

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

Class 0		Sample 59933 	Sample 58923 	Sample 33011 	Sample 51163 
Class 1		Sample 13767 	Sample 18720 	Sample 46375 	Sample 56855 
Class 2		Sample 3518 	Sample 53758 	Sample 53579 	Sample 18913 
Class 3		Sample 28687 	Sample 36680 	Sample 53509 	Sample 14842 
Class 4		Sample 30335 	Sample 43937 	Sample 17267 	Sample 5346 
Class 5		Sample 16895 	Sample 44193 	Sample 20982 	Sample 18906 
Class 6		Sample 344 	Sample 40687 	Sample 41019 	Sample 55023 
Class 7		Sample 51327 	Sample 58102 	Sample 47527 	Sample 51601 
Class 8		Sample 28998 	Sample 43370 	Sample 56147 	Sample 29088 
Class 9		Sample 32622 	Sample 28192 	Sample 26636 	Sample 33141 
	Mean	Closest sample to mean	2nd closest sample to mean	2nd furthest sample from mean	Furthest sample from mean

Analysis of our results

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).

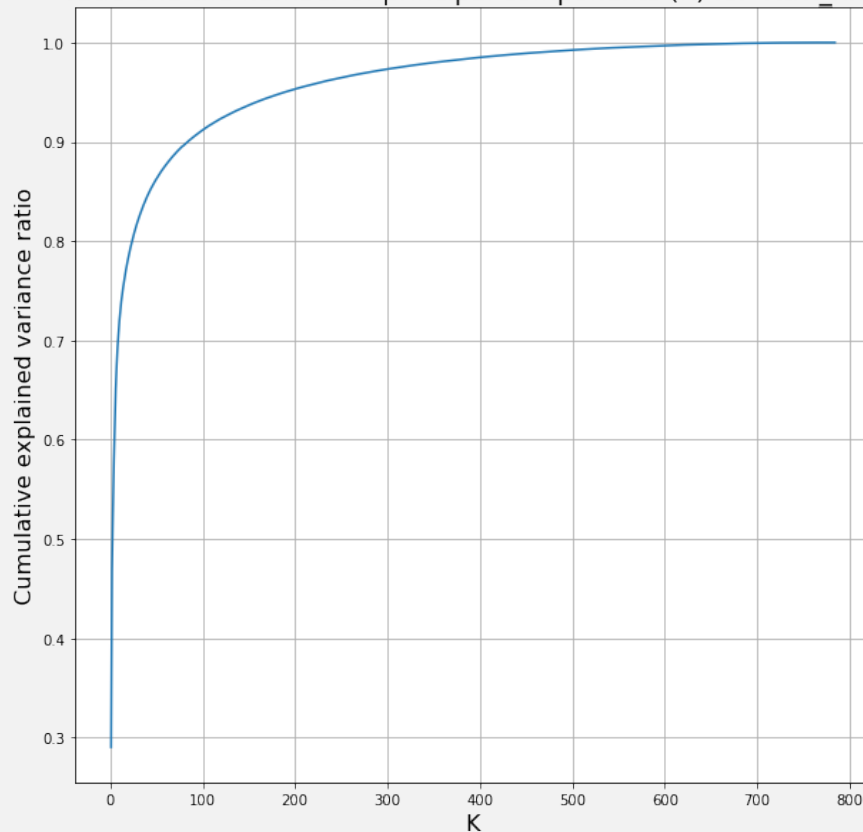
1.3 (3 points) Apply Principal Component Analysis (PCA) to the data of `Xtrn_nm` using `sklearn.decomposition.PCA`, and report the variances of projected data for the first five principal components in a table. Note that you should use `Xtrn_nm` instead of `Xtrn`.

**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

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.

A graph to show the relationship between the cumulative explained variance ratio and the number of principal components (K) for 'Xtrn_nm'



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

***We must note that that the total variance as described in the equation above is the cumulative explained variance from the original 784 dimension dataset, and thereby is fixed for all ratio calculations where $1 \leq K \leq 784$.*

Analysis of our results

This cumulative explained variance ratio ultimately represents how much detail our new data retains from our original data. The reason the 'detail' of our data is measured by variance is because variance measures the average difference between all samples in the data, we want data with a high variance as this allows for more accurate classification.

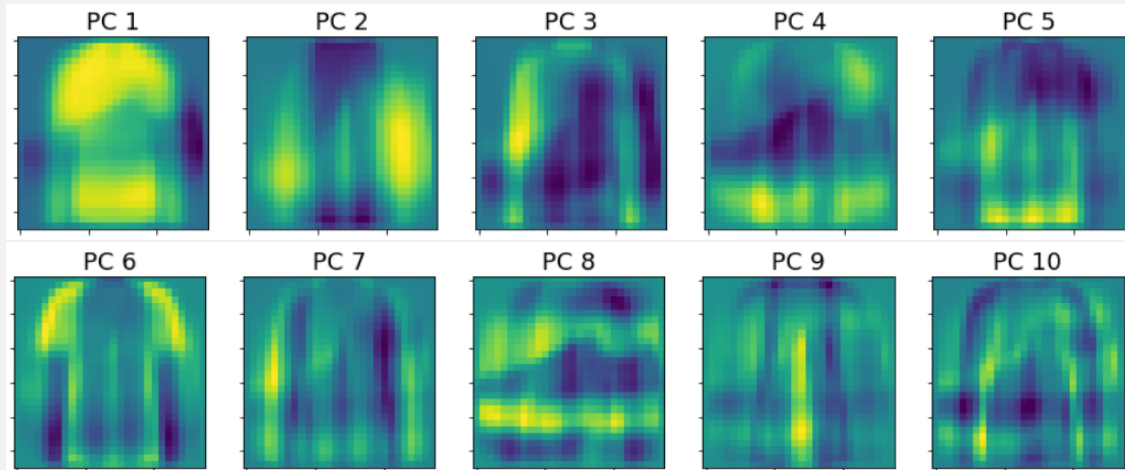
Given this information it is evident to see that this graphing model ultimately represents the relationship between the 'detail' in the data and the number of dimensions. This is very useful when you have high dimensional data that you want to reduce whilst still keeping as much 'detail' in the data as possible.

