ML 2024 Project

Authors:

- Sounak Mukhopadhyay (684352) - Masters in Computer Science
  s.mukhopadhyay@studenti.unipi.it
- Charchit Bansal (681742) - Masters in Computer Science (ICT)
  c.bansal@studenti.unipi.it

Project Type: A

Team Name: Traders

Objectives

The objectives of this project are:

• Implementing a neural network from scratch, in python without using standard libraries

• Testing different models on Monk Dataset and CUP Dataset

Method

We implement:

- A fully connected feed-forward neural network;
- Backpropagation with Batch Gradient Descent;
- Different activation functions;
- Momentum technique;
- Tikhonov regularization;
- Different stop conditions.

Model

We decide to divide the neural network into layers and units to better manage the model.

- Each layer has a set of units
- Each units contains its weights, value and deltas for backpropagation.

The model permit to add a single layer at time, with its activation function and its number of units.

Activation functions and Momentum

For activation function we choose to implement :

- Sigmoid
- Identity
- Tanh
- Relu

For each activation functions we implement also the derivative.

We also implement Momentum for the adjustment of weights during Back-propagation phase.

Regularization and stop criteria

- The dataset could present some noise, for that we decide to implement regularization.
- In particular we focused on Tikhonov regularization, also known as Ridge regression.
- We decide to fix the epochs on the training phase to stop the iterations.
- This isn’t a good practise, so we decide to implement an early stop criteria.
- The training phase stop when the difference between the previous iteration and the current iteration is smaller than an error_rate.

Initialization of the model

- The model required in input the number of units for the input layer.
- Then you can add as many levels as you want, specifying the number of units and the activation function for each layer.
- For each layer, a specific number of units are created.
- For each unit, a certain number of weights (equal to the number of units of the previous layer) are initialized randomly using fan-in method, with a uniform distribution.

Validation schema

For the validation part we implement grid search algorithm and k-fold cross-validation.

For this part of the project:

- With grid search algorithm we have selected the model with best evaluation (score or MSE), using different combinations of hyper-parameters.
- We evaluate the performance of the final model that we choose (k-fold cross-validation).

The results reported are the average of 10 iterations.

