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,:]`.

First 4 elements of the first and last samples from the normalized training dataset Xtrn_nm

```
First 4 elements of the first training sample in Xtrn_nm:
[-3.13725490e-06 -2.26797386e-05 -1.17973856e-04 -4.07058824e-04]

First 4 elements of the last training sample in Xtrn_nm:
[-3.13725490e-06 -2.26797386e-05 -1.17973856e-04 -4.07058824e-04]
```

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.

A grid to show the mean vectors for each class along with the closest and furthest samples from these means

There is an interesting trend amongst the closest and furthest samples for each given class in the dataset. Notice that all the 'closest' samples are a subtle gray, and in contrast all the 'furthest' samples are very dark or very light (eg. the class 4 samples). We can deduce that this is due to the fact that the intensely dark/light colour pixels magnify the difference (and thus Euclidean distance) between corresponding pixel values in different samples.

This is a very important observation as it highlights the significance of colour intensity when calculating the similarity between different samples. This colour intensity could ultimately skew the performance of our classifier when predicting the type of clothing for uniquely dark/light samples. This is especially problematic if a given class (type of clothing) is more likely to be dark/light as this will make it more likely to classify uniquely dark/light samples from other classes.

To prevent this issue, given that we are classifying the type of clothing and not the colour, we could either use a method specifically for classifying shapes (image segmentation) or we could normalize the colour intensity of samples upon input (normalize the values of the pixels to be within a certain range).

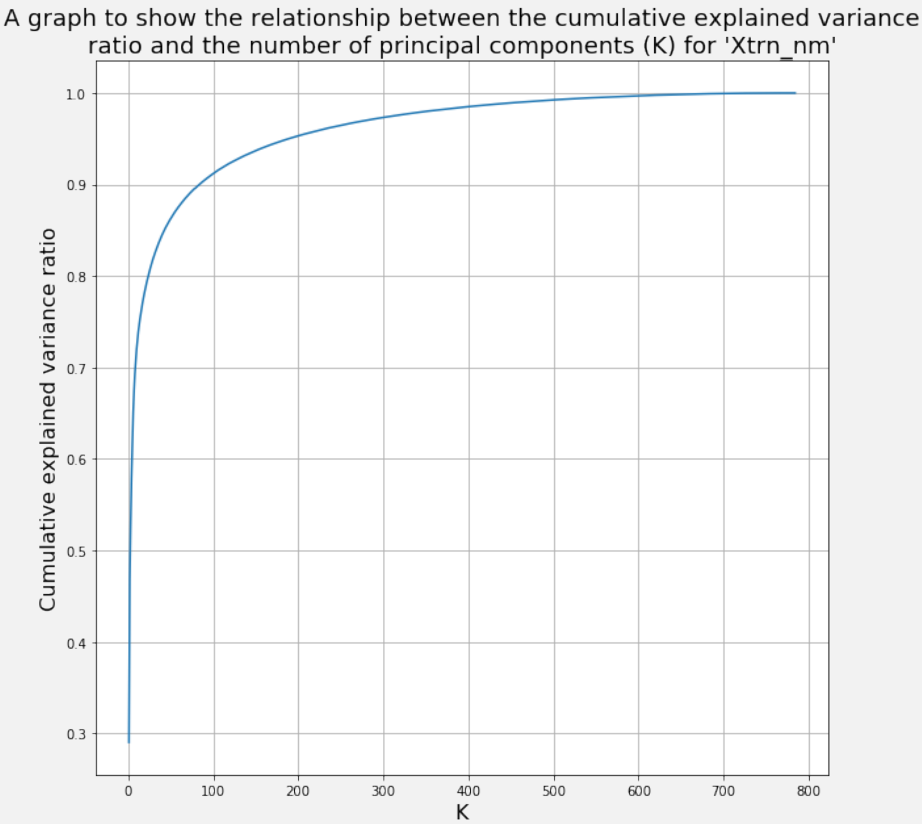
1.3 (3 points) Apply Principal Component Analysis (PCA) to the data of `Xtrn_nm` using sklearn.decomposition.PCA, and report the variances of projected data for the first five principal components in a table. Note that you should use `Xtrn_nm` instead of `Xtrn`.

Calculating the variances of the projected data in Xtrn_nm for the first five principal components

Principal Component #	**Explained Variance**
1	19.81
2	12.112
3	4.106
4	3.382
5	2.625

Question 1 continued on next page...

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.



Cumulative explained variance ratio $= \frac{\sum_{k=1}^{K} \sigma_k^2}{\sigma_{total}^2}$

Given that the explained variance ratio for a given principal component represents the variance of that principal component divided by the total variance, we can see that the results of this graph showing the relationship between the cumulative explained ratio and the number of principal components (K) are as expected.

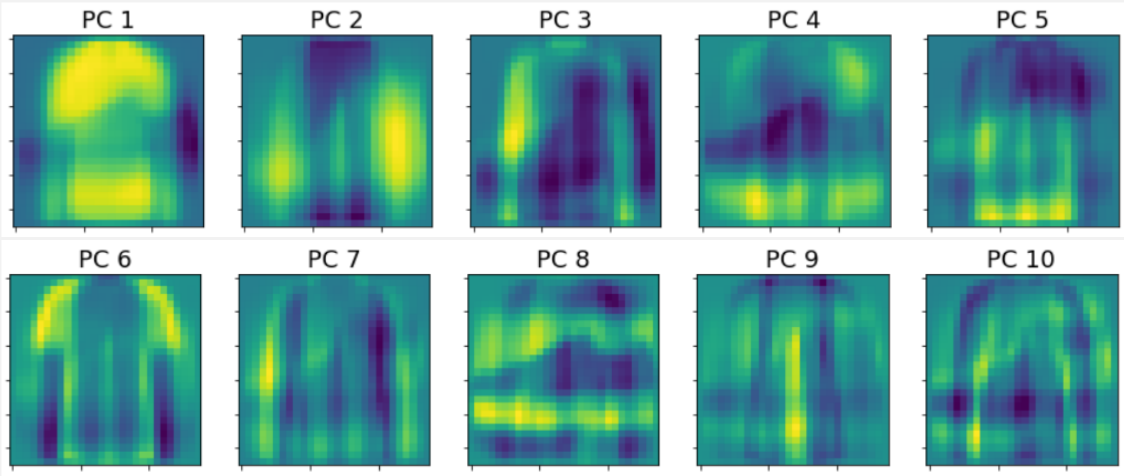
Although PCA chooses dimensions with maximum variance, as we increase the number of dimensions (K) we automatically increase the number of variances (which have been maximised by PCA) we must summate. We also know that the total variance is fixed throughout all different values of K so the value of all cumulative variance ratios are purely dependent on ther respective explained variance ratios. Thereby, it is evident as illustrated by the plot that the number of principal components (K) is directly proportional to the cumulative explained ratio in a hyperbolic fashion.

