IDS Assigment 5 and 6 - Stefano Pellegrini

Bayesian Statistics

Exercise 1

$$y = ax + b + \varepsilon$$
$$\varepsilon \sim \mathcal{N}\left(0, \sigma^2\right)$$

a) Write explicitly the probability density function of ε

$$f(\varepsilon) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\varepsilon}{\sigma}\right)^2}$$

b) Write the conditional probability distribution of y knowing a, x and b

$$\begin{aligned} P(y|(x,a,b),\sigma^2) &= \mathcal{N}(y|ax+b,\sigma^2) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(ax+b-y)^2}{\sigma^2}} \end{aligned}$$

c) What are the parameters of the linear model?

$$Parameters = a, b, \sigma^2$$

d)

- **How many priors are needed, and over which parameters.**

  We need a prior for each parameter, which is our prior belief about the true value of the parameter. So in our linear model we need three priors. In case we want to use a non-informative prior (ignorance prior) we can use a Jeffreys prior.

- **Can you use Gaussian priors for all parameters?**

  We can use Gaussian priors for all parameters except for the variance (σ^2), because σ^2 must have a a non-negative value.

Linear regression

Exercise 2 (Using Linear Regression)

b)

- **Parameters of the model with only fixed weights.**

  $\boldsymbol{w_i}$ = (5.626, 0.08766808)

- **What can you learn from them?**

  Both coefficients have positive values. W0 (the intercept) is the expected mean value of the response variable Y when all X equal zero. W1 (fixed acidity coefficient) is quite small and I can't determine whether it is significatively different from zero without a statistical test. Anyway, since it is positive, if the difference from zero would be significant, it would have meant that there is a (weak) positive linear relation between fixed acidity and the quality of the wine. And so that if the predictor variable (fixed acidity) increases, the response variable also increases.

c)

- **Parameters of the the full model.**

  $\boldsymbol{w_i}$ = (5.62600000e+00, 3.40950277e-02, -1.91855307e-01, 5.06796985e-03, 6.96665078e-02, -1.34723032e-01, 5.83177511e-02, -1.28121788e-01, -8.94314976e-02, -6.78094266e-02, 1.50312454e-01, 2.49543059e-01)

- **What do they tell you about how the different physiochemical properties relate to the wine quality?**

  Sorted coefficients:
  5.6260000 = intercept
  0.2495431 = alcohol
  -0.1918553 = volatile acidity
  0.1503125 = sulfates
  -0.1347230 = chlorides
  -0.1281218 = total sulfur dioxide
  -0.0894315 = density
  0.0696665 = residual sugar
  -0.0678094 = pH
  0.0583178 = free sulfur dioxide
  0.0340950 = fixed acidity
  0.0050680 = citric acid

  The weights can tell us which are the most relevant physiochemical properties to determine the quality of the wine and if the relation between the quality and these property is positive or negative. In our case, it is possible to observe that alcohol is the most important feature. Since it has a positive coefficients, it has a positive linear relation with the quality of the wine. Also, others important features are volatile acidity, sulfates, chlorides and total sulfur dioxide. The input data has been standardized prior to linear regression, because the features are not on the same scale.

Exercise 3 (Evaluating Linear Regression)

b)

- **RMSE model for 1-dimensional input variables (fixed acidity).**

  $RMSE(\mathbf{w}) = 0.7860892754162238$

c)

- **RMSE full model.**

  $RMSE(\mathbf{w}) = 0.6447172773067069$

- **How does it compare to your answer from b?**

  The full model perform quite better than the model with only the fixed acidity predictor. This was expected since the fixed acidity did not result to be a really important predictor.

Random forest

Exercise 4 (Random forest & normalization)

- **Using this normalization affects different classification methods differently. How does it affect a random forest classifier?**

Random forest consists of a large number of individual decision trees that operate as an ensemble. A decision tree is an upside down tree-like structure, where each node corresponds to a condition that splits the data based on a threshold, and the value of a specific feature. For this reason random forest is invariant to monotonic transformation and so normalizing the data will have no effect.

Eexrcise 5 (Applying random forrest)

- **prediction accuracy of the random forest.**

  Random forest accuracy $\approx 96.69\%$
  K-nearest neighbor accuracy $= 95.47\%$ (not requested in deliverables)

  Random forest beat the neirest neighbour classifier.

Gradient descent

Exercise 6 (Gradient descent & learning rates)

