Assignment 2

Name : Smitesh Patil

Student ID : 22223696

In this study we have documented our efforts to create a perceptron and multi-layer perceptron from scratch and compare them to pre-defined models of the same from scikit-learn library.

* Creating Model for Perceptron

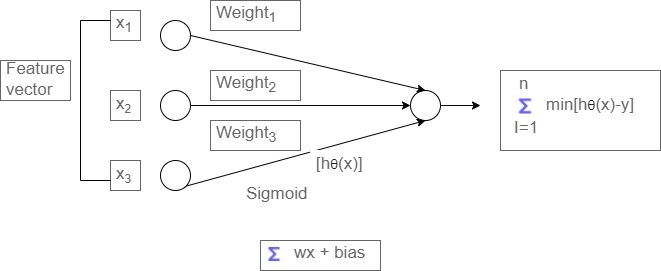
Perceptron is a supervised learning algorithm; it is the basic building block of a neural network. A linear classifier that classifies given input into binary outputs.

1. Structure of a perceptron:

In perceptron we basically take our features or dependent variables as input. For example, if we have 10 features, we have 10 input nodes in out perceptron. These features have certain weights associated with them which determine how strongly that independent variable is associated with the output. The output is calculated by taking a linear combination of these weights and features along with a bias. This output is not a distinct value hence it is fed to an activation function like tanh, sigmoid or relu so that the output is binary.

For this study we had a dataset with many features we selected [‘temp’, ‘drought\_code’, ‘buildup\_index’] as they are most correlated to the dependent variable that predicts wildfire.





So, these 3 features would be input to our model and the hypothesis will be generated on these three input features.

1. Feedforward & adjusting weights:

During the training process we initially randomise weights and feed training samples to the models to generate the hypothesis, then this hypothesis value is compared with the actual value from the training sample and error generated is used to adjust weights and bias. the weights adjustment is done using a learning parameter ‘alpha’. The smaller the alpha value the longer it takes to converge and if the alpha value is high it cannot converge at all, and the model generated is underfitted.

For this study we have taking value of alpha or ‘learning rate’ as 0.00001 and sigmoid as the activation function.

* Creating mode for Multi-Layer Perceptron

A multi-layer perceptron is basically made of many perceptrons connected with hidden layers to develop complex non-linear hypotheses that would be hard for a single perceptron to comprehend. Various architectures of multi-layer perceptron are used for machine learning tasks. Like RNN(Recurrent Neural Networks) for Natural Language Processing and CNN (Convolutional Neural Network) for image classification. For the purpose of this study a normal MLP algorithm is sufficient.

1. Structure and design decision:

In this implementation of multi-layer perceptron, we have taken three features like we did while implementing perceptron for the similar reasons. These three input nodes are then connected to a hidden layer with 25 nodes after applying the activation function. Then the weights of this hidden layer are used to generate our hypotheses after applying the activation function.

Diagram

Description automatically generated

1. Forward Pass:

While going through training examples we initially randomize the weights of all the layers while traversing through all the layers after assigning the weights we also assign them to an activation function and generate the hypotheses.

Hence for this model we traverse the network in following way:

Input -> Activation() -> Hidden\_layer() -> Activation() -> binary\_output

1. Backward Pass: After generating hypotheses we calculate the error using a loss function. In this model we have used Mean Squared Error to calculate the error value. This is later used to calculate the partial derivatives of the previous nodes. In other words, to adjust the weights of a node we need to calculate the difference of the weight with respect to the partial derivatives of the previous nodes. The value of difference is regulated using the learning parameter ‘alpha’. The same process for bias.

* Training for the model developed for perceptron and multi-layer perceptron

Dataset for wildfires was initially divided into two sets training and testing. With 2/3 data for training and 1/3 for testing. Later the training data was divided into 5 bins. For 5 iterations, one bin is assigned for testing and the rest for training. During each iteration a model is trained, after the five iterations we have 5 models. The final hypotheses are calculated using the mean of these 5 model’s hypotheses.

This process is similar for both perceptron and multi-layer perceptron developed from scratch and references from sklearn libraries.

1. Cross-validation training results for Perceptron 2. Cross-validation training results for MLP

Table

Description automatically generated with low confidence Table

Description automatically generated

1. Cross-Validation training for sklearn Perceptron 2. Cross-Validation for sklearn MLP

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

For testing on the test set we have taken the average of all the 5 models trained using cross-validation. The output is between the range 0 and 1 as a probability. It is rounded off to discrete values 0 and 1. The probabilities are kept for plotting the roc curve.

1. Testing results for Perceptron: 2. Testing results for MLP from sratch

Table

Description automatically generatedTable

Description automatically generated

1. Testing results for sklearn Perceptron 2. Testing results for sklearn MLP

Text, table

Description automatically generated with medium confidenceTable

Description automatically generated

1. Multi-layer Perceptron: Similar to the scratch multi-layer perceptron we are taking a hidden layer with 25 nodes and a single output node that should classify the data.

One key difference between the models trained by us and the sklearn models is that the sklearn models output the predictions as discrete values 1 and 0 and not a probability. Hence, while comparing the algorithms we cannot use ROC curve for sklearn models as there are no cut-off values.

* Conclusion

It can be observed that models from sklearn generally performed better than the algorithms we developed from scratch. Structure of both the algorithms Perceptron and Sklearn Perceptron, Multi-layer Perceptron and Sklearn Perceptron are the same as well. It can be inferred that because machine learning libraries are finely tuned based on many years of development, they are more robust and thus produce better results.

For Perceptron :

It can be observed that perceptron developed from scratch wrongly classifies 5 data samples as False positives whereas for sklearn perceptron only 1 gets classified wrongly. Hence Perceptron with sklearn is more robust in detecting false positives.

For Multi-layer Perceptron:

Multi-layer perceptron from scratch performs better than perceptron from scratch but still sklearn MLPClassifier still works better in this case as well. Scratch MLP improves on Scratch Perceptron as it classifies only one sample as False positive but in terms of False negatives it performs worse than sklearn MLP.