Question 1 : (30 total points) Image data analysis with PCA

In this question we employ PCA to analyse image data

1.1 (3 points) Once you have applied the normalisation from Step 1 to Step 4 above, report the values of the first 4 elements for the first training sample in `Xtrn_nm`, i.e. `Xtrn_nm[0,:]` and the last training sample, i.e. `Xtrn_nm[-1,:]`.

The first 4 elements of the first training sample in Xtrn_nm, to 4 decimal places, are:
[-3.137x10^{-6}, $-2.268x10^{-5}$, -0.0001, -0.0004]
The first 4 elements of the last training sample in Xtrn_nm are (exact values):
$[1.0, 1.0, 1.0, 1.0]$

1.2 (4 points) Using `Xtrn` and Euclidean distance measure, for each class, find the two closest samples and two furthest samples of that class to the mean vector of the class.

The images that are classed as furthest or second furthest from the mean of each class are generally a irregular shape or have markings that cause dramatic differences in pixel values. Examples of this include class 1, where a pair of shorts is seen to be the furthest image from the mean of the trousers class, and also class 4, where a coat with drastically different coloured sleeves and torso is seen as the furthest image from the mean of the coat class. The closest images to the mean tend to be slightly lighter than the furthest images, show more detail and have quite regular shapes. In contrast, a lot of the furthest images tend to be very dark, almost resembling silhouettes of the objects, particularly in the first few classes.

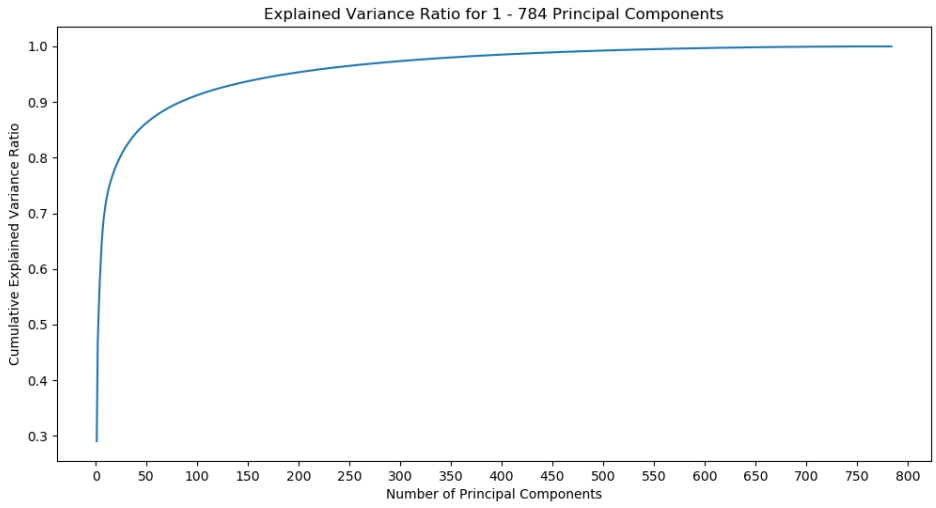
1.3 (3 points) Apply Principal Component Analysis (PCA) to the data of `Xtrn_nm` using sklearn.decomposition.PCA, and report the variances of projected data for the first five principal components in a table. Note that you should use `Xtrn_nm` instead of `Xtrn`.

The variances of the first five principal components for Xtrn_nm are, to 3 decimal places:

Principal Component	Variance
1	19.816
2	12.114
3	4.106
4	3.382
5	2.625

1.4 (3 points) Plot a graph of the cumulative explained variance ratio as a function of the number of principal components, K, where $1 \leq K \leq 784$. Discuss the result briefly.

The below graph shows that the cumulative explained variance ratio initially increases rapidly, with around 90% of the total variance being accounted for at roughly 50 components. When all 784 components are taken into account, the cumulative explained variance is 1, meaning that all the variance is accounted for, which is to be expected.