This hyperbolic shape is due to the inversely proportional relationship between the number of dimensions (K) and the explained variance for each new dimension (dimension K). This is evident as PCA always chooses the dimensions that maximize variance, and thereby new dimensions will always have smaller explained variances than that of the other preceding K-1 dimensions. These ordered dimension variances ultimately give us the resultant smooth curve as shown above whose gradient slows down as we increase the value of K.

This graphing model is very useful in order to determine the optimal number of dimensions to reduce the data to whilst still retaining as much detail as possible (maximising cumulative variance). We can see here that using just 100 dimensions (about 87.24% less data) already secures roughly 92% of the total cumulative variance from the original data.

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.

A grid to show the images of the first 10 principal components



Discuss your findings briefly...
We must note that these images are not distinct given they were calculated using the normalized dataset
***Should show UNnormalized photos too!

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.

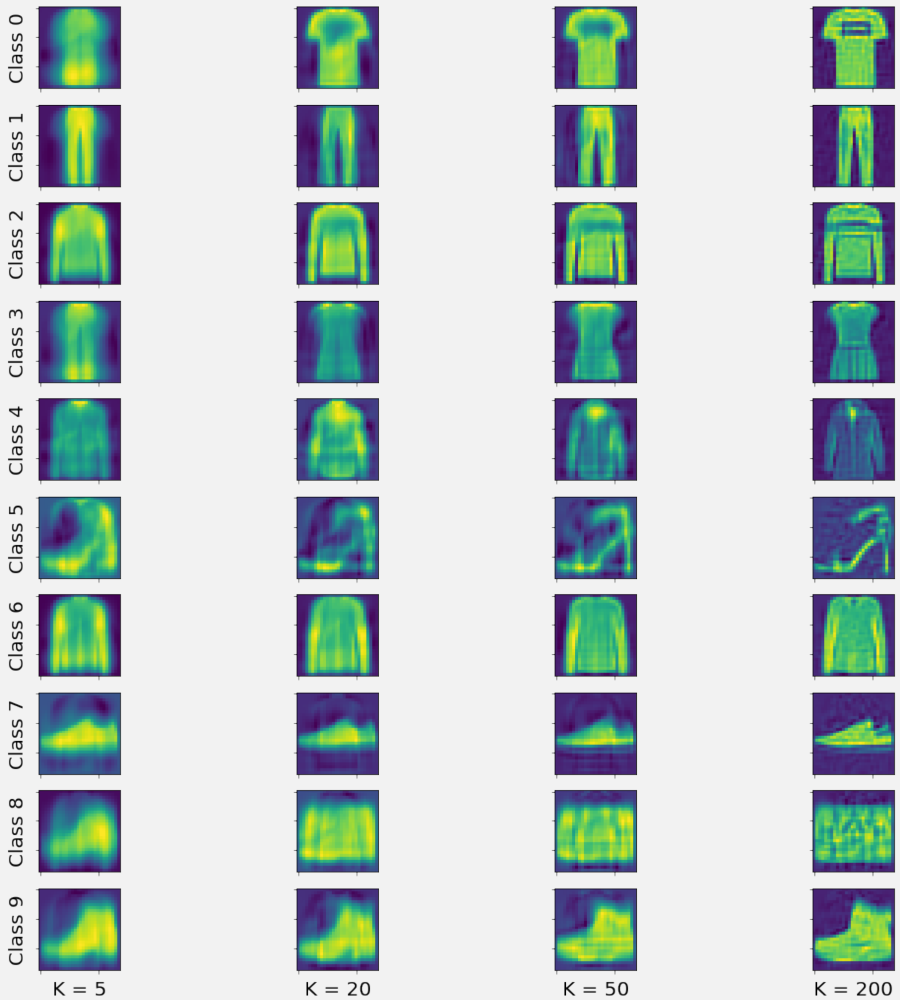
A table to show the RMSE between the original and the reconstructed version of the first sample for every class with varying numbers of PCA components (K)

*Each class sample is reconstructed by reducing the sample to K dimensions and then transforming it back to the original number of dimensions, this is all done via the sklearn PCA implementation.

