IDS2018 - Assignment 5-6

Isabela Blucher

April 12, 2018

Exercise 2 (Using Linear Regression)

The Multivariate Linear Regression was implemented with the analytical formula for the weights $w = (X^T X)^{-1} X^T y$. The function receives the matrix X and vector y and returns the vector w with the weights. The weight vector interpretation is that w_i is the "expected change" in y for a one-unit change in x_i, which means that it shows the effects of a certain feature in our output vector.

For item b, we find the following weights (5.2057261, 0.05035934). We can learn from them that the fixed acidity of the wine is not significantly important to the wine quality score.

For item c, we find the weight vector (5.16573717e+01, 1.95852727e-02, -1.06193618e+00, 2.58896286e-02, 5.02281634e-02, -2.75489463e+00, 5.65346092e-03, -3.80728880e-03, -4.72092423e+01, -4.26639379e-01, 8.50478130e-01, 2.37895900e-01). The results show that different physiochemical properties have different effects on wine quality. For example, the density w_i is -4.72092423e+01, which means wines that are denser will have lower quality scores. Looking at the different weights it's clear that for a high quality score, a wine has to have a good balance between all its physiochemical properties.

Exercise 3 (Evaluating Linear Regression)

We build a regression model using our test set input variables and the weights computed in exercise 2 to find an estimated output vector. We then use the RMSE score to see if our predictions are a good fit on our real labels.

The RMSE score for exercise 3b is 0.786089275416 and the score for exercise 3c is 0.644717277283. The fact that our score for the full dataset is lower is a good thing, because it means that the regression has a better fit when all the features are being used for prediction and model building.

Exercise 4 (Random forest and normalization)

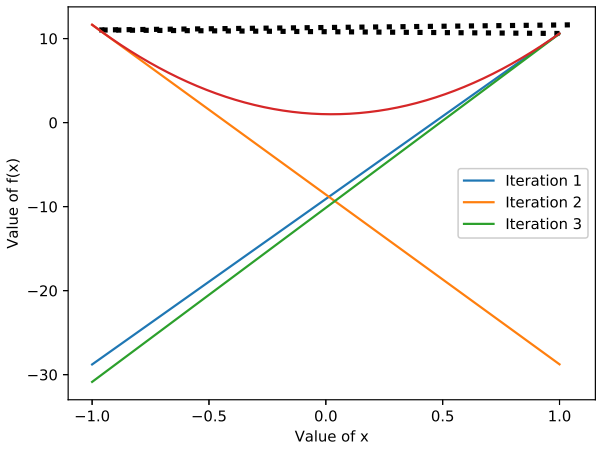
Normalization as a preprocessing for the data does not affect the random forest algorithm. The nature of RF is that convergence and numerical precision issues aren't as important as for other algorithms. Also RF is invariant to feature scaling since the features are never compared in magnitude to one another. In conclusion, we don't need to normalize the dataset before a random forest classifier.

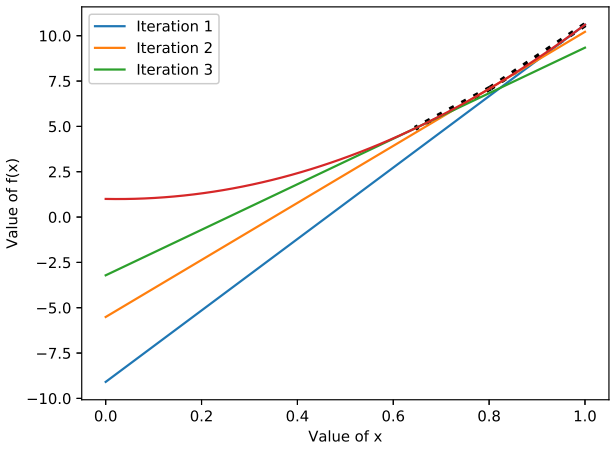
Exercise 5 (Applying random forest)

We initialize a random forest classifier with 50 trees, fit it with the training data `IDSWeedCropTrain.csv` and then test it with `IDSWeedCropTest.csv`. The accuracy score we obtain here is 0.966898954704. The accuracy score obtained for the nearest-neighbor classifier in Assignment 3 was 0.945993031359. While both scores are good, the random forest classifier performed better.

