project_report

December 6, 2019

```
In [1]: from data_loader import load_data
        from two_layer import TwoLayerNet

        from sklearn.metrics import accuracy_score
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn import tree
        %matplotlib inline
```

0.0.1 Load Data

The data_loader.py file contains the template program for loading the dataset. The file includes a function named load_data(), which reads the input data from a file and generates the appropriate training and test sets. You need to modify the load_data() function to complete this task.

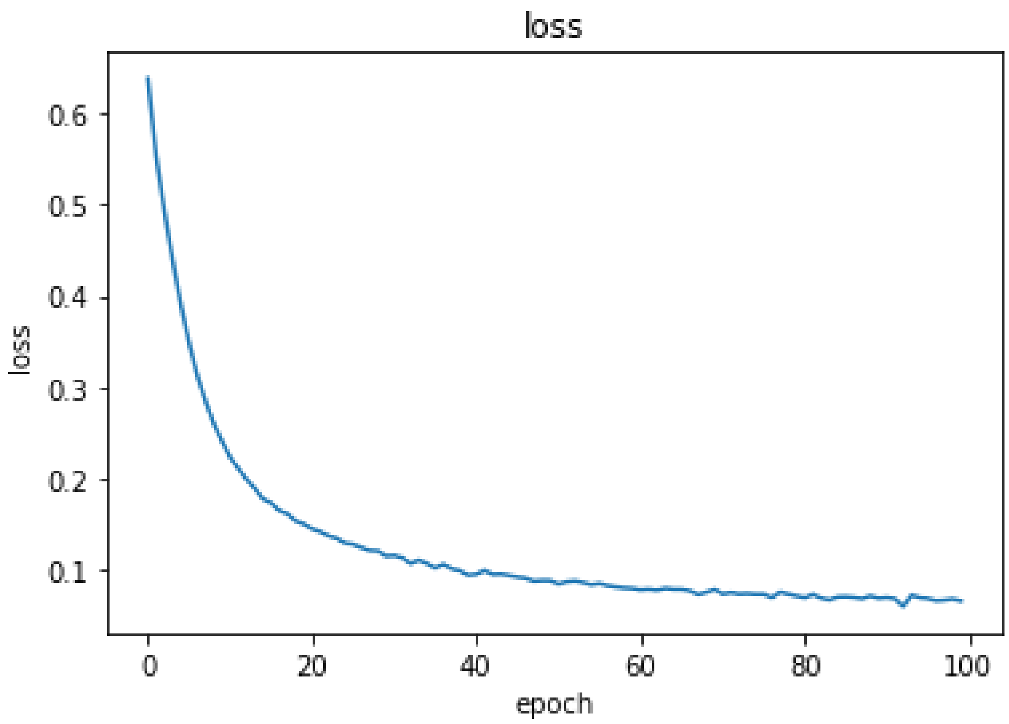
```
In [2]: file_path = './BCW-data.csv'
        X_train, X_test, Y_train, Y_test = load_data(file_path)
```

0.0.2 Two layer classifier

Test-1: A two layer model with sigmoid acitvation function for input layer.

```
In [58]: model = TwoLayerNet(input_dim = 9, hidden_dim = 5, epochs =100,learning_rate= 0.01 ,a
         model.fit(X_train, Y_train, plot_loss = True )
         Y_hat = model.predict(X_test)
         print('Test Accuracy',accuracy_score(Y_test, Y_hat))
```

```
epoch:: 100%|| 100/100 [00:11<00:00,  8.93it/s]
```

