University of Lincoln
School of Computer Science
2019 – 2020

Assessment Item 1 of 2 Briefing Document

Title: CMP3751M Machine Learning	**Indicative Weighting: 50%**

Learning Outcomes

On successful completion of this component a student will have demonstrated competence in the following areas:

- LO1 Critique and appraise the scope and limits of machine learning methods by identifying their strengths and weaknesses

Task 1 (50%)

In this task we have to solve a polynomial regression example. Download the dataset pol_regression.zip. It contains a simple 1-dimensional dataset with inputs x (1 input dimension) and output y (1 output dimension). The dataset has been generated from an unknown polynomial function plus an additional noise term. In this task, we want to analyse the performance of polynomial regression and use it to estimate the degree of the polynomial as well as the parameters. To achieve the task, you will need to implement polynomial regression and evaluate its performance on the training as well on an independent test set. You will also have to analyse these performance metrics and discuss the relation of the degree of the polynomial to over- and underfitting.

Task 1 Report Guidance

Your report must conform to the below structure and include the required content as described, information on specific marking criteria for each section is available in the accompanying CRG document. You must supply a written report containing **three distinct sections** that provide a full and reflective account of the processes undertaken. You will be asked to provide several figures that demonstrate the performance of the algorithm. For all figures, please provide all necessary code snippets (including basic documentation) such that your results are reproduce-able. All programming exercises of this assignment must be implemented in **python**. You are allowed to use the **pandas library** for data-management, however, **scikit learn or other pre-built libraries such as sklearn.preprocessing.PolynomialFeatures** are **prohibited** in this assignment and will lead to a significant reduction of points. The regression algorithm needs to be implemented by yourself using numpy matrix operations only. Do not use pre-build functions to perform the regression. You are allowed to use the numpy.linalg.solve function. You can reuse material from the workshops.

Section 1.1: Description of Polynomial Regression (16%)

Using your own words and the lecture material, explain polynomial regression. Your description should cover the following points:

- Error function used for regression
- Linear regression models
- Least squares solution
- Polynomial feature expansions

- How to use polynomial features in linear regression
- Difference between training set and test set performance

Section 1.2: Implementation of Polynomial Regression (20%)

In this section, you are asked to implement the polynomial regression algorithm. To do so, you are required to implement the following function:

```
def pol_regression(features_train,  y_train, degree):
    # code comes here
    parameters = ...
    return parameters
```

The function pol_regression takes the training input and output values as input (both should be given as 1D numpy array with number of data-points elements). In addition, the last argument defines the degree of the polynomial that we want to fit. It should return the parameters of the polynomial (note that for a polynomial of n-th degree, we have n + 1 parameters); parameters should be a 1-D numpy array. You need to implement the least squares solution using a polynomial feature expansion in order to obtain the parameters using numpy matrix operations.

After implementing the pol_regression function, use this function to regress the data set given in the .zip file as part of this assignment. Regress a polynomial of the following degrees: 0, 1, 2, 3, 5, 10. Plot the resulting polynomial in the range of [-5, 5] for the inputs x. In addition, plot the training points. Interpret and elaborate on your results. Which polynomial would you choose?

Hint:
Try to avoid direct matrix inversions by using the numpy.linalg.solve function.

Section 1.3: Evaluation (14%)

Using the pol_regression function from the last section, you need now to evaluate the performance of your algorithm. To do so, you first need to implement the following function:

```
def eval_pol_regression(parameters, x, y, degree):
   #your brilliant code comes here
    rmse = ...
          return rmse
```

The function takes the parameters computed by the pol_regression function and evaluates its performance on the dataset given by the input values x and output values y (again 1D numpy arrays). The last argument of the function again specifies the degree of the polynomial. In this function, you need to compute the *root mean squared error* (rmse) of the polynomial given by the parameters vector.
Now you again need to train your polynomial functions with degrees 0, 1, 2, 3, 5 and 10. However, this time, split the dataset provided (same as before) into 70% train and 30% test set. Evaluate the training set rmse and the test set rmse of all the given degrees. Plot both rmse values using the degree of the polynomial as x-axis of the plot. Interpret your results. Which degree would you now choose? Are there any degrees of the polynomials where you can clearly identify over and underfitting? Explain your conclusions!

Task 2 (50%)

In this task, several data for specific dog breeds have been collected. The data include four features, which are height, tail length, leg length, and nose circumference for each dog breed. The purpose of this experiment is to figure out whether these features could be used to cluster the dog breeds into three groups/clusters. Therefore, you are asked to make a clustering analysis over the data, which will help the animal scientists to understand their experimental results.

You need to write a short report to discuss how you complete the task (see Report Guidance) and go into enough depth to demonstrate knowledge and critical understanding of the relevant processes involved.

Task 2 Report Guidance

Your report must conform to the below structure and include the required content as described. Information on specific marking criteria for each section is available in the accompanying CRG document. You must supply a written report containing **two distinct sections** that provide a full and reflective account of the processes undertaken.

Section 2.1: Description of the K-Means Clustering (20%)

Using your own words and the lecture material, explain what is the (simple) K-Means clustering. Your description should include the following points:

- Objective function
- Centroids
- Euclidean distance
- Assignment step
- Update step
- Strengths and weaknesses of the K-Means clustering

Section 2.2: Implementation of the K-Means Clustering (30%)

You need to implement the (simple) K-Means Clustering algorithm by creating the following functions:

```
def compute_euclidean_distance(vec_1, vec_2):
    # your code comes here
    return distance

def initialise_centroids(dataset, k=2,3):
    # your code comes here
    return centroids

def kmeans(dataset, k=2,3):
    # your code comes here
    return centroids, cluster_assigned
```

- The function compute_euclidean_distance() calculates the distance of two vectors, e.g., Euclidean distance

- The function initialise_centroids() randomly initializes the centroids
- The function kmeans() clusters the data into k groups

After implementing the kmeans() function, use this function to cluster data. The four features are used as the input to your algorithm. The parameter k is set to be 2 and 3. That means that after you have developed your functions, you need to use two different values of k.

Draw three sub-plots for the K-Means clustering:

- The first plot is a scatter plot, where x axis is "height" and y axis is "tail length"; data points should have different colours.
- The second plot is a scatter plot, where x axis is "height" and y axis is "leg length"; data points should have different colours.
- The third plot is the line plot, where x axis is iteration step and y axis is objective function. The plot should show a decreasing trend!

Please generate the above requested plots for both values of k, i.e. two plots for k=2 and another two for k=3.

Note: You need to implement the simple K-Means clustering following the above steps by yourself! You are allowed to use numpy, pandas and matplotlib libraries for processing and plotting data. However, other data science and machine learning libraries such as scikit-learn or other built-in libraries that have implemented the k-means algorithm are **prohibited** in this coursework.

<u>Tasks 1 and 2 Submission Instructions</u>

The report should be a **maximum of 20 pages (including everything!).** Keep in mind that:

- The report must contain your name, student number, module name
- The report must be in PDF
- The report must be formatted in single line spacing and use an 11pt font
- The report does not include this briefing document
- You describe and justify each step that is needed to reproduce your results by using code-snippets, screenshots and plots

The deadline for submission of this work is included in the School Submission dates on Blackboard. You must make an electronic submission of your work to **Blackboard** the **Turnitin upload area** for assessment item 1

This assessment is an individually assessed component. Your work must be presented according to the School of Computer Science guidelines for the presentation of assessed written work. Please make sure you have a clear understanding of the grading principles for this component as detailed in the accompanying Criterion Reference Grid. If you are unsure about any aspect of this assessment item, please seek the advice of the delivery team.