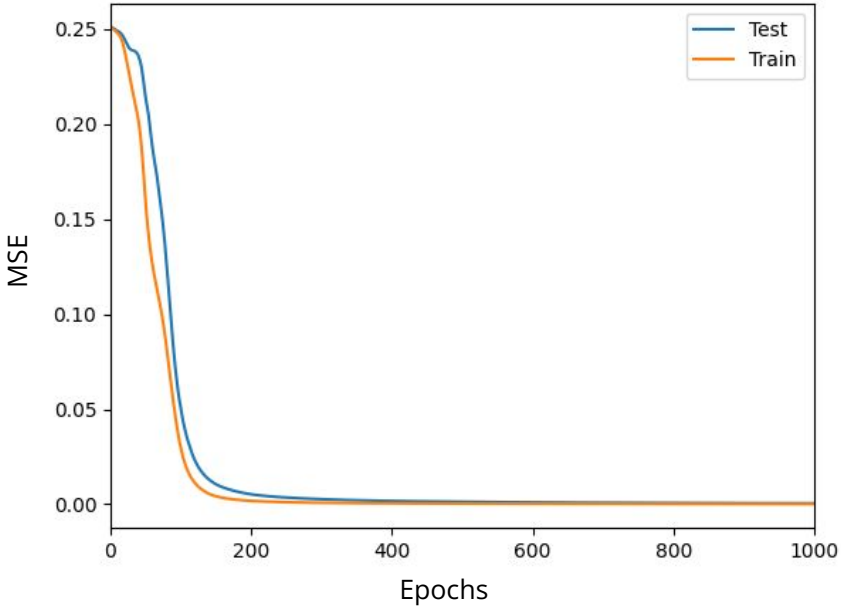
Monk's results

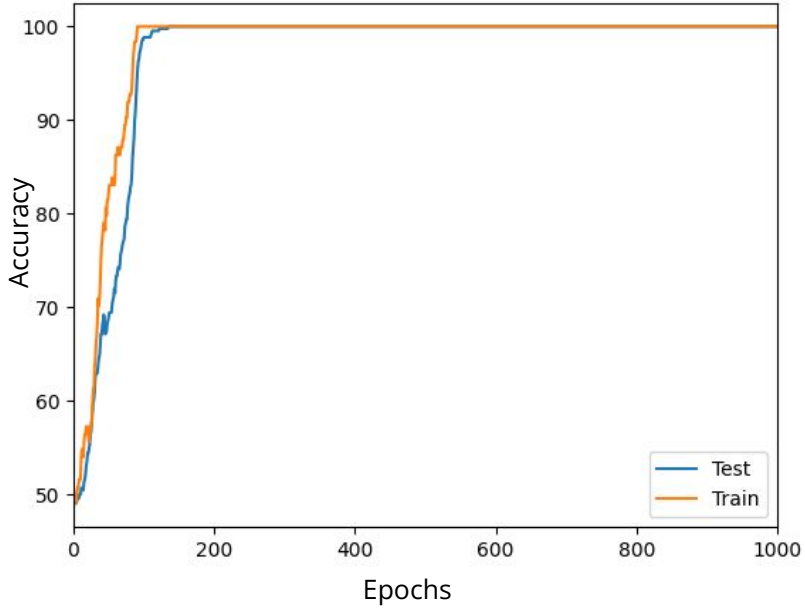
For monk's datasets we use:

- Relu as activation function for hidden layer
- Sigmoid as activation function for output layer.

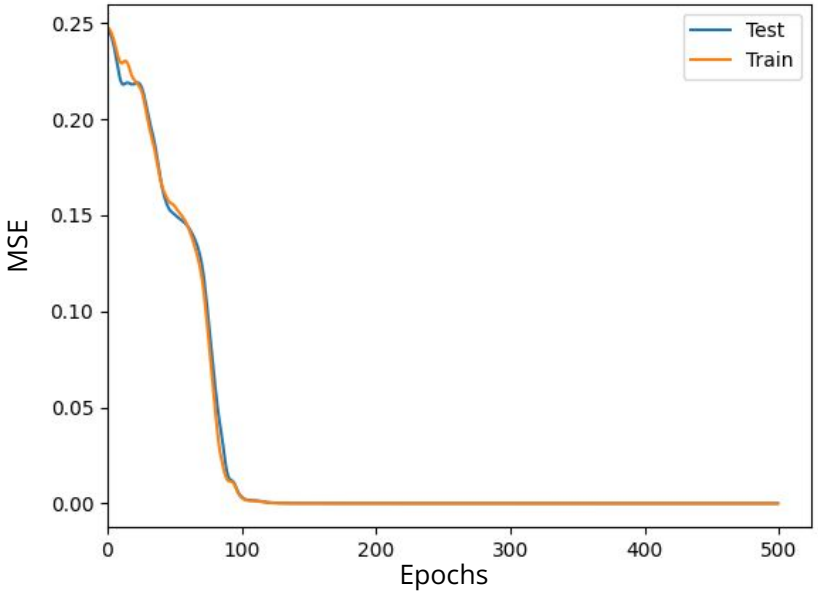
Task	Layers	Units	Eta	Alpha	Lambda	MSE (TR, TS)	Accuracy (TR, TS)
Monk 1	1	4	0.75	0.85	0	0.01, 0.02	99%, 97.8%
Monk 2	1	6	0.8	0.95	0	0.01, 0.009	100%, 100%
Monk 3	1	4	0.75	0.75	0	0.01, 0.04	98.2%, 94.3%