1.5 (4 points) Display the images of the first 10 principal components in a 2-by-5 grid, putting the image of 1st principal component on the top left corner, followed by the one of 2nd component to the right. Discuss your findings briefly.

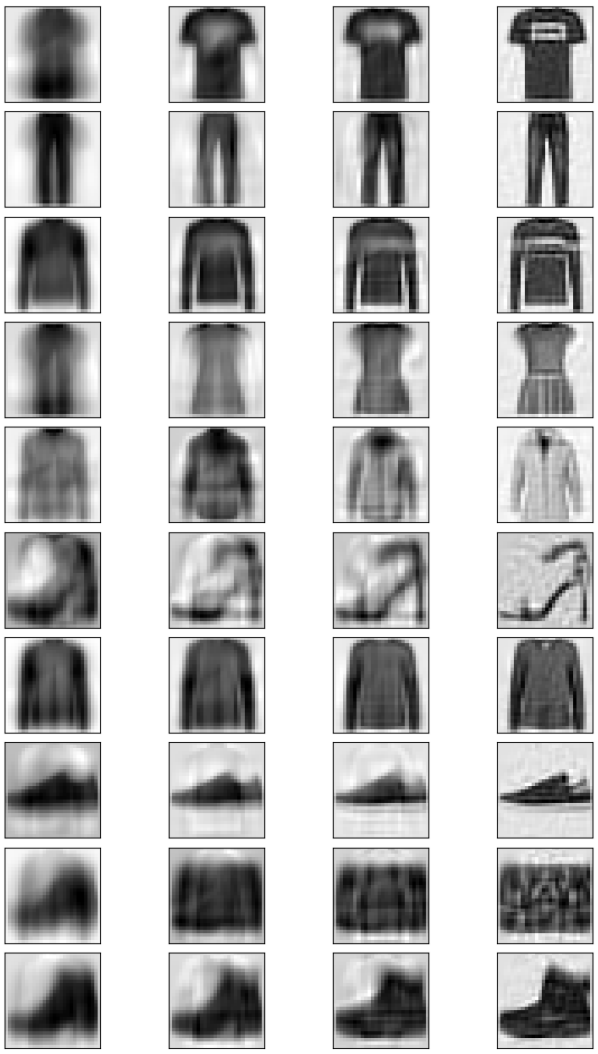
1.6 (5 points) Using `Xtrn_nm`, for each class and for each number of principal components $K = 5, 20, 50, 200$, apply dimensionality reduction with PCA to the first sample in the class, reconstruct the sample from the dimensionality-reduced sample, and report the Root Mean Square Error (RMSE) between the original sample in `Xtrn_nm` and reconstructed one.

In general, it appears that the Root Mean square Error RMSE decreases as the number of Principal Components (PCs) K increases. Note that each time my code is run a slightly different answer is generated.

Class	5 PCs RMSE	20 PCs RMSE	50 PCs RMSE	200 PCs RMSE
0	0.256	0.150	0.127	0.061
1	0.198	0.140	0.095	0.037
2	0.199	0.146	0.123	0.080
3	0.146	0.107	0.084	0.057
4	0.118	0.103	0.088	0.047
5	0.181	0.159	0.143	0.091
6	0.129	0.096	0.071	0.047
7	0.166	0.128	0.107	0.062
8	0.223	0.145	0.124	0.093
9	0.184	0.151	0.122	0.073

