QBUS6850 Assignment 1:

Due dates: Monday 3 September 2018

Value: 10%

Notes to Students

1. The assignment MUST be submitted electronically to Turnitin through QBUS6850 Canvas site. Please do NOT submit a zipped file.
2. The assignment is due at **17:00pm on Monday, 3 September 2018**. The late penalty for the assignment is 10% of the assigned mark per day, starting after 17:00pm on the due date. The closing date Monday, 10 September 2018, 17:00pm is the last date on which an assessment will be accepted for marking.
3. Your answers shall be provided as a word-processed report giving full explanation and interpretation of any results you obtain. Output without explanation will receive **zero** marks.
4. Be warned that plagiarism between individuals is always obvious to the markers of the assignment and can be easily detected by Turnitin.
5. The data sets for this assignment can be downloaded from Canvas.
6. Presentation of the assignment is part of the assignment. Markers will reduce to 10% of the mark for poor writing in clarity and presentation. It is recommended that you should include your Python code as appendix to your report, however you may insert small section of your code into the report for better interpretation when necessary. Think about the best and most structured way to present your work, summarise the procedures implemented, support your results/findings and prove the originality of your work.
7. Numbers with decimals should be reported to the third decimal point.
8. The report should be NOT more than 10 pages including everything like text, figure, tables, small sections of inserted codes etc but excluding the appendix containing Python code.

Tasks

Question 1 (50 Marks)

You will work on the UCI ML housing dataset **https://archive.ics.uci.edu/ml/machine-learning-databases/housing/**. A template Python program has been prepared for you. The program can help you get the dataset from sklearn dataset repository. Please test and play with the template program to fully understand the dataset.

For further information, please visit
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.names.

(a) Suppose you are interested in using the house age AGE (proportion of owner-occupied units built prior to 1940) as the first feature x_1 and the full-value property-tax rate TAX as the second feature x_2 to predict the MEDV (median value of owner-occupied homes in $1000's) as the target t. Write code to extract

these two features and the target from the dataset.

Use the dataset (two chosen features and one target) to plot the loss function

$$L(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{n=1}^{N}(f(\mathbf{x}_n, \boldsymbol{\beta}) - t_n)^2 \qquad \text{with} \quad f(\mathbf{x}_n, \boldsymbol{\beta}) = \beta_1 x_1 + \beta_2 x_2$$

That is, we are using a linear regression model without the intercept term β_0.

Hint: This is a 3D plot and you will need to iterate over a range of β_1 and β_2 values.

(b) Use the linear regression model `LinearRegression` in the scikit-learn package to do two linear regression models to predict the target, with and without the intercept term. You may use 90% of the data as your training data, and the remaining 10% as your testing data. Compare the performance of two models and explain the importance of the intercept term.

Hint: The argument `fit_intercept` of the `LinearRegression` controls whether an intercept term is included in the model by `fit_intercept = True` or `fit_intercept = False`.

(c) Take 90% of data as training data. Construct the centred training dataset by conducting the following steps in your Python code:

(i) Take the mean of all the training target values, then deduct this mean from each training target value MEDV. Take the resulting target values as the new training target values $\mathbf{t}_{new}$;

(ii) In the training data, take the mean of all the first feature values AGE, then deduct this mean from each of first feature values. Take the result as the new first feature values $\mathbf{x}^{\mathbf{1}}_{new}$;

(iii) In the training data, do the same for the second feature TAX. The result is $\mathbf{x}^{\mathbf{2}}_{new}$;

Now build linear regressions with and without the intercept to fit to the new training data. Report and compare the coefficients and the intercept. Compare the performance of two models over the testing data. Note that, when you take your testing data into the model to calculate performance scores, you shall take the relevant training means from the testing features and targets.

(d) Consider the closed-form solution of the linear regression below, see slide 25 (the number may change) of Lecture 2,

$$\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\boldsymbol{X}^T\mathbf{t}$$

where $\mathbf{X}$ is the design (data) matrix whose first column is all 1s, and the first component in $\boldsymbol{\beta}$ is the intercept. Suppose that the data are centred (refer to (c)). Now prove that, in the case of centred data, the intercept β_0 in the solution above is zero.