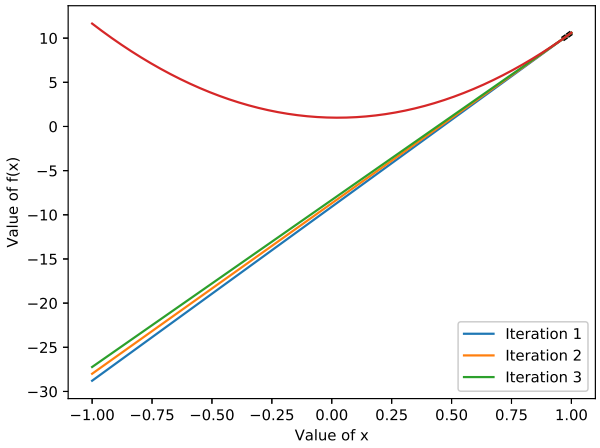
Exercise 6 (Gradient descent and learning rates)

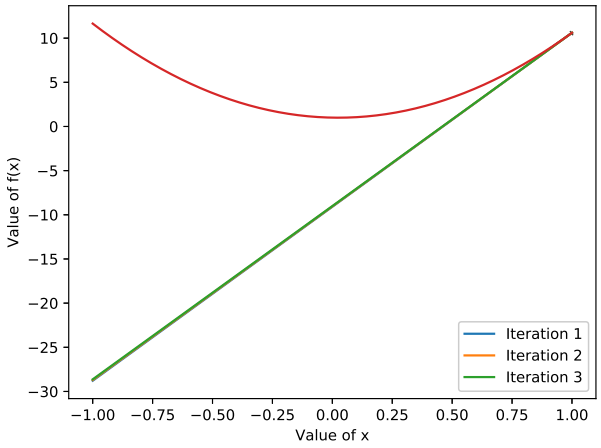
Gradient descent was implemented with a gradient tolerance of 10^{-10} and a maximum iteration tolerance of 10^4. The Figures below show the algorithm steps and the tangent lines for the first 3 iterations.