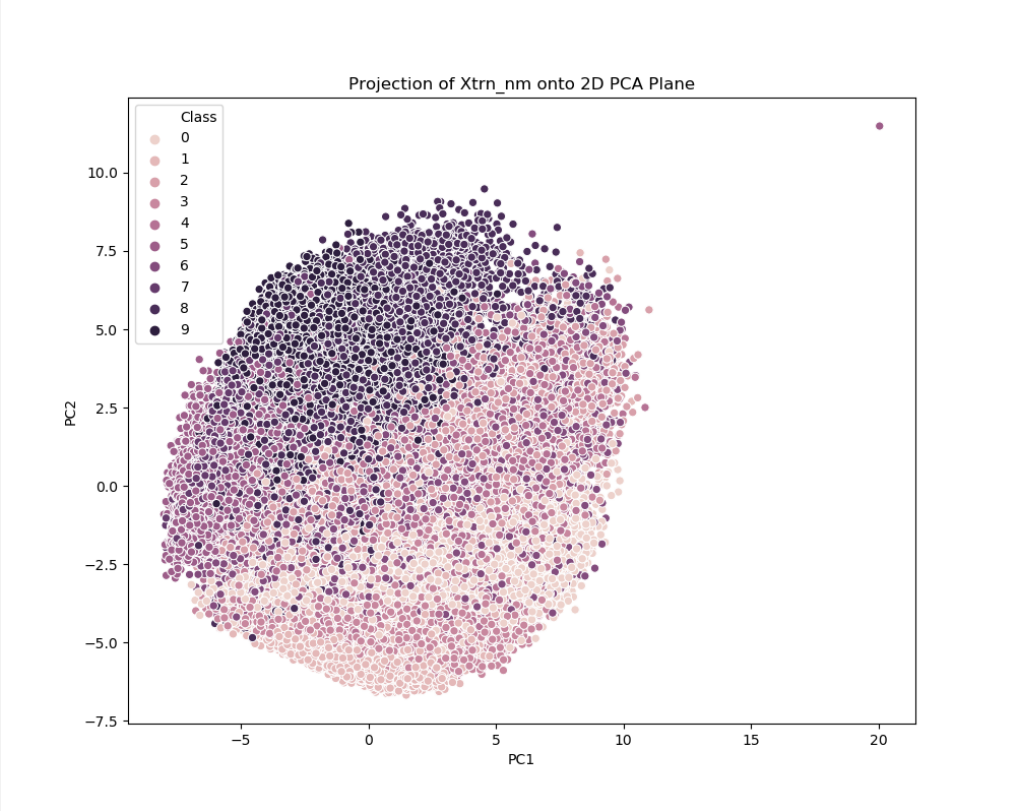
1.7 (4 points) Display the image for each of the reconstructed samples in a 10-by-4 grid, where each row corresponds to a class and each row column corresponds to a value of $K = 5,\ 20,\ 50,\ 200$.

In general, as K increases, the images become more concrete and recognisable as a single object rather than a mesh of a few objects. The more components used, the more certain you can be that an image is of a particular class.



1.8 (4 points) Plot all the training samples (`Xtrn_nm`) on the two-dimensional PCA plane you obtained in Question 1.3, where each sample is represented as a small point with a colour specific to the class of the sample. Use the 'coolwarm' colormap for plotting.

Most of the data was projected into one large circle, however there does seem to be one outlier value in the upper right corner, which may have contributed to the choosing of the principal axes by increasing the variance in the direction of each of the principal components.
The images in the same class are projected close to each other, while images that are very different with respect to the two components are projected far away from each other. There is a lot of class overlap in the data.



Question 2 : (25 total points) Logistic regression and SVM

In this question we will explore classification of image data with logistic regression and support vector machines (SVM) and visualisation of decision regions.

2.1 (3 points) Carry out a classification experiment with multinomial logistic regression, and report the classification accuracy and confusion matrix (in numbers rather than in graphical representation such as heatmap) for the test set.

Classification Accuracy, to 3 decimal places: 0.841
Confusion Matrix:

$$\begin{bmatrix} 819 & 3 & 15 & 50 & 7 & 5 & 88 & 1 & 12 & 0 \\ 5 & 953 & 4 & 27 & 5 & 0 & 3 & 1 & 2 & 0 \\ 27 & 4 & 731 & 11 & 133 & 0 & 82 & 2 & 9 & 1 \\ 31 & 15 & 14 & 866 & 33 & 0 & 37 & 0 & 4 & 0 \\ 0 & 3 & 115 & 38 & 760 & 1 & 73 & 0 & 10 & 0 \\ 2 & 0 & 0 & 1 & 0 & 914 & 0 & 55 & 10 & 18 \\ 147 & 3 & 128 & 46 & 108 & 0 & 539 & 0 & 28 & 1 \\ 0 & 0 & 0 & 0 & 0 & 32 & 0 & 936 & 1 & 31 \\ 5 & 1 & 7 & 11 & 2 & 7 & 15 & 5 & 947 & 0 \\ 0 & 0 & 0 & 1 & 0 & 15 & 1 & 42 & 0 & 941 \end{bmatrix}$$

