Machine Learning

Digits Data Set – MLP Classifier (Neural Networks with varying Activation Function)

Build a 10-class classifier based on varying the following:

(a) Number of nodes in the hidden layers.
(b) Tanh, Relu, and Logistic activation functions
(c) Hyperparameters: Momentum term, Early stopping, and Learning Rate

SOLUTION:

CODE:

Please find the code committed for MP Classifier as
01_NN_MLP_Classifier_Vary_ActivationFn_DigitsDataSet_Imp.py

- **MLP classifiers** from sklearn is used to solve this task.
- The digit data set dataset is obtained and normalised so that the data frame has values 0 and 1
- Various MLP classifiers are created applying many permutations obtained on varying :-.
  - **Activation function** as tanh, relu, logistic.
  - **Momentum** as 0.9, 0.8, 0.7.
  - **Hidden layers** as 1-Layer([100]), 2-Layers([100,80]), 3-Layers([100,80,50]), 4-layers([100,80,50,20]).
  - **Learning rate** as constant, adaptive
- **Accuracy** is computed and documented in Result section below with each of these permutations.

RESULT:

Activation Function = tanh

==========

using default momentum = 0.9 having 1 Layer having 100 nodes per layer, Constant Learning Rate
===> Accuracy for MLP Classifier train size 80%= **0.987201**
using default momentum = 0.9 having 2 Layer having [100,80] nodes per layer, Constant Learning Rate
===> Accuracy for MLP Classifier train size 80%= **0.989983**
using default momentum = 0.9 having 3 Layer having [100,80,50] nodes per layer, Constant Learning Rate
===> Accuracy for MLP Classifier train size 80%= **0.988870**
using default momentum = 0.9 having 4 Layer having [100,80,50,20] nodes per layer, Constant Learning Rate
===> Accuracy for MLP Classifier train size 80%= **0.992209**
using default momentum(0.9), 4 Hidden Layers having [100,100,100,100] nodes per layer, Adaptive Learning rate
===> Accuracy for MLP Classifier train size 80%= **0.983306**
using momentum 0.8 with 4 Hidden Layers, having 100,100,100,100 nodes per layer, Constant Learning Rate
===> Accuracy for MLP Classifier train size 80%= **0.986644**
using momentum 0.7 with 4 Hidden Layers, having 100,100,100,100 nodes per layer, Constant Learning Rate
===> Accuracy for MLP Classifier train size 80%= **0.988870**

Activation Function = relu

==========

using default momentum = 0.9 having 1 Layer having 100 nodes per layer, Constant Learning Rate

===> Accuracy for MLP Classifier train size 80%= **0.985531**

using default momentum = 0.9 having 2 Layer having [100,80] nodes per layer, Constant Learning Rate

===> Accuracy for MLP Classifier train size 80%= **0.981636**

using default momentum = 0.9 having 3 Layer having [100,80,50] nodes per layer, Constant Learning Rate

===> Accuracy for MLP Classifier train size 80%= **0.983862**

using default momentum = 0.9 having 4 Layer having [100,80,50,20] nodes per layer, Constant Learning Rate

===> Accuracy for MLP Classifier train size 80%= **0.983862**

using default momentum(0.9), 4 Hidden Layers having [100,100,100,100] nodes per layer, Adaptive Learning rate

===> Accuracy for MLP Classifier train size 80%= **0.982749**

using momentum 0.8 with 4 Hidden Layers, having 100,100,100,100 nodes per layer, Constant Learning Rate

===> Accuracy for MLP Classifier train size 80%= **0.986088**

using momentum 0.7 with 4 Hidden Layers, having 100,100,100,100 nodes per layer, Constant Learning Rate

===> Accuracy for MLP Classifier train size 80%= 0.986088

Activation Function = logistic

==========

using default momentum = 0.9 having 1 Layer having 100 nodes per layer, Constant Learning Rate

===> Accuracy for MLP Classifier train size 80%= **0.968280**

using default momentum = 0.9 having 2 Layer having [100,80] nodes per layer, Constant Learning Rate

===> Accuracy for MLP Classifier train size 80%= **0.957151**

using default momentum = 0.9 having 3 Layer having [100,80,50] nodes per layer, Constant Learning Rate

===> Accuracy for MLP Classifier train size 80%= **0.821925**

using default momentum = 0.9 having 4 Layer having [100,80,50,20] nodes per layer, Constant Learning Rate

===> Accuracy for MLP Classifier train size 80%= **0.100723**

using default momentum(0.9), 4 Hidden Layers having [100,100,100,100] nodes per layer, Adaptive Learning rate