Plots of $\eta = 0.1$ and 0.01

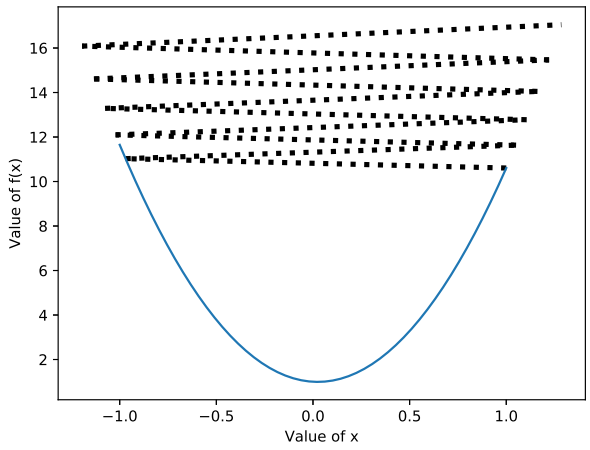


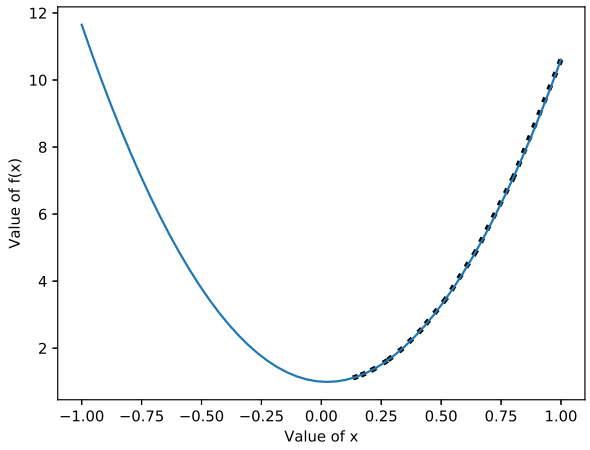


Plots of $\eta = 0.001$ and 0.0001

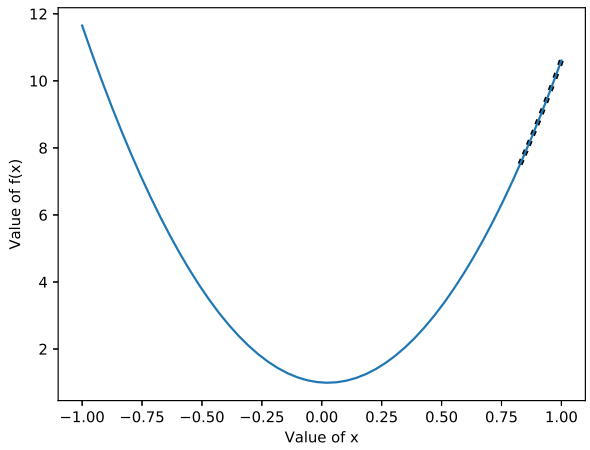
As we can see, for the largest value of the learning rate the step size is too big, but for too small a learning rate the steps are not very significant.

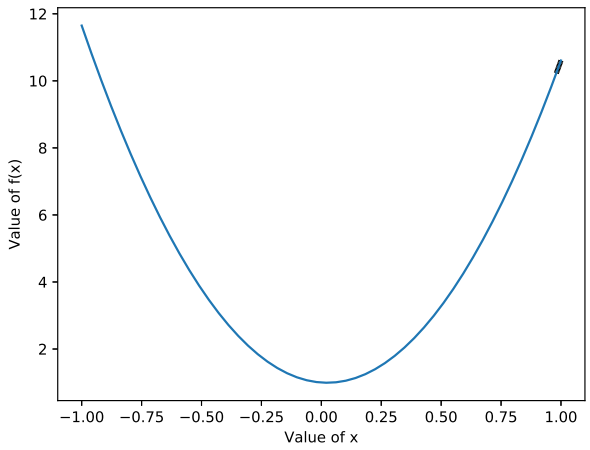
Now we plot the gradient descent steps for the first 10 iterations, for all 4 values of the learning rate.





Plots of $\eta = 0.1$ and 0.01





Plots of $\eta = 0.001$ and 0.0001

With this new set of plots we can conclude that the optimal learning rate for gradient descent for our given function value and starting point is $\eta = 0.01$, as it does converge and takes steps of significant length.

The following function values at the final iteration and number of iterations until convergence were found for all the learning rates.

	$\eta = 0.1$	$\eta = 0.01$	$\eta = 0.001$	$\eta = 0.0001$
Function value	does not converge	0.993826847811	0.993826847811	0.993826847811
Number of iterations	stops at 70	116	1273	10000

Table 1: Function values and iterations until convergence for all learning rates of gradient descent

Exercise 7 (Logistic Regression implementation)

For the two given datasets we have Figures 1 and 2, which are scatter plots with the datapoints colored according to their respective label. The training and test sets were appended to make one big dataset.

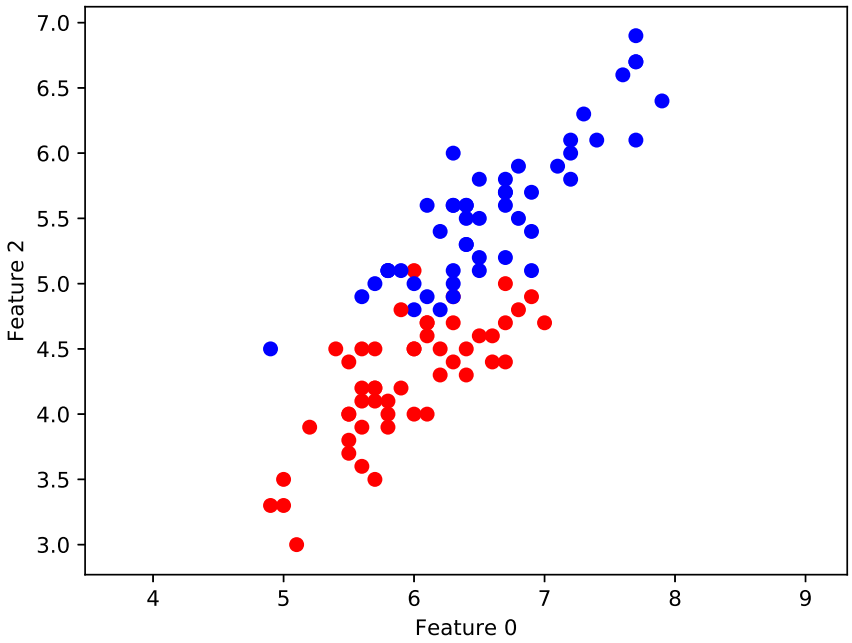


Figure 1: Scatter plot of dataset Iris2D1, with points colored according to labels