We can see from the figure above that using just 100 dimensions (about 87.24% less data) already secures about 91.17% of the total cumulative variance from the original data.

Choosing the optimal number of dimensions for a given dataset depends entirely on the context of the data/classification model, as this directly denotes how important the trade-off between the size of the data and the quality of the data is. For example, when working with critical data we can not afford to lose any quality thus we would prefer to not reduce the dimensions. However, reducing dimensions can be very useful for many supervised methods, making problems that are not linearly separable into ones that are, reducing noise in the data, and being able to visually represent high-dimensional 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



***These images represent the eigenvectors for each of their respective dimensions. Each of these eigenvectors has a direction and a corresponding eigenvalue. These are used for projecting data into new spaces.*

Analysis of our results

What is useful is that we notice that many of the different clothing shapes are retained in these eigenvectors. In most of these images you can see the resemblances of most of the different clothing types from our dataset, however, what is truly interesting is the frequency and intensity of the different clothing shapes found amongst these. These factors are very important as these suggest what are the most significant features throughout all of our different classes for classification.

With regards to the frequency of the different clothing shapes, I am referring to the number of eigenvectors in which we can visibly see the resemblance of the different clothing types. For example, in contrast to other clothing shapes, we can see in almost all of the images the resemblance of a long sleeved shirt. This ultimately suggests that this long sleeved shirt shape was most common from our training data. We know this is not due to class imbalance in our training dataset as we used an equal number of samples for each class, but rather due to the similarity between different classes. This is evident as illustrated by the class images in Q1.2 where classes 2 (pullover), 4 (coat) and 6 (shirt) all appear to take very similar shapes, resembling that of a long sleeved shirt.

With regards to the intensity of the different clothing shapes, I am referring to the most prominent/visible shapes for each eigenvector. For example, PC 1 shows the resemblance of a long sleeved shirt, PC 2 shows the resemblance of trousers, PC 3 shows the resemblance of a boot, and PC 4 shows the resemblance of a sneaker. This ultimately represents how the separate dimensions in our dataset may be used predominantly for classifying certain classes. This makes perfect sense as this allows our model to apply different weightings to each dimension.

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.

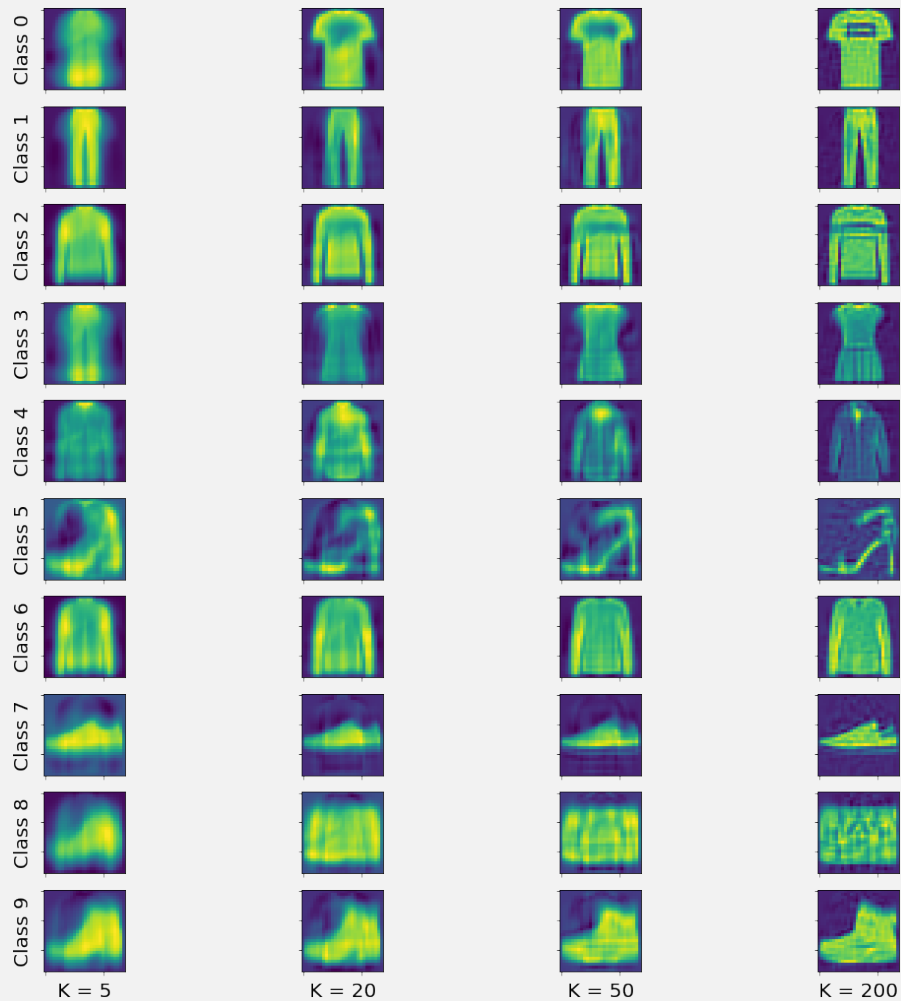
A table to show the RMSE between the original and the reconstructed version of the first sample for every class with varying numbers of PCA components (K)

