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

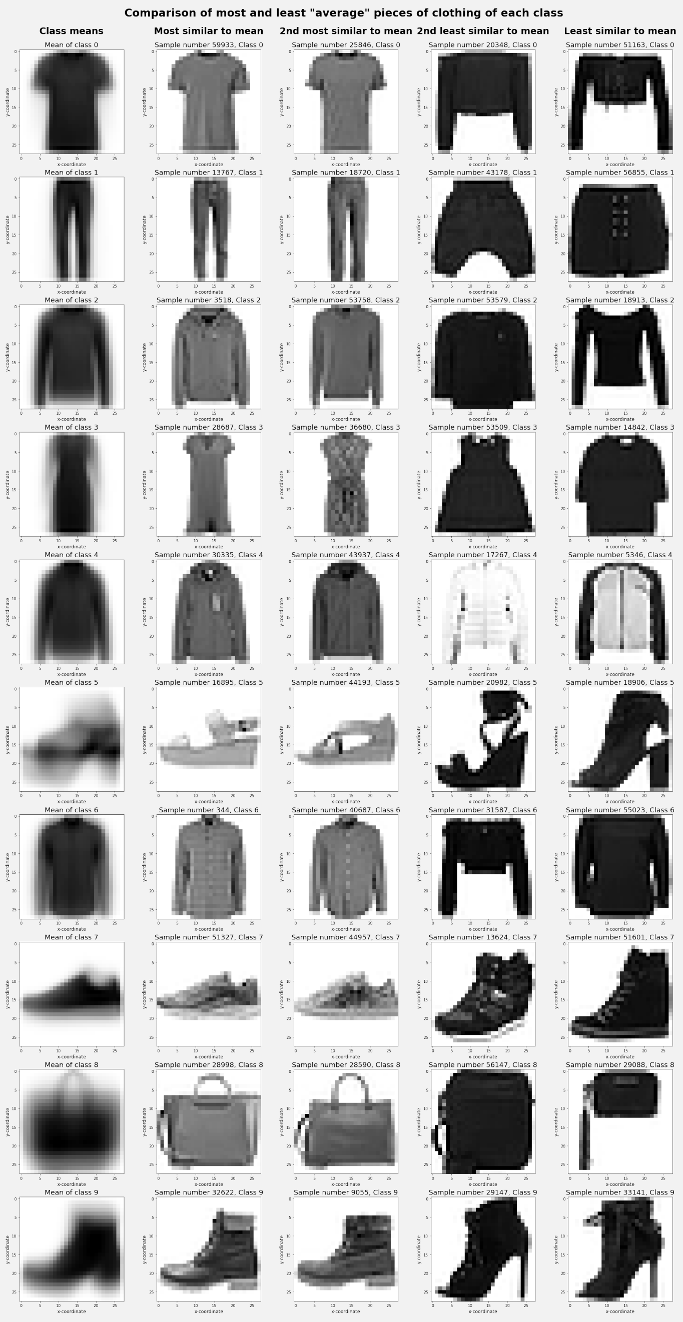
In this question we employ PCA to analyse image data

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 row of Xtrn_nm contains the same first 4 elements as its last row: $-3.14 \cdot 10^{-6}, -2.27 \cdot 10^{-5}, -1.18 \cdot 10^{-4}$ and $-4.07 \cdot 10^{-4}$.

This result makes it seem very likely that the according elements in Xtrn_orig are all 0, lying slightly below the (rather small) mean for their corresponding attribute. Since we are dealing with a bitmap and since the first 4 attributes correspond to pixels at the border of our grid, this is not unexpected.

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.



For each class, the "mean image" looks like what we would an expect an instance of that class to look like, but blurry, especially around the edges. This teaches us that most instances of the same class have very similar shapes. Some classes (e.g. class 5) seem to show more variety than others (e.g. class 0), evidenced by a blurrier mean. The "most average" instances generally look like we removed the blur from the mean. The "least average" ones feature shapes that appear to be less typical for the class as the mean does not represent them.

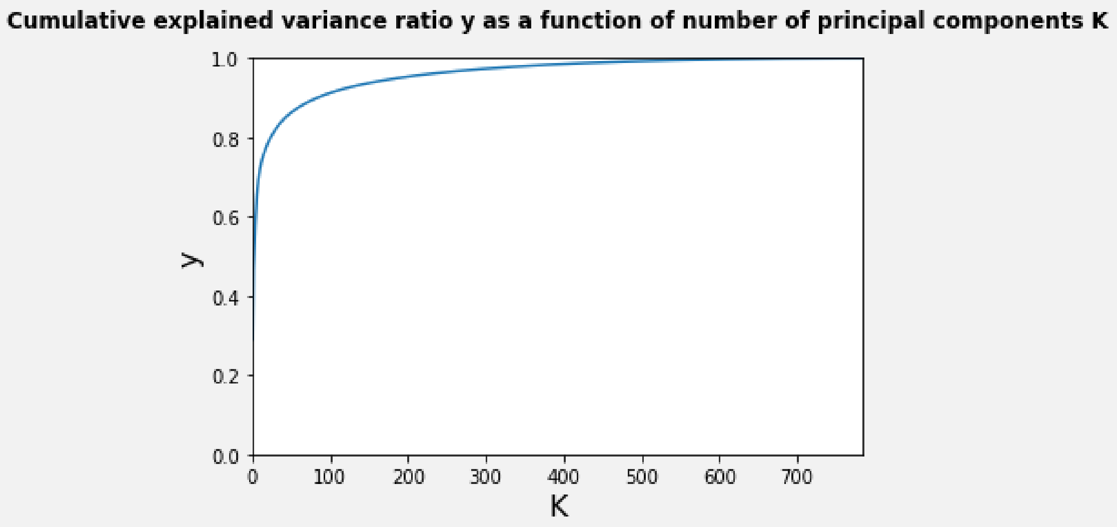
1.3 (3 points) Apply Principal Component Analysis (PCA) to the data of `Xtrn_nm` using sklearn.decomposition.PCA, and report the variances of projected data for the first five principal components in a table. Note that you should use `Xtrn_nm` instead of `Xtrn`.

