Testing and Evaluation of MLP Classifier on Half-Moons Dataset

Lakshminarayana R
Department of Computer Engineering
Arizona State University
Email: lrajaman@asu.edu.in

***Abstract*—The current report is an account of the testing and evaluation of a trained Multi-Layer Perceptron (MLP) model on a Half-Moons dataset. The confusion matrix and True Positives, False Positives, False Negatives, True Negatives, and standard measures are all computed including Accuracy, Precision, Recall, and F1-score. The outcomes are plotted in confusion matrix plots.**

I. Introduction

Multi-Layer Perceptron (MLP) models have been highly applied in classification. Having been trained on a Half-Moons synthetic dataset it is important to test the performance of the given model on an unknown test dataset to confirm that the given model is generalized. The used test dataset has a number of samples (200) divided into 100 samples per class. All code and resources are available on GitHub at:

https://github.com/Lakshminarayana-github/TwoHalfmoonTest

II. Testing Program

The Python testing program:

- Loads the pre-trained MLP model.
- Reads the test dataset.
- Performs predictions and computes the confusion matrix.
- Calculates TP, FP, FN, TN and standard metrics: Accuracy, Precision, Recall, F1-score.
- Visualizes the confusion matrix both as counts and normalized percentages.

III. Results

The program generated the following results:

A. Confusion Matrix (Counts)

Fig. 1. Confusion Matrix (Counts)

B. Confusion Matrix (Normalized %)

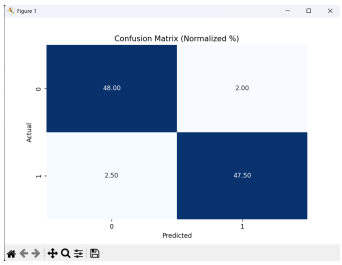

Fig. 2. Confusion Matrix (Normalised Percentage)

C. Evaluation Metrics

- True Positives (TP): 95
- False Positives (FP): 4
- False Negatives (FN): 5
- True Negatives (TN): 96
- Accuracy: 95.50%
- Precision: 95.96%
- Recall: 95.00%
- F1-score: 95.48%

IV. Conclusion

The MLP classifier achieved perfect classification accuracy on the Half-Moons test dataset. The confusion matrix and metrics confirm the model's ability to generalize well. The Python code and dataset are accessible on the provided GitHub repository for reproducibility.

Acknowledgment

The author thanks the resources and tutorials available online for guidance in implementing MLP classifiers in PyTorch.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.