ECE 449 Machine Learning (ZJU-UIUC Institute)
Assignment 1

Due Date: October 29, 2020.

1 Overview

Welcome to ECE449 assignment 1!

In this assignment, you will complete 16 Python functions in 4 IPython Notebooks using your knowledge and skills in machine learning topics, and you will answer 5 conceptual questions to demonstrate your understanding of the course materials. This assignment has 64 pts (programming) + 36 pts (conceptual) = 100 pts in total.

2 Programming Questions

2.1 Programming environment installation

You need Python, jupyter notebook, matplotlib, numpy, scikit-learn, and scipy to run the 4 IPython notebook files and write your implementations. You can use Python 3.7 or Python 3.8 and use conda to create a virtual environment for programming assignments in ECE449 only. We recommend the following steps to install a working environment on your computer for programming assignments in ECE449:

1. Go to anaconda website for downloading individual version installers (https://www.anaconda.com/products/individualDownload), download the anaconda installer according to your OS and install anaconda on your computer. You can use either user mode installation or admin(root) mode installation.

2. Launch anaconda prompt from your computer, and type the following command to create an environment named ece449_hws:

```
conda create -n ece449_hws python=3.7 matplotlib numpy scikit-learn scipy
```

3. Each time before you run the jupyter notebook files, remember to activate your environment by typing the following command in your anaconda prompt:

```
conda activate ece449_hws
```

4. In the anaconda prompt, use command `cd` to change to the directory the 4 IPython notebooks are in, then type:

```
jupyter notebook
```

to launch jupyter notebook and its web interface.

Now you should have a working environment that will run the jupyter notebook files in assignment 1.

2.2 Programming questions brief descriptions

There are 4 notebook files you need to work on in this assignment.

The first notebook, `exercise1-template.ipynb` is about linear regression, you will implement 4 Python functions (4pts each):

`warmUpExercise`, `computeCost`, `gradientDescent`, and `fit_lr_and_predict`

The second notebook, `exercise2-template.ipynb` is about logistic regression, you will implement 4 Python functions (4pts each):

`sigmoid`, `costFunction`, `predict`, and `costFunctionReg`

The third notebook, `exercise3-template.ipynb` is about regularized linear regression, you will implement 4 Python functions (4pts each):

`linearRegCostFunction`, `learningCurve`, `polyFeatures`, and `validationCurve`

The fourth notebook, `exercise4-template.ipynb` is about SVM, you will implement 4 Python functions (4pts each):

`gaussianKernel`, `dataset3Params`, `processEmail`, and `emailFeatures`

The programming part has $4 \times 16 = 64$ pts in total.

You will be provided with the 4 IPython notebook template files, a Python script file `utils.py`(which contains helper functions for exercise4), and two folders `Data` and `Figures` with data files and sample figures for the 4 notebooks. To run the Python implementations in your notebooks, you should place `utils.py` and the two folders `Data` and `Figures` under the same directory as your notebooks. You only need to modify the 4 IPython notebooks in this assignment.

2.3 Autograder notice

We use autograder to grade your jupyter notebook, and here are some things we want to highlight:

1. We only grade the 4 Python functions you write in each notebook.

2. Do NOT modify input and output interfaces of any of these graded functions, autograder gives 0 for function with modified input/output interfaces.

3. Do NOT define any of these graded functions multiple times in different code cells, autograder gives 0 for function defined multiple times in different code cells.

4. You don't need and should not import extra Python modules in your implementations of any of these graded functions, autograder gives 0 for function that needs to import extra Python modules. You should be able to complete all your implementations with modules and functions imported at the beginning of the four IPython notebooks.

5. No partial credit for the graded functions. Each correct function gives you 4 pts.

When you submit your notebook, make sure the jupyter notebook file names satisfy the requirements stated in the ”Submission” section. If you do not rename your notebook properly, autograder will not be able to see your notebook and it grades that notebook 0 pts.

2.4 Collaboration policy

You should not look at python code from other students or from the web, the Python implementations in your 16 functions should be your own work.

3 Conceptual Questions

There are 5 questions you need to answer in the conceptual question part. You can write down your answer on a few sheets of paper and scan them to a single PDF file, or you can use digital pen to work on these questions and export your answers to a single PDF file.