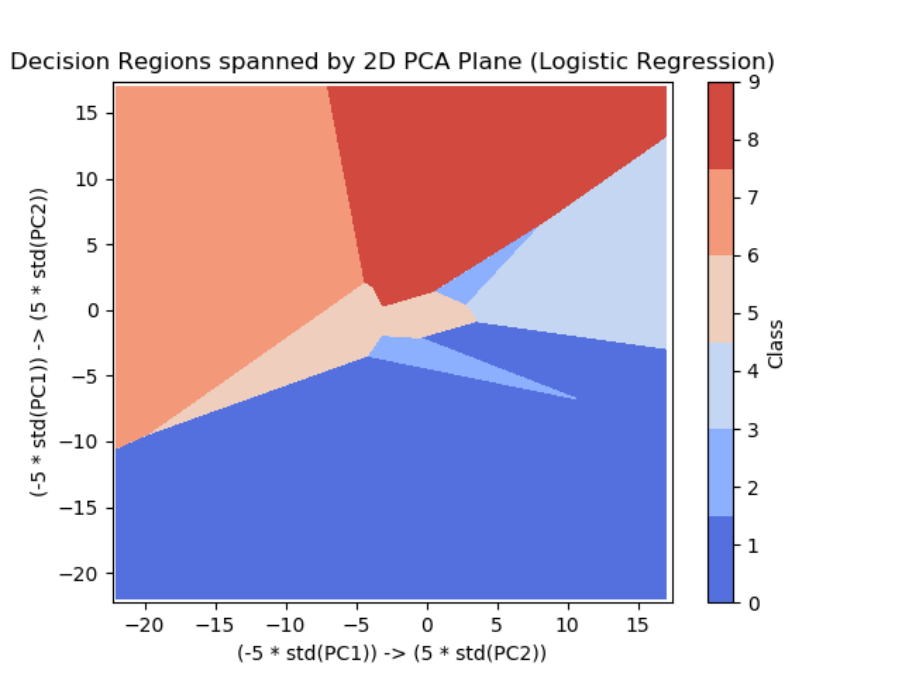
2.2 (3 points) Carry out a classification experiment with SVM classifiers, and report the mean accuracy and confusion matrix (in numbers) for the test set.

Classification Accuracy, to 3 decimal places: 0.846
Confusion Matrix:

$$\begin{bmatrix} 845 & 2 & 8 & 51 & 4 & 4 & 72 & 0 & 14 & 0 \\ 4 & 951 & 7 & 31 & 5 & 0 & 1 & 0 & 1 & 0 \\ 15 & 2 & 748 & 11 & 137 & 0 & 79 & 0 & 8 & 0 \\ 32 & 6 & 12 & 881 & 26 & 0 & 40 & 0 & 3 & 0 \\ 1 & 0 & 98 & 36 & 775 & 0 & 86 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 & 0 & 913 & 0 & 57 & 3 & 26 \\ 185 & 1 & 122 & 39 & 95 & 0 & 533 & 0 & 25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 34 & 0 & 925 & 0 & 41 \\ 3 & 1 & 8 & 5 & 2 & 4 & 13 & 4 & 959 & 1 \\ 0 & 0 & 0 & 0 & 0 & 22 & 0 & 47 & 1 & 930 \end{bmatrix}$$

