Modern Machine Learning
Computer Assignment #1

1. *Linear Regression via Gradient Descent:* using Python or MATLAB, implement the gradient descent algorithm to solve linear regression

   Background: The gradient descent algorithm for linear regression consists of updates of the form

$$
\begin{aligned}
\beta_0^{k+1} &= \beta_0^k - \alpha \sum_{i=1}^{n} \big(h_{\beta^k}(\mathbf{z}_i) - y_i\big) \\
\beta_1^{k+1} &= \beta_1^k - \alpha \sum_{i=1}^{n} \big(h_{\beta^k}(\mathbf{z}_i) - y_i\big)\, z_{i1} \\
\beta_2^{k+1} &= \beta_2^k - \alpha \sum_{i=1}^{n} \big(h_{\beta^k}(\mathbf{z}_i) - y_i\big)\, z_{i2} \\
&\vdots \\
\beta_p^{k+1} &= \beta_p^k - \alpha \sum_{i=1}^{n} \big(h_{\beta^k}(\mathbf{z}_i) - y_i\big)\, z_{ip},
\end{aligned}
$$

   until convergence is achieved, where $h_{\beta^k}(\mathbf{z}_i) = \beta_0^k + \beta_1^k z_{i1} + \beta_2^k z_{i2} + \cdots + \beta_p^k z_{ip}$. You can assume that the algorithm has converged when the entries β_j vary less than a specified value ϵ.

   Python file: The script lin_regression.py defines a Python function *gradient_descent*$(\cdot)$ to obtain the least squares solution using randomly generated data. It also plots the least squares approximation.
   Submission guidelines: Your submission should include:

   - A unique **zip folder** consisting of two files: a modified version of lin_regression.py to realize the *gradient_descent*$(\cdot)$ function and a pdf file with figure(s) to explain your coding logic and results.
   - The structure of the function *gradient_descent*$(\cdot)$ is described as followings: $\boldsymbol{\beta}_{\text{next}} = \text{gradient\_descent}(\boldsymbol{\beta}, \alpha, \mathbf{X}, \mathbf{y})$, where $\boldsymbol{\beta}$ are the current learned coefficients, α is the learning rate, $\mathbf{X}$ indicate the observations, $\boldsymbol{\beta}_{next}$ are the learned linear regression coefficients, $\mathbf{y}$ is the output $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, and $\boldsymbol{\epsilon}$ is Gaussian noise. This function

should implement gradient descent to obtain the least squares solution. Please realize this function where is indicated in the script. You do not need to modify other functions or codes in the script.

- Please rename the modified file lin_regression.m by adding your last name to the script name. For example lin_regression_smith.m. **This should be the main function**.

MATLAB files: The script lin_regression_example.m uses the MATLAB function $fitlm(\cdot)$ to obtain the least squares solution using randomly generated data. It also plots the least squares approximation.
Submission guidelines: Your submission should include:

- A unique **zip folder**, which should include a modified version of lin_regression_example.m and mylinreg.m. The structure of the function mylinreg.m is the following: $\hat{\boldsymbol{\beta}}_{LS} = \text{mylinreg}(\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ are the observations, $\hat{\boldsymbol{\beta}}_{LS}$ are the linear regression coefficients, and $\mathbf{y}$ is the output of the model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is Gaussian noise. This function should implement gradient descent to obtain the least squares solution. Please insert this function where is indicated in the script. Make sure the results you obtain are very close to the ones returned by the function fitlm. Please rename the modified file lin_regression_example.m replacing the word 'example' in the provided script with your last name. For example lin_regression_smith.m. **This should be the main function**.
- A pdf file with a comparative figure of the two lines obtained by your function and the function fitlm. Explain possible differences.