The following table displays the variance of projected data for the first five principal components, from largest to fifth-largest.

Ranking	Variance
1	19.81
2	12.11
3	4.11
4	3.38
5	2.62

Question 1 continued on next page...

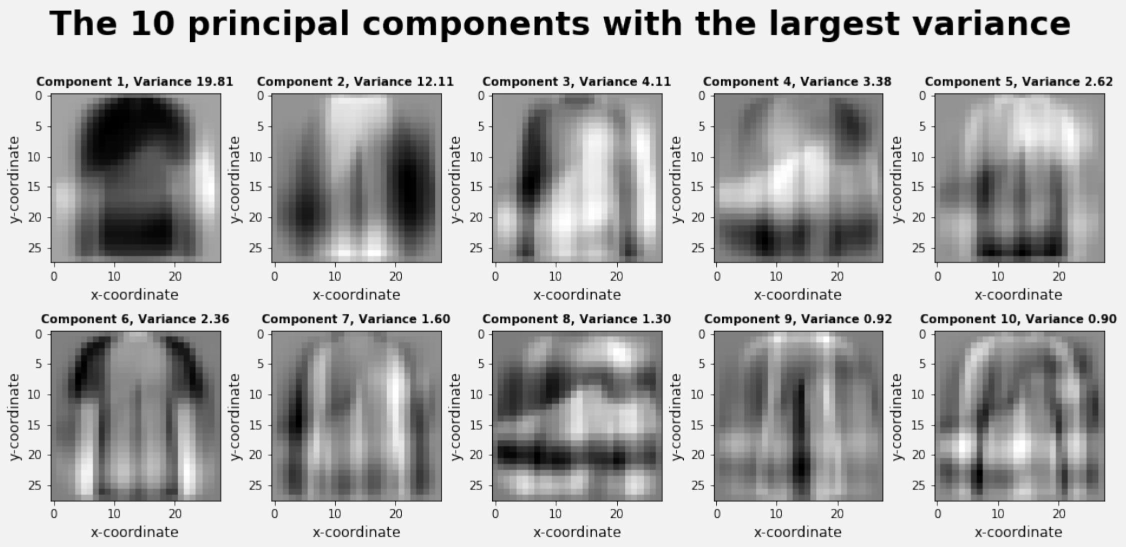
1.4 (3 points) Plot a graph of the cumulative explained variance ratio as a function of the number of principal components, K, where $1 \leq K \leq 784$. Discuss the result briefly.



We notice that most of the variance in the data can be "explained" by a relatively small number of principal components. E.g. the first 5 components account for 61.62%, the first 50 for 86.27%, the first 84 for 90.06% and the first 187 for 95.00% of the total variance. Since the 784 components are sorted by their explained variance, the cumulative explained variance of course rises less steeply for larger K. The "knee of the curve" lies at about K=50. As we would expect, the cumulative explained variance ratio slowly approaches 1 as K approaches 784, the total number of components. At K=784, all of the variance is "accounted for".

Question 1 continued on next page. . .

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.



The best way I can describe these principal components is by saying that they look like superimposed images of the mean of one class and the negative image of the mean of another class. I am simplifying things slightly. More than two classes seem to be "present" in some of the components, e.g. in component 4 where we can make out the shape of what looks like a jacket, as well as the "inverted" shapes of a shoe and a pair of trousers. The principal components represent the ways in which instances/images in our dataset vary the most, which in our case is closely related to which type of clothing they are. They allow us to approximate each data instance as a linear combination of the components. E.g. if we tried to only use the first component, an image of a jacket would have a positive coefficient in the new 1D feature space. Because we centered all 784 attributes at the mean, a typical image of a shoe would have a negative value.

Question 1 continued on next page. . .