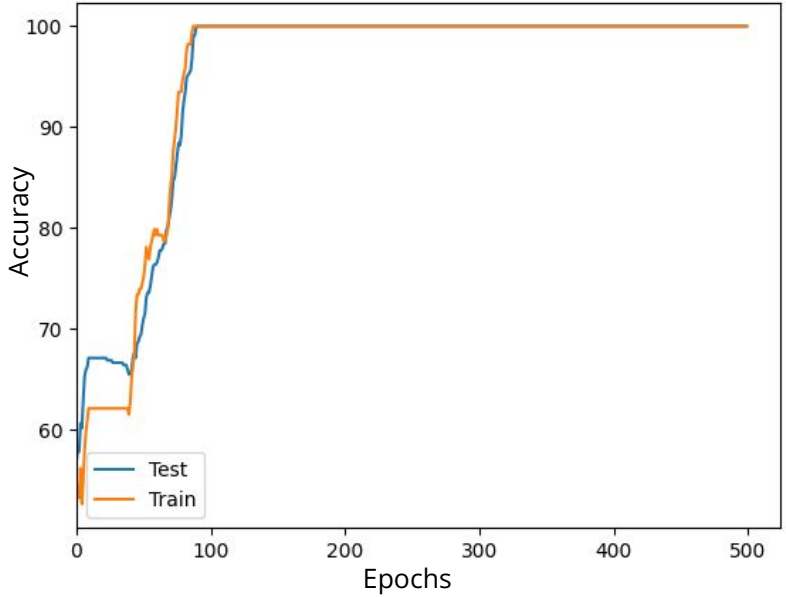
Monk Results 1



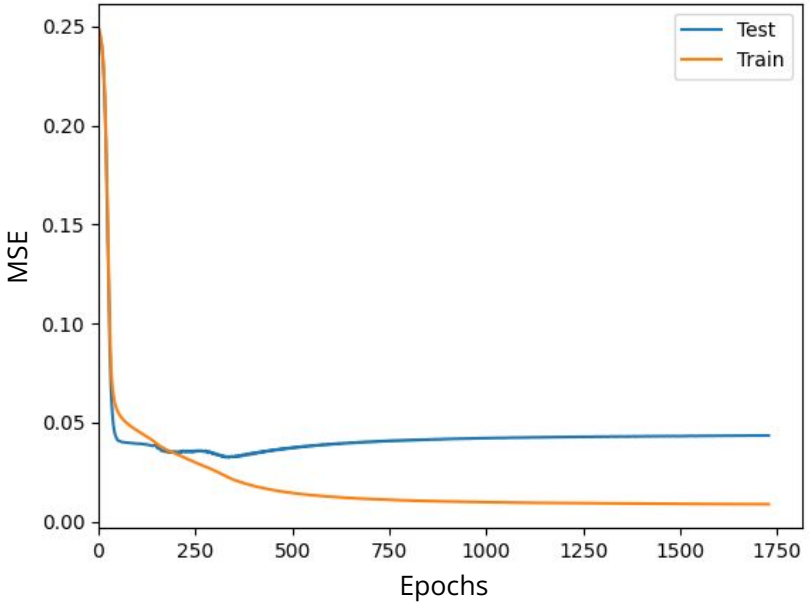


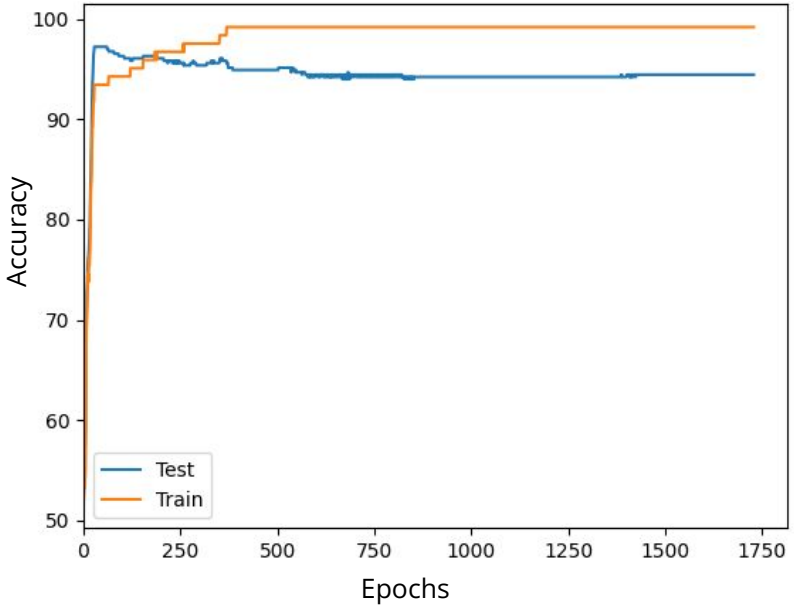
Monk Results 2





Monk Results 3





CUP validation schema

The CUP dataset doesn’t present a dataset test, so we decide to extract an internal test from the train dataset.