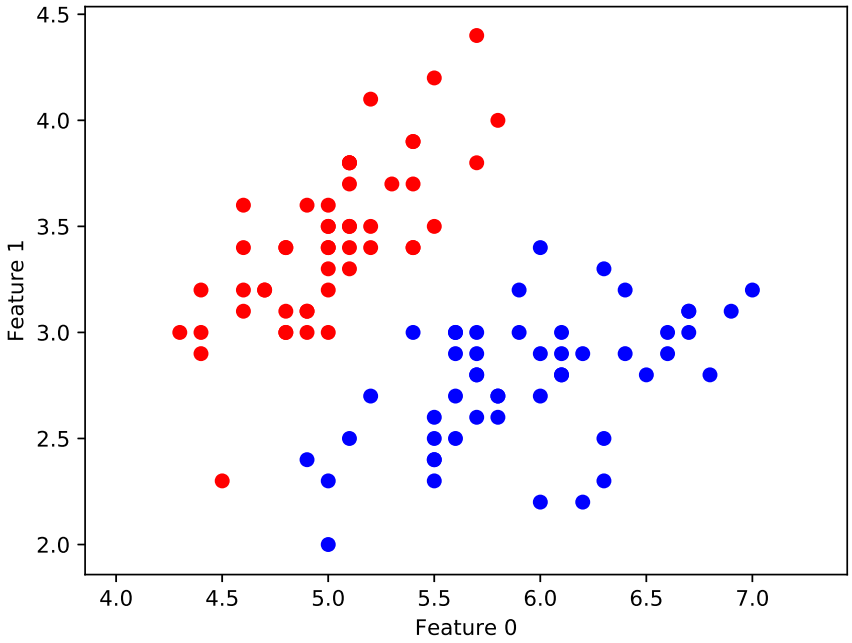


Figure 2: Scatter plot of dataset Iris2D2, with points colored according to labels

Figures 1 and 2 show that the values of each feature define very well which label the datapoint gets since there is no significant overlap between colors in the plots.

For the version of the Logistic Regression algorithm that is being used here the labels need to be 1 and -1, so we transform all the 0 labels to -1. The parameters being used are 10^{-4} for the norm of the gradient tolerance, 0.3 for the learning rate and the initial $w_0 = (0, 0, 0)$.

For the Iris2D1 dataset, we first find the final w and then test it on the training and test datasets. The training error measured by the zero-one loss for Iris2D1 is 0.0428571428571 and the training error for Iris2D2 is 0.0. The test error, also measured by the zero-one loss, for Iris2D1 is 0.1 and the test error for Iris2D2 is 0.0.

Also the parameters for the affine linear model for Iris2D1 are (-28.27830299, -4.18419265, 11.0818053) and for Iris2D2 they are (-40.65871962, 16.54775203, -15.56797156).

Exercise 9 (Clustering and classification I)

For this exercise we use the KMeans clustering algorithm for $k = 3$, so we create 3 different clusters. We feed our KMeans clustering algorithm with 3 random starting points from our MNIST dataset and then we get the cluster labels for all datapoints and the final cluster centers. With that we are able to compute the percentage of 1s, 7s and 9s in each cluster, as well as plot the cluster centers as images.

Figures 3, 4 and 5 show us the centers of clusters 0, 1 and 2.

