3
March 9, 2017

95/100

Problem 1:

Summary:

In this problem, we are given a data set $X \in \mathbb{R}^{13\times178}$ which describes 178 different bottles of wine, and each data point has 13 different features which describe the wine. Also, we are given that the wines come from three cultivars, and the origin of each wine is also given as an annotation vector where '1' represents the first cultivar, '2' represents the second and '3' represents the third.

The premise of the problem is to find the first three LDA (Linear Discriminant Analysis) directions of the data and show a plot of them. You can do this because the data itself is annotated into three different clusters. The LDA algorithm is given in LDA m as a function, which takes in a data set, the annotation vector and the number of clusters, performs the LDA and returns the eigenvalues and associated eigenvectors of the data.

To see the data in its raw form, the first two principal components of the centered data were plotted against each other. To do this, the data was centered using a global centroid, and then a Singular Value Decomposition was run such that $X_c = UDV^T$. Centering of the data is important because the values of each feature are on different scales and have different ranges. Centering the data allows us to see the data around the origin, which makes it easier to visualize the Singular Value Decomposition of the data. We only need the first two feature vectors, and we can also represent $X_c \approx U_k D_k V_k^T$ where $k=2$ to get the first few feature vectors. From there, we can set $Z_k = U_k^T X_c$, which performs the projection of the first principal components of the centered data. **Figure 1** shows the plot of the first two principal components.

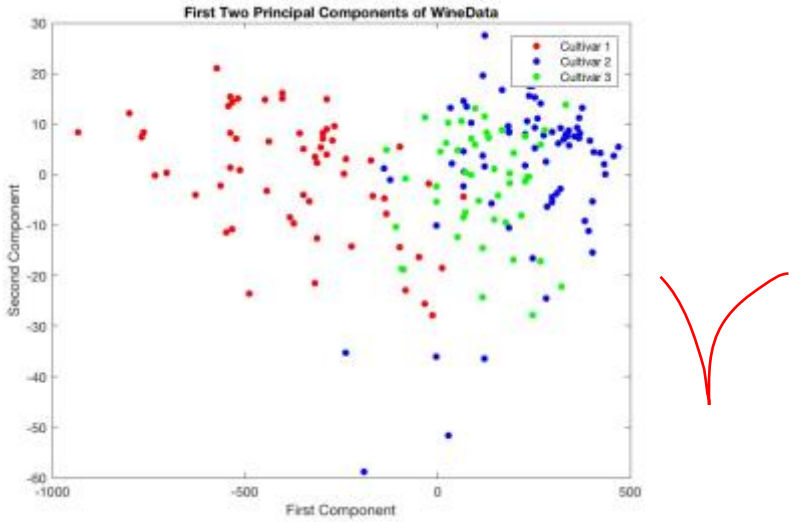

Figure 1: The first two principal components of the centered data matrix, to show the spread of the three clusters.

As you can see, the three wine clusters are separable, however there is an overlap. To show a better approximation of the spread, the LDA must be performed.

After finding the first three LDA directions of the data, the three combinations of the directions were plotted against each other. To do this, the data was transformed into LDA space by multiplying the transpose of the eigenvector matrix to the data. In matlab, it looks like this:

```
[Xw, directions, vals] = LDA(X,I,3);
Xlda = vals*directions'*X;
```

The first two rows of Xlda were plotted as they represent the LDA directions and they are shown in **Figure 2.**

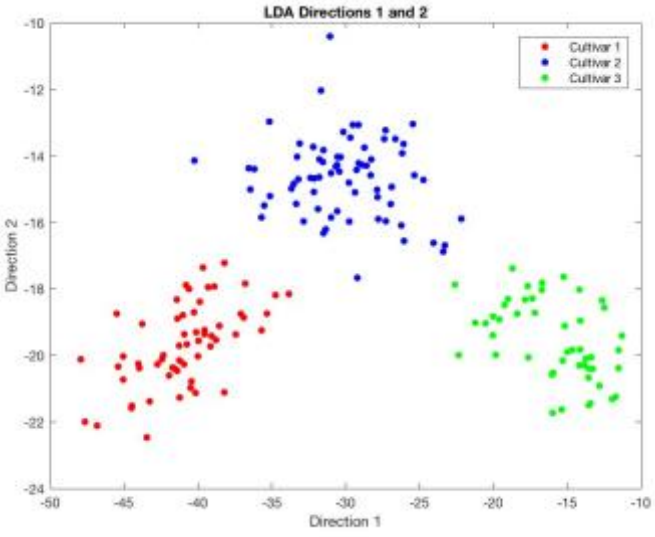

Figure 2: Clustering of the Wine Data along the first two LDA directions.