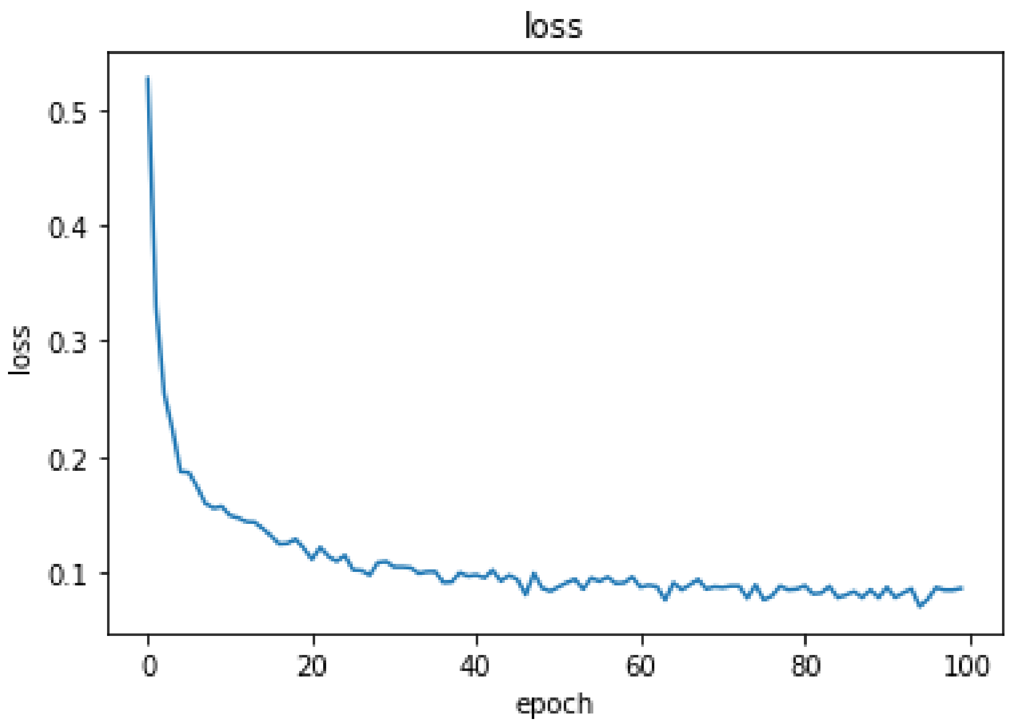


```
('Test Accuracy', 0.9590643274853801)
```

Test-2: A two layer model with ReLU acitvation function for input layer.

```
In [60]: model = TwoLayerNet(input_dim = 9, hidden_dim = 5, epochs =100 ,learning_rate= 0.01, a
         model.fit(X_train, Y_train, plot_loss = True  )
         Y_hat = model.predict(X_test)
         print('Test Accuracy',accuracy_score(Y_test, Y_hat))
```

```
epoch:: 100%|| 100/100 [00:11<00:00,  8.15it/s]
```



```
('Test Accuracy', 0.9502923976608187)
```

Discussion: The accuracy obtained by both sigmoid and ReLU activation are very close to each other. (approx 95%).