*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)



Analysis of our results

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 the variance in the data, this is evident to see as the reconstructed samples with small K highlight the more prominent 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.



Analysis on the separations/congregations of class samples

As expected from our analysis in Q1.5 classes 2 (pullover), 4 (coat) and 6 (shirt) have very similar trends in their data comprising most of the data in the middle of this set of scatter points (almost following a $y=x$ diagonal). This is evidently due to their similarity in shapes, all resembling a long sleeved shirt. We can deduce that these points are in the middle of this scatter due to their frequency of shape (as discussed in Q1.5). Directly below this collection of classes we can see that class 0 (T-shirt) follows a similar trend, this is evidently due to it also being the shape of a shirt but with the slight difference of short sleeves.

We can also see on the left side of the scatter the congregation of the samples from classes 5 (sandals), 7 (sneaker), and 9 (ankle boots). This is evidently due to their similarity in shapes given that they are all different types of shoes. As expected, class 7 (sneaker) makes up the middle of this congregation of points, this is due to the fact that a sneaker is more similar to both a sandal and an ankle boot than they are to each other. At the very top of the scatter is class 8 (bag) which seems to merge into class 9 (ankle boots) quite alot. I found this quite surprising initially but realized this similarity can be attributed to the typical wideness we see in both boots and bags (as shown by the means in Q1.2).

The most consistent/unique of all classes is that of class 1 (trouser) at the very bottom of the scatter. This consistency is evident due to the very pure section of blue. We can deduce this consistency is due to the uniqueness in shape of a pair of trousers. This is evident when we notice the narrowness of a pair of trousers in comparison to other classes (as illustrated by the mean in Q1.2), and the gap between trouser legs which exclude the points in the center of the image which practically all other classes make use of.

Just above class 1 (trouser) and below class 0 (T-shirt) is class 3 (dress). This makes sense due to a dress' short sleeves (like a t-shirt) and narrowness (like a pair of trousers - again illustrated by the mean in Q1.2).

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:

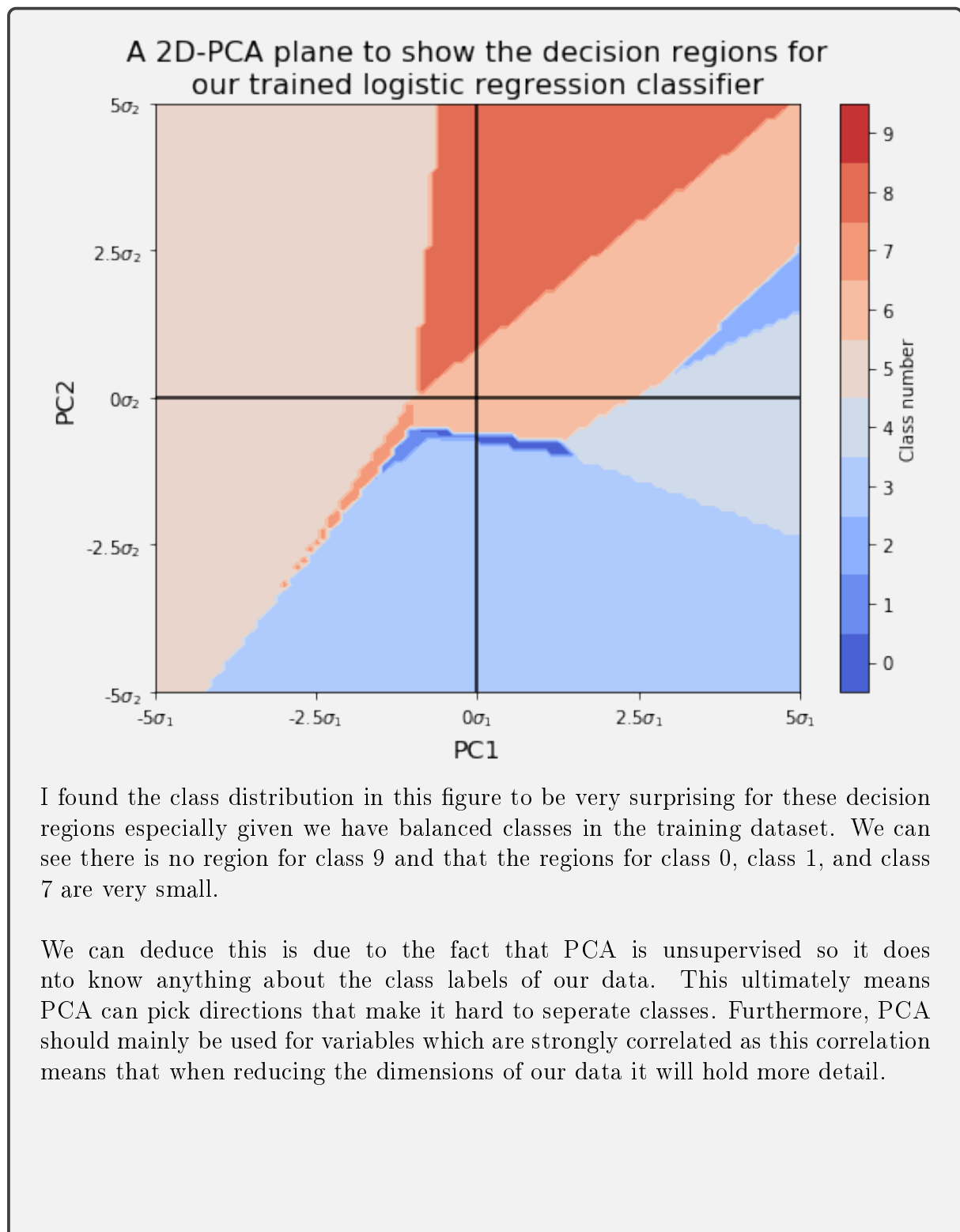
Predicted Actual	0	1	2	3	4	5	6	7	8	9
0	845	4	15	32	1	0	185	0	3	0
1	2	951	2	6	0	0	1	0	1	0
2	8	7	748	12	98	0	122	0	8	0
3	51	31	11	881	36	1	39	0	5	0
4	4	5	137	26	775	0	95	0	2	0
5	4	0	0	0	0	914	0	34	4	22
6	72	1	79	40	86	0	533	0	13	0
7	0	0	0	0	0	57	0	925	4	47
8	14	1	8	3	4	2	25	0	959	1
9	0	0	0	0	0	26	0	41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

