CS4100/CS5100 Coursework 2

This coursework must be submitted by **Tuesday, 29 November 2022, at 10 am**. Feedback will be possibly provided by Friday, 23 December 2022.

How to submit

To submit your work, follow the steps below:

1. Start with the empty Jupyter notebook template `dataAnalysisCW2.ipynb` we provide in Moodle (click here to download the template).
2. Write all your programs/solutions in the Jupyter notebook.
3. Upload the Jupyter notebook to Moodle via this link.
4. All submissions will be graded anonymously. Your name and student ID must not appear in anywhere (including filename) of your file.

Failure to follow the above submission steps will receive a penalty of 10 marks.

More Instructions

This coursework is divided into two parts: theory and practical. The theory part takes up 40%. It includes questions testing your understanding on the theoretic foundations of data analysis. The practical part takes up 60%. It includes tasks to test your ability to implement data analysis algorithms in Python.

When writing in a Jupyter notebook, the programs/codes should be put in cells with "code mode", while all other things should be put in cells with "markdown mode". Revise lab handout 1 to understand what "code mode" and "markdown mode" are. Click here for a webpage which contains useful information about writing in markdown mode.

All your programs should be written from scratch. **You are not allowed to use any existing implementation of Agglomerative Hierarchical Clustering. The only library packages you can import are `random` and `matplotlib`.**

> **Note:** All the work you submit should be solely your own work. Coursework submissions are routinely checked for this.

You can overwrite your submission in Moodle as often as you like. Only the last version submitted will be marked. Submissions uploaded after the deadline may be accepted but will be automatically recorded as *late*.

Learning outcomes assessed

- The ability to explain the theoretic foundations in data analysis.
- The ability to perform simple calculations that appear in data analysis.
- The ability to develop data analysis algorithm, and to present its outputs.

Marking criteria

To be awarded full marks, you need to i) write concise and correct answers to the questions in the theory part; ii) write working and well-written codes for all tasks in the practical part; iii) obtain good numerical results, present them coherently, and show how they complete the tasks.

If your results are not completely correct, we will evaluate your work and partial marks may be awarded. The overall coursework marks will be announced as letter grade (A, B, C, D, E or F).

1 Theory Part (40 Marks)

Answer the questions below using the Jupyter notebook template. Follow the instructions in the template to write your answers in the appropriate cells. You are expected to provide clear explanations for all your answers. In particular, you must show the steps of all calculations you have made.

Question 1 (6 marks)

Consider a shallow neural network f with 2 inputs, 3 hidden nodes, one output, and reLu activation functions for all the hidden neurons. Assume that the parameters of the neural network are

$$\lambda^\circ = [0, -1, 1, 0]$$
$$\lambda^h = [[-1, 1, 0], [1, 0, -1], [0, 1, -1]],$$

and the input is $x = [-1, 1]$. Let $z_i^h = \lambda_{i,0}^h + \sum_{j=1}^2 \lambda_{i,j}^h x_j$, for $i = 1, 2, 3$, be the value of the hidden neurons. Let $z^\circ = \lambda_0^\circ + \sum_{i=1}^3 \lambda_i^\circ \cdot \mathsf{reLu}(z_i^h)$ be the value of the output neuron. How do you choose the activation function of the output neuron to use the network for completing a binary classification task? Compute the label predicted by the network if we assume the labels are $\mathcal{Y} = \{\text{yes}, \text{no}\}$, and $f(x, (\lambda^\circ, \lambda^h)) = \text{Prob}(Y = \text{yes}|x)$.

Question 2 (6 marks)

We can augment a linear regression model by adding pre-defined functions of the object attributes. For example, let the original object attributes are $X_1, X_2, \ldots X_d$ and consider an augmented model with input

$$X_{\text{aug}} = \left[X_1, X_2, \ldots, X_d, \log(X_1), \log(X_2), \ldots, \log(X_d), (X_1)^2, (X_2)^2, \ldots, (X_d)^2\right].$$

What is the dimensionality of the parameter of the following augmented model?

$$f(X_{\text{aug}}, \lambda_{\text{aug}}) = \lambda_0 + \lambda_{\text{aug},1} X_{\text{aug},1} + \lambda_{\text{aug},2} X_{\text{aug},2} + \cdots$$

How do you expect the augmentation to affect the model's flexibility? Discuss the expected bias and variance of the augmented model with respect to the bias and variance of a standard linear regression model $f(X, \lambda) = \lambda_0 + \lambda_1 X_1 + \cdots + \lambda_d X_d$.

Question 3 (8 marks)

Compute the optimal binary split of the following dataset $\{(x_n, y_n) \in \mathbb{R} \times \{0, 1, 2\}\}_{i=1,2,\ldots,5}$, where each y_n is a qualitative/categorical label.

$$[(1.2, 0),\ (2, 1),\ (0.1, 2),\ (2.1, 0),\ (-0.1, 1)].$$

You can define the information gain of the split in terms of either the Gini index or the entropy function. Compute the label predicted by the decision stump associated with the optimal binary split on input $x = 1.1$.

