ML 2023 PROJECT

Using Beamer

Matteo Rofrano, Francesco Lanci Lanci
Group name: Golia

Master degree in Data Science and Business Informatics
emails:
matteo.rofrano@studenti.unipi.it
f.lancilanci@studenti.unipi.it

PROJECT TYPE B

Outline

Introduction and Objectives

In the following project, we will test and compare different models, structures and hyperparameters settings in a scientific way in order to find the best model configuration for the task at hand.

The models and algorithms used are:

- Sklearn Neural network with SGD for MONK and CUP
- Keras Neural network with SGD for MONK and CUP
- Pytorch Neural network with SGD for MONK and CUP
- Sklearn Support Vector Machine for MONK and CUP
- Sklearn KNN for MONK and CUP
- XGBoost for MONK and CUP

For the cup we just used the L2 regularization since we noticed that the different NNs tend to not overfit easily. We used MSE for the training and MEE for the cup evaluation.

Method

- Pandas, NumPy, and Sklearn for the preprocessing (transformation and data splitting)
- Sklearn, Keras, PyTorch and xgboost functions for the models
- Sklearn for the model selection (explorative randomized search for the NNs and detailed grid search)
- Matplotlib for the plots

Method

Implementation	Sklearn	Keras	PyTorch
Architecture	sequential MLP	sequential MLP	sequential MLP
N layers	1	1	1
Activation function	tanh/sigmoid	tanh/sigmoid	tanh/sigmoid
Activation function output	sigmoid	sigmoid	sigmoid
Training algorithm	SGD mini-batch	SGD mini-batch	SGD mini-batch
Regularization	L2 only monk3	L2 only monk3	L2 only monk3
Initialization	Glorot	Glorot	Glorot

Table: Structure NNs Monk

Implementation	Sklearn	Keras	PyTorch
Architecture	sequential MLP	sequential MLP	sequential MLP
N layers	1/more	1/more	1/more
Activation functions	tanh/sigmoid	tanh/sigmoid	tanh/ReLU
Activation function output	linear	linear	linear
Training algorithm	SGD mini-batch	SGD mini-batch	SGD mini-batch
Regularization	L2	L2	L2
Initialization	Glorot	Glorot	Glorot

Table: Structure NNs CUP

Method

SVM

- C
- Kernel
- Gamma
- Epsilon

KNN

- N neighbors
- Weights
- Metric

XGBoost

- N estimators
- Max depth
- Learning rate
- Gamma regularitation
- Lambda regularization

MONK results

Task	Parameters	MSE (TR/TS)	Accuracy (TR/TS)
MONK 1	C = 1, epsilon = 0.3, gamma = 0.3, kernel = polynomial	0.0805/0.0993	100%/100%
MONK 2	C = 10, epsilon = 0.4, gamma = 0.1, kernel = rbf	0.1278/0.1708	100%/80.56%
MONK 3	C = 1, epsilon = 0.3, gamma = 0.4, kernel = polynomial	0.0688/0.1102	100%/93.52%

Table: Results SVM

Task	Parameters	MSE (TR/TS)	Accuracy (TR/TS)
MONK 1	N neighbors = 9, weights = uniform, metric = euclidean	0.1250/0.1685	86.30%/77.31%
MONK 2	N neighbors = 23, weights = distance, metric = euclidean	0/0.1268	100%/78.70%
MONK 3	N neighbors = 5, weights = distance, metric = manhattan	0/0.0950	100%/88.66%

Table: Results KNN

Task	Parameters	MSE (TR/TS)	Accuracy (TR/TS)
MONK 1	N estimators = 200, lr = 0.01, max depth = 5, gamma = 0.2, lamda = 0.5	0.0133/0.0218	100%/100%
MONK 1	N estimators = 300, lr = 0.001, max depth = 10, gamma = 0, lamda = 0.1	0.1396/0.1855	100%/81.48%
MONK 1	N estimators = 300, lr = 0.3, max depth = 25, gamma = 0.2, lamda = 0.1	0.0129/0.0376	100%/93.06%