PERCENTAGE CONFUSION MATRIX:

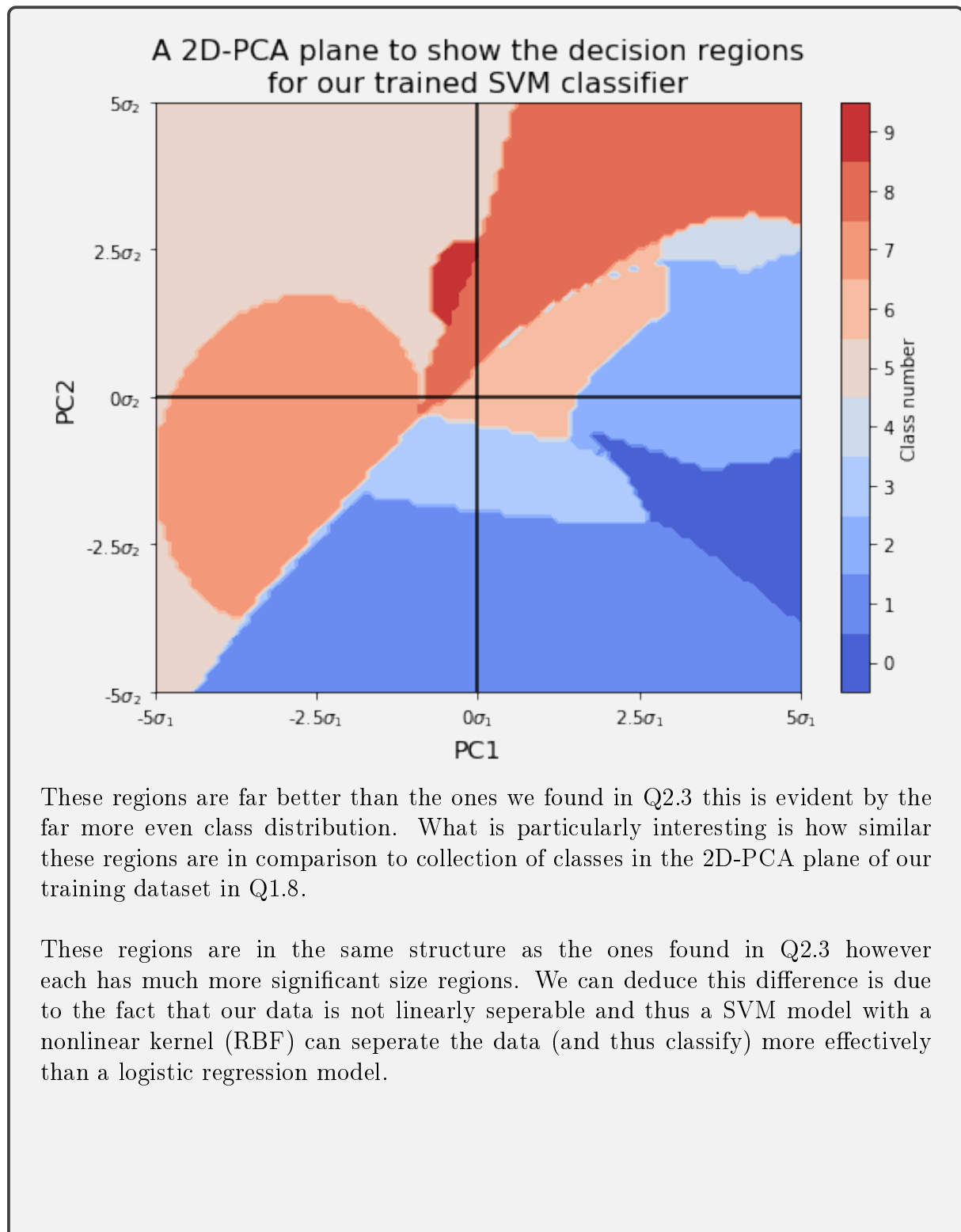
Predicted Actual	0	1	2	3	4	5	6	7	8	9
0	77.9	0.4	1.5	3.0	0.1	0.0	22.5	0.0	0.3	0.0
1	0.2	98.8	0.2	0.6	0.0	0.0	0.1	0.0	0.1	0.0
2	0.7	0.7	74.6	1.1	9.4	0.0	14.8	0.0	0.8	0.0
3	4.7	3.2	1.1	83.5	3.4	0.1	4.7	0.0	0.5	0.0
4	0.4	0.5	13.7	2.5	74.2	0.0	11.5	0.0	0.2	0.0
5	0.4	0.0	0.0	0.0	0.0	93.5	0.0	3.3	0.4	2.2
6	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%