6.2 Visualize the tangent lines and gradient descent steps for the first three iterations.

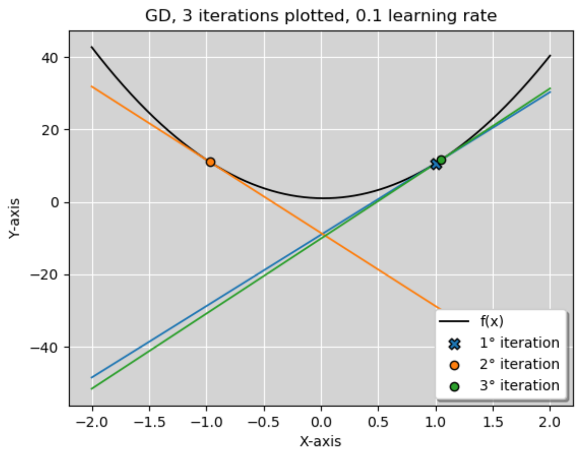


Figure 1: $\boldsymbol{\eta = 0.1}$

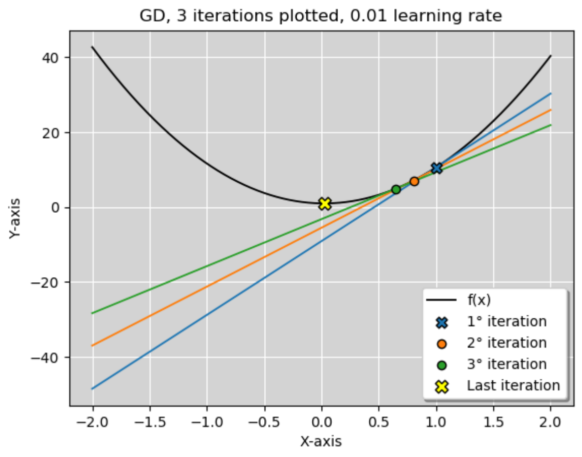


Figure 2: $\boldsymbol{\eta = 0.01}$

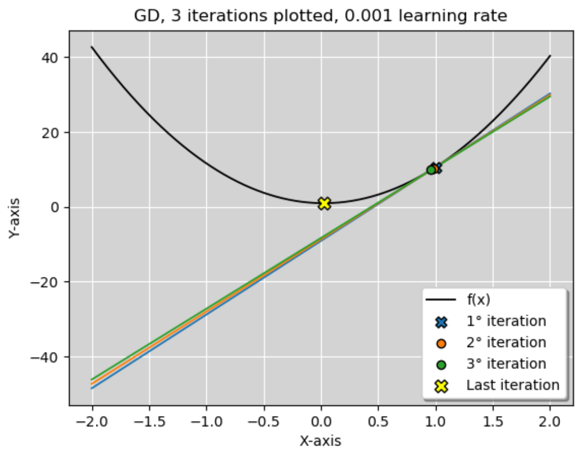


Figure 3: $\boldsymbol{\eta = 0.001}$

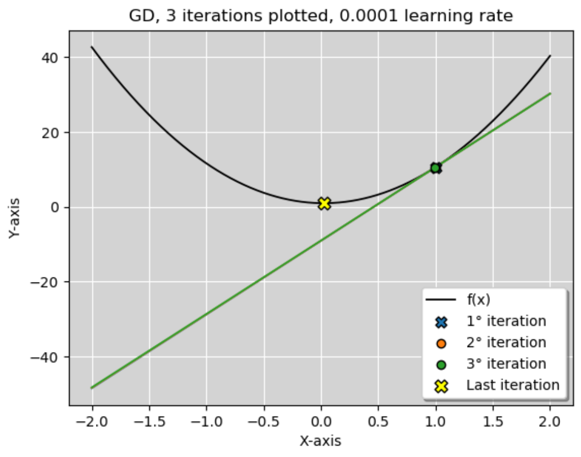


Figure 4: $\boldsymbol{\eta = 0.0001}$

6.3 Visualize gradient descent steps for the first 10 iterations.

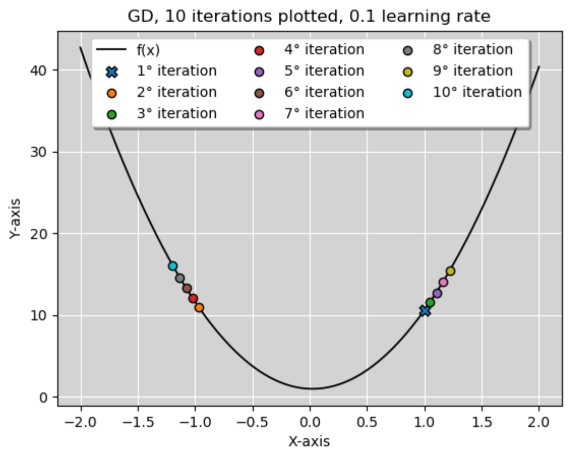


Figure 5: $\boldsymbol{\eta = 0.1}$

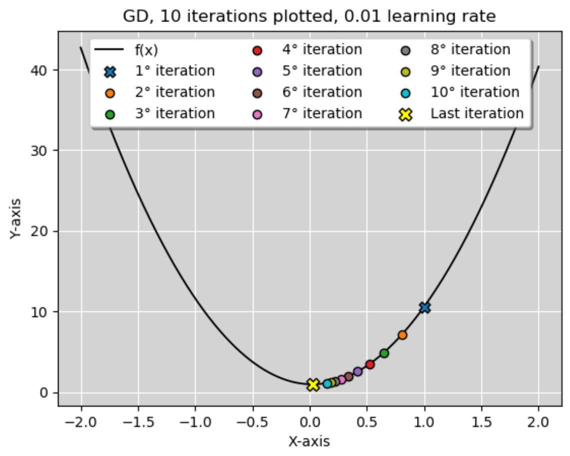


Figure 6: $\boldsymbol{\eta = 0.01}$

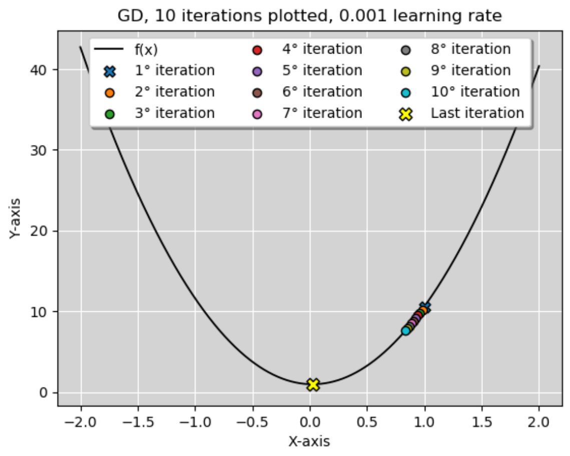


Figure 7: $\boldsymbol{\eta = 0.001}$

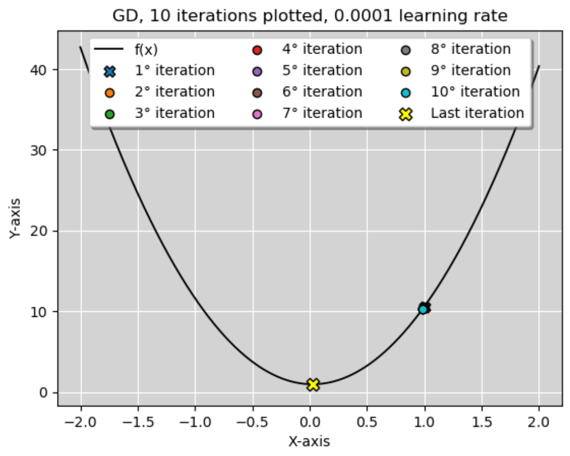


Figure 8: $\boldsymbol{\eta = 0.0001}$

6.4 Run the algorithm until the magnitude of the gradient falls below the threshold or the algorithm has exceeded 10,000 iterations.

- **Rate 0.1:**
  Function value = NA
  Number of iterations = 10000
- **Rate 0.01:**
  Function value = 0.993826847811084
  Number of iterations = 116
- **Rate 0.001:**
  Function value = 0.9938268478110839
  N. iterations = 1273
- **Rate 0.0001:**
  Function value = 0.993826847811084
  Number of iterations = 10000

Logistic Regression

$$f(x) = e^{-x/2} + 10x^2$$

Exercise 7 (Logistic regression implementation)

