**Laboratory Exercise 2: Neural Networks *CI 21/22***Authors: Ramón Mateo Navarro and Benet Manzanares Salor

# Models

With the statement of work as reference, with defined two models:

* **Model1**: logsig for the hidden layer, logsig for the output layer and mean squared error. Its definition and test function can be found at the *model1.m* file.
* **Model2**: logsig for the hidden layer, softmax for the output layer and cross-entropy. Its definition and test function can be found at the *model2.m* file.

The code in the *p2.m* file performs the hyperparameters search and experiments making usage of these models. In addition, it is important to note that both models will use Gradient Descent with Momentum and Adaptive LR as optimization function.

# Hyperparameters search

In order to find the optimal number of epochs, learning rate and momentum (in this document, called hyperparameters) for our experiments, we performed a grid-based search based on the obtained accuracy for the validation set. This process is performed during the execution of the *p2.m* file if the variable *doHyperparametersSearch* has a true value.  
Being aware that the search space scales exponentially in function of the testing values, we limited the amount to three for each hyperparameter (27 combinations). Additionally, we decided to fix the number of hidden units to 50 (the minimum, for a less execution time), using the first train/validation/test partition proposed (80/10/10) and only using Model1. We know that a search with these parameters would have provided a more precise result, but the search space would have been significantly incremented.

On this basis, we performed hyperparameters search for these values:

* Number of epochs: 500, 1000 and 2000
* Learning rate: 0.1, 0.01 and 0.001
* Momentum: 0.1, 0.2, 0.3

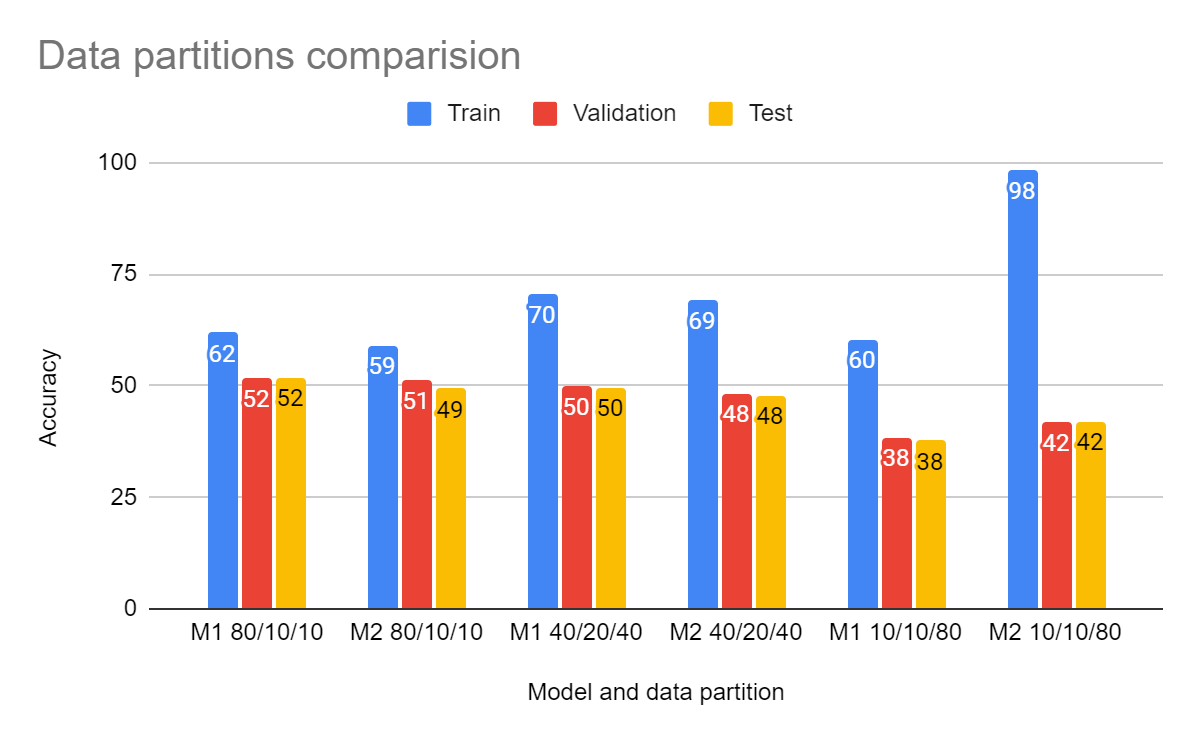
The best validation accuracy obtained has been 43,71%, using the combination of hyperparameters of 1000 epochs, a learning rate of 0,001 and a momentum of 0,2. These values are used for all the following experiments.

# Experiments

Following the statement of work, we tested Model1 and Model2 using multiple numbers of hidden units and train/validation/test partitions. Each test was performed three times and then accuracies for training, validation and test partitions are averaged. The results obtained for each model are exposed and commented below. For reasons of explainability, those comments are divided into data partitions, hidden units and models studies.