RMSE	K = 5	K = 20	K = 50	K = 200
Class = 0	0.256	0.15	0.128	0.063
Class = 1	0.198	0.14	0.095	0.037
Class = 2	0.199	0.146	0.124	0.08
Class = 3	0.146	0.107	0.083	0.056
Class = 4	0.118	0.103	0.088	0.045
Class = 5	0.181	0.159	0.142	0.09
Class = 6	0.129	0.096	0.072	0.045
Class = 7	0.166	0.128	0.106	0.063
Class = 8	0.223	0.145	0.124	0.094
Class = 9	0.184	0.151	0.122	0.072

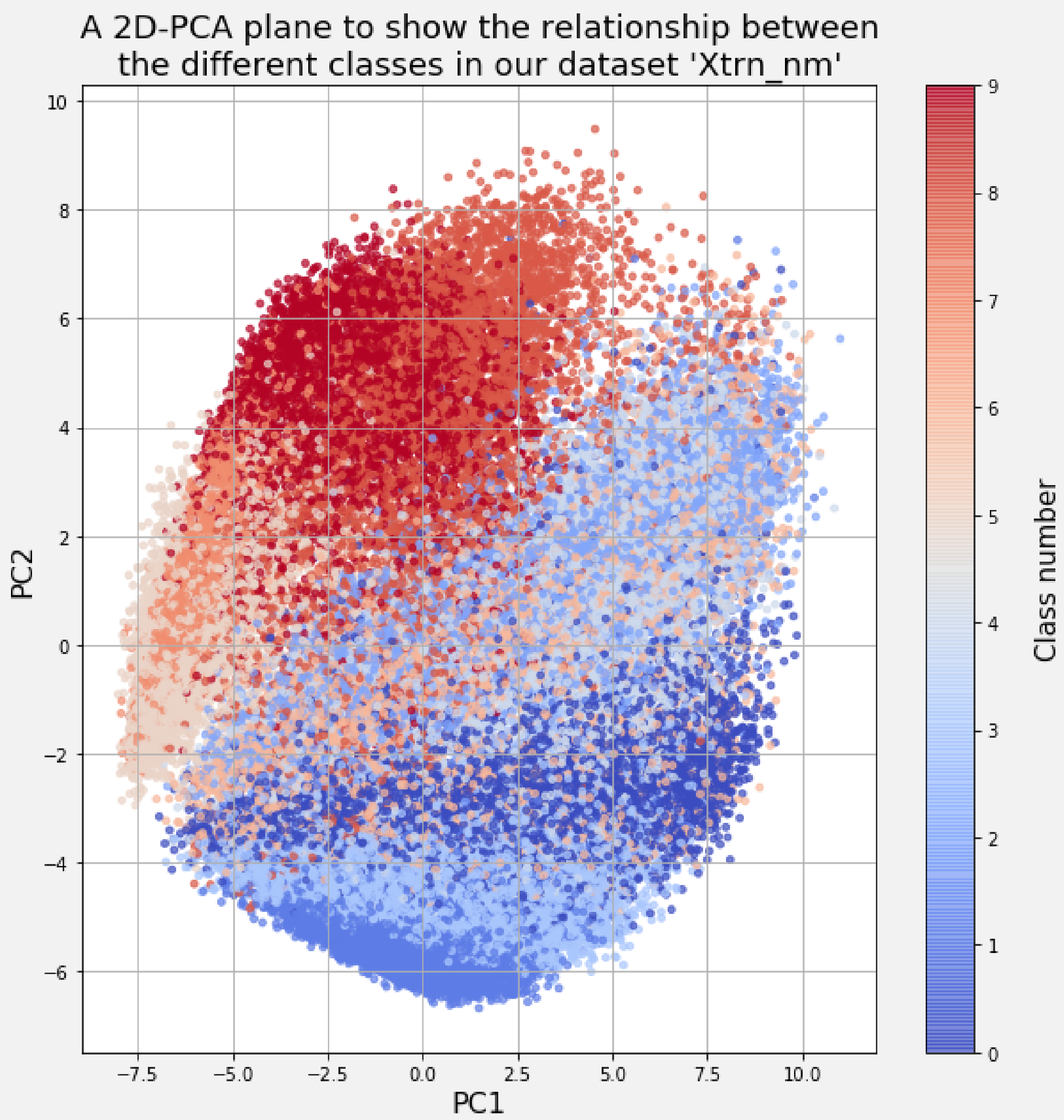
1.7 (4 points) Display the image for each of the reconstructed samples in a 10-by-4 grid, where each row corresponds to a class and each row column corresponds to a value of $K = 5,\ 20,\ 50,\ 200$.

A grid to show the images of the reconstructed class samples for varying amounts of dimension reductions (K)



As expected the reconstructed samples become more detailed as we increase the value of K. This is evident due to the increased number of dimensions to retain detail within the data. PCA reduces dataset dimensions by choosing dimensions which maximize variance in the data, this is evident to see as the reconstructed samples with small K highlight the more promiment features from their respective original samples.

1.8 (4 points) Plot all the test 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.



Give comments on the seperation of the classes and explain findings briefly...

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.

Confusion matrices to show the classification accuracy of our trained multinomial logistic regression model on the test set

FREQUENCY CONFUSION MATRIX:

Predicted / Actual	0	1	2	3	4	5	6	7	8	9
0	819	5	27	31	0	2	147	0	7	0
1	3	953	4	15	3	0	3	0	1	0
2	15	4	731	14	115	0	128	0	6	0
3	50	27	11	866	38	1	46	0	11	1
4	7	5	133	33	760	0	108	0	3	0
5	4	0	0	0	2	911	0	32	7	15
6	89	3	82	37	72	0	539	0	15	1
7	1	1	2	0	0	56	0	936	5	42
8	12	2	9	4	10	10	28	1	945	0
9	0	0	1	0	0	20	1	31	0	941

PERCENTAGE CONFUSION MATRIX:

Predicted / Actual	0	1	2	3	4	5	6	7	8	9
0	78.9	0.5	2.7	2.9	0.0	0.2	17.5	0.0	0.7	0.0
1	0.3	97.0	0.4	1.4	0.3	0.0	0.4	0.0	0.1	0.0
2	1.4	0.4	72.2	1.3	11.0	0.0	15.3	0.0	0.6	0.0
3	4.8	2.7	1.1	82.4	3.6	0.1	5.5	0.0	1.1	0.1
4	0.7	0.5	13.1	3.1	72.4	0.0	12.9	0.0	0.3	0.0
5	0.4	0.0	0.0	0.0	0.2	93.8	0.0	3.1	0.7	1.5
6	8.6	0.3	8.1	3.5	6.9	0.0	64.3	0.0	1.5	0.1
7	0.1	0.1	0.2	0.0	0.0	5.8	0.0	89.7	0.5	4.2
8	1.2	0.2	0.9	0.4	1.0	1.0	3.3	0.1	92.6	0.0
9	0.0	0.0	0.1	0.0	0.0	2.1	0.1	3.0	0.0	94.7

Classification accuracy = 84.01%

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.