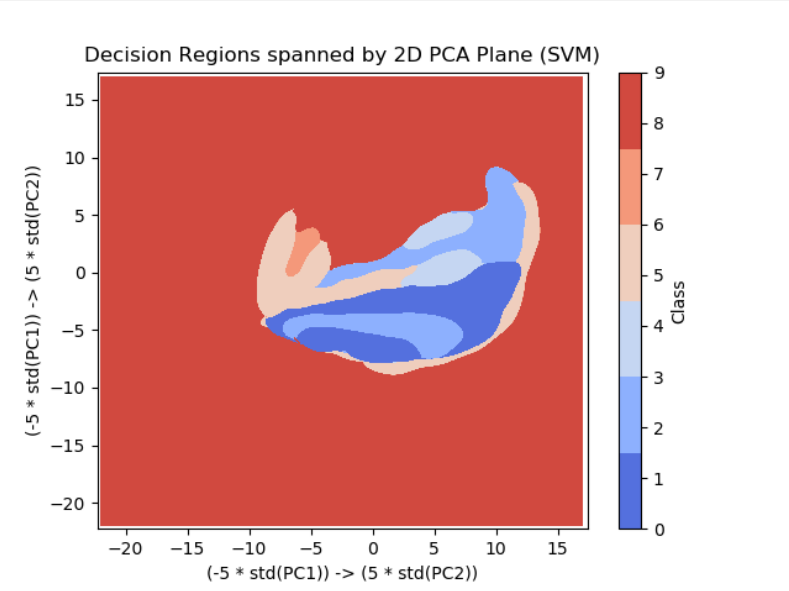
2.3 (6 points) We now want to visualise the decision regions for the logistic regression classifier we trained in Question 2.1.

The graph below shows that the decision regions in the 2D plane are quite consistent, with all classes having one singular decision region, and no small floating regions in the plane, which would have indicated over-fitting in the data. Most of the adjacent classes within the colormap (8 and 9, 6 and 7, 0 and 1) all share a boundary when represented y this limited set of colours. The only classes represented by the same colour that have significantly different boundaries are classes 2 and 3.



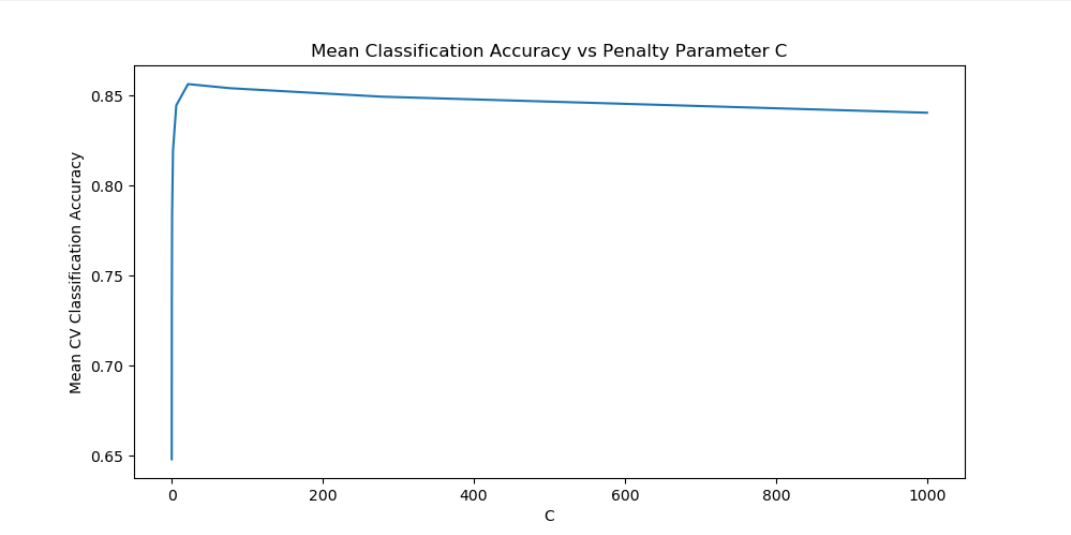
2.4 (4 points) Using the same method as the one above, plot the decision regions for the SVM classifier you trained in Question 2.2. Comparing the result with that you obtained in Question 2.3, discuss your findings briefly.