Question 4 (10 marks)

(a) In a data analysis problem, there are p attributes, where p is a random number generated by a code snippet in the Jupyter notebook. The label of each datapoint is either "cat", "dog" or "fox". If we use random forest for the problem, in each node of each decision tree, how many attributes should be sampled? (3 marks)

(b) Recall that in the standard bagging procedure, each BTS is generated by sampling n datapoints with replacement, where n is the number of datapoints in the original dataset. In this part, we consider a variant of bagging, in which each BTS is generated by sampling $2n$ datapoints with replacement instead. For a particular datapoint, what is the formula for computing the probability that the datapoint is sampled in a BTS? Compute the probability to 4 decimal places for $n = 500$. (4 marks)

(c) Although boosting is a general ensemble method technique for regression problems, it is ineffective when used with the KNN model. Why? (3 marks)

Question 5 (10 marks)

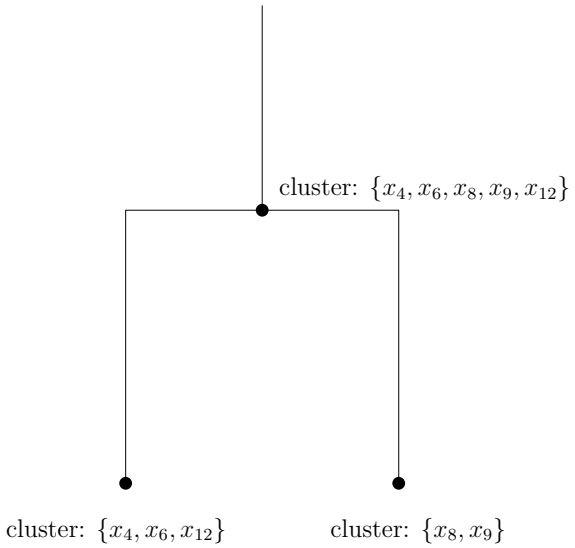
(a) In the Jupyter notebook, there is a code snippet which generates a dataset with 7 random datapoints in $\mathbb{R}^2$, and puts the datapoints into three clusters initially. Run the K-Means algorithm on this dataset for two rounds. What are the cluster assignments after the first round and after the second round? (6 marks)

(b) It is found that when Agglomerative Hierarchical Clustering (AHC) is used with single linkage, the clusters produced are often *biased*. What is meant by "biased"? Also, explain why biased clusters are often produced. (4 marks)

2 Practical Part (60 Marks)

In this part, you will be guided to implement Agglomerative Hierarchical Clustering (AHC) using Object Oriented Programming (OOP). All your programs should be written from scratch. **You are not allowed to use any existing implementation of Agglomerative Hierarchical Clustering. The only library packages you can import are `random` and `matplotlib`.**

Recall that the output of AHC can be visualized as a tree, where each node corresponds to a cluster. An internal node has two children nodes; the cluster of the internal node is the union of the clusters of the two children nodes. See the picture below for an example.



In our OOP implementation, `Node` is a class. Each node in the AHC tree is an object, and it has the following four variables at least:

- `cluster`: this variable saves the datapoints in the cluster as a list;
- `height`: this variable saves the height of the node in the AHC tree;
- `leftChild`: this variable saves the left child of this node; if the node is a leaf and it has no left child, `leftChild` is set to `None`;
- `rightChild`: this variable saves the right child of this node; if the node is a leaf and it has no right child, `rightChild` is set to `None`.

You are welcome to add more variables to this class to help yourself completing the tasks below.

Task 1: Generating synthetic data (3 marks)

Use what you learn in the lab handouts to generate a list of 100 random datapoints in $\mathbb{R}^3$, where the numbers in each datapoint are integers drawn uniformly between -50 and 50 (inclusive). Save the list under the variable name `datapoints`. The random seed you should use is your birthday: for instance, if you were born on 31 August, the random seed you should use is 831; and if you were born on 4 June, the random seed you should use is 604.

Task 2: `distance` function (3 marks)

Implement a function `distance` which takes two points in $\mathbb{R}^3$ as input, and it outputs the Euclidean distance between the two points.

Hint: to find the square root of a number y in Python, you can use the command `pow(y,0.5)`.

Task 3: `Node` class, and initialization of AHC (4 marks)

Implement the `Node` class specified above. To initalize AHC, for each point x in `datapoints`, initialize a `Node` object, in which `cluster` is a list which contains only the point x, `height` is set to be 0, while Save all these nodes in a list under the variable name `currentClusters`.

Task 4: Linkage functions (14 marks)

Implement a function `singleLinkage` which takes two `Node` objects as input, and it outputs the single linkage between the two `clusters` in the two `Node` objects.

Then implement `completeLinkage`, `averageLinkage` and `centroidLinkage` accordingly.

Task 5: Deciding which two clusters to merge (10 marks)

Implement a function `clustersToMerge`, which takes `currentClusters` and a linkage function you have implemented in Task 4 as input, and decides which two nodes in `currentClusters` should be merged. The output of this function should be a tuple (i, j, h), which indicates that the i-th node and the j-th node in `currentClusters` should be merged next to form a new Node, whose height is h.

Task 6: AHC algorithm (16 marks)

Implement a function `ahc`, which takes `datapoints` created in Task 1 and a linkage function you have implemented in Task 4 as input, and it outputs the

root node of the AHC tree. The functions you have implemented in the previous tasks will help you a lot to implement `ahc` efficiently.

When you merge two nodes, you should create a new `Node` object, and set its `cluster`, `height`, `leftChild` and `rightChild` appropriately.

Task 7: Cutting the AHC tree, and plot the datapoints in 3D (10 marks)

Implement a function `cutTree`, which takes the root node output by `ahc`, and an integer k as input, and it outputs the k clusters obtained by cutting the AHC tree at an appropriate height.

Then use the `cutTree` function and the `matplotlib` package to plot the datapoints generated in Task 1, and color them according to the clusters they belong to, for $k = 5$. Do this for different linkage functions, and point out your observations.

Hint: to make 3D plot using `matplotlib`, the tutorial provided by the official webpage (click here!) is useful.