EEL 6814
HMW # 2

1-

To design a single hidden layer MLP to classify Wine using the features in the presented dataset, I have used a single hidden layer with relu activation function and a softmax output layer with 3 PE's. I have used categorical crossentropy loss and RMSprop optimizer for my implementation.

Keeping the above said parameters constant, I would like to produce results of different test cases by altering a single hyperparameter to visualize the effect of that single parameter on the performance:

CASE 1: Altering Parameters-No. of hidden layer PE's
No. of hidden layer PE's-26 , learning rate-0.002 , Epoch-1000

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 26)                364
_________________________________________________________________
layer3 (Dense)               (None, 3)                 81
=================================================================
Total params: 445
Trainable params: 445
Non-trainable params: 0
_________________________________________________________________
```

Confusion matrix for training dataset

```
array([[43,  1,  0],
       [ 0, 53,  0],
       [ 0,  0, 36]], dtype=int64)
```

Confusion matrix for training dataset

```
array([[15,  0,  0],
       [ 1, 17,  0],
       [ 0,  0, 12]], dtype=int64)
```

Model accuracy and loss on training and testing dataset

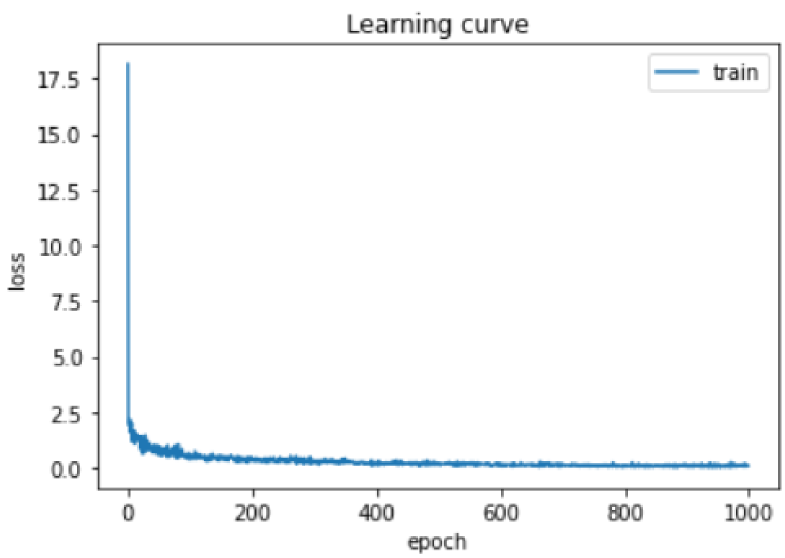
```
score = model.evaluate(X_train,y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Test loss: 0.031354221386687926
Test accuracy: 0.99248123
```

```
score = model.evaluate(X_test,y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.06381204389035702
Test accuracy: 0.9777778
```

Learning Curve



No. of hidden layer PE's-50 , learning rate-0.002, Epoch-1000

```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 50)                700
_________________________________________________________________
layer3 (Dense)               (None, 3)                 153
=================================================================
Total params: 853
Trainable params: 853
Non-trainable params: 0
_________________________________________________________________
```

Confusion matrix for training dataset

```
array([[43,  1,  0],
       [ 0, 53,  0],
       [ 0,  2, 34]], dtype=int64)
```

Confusion matrix for testing dataset

