Problem 4

Back propagation Algo:
The backpropagation part is the part where we use outputs and nets of layers along with activation functions to obtain the gradients which we then use to update the weights in order to reduce the cost.
I implemented the backprop algo from scratch using class.

Input size:

Before applying PCA:
the dimension of X_test before applying pca is : (10, 10201)
the dimension of X_train before applying pca is : (20, 10201)

the dimension of y_test that is labels of test dataset : (10,)
the dimension of y_train that is labels of train dataset : (20,)

After applying PCA:

the dimension of X_test after applying pca is : (12, 10)
the dimension of X_train after applying pca is : (12, 20)

the dimension of y_test that is labels of test dataset : (10,)
the dimension of y_train that is labels of train dataset : (20,)
layers_dims1 = [12,10,10,2] //10 nodes in hidden layer
layers_dims2 = [12,15,15,2] //15 nodes in hidden layer

Output of BackProp for different cases is given below:

TRAIN DATA

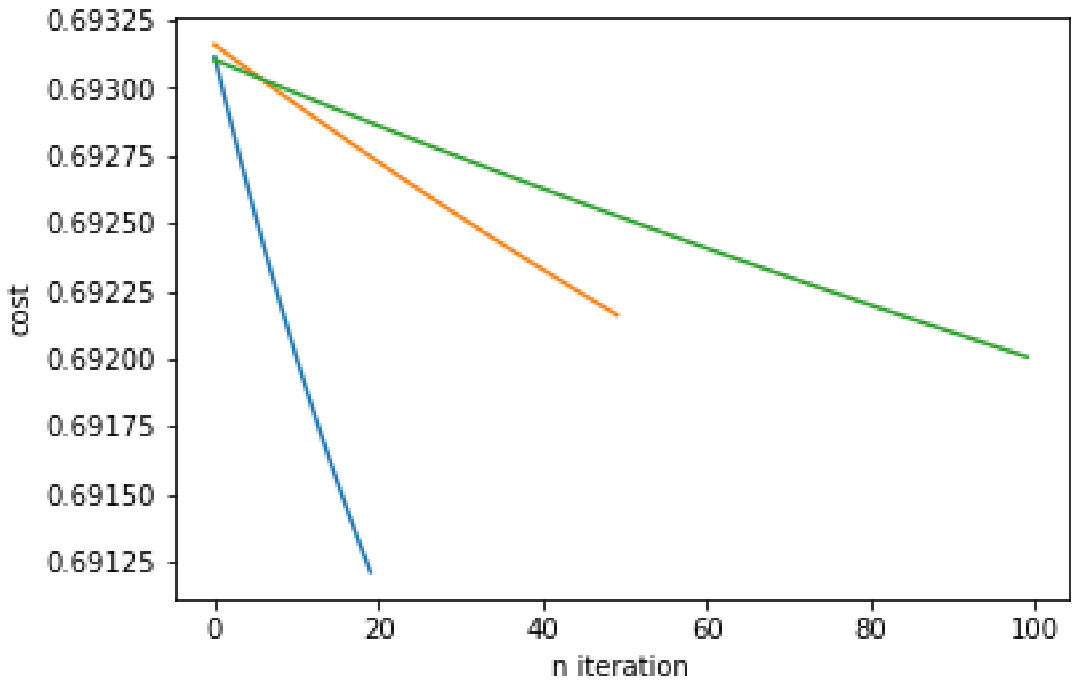
Case 1: when hidden nodes are 10

Train accuracy for different number of iterations with different learning rates:

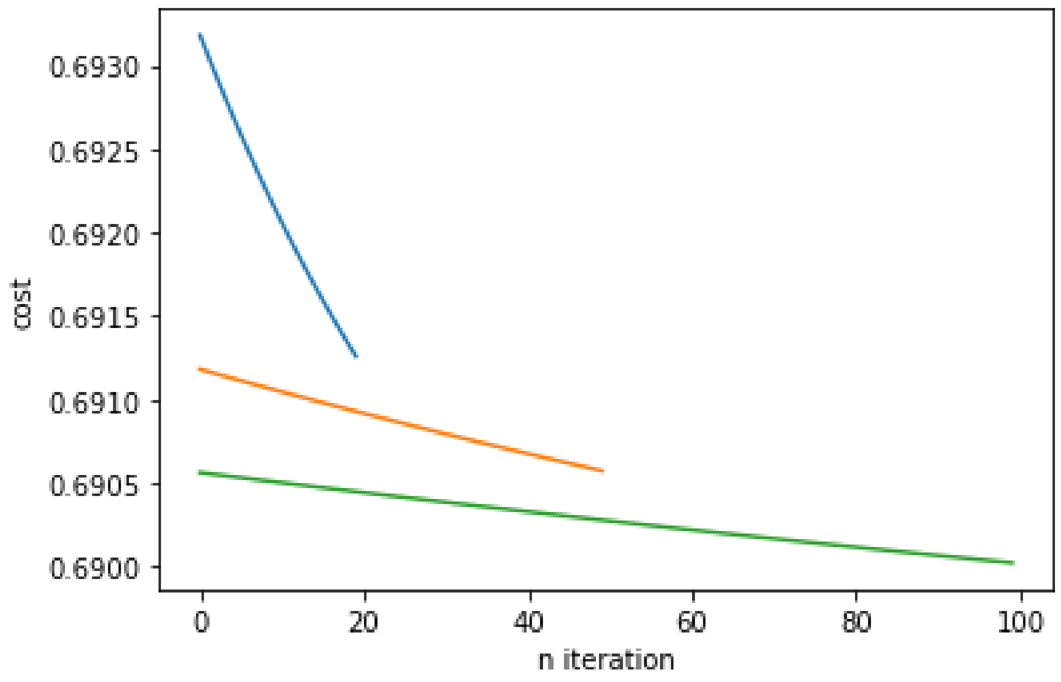
1. The blue line shows the cost value when the backprop algo is run for 20 iterations with learning rate 0.05

The orange line shows the cost value when the backprop algo is run for 50 iterations with learning rate 0.009

The green line shows the cost value when the backprop algo is run for 100 iterations with learning rate 0.005

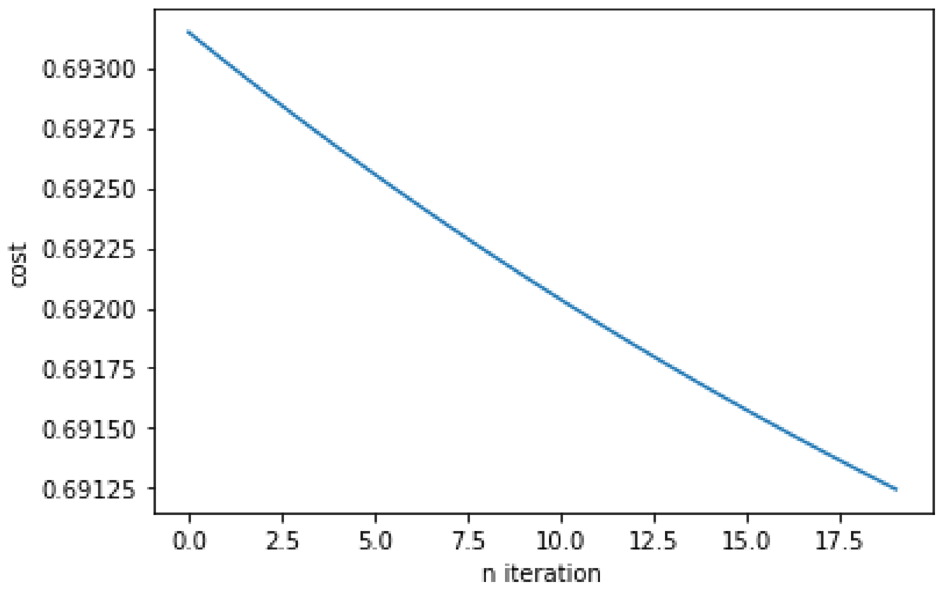


Case 2: when hidden nodes are 15
Train accuracy for different number of iterations with different learning rates:
1. The blue line shows the cost value when the backprop algo is run for 20 iterations with learning rate 0.05
The orange line shows the cost value when the backprop algo is run for 50 iterations with learning rate 0.009
The green line shows the cost value when the backprop algo is run for 100 iterations with learning rate 0.005