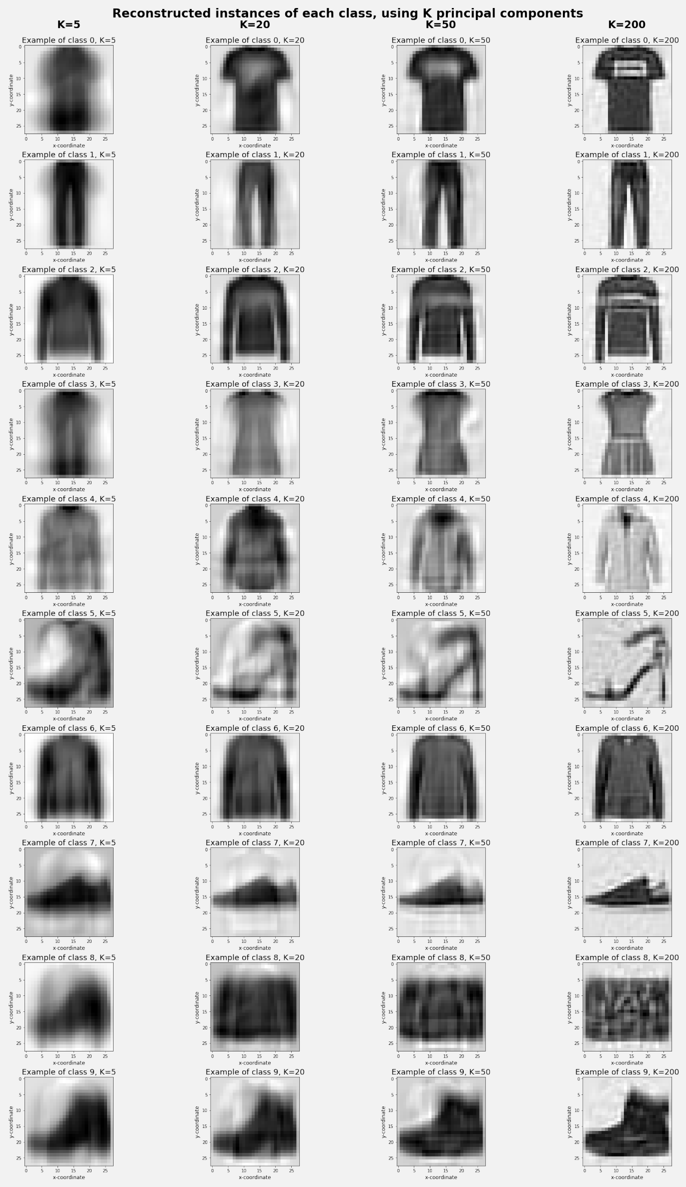
1.6 (5 points) Using `Xtrn_nm`, for each class and for each number of principal components $K = 5, 20, 50, 200$, apply dimensionality reduction with PCA to the first sample in the class, reconstruct the sample from the dimensionality-reduced sample, and report the Root Mean Square Error (RMSE) between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

The following table contains the RMSE between the first sample of each class and its reconstruction using K principal components.

Class	K=5	K=20	K=50	K=200
0	0.2561	0.1501	0.1277	0.0605
1	0.1980	0.1404	0.0954	0.0359
2	0.1987	0.1456	0.1233	0.0802
3	0.1457	0.1072	0.0837	0.0565
4	0.1182	0.1026	0.0879	0.0458
5	0.1811	0.1587	0.1429	0.0888
6	0.1295	0.0959	0.0724	0.0458
7	0.1656	0.1278	0.1067	0.0642
8	0.2234	0.1450	0.1234	0.0904
9	0.1835	0.1511	0.1217	0.0726

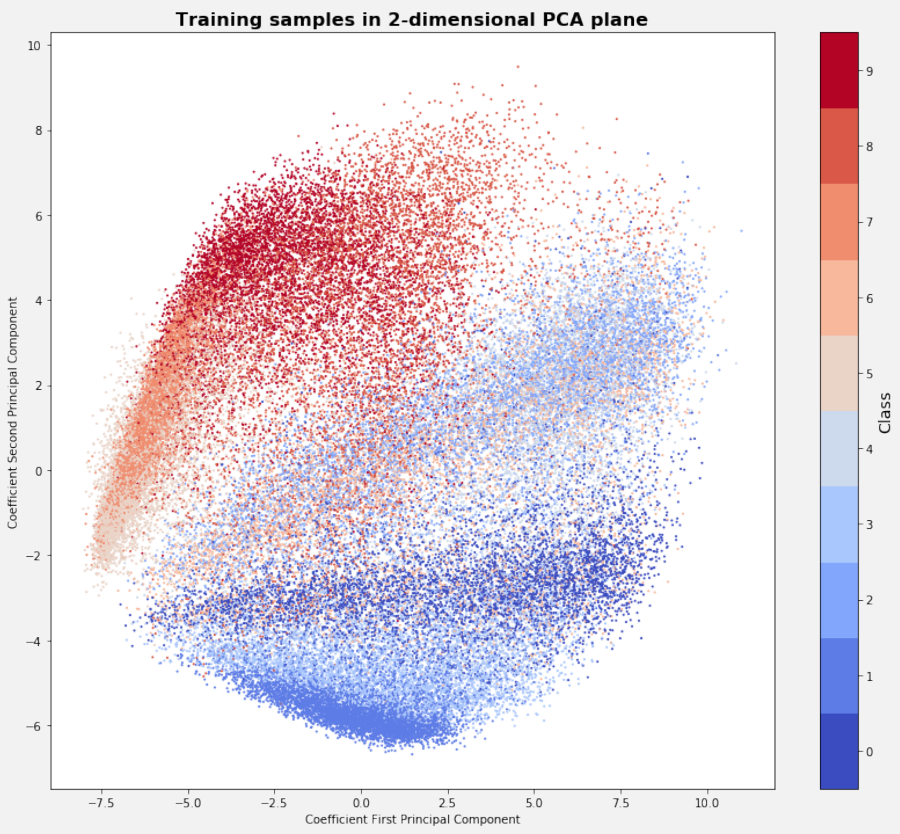
Question 1 continued on next page...