Confusion matrices to show the classification accuracy of our trained SVM model on the test set

FREQUENCY CONFUSION MATRIX:

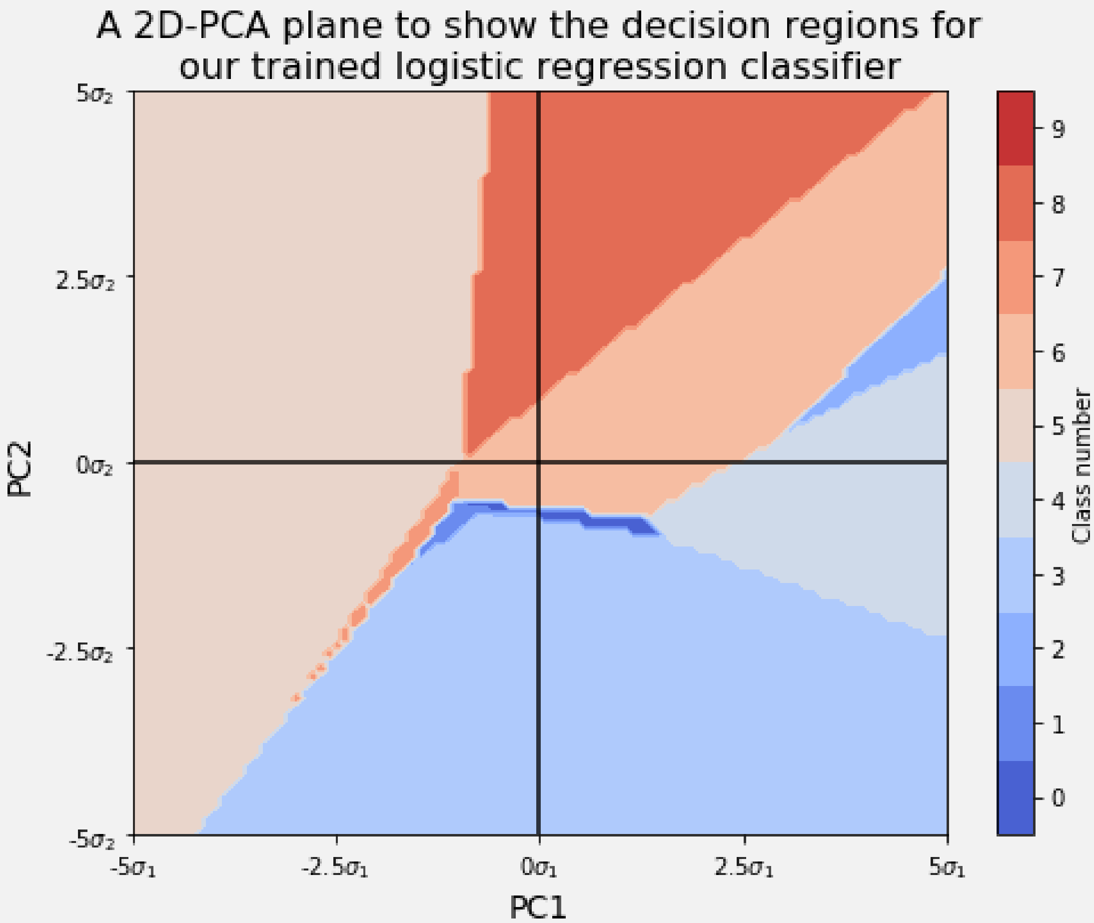
Predicted / Actual	0	1	2	3	4	5	6	7	8	9
0	845	4	15	32	1	0	185	0	3	0
1	2	951	2	6	0	0	1	0	1	0
2	8	7	748	12	98	0	122	0	8	0
3	51	31	11	881	36	1	39	0	5	0
4	4	5	137	26	775	0	95	0	2	0
5	4	0	0	0	0	914	0	34	4	22
6	72	1	79	40	86	0	533	0	13	0
7	0	0	0	0	0	57	0	925	4	47
8	14	1	8	3	4	2	25	0	959	1
9	0	0	0	0	0	26	0	41	1	930

PERCENTAGE CONFUSION MATRIX:

Predicted / Actual	0	1	2	3	4	5	6	7	8	9
0	77.9	0.4	1.5	3.0	0.1	0.0	22.5	0.0	0.3	0.0
1	0.2	98.8	0.2	0.6	0.0	0.0	0.1	0.0	0.1	0.0
2	0.7	0.7	74.6	1.1	9.4	0.0	14.8	0.0	0.8	0.0
3	4.7	3.2	1.1	83.5	3.4	0.1	4.7	0.0	0.5	0.0
4	0.4	0.5	13.7	2.5	74.2	0.0	11.5	0.0	0.2	0.0
5	0.4	0.0	0.0	0.0	0.0	93.5	0.0	3.3	0.4	2.2
6	6.6	0.1	7.9	3.8	8.2	0.0	64.7	0.0	1.3	0.0
7	0.0	0.0	0.0	0.0	0.0	5.8	0.0	89.5	0.4	4.7
8	1.3	0.1	0.8	0.3	0.4	0.2	3.0	0.0	94.3	0.1
9	0.0	0.0	0.0	0.0	0.0	2.7	0.0	4.0	0.1	93.2