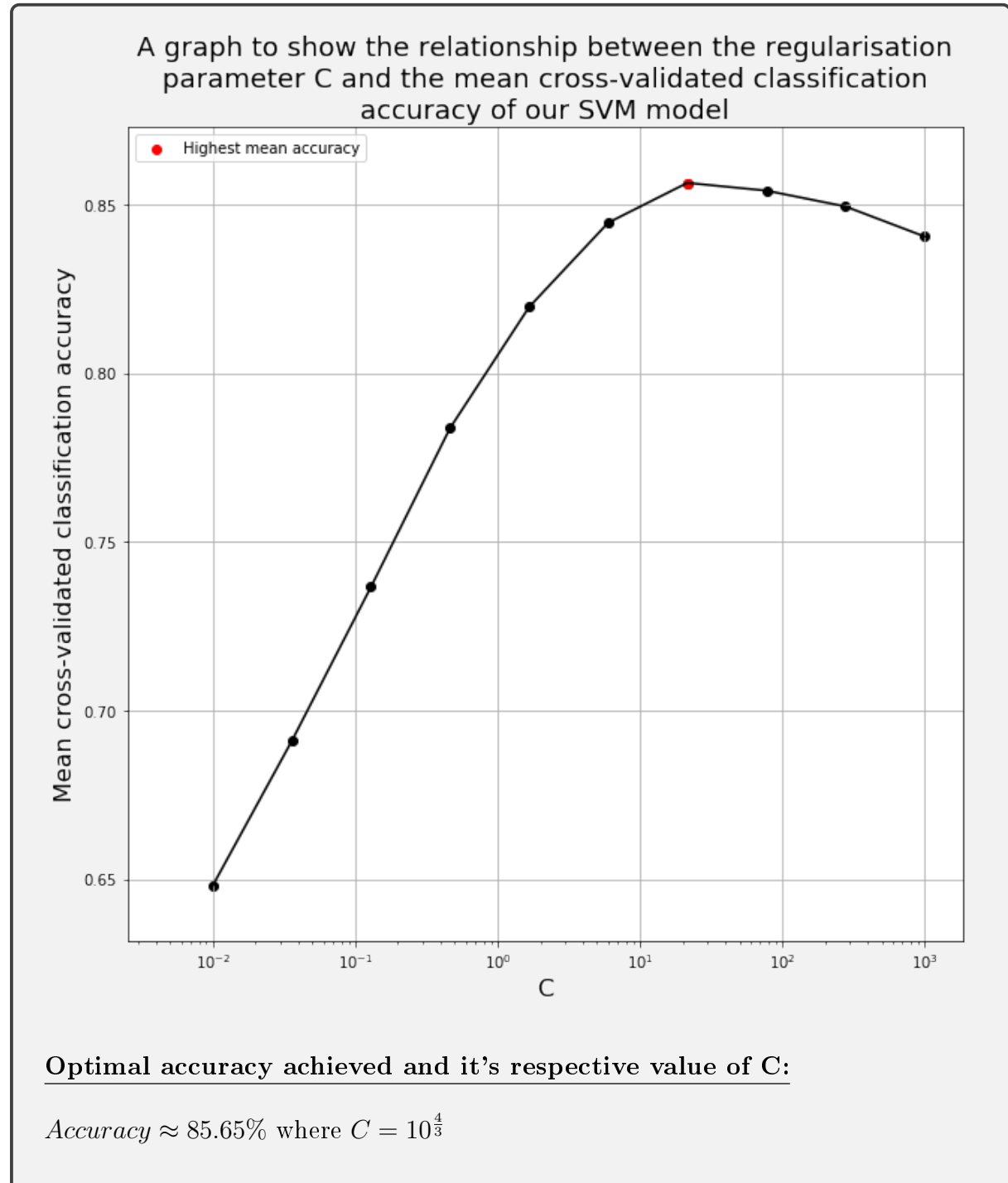
2.3 (6 points) We now want to visualise the decision regions for the logistic regression classifier we trained in Question 2.1.



