ML report

Gabriele Barreca, Gioele Bertoncini
gabriele.barreca93@hotmail.com, gioele.bert@gmail.com

ML, accademic year: 2018/2019
15/07/2019
Project type B

Abstract

In this report we compare the performances of the regression of three models: MLP, SVM and Random Forest Regressor. We found that the MLP is very much slower than the other models and that the Extremely Randomized Trees (a more randomized variation of Random Forest) perform better than the other models while being also very fast.

1 Introduction

In this report we compare the performances of three model for regression tasks: feed forward multi-layer perceptron (MLP), Support Vector Machine (SVM) and Random Forest. We use the grid-search technique to explore the hyperparameter space. We also use Train-Validation split and k-Fold Cross Validation to evaluate the models. In the end we assess the models performance on a test set chosen randomly from the original data.

2 Method

We use scikit-learn for the preprocessing, the Support Vector Regressor (SVR), the Random Forest Reressor (RFR), the Extremely Randomized Trees (ETR) and the other model selection algorithms. For the MLP implementation we choose Keras. At first we split the data in Test set (10%) and Training set (90%). For the model selection we use k-fold Cross Validation with k = 3 (more details in sections 3.2, 3.3, 3.4). Before the training of the models we standardize the data with mean 0 and standard deviation 1. During the training of the model the loss is computed with the Mean Squared Error:

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(\mathbf{o}_i - \mathbf{t}_i)^2$$

where N is the number of input data, and $\mathbf{o}_i$ and $\mathbf{t}_i$ are respectively the output computed by the model and the target value for the pattern i. For the model selection and the final test results instead we use the Mean Euclidean Error, defined as:

$$MEE = \frac{1}{N}\sum_{i=1}^{n} ||\mathbf{o}_i - \mathbf{t}_i||_2$$

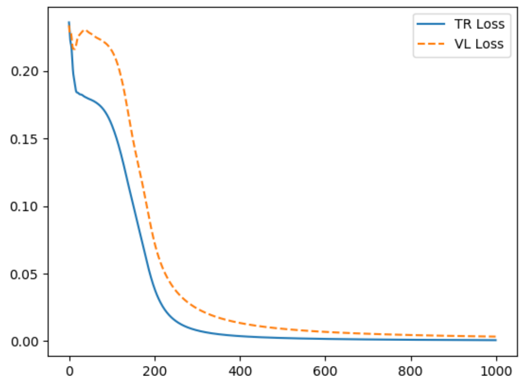
3 Experiments

All the plots in the sections 3.1 and 3.2 show the MSE values of train, validation or test over the number of epochs.

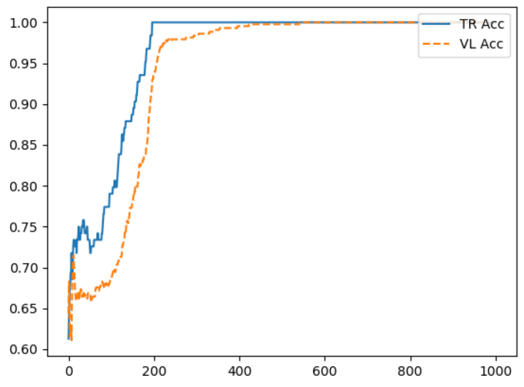
3.1 MONK's results

Task	Network	η	ν	λ	Activation function	Test Accuracy
MONK-1	1x7	0.6	0.9	0	Sigmoid	100%
MONK-2	1x7	0.6	0.9	0	Tanh	100%
MONK-3	1x7	0.6	0.6	0.003	Tanh	96%

Table 1: Prediction results obtained for the MONK's tasks with Keras MLP.