7.1 Make a scatter plot of each dataset, with point color depending on classes.

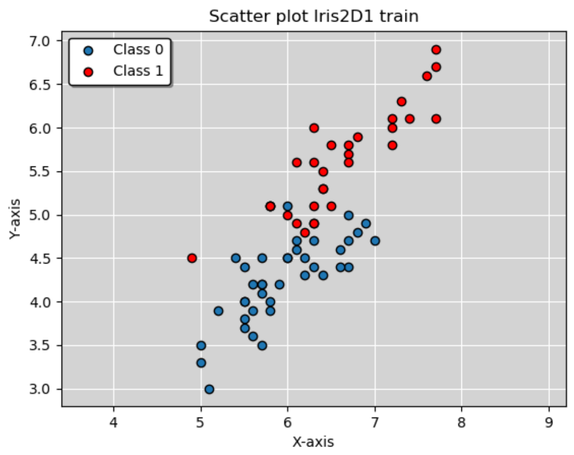


Figure 9: **Iris2d1 train**

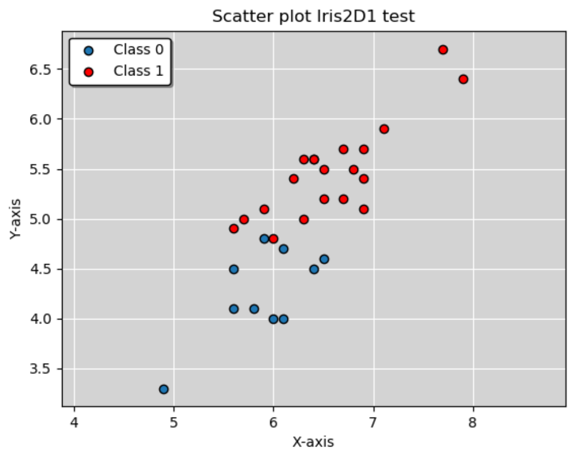


Figure 10: **Iris2d1 test**

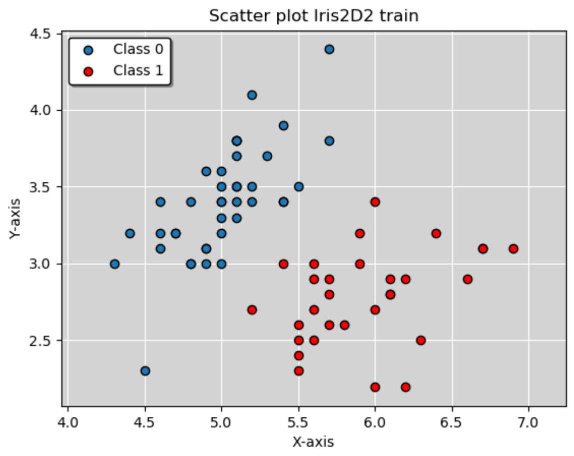


Figure 11: **Iris2d2 train**

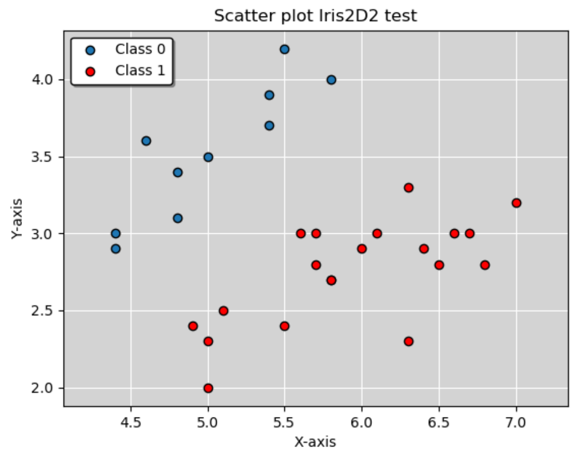


Figure 12: **Iris2d2 test**

- **What do you observe?**

  It is possible to observe that, on averge, the length and the width of sepals and petals of the class 1 points is greater than those of class 0 points. Also, in the Iris2D2 train and test data there is a clear linear boundary division between the two classes points and the boundary seems to be represented by a straight line. In Iris2D1 train and test data the boundary division seems to be less clear, in this case it doesn't seem to be represented by a straight line.