The biggest difference between the decision boundaries for SVM and Logistic Regression is that for SVM, the decision boundaries for classes 0-7 are fully bounded, with classes 8 and 9 taking the outside regions. For logistic regression, most of the regions were not fully bounded and instead extended to the edges of the plot.
Another difference is the shape of the regions. For logistic regression, they were triangles, while for SVM they are all irregularly shaped blobs, which seem to be a lot more detailed. The regions for the classes seem to be in roughly the same positions with respect to each other for both classifiers, with two exceptions. The region for either class 5 or 6 has an area to the right of the blue regions for SVM, which it does not have for logistic regression. Also, classes 8 and 9 have a much larger region that spans most of the plot.



2.5 (6 points) We used default parameters for the SVM in Question 2.2. We now want to tune the parameters by using cross-validation. To reduce the time for experiments, you pick up the first 1000 training samples from each class to create `Xsmall`, so that `Xsmall` contains 10,000 samples in total. Accordingly, you create labels, `Ysmall`.

Highest Mean Classification Accuracy: 85.650%
Optimal C value: 21.544



2.6 (3 points) Train the SVM classifier on the whole training set by using the optimal value of C you found in Question 2.5.

Using the optimal C value 21.544346900318846, the accuracies of the SVM classifier are, to 3dp:
Training Accuracy: 0.908
Testing Accuracy: 0.877

Question 3 : (20 total points) Clustering and Gaussian Mixture Models

In this question we will explore K-means clustering, hierarchical clustering, and GMMs.

3.1 (3 points) Apply k-means clustering on `Xtrn` for $k = 22$, where we use sklearn.cluster.KMeans with the parameters `n_clusters=22` and `random_state=1`. Report the sum of squared distances of samples to their closest cluster centre, and the number of samples for each cluster.

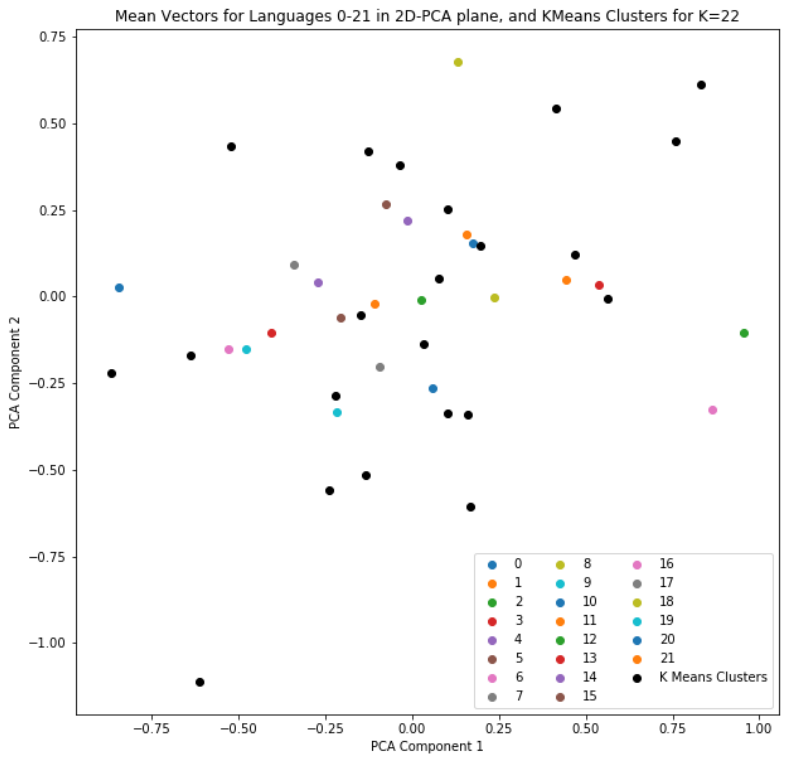
Sum of Squared distances of samples to their closest cluster centre = 38185.817
Number of samples for each cluster:

Cluster	Number of Samples
0	1018
1	1125
2	1191
3	890
4	1162
5	1332
6	839
7	623
8	1400
9	838
10	659
11	1276
12	121
13	152
14	950
15	1971
16	1251
17	845
18	896
19	930
20	1065
21	1466

Question 3 continued on next page...

