Samar Ibrahim Antar

Assignment#5 Neural Network

1) __init__() function:

- It takes (no_of_in_nodes, no_of_out_nodes, no_of_hidden_nodes,learning_rate=0.01,gamma=0,n_epoch=100)
- Initialize the weights matrix by:
  Self.w1 >> np.random.randn(no_of_in_nodes, no_of_hidden_nodes) weight matrix from input to hidden layer [first set of weights].
  Self.w2 >>np.random.randn(no_of_hidden_nodes,no_of_out_nodes) weight matrix from hidden to output layer [second set of weights].

2) forward_propagation() function:

- It takes >> x (features)
- Then do dot product of x (input) & first set of weights (self.w1) and store result in (self.Z)
- self.Z enters the activation function (sigmoid) >> store result in (self.Z2)
- Dot product of hidden layer (Z2) & second set of weights(self.w2) >> store in (self.Z3)
- Self.Z3 enters the sigmoid function

3) Sigmoid() function:

- It takes >> variable [s] and derivative of sigmoid= False until change it to True when calling sigmoid() in backward() function
- 1/(1+ np.exp(-s)) & its derivative >>s * (1 - s)

4) back_propagation() function:

- It takes X_train, y_train & output of forward_propagation()
- First calculate error in output (y_train - output)
- Calculate delta for output layer by (error * sigmoid() and will change [derivative= False] to True) and store result in self.output_delta
- Calculate how much our hidden layer weights contribute to output error by dot product of (self.output_delta & self.w2)and store result in self.Z2_error
- Applying derivative of sigmoid to z2_error >> store result in z2_delta
- Then call function update_weights() with or without momentum, it takes x, z2_delta, output_delta.

5) Fit() function:

- trains the NN n_epochs times
- Each epoch, call the forward_propagation() & back_propagation()
- Then calculate loss and accuracy of train

6) Predict() function:

- It predicts the class value for the row of test data by returning the index in the network output that has the largest probability.

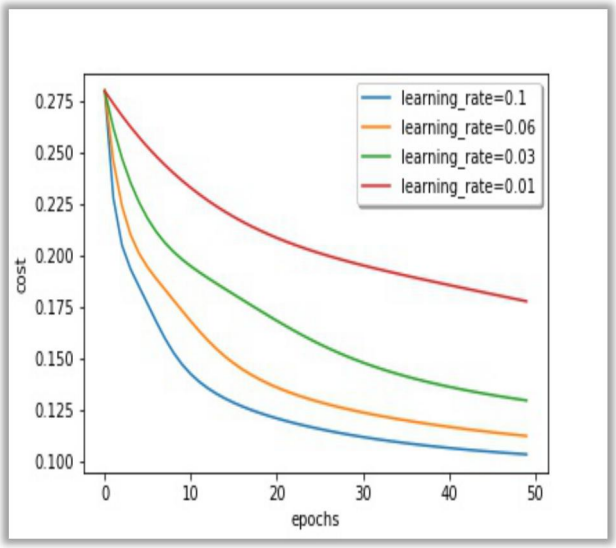
7) **One_hot_encoding() function:**

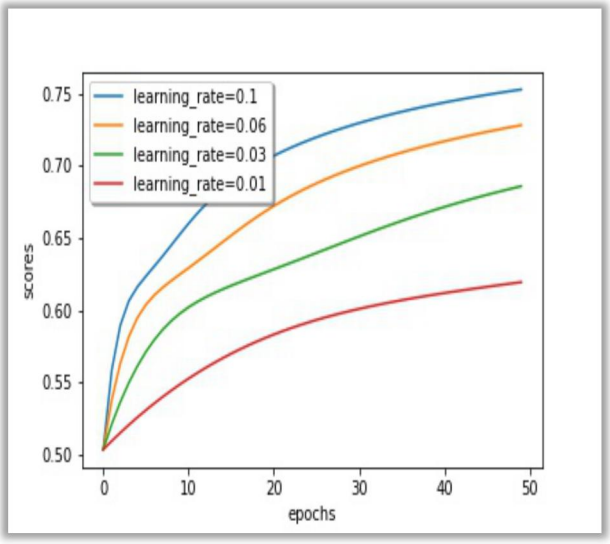
- It take classes values [y] and then encoding them,
  for example class value=0 or "any string" >> encoding to [1,0,0]
  Class value=1 >> encoding to [0,1,0] and so on.
- This function helps in predicting the class value of test data, when predict() return the index of largest probability, this index refers to class value according to its encoding.