```
array([[14,  1,  0],
       [ 0, 18,  0],
       [ 0,  3,  9]], dtype=int64)
```

Model accuracy and loss on training and testing dataset

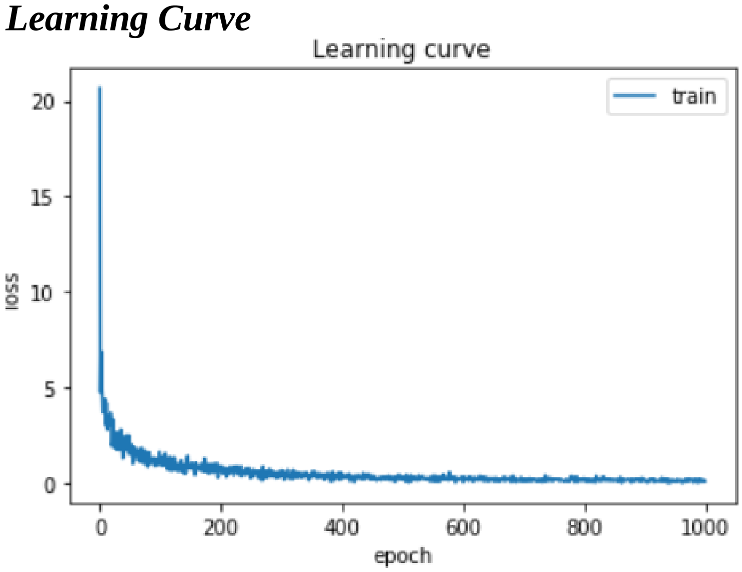
```
score = model.evaluate(X_train,y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 0.10934613475142112
Train accuracy: 0.97744364
```

```
score = model.evaluate(X_test,y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.6302849875556098
Test accuracy: 0.9111111
```

Learning Curve



CASE 2: Altering Parameters-learning rate
No. of hidden layer PE's-26 , learning rate-0.01 Epoch-1000

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
layer1 (Dense)               (None, 26)                364       
_________________________________________________________________
layer3 (Dense)               (None, 3)                 81        
=================================================================
Total params: 445
Trainable params: 445
Non-trainable params: 0
_________________________________________________________________
```

Confusion matrix for training dataset

```
array([[34, 10,  0],
       [ 0, 53,  0],
       [ 0,  1, 35]], dtype=int64)
```

Confusion matrix for testing dataset

```
array([[14,  1,  0],
       [ 0, 18,  0],
       [ 0,  1, 11]], dtype=int64)
```

Model accuracy and loss on training and testing dataset

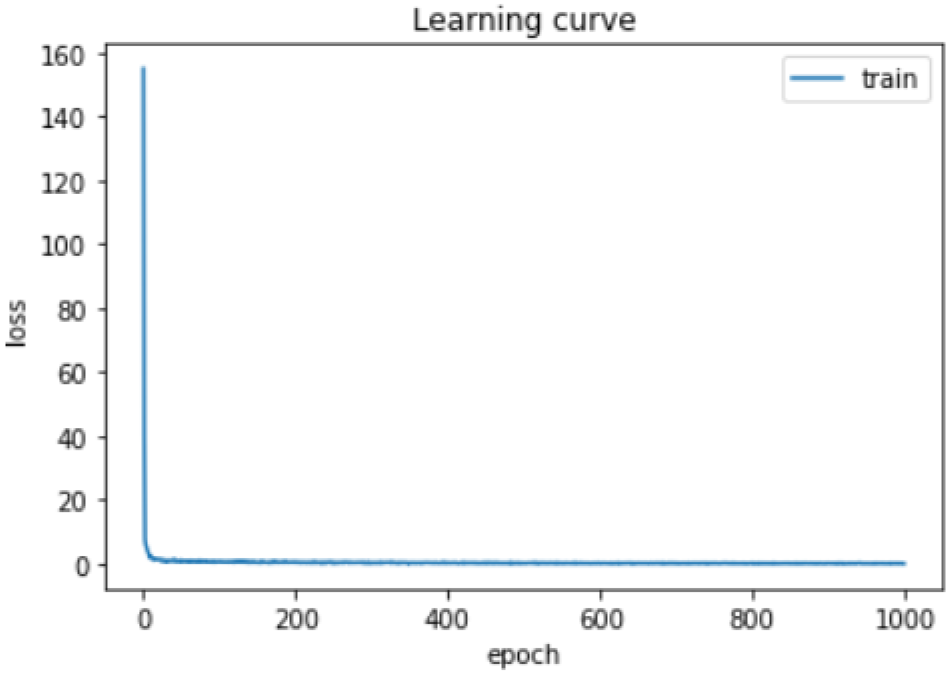
```
score = model.evaluate(X_train,y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 0.3414562958523662
Train accuracy: 0.91729325
```

```
score = model.evaluate(X_test,y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.16603710932864083
Test accuracy: 0.95555556
```