===> Accuracy for MLP Classifier train size 80%= **0.101836**

using momentum 0.8 with 4 Hidden Layers, having 100,100,100,100 nodes per layer, Constant Learning Rate

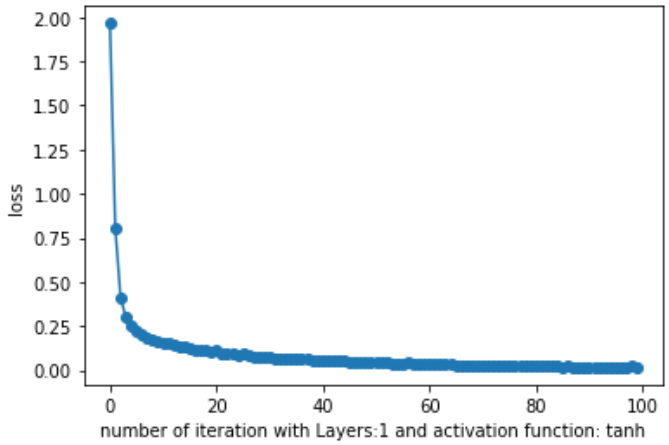
===> Accuracy for MLP Classifier train size 80%= **0.098497**

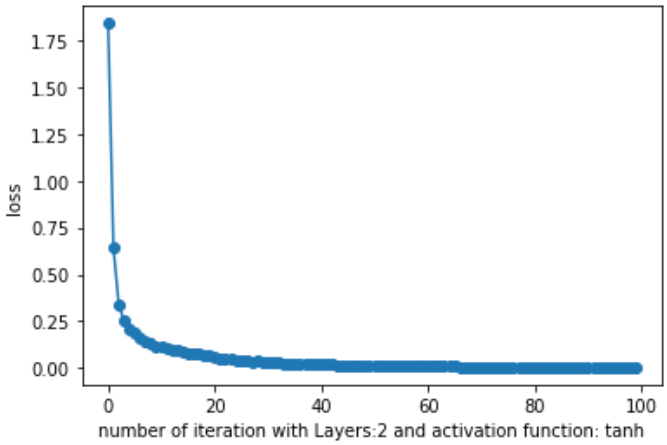
using momentum 0.7 with 4 Hidden Layers, having 100,100,100,100 nodes per layer, Constant Learning Rate

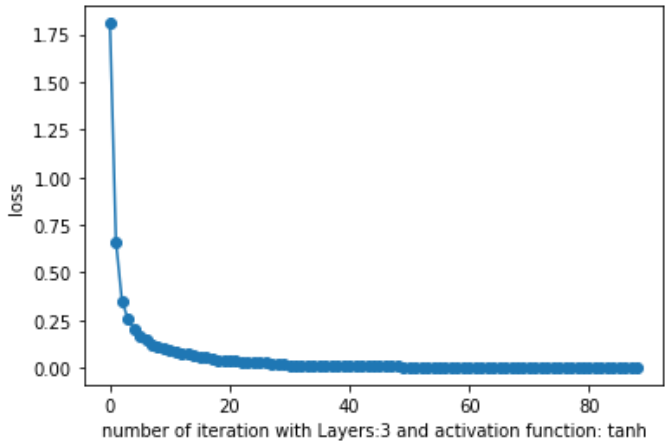
===> Accuracy for MLP Classifier train size 80%= **0.100723**

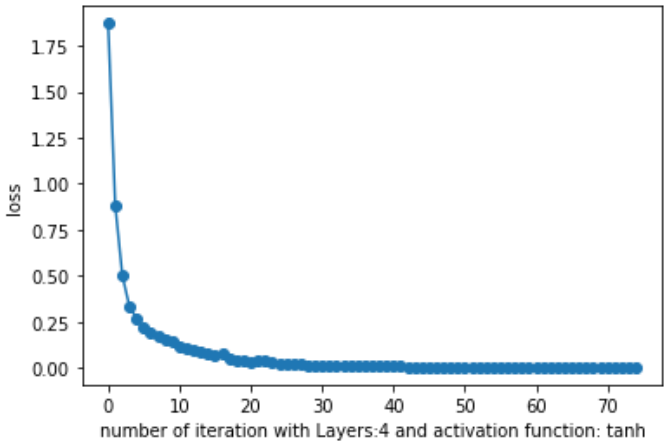
PLOTS:

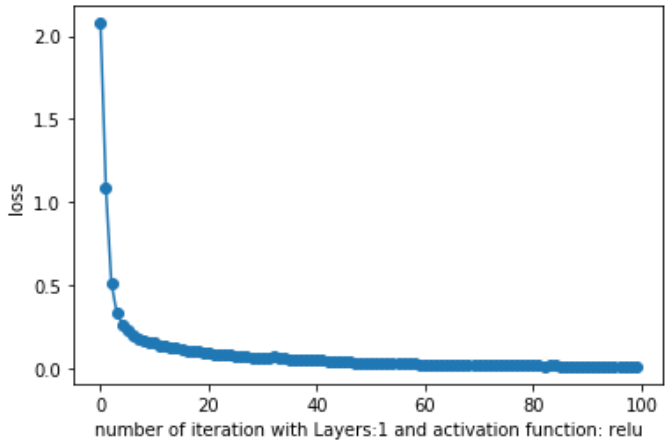
Below are various plots of loss curve with MLP classifier

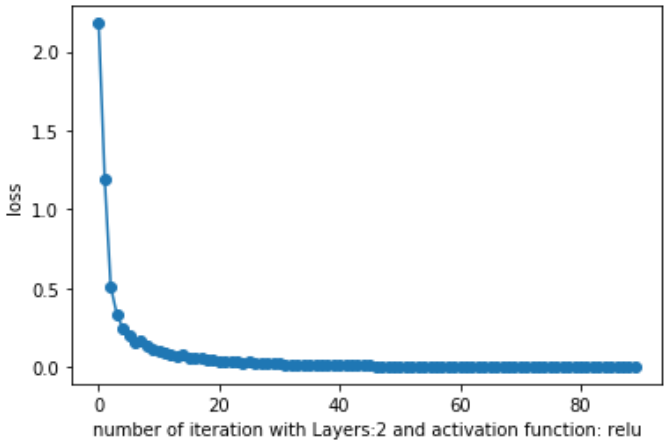


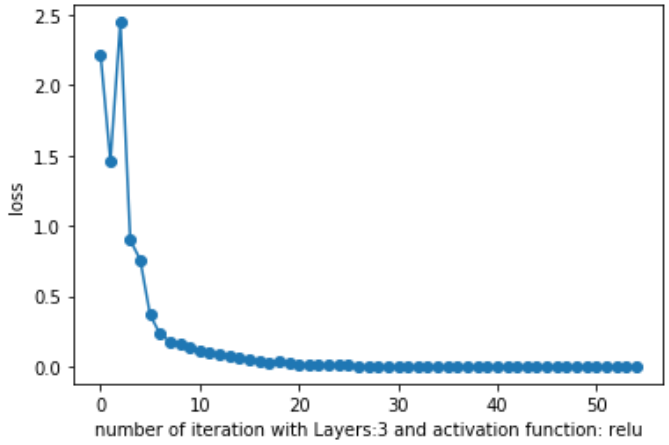


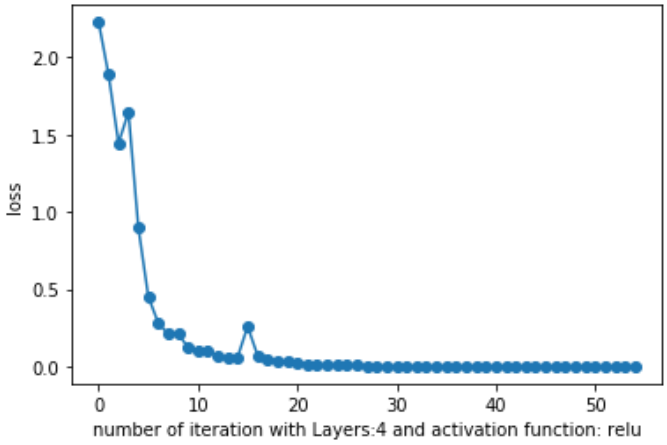


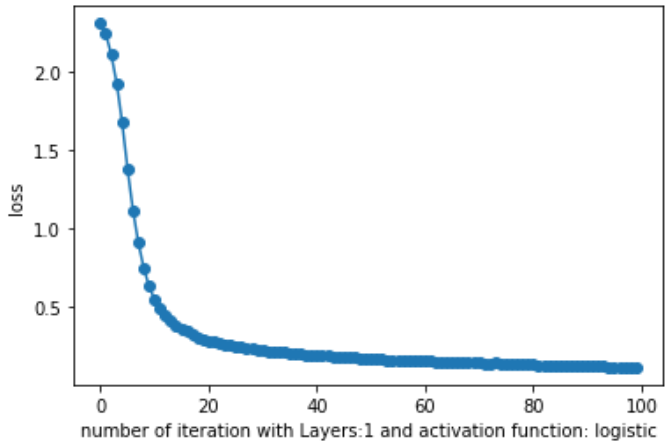


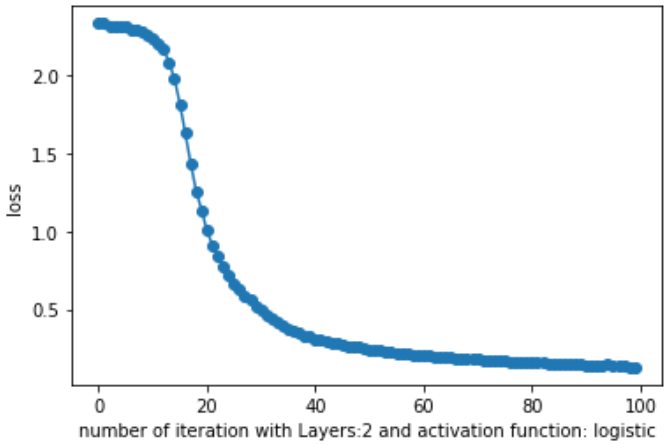


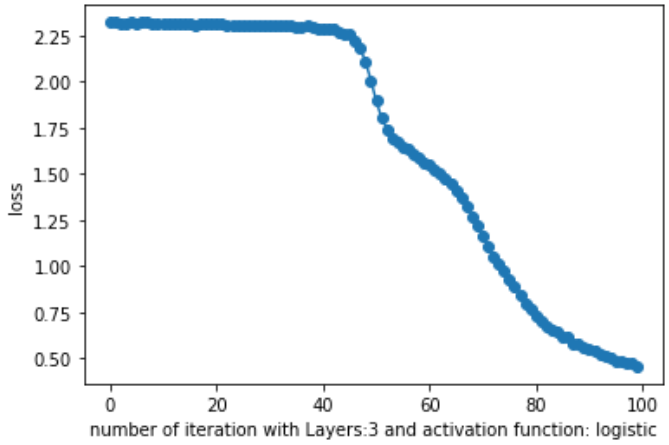


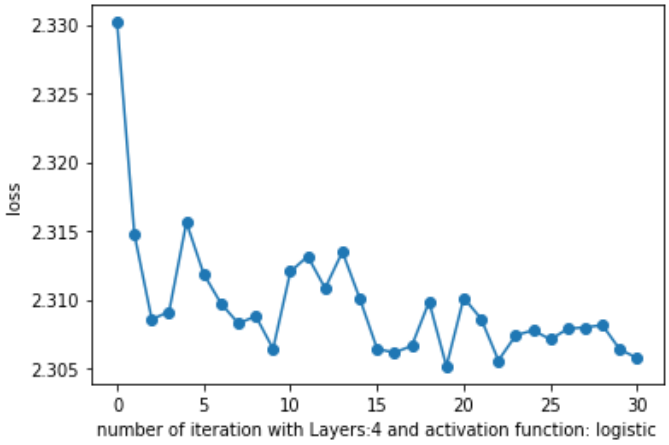












INFERENCE/ANALYSIS:

- **MLP Classifiers** works well and provides a high accuracy of ~98% for the digit's dataset.
- **In digit's dataset** there are 1997 samples with 64 features which forms X and the class variable y represents one of the 10 digit classes.
- From accuracy results we can observe **tanh and relu are performing relatively better** than logistic for the given digit's data set
- With tanh, relu as activation functions, with increase in hidden layer number, the accuracy is improving whereas the reverse is the case with logistic function
- From Loss curve we can observe that with for tanh and relu, the loss function curve converges smoothly as Hyperbolic curve whereas for logistic it is more of a zig zag drop.
- Adaptive learning rate gives better results when compared to constant learning rate

MLP Classifier is applied over digit's dataset completely, the average accuracy of classification is very high. Additionally, when MLP regressor is applied on the Boston Housing Data, Low RMSE results were seen when we used data pre-processing and main contributing features for training our ML Models. Through the exercise of generating many permutations by varying key parameters, we can clearly see that tanh/relu are better performing activation functions for these data sets and with increasing number of hidden layers the performance is getting better.

RESOURCES USED FOR THE ASSIGNMENT:

• **Environment:** Anaconda, Jupyter notebook
• **Software :** Python **Python libraries/modules:** Pandas, Numpy, SkLearn, Scipy, matplotlib, etc