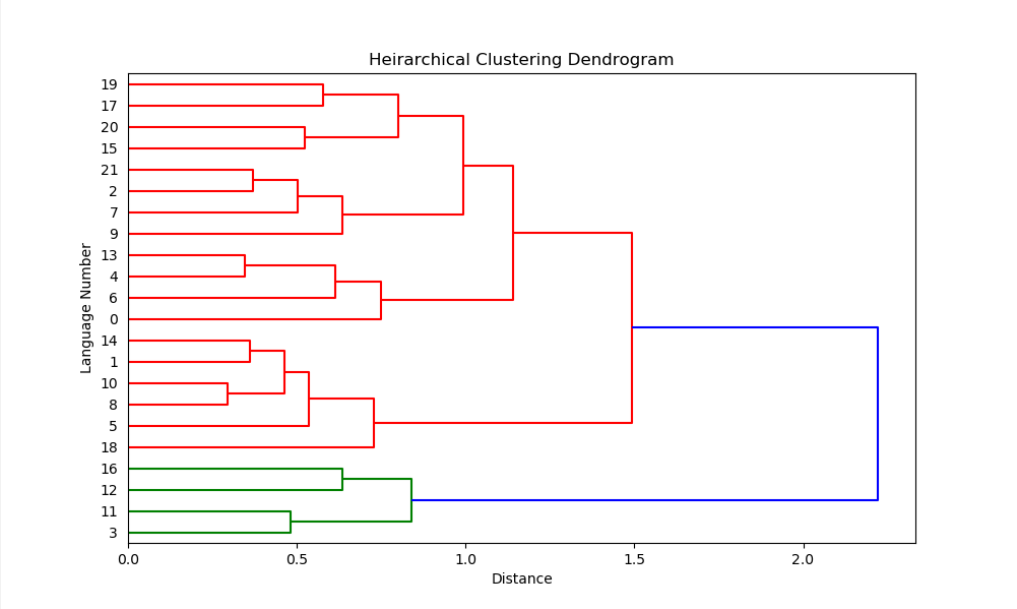
3.2 (3 points) Using the training set only, calculate the mean vector for each language, and plot the mean vectors of all the 22 languages on a 2D-PCA plane, where you apply PCA on the set of 22 mean vectors without applying standardisation. On the same figure, plot the cluster centres obtained in Question 3.1.

The clusters from KMeans and the mean vectors of each language, shown by their number 0 - 21, are not very consistent at all. This was to be expected because of the amount of simplification applied to the data.
Some of the clusters in the centre where the data is more dense are more similar to their corresponding mean vectors, however the data further out has significant differences between the two.



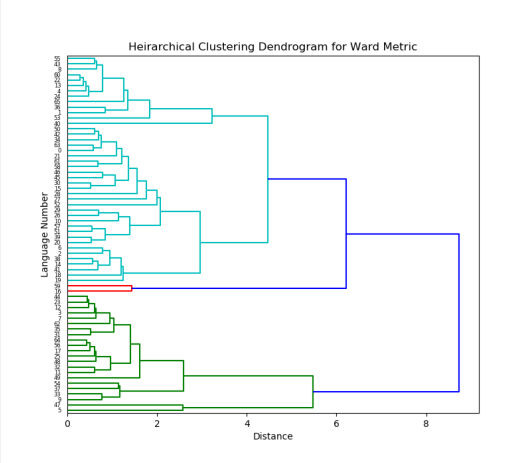
3.3 (3 points) We now apply hierarchical clustering on the training data set to see if there are any structures in the spoken languages.

