0616038 蔡雨恩

Machine Learning HW4

```
def getLoss(predict, target):
    return np.mean(np.absolute(predict-target))


def Sigmoid(x):
    return 1.0/(1.0 + np.exp(-(np.clip(x, -100, 100))))


def Sigmoid_Deriv(x):   #input Sigmoid and output its differential
    return x * (1.0 - x)

def getAcc(predict, target):
    correct = 0
    for i in range(len(target)):
        if(predict[0][i] <= 0.5):
            a = 0
        else:
            a = 1
        if (a == target[i]):
            correct += 1
    return correct / len(target)
```

getLoss wil calculate the loss using predict and target value

Sigmoid and Sigmoid_Deriv will calculate Sigmoid and its differential

getAcc will calculate the accuracy of each target

```
def ForwardPropagation(w,b, data):
    a = []
    for i in range(3):
        if(i==0):
            f = data
        else:
            f = a[i-1]
        a.append(Sigmoid(w[i].dot(f) + b[i]))
    return a
```

if i equals to 0 then set f's value to data

if i do not equal to 0 then set f's value to a[i-1]

then calculate the value using Sigmoid

```python
def Propagation(ws, bs, data, target, alpha):
    a = ForwardPropagation(ws, bs, data)

    proArr = []
    proArr.append(data.T)
    for i in range(len(a) - 1):
        proArr.append(a[i].T)

    sigD = Sigmoid_Deriv(np.array(a))

    predictL = []
    predictL = (a[-1] - target)[0]

    rev = []
    rev.append(np.array(predictL) * sigD[-1])
    rev.append(ws[2].T.dot(rev[-1]) * sigD[-2])
    rev.append(ws[1].T.dot(rev[-1]) * sigD[-3])
    rev.reverse()

    temp1 = np.ones( (len(target),1) )
    for i in range(3):
        ws[i] -= alpha * rev[i].dot(proArr[i])
        bs[i] -= alpha * np.sum(rev[i].dot(temp1))
```

Use proArr to save data and a's value

Use sigD to save Sigmoid's differential's value

Use rev to save predict and ws multiply sigD's value

Use ws to calculate the value of alpha multiply rev dot proArr

Use bs to calculate the value of alpha multiply rev got temp1's sum

```python
def main():
    #seed(123123)
    hiddenN = 50
    alpha = 0.13


    data = np.genfromtxt('data.txt', delimiter = ',')
    data = data.T
    target = copy.deepcopy(data[2])

    data = np.delete(data, 2, 0)


    ws = [ np.random.randn(hiddenN,2), np.random.randn(hiddenN, hiddenN), np.random.randn(1, hiddenN)]

    bs = [0,0,0]

    allAcc = []
    epochNum = 100000

    for i in range(epochNum):
        Propagation( ws, bs , data, target,alpha)
        if ( (i+1) % 10000 == 0 ) :
            predict = ForwardPropagation(ws, bs, data)[-1]
            loss = getLoss(predict, target)
            print('epochs ', i+1, end=' ')
            print('loss:', loss)
            allAcc.append(getAcc(predict, target))
    print(allAcc)
    #showPlot("groudtruth", data, target)
    #showPlot("L2_Norm_Predict", data, ForwardPropagation(iw, hw, ow, ib, hb, ob, data)[-1].T)
```

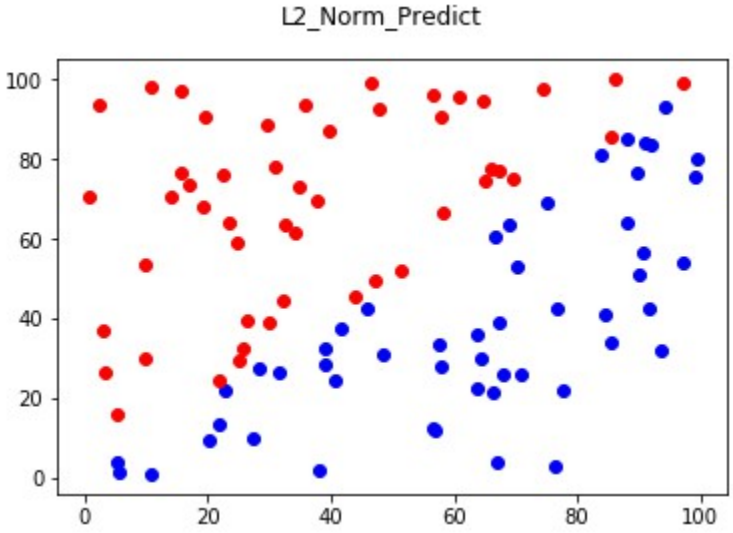
Read data.txt and set target's value

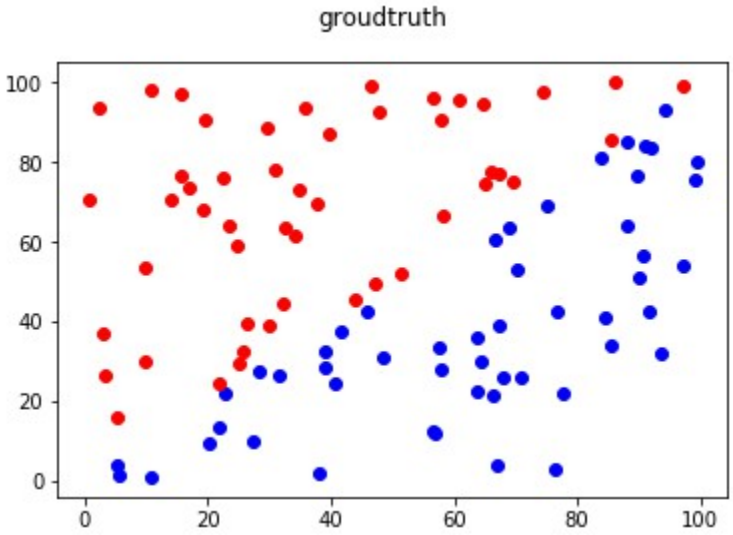
print epochs and loss

print accuracy

Loss and Accuracy

```
epochs  10000 loss: 0.001887132359263446
epochs  20000 loss: 0.0011241936165900619
epochs  30000 loss: 0.0008857019245271886
epochs  40000 loss: 0.0007557245751308573
epochs  50000 loss: 0.0006703823414943233
epochs  60000 loss: 0.0006086789050135876
epochs  70000 loss: 0.0005613273958011738
epochs  80000 loss: 0.0005234794599663209
epochs  90000 loss: 0.0004923164921719418
epochs  100000 loss: 0.00046607046334221617
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```


L2_Norm_Predict
100
80
60
40
20
0
0
20
40
60
80
100


groudtruth
100
80
60
40
20
0
0
20
40
60
80
100