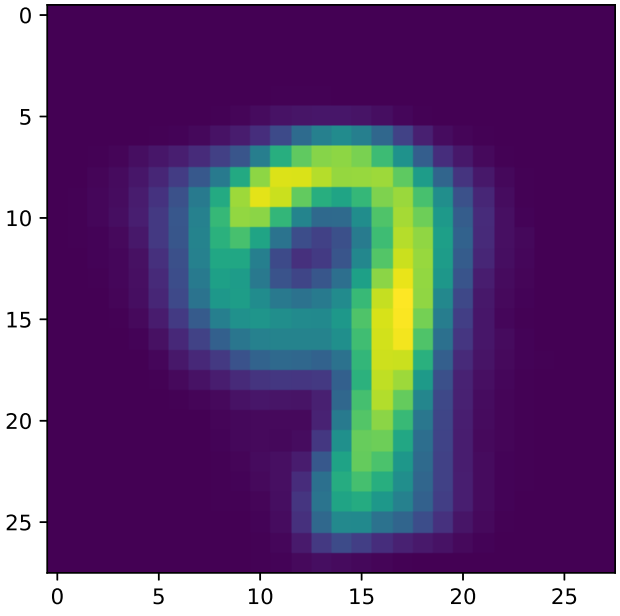


Figure 3: Center of cluster 0

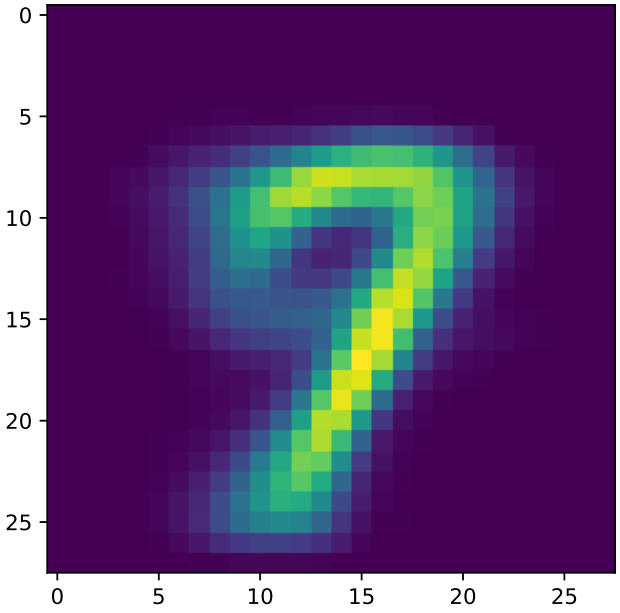


Figure 4: Center of cluster 1

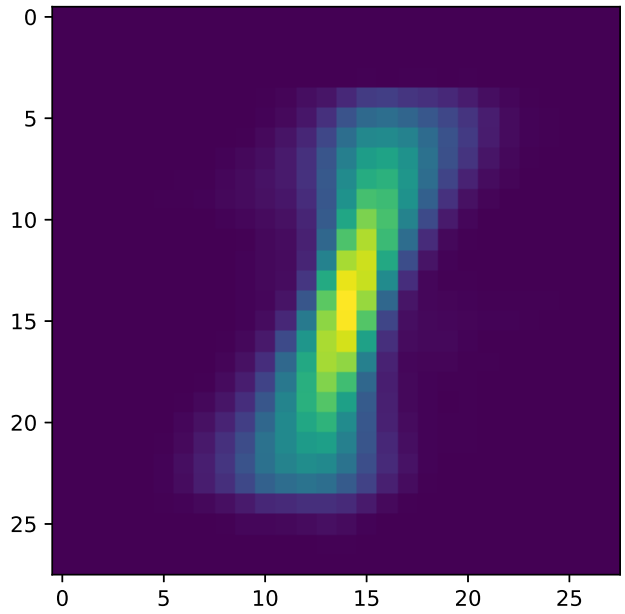


Figure 5: Center of cluster 2

From visual inspection we can observe that cluster 0 seems to have more 9s, cluster 1 more 7s and cluster 2 more 1s. To confirm this observation Table 2 has the percentages of each label per cluster, where we can see a clear majority of labels for each cluster.

	Percentage of 1s	Percentage of 7s	Percentage of 9s
Cluster 0	0.89%	33.13%	65.98%
Cluster 1	0.28%	62.22%	37.5%
Cluster 2	89.83%	7.74%	2.43%

Table 2: Percentages of labels per cluster

For the second part of exercise 9 we need to train a k-nn classifier on the dataset using n-fold validation to find an optimal k. We separate our original dataset so that the first 900 datapoints are our training set and the rest are the test set. In this case we use $n = 5$ and test for values of $k = \{1, 3, 5, 7, 9, 11\}$ as we did for the past assignments. To find out the optimal value, we measure the average zero-one loss for each k that we are testing. The smaller average zero-one loss was found for $k = 1$.