Learning Curve



No. of hidden layer PE’s-26 , learning rate-0.1 Epoch-1000

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 26)                364
_________________________________________________________________
layer3 (Dense)               (None, 3)                 81
=================================================================
Total params: 445
Trainable params: 445
Non-trainable params: 0
_________________________________________________________________
```

Confusion matrix for training dataset

```
array([[ 0, 44,  0],
       [ 0, 53,  0],
       [ 0, 36,  0]], dtype=int64)
```

Confusion matrix for testing dataset

```
array([[ 0, 15,  0],
       [ 0, 18,  0],
       [ 0, 12,  0]], dtype=int64)
```

Model accuracy and loss on training and testing dataset

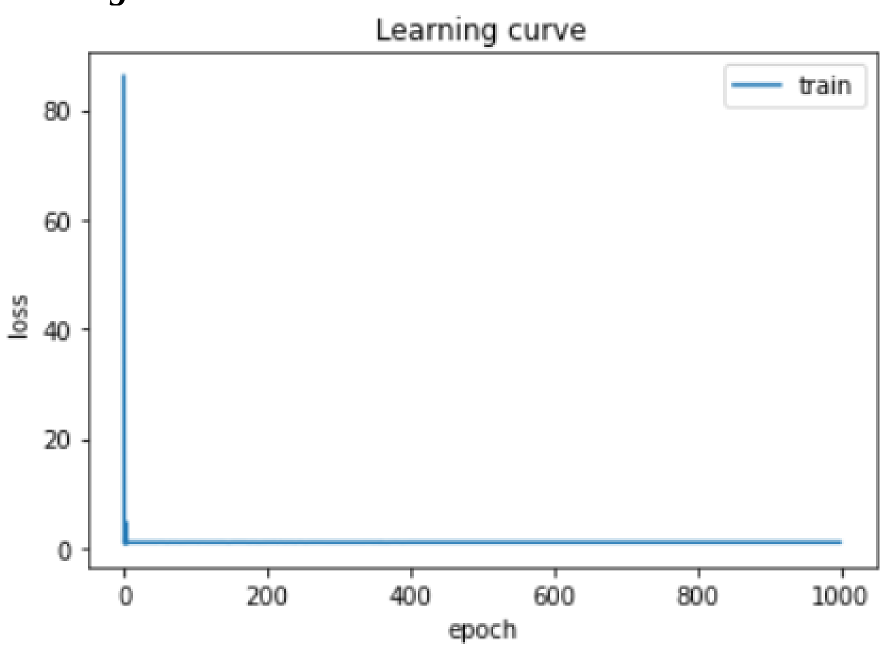
```
score = model.evaluate(X_train,y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 1.0943509662958015
Train accuracy: 0.39849624
```

```
score = model.evaluate(X_test,y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 1.0934284713533189
Test accuracy: 0.4
```

Learning Curve



It can be seen that for a 26 PE hidden layer, when the lr was increased from 0.002 to 0.01 and 0.1 the accuracy of the model on train dataset and the test dataset has fallen. Also, it can be noticed that the model converges previously with a much lower error i.e, 0.0313 while after the lr has been increased the model converges at loss=0.3 and loss=1.094. This scenario arrives as the learning rate is now a bigger value and the weights oscillate at a local minima.

CASE 3 : Altering Parameters-Epochs
No. of hidden layer PE's-26 , learning rate-0.002 , Epoch-300

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 26)                364
_________________________________________________________________
layer3 (Dense)               (None, 3)                 81
=================================================================
Total params: 445
Trainable params: 445
Non-trainable params: 0
_________________________________________________________________
```

Confusion matrix for training dataset

```
array([[44,  0,  0],
       [53,  0,  0],
       [36,  0,  0]], dtype=int64)
```

