MACHINE LEARNING HOMEWORK 5

		Q1	Total
Grade	Max	5	5 points
	Expected	5	5

STUDENT

NAME SURNAME : ALPEREN KANTARCI

STUDENT NUMBER : 504191504

DEADLINE : 13/01/2020

COURSE NAME : MACHINE LEARNING

INSTRUCTOR : YUSUF YASLAN

QUESTION 1:

Autoencoders are used for data reconstruction, image denoising, dimensionality reduction and latent space creation tasks. In this question we are asked to create an autoencoder for opdigit dataset that have 64 features for each digit. We create a latent space with 2 dimensions and we reconstruct the features of the digits by using these 2 dimensions. Therefore, in 2 dimensional same digits should be clustered in the 2 dimensional latent space.

To create the autoencoder we first preprocess the dataset. Features are scaled to [0,1] range for easier learning in the multi layer perceptron model.

```
data = pd.read_csv('data.txt', delimiter=",", header = None)
data = data.to_numpy()
# Randomize the input
np.random.shuffle(data)
data_features = data[:,:64]
min_max_scaler = MinMaxScaler((0 ,1))
data_features = min_max_scaler.fit_transform(data_features)
data_labels = data[:,-1:]
```

Then we create 2 layer neural network which is called as multi layer perceptron. Input has 64 features (dimensions), hidden (latent) space has 2 dimensions and output also has 64 dimensions.

```
w1 = np.random.uniform(0, 1, (64, 2))
w2 = np.random.uniform(0, 1, (2, 64))

b1 = np.full((1, 2), 0.1)
b2 = np.full((1, 64), 0.1)

for i in range(epochs):
    a1 = data_features.copy()
    z2 = a1.dot(w1) + b1
    a2 = sigmoid(z2)
    z3 = a2.dot(w2) + b2
    a3 = sigmoid(z3)

    cost = np.sum((a3 - data_features)**2)/2

    a3_delta = (a3-data_features)
    z3_delta = sigmoid_derivative(a3_delta)
    dw2 = a2.T.dot(z3_delta)
    db2 = np.sum(z3_delta,axis=0, keepdims=True)

    z2_delta = z3_delta.dot(w2.T) * sigmoid_derivative(z2)
    dw1 = a1.T.dot(z2_delta)
    db1 = np.sum(z2_delta,axis=0, keepdims=True)

    # update parameters
    for param, gradient in zip([w1, w2, b1, b2], [dw1, dw2, db1, db2]):
        param -= learning_rate * gradient/(gradient*gradient)
```

Above you can see the both forward and backward propagation of the autoencoder. In the last layer we have sigmoid because outputs are also normalized to [0,1] range. For 1000 epochs we make forward and backward propagations while adjusting weights. You can see the loss decrease of the network. Mean Squared Error (MSE) loss is used to asses the error of the network which is reconstruction of the input.

```
Epoch: 0/1000 Error: 170329.7401
Epoch: 100/1000 Error: 112344.9161
Epoch: 200/1000 Error: 110294.0570
Epoch: 300/1000 Error: 108261.1142
Epoch: 400/1000 Error: 106286.0841
Epoch: 500/1000 Error: 104559.5481
Epoch: 600/1000 Error: 103211.1479
Epoch: 700/1000 Error: 101688.2893
Epoch: 800/1000 Error: 100391.3117
Epoch: 900/1000 Error: 99174.3023
```

After training is finished, dataset is given to the network and 1 forward propagation has been made. Then latent space outputs (2 dimension) is plotted. You can see the plot below. As you can see in the plot, digits are clustered between 0,1 range as Alpaydın's book. Also at the corners same digits are clustered. It should have been more scattered around latent space but i think this error because of the sigmoid overflow and 0,1 scaling.