7.3 Apply logistic regression to each dataset.

- **Iris2D1 train data:**
  Parameters of the linear model = (-18.88806165, -4.20325408, 9.1606775)

Training error = 0.04285714285714286
Test error = 0.1

- **Iris2D2 train data:**
Reached convergence at 20000 iterations
Parameters of the linear model = (-47.39211726, 18.86166421, -17.37941881)
Training error = 0.0
Test error = 0.0

- **Plot the decision boundary (not requested):**

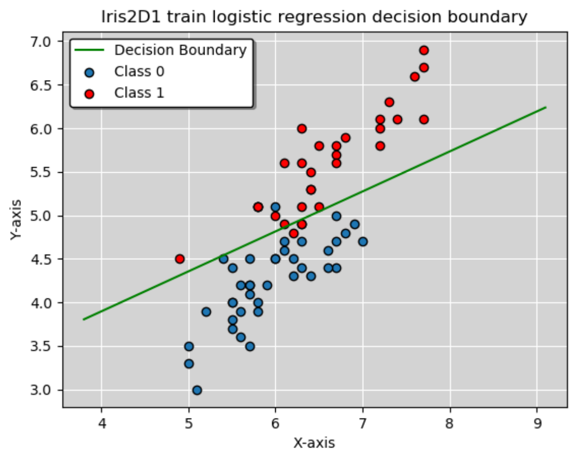


Figure 13: **Iris2d1 train**

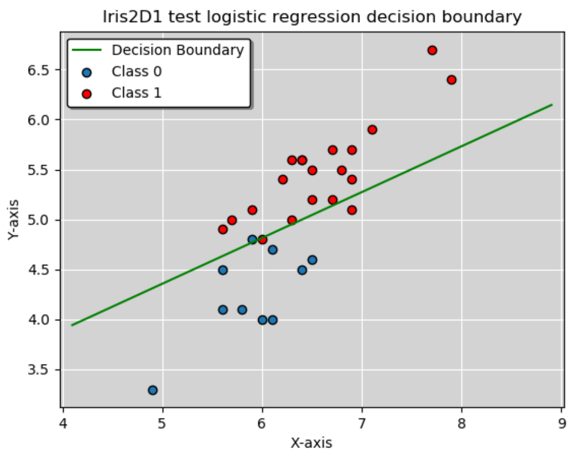


Figure 14: **Iris2d1 test**

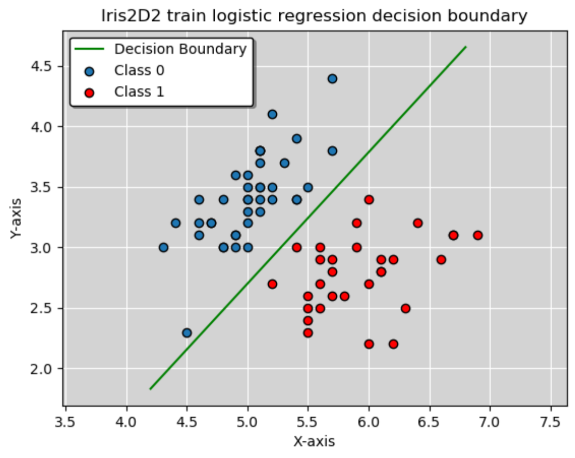


Figure 15: **Iris2d2 train**

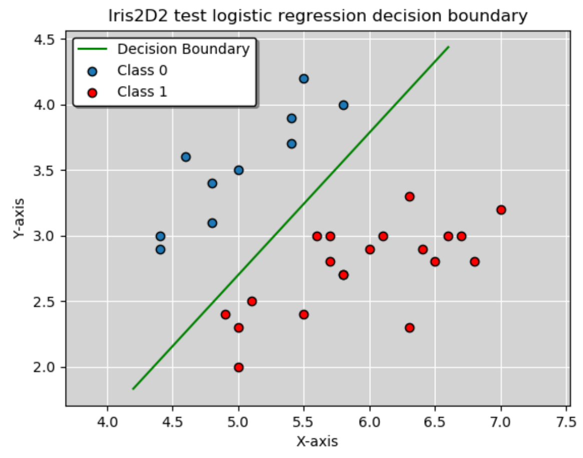


Figure 16: **Iris2d2 test**

Exercise 8 (Logistic regression loss-gradient)