2. *Lasso Regression via Coordinate Descent:* using MATLAB, implement the coordinate descent algorithm to solve lasso regression given by

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1.$$

Background: The coordinate descent algorithm consists of sequential updates of the form

$$\begin{aligned}
\beta_1^{k+1} &= \arg\min_{\beta_1^k} f(\beta_1^k, \beta_2^k, \ldots, \beta_p^k)\\
\beta_2^{k+1} &= \arg\min_{\beta_2^k} f(\beta_1^k, \beta_2^k, \ldots, \beta_p^k)\\
&\qquad\qquad\qquad\vdots\\
\beta_p^{k+1} &= \arg\min_{\beta_p^k} f(\beta_1^k, \beta_2^k, \ldots, \beta_p^k),
\end{aligned}$$

until convergence is achieved. You can assume that the algorithm has converged when the coordinates vary less than a specified value ϵ. Remember that the lasso solution is given by

$$\beta_i^{k+1} = \eta^S_{\lambda/\|\mathbf{X}_i\|_2^2}(\tilde{g}_i^k),$$

where $\tilde{g}_i^k = \frac{\mathbf{X}_i^T(\mathbf{y}-\mathbf{X}_{-i}\boldsymbol{\beta}_{-i}^k)}{\mathbf{X}_i^T\mathbf{X}_i}$, and the operator $\eta^S_\lambda(z) = \begin{cases} z-\lambda & \text{if } z > \lambda \\ z+\lambda & \text{if } z < -\lambda \\ 0 & \text{if } -\lambda \le z \le \lambda \end{cases}$

Python file: The script lasso_regression.py defines a Python function *lasso_regression*$(\cdot)$ to obtain the lasso solution using randomly generated data. It also plots the lasso regression results with three different values of the parameter λ.

Submission guidelines: Your submission should include:

- A unique **zip folder** consisting of two files: a modified version of lasso_regression.py to realize the *lasso_regression*$(\cdot)$ function and a pdf file with figure(s) to explain your coding logic and results.
- The structure of the function *lasso_regressiont*$(\cdot)$ is described as followings: $\boldsymbol{\beta}_{\text{next}} = \text{lasso\_regression}(\boldsymbol{\beta}, \lambda, \mathbf{X}, \mathbf{y})$, where $\boldsymbol{\beta}$ are the current learned coefficients, λ is the parameter indicating the $\mathcal{L}_1$ norm penalty, $\mathbf{X}$ indicate the observations, $\boldsymbol{\beta}_{next}$ are the learned lasso regression coefficients, and $\mathbf{y}$ is the output of the model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is Gaussian noise. This function should implement lasso regression to obtain the solution. Please realize this function where is indicated in the script. You do not need to modify other functions or codes in the script.

- The script pre-define three values of λ to indicate the $\mathcal{L}_1$ norm penalty. Please compare the learned coefficients in different λ and explain the effect of λ on the learned coefficients in the pdf files.
- Please rename the modified file lasso_regression.m by adding your last name to the script name. For example lasso_regression_smith.m. **This should be the main function**.

MATLAB files: The script lasso_example.m uses the MATLAB function $lasso(\cdot)$ to obtain the lasso solution using randomly generated data.
Submission guidelines: Your submission should include:

- A unique **zip folder**, which should include a modified version of lasso_example.m and mylasso.m. The function mylasso.m has the following structure: $\hat{\boldsymbol{\beta}}_{lasso}$ = mylasso($\mathbf{X}, \mathbf{y}, \lambda$), where $\mathbf{X}$ are the observations, $\hat{\boldsymbol{\beta}}_{lasso}$ are the lasso coefficients, λ is the regularization coefficient and $\mathbf{y}$ is the output of the model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where ϵ is Gaussian noise. This function should implement coordinate descent to obtain the lasso solution. Please insert this function where is indicated in the script. Make sure the results you obtain are very close to the ones returned by the function lasso.

  Please rename the modified file lasso_example.m replacing the word 'example' in the provided script with your last name. As in the previous exercise. **This should be the main function**.
- A pdf file with a figure showing the lasso coefficients $\hat{\boldsymbol{\beta}}_{lasso}$ vs the regularization parameter λ. Vary λ between 0.001 and 10 for this experiment. Explain your results.