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 $K = 5,\ 20,\ 50,\ 200$.



The reconstructed images match the original more closely as K grows large. For small values of K, the images are blurry and we can generally make out the shape of what looks like instances of another class (or the negative image thereof), superimposed with a shape that resembles the original instance. For larger K, the "superimposed negative images" mostly disappear, however there is still some noise in the images. Some instances are clearly more difficult to reconstruct than others. E.g. the reconstruction of the class 8 instance for K=5 is bad, because the first 5 principal components do not include any shape that resembles the original. This is reflected by the according variance in our table.

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.



While our 2D-representation does not make the data linearly separable, and while we observe a strong overlap between some of the classes (e.g. 5 and 6), we do notice that most instances of each class are mapped to somewhat similar points in our 2D space. E.g. members of class 1 tend to have a coefficient relatively close to 0 in the 1st, and a large negative coefficient in the second component (Note: Since the 2nd component from Q1.5 features the "inverse shape" of a pair of trousers, this is not surprising). So while a classifier that tries to guess the class of an instance based on its 2D representation could not be very accurate, it could at least make an "informed guess" - which is remarkable, given that we started with 784 dimensions.

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.

‘The classifier reaches a classification accuracy of 84.01%.
Our computations yield the following confusion matrix:
(Note that the entries of each row add up to 1000, giving us the number of samples of each class in the testing data.)

		Predicted Class										
		0	1	2	3	4	5	6	7	8	9	Total
	0	819	3	15	50	7	4	90	1	11	0	1000
	1	5	953	4	27	5	0	3	1	2	0	1000
	2	27	4	731	11	133	0	82	2	9	1	1000
	3	31	15	14	866	33	0	37	0	4	0	1000
	4	0	3	115	38	760	2	72	0	10	0	1000
Real class	5	2	0	0	1	0	911	0	56	10	20	1000
	6	147	3	128	46	108	0	539	0	28	1	1000
	7	0	0	0	0	0	32	0	936	1	31	1000
	8	7	1	6	11	3	7	15	5	945	0	1000
	9	0	0	0	1	0	15	1	42	0	941	1000
	Total	1038	982	1013	1051	1049	971	839	1043	1020	994	10000

Question 2 continued on next page. . .

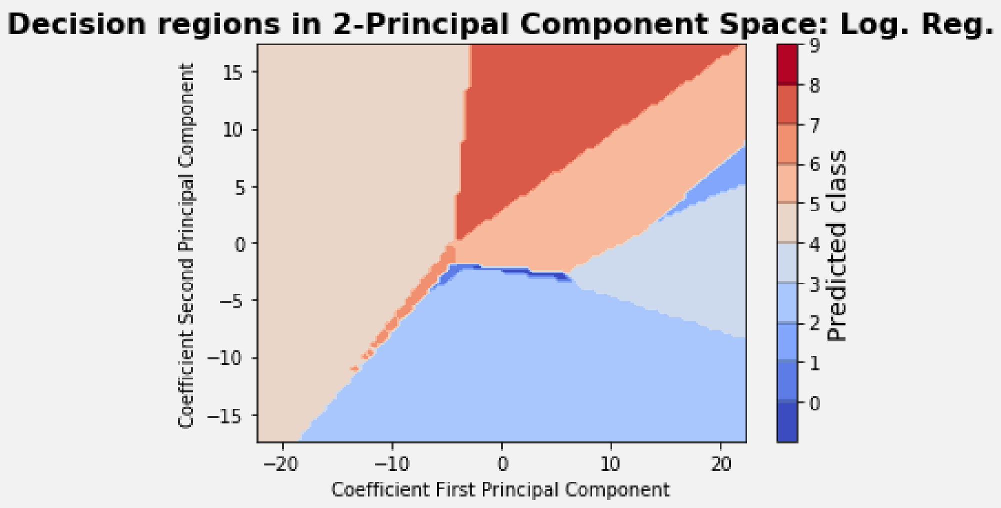
2.2 (3 points) Carry out a classification experiment with SVM classifiers, and report the mean accuracy and confusion matrix (in numbers) for the test set.