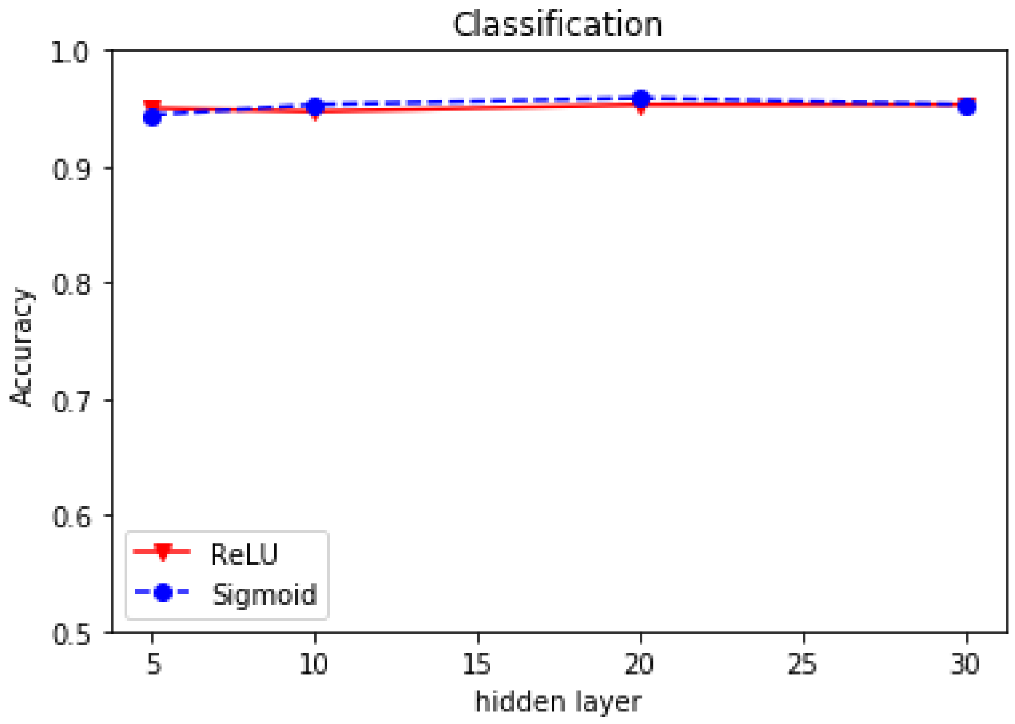
Question: What is the effect of the hidden_dim on the model accuracy for each activation function?

```
In [50]: hidden_dims = [ 5, 10, 20, 30]
         ReLU_loss = list()
         Sigmoid_loss = list()

         index = 0

         np.random.seed(1)
         for hidden_dim in hidden_dims:
             model = TwoLayerNet(input_dim = 9, hidden_dim = hidden_dim , epochs =100 ,learnin
             model.fit(X_train, Y_train, plot_loss = False )
             Y_hat = model.predict(X_test)
             Sigmoid_loss.append(accuracy_score(Y_test, Y_hat))

             model = TwoLayerNet(input_dim = 9, hidden_dim = hidden_dim , epochs =100 ,learnin
             model.fit(X_train, Y_train, plot_loss = False )
```

```
            Y_hat = model.predict(X_test)
            ReLU_loss.append(accuracy_score(Y_test, Y_hat))


        plt.plot(hidden_dims, ReLU_loss, 'rv-')
        plt.plot(hidden_dims, Sigmoid_loss, 'bo--')
        plt.xlabel('hidden layer')
        plt.ylabel('Accuracy')
        plt.title('Classification')
        plt.legend(['ReLU', 'Sigmoid'])
        plt.ylim([0.5,1.0])
```

```
epoch:: 100%|| 100/100 [00:09<00:00, 10.47it/s]
epoch:: 100%|| 100/100 [00:09<00:00,  9.66it/s]
epoch:: 100%|| 100/100 [00:09<00:00, 10.08it/s]
epoch:: 100%|| 100/100 [00:10<00:00,  8.93it/s]
epoch:: 100%|| 100/100 [00:09<00:00, 10.24it/s]
epoch:: 100%|| 100/100 [00:10<00:00,  9.88it/s]
epoch:: 100%|| 100/100 [00:09<00:00, 10.48it/s]
epoch:: 100%|| 100/100 [00:10<00:00,  9.88it/s]
```

```
Out[50]: (0.5, 1.0)
```



Discussion : With the increase in the the dimension of the hidden layer, the accuracy for sigmoid activation slightly increases and then saturates. For the ReLU activation, The accuracy is pretty much constant and at par with maximum accuracy obtained by sigmoid. Thus, ReLU activation can give good accuracy with lower dimension of the hidden layer as compared to sigmoid activation.

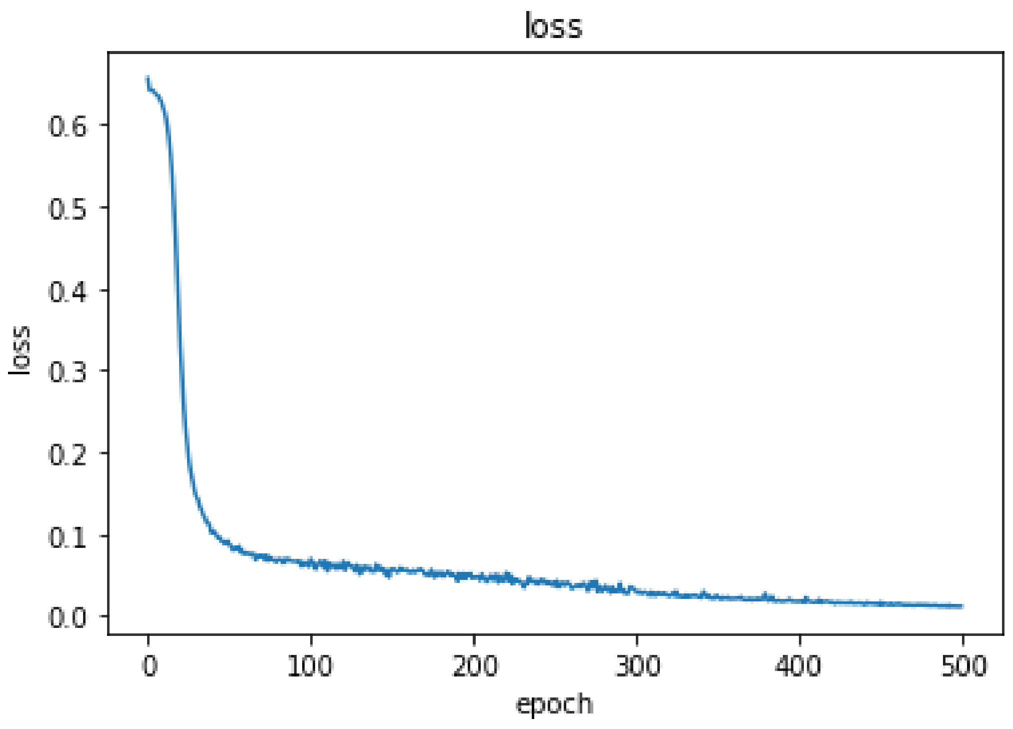
0.0.3 Multi-layer

```
In [7]: from multi_layer import Layer, MultiLayerNet
```