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.



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 X_{small} , so that X_{small} contains 10,000 samples in total. Accordingly, you create labels, Y_{small} .



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.

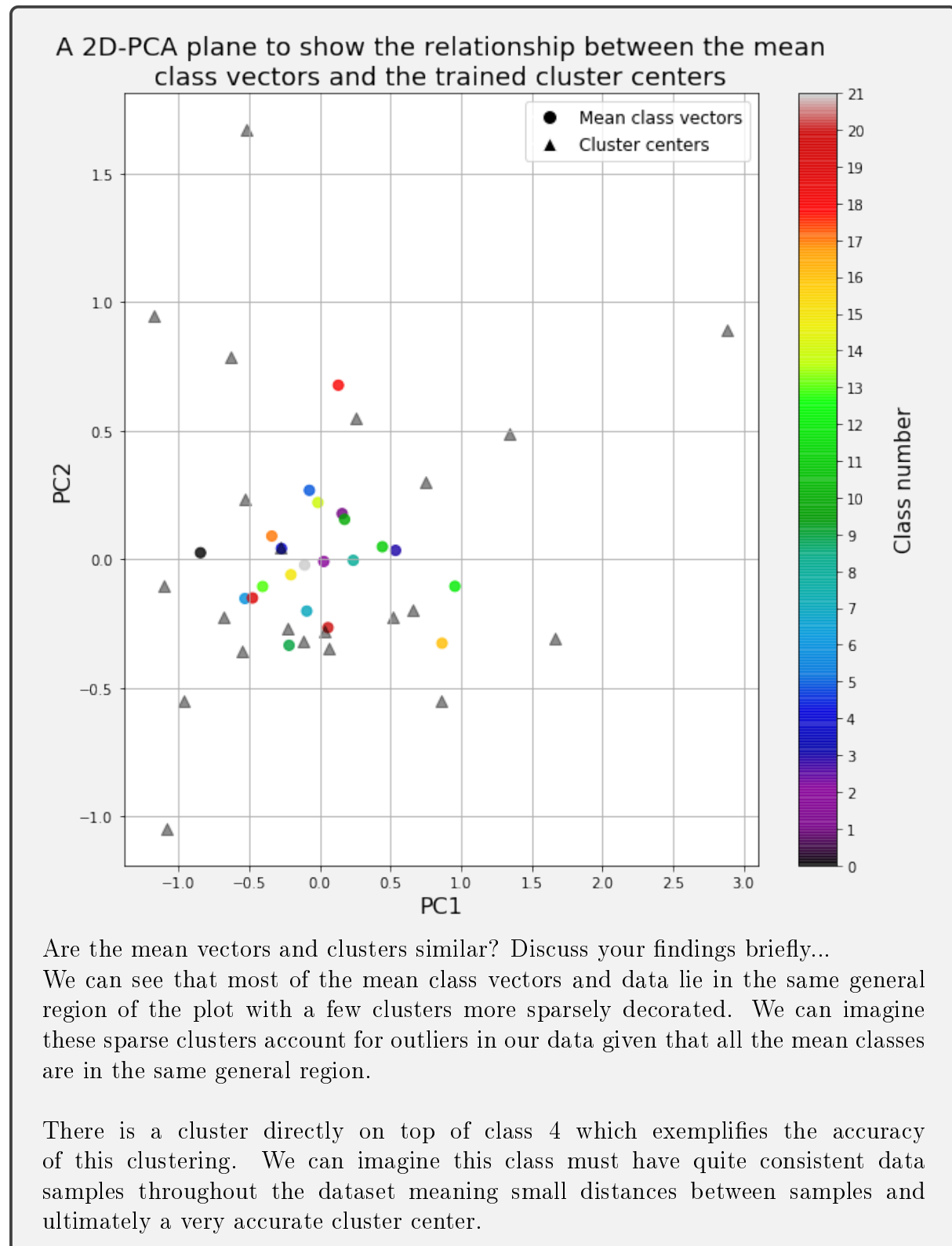
Metrics for our trained k-means clustering model

