Assignment on ANN

IIT2019044
Shreesh Swaraj
Semester 5

02/11/2021

Google Colab Link for Code

https://colab.research.google.com/drive/1uOg4pbVzDVsUd35nU4yXmyVYtUkEsOND?usp=sharing

Problem

A medical dataset along with its description is attached. The main idea of this data set is to prepare the algorithm of the expert-system, which will perform the presumptive diagnosis of two diseases of urinary system.

Build the above expert system using an artificial neural network using only the Numpy library and implement the following:

1. Label encode the different categorical features using binary identifiers[1,0].
2. Fix the number of hidden layers to just 1; use sigmoid activation for this hidden layer. Also set the number of nodes in the hidden layer to 7.
3. Perform the forward and backward pass and report the weight matrices for first three iterations.
4. Finally train the model for 50 epochs and report the accuracy of the model on the test set.
5. Repeat the above steps for two hidden layers with 5 and 10 nodes respectively; for the two hidden layers use linear and sigmoid activation in the same order.

Loading Dataset

	c1	c2	c3	c4	c5	c6	c7	c8
0	35.5	no	yes	no	no	no	no	no
1	35.9	no	no	yes	yes	yes	yes	no
2	35.9	no	yes	no	no	no	no	no
3	36.0	no	no	yes	yes	yes	yes	no
4	36.0	no	yes	no	no	no	no	no
...	...	...	...	...	...	...	...	...
115	41.4	no	yes	yes	no	yes	no	yes
116	41.5	no	no	no	no	no	no	no
117	41.5	yes	yes	no	yes	no	no	yes
118	41.5	no	yes	yes	no	yes	no	yes
119	41.5	no	yes	yes	no	yes	no	yes

120 rows × 8 columns

Different categorical features label encoded using binary identifiers [1,0]

	c1	c2	c3	c4	c5	c6	c7	c8
0	35.5	0	1	0	0	0	0	0
1	35.9	0	0	1	1	1	1	0
2	35.9	0	1	0	0	0	0	0
3	36.0	0	0	1	1	1	1	0
4	36.0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...
115	41.4	0	1	1	0	1	0	1
116	41.5	0	0	0	0	0	0	0
117	41.5	1	1	0	1	0	0	1
118	41.5	0	1	1	0	1	0	1
119	41.5	no	yes	yes	no	yes	no	yes

Normalising Dataset

	c1	c2	c3	c4	c5	c6
0	0.855422	0	1	0	0	0
1	0.865060	0	0	1	1	1
2	0.865060	0	1	0	0	0
3	0.867470	0	0	1	1	1
4	0.867470	0	1	0	0	0
...	...	...	...	...	...	...
115	0.997590	0	1	1	0	1
116	1.000000	0	0	0	0	0
117	1.000000	1	1	0	1	0
118	1.000000	0	1	1	0	1
119	1.000000	no	yes	yes	no	yes

Weights after first iteration

```
W1: [[-0.16838905731313505 0.528686685787953 -1.3246784885124872
  -1.88181078523066 -1.1368369840612471 1.75234403353225
  -0.7910327155627892]
 [-0.5363300500557713 0.09424337013735706 0.264511599832538
  -0.5987572023094798 1.3686834595087864 -1.021755163434921
  0.36259855036893357]
 [0.3103672741466679 0.848906997468438 -0.14828989317888402
  -1.5531235884434496 0.6687330142348022 -1.3596319343382313
  0.3590808006119497]
 [0.22610360665072754 -0.011718779869319892 0.6312012517646344
  -1.089163472934523 -1.1082653389407506 -0.43859932303935706
  0.6732077260569097]
 [1.4630621007136946 1.1921489073050464 -1.2383322567160981
  0.3323271279030259 -0.14235819055303836 -0.1972567073479907
  -0.5662106008112515]
 [-0.5435607283553581 -1.535726981329902 -1.0457001274499862
  -1.0057251858153464 -1.3468919210072394 0.02263691403260983
  -0.5761363515205895]]

W2:  [[0.22548141098669186 -0.2791198893167637]
 [-0.3165321438934706 -1.5325773026020084]
 [-0.23139097940234543 0.15045066804285076]
 [-0.7777618630874495 -0.44438810334388534]
 [-0.6440564208205178 0.5013721360309237]
 [2.145932699068191 -0.906708744842806]
 [-0.0571237533164734 -1.2849529410994154]]
```