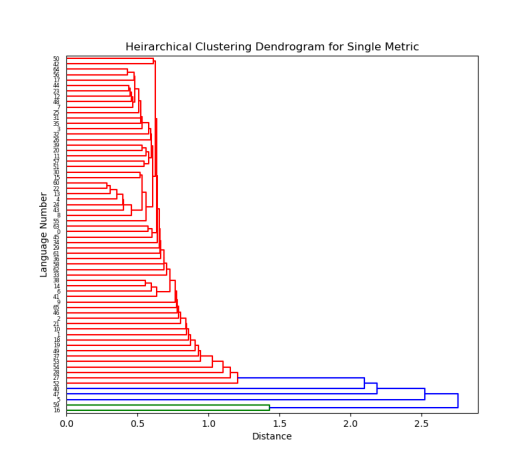
The below dendrogram shows that the two closest languages are languages 8 and 10, and the second two closest languages are 4 and 13. It also shows that there is some sort of structure to the languages, as they are quite evenly split into clusters if we take the distance threshold to be 0.9, for example. The dendrogram can't tell us how many clusters we should have, but does tell us that a lot of the languages are quite similar, with a lot of the links taking place between clusters with a distance under 0.5. This dendrogram was by default split into two clusters, denoted by the red and green links.

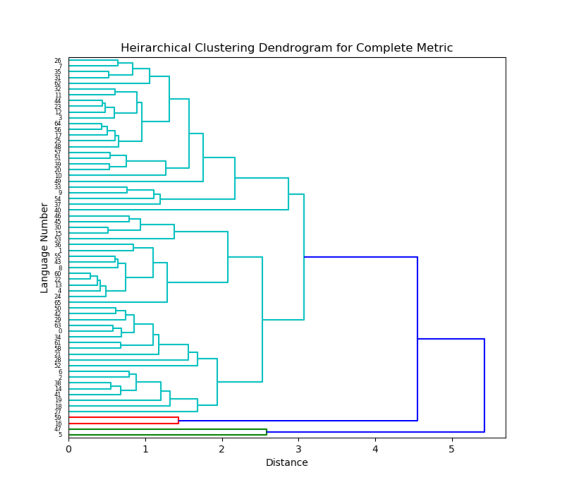


3.4 (5 points) We here extend the hierarchical clustering done in Question 3.3 by using multiple samples from each language.

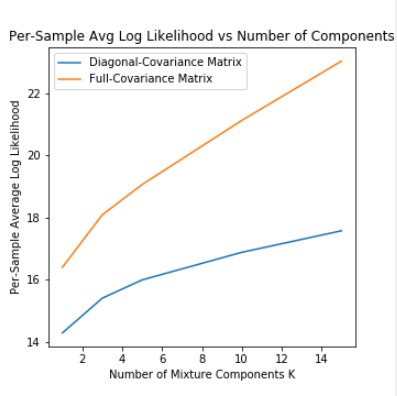
The dendrogram for the ward linkage method produces the most balanced dendrogram, while both the single and complete linkage methods produce very one-sided outputs, with a huge majority of the languages being clustered together at an early stage. The single linkage method links clusters to their closest cluster, which meant that at higher distance values, it was always just one language point being added to a cluster, and never two non-singleton clusters being merged.
The complete linkage method is also very one-sided, and ends up with one large cluster and two two-language clusters before the clusters are all merged together. The ward linkage method also has a two-language cluster, but the other two clusters formed before the final links are a lot more balanced, and there are a lot more merges of larger clusters.







3.5 (6 points) We now consider Gaussian mixture model (GMM), whose probability distribution function (pdf) is given as a linear combination of Gaussian or normal distributions, i.e.,



K	Diag-Cov Matrix Avg Log Likelihood	Full-Cov Matrix Avg Log Likelihood
1	14.280	16.398
3	15.398	18.087
5	15.987	19.177
10	16.962	20.927
15	17.597	22.906

Note that every time my code runs a slightly different answer is generated. As K increases, the average log likelihood also tends to increase. Also, the likelihoods for the full covariance matrix are always higher than the likelihoods for the diagonal covariance matrix.