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

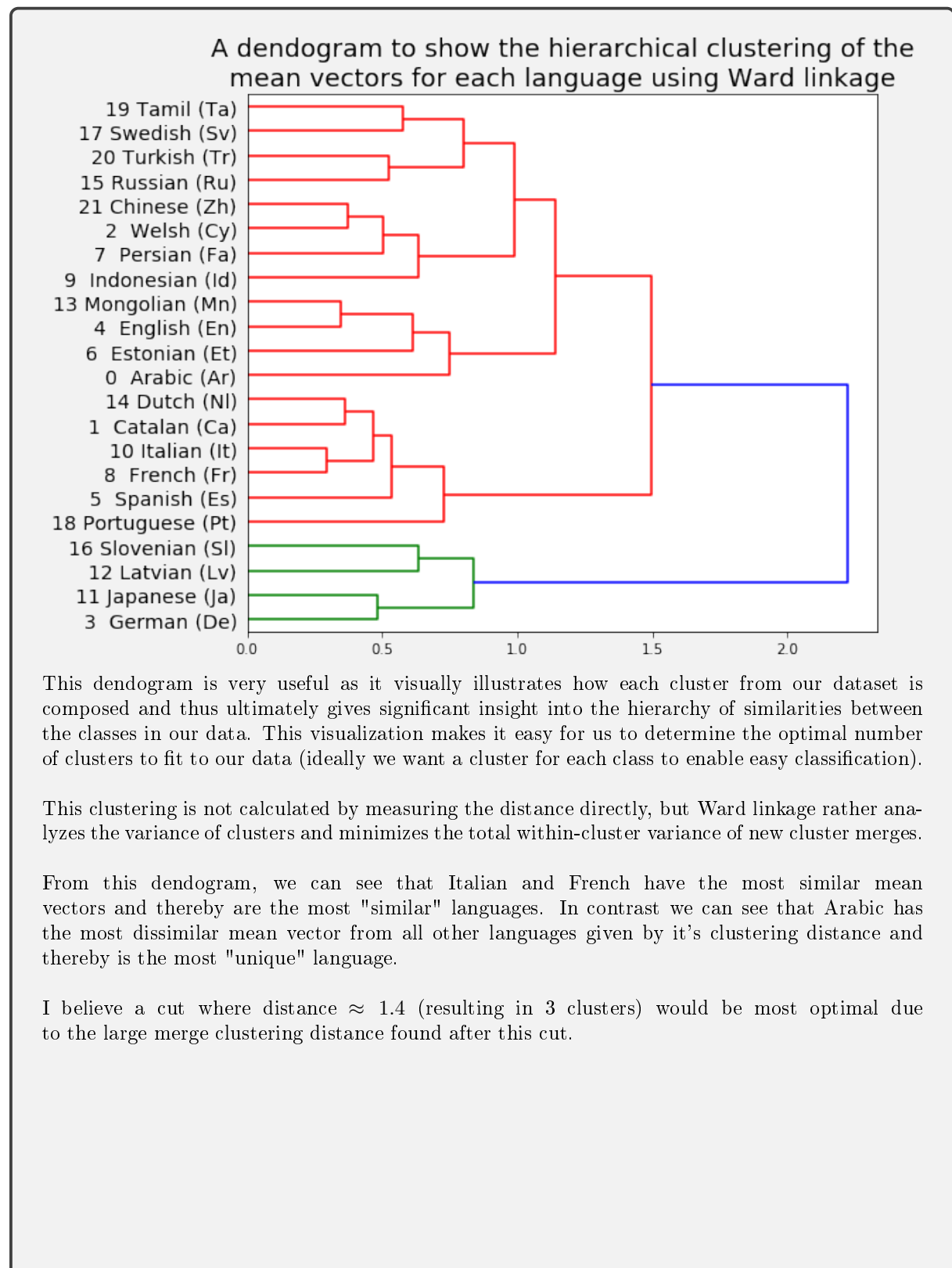
```
Number of samples for each cluster:
```

```
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
```

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.

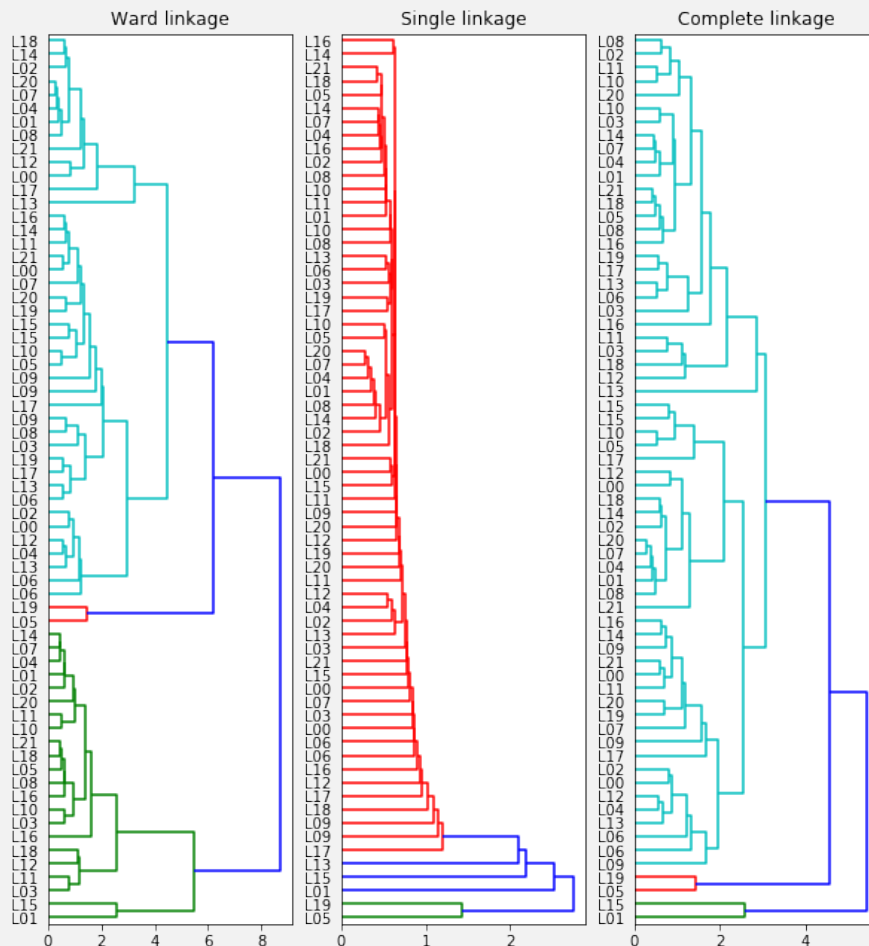


3.3 (3 points) We now apply hierarchical clustering on the training data set to see if there are any structures in the spoken languages.