8.1 Show that the gradient of the in-sample function Ein(w), as defined in the lecture, is given by

$$\nabla E_{in}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n\mathbf{x}_n}{1+e^{y_n\mathbf{w}^\top\mathbf{x}_n}} = \frac{1}{N}\sum_{n=1}^{N}-y_n\mathbf{x}_n\theta\left(-y_n\mathbf{w}^\top\mathbf{x}_n\right) \tag{1}$$

- Assuming independent data points $(x1, y1), ..., (xN, yN)$, in logistic regression the probability of observing all these y_n, given the inputs x_n is given by

$$\prod_{n=1}^{N} P\left(y_n|\boldsymbol{x}_n\right) \tag{2}$$

- I want to select the model parameters $\mathbf{w}$ that maximize the probability of observing the data, that is

$$\operatorname{argmax}_{\boldsymbol{w}} \prod_{n=1}^{N} P\left(y_n|\boldsymbol{x}_n\right) \tag{3}$$

- Maximizing $\prod_{n=1}^{N} P\left(y_n|x_n\right)$ is equivalent of minimizing

$$-\frac{1}{N}\ln\left(\prod_{n=1}^{N} P\left(y_n|\boldsymbol{x}_n\right)\right) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1}{P\left(y_n|\boldsymbol{x}_n\right)}\right) \tag{4}$$

- Since $P\left(y_n|\boldsymbol{x}_n\right) = \theta\left(y_n\boldsymbol{w}^T\boldsymbol{x}_n\right)$, and $\theta(s) = \frac{e^s}{1+e^s}$, I end up minimizing

$$\begin{aligned} E_{in} =& \frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1}{\theta\left(y_n\boldsymbol{w}^T\boldsymbol{x}_n\right)}\right) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1+e^{y_n\boldsymbol{w}^T\boldsymbol{x}_n}}{e^{y_n\boldsymbol{w}^T\boldsymbol{x}_n}}\right) \\ =& \frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1}{e^{y_n\boldsymbol{w}^T\boldsymbol{x}_n}}+1\right) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(1+e^{-y_n\boldsymbol{w}^T\boldsymbol{x}_n}\right) \end{aligned} \tag{5}$$

- To minimize the error function, I find the gradient by computing the partial derivatives for w_i

  `I apply the two following derivation rules`

$$\frac{d}{dx}\left[e^{f(x)}\right] = e^{f(x)}\cdot f'^{(x)} \tag{6}$$

$$\frac{d}{dx}\ln[f(x)] = \frac{1}{f(x)}f(x) \tag{7}$$

`and obtain`

$$\begin{aligned} \nabla E_{in}(\mathbf{w}) =& \frac{1}{N}\sum_{n=1}^{N}\frac{1}{1+e^{-y_n\boldsymbol{w}^T\boldsymbol{x}_n}}\left(-y_n\boldsymbol{x}_n e^{-y_n\boldsymbol{w}^T\boldsymbol{x}_n}\right) \\ =& \frac{1}{N}\sum_{n=1}^{N}\frac{e^{-y_n\boldsymbol{w}^T\boldsymbol{x}_n}}{1+e^{-y_n\boldsymbol{w}^T\boldsymbol{x}_n}}\left(-y_n\boldsymbol{x}_n\right) \end{aligned} \tag{8}$$

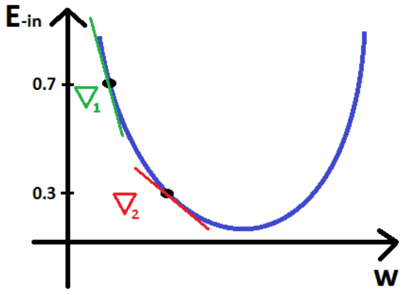
- Since $\theta(s) = \frac{e^s}{1+e^s}$, I obtain

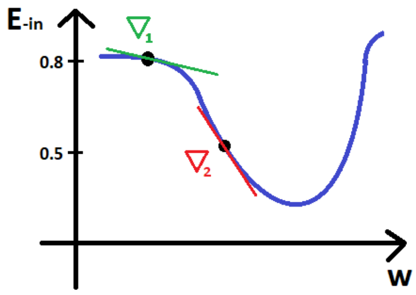
$$\nabla E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} -y_n\mathbf{x}_n\theta\left(-y_n\mathbf{w}^\top\mathbf{x}_n\right) \tag{9}$$

8.2 Argue that a 'misclassified' example contributes more to the gradient than a correctly classified one.

I think that it depends on the used loss function. If it is a strictly convex loss function, then a "missclassified" example will contributes more to the gradient than a correctly classified one. That is because, contrarily to a correcly classified one, a "missclassified" data-point will increases the in-sample error. In a strictly convex function a greater error corresponds to a point of the curve where the slope is steper. And the slope of the function at any single point is given by its gradient function. This is not always true if the loss function is not strictly convex.

Next, I provide two sketches showing two loss functions: the first one shows a strictly convex function where the statement about the gradient is always valid, the second one shows a non convex function where the statement is not always valid.





Dimensionality Reduction, Classification and Clustering

Exercise 9 (Clustering and classification I)

a) Run the k-Means algorithm with k = 3 to cluster the dataset.