Confusion matrix for testing dataset

```
array([[15,  0,  0],
       [18,  0,  0],
       [12,  0,  0]], dtype=int64)
```

Model accuracy and loss on training and testing dataset

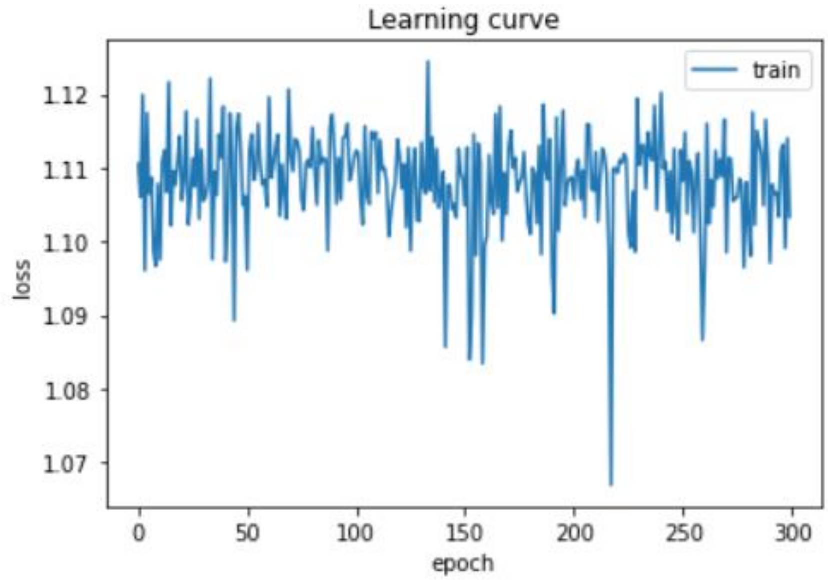
```
score = model.evaluate(X_train,y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 1.0953202516512763
Train accuracy: 0.33082706
```

```
score = model.evaluate(X_test,y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 1.0947960403230454
Test accuracy: 0.33333334
```

Learning Curve



No. of hidden layer PE's-26 , learning rate-0.002 , Epoch-500

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 26)                364
_________________________________________________________________
layer3 (Dense)               (None, 3)                 81
=================================================================
Total params: 445
Trainable params: 445
Non-trainable params: 0
_________________________________________________________________
```

Confusion matrix for training dataset

```
array([[ 0, 44,  0],
       [ 0, 53,  0],
       [ 0, 36,  0]], dtype=int64)
```

Confusion matrix for testing dataset

```
array([[ 0, 15,  0],
       [ 0, 18,  0],
       [ 0, 12,  0]], dtype=int64)
```

Model accuracy and loss on training and testing dataset

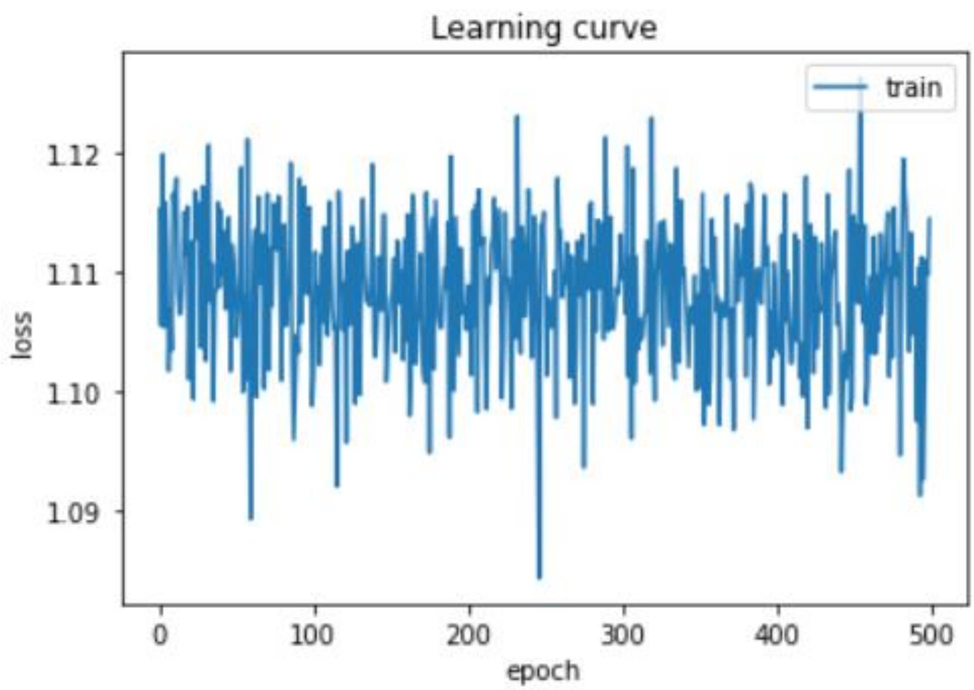
```
score = model.evaluate(X_train,y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 1.0886443925083131
Train accuracy: 0.39849624
```