The classifier reaches a classification accuracy of 84.61%.
Our computations yield the following confusion matrix:

		Predicted Class										
		0	1	2	3	4	5	6	7	8	9	Total
	0	845	2	8	51	4	4	72	0	14	0	1000
	1	4	951	7	31	5	0	1	0	1	0	1000
	2	15	2	748	11	137	0	79	0	8	0	1000
	3	32	6	12	881	26	0	40	0	3	0	1000
	4	1	0	98	36	775	0	86	0	4	0	1000
Real class	5	0	0	0	1	0	914	0	57	2	26	1000
	6	185	1	122	39	95	0	533	0	25	0	1000
	7	0	0	0	0	0	34	0	925	0	41	1000
	8	3	1	8	5	2	4	13	4	959	1	1000
	9	0	0	0	0	0	22	0	47	1	930	1000
	Total	1085	963	1003	1055	1044	978	824	1033	1017	998	10000

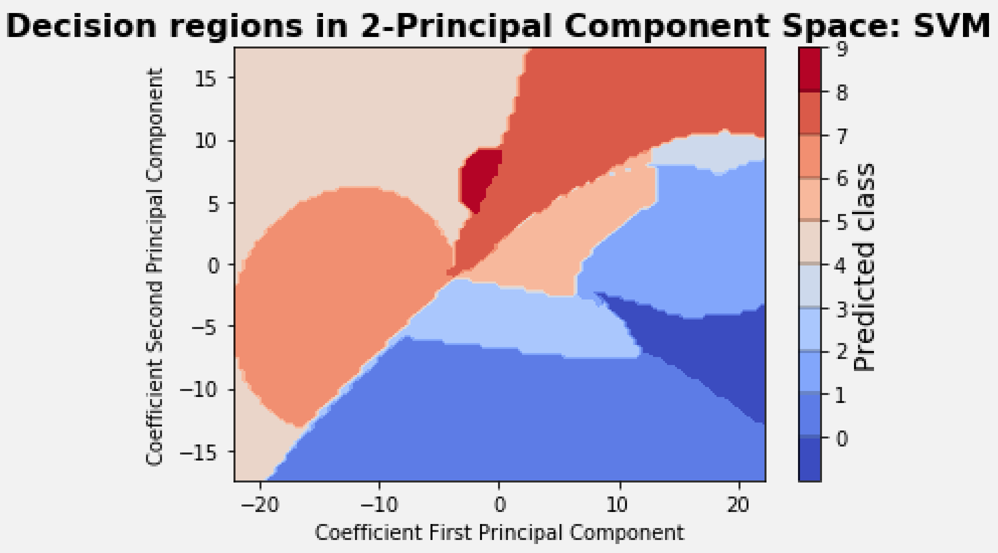
Question 2 continued on next page...

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



Because our Logistic Regression model is a linear classifier, it is no surprise that the decision boundaries for our projected data are linear. We notice that the decision regions vaguely resemble the scatter plot of the training data from Q1.8 for coefficients relatively close to 0. Some classes (especially 3 and 5) take in massive regions and are thus being "over-represented" in this plot. This is amplified by the large range of coefficients we are considering - decision regions seem to continue indefinitely as we depart from the range of coefficients we observed in our training data. The classes 0,1, and 7 (and 2 to some extent) on the other hand only have a tiny region corresponding to each of them, and class 9 does not have a region at all! There are two other possible explanations for this imbalance: Maybe the instances of some classes are "much closer together", projected to our 2D space, than others (which would not quite explain the absence of class 9). But mostly, we should not expect this 2D representation to be very representative of the performance of our classifier in the entirety of our 784-dimensional feature space.

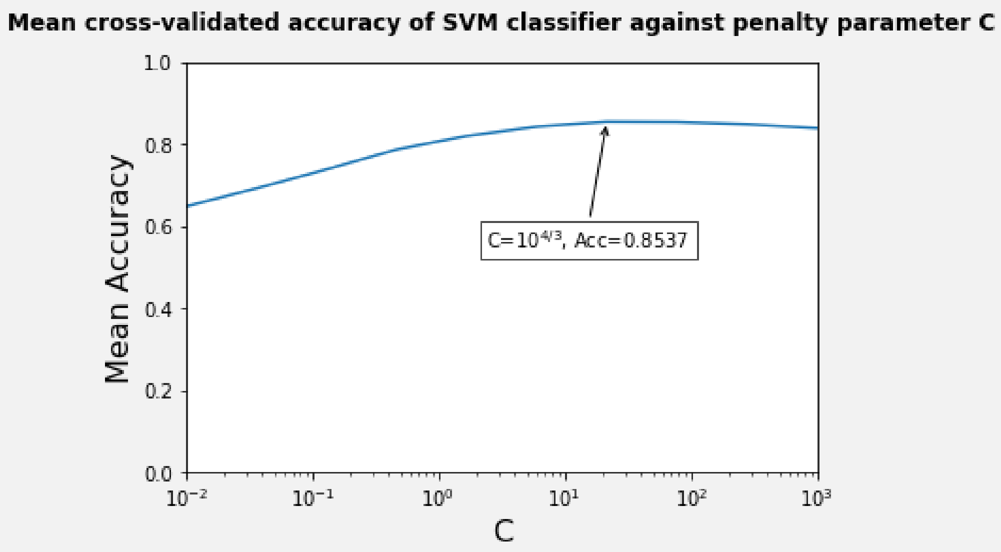
2.4 (4 points) Using the same method as the one above, plot the decision regions for the SVM classifier you trained in Question 2.2. Comparing the result with that you obtained in Question 2.3, discuss your findings briefly.