Weights after second iteration

```
W1: [[3.86719431249992 4.805002137987922 4.0734164829480015
  4.4217670501025665 4.580390379974492 4.8716215867323545
  4.7262767554660545]
 [-0.3748100358799615 0.40830135743830526 -2.16866672166017
  0.4691978077031772 0.04355538243724765 -5.4507934621585505
  -1.3325342897618244]
 [2.7643310945417072 1.5016968119599612 2.2435633625467997
  1.3907957063122016 1.8269177257403526 0.5608546244307439
  0.8176421770134429]
 [3.2029402499173485 2.576984612516074 3.336418659320227
  2.353537981663988 2.117173403820487 3.266231030092238
  1.8790094242150555]
 [1.3912570706930327 -0.096704820681882 1.3761055698774 2.097058013307806
  0.4886762589485929 -2.0442928719661855 0.9457008367073414]
 [-0.8772610849805224 1.1954510337393143 -1.2114398809921314
  0.07582015224132192 1.9084870522935045 0.43623549978648296
  0.8000610554480523]]

W2:  [[2.0617111783604605 -3.4050216025952307]
 [1.6335867535469155 -4.133899045938711]
 [1.5442076681215184 -3.796750425063303]
 [2.173439861773985 -4.450649896972071]
 [0.06081940409867489 -3.4300687536006333]
 [1.8744269766152801 -2.8580505857256346]
 [1.3914848424121873 -3.599906078309836]]
```

Defining ANN Class

```
class ANN():
 def   init  (self, input size, hidden_size, output_size, data_len):
   self.input size = input size
   self.hidden size = hidden size
   self.output size = output_size
   self.m = data len
   self.weight1 = np.random.randn(self.input size,self.hidden size)
   self.weight2 = np.random.randn(self.hidden_size,self.output_size)

 def sigmoid(self,X,deriv = False):
     if deriv:
       return X*(1-X)
     else:
       return 1/(1+np.exp(X))

 def feed forward(self,X):
   self.o1 = np.dot(X,self.weight1).astype(float)
   self.s1 = self.sigmoid(self.o1)
   self.o2 = np.dot(self.s1,self.weight2).astype(float)
   self.s2 = self.sigmoid(self.o2)
   return self.s2,self.s1

 def loss(self,y,output):
   loss = 0
   for i in range(len(output)):
     loss += -(y test.iloc[i][0]*np.log(output[0][i][0]) +
(1-y test.iloc[i][0])*(1-np.log(output[0][i][0])))
     loss += -(y test.iloc[i][1]*np.log(output[0][i][1]) +
(1-y_test.iloc[i][1])*(1-np.log(output[0][i][1])))

   return loss
```

```
def back_prop(self,x,y,output_y,lr = 0.4):
   self.dz2 = output y - y
   self.dw2 = (1/self.m)*np.dot(self.s1.T,self.dz2)
   self.dz1 = self.weight2.dot(self.dz2.T).T*self.s1*(1-self.s1)
   self.dw1 = (1/self.m)*np.dot(self.dz1.T,x).T

   self.weight2 = self.weight2 - lr*self.dw2
   self.weight1 = self.weight1 - lr*self.dw1

   return self.weight2,self.weight1

 def train(self,x,y):
   output = self.feed forward(np.array(x[:]))
   w2,w1 = self.back_prop(np.array(x[:]),np.array(y[:]),output[0])
   loss = self.loss(np.array(y[:]),output)
   return loss,w2,w1
```

Training for 1000 epochs

Plotting Epochs v/s Loss