3.1 Problem 1

Assume we have 3 points, i.e., (1,2), (2,1), (3,2), in a 2-D Euclidean space. We want to fit a line which is $y = w_0 + w_1 x$ with respect to these 3 points. Derive the optimal solution for w_0 and w_1 with mean square error (MSE) loss. Show your steps for full score. (8pts)

3.2 Problem 2

Assume we have 4 points, i.e., (-2,1), (-1,0), (1,0), (2,2), in a 2-D Euclidean space. We want to fit a 2nd order polynomial function $y = w_0 + w_1 x + w_2 x^2$ with respect to these 4 points. Remember the closed form solution would be $\hat{w} = argmin(\boldsymbol{H}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{H}\boldsymbol{w} - \boldsymbol{y})$, where $\boldsymbol{w} = [w_0, w_1, w_2]^T$. Write down what would be $\boldsymbol{H}$ and $\boldsymbol{y}$. (7pts)

3.3 Problem 3

Assume we have 6 points showing 6 observations in a 2-D Euclidean space as below:

Observation	X_1	X_2	Y
1	0	2	class 1
2	0	3	class 1
3	1	3	class 1
4	1	1	class 2
5	2	1	class 2
6	2	2	class 2

Using a maximum margin classifier, determine the optimal separating hyperplane and give an equation for it, and determine which observations are the support vectors. (7pts)

3.4 Problem 4

Describe at least 4 differences or similarities between bagging and boosting. (8pts)

3.5 Problem 5

LOOCV is a statistical cross-validation method often used when our dataset is quite small. It stands for "Leave-One-Out Cross Validation". If we apply LOOCV method with some model on a dataset with N samples, we partition the dataset N times, in N iterations: in iteration i we partition the dataset into a train set by removing the i-th sample from original dataset and use the test set with only the i-th sample to validate our model.

The LOOCV estimate of the test Mean-Square-Error(MSE) is defined as:

$$CV_{(n)} = \frac{1}{N}\sum_{i=1}^{N} MSE_i$$

Where $MSE_i = \frac{(y_i - \hat{y}_i)^2}{1} = (y_i - \hat{y}_i)^2$. y_i is the true label value of the i-th sample, and $\hat{y}_i$ is the predicted label value of the i-th sample.

Even for simple models like ordinary least square linear regression, LOOCV could take a significant amount of computation. Luckily, a "shortcut" exists for LOOCV on ordinary least square linear regression. For example, for a univariate least square linear regression model we obtain

$$CV_{(n)} = \frac{1}{N}\sum_{i=1}^{N}(\frac{y_i - \hat{y}_i}{1 - h_i})^2$$

Where h_i is the leverage statistic for the i-th sample, defined as

$$h_i = \frac{1}{N} + \frac{(x_i - \bar{x})^2}{\sum_{j=1}^{N}(x_j - \bar{x})^2}$$

$\bar{x}$ is the mean of all the sample features x_i.

Prove that for univariate ordinary least square linear regression model we have the following:

$$CV_{(n)} = \frac{1}{N}\sum_{i=1}^{N} MSE_i = \frac{1}{N}\sum_{i=1}^{N}(\frac{y_i - \hat{y}_i}{1 - h_i})^2$$

Hint: With the following notations:

X is an N by 2 matrix: $X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} = \begin{bmatrix} \vec{x_0} & \vec{x} \end{bmatrix}$

$\vec{x_i}$ is the i-th row of the matrix X: $\vec{x_i} = \begin{bmatrix} 1 & x_i \end{bmatrix}$

$\vec{x_0}$ is a vector of 1s with length N: $\vec{x_0} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$

$\vec{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$ is the coefficient vector for the least square model,

You can try proving $\vec{x_i}(X^TX)^{-1}\vec{x_i}^T = h_i$. Try your best to connect your understanding and interpretations of the closed form solution for ordinary least square linear regression $\vec{\beta} = (X^TX)^{-1}X^T\vec{y}$ with the formula $\vec{x_i}(X^TX)^{-1}\vec{x_i}^T = h_i$, then you should be able to write your proof. (6pts)

4 Submission

You only need to submit 4 IPython notebook files with your 16 Python function implementations and a PDF file containing your answers to the 5 conceptual questions to BlackBoard. Pay attention that your 4 IPython notebook files should be renamed in the format of `exercise[i]-[your_studentID].ipynb`, and your PDF file should be renamed in the format of `conceptual-[your_studentID].pdf`. For example, a student whose ID number is 3170100111 should submit the following 5 files to BlackBoard:

```
exercise1-3170100111.ipynb, exercise2-3170100111.ipynb
exercise3-3170100111.ipynb, exercise4-3170100111.ipynb
conceptual-3170100111.pdf
```

Please do NOT submit zipped files.

Any of the 5 files not submitted or not meeting the above renaming requirements will be graded 0 pts.