- **Description of the software.**

  In order to obtain the centroids I used my own implementation of the unsupervised k-means clustering algorithm that I used in the previous assigment. I initialized the algorithm by placing the 3 centroids at the 3 first points of the dataset. To evaluate the quality of the clusters I used my implemented loss function that computes the sum of squared errors among them. Based on the distances between the points and the centroids, I assign each point to the nearest cluster. Then I update the new centroids computing the mean of the new assigned points, and finally, I compute the loss of the new clusters. The algorithm continues this process until the centroids converge, which occurs when the loss also converge. Once I obtained the centroids, I used them to obtain the indexes of the data points assigned to each clusters. Finally I used these indexes and the true labels to count the proportion of each digit I obtained in each cluster.

- **Three cluster centers as images (no MDS priot to clustering).**

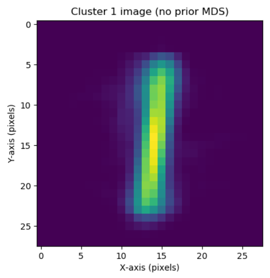


Figure 17: Centroid 1 image

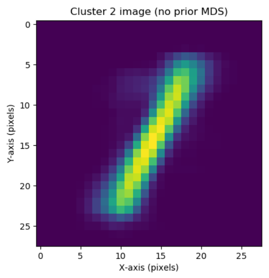


Figure 18: Centroid 2 image

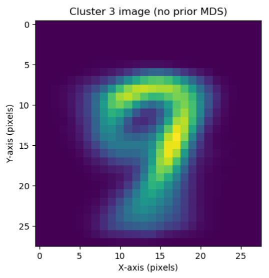


Figure 19: Centroid 3 image

- **Percentage of each digit per cluster(no MDS prior to clustering):**

Table 1: **no MDS prior to clustering**

	Proportion of 1	Proportion of 7	Proportion of 9
Centroid 1	83.33%	8.94%	7.72%
Centroid 2	85.35%	11.62%	3.03%
Centroid 3	0.15%	48.46%	51.40%

- **Are the results meaningful?**

  As it is possible to observe from the proportions, the result is not optimal. Both clusters 1 and 2 contain mainly the digit 1, while cluster 3 contains almost equal proportion of digit 7 and 9. Even if the proportion are not optimal, from the images of the centroids we can see that we have a meaningful result. The digits 1 and 9 are well defined in the centroids image 1 and 3, respectively, that is probably due to the fact that digit 7 partially overlaps with digit 9. While in centroid image 2 we can only see part of the digit 7.

- **Plot true classes division in 2D and k-means predicted clusters division in 2D (no MDS prior to clustering) (both not requested).**

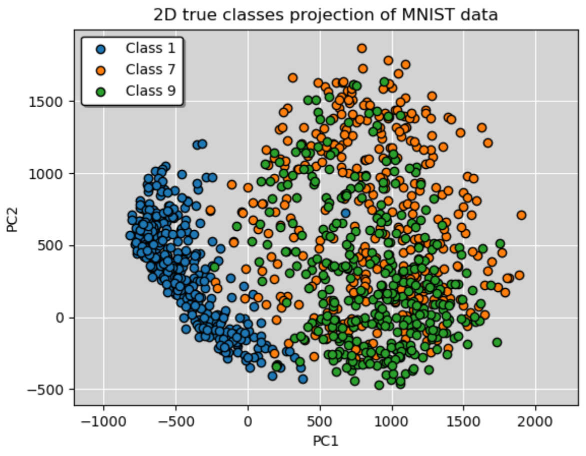


Figure 20: **True classes division**

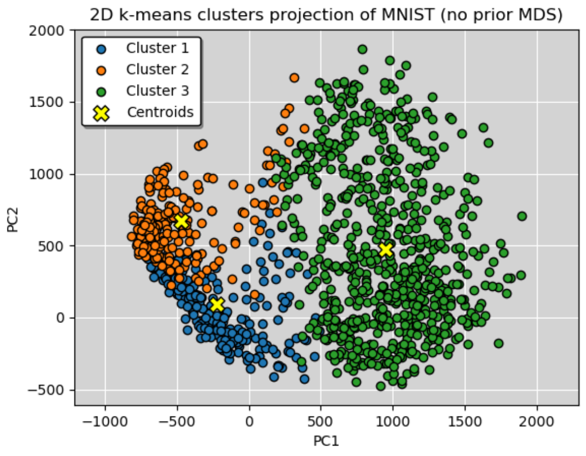


Figure 21: **Clustering (No prior MDS)**

b) K-NN test accuracy for k-best.

- **K-NN (no MDS prior to prediction)**
  K-best = 1
  1-NN test classification accuracy = 0.96%

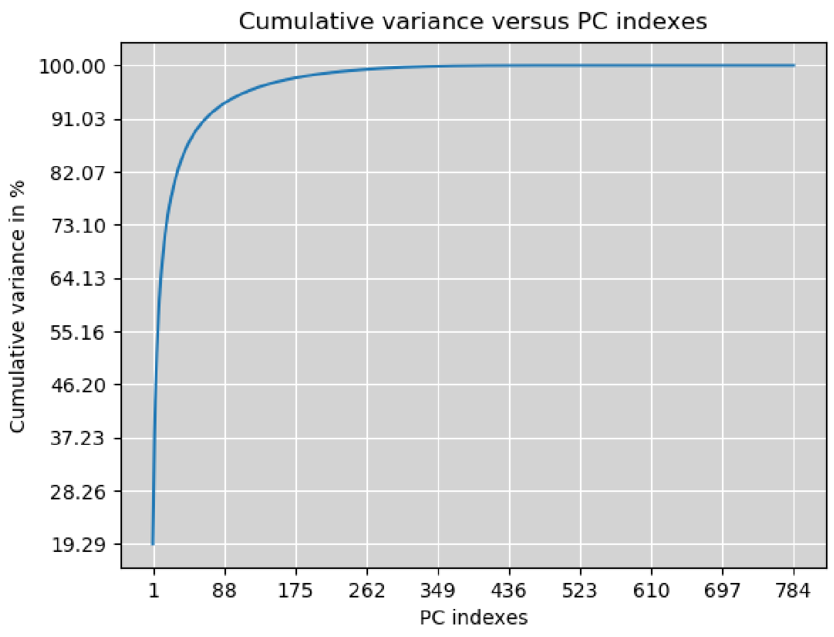
Exercise 10 (Clustering and classification after dimensionality reduction)

