## Description:

File HW01-6.py, HW01-7.py, and HW01-8.py, is corresponding to Problem 6, 7, and 8. Three file use python3 to write, and import numpy and matpltlib as the external liberary, so they should be installed then will be run properly.

## Environment

- python3.7
  - numpy
    ```
    $ pip3 install numpy
    ```
  - matplotlib
    ```
    $ pip3 install matplotlib
    ```

## How to run the code:

For Windows:
Open Anaconda shell then go to file folder
```
$ cd ~/path/to/the/folder
```
then type
```
$ python3 *.py # * is the file name
```
to run the code.

For MacOS or Linux:
Open terminal then go to file folder
```
$ cd ~/path/to/the/folder
```
then type
```
$ python3 *.py # * is the file name
```
to run the code.

## Code structure

### Problem 06

```
class PLA(object):
    """This is the PLA class,
    which user need to input the data path, the dimension of x features, and random se
    Build in function include preprocess the x input and y output,
    also the PLA algorithm in iteration.
    """
    def __init__(self, path, xDim, randomSeed):
        self.path = path
        self.Dim = xDim
        self.randomSeed = randomSeed

    def preprocess(self):
        data = np.loadtxt(self.path, dtype = float)
        if self.randomSeed == True : np.random.shuffle(data)
        else: pass
        x = np.c_[np.ones((data.shape[0],1),dtype = float), data[:,:self.Dim]]
        y = data[:,self.Dim]
        return x,y

    def iteration(self):
        x, y = self.preprocess()
        w = np.zeros((self.Dim+1,1))
        update = 0
        while(True):
            finish = True
            for i in range(len(x)):
                if np.dot(x[i], w)*y[i] <= 0:
                    w += (x[i]*y[i]).reshape(self.Dim+1,1)
                    update += 1
                    finish = False
            if finish == True:
                break
        return update
```

This is a class to do the PLA. `preprocess()` will sperate the `.dat` to label x and label y matrix. Then `iteration()` will do the percetron learning algorithm.

```
if __name__ == "__main__":
    result = pocketPLA('hw1_7_train.dat', 4, True)
    testX, testY = result.preprocess('hw1_7_test.dat')

    print("Iteration Start. Please wait...")
    count = []
    for i in range(1126):
```

```
        w = result.iteration()
        print(f'Iteration {i} Finish')
        count.append(result.verification(testX, testY, w))
    print("Finish")
    print(f"Average error rate is {sum(count)/1126.0}")

    plt.hist(count)
    plt.xlabel('Error Rate')
    plt.ylabel('Frequency')
    plt.show()
```

This is running part for 1126 times to test the update number and show the histogram.

### Problem 07

The code is typically same as Problem 06, but

```
while(update < 100):
    for i in range(len(x)):
        if np.dot(x[i], w)*y[i] <= 0:
            w += 0.5*(x[i]*y[i]).reshape(self.Dim+1,1)
            update += 1
            if self.verification(x, y, w) < self.verification(x, y, wPk):
                wPk = np.copy(w)
return wPk
```

This show the pocket process

```
def verification(self, x, y, w):
    errorCount = 0
    for i in range(len(x)):
        if np.dot(x[i], w)*y[i] <= 0:
            errorCount += 1
    return errorCount/len(x)
```

This is the verification function.

### Problem 08

```
while(update < 100):
    for i in range(len(x)):
        if np.dot(x[i], w)*y[i] <= 0:
            w += (0.5*x[i]*y[i]).reshape(self.Dim+1,1)
            update += 1
return w
```

The part of PLA update 100 times.