As shown, the LDA projection provides a much better projection of the clustering of the wine data in the three cultivars, and therefore provides a better visualization to see the different clusters. For completeness, the remaining two combinations of the data projections are shown in **Figure 3**.

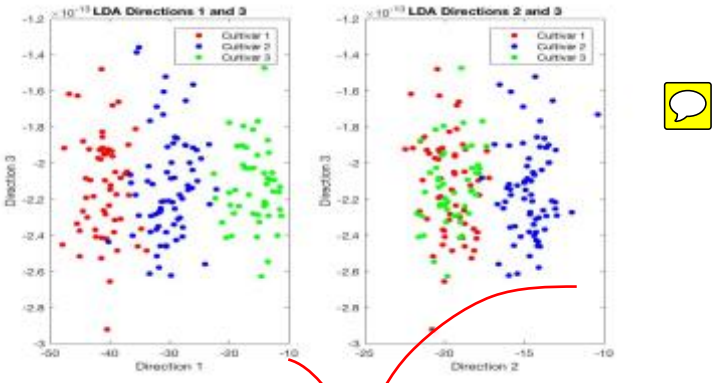

Figure 3: The remaining two combinations of the LDA projections. The left shows the 1st and 3rd LDA directions while the right shows the 2nd and 3rd LDA directions.

LDA direction 3 is problematic because the axis is scaled by a factor of 10^{13}. This is because there are only two significant eigenvalues in the LDA projection for a dataset with three clusters. Further research shows that for any data set with **n** clusters has only **n – 1** eigenvalues that are significant. In this data set, that manifests as showing the three clusters on essentially a single line if the axes were scaled properly. However, even on the one dimensional line the data was still separated between the 1st and 3rd LDA directions, which shows that you only need one eigenvector to show the separation in the data.

Instructions:

To get the plots in this chart, run `problem1.m`, which contains the scripts for the plots and the set-up of the data. The algorithm used to perform Linear Discriminant Analysis is called `LDA.m`, which should be in the same folder when running `problem1.m`. All files are attached.

Problem 2

In this problem, we are re-acquainted with the handwritten digits. There are two parts to this: the first is to perform the Linear Discriminant Analysis on two chosen digits and the second is to perform a nonnegative factorization of the digits and assess their quality. The two parts are separated as part 'A' and part 'B'.

Part A

This part of the problem asked us to plot the two digits in their first two LDA directions. This process was done entirely in `problem2a.m`. To compare, the first two principal components were plotted to see the spread of the data. The principal components of the centered data were taken to be able to compare the data of the two digits with similar ranges. **Figure 4** shows the first two principal components.

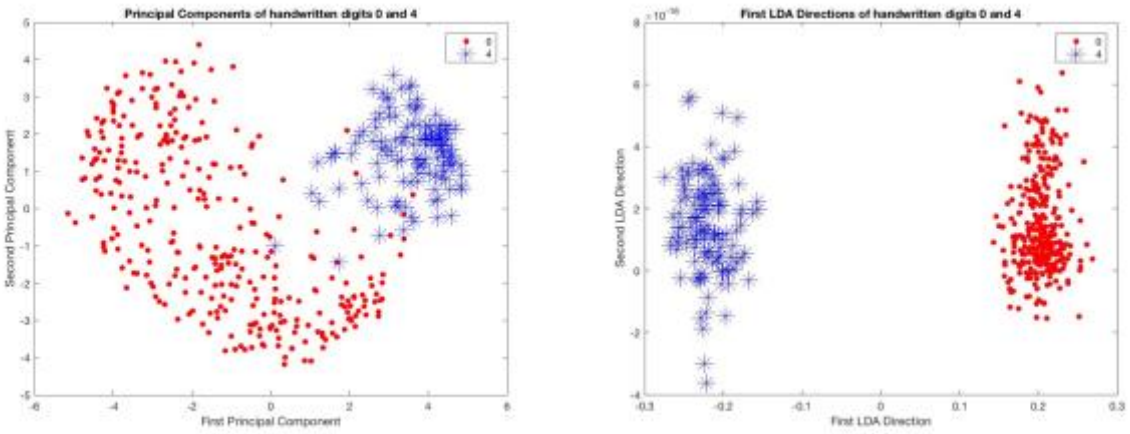

Figure 4: Left: The first two principal components of the digits 0 and 4. Right: The first LDA direction of the handwritten digit 0 and 4.