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

Hierarchical clustering dendrograms with varying linkage trained over the three cluster centers for each of the languages in the dataset



***The labels on all of these dendrograms refer to the language that the given cluster center point came from.*

These dendrograms are very useful as they make it easier for us to compare the results of varying linkage methods for hierarchical clustering on our data. This ultimately allows us to choose the method that produces the optimal results based on the classes in our data.

We can see that in all of these dendrograms that there is a isolated cluster for language 19 and 5. This suggests that this cluster is the most dissimilar to all others among all different linkage criterion.

I believe single linkage is the least useful criterion in this context because of it's disproportionally large red cluster. The distances between all of the language cluster centers in this red section are on average very small (as illustrated by the scale on the x-axis) which make it far less useful as it does not separate the different classes in a meaningful way.

I believe Ward linkage is the most useful as it has the largest overall distance between language cluster centers this is evident by looking at the scale of the x-axis in comparison to single and complete linkage. This larger distance ultimately implies higher variance in the data based on this criterion and thus is far more useful for classification.

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.,

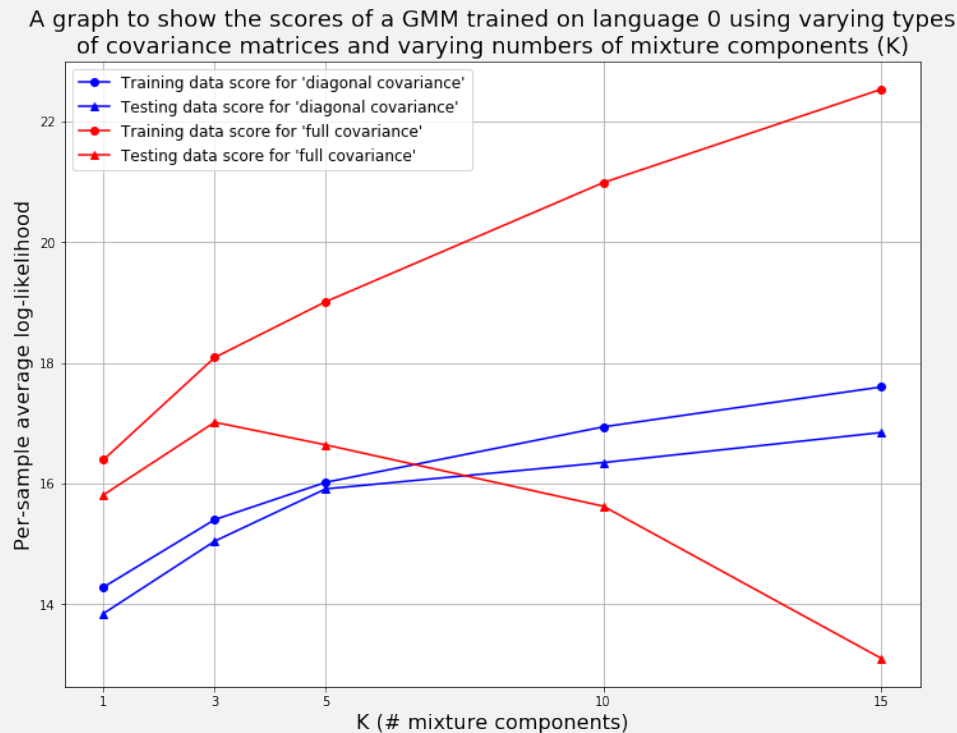


Table to show the average per-sample log-likelihood scores for varying parameters of a GMM

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

The log-likelihood of a mixture model ultimately represents the probability of observing our data as a function of the model's parameters. Given this we know that the higher the per-sample average log-likelihood the better the accuracy of the model.

The only difference between these 2 models is the type of covariance matrix used. A full covariance matrix assumes data dependence and a diagonal covariance matrix assumes data independence. Thereby this graph is perfect for identifying if there are any relationships between variables in the data and choosing the optimal model for a given value of K.

As shown by the figure above we can see that the GMM model that uses the full covariance matrix has the best training accuracy by far, however, we can see this accuracy drops dramatically from $K > 3$ for the training data. We can attribute this to the fact that our model was overfit to our training data. This overfitting makes perfect sense due to the assumption of variable dependence. This assumed variable dependence ultimately makes our model far more susceptible for capturing false relationships between variables in the training dataset.

In contrast, the GMM model that uses the diagonal covariance matrix has a worse overall training accuracy, however, unlike for the full covariance model the testing accuracy does not drop as we increase K but rather stays consistent with it's respective training accuracy.

In conclusion, we can deduce that for this dataset the number of mixture components (K) is inversely proportional to the dependence between variables in the dataset. Such that for $K \geq 8$ our GMM scores better when assuming data is dependent and visa-versa for when assuming data is independent.