IERG4320/IEMS5726 Data Science in Practice (Summer 2024)
Assignment 3
Expected time: 6 hours

Learning outcomes:
1. To understand the basic machine learning algorithms, including classification, clustering and regression. 2. To practise a simple ANN model.

Instructions:

1. Do your own work. You are welcome to discuss the problems with your fellow classmates. Sharing ideas is great, and do write your own explanations.
2. If you use help from the AI tools, e.g. ChatGPT, write clearly how much you obtain help from the AI tools. No marks will be taken away for using any AI tools with a clear declaration.
3. All work should be submitted onto the blackboard before the due date.
4. You are advised to submit a single zip/rar file containing the following items.
    a. A .pdf file containing the answers of the written part.
    b. A Assignment3_1155xxxxxx.py file storing all your programs for the five problems. (1155xxxxxx refers to your student ID)
5. Do type/write your work neatly. If we cannot read your work, we cannot grade your work.
6. If you do not put down your name, student ID in your submission (both .pdf and .py), you will receive a 10% mark penalty out of the assignment 3.
7. Due date: 29th April, 2023 (Monday) 23:59

Short questions (25%)
Answer all questions.

1. You are working as a senior data analyst in an online store company, having client information, goods information and history of purchase. You encounter the following situations.

    a. Based on the history of purchase and goods information, you wish to predict the marked price of the goods on the online store. What is the type of this machine learning problem? (2%) What metrics can be used in assessing the performance of this problem? Name and explain one metric. (2%)

    b. Name and explain three different types of data available for the goods information. (6%)

    c. Your junior analyst mentioned that he wishes to perform a 10-fold cross validation for predicting whether the client purchases the new good or not. He divides the dataset into 10 folds, trains a model based on 9 folds and calculates the accuracy based on the left-out fold. At the end, the 10-fold cross validation accuracy is the average of the 10 intermediate accuracy. Is he correct? Explain your answer. (6%)

d. Another junior analyst tries to perform the correlation analysis between the occupation of the clients and the types of purchased goods. If some good correlations are found, he can make good use of the correlations to build a recommender system. What is correlation analysis? (2%) What is a recommender system? (2%) Name two successful recommender systems in our life. (2%)

e. Another junior analyst tries to perform cluster algorithms on the client information. Suggest the major issue why assessing the performance of clustering algorithms is difficult? (1%) What are the two major considerations in assessing whether the cluster identified is a high-quality cluster or not. (2%)

Practical problems (75%)
Work on the following five problems in a single .py file.
If your Python program cannot be run, you will receive no scores for the problem.
Put down your student ID as a comment in your first line of the program.

2. Write a function that reports the testing precision, recall and f1-score of a classifier based on an artificial neural network. Also report the trained model. (15%)

   a. Create a list of NN models, based on the list of numbers in hidden.
   b. Study the data loader, then, implement the training loop using the provided loss_function and optimizer.
   c. Study the data loader in the training loop, use the data loader for the testing set.
   d. Calculate the metrics (precision, recall and f1-score) based on the testing set.

   Link to dataset:
   https://www.kaggle.com/datasets/nelgiriyewithana/credit-card-fraud-detection-dataset-2023
   Note: You can increase the max_epoch, and it will take a very long time to train.
   Note: The example model in this problem is too small and insufficient to solve this problem. If you really want to solve this problem using ANN, you need to use a much bigger one.

   Code:

```
# <Your student ID>
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import metrics
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
# Problem 2
# when you include batch size in the training process,
# you need to use dataloader
class MyDataset(Dataset):
    def __init__(self, X, y):
        self.X=torch.tensor(X.values, dtype=torch.float32)
        self.y=torch.tensor(y.values, dtype=torch.float32)

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        return self.X[idx], self.y[idx]

def problem_2(df, Xlabel, ylabel, hidden=[3,3,3], test_size=0.3,
batch_size=100, learning_rate=0.01, max_epochs=50000):
    # write your logic here, model is the trained ANN model
    model = []
    precision = 0
    recall = 0
    f1score = 0
    random_state = 4320  # default is 4320
    module = []

    # YOUR CODE HERE: create the list of modules


    # the output layer is fixed to 1 neural and sigmoid
    module.append(nn.Linear(hidden[-1],1))
    module.append(nn.Sigmoid())
    # create the nn model based on the list
    model = nn.Sequential(*module)

    # split the dataset
    X_train, X_test, y_train, y_test = train_test_split(df[Xlabel],
df[ylabel], test_size=test_size, random_state=random_state)
    # create the DataLoader for the training set
    mytrain = MyDataset(X_train, y_train)
    train_loader = DataLoader(mytrain, batch_size=batch_size,
shuffle=False)

    # use MSE loss function, and SGD optimizer
    loss_function = nn.MSELoss()
    optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

    # training loop
    for epoch in range(max_epochs):
        # you can uncomment the line below,
        # to visualize the slow training process
        # print("Debug: at epoch: ", epoch)
        for data, labels in train_loader:
            # YOUR CODE HERE: training loop

    # YOUR CODE HERE: follow the training set
    # to create dataloader for testing
    # then, calculate the metrics

    return model, precision, recall, f1score
```

