Multi Layer Perceptron Classifier Using Landsat Satellite Dataset

Group F:
Abhinandan De(19CS10069)
Yashica Patodia(19CS10067)

1. Introduction

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function by training on a dataset. Given a set of features and a target, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers.The leftmost layer, known as the input layer, consists of a set of neurons representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation followed by a non-linear activation function - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.

2. Experimental Observation

We have used the MLP Algorithm for designing the classifier.We have tried implementing all these things mentioned in the previous section, in python with and without using in-built library files of python.

2.1. Step-Wise Description of Implementation of Major Functions

2.1.1. data_handling.py

This file handles our input dataset both training and testing dataset.It is also responsible for normalizing dataset.

1) **_init_:**We initialize the dataset here.
2) **Norm:** We normalize the dataset here.
3) **_getitem_ and _len_:** Magic methods to aid the design.

2.1.2. utils.py

This python file contains the code which performs Principal Component Analysis(PCA).

1) **_init_:** It learns the projection matrix using the training set
2) **project:** Project the matrix to a new dimension using the projection matrix.

2.1.3. model.py

This is a generic neural network model with an arbitrary number of hidden layers.

1) **_init_:** This function initializes the network with given params and random weights.
2) **forward:** This function used the feed forward method for future improvement.The activation function relu has been used here.

2.1.4. solve.py

This is the main file which contains the implementations of all the parts mentioned in the assignment.

1) **train_network**:Function to train the neural network using SGD and CrossEntropyLoss.
2) **compute_accuracy**:Function to compute accuracy of the neural network
3) **compute_for_all_networks_and_plot**: Q2 and 3 solver: Trains and computes accuracy for all networks Then constructs the required plots.
4) **pca_n_scatterplot**:Performs a dimension reduction and makes a 2-d scatterplot.
5) **learn_with_reduction**:Applies the algorithm on the reduced dimensions.

3. Detailed Explanation of each part

3.1. PART 1 : Hyperparameters Needed

1) Number of nodes in Input: 36
2) Number of nodes in Output: 7

Justification : Input size is the number of attributes which is 36 and output size is the number of target values which are 7.Other hyperparameters required_learning_rates and required_hidden_layers are provided in the question itself and hence are needed.

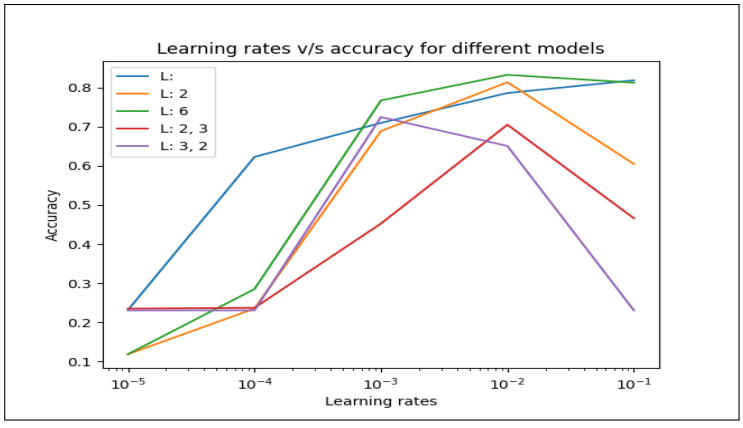
3.2. PART 2: Computation of accuracy

1) The compute_for_all_networks_and_plot trains and compute accuracy for all the parts of this section.The best accuracy model is highlighted in orange colour.

#HiddenLayers\ LearningRates	0.1	0.01	0.001	0.0001	0.00001
0	81.9	78.6	71.0	62.3	23.1
1 (2 nodes)	60.5	81.4	68.9	23.5	11.8
1(6 nodes)	81.3	83.3	76.7	28.5	11.9
2 (2 and 3 nodes)	46.6	70.5	45.2	23.7	23.5
2(3 and 2 nodes)	23.1	65.1	72.5	23.1	23.1

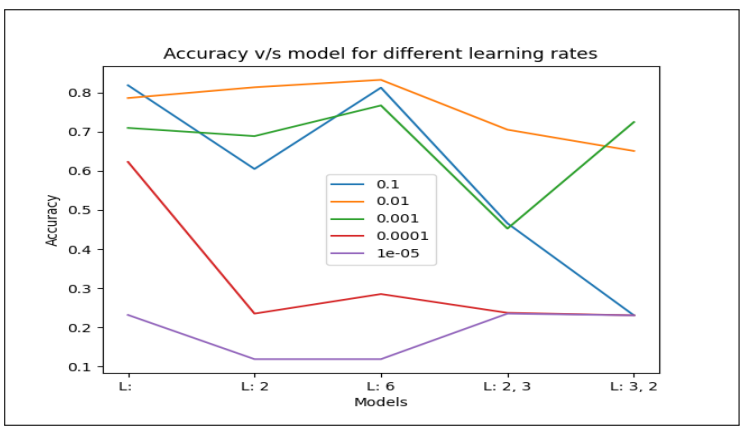
3.3. PART 3: Plotting graphs

1) Learning rate vs accuracy for each model (in one plot)



Explanation: When the learning rate decreases the accuracy decreases as the number of epochs decreases. Therefore we are not giving it sufficient number of iterations(time) to converge to a local minimum and hence the accuracy decreases. If we increase the learning rate the accuracy tends to decrease for some models because for them instead of converging to global minima it diverges.