8) Score() function:

- It takes actual, predicted values and counts How many correct ones and calculates sum number of them , then divides on the number of total

9) Costs and accuracy of iris train data vs. N_epochs plot at different learning rates & 2 hidden nodes:

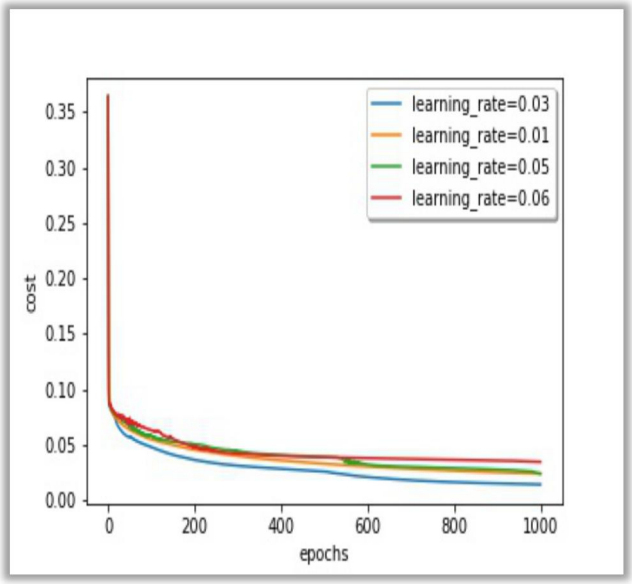


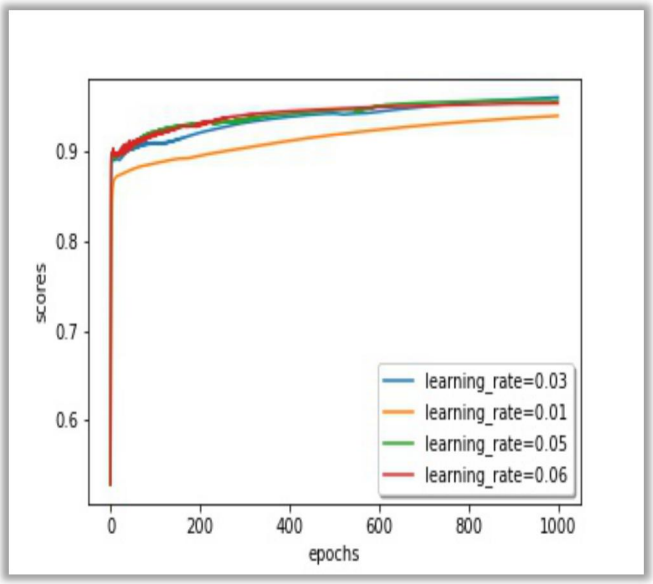


10) accuracy of iris test data at different learning rates & 2 hidden nodes: best at learning rate =0.1 >> accuracy= 97.778

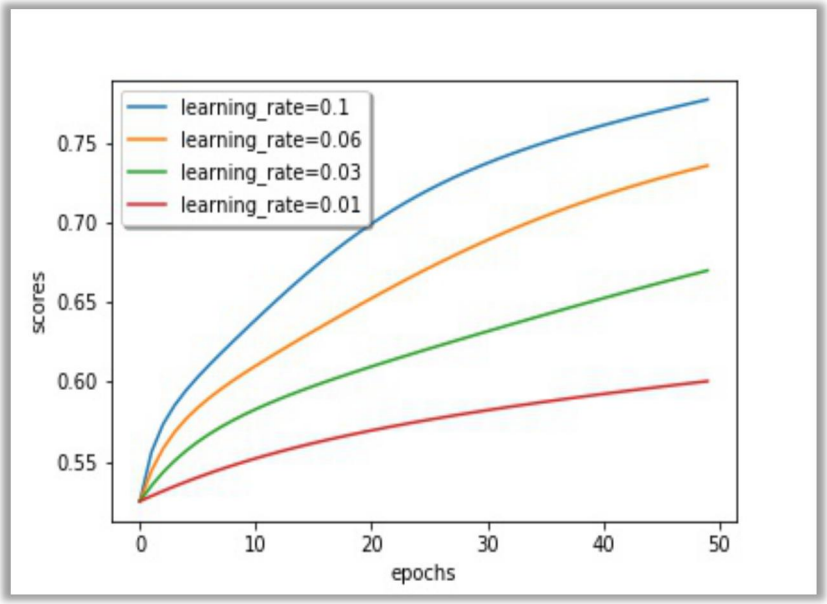
```
Reloaded modules: neural_networks
[97.77777777777777, 88.88888888888889, 77.77777777777779, 37.77777777777778]
```