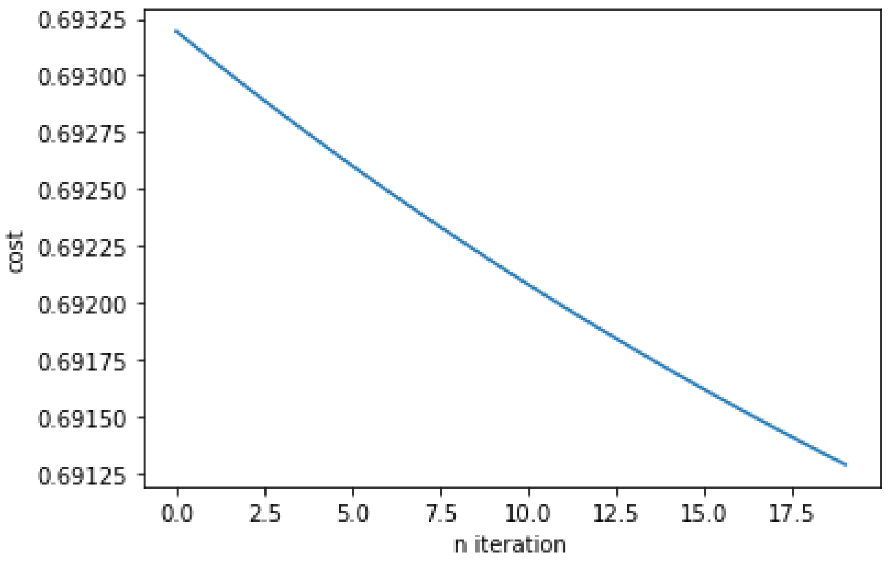


Test accuracy for different cases:
Y=[0 1 1 1 0 1 1 0 1 0]
predicted: [0 1 1 0 0 0 0 0 1 0]

Test accuracy of the model on 20 iterations and when hidden layers have nodes 10 is : 0.7



For 15 nodes in hidden layer:
predicted output :[1 1 1 1 1 1 1 1 1 0]
Test Data Accuracy: 0.7
Test accuracy of the model on 20 iterations and when hidden layers have nodes 15 is : 0.7



Note: As it is random initlization of weights hence accuracy will change on every run. Also note that for all other observations the cost/ error curve decreases as we increase number of iterations.

With momentum (0.9):

outputs :

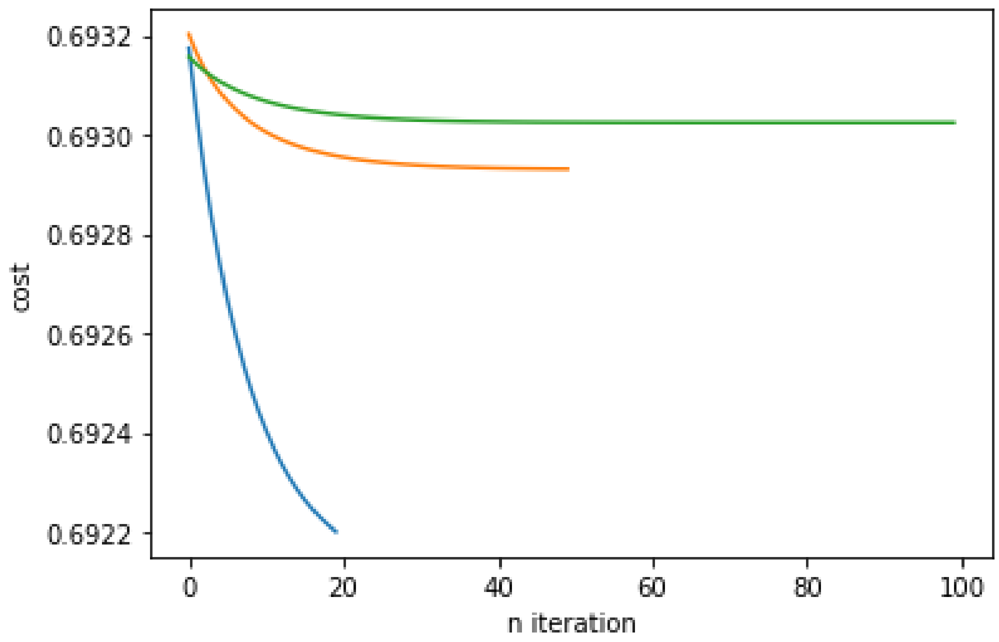
Note:with momentum the accuracy of model on test data is not decreasing below 60% .This is because of model is more generalizable.