Hint: You may need that following fact that

$$\begin{pmatrix} \mathbf{A} & 0 \\ 0 & \mathbf{B} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} & 0 \\ 0 & \mathbf{B}^{-1} \end{pmatrix}$$

where both matrices **A** and **B** are invertible.

Question 2 (50 Marks)

Use Logistic Regression to predict diagnosis of breast cancer patients on the Breast Cancer Wisconsin (Diagnostic) Dataset (`wdbc.data`). See Section About Datasets. This question aims to test your ability in programming in matrix operation for Logistic Regression.

(a) Write Python code to load the data into your program. For the target feature Diagnosis, change its literal M (malignant) to 0 and B (benign) to 1. Split the data into training and validation sets (80%, 20% split). Then define and train a logistic regression model by using scikit-learn's `LogisticRegression` model.

(b) Using the logistic regression model function below and the estimated parameters from your model, calculate the probability of sample ID 8510426 (20th sample) having a benign diagnosis.

$$f(\mathbf{x}_n, \boldsymbol{\beta}) = \frac{1}{1 + e^{-\mathbf{x}_n^T \boldsymbol{\beta}}}$$

(c) The objective of logistic regression is defined as, on slide 17 (the number may change) of Lecture 3,

$$L(\boldsymbol{\beta}) = -\frac{1}{N}\left[\sum_{n=1}^{N}\left(t_n \log\left(f\left(\mathbf{x}_n, \boldsymbol{\beta}\right)\right) + (1 - t_n)\log\left(1 - f\left(\mathbf{x}_n, \boldsymbol{\beta}\right)\right)\right)\right]$$

where both the parameter $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_d)^{\boldsymbol{T}}$ and sample $\mathbf{x}_n = (x_{n0}, x_{n1}, \ldots, x_{nd})^T$ are d+1 dimensional vectors, where the intercept feature $x_{n0} = 1$. For Wisconsin Dataset d = 30. It is easy to prove that (you don't need to prove this)

$$\frac{\partial L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \frac{1}{N}\mathbf{X}^T(\mathbf{f}(\mathbf{X}, \boldsymbol{\beta}) - \mathbf{t})$$

where $\mathbf{f}(\mathbf{X}, \boldsymbol{\beta}) = \left(f(\mathbf{x}_1, \boldsymbol{\beta}), f(\mathbf{x}_2, \boldsymbol{\beta}), \ldots, f(\mathbf{x}_N, \boldsymbol{\beta})\right)^T$ and $t = (t_1, t_2, \ldots, t_N)^T$.

Write your own python code to use this derivative formula to implement the gradient descent algorithm for the logistic regression. You may write a python function named such as `myLogisticGD`, which accepts an data matrix `X`, an initial parameter `beta_0`, and a number of GD iterations `T` and other arguments you see appropriate. Your function should return the learned parameter $\boldsymbol{\beta}$.

Hint: In python, you can use the following way to get the vector $\mathbf{F} = \mathbf{f}(\mathbf{X}, \boldsymbol{\beta})$. First define the sigmoid function by

```
def sigmoid(x):
    return (1 / (1 + np.exp(-x)))
```

then

```
F = sigmoid(np.dot(X, beta))
```

or similar.

(d) Based on task (c) and the training data used in (a), write python code to use different initial values $\boldsymbol{\beta} = (0,\ 0, \ldots, 0)^{\boldsymbol{T}}$, $\boldsymbol{\beta} = (1,\ 1, \ldots, 1)^{\boldsymbol{T}}$, and a random initial $\boldsymbol{\beta}$ to start the gradient descent algorithm to minimise the objective of logistic regression with respect to the parameter $\boldsymbol{\beta}$. You set the number of iteration `T=200`. Use each resulting $\boldsymbol{\beta}$ to re-do task (b). Compare the results and explain the major reasons why you may have different answers with different initial value for $\boldsymbol{\beta}$.

Hint: As mentioned on slide 29 of Lecture 2, it is a good practice to normalize your data before you send them to your algorithm.

About Datasets

Breast Cancer Wisconsin (Diagnostic): wdbc.data

Attribute information

1: ID number

2: Diagnosis (M = malignant, B = benign)

3-32: Ten real-valued features are computed for two cell nuclei:

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

For more information, please refer
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.names