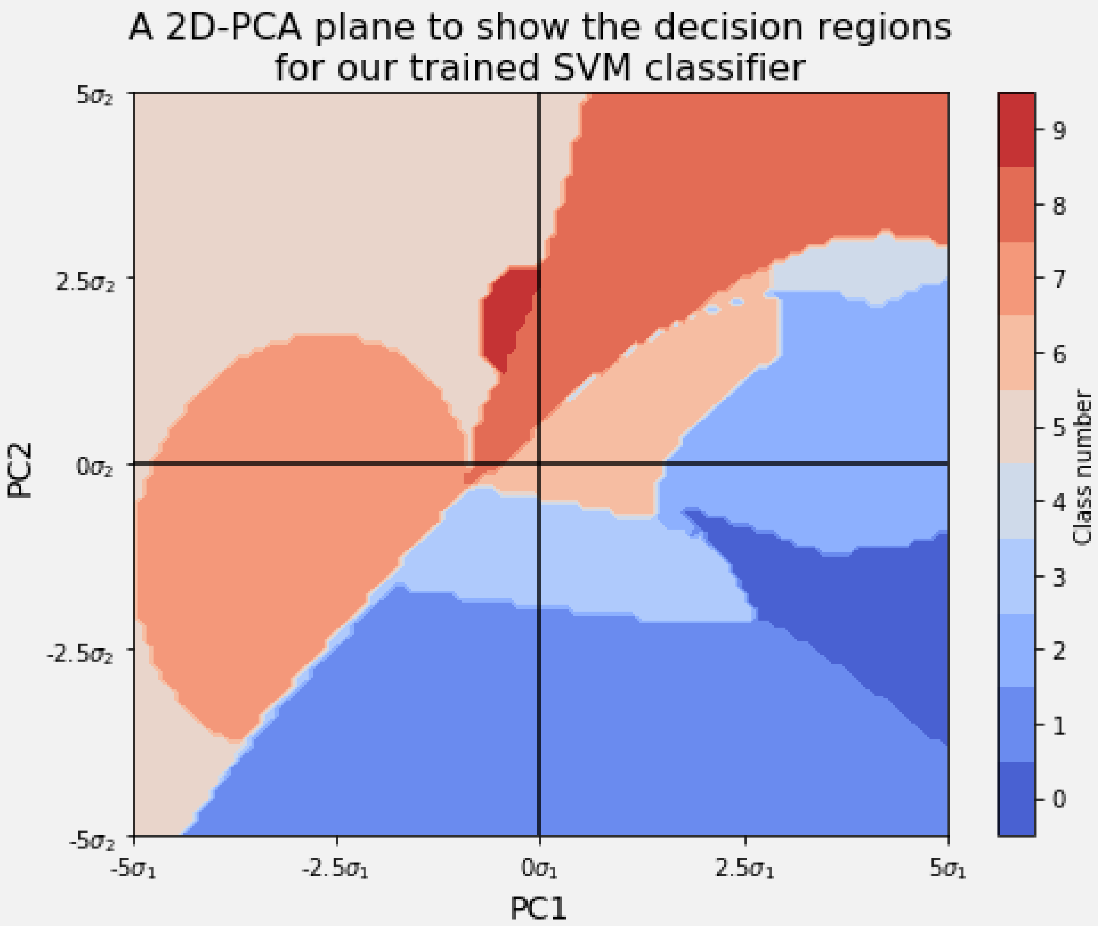
Classification accuracy = 84.412%

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



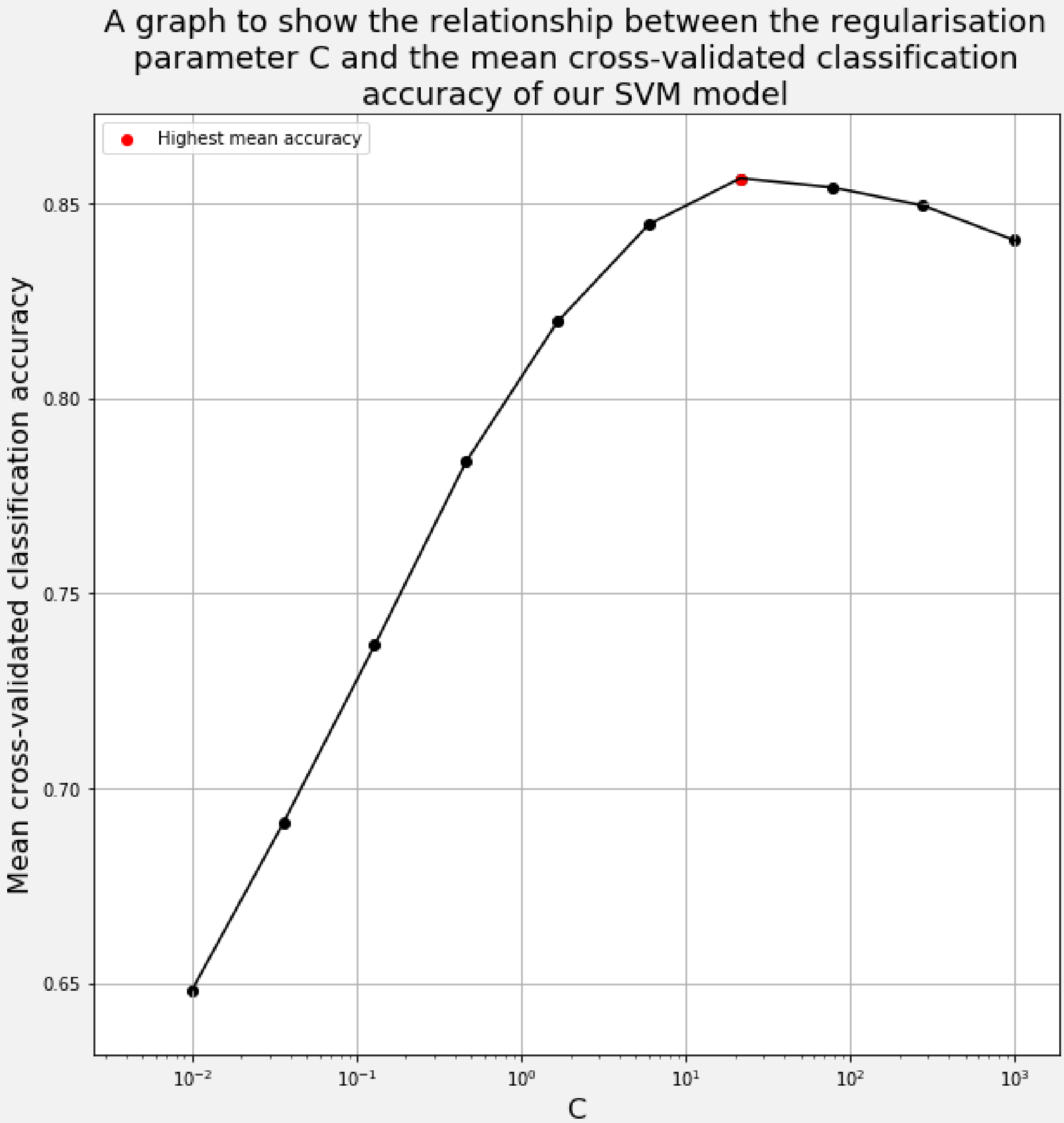
Discuss your findings briefly...

