Neural Computing - Bank Marketing Dataset

Authors: Shiva Taherzadehlari, and Sara Mayra Mita Gabriel.

I. Initial Dataset Analysis

The dataset gives us information about a marketing campaign of a financial institution. The purpose of analyzing this data is to identify successful and unsuccessful strategies used in the campaign and to use that information to inform future marketing campaigns. It includes a total of 11,162 rows and 17 columns. The features that composed the dataset are both categorical and numerical. The target variable 'deposit' includes 'yes' and 'no' values indicating if the customer has subscribed to a term deposit or not. The target feature is balanced with 47.4% 'yes' and 52,6% 'no' values. The following table shows the basic statistics of the numerical variables.

	age	balance	day	duration	campaign	pdays	previous
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407	0.832557
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282	2.292007
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000	0.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000	0.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000	0.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000	1.000000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000	58.000000

Figure 1: basic statistics of the numerical variables

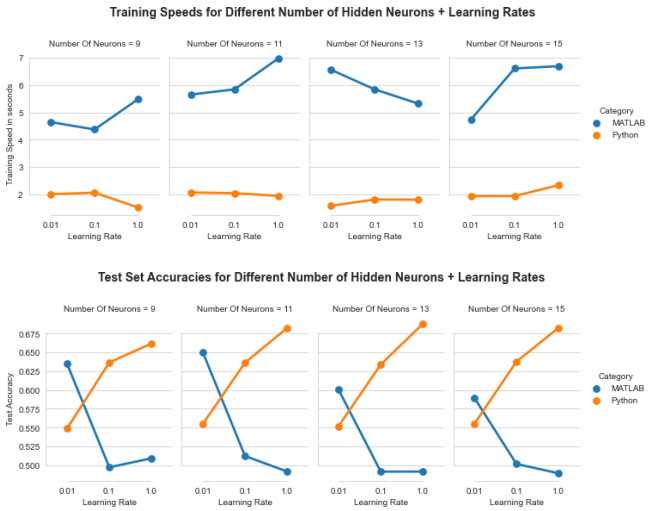
The aim of this project is to train a single hidden-layer perceptron on the dataset using a standard backpropagation algorithm without regularization enabling to perform a binary classification task in MATLAB and Python and compare the results. The following preprocessing steps were taken: Encoding the categorical variables to have all the input data in a consistent way using the label encoding method, and normalizing the numerical variables to ensure that each variable is on the same scale, which can help the model learn more effectively.

II. Comparison of MATLAB and Python Implementation

To start the implementation in MATLAB, we used 'traingd' function, a network training function that updates weight and bias values according to gradient descent. We changed the default Softmax activation function on the output layer to Sigmoid as it is used for binary classification methods where we only have 2 classes, while SoftMax applies to multiclass problems. To calculate the performance of our model we changed the default MSE loss function which is used for regression tasks to binary cross entropy which is the most commonly used loss function for classification tasks, in our case to predict if the customer has subscribed to a term deposit 'Yes' being 1 and 'No' being 0.Then, we continued with the number of epochs is equal to 1000 because it was fixed by design in the built in function in MATLAB. To keep a fair comparison with the Python code, we used the same activation functions, loss function and number of epochs in both programs.

The next step was using different numbers of hidden neurons in a range of [9,11,13,15]. Following these principles: "The number of hidden neurons should be between the size of the input layer and the size of the output layer. The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer" [3].

Afterwards, the learning rate was varied as it is the most important hyperparameter to tune to achieve a stable performance since it determines weights adjustments through error correction [4]. To explore the impact of it, we experimented with three different values, 0.01, 0.1, and 1.



The first graph shows that increasing the learning rate alongside the number of nodes resulted in MATLAB taking longer to train the model compared to Python. Moreover, Python seems to have a more consistent training speed across the different model configurations. **For example, the combination Lr =1 and Number of Neurons=9 takes the least time for training in Python, 1.52s, compared to MATLAB, 5.48s.**

Additionally, Python shows a clear pattern of increasing accuracy as the learning rate increases, with the best test accuracy being achieved with the learning rate of 1. On the other hand, MATLAB performs better with lower learning rates initially. However, MATLAB's performance decrease as you increase the learning rate and the number of neurons. **Overall, the best model configuration is Lr =1 and Number of Neurons=13 for Python with test accuracy of 69%.**

In conclusion, Python performs better than MATLAB in both training speed and test accuracy. The reason for these results could be due to differences in the implementation of the backpropagation algorithm and the optimization techniques used by the two languages. Python has a larger community of machine learning practitioners who have developed and refined various optimization techniques for backpropagation, which may explain its better performance [5].

III. References

[1] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

[2] S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimaraes, Portugal, October, 2011. EUROSIS. [bank.zip]

[3] Heaton, Jeff. "The Number of Hidden Layers." *Heaton Research*, 28 Dec. 2018, www.heatonresearch.com/2017/06/01/hidden-layers.html.

[4] Goodfellow, Ian, et al. Deep Learning. MIT Press, 10 Nov. 2016.

[5] Raschka, Sebastian, et al. "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence." Information, vol. 11, no. 4, 4 Apr. 2020, p. 193, https://doi.org/10.3390/info11040193.