The internal test is composed by 20% of data from training set.

Then we use a 5-fold cross validation for the phase of model assessment.

To predict the blind set, we decide to apply a retraining.

Hyper Parameters

Hyperparameters	Range
Layers	1-2
Units	10, 20, 30
Activation functions for hidden layers	Tanh, Sigmoid
Eta	0.3 - 0.9
Alpha	0.5 - 0.9
Lambda	0.001 - 0.009

Grid search

During this phase we try different combinations of hyperparameters.

We decide for first to use Identity function like activation functions of output layer.

We decide to use different range for weights initialization, we fix it from -0.8 to 0.8.

All our experiments were performed on Intel i7 CPU of 13th generation.

Final Model

We choose the final model that perform better on training and internal test set.

We decide to use 1 hidden layer with 30 units.

Other parameter:

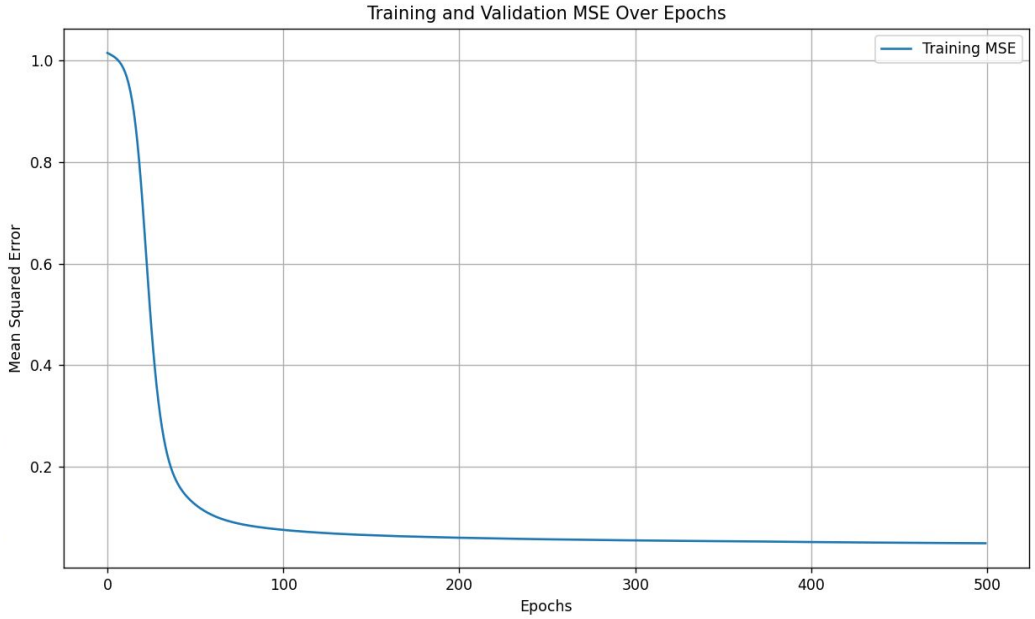
- eta = 0,1;
- Alpha = 0,6;
- Lambda = 0,006.

For the activation functions of hidden layers we decide to use Tanh, because it maps also on negative numbers, respect to Sigmoid.

We choose Identity as activation function of output layer.

Results of Cup

Task	Mee (TR, VL, TS)
Training set	1.67
Validation	1.23
Internal test set	2.50



Analysis

The results reported are the average of 10 iterations, because the weights initialization is random.

For the final model we choose Tanh, which is slower than Sigmoid, but is more accurate.

From this project we learn more about the chosen of hyperparameters space, because if we try many range of parameters, the time required to compute the results is very high.

Conclusions

During development we became increasingly familiar with neural networks.

In particular, we liked implementing a neural network from scratch, and this allowed us to better learn the theory studied during the course.

As regards future work: it would be interesting to see how the results change with online approach on backpropagation.

The Blind Test Results are saved into *Traders_ML-CUP24-TS.csv*.