```
score = model.evaluate(X_test,y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 1.0881434281667073
Test accuracy: 0.4
```

Learning Curve



By altering the epoch from 1000 in the CASE1 model to 300 and 500 respectively, the performance of the model falls drastically. This is because, for the 300 and 500 epoch cases, the training error has not converged yet but the training process has ended. And hence the global minima is not reached yet.

CASE 3 : Altering Parameters-Number of Layers.
No. of layers-2
No. of hidden layer 1 PE’s-26
No. of hidden layer 2 PE’s-13
learning rate-0.002 , Epoch-500

```
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 26)                364
_________________________________________________________________
layer2 (Dense)               (None, 13)                351
_________________________________________________________________
layer3 (Dense)               (None, 3)                 42
=================================================================
Total params: 757
Trainable params: 757
Non-trainable params: 0
_________________________________________________________________
```

Confusion matrix for training dataset

```
array([[41,  3,  0],
       [ 0, 53,  0],
       [ 0,  0, 36]], dtype=int64)
```

Confusion matrix for testing dataset

```
array([[14,  1,  0],
       [ 0, 18,  0],
       [ 0,  0, 12]], dtype=int64)
```

Model accuracy and loss on training and testing dataset

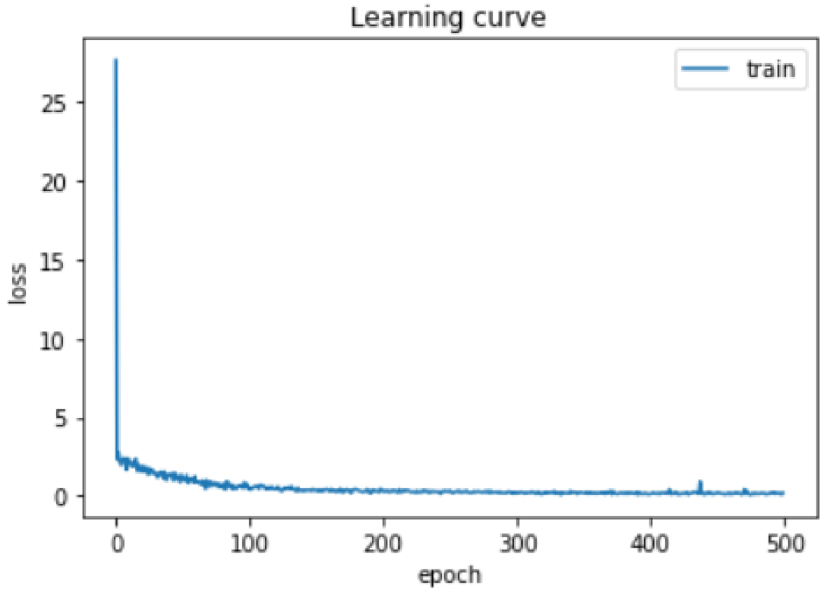
```
score = model.evaluate(X_train,y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 0.10750413807616674
Train accuracy: 0.97744364
```