11) Costs and accuracy of vowel train data vs. N_epochs plot at different learning rates & 20 hidden nodes:

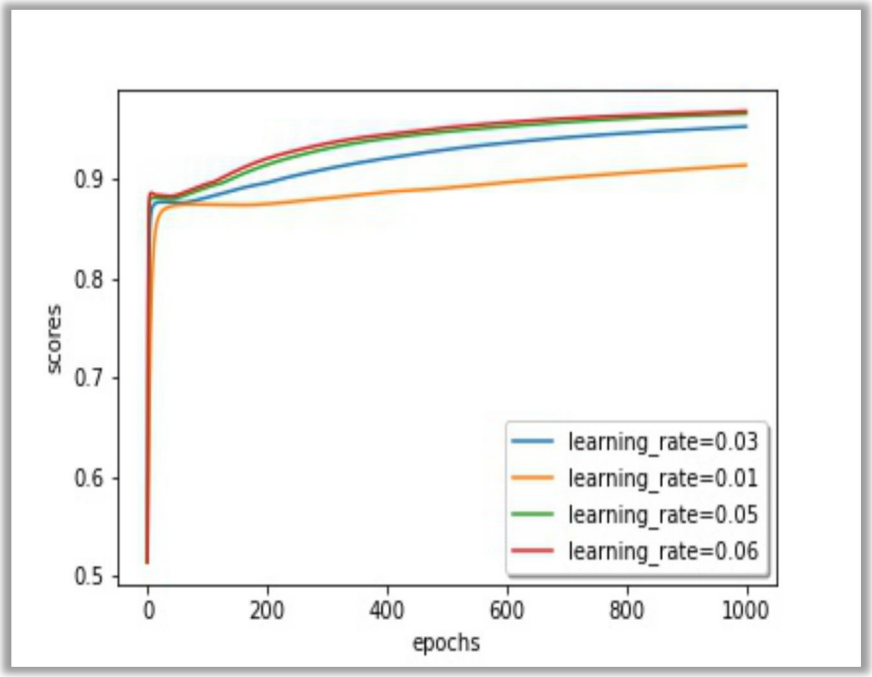




12) **accuracy of iris test data vs. N_epochs plot at different learning rates:**



13) **accuracy of vowel test data vs. N_epochs plot at different learning rates:**



Problem_5:

- the effect of different learning rates on the algorithm's performance:
  Some learning rates increase the performance and other not, According the iris data best learning rate with best number of hidden nodes are [learning_rate=0.1 & hidden_nodes=2] which give accuracy=97.78. According vowel data are [learning_rate=0.03 & hidden nodes=20] which give accuracy=79.435.

- Thus, different numbers of hidden units and different learning rates affect on performance, they can increase or decrease the accuracy, according the data too, the values of them differ. They are not the same for any data.

- For iris data >> H=105/10*(4+3)=1.5 is almost equals>> 2 (same as chosen)
  For vowel data >> H=742/10*(10+11)=3.5 is almost equals>>4 (here chosen hidden nodes=20 because it gives better accuracy compared to hidden nodes=4).
  Accuracy at different learning rates with number of hidden nodes=4

```
Reloaded modules: neural_networks
[48.38709677419355, 44.354838709677416, 49.596774193548384, 48.38709677419355]
```

accuracy of vowel test data at different learning rates & 20 hidden nodes: best at learning rate =0.03 >> accuracy=79.435

```
Reloaded modules: neural_networks
[79.43548387096774, 72.17741935483872, 70.56451612903226, 62.903225806451616]

In [378]:
```