|  |  |
| --- | --- |
| **University of Lincoln**  **School of Computer Science**  **2018 – 2019** | |
| **Assessment Item 1 of 2 Briefing Document** | |
| **Title: CMP3744M Algorithms for Data Mining** | **Indicative Weighting: 50%** |
| Learning Outcomes  On successful completion of this component a student will have demonstrated competence in the following areas:   * LO1 Critically apply the mathematical foundations of data mining algorithms * LO2 Appraise a range of data mining algorithms, including the identification of their strengths and weaknesses | |
| **Task 1 (25%)**  In this task we have to solve a ridge regression example. Download the dataset ridge\_regression.zip. It contains a simple 1-dimensional dataset with inputs x (1 input dimension, used for plotting), features of x, and output y (1 output dimension). The dataset has been generated from an unknown ridge function plus an additional noise term. We have 12 different features (‘features0’ - ‘features11’) that define the features space of the linear regression. In this task, we want to analyse the performance of ridge regression in the given feature space and estimate the optimal regularisation parameter. To achieve the task, you will need to implement ridge 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 choice of the regularisation parameter and its relation to under- and over-fitting. | |
| **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 have to be implemented in python. You are allowed to use the pandas library for data-management, however, scikit learn or other pre-built libraries 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 as well. You can reuse material from the workshops.  **Section 1.1: Description of Ridge Regression (8%)**  Using your own words and the lecture material, explain ridge regression. Your description should cover the following points:   * MSE and SSE * Linear regression models * Least squares solution * Use of features in linear regression * Overfitting and Underfitting * Intuition of the weight penalty term * Objective function of ridge regression   **Section 1.2: Implementation of Ridge Regression (10%)**  In this section, you are asked to implement the ridge regression algorithm. To do so, you are required to implement the following function:  def ridge\_regression(features\_train, y\_train, regularisationFactor):  # your brilliant code comes here  parameters = ...  return parameters  The function *ridge\_regression* takes the training input features and output values as input (both should be given as numpy array with number of data-points rows. *features\_train* has 12 columns while *y\_train* only has 1). In addition, the last argument defines the regularisation factor used for ridge regression. The function should return the parameters of function which can be thought of as weightings for each feature (note that we have 12 parameters in this case). *parameters* should be a 1-D numpy array. You need to implement the least squares solution using the ridge term to obtain *parameters* using numpy matrix operations.  After implementing the *ridge\_regression* function, use this function to regress the training data set given in the ridge\_regression.zip file. Regress the function using the following regularisation factors: [10-6, 10-4, 10-2, 10-1]. Plot the resulting function in the range of [-5, 5]. Use the plotting dataset provided in the zip file to plot the learned function (multiply the parameter vector with the features from the test set to get the output values. Use the “x” column in the dataset for the x-axis of the plot. Limit the y-axis in the plot from –1000 to 1000 (using matplotlib.pyplot.ylim). In addition, plot the training points. Interpret your results. Which regularisation factor would you choose?  **Hint:**  Try to avoid direct matrix inversions by using the numpy.linalg.solve function.  **Section 1.3: Evaluation (7%)**  Using the ridge\_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\_regression(parameters, features, y):  # your brilliant code comes here  rmse = ...  return rmse  The function takes the parameters computed by the ridge\_regression function and evaluates its performance on the dataset given by the features vector and output values y (again numpy arrays). In this function, you need to compute the *root mean squared error* (rmse) of the function given by the parameters vector.  a) Now you again need to train your function with regularisation parameters [10-6, 10-4, 10-2, 10-1, 100, 101, 102, 103]. However, this time, split the dataset into 70% train and 30% test set. Evaluate the training set’s rmse performance and do the same for the test set’s rmse as well, of all the given degrees. Plot both rmse values using the regularisation factor as x-axis of the plot. Use a logarithmic scaling for the x as well as the y axis (using matplotlib.pyplot.loglog). Interpret your results. Which regularisation factor would you now choose? Are there any values of the regularisation factor where you can clearly identify over- and under- fitting? Explain and elaborate on your conclusions!  b) Repeat the previous experiment 10 times with different splits of training and test set by randomising the order of the data points. Compute the average of the 10 rmse values for the training and test set for each degree given above and plot again the average train and test rmse. Would you now come to a different conclusion? Explain the potential differences in the plots you have produced for a) and b). | |
| **Task 2**  In this task, several data for a specific plant have been collected. The data include four features, which are stem length, stem diameter, leaf length, and leaf width (in cm) of this plant. The purpose of this experiment is to figure out whether these features could be used to cluster the plants into 3 groups. Therefore, you are asked to make a clustering analysis over the data, which will help the plant biologists 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 sufficient 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 (10%)**  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 (15%)**  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=3,4):  # your code comes here  return centroids  def kmeans(dataset, k=3,4):  # 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 3 and 4. That means that after you have developed your functions, you need to use two different values of k.  Create two plots for the K-Means clustering:   * The first plot is a scatter plot, where x axis is `stem\_length’ and y axis is `stem\_diameter’, and the data points should have different colors for differentiating different clusters. * The second plot is the line plot, where x axis is iteration step and y axis is the objective function.   Please generate the above requested plots for both values of k, i.e. two plots for k=3 and another two for k=4.  **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. | |
| **Tasks 1 and 2 Submission Instructions**  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 Georgios Leontidis [GLeontidis@lincoln.ac.uk](mailto:GLeontidis@lincoln.ac.uk) and Mingjun Zhong [MZhong@lincoln.ac.uk](mailto:MZhong@lincoln.ac.uk). | |