Q4-NN

May 5, 2019

0.0.1 import modules

```
In [1]: import numpy as np
        import pandas as pd
```

0.0.2 Define sigmoid function and E (error) function

```
In [2]: def sigmoid(x):
            return (1 / (1 + np.exp(-x)))

In [3]: def lossfunc(y,o):
            return (y-o)**2/2
```

0.0.3 Set X and y

```
In [4]: X = np.array([1,0.05,0.1])
        y = 0.9
```

0.1 (a) start initail values and calculate h_i, o, E

Set functions in the problem and initialize a_i, b_i, w_i

$$a_i \overset{\text{iid}}{\sim} N(0,1) b_i \overset{\text{iid}}{\sim} N(0,1) w_i \overset{\text{iid}}{\sim} N(0,1)$$

for $i = 0, 1, 2$

```
In [5]: np.random.seed(0)
        A_0 = np.random.randn(3)
        B_0 = np.random.randn(3)
        W_0 = np.random.randn(3)

        H_0 = np.array([1,sigmoid(np.matmul(A_0,X)),sigmoid(np.matmul(B_0,X))])
        O_0 = sigmoid(np.matmul(W_0,H_0))

In [6]: print("a_i's are", A_0)
        print("b_i's are", B_0)
        print("w_i's are", W_0)
```

```
a_i's are [1.76405235 0.40015721 0.97873798]
b_i's are [ 2.2408932   1.86755799 -0.97727788]
w_i's are [ 0.95008842 -0.15135721 -0.10321885]
```

```
In [7]: print("h_i's are", H_0)
        print("Threrfore\no : %f \nE : %f" %(O_0,lossfunc(y,O_0)))
```

```
h_i's are [1.         0.86783311 0.90348345]
Threrfore
o : 0.673812
E : 0.025580
```

0.2 (b) Update w_i's and o and E

let $\alpha_k = 1$ then

```
In [8]: alphak = 1
        W_1 = W_0 + alphak*(y-O_0)*O_0*(1-O_0)*H_0
        O_1 = sigmoid(np.matmul(W_1,H_0))
        print("updated w_i's are", W_1)
        print("updated o : %f" %O_1)
        print("updated E : %f" %lossfunc(y,O_1))
```

```
updated w_i's are [ 0.99980202 -0.1082141  -0.05830344]
updated o : 0.701241
updated E : 0.019753
```

0.3 (c) Update a_i's and b_i's and o and E

```
In [9]: A_1 = A_0 + alphak *(y-O_0)*O_0*(1-O_0)\
                    * W_1[1]*H_0[1]*(1-H_0[1])*X
        B_1 = B_0 + alphak *(y-O_0)*O_0*(1-O_0) \
                    * W_1[2]*H_0[2]*(1-H_0[2])*X

        H_1 = np.array([1,sigmoid(np.matmul(A_1,X)),sigmoid(np.matmul(B_1,X))])
        O_1 = sigmoid(np.matmul(W_1,H_1))

        print("updated a_i's are", A_1)
        print("updated b_i's are", B_1)
        print("updated o : %f" %O_1)
        print("updated E : %f" %lossfunc(y,O_1))
```

```
updated a_i's are [1.7634353  0.40012636 0.97867628]
updated b_i's are [ 2.24064045  1.86754535 -0.97730315]
updated o : 0.701243
updated E : 0.019752
```

0.4 (d) Repeat 10000 times

```
In [10]: for i in range(10000):
             W_0 = W_0 + alphak*(y-O_0)*O_0*(1-O_0)*H_0
             A_0 = A_0 + alphak *(y-O_0)*O_0*(1-O_0) \
                         * W_1[1]*H_0[1]*(1-H_0[1])*X
             B_0 = B_0 + alphak *(y-O_0)*O_0*(1-O_0) \
                         * W_1[2]*H_0[2]*(1-H_0[2])*X

             H_0 = np.array([1,sigmoid(np.matmul(A_0,X))\
                             ,sigmoid(np.matmul(B_0,X))])
             O_0 = sigmoid(np.matmul(W_0,H_0))

         print("repeated a_i's are", A_0)
         print("repeated b_i's are", B_0)
         print("repeated w_i's are", W_0)
         print("repeated o : %f" %O_0)
         print("repeated E : %f" %lossfunc(y,O_0))
```

```
repeated a_i's are [1.75692136 0.39980066 0.97802489]
repeated b_i's are [ 2.23797638  1.86741215 -0.97756956]
repeated w_i's are [1.52313809 0.34572447 0.41445062]
repeated o : 0.900000
repeated E : 0.000000
```