Execution:

```
> df = pd.read_csv("creditcard_2023.csv")
> Xlabel =
["V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13"
,"V14","V15","V16","V17","V18","V19","V20","V21","V22","V23","V24","V
25","V26","V27","V28","Amount"]
> ylabel = ["Class"]
> model, p, r, f = problem_2(df, Xlabel, ylabel, hidden=[3,5,5,3],
max_epochs=100)
> print("model: ", model)
model:  Sequential(
  (0): Linear(in_features=29, out_features=3, bias=True)
  (1): ReLU()
  (2): Linear(in_features=3, out_features=5, bias=True)
  (3): ReLU()
  (4): Linear(in_features=5, out_features=5, bias=True)
  (5): ReLU()
  (6): Linear(in_features=5, out_features=3, bias=True)
  (7): ReLU()
  (8): Linear(in_features=3, out_features=1, bias=True)
  (9): Sigmoid()
)
> print("precision: ", p)
precision:  0.49938741653916724
> print("recall: ", r)
recall:  1.0
> print("f1-score: ", f)
f1-score:  0.6661219255685572
```

3. Write a function to process the SIFT keypoints and descriptors of an image based on its grayscale image. (15%)

Code:

```
# Problem 3
import cv2 as cv
import matplotlib.pyplot as plt
def problem_3(image_filename):
    # write your logic here, keypoint and descriptor are SIFT object
    keypoint = 0
    descriptor = 0


    return keypoint, descriptor
```

Execution:

```
> kp, des = problem_3("sample1.jpg")
> print(kp)
(< cv2.KeyPoint 000001ADA5A932D0>, < cv2.KeyPoint 000001ADA5A932A0>,
< cv2.KeyPoint 000001ADA1587210>, < cv2.KeyPoint
000001ADA88197A0>,...., < cv2.KeyPoint 000001ADA881A610>, <
cv2.KeyPoint 000001ADA881A640>)
> print(len(kp))
158
> print(des)
[[  0.  35. 145. ...   0.   0.  24.]
 [  1.  55.  20. ...   4.   0.   4.]
 [ 20.  77.   4. ...   0.  23.  16.]
 ...
 [ 51.   0.   0. ...   0.   0.   9.]
 [  9.   0.   4. ...   0.   0.   0.]
 [ 57.   2.   5. ...   0.   0.   2.]]
> print(des.shape)
(158, 128)
```

4. Write a function that runs the k-mean cluster algorithm on the descriptors. (15%)
   a. Based on the list of descriptors and the value of k, run the k-means algorithms on the descriptors.
   b. Return the k centers of the clusters (i.e. visual words).
   c. Use a random state as 5726 on the k-mean clustering for reproducibility of the results.
   d. You need to complete problem 3 before moving forward to this problem.

Code:

```
# Problem 4
def problem_4(descriptor_list, k=5):
    # write your logic here, visual_words are the cluster centers
    visual_words = 0


    return visual_words
```

Execution:

```
> kp, des = problem_3("sample1.jpg")
> visual_words = problem_4(des)
> print(visual_words)
[[ 15.807692     6.730768     5.6923122    21.961538    42.153843
   ...
    7.5769224    6.884615     9.653847   ]
 ...
 [ 18.142855    11.142856    16.085712    18.514282    13.799997
   ...
```

```
    0.742856     2.1714282   13.457143  ]]
> print(visual_words.shape)
(5, 128)
```

5. Write a function that converts the list of images to a dictionary of the visual bag of words based on the list of images. (15%)
   a. Based on the list of images, reuse problem 3 and collect the list of all descriptors
   b. Reuse problem 4, runs the k-means algorithm based on the value of k.
   c. Construct the visual bag of words using a dictionary, in which key is the file name of the image, value is the number of the descriptors falling in the k clusters (i.e. counts).
   d. You need to complete both problem 3 and problem 4 before moving forward to this problem.

