AML Assignment 2

Francesco Stranieri (816551)

f.stranieri1@campus.unimib.it

Introduction

The assignment consists in the prediction of default payments using a *neural network*.

The dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005.

Preprocessing

First of all, I removed the *ID* column, since I did not consider it useful.

Then, once the absence of missing values has been checked, I removed all the 25 duplicate rows.

In order to achieve better results [1], I *standardized* the entire dataset. Finally, I split it into training and validation set. Since the dataset is *unbalanced* (positive examples are 22.18% of total), I applied a stratified splitting with a validation size equal to 0.2.

NN Architecture

The input layer has a number of neurons equivalent to the number of features, which corresponds to 23 in this case. The output layer has a single node, since this is *binary classification* task. For this reason the selected output function was the *sigmoid*, while the loss function was the *binary crossentropy*. For the activaction function, I decided for the *ReLu*.

Furthermore, I chose two hidden layers because they can represent functions with any kind of shape [2]. The "rule-of-thumb" method that I followed for the number of neurons in the hidden layers says that there should be less than twice the size of the input layer. So I chose 32 for the first layer and 16 for the second one.

[1] Shanker, M., Hu, M. Y., & Hung, M. S. (1996). Effect of data standardization on neural network training. Omega, 24(4), 385-397.

[2] Heaton, J. (2008). Introduction to neural networks with Java. Heaton Research, Inc..

Hyperparameters Tuning

In order to find the best configuration of the hyperparameters, I used the *grid-SearchCV* function, included in the *scikit-learn* package [3].

The hyperparameters chosen for the tuning are shown in Table 1.

Hyperparameter	**Values**
`epochs`	25, 50
`batch_size`	32, 64
`optimizer`	*adam*, *sgd*
`init_mode`	*glorot_normal*, *glorot_uniform*
`dropout_rate`	0, 0.3

Table 1: Hyperparameters and their relative values.

The best configuration found was the following: (`batch_size`: 32, `dropout_rate`: 0.3, `epochs`: 50, `init_mode`: *glorot_normal*, `optimizer`: *adam*). The results obtained with this setup, shown in Figure 1, exhibit a loss value of approximately 0.44 and an accuracy of 0.82 circa.

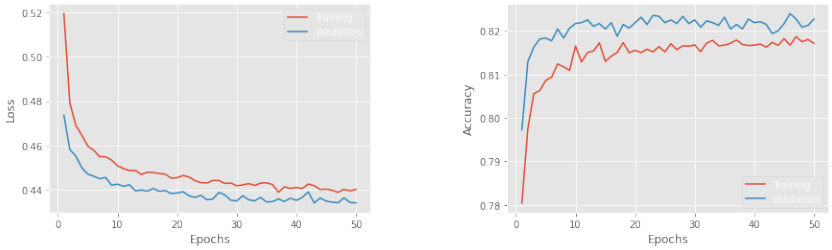

Figure 1: Results with the best configuration.

L1

The next experiment was to introduce the *L1* regularization (with a 0.001 value) on the best configuration. The results obtained are visible in Figure 2 and shown a loss value of approximately 0.48 and an accuracy, for the validation set, up to 0.82.

[3] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

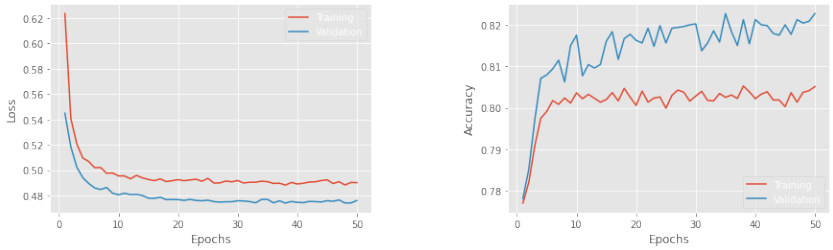

Figure 2: Results with L1.

L2

The final experiment was conducted by introducing the *L2* regularization (with a 0.001 value), always on the best configuration found before. The results obtained, shown in Figure 3, exhibit a loss value of about 0.47 and an accuracy closer to 0.81.

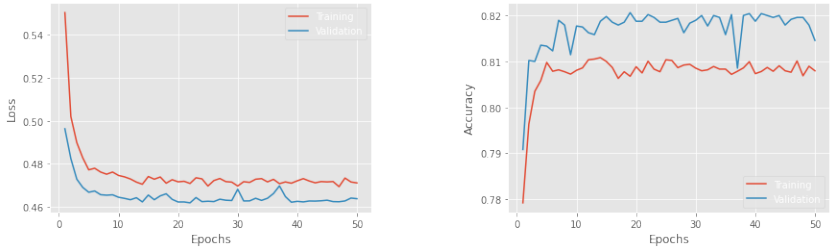

Figure 3: Results with L2.

Considerations

The three models tested show very similar values. However, for the first model the performances on training and validation set are close to each other, in terms of both loss and accuracy. Instead, for the L1 and L2 models, the performances seem more distant, especially for the L1. This can be explained by a higher capacity of *generalization*.

Lastly, also the sum of the weights about the different layers leads to interesting results. In fact, the value for the "baseline" model is about to 19.3, while for the L2 model is approximately 4.24 and for the L1 model is even about to 1.81. This behaviour was expected, since L1 forces some weights to zero [4].

[4]Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., & Knight, K. (2005). Sparsity and smoothness via the fused lasso. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(1), 91-108