We then compute the accuracy for our 1-nn classifier, which is 0.964444444444.

Exercise 10 (Clustering and classification after dimensionality reduction)

We compute the PCA of the training set for the MNIST dataset (for our case, the first 900 datapoints). With that we obtain the eigenvalues (variances) and the eigenvectors (PCs). We then compute and plot the cumulative variance with relation to the principal components, which is shown in Figure 6.

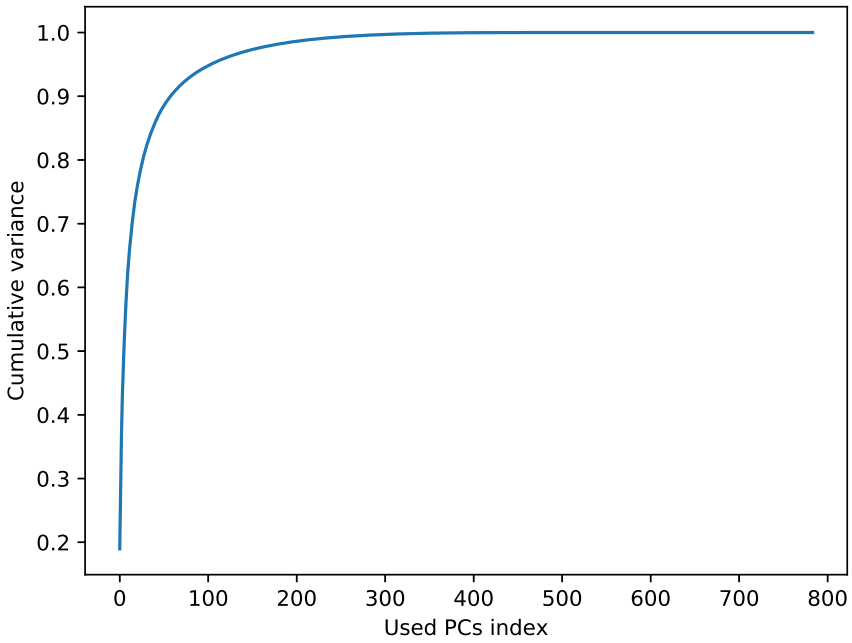


Figure 6: Cumulative variance with relation to principal components

After that we project the training data into the first 20 and 200 principal components.

(b) 20 PCs

We run the KMeans clustering algorithm to create 3 clusters on the data projected onto the first 20 principal components. We obtain the following images and percentages from running the same processes here that we did for Exercise 9.

The difference here is that we revert the clusters from a 20-dimensional space to a full 784-dimensional space. We do that by the following formula $center_{784} = center_{20} * eig_{20}^{T} + mean$, where $center_{20}$ were the 20-dimensional cluster centers, eig_{20} are the first 20 eigenvectors and $mean$ is the mean of the dataset. The same technique will be used on the cluster centers for the 200-dimensional case.

