2/2566 FRA501: Pattern Recognition

HW1 Clustering and Regression

Instructions

Answer the questions and upload your answers to google drive. Answers can be in Thai or English. Answers can be either typed or handwritten and scanned.

1. Clustering

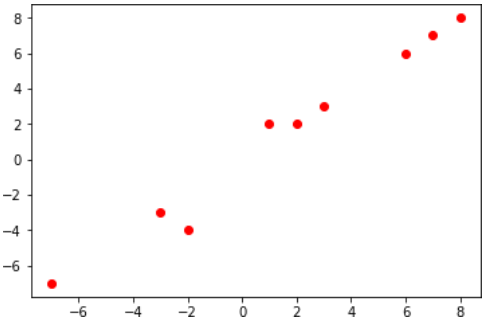
Recall from lecture that K-means has two main steps: the points assignment step, and the mean update step. After the initialization of the centroids, we assign each data point to a centroid. Then, each centroid is updated by re-estimating the means.

Concretely, if we are given N data points, x_1, x_2, ..., x_N, and we would like to form K clusters. We do the following

1. **Initialization**: Pick K random data points as K centroid locations c_1, c_2, ..., c_K.
2. **Assign**: For each data point k, find the closest centroid. Assign that data point to the centroid. The distance used is typically Euclidean distance.
3. **Update**: For each centroid, calculate the mean from the data points assigned to it.
4. **Repeat**: repeat step 2 and 3 until the centroids stop changing (convergence).

Given the following data points in x-y coordinates (2 dimensional)

x	y
1	2
3	3
2	2
8	8
6	6
7	7
-3	-3
-2	-4
-7	-7



1.1. If the starting points are (3,3), (2,2), and (-3,-3). Describe each assign and update

step. What are the points assigned? What are the updated centroids? You may do this calculation by hand or write a program to do it.

1.2. If the starting points are (-3,-3), (2,2), and (-7,-7), what happens?

1.3. Between the two-starting set of points in the previous two questions, which one do you think is better? How would you measure the 'goodness' quality of a set of starting points?

1.4. What would be the best K for this question? Describe your reasoning.

2. Regression

In this part of the exercise, we will work on the Titanic dataset provided by Kaggle. The Titanic dataset contains information of the passengers boarding the Titanic on its final voyage. We will work on predicting whether a given passenger will survive the trip. We start by importing the data using Pandas.

```
train_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
train = pd.read_csv(train_url) #training set

test_url =
"http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
test = pd.read_csv(test_url) #test set
```

Both train and test are dataframes. Use the function train.head() and train.tail() to explore the data. What do you see?

Use the function describe() to get a better understanding of the data. You can read the meaning of the data fields at https://www.kaggle.com/competitions/titanic/data.

Looking at the data, you will notice a lot of missing values. For example, some age is NaN. This is normal for real world data to have some missing values. There are several ways to handle missing values. The simplest is to throw away any rows that have missing values. However, this usually reduce the amount of training data you have. Another method is to guess what the missing value should be. The simplest guess is to use the Median or Mode of the data. For this exercise we will proceed with this.

2.1. What is the median age of the training set? You can easily modify the age in the dataframe by

```
train["Age"] = train["Age"].fillna(train["Age"].median())
```

Note that you need to modify the code above a bit to fill with mode() because mode() returns a series rather than a single value.

2.2. Some fields like 'Embarked' are categorical. They need to be converted to numbers first. We will represent S with 0, C with 1, and Q with 2. What is the mode of Embarked? Fill the missing values with the mode. You can set the value of Embarked easily with the following command.

```
train.loc[train["Embarked"] == "S", "Embarked"] = 0
```

Do the same for Sex.

2.3. Write a logistic regression classifier using gradient descent (from scratch, don't use scikit-learn framework or others) as learned in class (Survived). Use PClass, Sex, Age, and Embarked as input features. You can extract the features from Pandas to Numpy by

```
data = np.array(train[["PClass","Sex","Age","Embarked"]].values)
```

Check the datatype of each values in data, does it make sense? You can force the data to be of any datatype by using the command

```
data = np.array(train[["PClass","Sex","Age","Embarked"]].values, dtype = float)
```

2.4. When you evaluate the trained model on the test set, you will need to make a final decision. Since logistic regression outputs a score between 0 and 1, you will need to decide whether a score of 0.3 (or any other number) means the passenger survive or not. For now, we will say if the score is greater than or equal to 0.5, the passenger survives. If the score is lower than 0.5 the passenger will be dead. This process is often called 'Thresholding.' **Then, show your precision, recall, f1-score on the training set and test set.**

2.5. Try adding some higher order features to your training e.g. (x1^2, x1,x2,...). Does this model have better accuracy on the training set? How does it perform on the test set?

2.6. What happens if you reduce the number of features to just Sex and Age?

2.7. We want to show that matrix inversion yields the same answer as the gradient descent method. However, there is no closed form solution for logistic regression. Thus, we will use normal linear regression instead. Re-do the Titanic task as a regression problem by using linear regression. Use the gradient descent method. **Then, show your weights and loss values of training set and test set (Mean Squared Errors (MSE)).**

2.8. Now try using matrix inversion instead. However, are the weights learned from the two methods similar? Report the Mean Squared Errors (MSE) of the difference between the two weights (2.7 and 2.8).