There are quite a few differences beteen the SVM and the Log.Reg. regions. In the SVM case, there is now a (rather small) region corresponding to class 9! Some regions seem to have swapped positions. More importantly, the proportions of the regions have changed rather drastically. Some regions (e.g. 2, 7) are much larger, others(e.g. 3, 5, although region 5 is still large) have become much smaller. In particular, Class 1 now corresponds to the largest, instead of the smallest region. Overall, although some classes are still under-/over-represented, the distribution of areas is much more even than in the Log.Reg. case, where we observed many tiny/huge regions. Furthermore, we notice that the decision boundaries between regions are no longer linear. Lastly, I will say that the two plots are nevertheless similar in some ways. E.g. in both cases, regions 0 to 4 occupy most of the lower right half of the plot, whereas regions 5, 7, 8 and 9 occupy most of the upper left half (region 6 lies "on the diagonal").

Question 2 continued on next page…

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`.



We obtain the highest mean cross-validated accuracy for $C = 10^{\frac{4}{3}}$, with a "score" of 85.37%.

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.

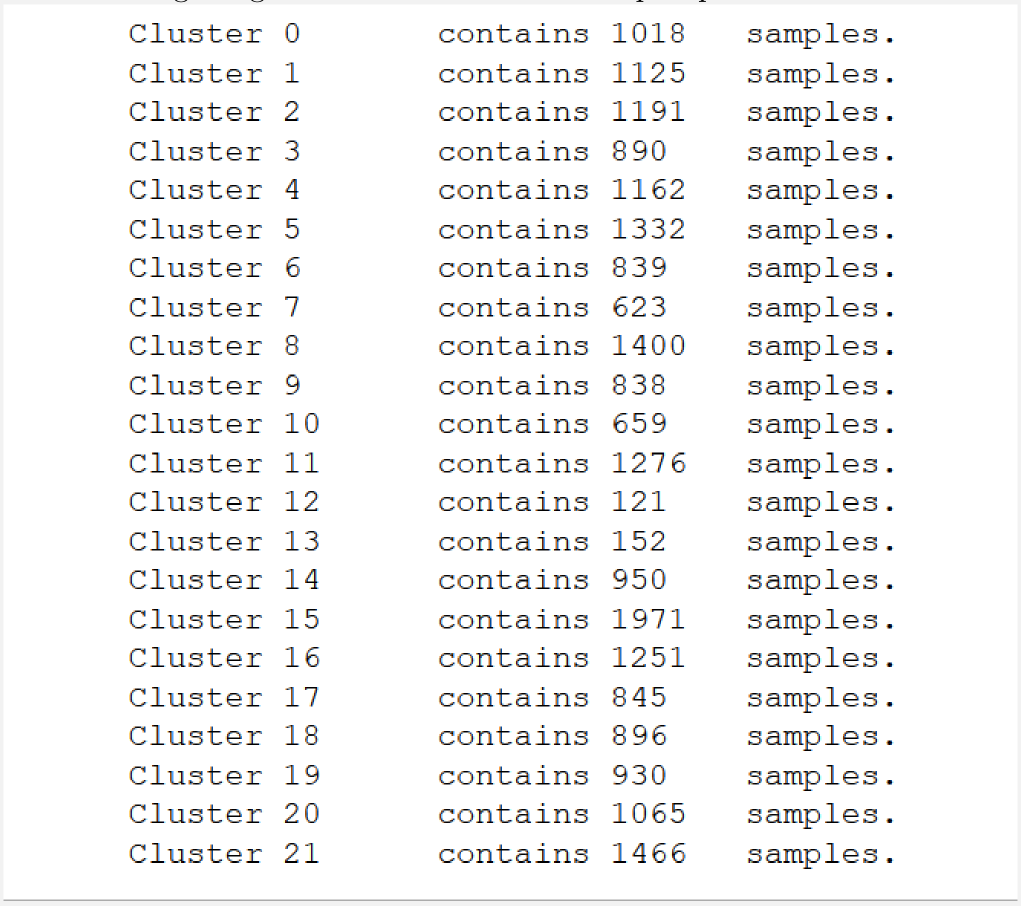
On the training set, we reach an accuracy of 90.84%.
On the testing set, we reach an accuracy of 87.65%.

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.