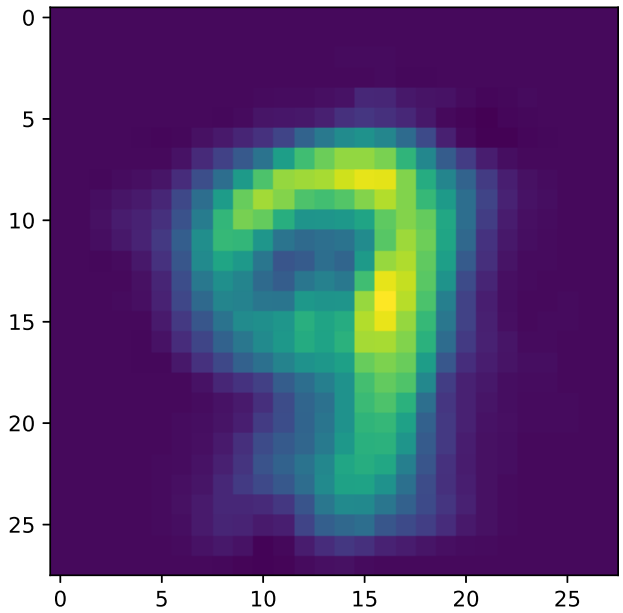


Figure 7: Center of cluster 0

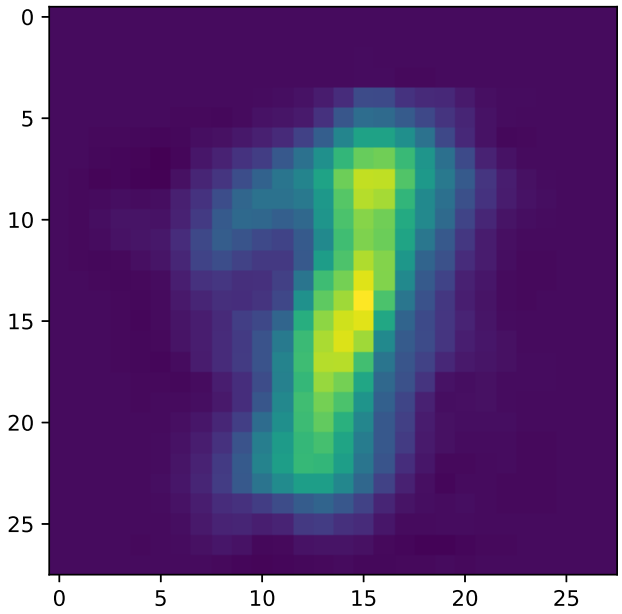


Figure 8: Center of cluster 1

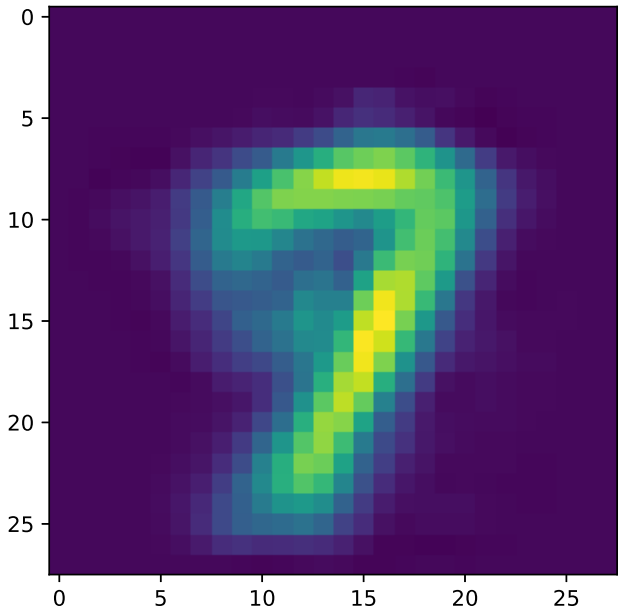


Figure 9: Center of cluster 2

	Percentage of 1s	Percentage of 7s	Percentage of 9s
Cluster 0	0.74%	32.84%	66.42%
Cluster 1	87.61%	9.43%	2.96%
Cluster 2	0.35%	61.72%	37.93%

Table 3: Percentages of labels per cluster

(b) 200 PCs

Running KMeans again with 3 clusters but this time on the data projected onto the first 200 principal components and then, with the cluster centers reverted back to the full dimensional space we obtain the following images and percentages.