Below is the error plot on train and test data:

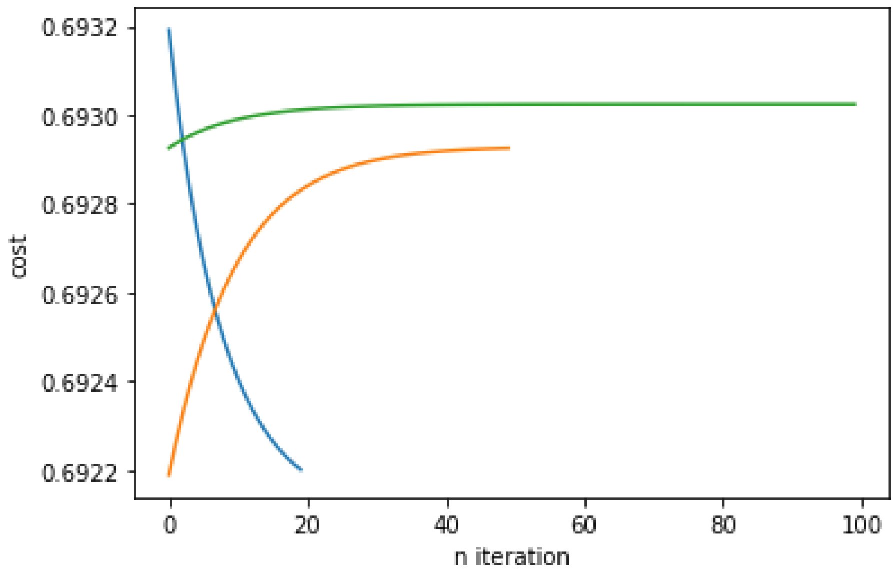
1. Train data

Blue line shows: 20 iterations
Orange line shows: 50 iterations
green line shows: 100 iterations