```
score = model.evaluate(X_test,y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.046632199982802075
Test accuracy: 0.9777778
```

Learning Curve



From the above results its is noticed that when one more hidden layer was added to the network with 13 PE's the similar performance of a single hidden layer with 26 Pe's is achieved in lesser number of epochs.

No. of layers-2
No. of hidden layer 1 PE's-80
No. of hidden layer 2 PE's-50
learning rate-0.002 , Epoch-1000

```
Model: "sequential_24"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, 80)                1120
_________________________________________________________________
layer2 (Dense)               (None, 50)                4050
_________________________________________________________________
layer3 (Dense)               (None, 3)                 153
=================================================================
Total params: 5,323
Trainable params: 5,323
Non-trainable params: 0
```

Confusion matrix for training dataset

```
array([[39,  5,  0],
       [ 0, 53,  0],
       [ 0,  4, 32]], dtype=int64)
```

Confusion matrix for testing dataset

```
array([[14,  1,  0],
       [ 0, 18,  0],
       [ 0,  2, 10]], dtype=int64)
```

Model accuracy and loss on training and testing dataset

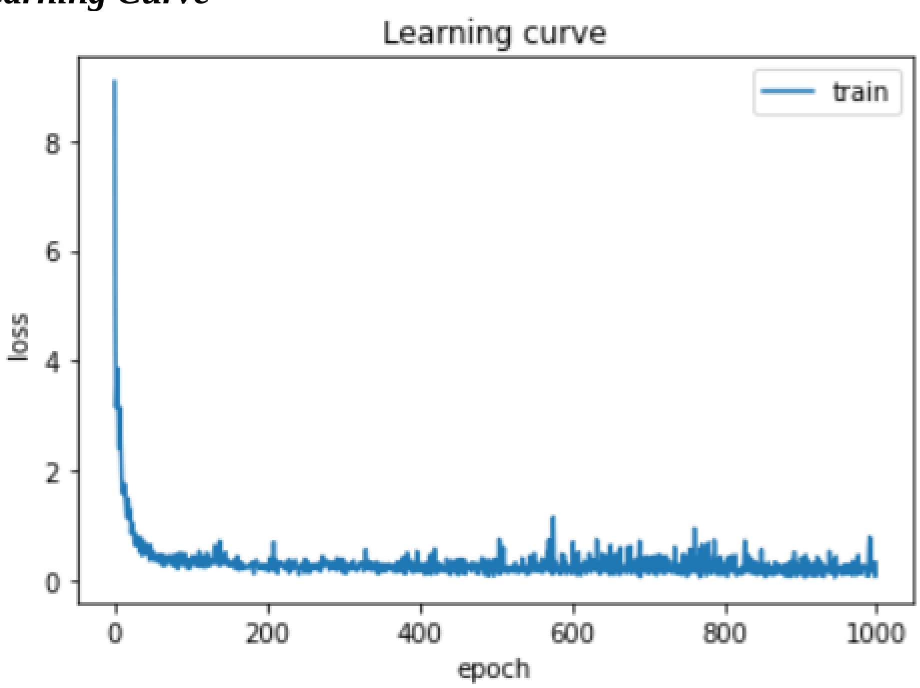
```
score = model.evaluate(X_train,y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 0.2431158723687485
Train accuracy: 0.93233085
```

```
score = model.evaluate(X_test,y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.28191664814949036
Test accuracy: 0.93333334
```

Learning Curve



In conclusion, I was able to design a single hidden layer MLP with 26 hidden PE's for classification of Wine data with an accuracy of 97.77% on the test set and 99.2% on the training set. This is best performance I have achieved on the dataset. Keeping this as the standard parameters, alterations were made and results of the same are produced above.

References:

[1] Kaggle.com. 2020. *NEURAL NETWORKS USING TENSORFLOW ON WINE DATA*. [online] Available at: <https://www.kaggle.com/tejaeduc/neural-networks-using-tensorflow-on-wine-data> [Accessed 8 October 2020].