Modern Machine Learning
Computer Assignment #2

1. *Binary Logistic Regression:* Using MATLAB or Python, implement the cost function for logistic regression and then minimize it.

   Background: The cost function to be minimized is the *negative* log-likelihood function

   $$NLL(\boldsymbol{\beta}) = -\sum_{i=1}^{n} y_i \log\left(p(\mathbf{x}_i, \boldsymbol{\beta})\right) + (1 - y_i) \log\left(1 - p(\mathbf{x}_i, \boldsymbol{\beta})\right),$$

   where $p(\mathbf{x}_i, \boldsymbol{\beta}) = \frac{1}{1+e^{-\boldsymbol{\beta}^T \mathbf{x}_i}}$ (logistic function). The derivative of the negative log-likelihood can be expressed as

   $$\frac{\partial NLL(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\sum_{i=1}^{n} \mathbf{x}_i \left(y_i - p(\mathbf{x}_i, \boldsymbol{\beta})\right)$$

   Python files: The log_regression_example.py uses the Tensorflow package to realize the logistic regression for binary classification based on randomly generated data. It also plots the data and the decision boundary.
   Submission guidelines: Your submission should include:

   - A unique **zip folder**, which should include a modified version of log_regression.py, in which you need to realize the function *logistic_regression*. The function *logistic_regression* is designed to minimizing the cost function $NLL(\boldsymbol{\beta})$ via gradient descent.
   - the structure of $logistic\_regression(\boldsymbol{beta}, lr, \boldsymbol{x\_batch}, \boldsymbol{y\_btach})$ is summarized as follows: (i) $\boldsymbol{beta}$ is a 3×1 vector, (ii) lr denotes the learning rate, (iii) $\boldsymbol{x\_batch}$ is the dataset of two linear models, the dimension of $\boldsymbol{x\_batch}$ is 2000×3, which means there are 2000 data points $\{(x, y, 1)\}_{i=1}^{2000}$, we extend $\{(x, y)\}_{i=1}^{2000}$ to $\{(x, y, 1)\}_{i=1}^{2000}$ for implement the matrix operation $\boldsymbol{x\_batch} \cdot \boldsymbol{beta}$, where $beta_3$ is the bias, e.g., $x \cdot beta_1 + y \cdot beta_2 + beta_3 \cdot 1$. In addition, $\boldsymbol{y\_btach}$ is the training labels for $\boldsymbol{x\_batch}$. If (x, y) belongs to the first linear model, then $y_{batach} = 1$, otherwise $y\_batach = 0$.

- Please rename the modified file lin_regression.py by adding your last name to the script name, e.g., lin_regression_smith.py.
- A pdf file with a comparative figure of the two lines obtained by your function and the log_regression_example.py. Explain possible differences.

MATLAB files: The script log_regression_example.m uses the MATLAB function $glmfit(\cdot)$ to obtain the logistic regression coefficients for binary classification using randomly generated data. It also plots the data and the decision boundary.
Submission guidelines: Your submission should include:

- A unique **zip folder**, which should include a modified version of log_regression_example.m and costLogistic.m. The structure of the function costLogistic.m is the following: $[NLL(\boldsymbol{\beta}), \frac{\partial NLL(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}] =$ costLogistic$(\boldsymbol{\beta}, \mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ are the observations, $\boldsymbol{\beta}$ are the logistic regression coefficients, and $\mathbf{y}$ are the class labels of the examples in $\mathbf{X}$. Make sure the results you obtain are very close to the ones returned by the funcion glmfit(). Please rename the modified file log_regression_example, replacing the word 'example' in the provided script with your last name. For example log_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 glmfit(). Explain possible differences.

2. *Multiclass classification and regularization:* Using MATLAB, implement the *one-vs-all* technique for multiclass classification using regularized logistic regression.

   Background: The one-vs-all approach consists of training K separate binary classifiers, where K is the number of classes, for each one of the different classes.

   - Train K separate binary classifiers producing $\boldsymbol{\beta}_j$ $(j = 1, 2, \ldots, K)$
   - For every new example $\mathbf{x}$, find the predicted label as

   $$\hat{j} = \arg\max_j p(\mathbf{x}, \boldsymbol{\beta}_j), \text{where } p(\mathbf{x}_i, \boldsymbol{\beta}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^T \mathbf{x}_i}}$$

   Recall that the equations for regularized logistic regression are the following

   $$NLL(\boldsymbol{\beta}) = -\sum_{i=1}^{n} y_i \log\left(p(\mathbf{x}_i, \boldsymbol{\beta})\right) + (1 - y_i) \log\left(1 - p(\mathbf{x}_i, \boldsymbol{\beta})\right) + \frac{\lambda}{2} \sum_{j=1}^{m} \beta_j^2,$$