## Data partitions study

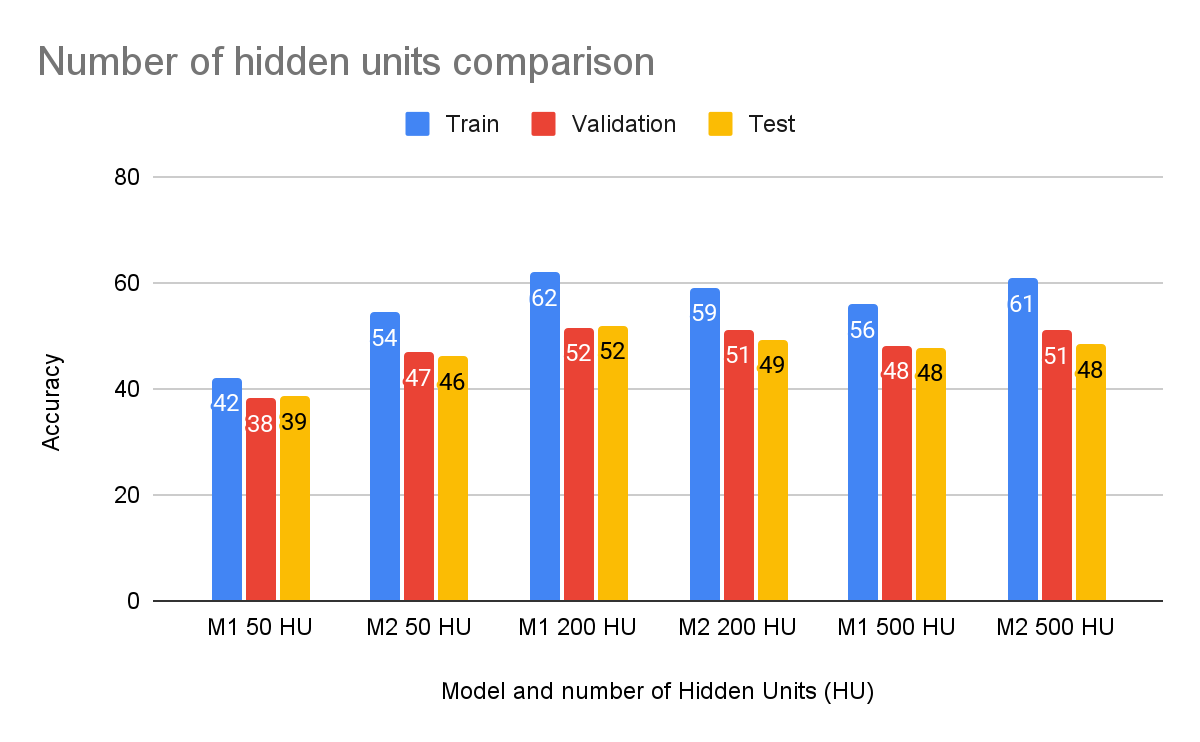
The following figure illustrate the accuracies obtained from testing both models with the three data partitions proposed. The number of hidden units has been fixed to 200 (the intermediate option).



For both models is easy to note that there is an overfitting directly related to the train set size. The smaller the train set, the bigger the overfitting. This is an expected behavior since model generalization suffers if the training data is not general enough and that's usual with reduced amount of data. On that basis, the 80/10/10 provides the results with less overfitting. In addition, it is important to note the huge overfitting obtained for model 2 with the 10/10/80 partition, which was easily noticeable from the Matlab’s performance window even at the epoch 200, when the train loss continued decreasing while and validation/test loss remained stable or even incremented.

## Hidden units study

Equivalently to the data partition study, we fixed the data partition to 80/10/10 (which we consider the more reasonable) and compared the results obtained with different amounts of hidden units. The next figure shows the comparison using both models.



As can be observed, accuracies noticeable improve from 50 HU to 200 HU, but this changes for the case of 500 HU. There are two possible reasons for this. First, that this is a consequence of performing the hyperparameters search only for 200 HU, so settings like the number of epochs are adapted to this amount, and maybe a greater amount is required for the 500 HU model to converge. On the other hand, this is potentially a sign that there is not a significant benefit in incrementing the number of HU to greater than 200, since it only increases training complexity rather than improve performance.

## Models study

For most of the tests performed, the accuracy difference between Model 1 and Model 2 has been minimal (around the 3% in favor to Model 1). The main differences are the better results obtained by Model 2 at the last section of the data partitions comparison (98% training accuracy) and second section of the number of hidden units comparison (+~10% accuracy for all sets). This is more remarkable if we take into account that hyperparameters have been tuned specifically for Model 1 (probably this is the reason why Model 1 gets the 3% extra in the rest of tests). In summary, Model 2 seems to be a better model and would have obtained the best results for all the tests if a specific hyperparameter search had been performed. The reason for this is probably that the multi-output+softmax is a better approach for multi-class classification task than the single output+logsig approach.

# Conclusions

In this practice, we have performed a hyperparameters search and multiple experiments with two multilayer perceptron models. These experiments have led us to first-hand observe how models, training and data settings affect the results. One of the most notorious effects is the relation between overfitting and the size of the training set. This shows how the generalization capabilities of neural networks are highly data-dependent and gives an intuition of how having more training data is better (if its distribution is similar to the expected in the wild/test). Another is how incrementing the number of neurons doesn’t improve results forever, since architectures have their limits. Additionally, there is not a one-fits-all configuration for neural networks models, they are complex system which require specific settings. Furthermore, the not magnificent accuracy obtained (around 50 percent) is another proof that using multilayer perceptrons with flatten 2-dimensional data (in this case, silhouettes) is not the ideal approach. Probably, a convolutional neural network would obtain better results since it would be able to take advantage of the locality information (knowing which “pixels” are closer to others). Finally, we have observed how the output layer can significantly change the outcome and requires special attention when defining the training.