2) Model vs accuracy for each learning rate (in one plot)



Explanation: Here different lines are representing different models.The dependence of layers(model) on the accuracy is due to variation in the dataset. In our dataset for only one layer we are getting the maximum accuracy and for more complex neural networks the accuracy seems to decrease. Accuracy first increases then decreases for most of the model except for the one with no hidden layers. This is because initially it converges but then after a threshold starts diverging leading to first increase and then decrease in accuracy.

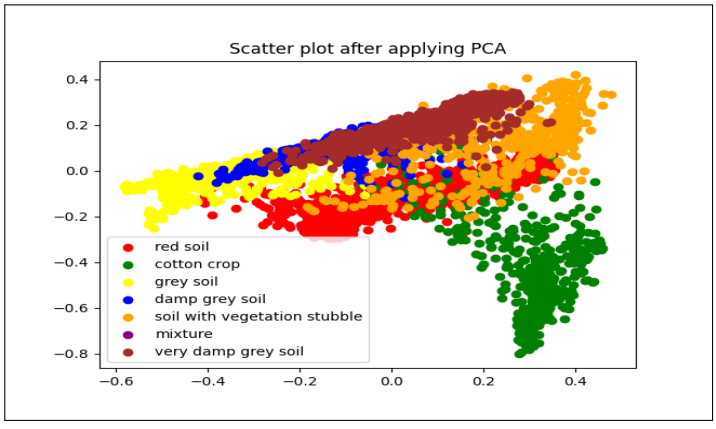
3.4. PART 4: Architecture and Hyper-parameters of the best found model

1) The best found model has **1 hidden layer with 6 nodes at a learning rate of 0.01**.It has the best accuracy of **83.3%** among all the architecture designs.
   Justification: The accuracy first increases then decreases depending on the complexity of the model (number of layers). Hence we can observe that the best accuracy is observed for one hidden layer.

   Secondly the best model is observed for a learning rate of 0.01 because first the accuracy increases with increase in learning rate due to greater number of epochs(iterations) which gives it greater time to converge and then due to further increase in learning rate the accuracy decreases as the starts to diverge.

3.5. PART 5: Dimension Reduction and 2-d Scatter-Plot

1) The feature dimension of the data has been reduced into a two dimensional feature space using Principle Component Analysis (PCA) and the following scatter plot is obtained.In the plot, all data points of a single class have same color and data points from different classes have different colors.



3.6. PART 6: MLP Classification on Reduced Dimension

1) Applying MLP on the reduced space. The best learning rate in **Part 2** is 0.01. The best model accuracy has been highlighted in orange colour.

#HiddenLayers\LearningRate	0.01(Reduced Dimension)	0.01(Original Dimension)
0	79.9	78.6
1(2 nodes)	79.3	81.4
1(6 nodes)	80.9	83.3
2(2 and 3 nodes)	78.6	70.5
2(3 and 2 nodes)	79.3	65.1

Model	Percentage change on reduction(%)	Increase/Decrease
0 hidden layers	1.6	Increase
1 hidden layers(2 nodes)	2.5	Decrease
1 hidden layers(6 nodes)	2.8	Decrease
2 hidden layers (2 and 3 nodes)	11.4	Increase
2 hidden layers(3 and 2 nodes)	21.8	Increase

Comparing: The dimensionality reduction reduced the accuracy of two models (1 hidden layer(2 nodes) and 1 hidden layer(3 nodes)) by a very less percentage (2.5%) and (2.8%) respectively. This is because reduction in dimensions leads to negligible loss in data for the above two models whereas for the other three models there has been a substantial increase in accuracy since the curse of dimensionality is overcome and the more contributions is seen from the features which actually matter.

3.7. Steps to Run The Code

- Download the code into your local machine.
- Ensure all the necessary dependencies with required version and latest version of Python3 are available (verify with requirements.txt) **pip3 install -r requirements.txt**
- Go into the src directory
- Run the solve.py python code using the following command: **python3 solve.py**

Acknowledgements

We are very grateful to Professor Jayanta Mukhopadhyay for giving us an opportunity to learn and apply the topics we have studied on some real-life problem. His guidance, support and teaching helped us a lot for making the skeleton of the project.