a) Compute the PCA of the training data.

- **Description of the software.**

  I implemented the PCA function using the linalg.eig function provided in the numpy package. I first obtained the covariance matrix of the dataset, then I used the linalg.eig function to perform the eigendecomposition of the transposed covariance matrix. This function return both, the principal components (eigenvectors) that maximize the variance of the projected data-points, and the captured variance (eigenvalues) by each principal component. It is interesting to note that in PCA, maximizing the projected variance is equivalent of minimazing the squared projection error. I used the numpy function argsort to sort both eigenvectors and eigenvalues by ascending order of captured variance, and finally I plotted the cumulative variance with respect to the principal components.

- **Plot the cumulative variance (in %).**



- **One liner about the cumulative variance plot.**

  It is possible to observe that most of the variance is captured within the first 100 principal components, and that around the 300th component the variance captured seems to be already 100%.

b) Run the k-Means algorithm, again with k = 3 to cluster the data projected on the first principal components. Experiment with 20 and 200 components.

- **Percentage of each digit per cluster.**

Table 2: **Clustering after MDS to 200 pcs**

	Proportion of 1	Proportion of 7	Proportion of 9
Centroid 1	83.33%	8.94%	7.72%
Centroid 2	85.35%	11.62%	3.03%
Centroid 3	0.15%	48.46%	51.40%

Table 3: **Clustering after MDS to 20 pcs**

	Proportion of 1	Proportion of 7	Proportion of 9
Centroid 1	83.33%	8.94%	7.72%
Centroid 2	86.22%	11.22%	2.55%
Centroid 3	0.15%	48.46%	51.39%

Table 4: **Clustering after MDS to 2 pcs (not requested)**

	Proportion of 1	Proportion of 7	Proportion of 9
Centroid 1	1.35%	37.89%	60.76%
Centroid 2	88.25%	7.91%	3.84%
Centroid 3	0.38%	66.03%	33.59%

- **Three cluster centers as images.**

Centroids images obtained after MDS to 200 pcs

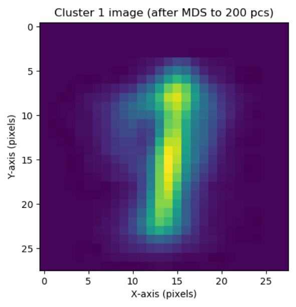


Figure 22: Centroid 1 image

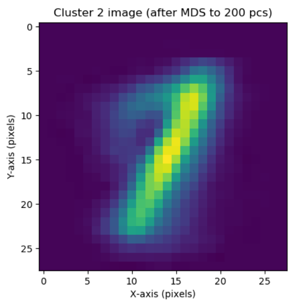


Figure 23: Centroid 2 image

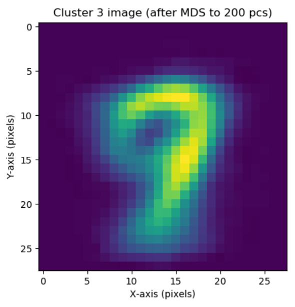


Figure 24: Centroid 3 image

Centroids images obtained after MDS to 20 pcs

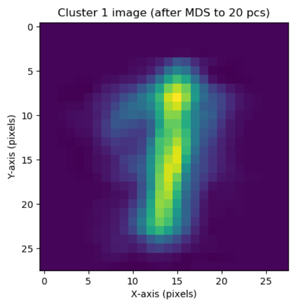


Figure 25: Centroid 1 image

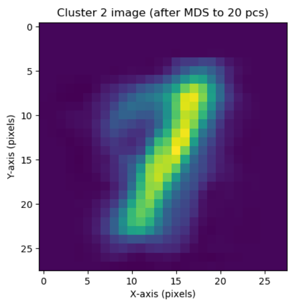


Figure 26: Centroid 2 image

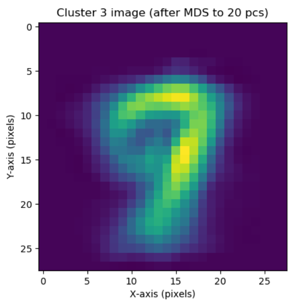


Figure 27: Centroid 3 image

Centroids images obtained after MDS to 2 pcs **(not requested)**

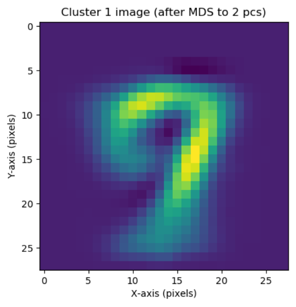


Figure 28: Centroid 1 image

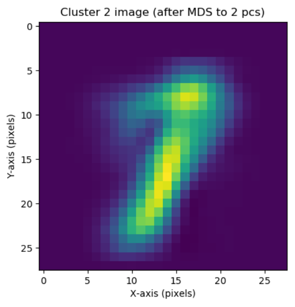


Figure 29: Centroid 2 image

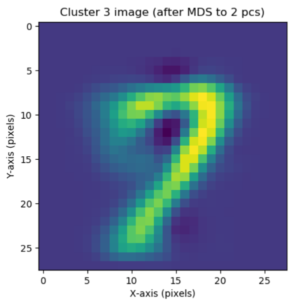


Figure 30: Centroid 3 image

- **2D projection of centroids and cluster division (k-means performed after PCA) (not requested).**