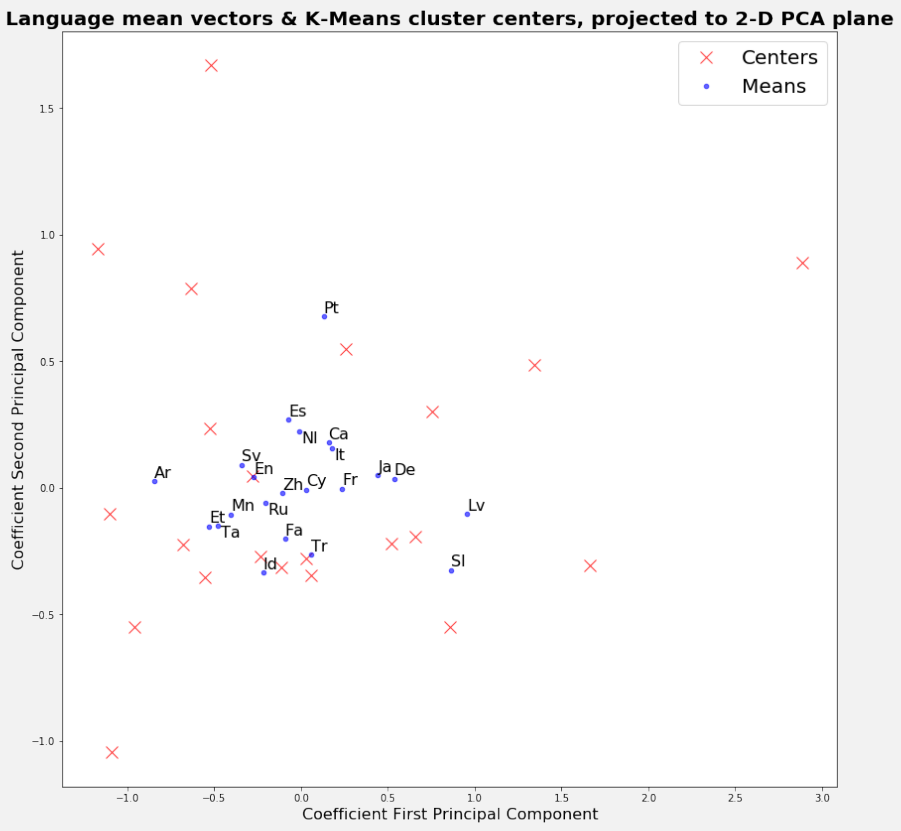
The sum of squared distances of samples to their closest cluster center is 38185.82. The following image shows the number of samples per cluster.



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

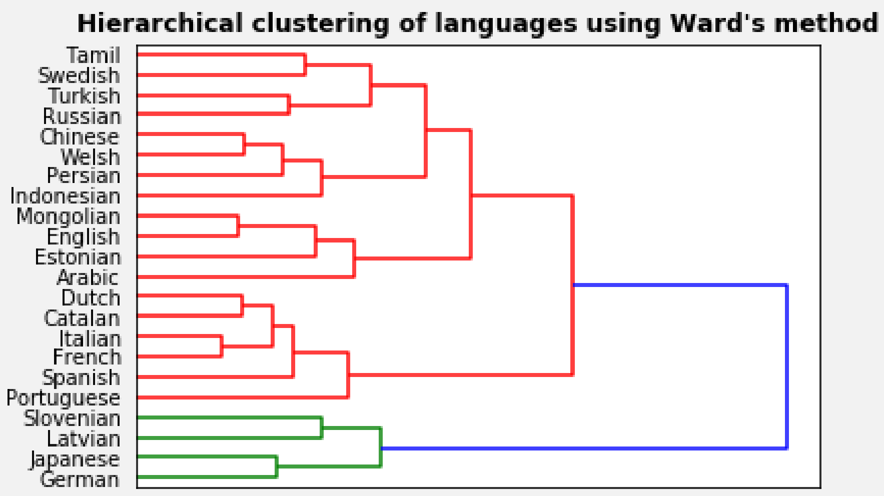
Question 3 continued on next page...

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.



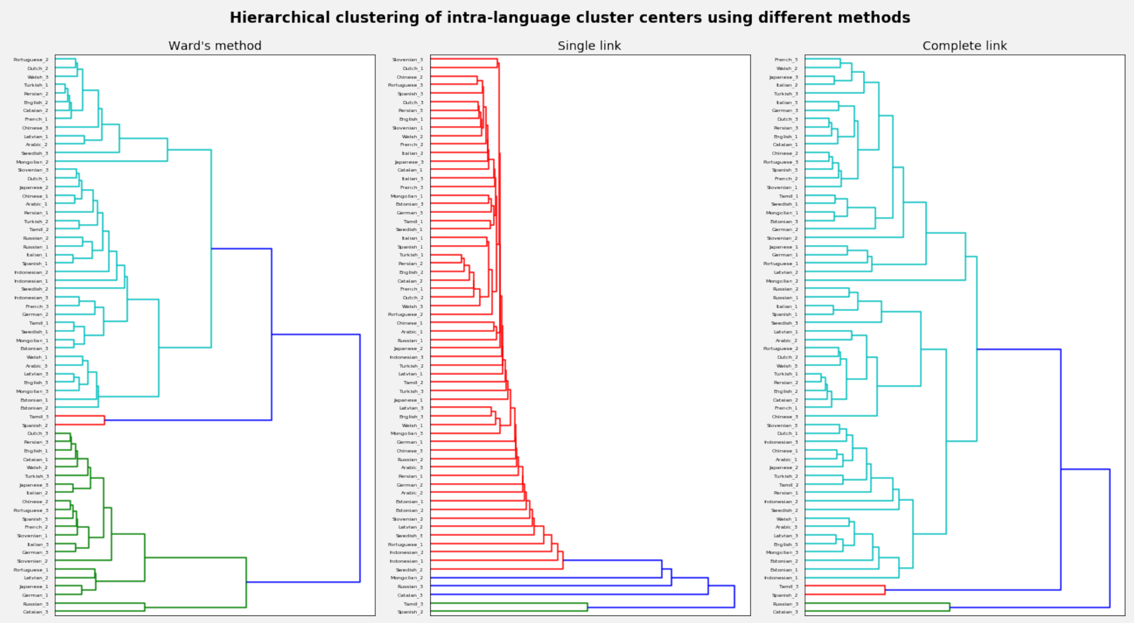
The cluster centers and the language means are only similar in that they both of course lie in roughly the same region of our 2-D PCA plane. There does not seem to be any 1-to-1 correspondence between means and centers however. The means are "huddled together" more closely than the centers, whereas the centers are "less dense". Some centers lie outside the main group of meanscenters, presumably corresponding to groups of outliers that do not all stem from the same class.