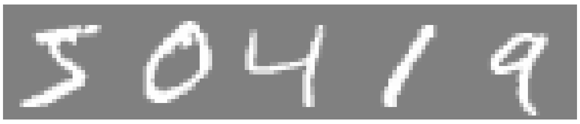

Figure 1: Example images from MINST database

where m is the number of variables, and $p(\mathbf{x}_i, \boldsymbol{\beta})$ is the logistic function defined above.

$$\frac{\partial NLL(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}_j} = -\sum_{i=1}^{n} x_i^{(j)} \left(y_i - p(\mathbf{x}_i, \boldsymbol{\beta})\right) + \lambda \beta_j$$

where $x_i^{(j)}$ is the j-th component of vector $\mathbf{x}_i$.

The data used in this exercise is a portion of the MINST database. It contains handwritten digits 0-9. The size of the images is 28×28 pixels, these pixels are vectorized producing features of size 784×1. Your MATLAB program will attempt to recognize these digits. Examples of the handwritten digits can be seen in Figure 1.

Python files: The multiclass_log_reg.py is the template to implment your one-vs-all classifier.
Submission guidelines: Your submission should include:

- To run the multiclass_log_reg.py script, you are required to install **scipy** package into your Anaconda environment. The command is **conda install scipy**.
- A unique **zip folder**, which should include a modified version of multiclass_log_reg.py, in which you need to realize the function *logistic_regression*. The function *logistic_regression* is designed to minimizing the cost function $NLL(\boldsymbol{\beta})$ with $L2$ norm regularization.
- the structure of $logistic\_regression(\boldsymbol{beta}, lr, \boldsymbol{x\_batch}, \boldsymbol{y\_btach}, \lambda 1)$ is summarized as follows: (i) $\boldsymbol{beta}$ is a 785×1 vector, (ii) lr is the learning rate, (iii) $\boldsymbol{x\_batch}$ is the MINIST dataset consisting 5000 images, the dimension of $\boldsymbol{x\_batch}$ is 5000×785, which means there are 5000 digit images $\{(\boldsymbol{x}, 1)\}_{i=1}^{5000}$, we extend $\{\boldsymbol{x}\}_{i=1}^{5000}$ to $\{(\boldsymbol{x}, 1)\}_{i=1}^{5000}$ for implement the matrix operation $\boldsymbol{x\_batch} \cdot \boldsymbol{beta}$, where $beta_{785}$ is the bias, e.g., $\sum_{i=1}^{784} x_i \cdot beta_i + beta_{785} \cdot 1$. $\boldsymbol{y\_btach}$ is the training labels for $\boldsymbol{x\_batch}$, its dimension is 5000×10. If $\boldsymbol{x}$ belongs to the ith digit, then $y_{batach} = [0, \cdots, 1, \cdots, 0]$.
- Please rename the modified file logistic_regression.py by adding your last name to the script name, e.g., lin_regression_smith.py.

- A pdf file with a figure showing the classification acuracy vs the regularization parameter λ. Vary λ from 0 to 20, in steps of 1. Explain your results.

MATLAB files: The script multiclass_log_reg_example.m can be used as a template to implement your one-vs-all classifier. **The function fmincg.m is also provided to speed up the optimization process**. This function works the same way as the function fminunc() provided by MATLAB, but it is faster.

Submission guidelines: Your submission should include:

- A unique **zip folder**, which should include a modified version of multiclass_log_reg_example.m and multiclassLog.m. The structure of the function multiclassLog.m is the following: $\hat{\boldsymbol{\beta}}_{matrix} = \text{multiclassLog}(\mathbf{X}, \mathbf{y}, K)$, where $\mathbf{X}$ are the observations, $\hat{\boldsymbol{\beta}}_{matrix}$ are the K classifiers stacked column-wise as a matrix, and K is the number of class labels, in this case $K = 10$. This function should train K binary classifiers using your past implementation of logistic regression. You can use a *for* loop to achieve this. Be careful setting the labels so that for class j, the training examples belonging to class j will be assigned label 1, and the other examples label 0. The function $pred = \text{predictmulticlass}(\hat{\boldsymbol{\beta}}_{matrix}, \mathbf{y})$ is provided to predict the class labels using $\hat{\boldsymbol{\beta}}_{matrix}$.

  Please rename the modified file multiclass_log_reg_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 classification acuracy vs the regularization parameter λ. Vary λ from 0 to 20, in steps of 1. Explain your results.

Hint: For $\lambda = 0$, the classification accuracy should be about 87.5%.