Table: Results XGBoost

MONK results

Task	Parameters	MSE (TR/TS)	Accuracy (TR/TS)
MONK 1	Units = 4, f = tanh, lr = 0.05, momentum = 0.9, batch size = 20, epochs = 200	0.001215/0.00246	100%/100%
MONK 2	Units = 4, f = tanh, lr = 0.1, momentum = 0.8, batch size = 20, epochs = 100	0.000609/0.000787	100%/100%
MONK 3	Units = 4, f = tanh, lr = 0.1, momentum = 0.9, batch size = 20, epochs = 200	0.0565/0.0455	93.08%/96.76%
MONK 3 (reg.)	Units = 4, f = tanh, lr = 0.01, momentum = 0.5 , batch size = 20, epochs = 200, l2 = 0.001	0.0735/0.0564	93.44%/97.22%

Table: Results Sklearn MLP

Task	Parameters	MSE (TR/TS)	Accuracy (TR/TS)
MONK 1	Units = 5, f = tanh, lr = 0.9, momentum = 0, batch size = 2, epochs = 100	0.00014/0.00019	100%/100%
MONK 2	Units = 5, f = tanh, lr = 0.1, momentum = 0.8, batch size = 2, epochs = 100	0.00020/0.00030	100%/100%
MONK 3	Units = 6, f = tanh, lr = 0.9, momentum = 0.8, batch size = 2, epochs = 200	0.0643/0.0375	93.55%/96.22%
MONK 3 (reg.)	Units = 4, f = sigmoid, lr = 0.1, momentum = 0, batch size = 2, epochs = 100, l2 = 0.01	0.1042/0.0866	93.44%/97.22%

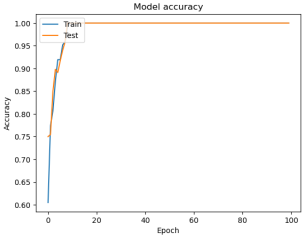
Table: Results Keras NN

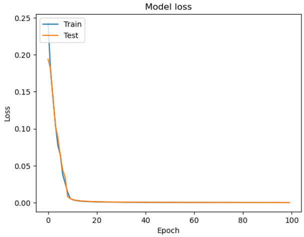
Task	Parameters	MSE (TR/TS)	Accuracy (TR/TS)
MONK 1	Units = 4, f = tanh, lr = 0.3, momentum = 0.5, batch size = 5, epochs = 100	0.00088/0.00162	100%/100%
MONK 2	Units = 5, f = tanh, lr = 0.3, momentum = 0.6, batch size = 5, epochs = 150	0.00020/0.00032	100%/100%
MONK 3	Units = 5, f = tanh, lr = 0.1, momentum = 0.7, batch size = 30, epochs = 80	0.0466/0.0362	94.26%/96.76%
MONK 3 (reg.)	Units = 6, f = tanh, lr = 0.01, momentum = 0.7, batch size = 5, epochs = 100, l2 = 0.01	0.0680/0.0546	93.44%/97.72%

Table: Results PyTorch NN

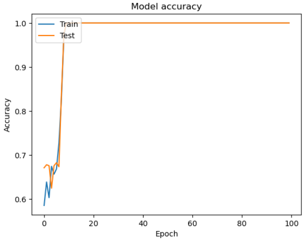
MONK results

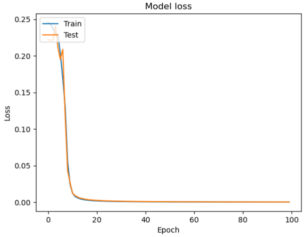
MONK 1





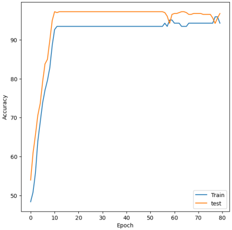
MONK 2

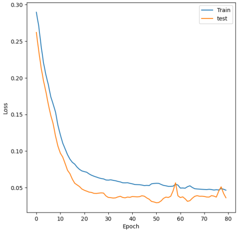




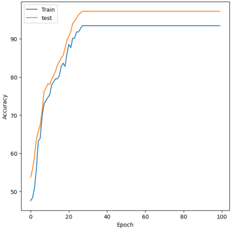
MONK results

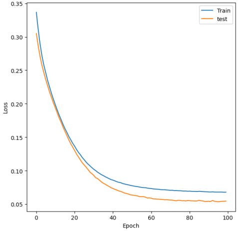
MONK 3 (no L2)





MONK 3 (L2)





MONK experiments

During the model selection phase we have analyzed the relations between the parameters that leads to overfitting and underfitting. In 1 a comparison between the final model of pytorch with L2 and a model that comes from the randomized search that underfits.