Looking at the principle components, it is easily visible that the clusters that represent 4 and 0 are easily identifiable and separated, therefore it should be easy to find an LDA direction that separates the two clusters. Furthermore, the data for 0 is a lot more sparse and wide than the data for 4. This will play a role in the Nonnegative Matrix Factorization of each of these digits. The right side shows the projection of the first two LDA directions of the handwritten digit data. The second direction is of the order 10^-16, which is smaller than machine epsilon. Compared to the first direction, this is just noise. This further confirms the theory that given *n* clusters of data, there are only *n-1* significant LDA directions.

All code and scripts to create graphs, perform LDA Analysis and Nonnegative Matrix Factorization are attached. To run part A, make sure all files are in the same folder and run `problem2a.m` to see the charts and graphs.

Part B

This problem asked us to perform the nonnegative matrix factorization of the Handwritten Digits data for two chosen digits. The chosen digits were, as before, 0 and 4. To perform the factorization, the ANLS (Alternating least squares algorithm) was used. This algorithm was taken from the notes in Chapter 6 and adapted to MATLAB. The algorithm can be found in `ANLS.m` and is referenced in the Appendix.

The data for each digit was combined to form the **data** matrix, and the ANLS Algorithm was run on that matrix.

Figure 5: A sample of feature vectors of the combined data matrix for the digits 0 and 4. The columns of the figure represents the first rank of the data, from left to right, k = 5, 10, 20. The rows denote which feature vector of the factorization it represent, from top to bottom: $W^{(1)}$, … , $W^{(5)}$.

Figure 5 shows the feature vectors for each digit 0 and 4. The second part of problem B asks us to identify one of these features that looks like a piece of a '0' and verify that the coefficients of the corresponding data vector in H reveals a '0' or a '4'. According to the figure above, a candidate for this analysis is the first feature vector with k = 5.

In the MATLAB code, the feature vector is extracted with the following commands:

```
sample = W5_k1k2(:, 1);
coeff = H5_k1k2(:, 1);
othercoeff = H5_k1k2(:, 2);
lincomb = zeros(size(sample));
otherlincomb = zeros(size(sample));
for i = 1:length(coeff)
   lincomb = lincomb + coeff(i)*sample;
   otherlincomb = otherlincomb + othercoeff(i)*sample;
end
diff = norm( X5_k1k2(:, 1) - lincomb, 2);
```

In the code, the coefficients of two different vectors are chosen, the first is one that corresponds to the '0' and the other is a vector that corresponds to a '4'. Each $x^{(1)}$ is rebuilt according to the formula on page 59 of the notes in Chapter 6, reproduced below. Then, the norm of the differences between the feature vector and the rebuilt vectors are calculated. It is found that the norm that has the mismatch, 'otherdiff,' has the higher norm and therefore does not reconstruct the zero as specified.

$$x^{(j)} \approx \sum_{l=1}^{k} h_{l_j} * u^{(j)}$$

Furthermore, inspection of the formula shows that the higher the coefficients of the vector $h^{(j)}$, the closer to the actual data vector $x^{(j)}$ the approximation becomes. Thus, the sum of each $h^{(j)}$ column can be taken and compared (the matching $h^{(j)}$ and the mismatching $h^{(l)}$). Doing so reveals that the matching $h^{(j)}$ has a higher sum. This is possible because the $h^{(j)}$ column contains entries between 0 and 1, corresponding to the proportion of $u^{(j)}$ that is being added to recreate the data vector. Doing so reveals that the matching coefficients have a higher sum, and recreates a '0'.

Appendix

Function Files:

ANLS.m
-Function that performs alternative nonnegative least squares algorithm to do a nonnegative matrix factorization on a given data set. Takes in inputs matrix X, #feature vectors k, and a tolerance tol. Returns W, H, and norms: residuals calculated at each step of the iteration.

entropy.m
-A function that calculates the distance entropy between two matrices. The formulation of this algorithm comes from the notes, Chapter 6. Takes in input of two matrices A,B and returns a scalar D.

LDA.m
-The linear discriminant analysis algorithm for a given data set. Works well for ordered clustering, not too well for clusters annotated with non-sequential integers (doesn't work for HandwrittenDigits that well).

Data Files:

HandwrittenDigits.mat
-The .mat file which contains the data for the handwritten digits. A visualization of the data is provided in Chapter 2 of the notes and PCA_HandwrittenDigits_Classroom.m

WineData.mat
-The .mat file that contains the data for each wine. Has 13 features and 178 feature vectors. This file is used in Problem 1 and in the file problem1.m

Scripts:

problem1.m
-This file contains scripts to create plots and graphs for everything relevant to Problem 1.

problem2a.m
-This file contains the LDA projections of two handwritten digits k1 = 0 and k2 = 4.

problem2b_ANLS.m
-This file performs the ANLS algorithm on two handwritten digits k1 = 0 and k2 = 4, then plots the first five feature vectors for k = 5, k = 10 and k = 20.