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.



Discuss findings briefly...

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`.



Optimal accuracy achieved and it's respective value of C:

$Accuracy \approx 85.65\%$ where $C = 10^{\frac{4}{3}}$

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.

Classification accuracy of our trained SVM model using the optimal value of C found ($10^{\frac{4}{3}}$)

Training accuracy $\approx 90.842\%$
Testing accuracy $\approx 87.65\%$

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.

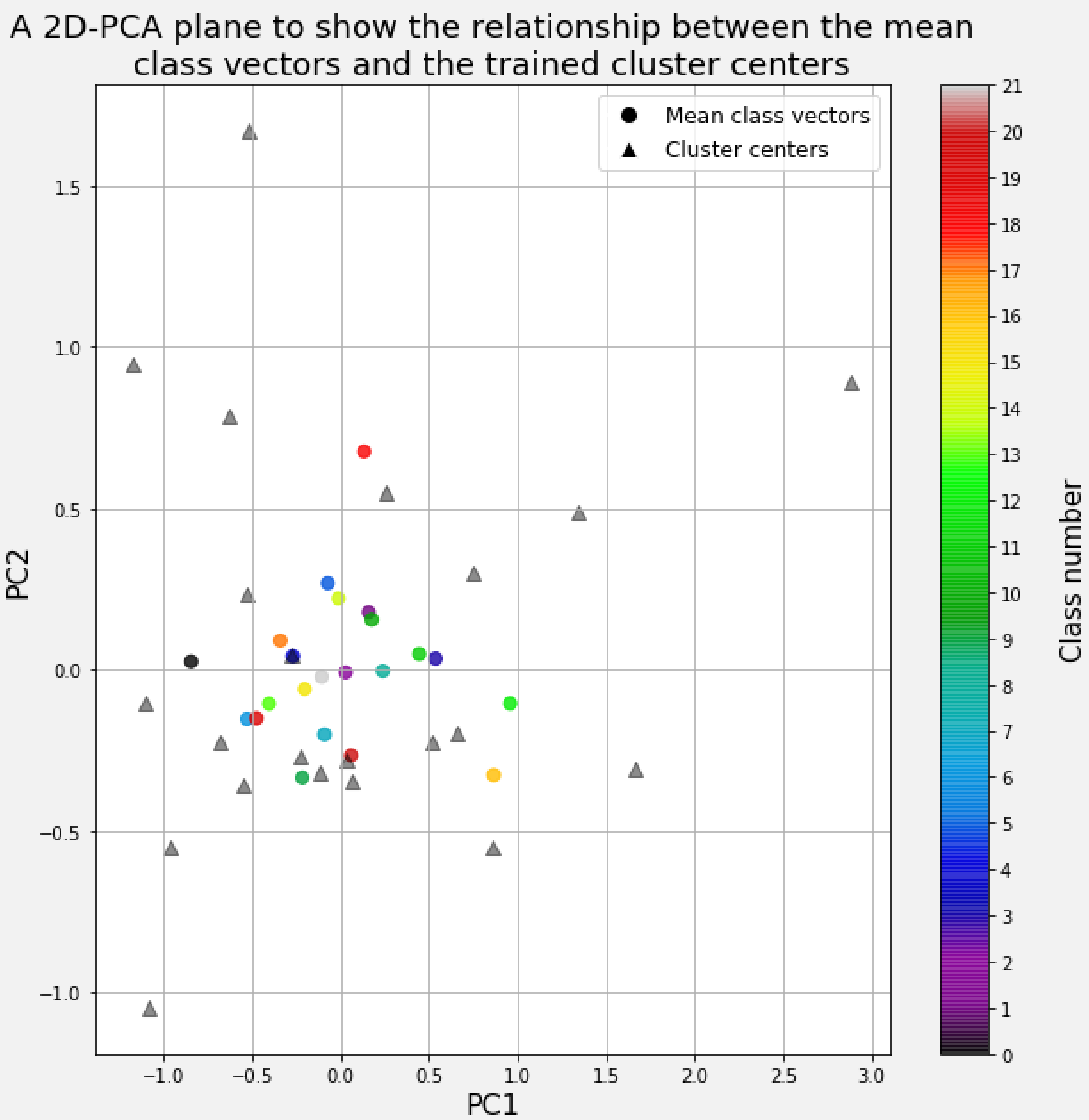
Metrics for our trained k-means clustering model

```
Sum of squared distances (Euclidean) of samples to their closest cluster center:
38185.81698349466

Number of samples for each cluster:
Cluster 1 = 1018
Cluster 2 = 1125
Cluster 3 = 1191
Cluster 4 = 890
Cluster 5 = 1162
Cluster 6 = 1332
Cluster 7 = 839
Cluster 8 = 623
Cluster 9 = 1400
Cluster 10 = 838
Cluster 11 = 659
Cluster 12 = 1276
Cluster 13 = 121
Cluster 14 = 152
Cluster 15 = 950
Cluster 16 = 1971
Cluster 17 = 1251
Cluster 18 = 845
Cluster 19 = 896
Cluster 20 = 930
Cluster 21 = 1065
Cluster 22 = 1466
```

Question 3 continued on next page. . .