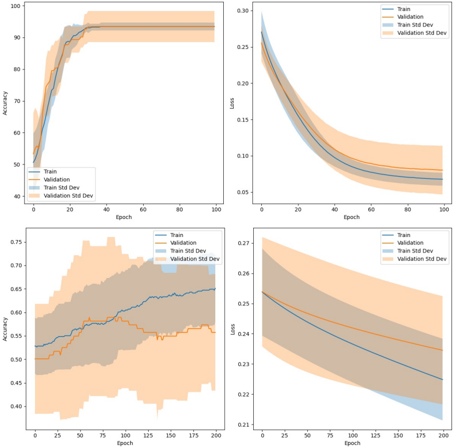


Figure: Loss and accuracy for training and validation with std

CUP validation schema

Repeated 7-fold-CV	
Training + Validation	80%

Hold-out	
Internal Test	20%

Table: Data splitting

- Explorative randomized search for the NNs
- Grid search
- Test of the best model on the internal test set
- Prediction on the blind test set

CUP experiments

The Approach that has lead to the best pytorch NN model is the following:

- starting a randomized search with a high range of parameters' values for a one hidden layer MLP since it is often enough for almost all tasks, and try to find the top 2 best configurations of parameters.
- use the top 2 best configurations to run a new randomized search with higher granularity and return the top 2 models.
- once the parameter for the number of units has converged and we have a small range of values for the other parameters we run a gridsearch.
- repeat this procedure of sequential randomized search + final gridsearch even for a multilayer model.
- compare the one hidden layer model MEE performance with the MEE performance of the multi layer model

CUP results

- PyTorch NN (best model) ranges
- 10 hours for the screening and the grid

Parameters	Hyperparameters tested
N layer	from 1 up to 5
Units for each layer	from 5 up to 250
Activation functions	ReLU, tanh
Learning rate	from 0.0001 up to 0.1
L2	from 0.0001 up to 0.1
Momentum	from 0.1 up to 0.9
Batch size	from 5 up to 100
Epochs	from 50 up to 300

Table: Screening phase

CUP results

At the end the best model to use is the one with the lowest MEE on the validation data. Then it has been tested on the internal test data.

Parameters	**Final Grid-search best model**
N layer	2 layers
Units first layer	180, 200
Units for second layer	20, 40
Activation functions	ReLU
Learning rate	0.001, 0.005, 0.0005
Momentum	0.9
L2	0.0001
Batch size	20, 40
Epochs	50, 100

Table: Values of parameters of best model

BEST PARAMETERS= 2 layers, 200 units first layer, 40 units second layer, momentum 0.9, learning rate 0.001, L2 0.0001, batch size 20, epochs 100

CUP results

MEE TR/VL	MEE TR/TS
0.813 std=0.041/1.0360 std=0.168	0.727/0.952

Table: Best model results

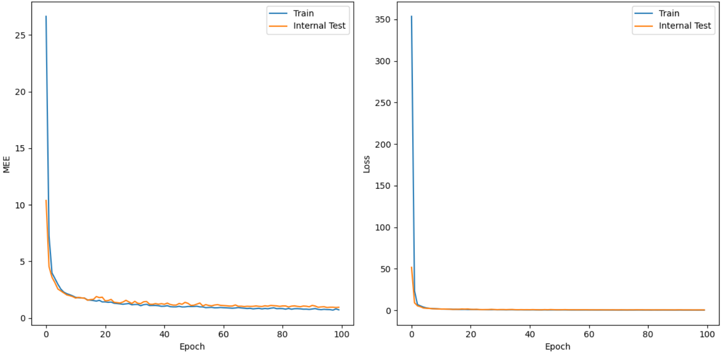


Figure: MSE and MEE for training and internal test

Conclusions

During the development of the project, we were able to put into practice the knowledge we acquired during the course, and we became confident in using several libraries. We became more practiced in model selection and assessment, more specifically, we saw how important a screening phase (randomized search in our case) is before the grid search, as it is too time-consuming if used with too many parameter combinations. In addition to a purely computational issue, the screening phase was also useful for us to understand what combinations of parameters did not get along and the direct effects on the loss and accuracy graphs.