Test-1: Your code should converge for the following architecture. You can change the learning_rate and and number of epochs.

```
In [61]: layers = [Layer(9,10, 'sigmoid'),Layer(10,5,'sigmoid'),Layer(5,1,'sigmoid') ]
         model = MultiLayerNet(layers, learning_rate= 0.01, epochs =500)
         model.fit(X_train, Y_train, plot_loss = True )
         Y_hat = model.predict(X_test)
         print('Test Accuracy',accuracy_score(Y_test, Y_hat))
```

```
epoch:: 100%|| 500/500 [01:09<00:00,  7.25it/s]
```

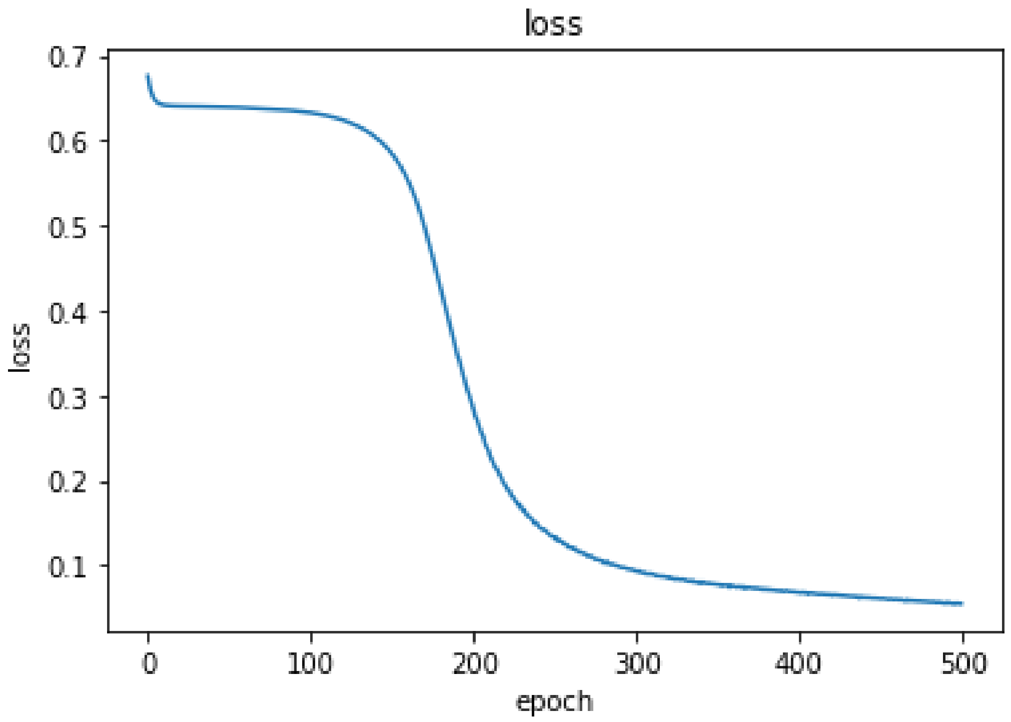


```
('Test Accuracy', 0.9532163742690059)
```