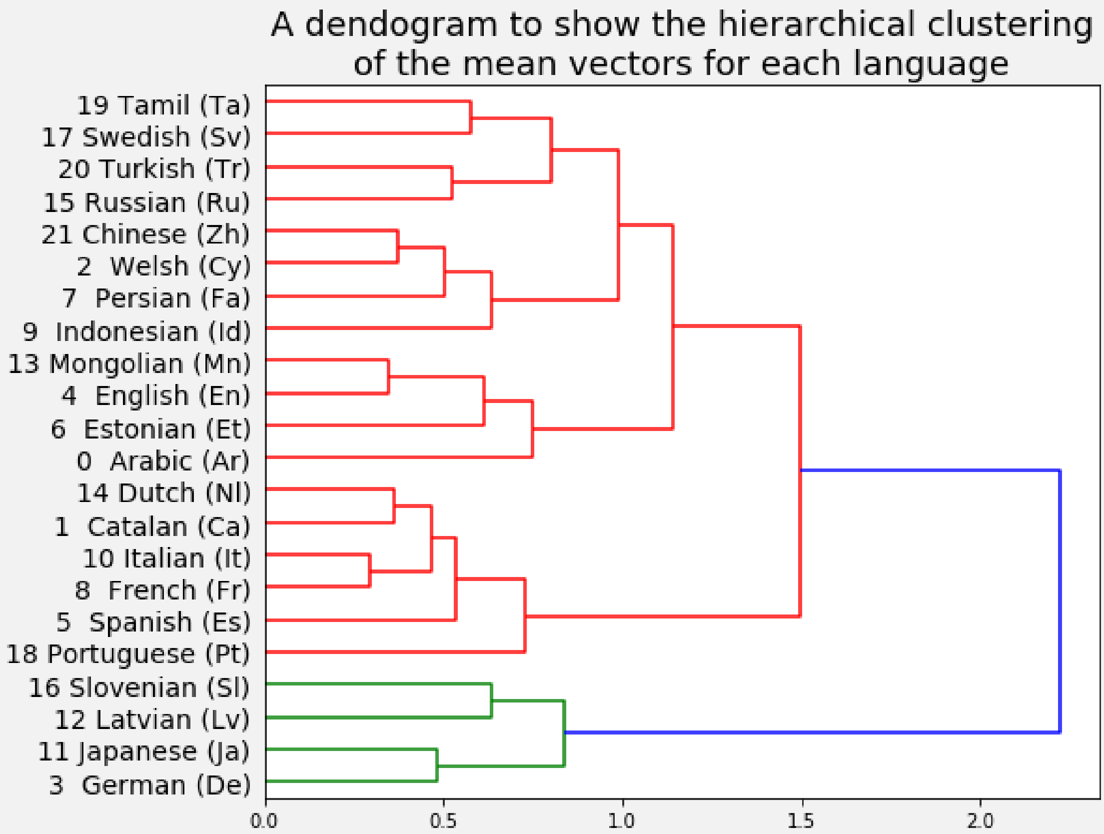
3.2 (3 points) Using the training set only, calculate the mean vector for each language, and plot the mean vectors of all the 22 languages on a 2D-PCA plane, where you apply PCA on the set of 22 mean vectors without applying standardisation. On the same figure, plot the cluster centres obtained in Question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



Are the mean vectors and clusters similar? Discuss your findings briefly...
We can see that most of the mean class vectors and data lie in the same general region of the plot with a few clusters more sparsely decorated. We can imagine these sparse clusters account for outliers in our data given that all the mean classes are in the same general region.

There is a cluster directly on top of class 4 which exemplifies the accuracy of this clustering. We can imagine this class must have quite consistent data samples throughout the dataset meaning small distances between samples and ultimately a very accurate cluster center.

