مقدمه:

در این تکلیف هدف استفاده از از الگوریتم Perceptron برای آموزش شبکه و جدا سازی کلاسهایی مختلف است.

سوال ۱:

در این سؤال یک ست ورودی و خروجی داده شده است که همان‌طور که در کتاب هم آمده است آن‌ها هر کدام یک کاراکتر را با فونت خاصی نشان می‌دهد

ما می‌خواهیم شبکه ای طراحی کنیم که این فونت هارا تشخیص دهد و از هم جدا سازد این کار را با الگوریتم Perceptron استفاده می‌کنیم

برای خواندن مقادیر از فایل از کد زیر استفاده شده است:

```
################################      Main      ##################################
def read_train_file(file="OCR_train.txt"):
    training_data_list = []
    train_file = open(file, "r")
    for line in train_file:
        line = list(line.replace(" ", ""))
        line = [int(x) * 2 - 1 for x in line if x != "\n"]
        training_data_list.extend([line[:]])
    return training_data_list
```

که در آن عدد های ۰ به -۱ و عدد ۱ به همان ۱ تغییر پیدا می‌کنند

برای پیشبینی مقادیر از کد زیر استفاده شده است:

```
# Make a prediction with weights
def predict_h(row, out_weight, out_bias):
    activation = out_bias
    for j in range(63):
        activation += out_weight[j] * row[j]
    return 1.0 if activation >= 0.0 else -1.0
```

و برای آموزش شبکه از کد زیر استفاده شده است:

```
# Estimate Perceptron weights using stochastic gradient descent
def train_weights_perception(train, l_rate, n_epoch):
    weights_vector = [[0.0 for i in range(63)] for j in range(7)]
    bias_vector = [0.0 for i in range(7)]
    sum_error = 0
    for i_epoch in range(n_epoch):
        for row in train:
            # print()
            # print(weights_vector[0])
            # print(bias_vector[0])
            for outPutNumber in range(7):
                sum_error = 0.0
                # print("Target is: ", row[64 + outPutNumber])
                prediction = predict_h(row, weights_vector[outPutNumber], bias_vector[outPutNumber])
                # print("Predict is: ", prediction)
                error = prediction - row[64 + outPutNumber]
                # print(error)
                sum_error += error ** 2
                # if sum_error != 0:
                #     print(sum_error)
                if sum_error != 0:
                    bias_vector[outPutNumber] = bias_vector[outPutNumber] + l_rate * row[64 + outPutNumber]
                    for j in range(63):
                        weights_vector[outPutNumber][j] = weights_vector[outPutNumber][j] + l_rate * row[64 + o
        print('>epoch=%d, lrate=%.3f, error=%.3f' % (i_epoch, l_rate, sum_error))
    return weights_vector, bias_vector
```

سپس کد را با صدا کردن تابع اجرا می کنیم:

```
epoch = 3
weights, bias = train_weights_perception(dataset, 0.2, epoch)
print(weights[0])
print(bias)
################           Enter your code above ...           ##################

print("\nThe Neural Network has been trained in " + str(epoch) + " epochs.")
```

و بعد از اجرا می‌بینیم که با ۳ epoch شبکه کامل‌ترین می‌شود

```
The Neural Network has been trained in 3 epochs.
```

برای تست کردن شبکه فایل تست را می خونیم و آن را به شبکه می دهیم:

```
################                    Testing                    ##################
test_dataset = np.array(read_train_file("OCR_test.txt"))

################          Enter your code below ...            ##################
_error = 0
_total = 0
for row in test_dataset:
    _total += 1
    for outPutNumber in range(7):
        predict = predict_h(row, weights[outPutNumber], bias[outPutNumber])
        error = predict - row[64 + outPutNumber]
        if error != 0:
            _error += 1
            break

################          Enter your code above ...            ##################

print("\n\nPercent of Error in NN: " + str(_error / _total))
```

بعد از اجرا می‌بینیم که ۳۶ مورد از آن‌ها دارای خطا شده است و ارور آن ۳۷٪ است:

```
Percent of Error in NN: 0.37142857142857144
```

سؤال ۲:

کد هایی برای این قسمت زده شده و به پیوست قرار داده شده است ولی به جواب نرسیده!