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 dendrogram provides us with an "ontology" of languages - one can say that it defines multiple flat clusterings, each with a different granularity. I.e. if we choose a distance threshold and cut the tree at the according point, we get a flat clustering, s.t. the number of clusters depends on the threshold. This way, we can e.g. single out a group of languages which the algorithm perceives to be standing out from the rest: Slovenian, Latvian, Japanese and German, in contrast to the remaining 18 languages. We could do so by cutting the tree at the rightmost (or "1st") juncture. Or we may e.g. decide to cut the tree at the leftmost (or "21st") juncture, singling out French and Italian as remarkably similar, whereas all other languages would simply remain in their singleton cluster.
With the exception of the 1st juncture, it appears that clusters are merged with a cluster of similar size (note that we are using a Bottom-up approach) at their corresponding juncture, leading to many "pairs of languages" and ultimately a structure baring some resemblance to evolutionary lineage in Biology.

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

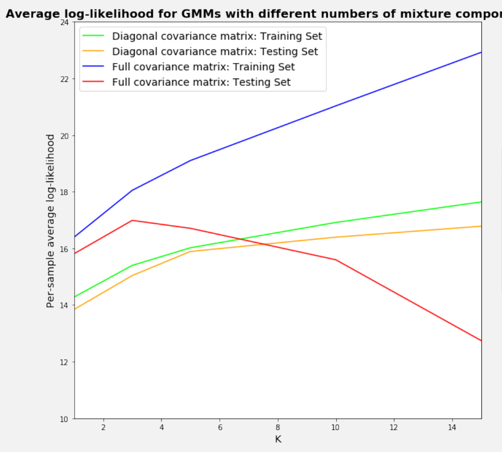


The result we obtain using Single Linkage is exemplary of the method. Using a bottom-up approach, clusters are merged based on the minimum distance of points in the different clusters. Because larger clusters are clearly "more likely" to be close to any given other cluster, this method yields long chains of elements - a very unbalanced dendrogram. Another way of looking at this is that singletons are more likely to be merged with large clusters, rather than e.g. other singletons.

Contrast this with Ward's Method and Complete Linkage which - in this case - appear to yield somewhat similar results. The precise structures of the respective ontologies are not exactly the same, but for both methods it's the case that most clusters are split roughly in half/merged with a similar-sized cluster at their corresponding juncture. This behaviour corresponds to relatively "spherical" clusters in feature space. In the Complete Linkage case, this is obviously to be expected. Two relatively small clusters are likelier to be the "closest" on any given step of the algorithm, since the distance measure is based on least close elements and because clusters with more elements tend to span a larger region in feature space. The fact that Ward's Method, which merges clusters based on smallest "minimum variance", performs similarly in this case is interesting. However, both methods also single out some pairs of "language varieties" at a very coarse granularity - e.g. Tamil_3 Spanish_2 on the rightmost juncture using Ward's Method, and on the 2nd-to-rightmost juncture using Complete Linkage. This leads me to the conjecture that both Tamil_3 and Spanish_2 are examples of outliers.

Question 3 continued on next page. . .

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. Train.	Diag. Test.	Full Train.	Full. Test.
1	14.28	13.84	16.39	15.81
3	15.40	15.04	18.05	16.99
5	16.02	15.89	19.10	16.71
10	16.91	16.39	21.03	15.60
15	17.64	16.78	22.92	12.74

On the training data, using a full covariance matrix gives better results than using a diagonal one, for any given K. And on the testing data, the best log likelihood is achieved using a full covariance matrix and K=3. However, we quickly run the risk of overfitting. Whereas the log likelihood on the training data keeps increasing as K grows (for both types of covariance matrix but more steeply in the full-cov case), the score of the full covariance model on the testing data declines once K exceeds 3, more rapidly as K grows. For K=10 and 15, the diagonal cov. model performs better on new data than the full cov. model. We know that a GMM model that uses a diagonal covariance matrix will also be overfitting once K is large enough. But for K=15, this is not yet the case, although the gap between the Diagonal-Training and Diagonal-Testing curves slowly begins to grow wider.