Conclusions

Nickname: Golia
Bind test results: Golia_ML-CUP23-TS.csv

We agree to the disclosure and publication of our names, and of the results with preliminary and final ranking.

References

https://pandas.pydata.org/
https://numpy.org/
https://scikit-learn.org/stable/
https://matplotlib.org/
https://keras.io/
https://pytorch.org/
https://xgboost.readthedocs.io/en/stable/

Appendix

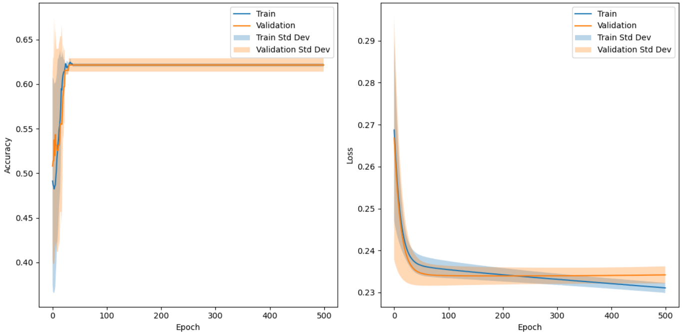


Figure: Low lr: MONK 2, batch = 2, lr = 0.001, momentum = 0.6

Appendix

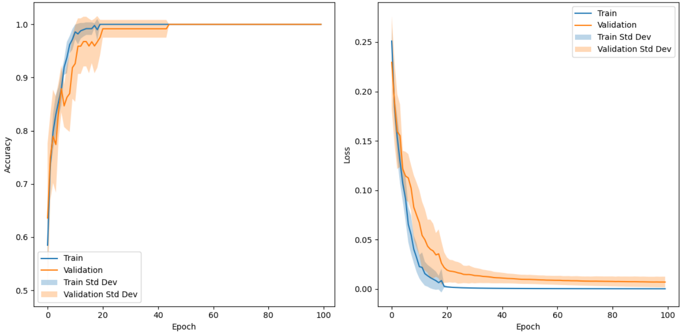


Figure: MONK 1: training and validation curves of the best model (Keras)

Appendix

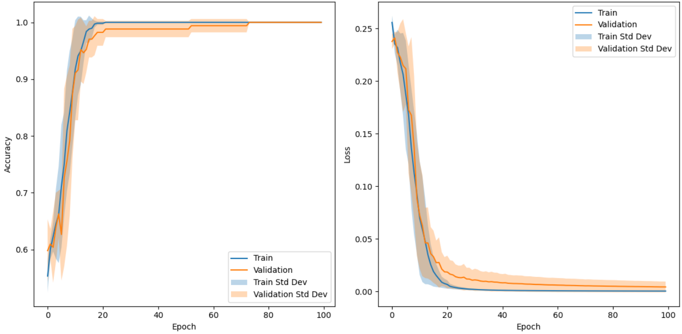


Figure: MONK 2: training and validation curves of the best model (Keras)

Appendix

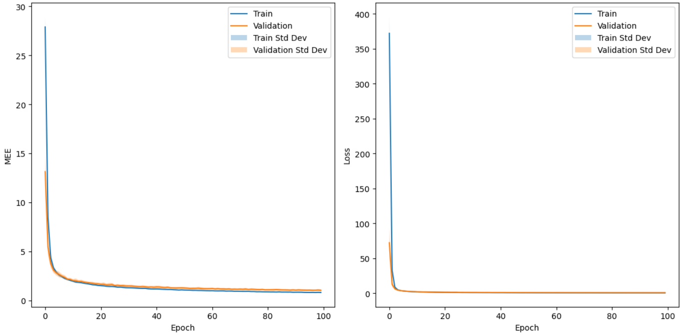


Figure: CUP: MSE and MEE with std on training and validation