Note: This problem is challenging
Hint: You may take a look:
https://medium.com/@aybukeyalcinerr/bag-of-visual-words-bovw-db9500331b2f

Code:

```
# Problem 5
# Takes 2 parameters. The first one is a dictionary that holds the
# descriptors that are separated class by class
# And the second parameter is an array that holds the central points
# (visual words) of the k means clustering
# Returns a dictionary that holds the histograms for each images that
# are separated class by class.
def image_class(all_bovw, centers):
    dict_feature = {}
    for key,value in all_bovw.items():
        category = []
        for img in value:
            histogram = np.zeros(len(centers))
            for each_feature in img:
                ind = find_index(each_feature, centers)
                histogram[ind] += 1
            category.append(histogram)
        dict_feature[key] = category
    return dict_feature

def find_index(each_feature, centers):
    best = 0
    dist = np.linalg.norm(each_feature-centers[0])
    for k in range(len(centers)):
        if np.linalg.norm(each_feature-centers[k]) < dist:
            best = k
            dist = np.linalg.norm(each_feature-centers[k])
    return best

def problem_5(list_of_image, k=5):
    # write your logic here, visual_bag_of_words is a dictionary
    sift_vectors = {}
    descriptor_list = []
    visual_bag_of_words = {}

    # YOUR CODE HERE:
    # reuse problem 3 and collect the list of all descriptors

    # YOUR CODE HERE:
    # reuse problem 4 and obtain the visual words

    # YOUR CODE HERE:
    # based on the visual words and cluster centers
    # construct the visual bow (dict)


    return visual_bag_of_words
```

Execution:

```
> list_of_image = ["sample1.jpg", "sample2.jpg", "sample3.jpg",
"sample4.jpg", "sample5.jpg"]
> vbow = problem_5(list_of_image, 10)
> print(vbow)
{'sample1.jpg': [array([ 4., 32., 26.,  4.,  3.,  3., 18., 53.,  4.,
11.])], 'sample2.jpg': [array([ 32., 126.,  28.,  35.,  47.,  14.,
32.,  40.,  21.,  69.])], 'sample3.jpg': [array([27., 53., 40., 24.,
33., 36., 36., 87., 12., 84.])], 'sample4.jpg': [array([15., 30.,
8.,  6., 10.,  3., 58., 44., 16., 12.])], 'sample5.jpg':
[array([201., 195., 221., 227., 211., 283.,  73., 263., 201.,
309.])]}
```

6. Write a function to perform k-fold cross validation using Ridge regression. Report cross validation RMSE and overall trained Ridge model. (15%)
   a. Separate the dataset to k folds.
   b. Repeat the training for k fold times, using the (k-1) folds to train the Ridge regression model, and accumulating the squared error of the left-out fold.
   c. Calculated the RMSE based on the accumulated squared error.
   d. Re-train the Ridge model using all data, and report this re-trained model.
   e. If a random number is used, it is better to set the global random state to 4320 before running the random process (to promote reproducibility of the result).
   f. As there are many different approaches to divide the dataset into k folds, it is fine as long as (i) your cv rmse is calculated based on the right procedure, and (ii) your reported rmse does not differ a lot from our sample.

Code:

```
# Problem 6
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler
import random
def problem_6(df_X, df_y, kfold=10, alpha=1.0):
    # write your logic here, model is the trained ridge model
    model = 0
    rmse = 0
    random_state = 4320       # default is 4320
    random.seed(random_state) # may be useful


    return model, rmse
```

Execution:

```
> df = pd.read_csv('USA_Housing.csv')
> df_X = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area
Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population']]
> df_X = pd.DataFrame(StandardScaler().fit_transform(df_X.values))
> df_y = df['Price']
> model, rmse = problem_6(df_X, df_y, kfold=10)
> print("model: ", model)
model:  Ridge()
> print("cv rmse: ", rmse)
cv rmse:  101195.53505726194
```

< End of Assignment >