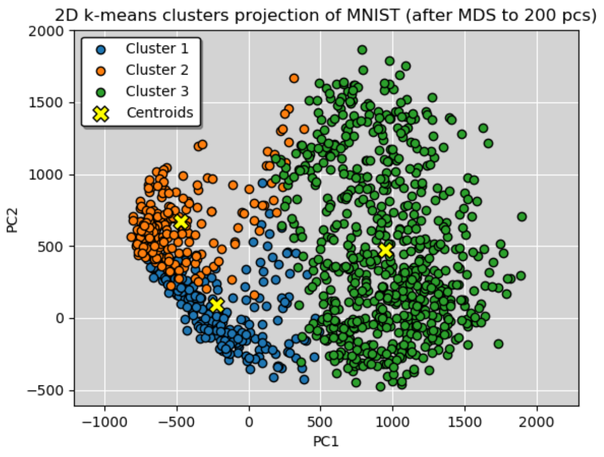


Figure 31: **Clustering after MDS to 200 pcs**

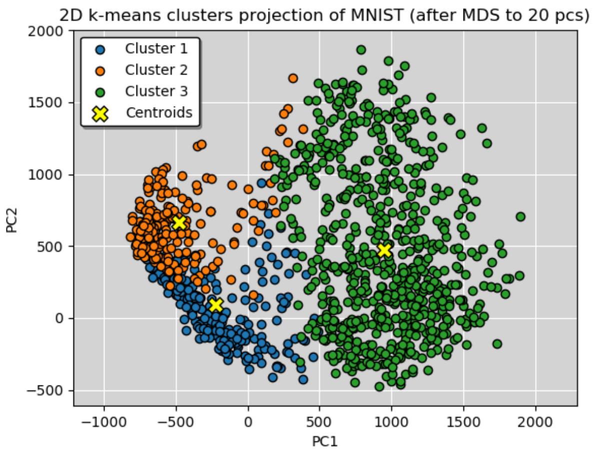


Figure 32: **Clustering after MDS to 20 pcs**

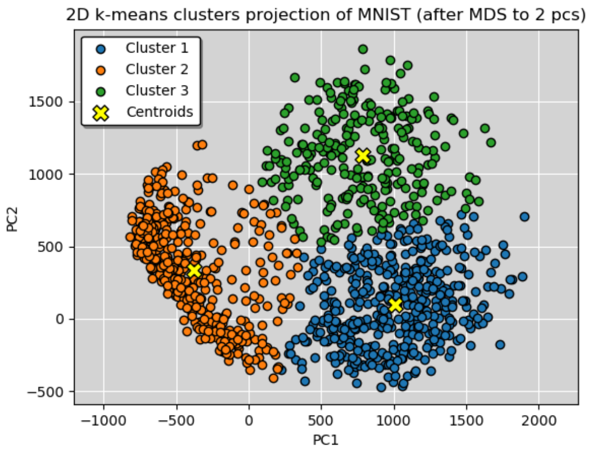


Figure 33: **Clustering after MDS to 2 pcs**

- **Compare with the non-PCA clustering result in the previous exercise.**

  K-means clustering find the optimal clusters by minimazing the sum of squared errors, which is based on the distance between all data-points and their closer centroids. So the effect of the PCA before k-means, which allows to cluster the data in low dimensional space, depends on how much distance between data-points is preserved. It is possible to observe that the results, of k-means without MDS and k-means after MDS to 200 PCs, are identical. This was expected since we observed that most of the variance is captured within the first 100 PCs. Also, there is only a slighly difference between the results just mentioned and the result obtained by performing k-means after MDS to 20 dimensions. This can be explained by the fact that the first 20 pcs capture a large amount of variance (more than 70%). As an extra step, not requested in the exercise, I performed K-means clustering after MDS to 2 dimensions. In this case I obtained a different, and in my opnion better, result: the first centroid captured a larger proportion of 9 (60.76%), the second a larger proportion of 1 (88.25%) and the third a larger proportion of 7 (66.03%).

c) Classification. Train a k-NN classifier on the dataset:

- **K-NN after MDS to 20 pcs:**
  K-best = 1

1-NN test classification accuracy = 0.97%

- **K-NN after MDS to 200 pcs:**
  K-best = 1
  1-NN test classification accuracy = 0.96%
- **Short description about comparison of performances in Ex.9b and Ex.10c.**

  The performance of the k-NN, without MDS and after MDS to 200 dimensions is identical. As in the k-means results, this was expected because most of the variance is captured within the first 100 pcs. While, the performance of the k-NN after MDS to 20 dimensions is slightly better.