**CSCI 4450/8456 Introduction to Artificial Intelligence**

**Instructor: Dasgupta**

**Homework 3**

**Full Points: 80**

Given on October 25, 2018 (Thursday)

Due on November 8, 2018 (Thursday) 11:59 PM (Problem 1 via Canvas, Problems 2-4 via loki)

Loki submission directory CSCI-4450-DG-F18-A3. As usual, include a README text file with instructions on how to compile and run your code for problems 2 through 4.

**Problem 1. K-means Calculations (by hand, calculator)** **(10 points)**

For the medicine data set discussed in class, use K-means with **the Manhattan distance metric** for clustering analysis by setting *K*=2 and initializing seeds as C1 = A and C2 = C. You should show the steps for the calculations made by the K-means algorithm to get full points. Then, answer the three questions below. Submit your answers as an electronically written file in Word or pdf format, via Canvas.

1. How many steps were required for convergence?
2. What are the memberships of two clusters after convergence?
3. What are the centroids (coordinates) of two clusters after convergence?

**Problem 2. K-means Programming (10 points)**

Our objective in this question is to verify the effect of choosing the initial number of clusters, given by parameter K, in the K-means algorithm. For this, first implement the K-means algorithm. To verify the effect of K, download the crime data set available on Canvas. Each data instance has four attributes: murder, assault, urban population and rape, and an integer label between 1 and 4. The name of the state is just an identifier, and can be ignored for clustering purposes. Make sure you store the crime dataset csv file in your submission folder. Your program should load the dataset from this locally stored csv file.

The k-values you should use are: k = 2, 3, 4, 5, and 6.

Your program should output the values of ‘k’, the distortion value and the number of iterations taken to run. A sample output is below:

K = 2, distortion = …, iterations = …

K = 3, distortion = …, iterations = …

**Problem 3. K-medoids Programming (30 points)**

In class, we discussed the Partition Around Medoids (PAM) algorithm. In this question, you will implement the PAM algorithm and report its output as a confusion matrix (see below) for different datasets.

You will use the iris dataset available from UCI Data Repository for testing your algorithm. The dataset is available at <https://archive.ics.uci.edu/ml/datasets/iris>. Each data instance contains four attributes or features and a label corresponding to the species of the flower. The attributes are 1) sepal length in cm, 2) sepal width in cm 3) petal length in cm, and 4) petal width in cm. The set of labels is *Iris Setosa*, *Iris Versicolour* and *Iris Virginica*.

The K-medoids algorithm that you will implement is adapted from Park and Jun’s 2009 paper ‘A simple and fast algorithm for K-medoids clustering’, as outlined below:

1. Select Initial Medoids
   1. Let *dij* = Euclidean distance between object i and object j. Let *D* denote the sum of Euclidean distances for all pairs of objects in the dataset.
   2. Calculate the measure *vj* for object j as ,where *j = 1…n*
   3. Sort *vj* -s in ascending order. Select *k* objects having the first *k* smallest values as initial medoids.
   4. Obtain the initial cluster result by assigning each object to the nearest medoid.
   5. Calculate the sum of distanced from all objects to heir medoids.
2. Update Medoids
   1. Find a new medoid of each cluster, which is the object minimizing the total distance to other objects in its cluster. Update the current medoid in each cluster by replacing with the new medoid.
3. Assign Objects to Medoids
   1. Assign each object to the nearest medoid and obtain the cluster result.
   2. Calculate the sum of distance from all objects to their medoids. If the sum is equal to the previous one, then stop the algorithm. Otherwise, go back to Step II.

To test the output of your algorithm with the iris dataset, you will use the confusion matrix. The confusion matrix of a prediction task with a dataset having *n* classes is a *n X n* matrix, where the columns represent the actual label and rows represent the predicted label. As you can guess, the diagonal entries represent number of correct classifications. An example taken from Wikipedia entry of confusion matrix is given below:

A classification system has been trained to distinguish between cats, dogs and rabbits, a confusion matrix summarizes the results of testing the algorithm. Assuming a sample of 27 animals — 8 cats, 6 dogs, and 13 rabbits, the resulting confusion matrix could look like the table below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | | Actual class | | | | **Cat** | **Dog** | **Rabbit** | | Predicted class | **Cat** | **5** | 2 | 0 | | **Dog** | 3 | **3** | 2 | | **Rabbit** | 0 | 1 | **11** | | In this confusion matrix, of the 8 actual cats, the system predicted that three were dogs, while for the six actual dogs, it predicted that one was a rabbit and two were cats. …All correct predictions are located in the diagonal of the table (highlighted in bold), so it is easy to visually inspect the table for prediction errors, as they will be represented by values outside the diagonal. |

For testing your K-medoids algorithm, once your algorithm has terminated and clusters determined, write a function or method that looks up the actual or ground truth label of each cluster’s center object from the dataset. Then, give the same label as the cluster center object to every object in the same cluster. These labels give the labels predicted by K-medoids. Now write a function or method that compares the predicted label with the ground truth label for each object and determines the confusion matrix. For the iris dataset the confusion matrix should be a 3 X 3 matrix as there are three classes or labels.

Finally, calculate the confusion matrix again using K-means algorithm implemented in Question 2, using K=3.

The final output of your program should be the confusion matrices calculated by K-means, followed by K-medoids algorithms.

Make sure you store the iris dataset csv file in your submission folder. Your program should load the dataset from your locally stored csv file.

**Problem 4. Markov Decision Process (MDP)**

(Adapted from Russell-Norvig Problem 17.8) **(30 points = 15 points each part)**

In class, we studied that one way to solve the Bellman update equation in MDPs is using the Value iteration algorithm. (Figure 17.4 of textbook).

1. Implement the ***value iteration*** algorithm to calculate the policy for navigating a robot (agent) with uncertain motion in a rectangular grid, similar to the situation discussed in class, from Section 17.1 of the textbook.
2. Calculate the same robot’s policy in the same environment, this time using the ***policy iteration*** algorithm.

You can combine these two parts into the same class or program and have the user input select the appropriate algorithm.

Your program should create the 3 x 3 grid world given in Figure 17.14 (a) of the textbook along with the corresponding rewards at each state (cell). (1, 1) should correspond to the bottom left corner cell of your environment. The coordinates of a cell should follow the convention (col number, row number).

The transition model for your agent is the same as that given in Section 17.1 (discussed in class) – 80% of the time it goes in the intended direction, 20% of the time it goes at right angles to its intended direction. You should accept the following values of r as input: 100, -3, 0 and +3. The input format is below:

Enter r: <value of r>

Enter 1 for Value Iteration, 2 for Policy Iteration, 3 to Exit: <1 or 2 or 3>

The output of your program should give the policy **and final utility** (remember that for a terminal cell (state), utility = reward) for each cell in the grid world calculated by your program(s). For value iteration, the policy at each state (cell) is calculated using the policy equation (Equation 17.4 of textbook). For policy iteration, the algorithm’s output is the policy for each state. **For values of constants used by value iteration, suggested values are epsilon = 10-3, gamma = 0.9.**

Output format:

Policy table calculated:

(1, 1): <action suggested by calculated policy>

(2, 1): <action suggested by calculated policy>

Utitlies:

(1, 1): <utility value>

(2, 1): <utility value>