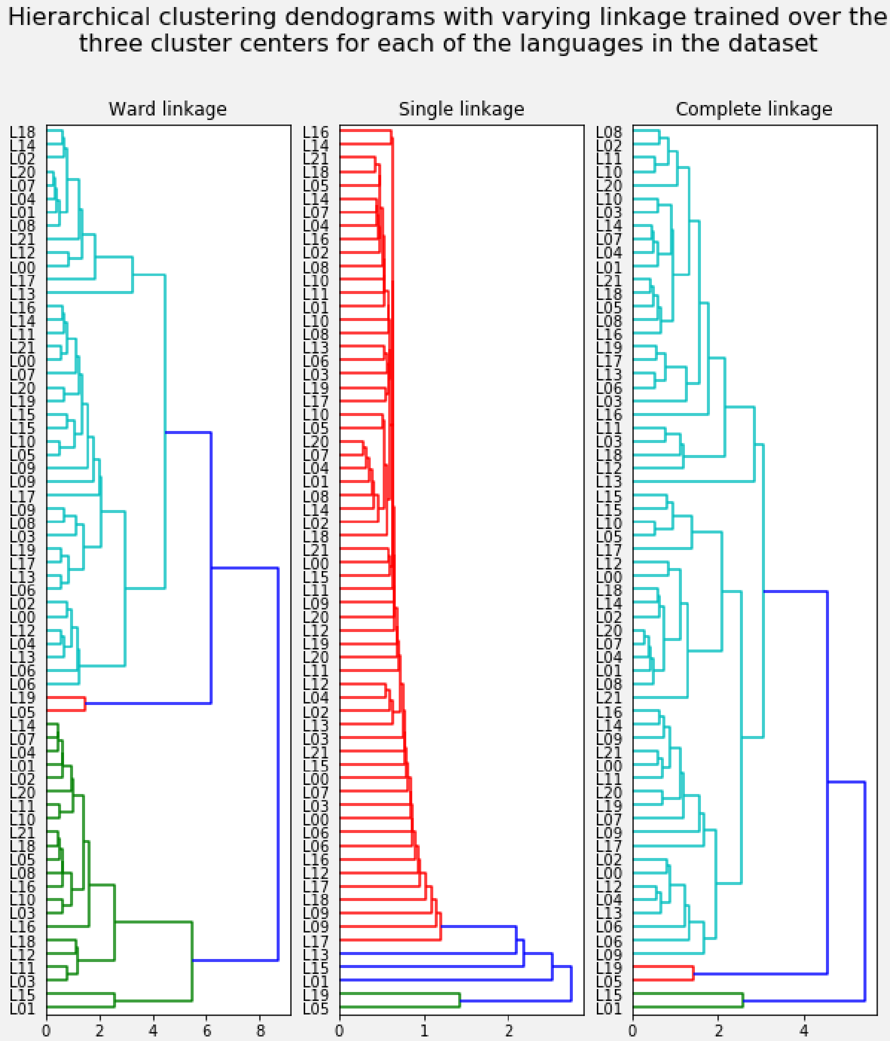
3.3 (3 points) We now apply hierarchical clustering on the training data set to see if there are any structures in the spoken languages.



This dendogram illustrates how each cluster from our dataset is composed and thus ultimately represents the hierarchy of similarities between the classes in our data.

From this we can deduce that Italian and French have the most similar mean vectors and thereby are the most "similar" languages. In contrast we can see that Latvian has the most dissimilar mean vector from all other languages and thereby is the most "unique" language.

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



Discuss results briefly...

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.,

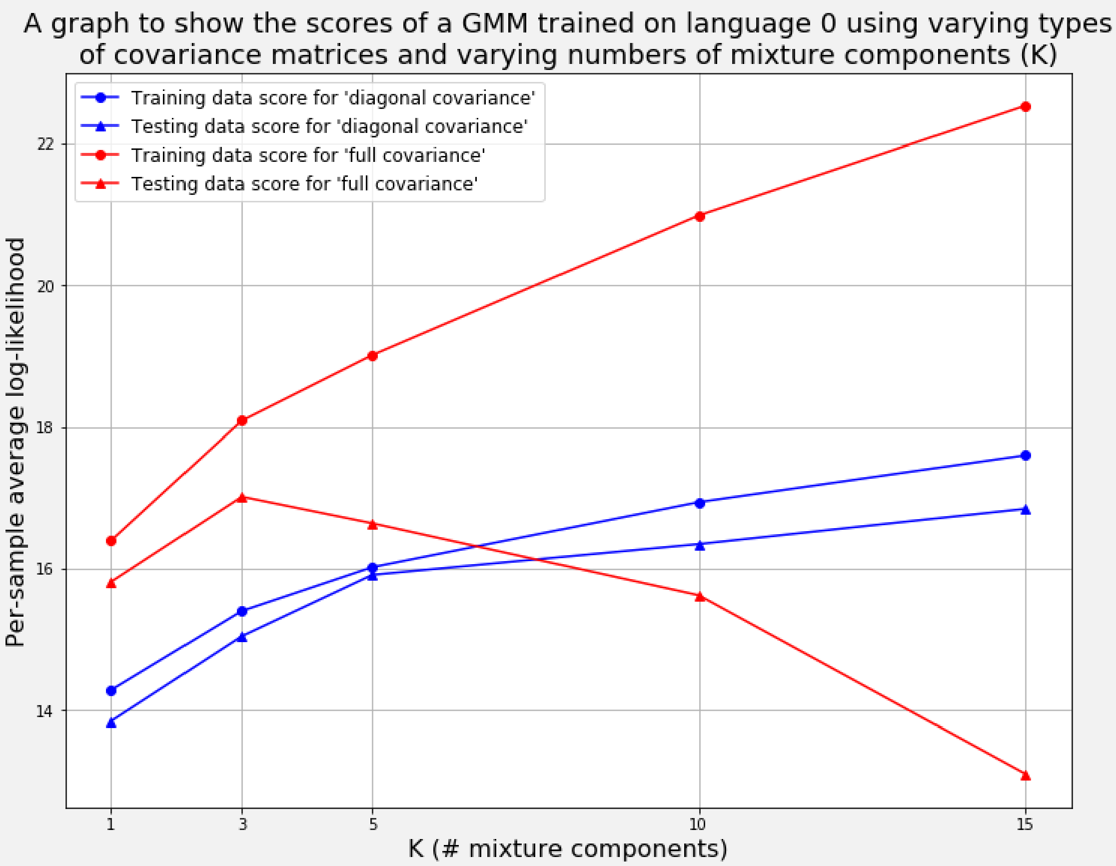


Table to show the average per-sample log-likelihood scores for varying parameters of a GMM

Language 0 data	Σ type	K = 1	K = 3	K = 5	K = 10	K = 15
Training	Diagonal	14.28	15.399	16.014	16.895	17.653
Testing	Diagonal	13.843	15.041	15.882	16.375	19.942
Training	Full	16.394	18	19.129	21.018	22.889
Testing	Full	15.811	16.895	16.704	15.19	10.787

Discuss your findings...