1.1 for hidden layer 10 nodes: for different iteratiions 20,50,100

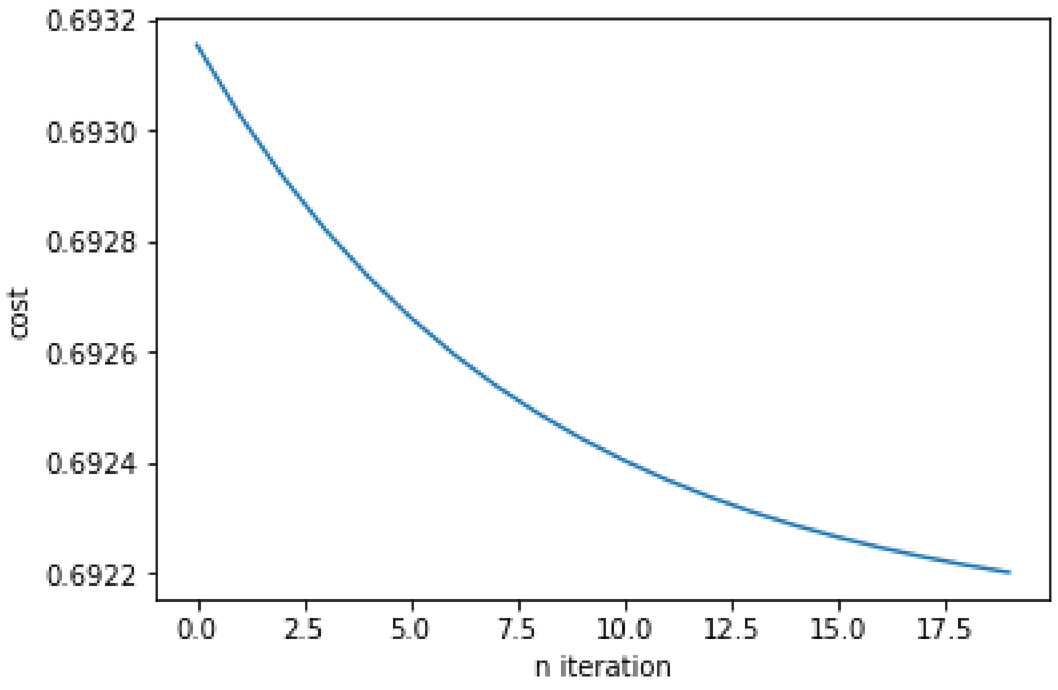


1.2 for hidden layers with nodes 15 ,iterations 20,50,100

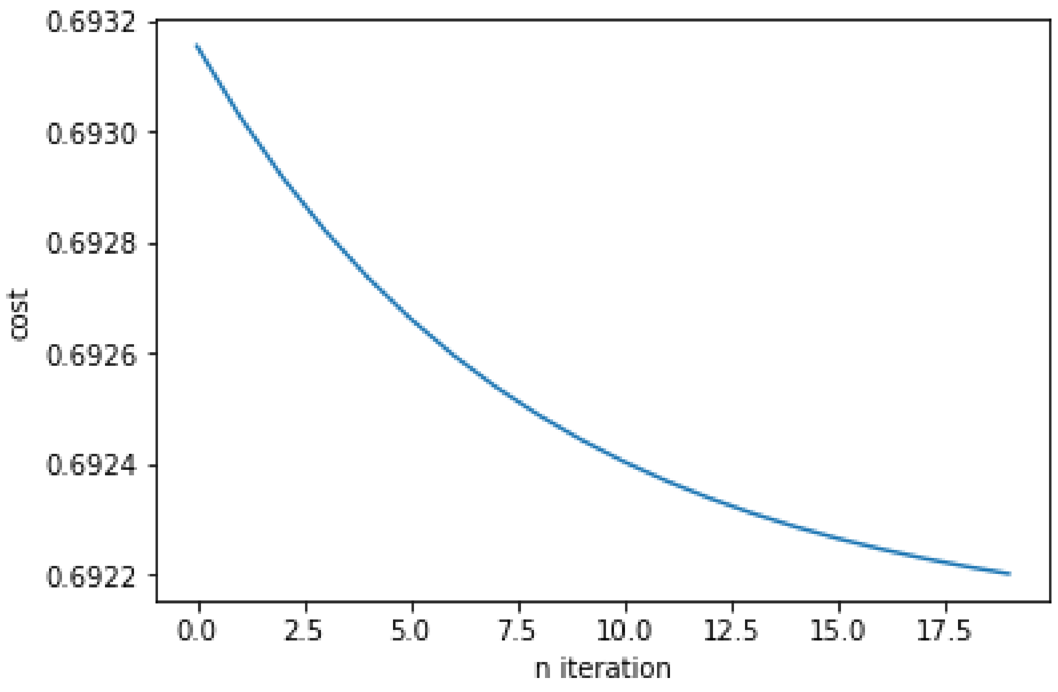


2. Test data
Blue line shows: 20 iterations

2.1 hidden layers with nodes 10 : Accuracy : 60%



2.2 hidden layers with nodes 15 : Accuracy 60%


cost
n iteration