A Perceptron-based Approach to Iris Species Prediction

Connor Langlois

Computer Science Department, University of Maine
Orono, ME 04469

connor.langlois@maine.edu

***Abstract* — This paper describes the prediction of iris species using perceptron neural networks trained on the Iris data set. It includes write-ups of the different types of perceptrons and their accuracy.**

***Keywords* — Perceptron: either a single-layer or mutli-layer feed-forward neural network.**

I. Introduction

The purpose of this paper is to describe the prediction of iris species using two different perceptron neural networks: a single-layer and multi-layer perceptron. Each perceptron is trained and evaluated on the Iris data set split into train and test sets.

II. Architecture

The main architecture of the neural networks are based on a single-layer perceptron and a multi-layer perceptron. Each network was trained using the categorical cross entropy loss function, since the problem consists of multiple classes, and the Adam optimizer, due to its practical advantage over alternatives. The Iris dataset was split beforehand into train and test sets. It consists of samples of 4 features, sepal length, sepal width, petal length, petal width, with 1 of 3 classes, I.setosa, I.versicolor, I.virginica.

A. Single-layer Perceptron

The single-layer perceptron is modeled with an input layer of 4 nodes, one for each feature, and an output layer of 3 nodes, one for each class, with a softmax activation function, since the problem consists of multiple classes. This was chosen to be a benchmark to see the performance increase with the multi-layer perceptron. This single-layer perceptron should have a very low accuracy due to its extreme simplicity.

B. Multi-layer Perceptron

The multi-layer perceptron is modeled with an input layer of 4 nodes, one for each feature, 2 hidden layers of 10 nodes with a ReLU activation function, and an output layer of 3 nodes, one for each class, with a softmax activation function, since the problem consists of multiple classes. ReLU was used due to its practical advantage seen in research papers. Theoretically, the 2 hidden layers should be able to learn higher abstract information that the single-layer perceptron could not model. Thus, it is expected to produce a higher accuracy than the single-layer perceptron.

In addition to the change in the network architecture, the multi-layer perceptron also received proprocessed data, that is, data scaled down to the range of -1 to 1. This was done exclusively for the features as the classes are simply represented as a 0 or 1 and thus do not need any scaling. Theoretically, this change should allow the network to train faster as it does not need to give priority to training features that are high in value.

Finally, after each layer in the network, batch normalization was run to further allow the network to train faster and better.

III. Results

As seen in the example run, the multi-layer perceptron performed much better than the single-layer perceptron. The accuracy for the single-layer perceptron and multi-layer perceptron was 53.3% and 100%, respectively. Thus, it can

easily be stated that the hidden layers and preprocessing of data allowed the multi-layer perceptron to much better learn the data.

The 2 hidden layers allowed the multi-layer perceptron to learn higher abstract features about the Iris data set and thus produced a higher accuracy. Furthermore, the data scaling and batch normalization increased the network's learning rate.

Although a 100% accuracy is astounding, this is not likely to continue if the test data set size increased.

IV. Conclusions

Overall, the results gathered follow the expectation/hypothesis, that is, that a more complex network, but not too complex, would be able to learn much more effectively than a simple one. This is likely in many scenarios. However, it is likely that the bias-variance tradeoff effect would come into play here. That is, as the bias of a model increases, its variance will as well, and vise-versa. A simple model has extreme bias, but no variance. A extremely complex model has high variance, but almost no bias. Thus, an extremely complex model is likely to overfit a data set, while a simple model is likely to underfit a data set. Therefore, it is evident that caution must be taken to find a point where the bias and variance are minimized while still producing an effective model. This can be done by using a technique like cross-validation to produce models that are each trained and validated against different "folds" of the train data set. Then, the model that minimizes the loss function (against the sum of error of the validation folds) can be used as the final model and can be run on the test set to produce the resulting accuracy.

Acknowledgment

Connor Langlois wishes to acknowledge Roy M. Turner for his teachings and guidance in Artificial Intelligence.

References

[1] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., Upper Saddle River, New Jersey, United States: Prentice Hall Press, 2009.