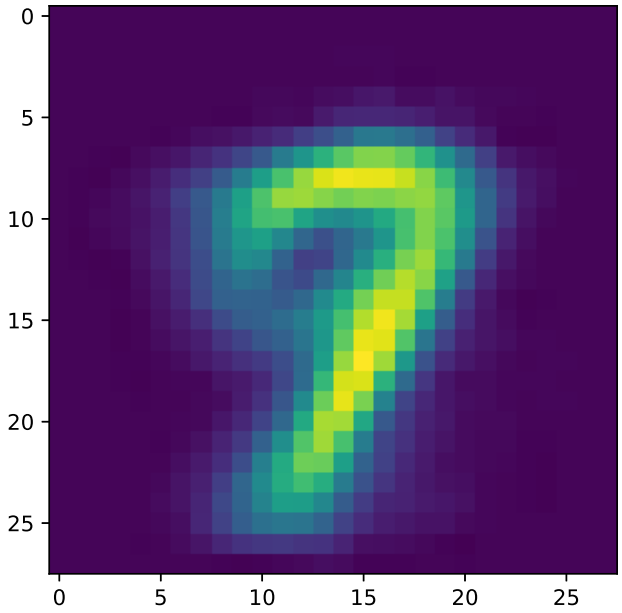


Figure 10: Center of cluster 0

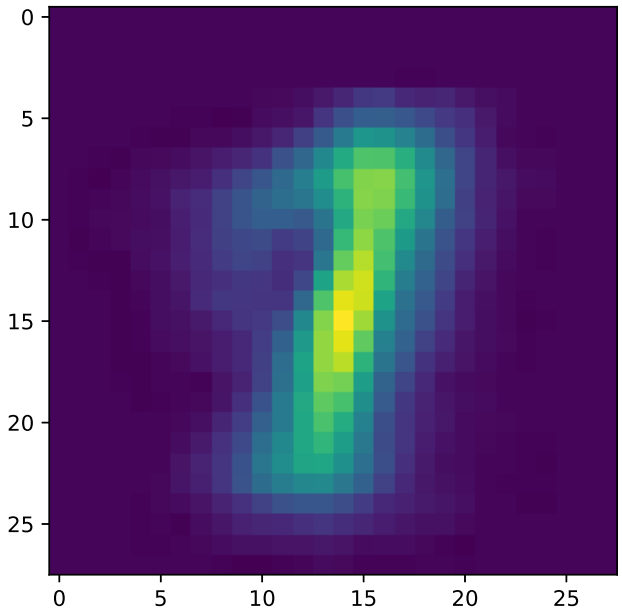


Figure 11: Center of cluster 1

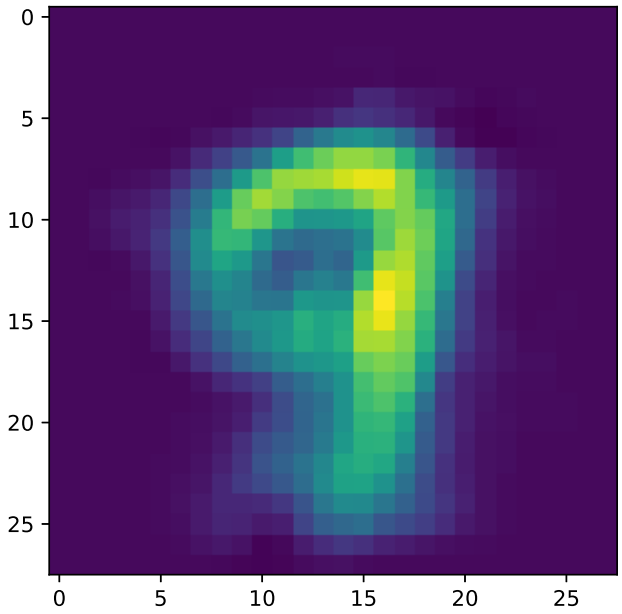


Figure 12: Center of cluster 2

	Percentage of 1s	Percentage of 7s	Percentage of 9s
Cluster 0	0.36%	61.18%	38.46%
Cluster 1	87.61%	9.43%	2.96%
Cluster 2	0.74%	33.81%	65.45%

Table 4: Percentages of labels per cluster

In comparison to Exercise 9, it seems that the images become less clear and precise, though the statistics seem to stay the same. We can see that for the first 200 PCs it is clearer than for the first 20 PCs because, as we can observe in Figure 6, the first hundred PCs capture the majority of the variance in the dataset.

(c) 20 PCs

Doing the same method here as we did in Exercise 9, we use 5-fold cross validation on the dataset projected onto the first 20 PCs. For both datasets we separated it again with an 80 to 20 ratio, so that our training set has 720 datapoints and the test set has 180.

We measure the average zero-one loss for values of $k = \{1, 3, 5, 7, 9, 11\}$. The smaller loss was for $k = 3$ in this case, though the loss value was very similar to $k = 1$. The test accuracy in this case is 0.977777777778.

(c) 200 PCs

Measuring the zero-one loss with 5-fold cross validation for the same values of k, we get that the optimal is $k = 1$ and the test accuracy is 0.983333333333.

From the results in Exercise 9 and the test accuracies measured here we see that the accuracy here is better. This may be due to less elements being used in the prediction, and thus less chances of there being outlying values that could make the prediction accuracy worse.