Test-2 Your code should converge for the following architecture. You can change the learning_rate and and number of epochs.

```
In [63]: layers = [Layer(9,20, 'sigmoid'),Layer(20,20,'ReLU'),Layer(20,5,'sigmoid'),Layer(5,1,
         model = MultiLayerNet(layers, learning_rate= 0.001, epochs =500)
         model.fit(X_train, Y_train, plot_loss = True )
         Y_hat = model.predict(X_test)
         print('Test Accuracy',accuracy_score(Y_test, Y_hat))
```

```
epoch:: 100%|| 500/500 [01:27<00:00,  5.88it/s]
```

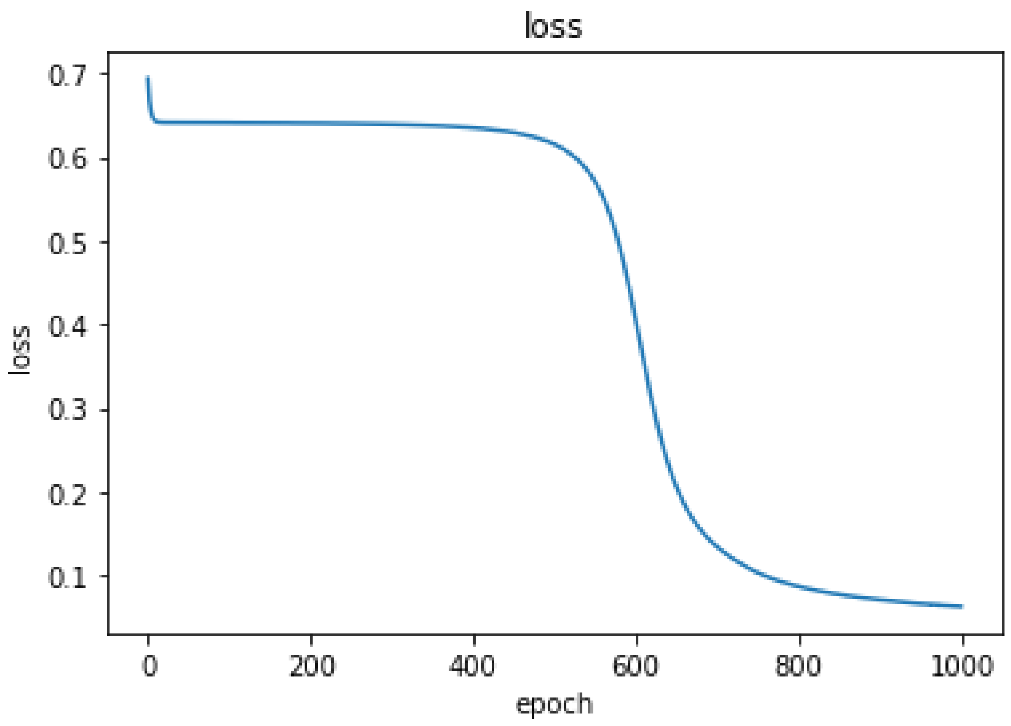


```
('Test Accuracy', 0.956140350877193)
```

Question What is the effect of number of layers if you only use sigmoid activation function?

```
In [66]: layers = [Layer(9,20, 'sigmoid'),Layer(20,20,'sigmoid'),Layer(20,5,'sigmoid'),Layer(5
         model = MultiLayerNet(layers, learning_rate= 0.001, epochs =1000)
         model.fit(X_train, Y_train, plot_loss = True )
         Y_hat = model.predict(X_test)
         print('Test Accuracy',accuracy_score(Y_test, Y_hat))
```

```
epoch:: 100%|| 1000/1000 [02:56<00:00,  5.67it/s]
```



```
('Test Accuracy', 0.9590643274853801)
```

Discussion : With the sigmoid activation in all the hidden layer, the accuracy for sigmoid activation almost equal to the ReLU activation fo rthe previous case (96 %). However, for same learning rate of 0.001, the network with only sigmoid activation takes twice number of epochs (1000) as compared to Relu (500) to converge. This indicates that ReLu activation is computationally more efficient while training.

Other Trials

```
In [33]: layers = [Layer(9,20, 'ReLU'),Layer(20,20,'ReLU'),Layer(20,5,'ReLU'),Layer(5,1,'sigmo
         model = MultiLayerNet(layers, learning_rate= 0.002, epochs =200)
         model.fit(X_train, Y_train, plot_loss = True )
         Y_hat = model.predict(X_test)
         print('Test Accuracy',accuracy_score(Y_test, Y_hat))
```

```
epoch:: 100%|| 200/200 [00:14<00:00, 13.32it/s]
```