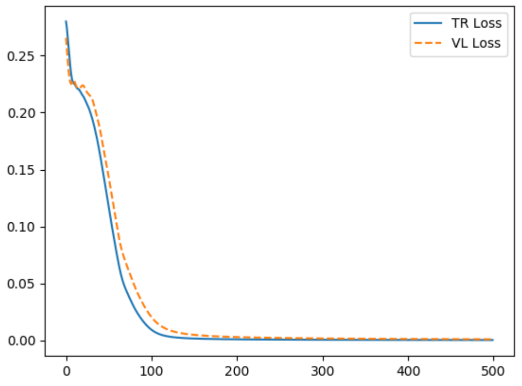


(a) Train and test loss.

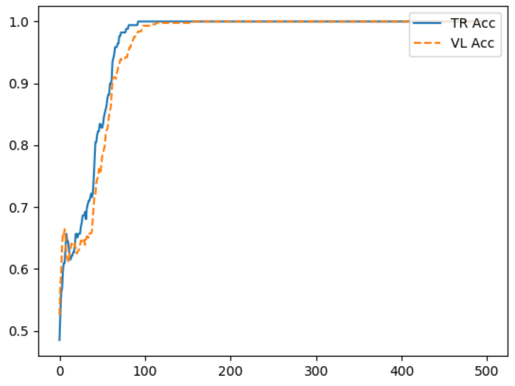


(b) Train and test accuracy.

Figure 1: MONK-1

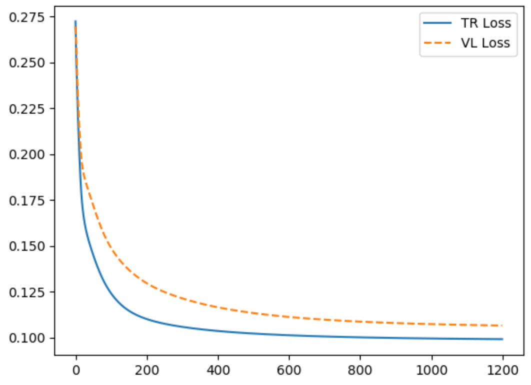


(a) Train and test loss.

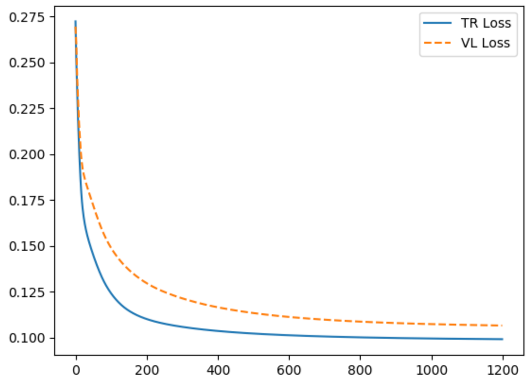


(b) Train and test accuracy.

Figure 2: MONK-2



(a) Train and test loss.



(b) Train and test accuracy.

Figure 3: MONK-3

3.2 Neural Network

We use a fully connected MLP implemented in Keras, optimized with Stochastic Gradient Descent. We use the Rectified Linear Unit as activation function. For the model selection we first perform a grid search with 3-fold cross validation over some network architectures (changing the hidden layers size, the number of hidden layers, the batch size and the learning rate). Then we split the training data in two sets (80%/20%) and we try to adjust the parameters looking at the learning curve. We run many experiments but in the following section are reported only the most relevant fore space reasons. We do this for two reasons: first the training of the MLP with cross validation takes too much time to be able to perform a wide enough grid search over all hyperparameters. Then because looking at the learning curve is very useful to evaluate the quality of the training (smoothness of the curve) and to consider an early stopping in case of overfitting. The values of the hyperparameters that we use for the grid search are the following:

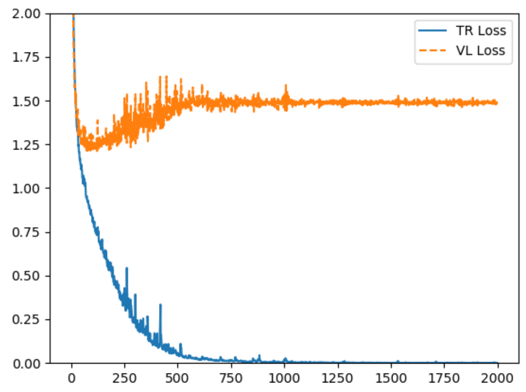
- $\eta = 0.001, 0.0001$
- batch size $= 64, 128, 915$
- n. of hidden layers $= 1, 2, 3, 5$
- hidden layers size $= 10, 50, 100, 500$

In Table 2 are showed the validation loss and the MEE of the best architectures founded by the grid search, we can see that the models with the highiest number of neurons are the ones with the lowest MEE, also in the table are reported only the different architectures, but the larger networks overperform the other models even with different values of learning rate and batch size.

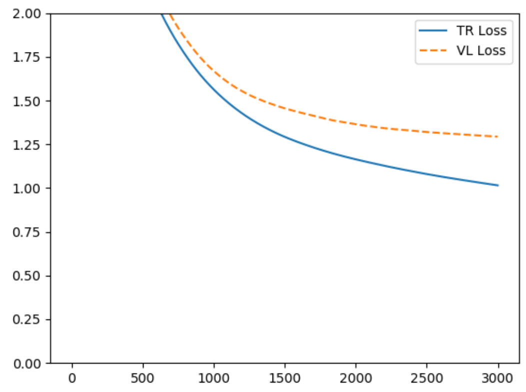
Hidden layers	**Units**	**Batch size**	η	**Validation loss**	**MEE**
5	500	915	0.001	1.453	1.228
3	500	128	0.0001	1.428	1.251
5	100	915	0.0001	1.398	1.269
3	100	128	0.0001	1.414	1.275
1	500	128	0.001	1.464	1.287
2	50	915	0.001	1.410	1.284
3	50	128	0.0001	1.438	1.292
1	100	128	0.001	1.465	1.310
3	10	64	0.0001	1.503	1.319
1	50	64	0.0001	1.466	1.329

Table 2: The result of the grid search after 5000 epochs, with $\nu = 0.9$, ordered by MEE (top 10 architectures).

From the grid search we found that the models trained with high size of mini-batch or with complete batch performs better. In Figure 4 are showed the learning curves of the best model trained with different batch sizes. The one trained with complete batch has a much better learning curve as it decrease constantly and smoothly. So we choose to use the batch algorithm for all the following experiments.



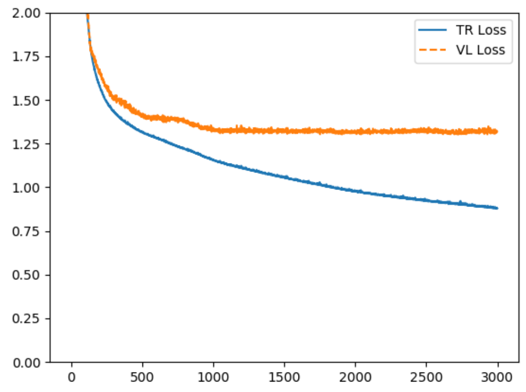
(a) $\eta = 0.0001$, batch size $= 128$.



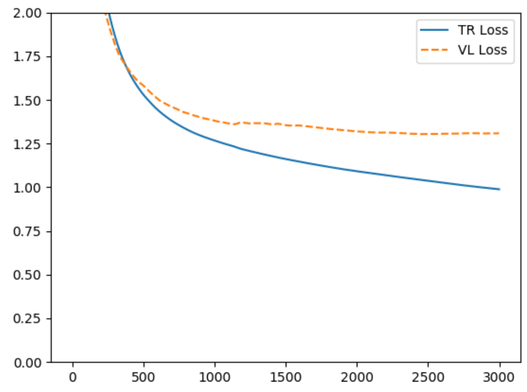
(b) $\eta = 0.0001$, complete batch.

Figure 4: Learning curves for the 5x500 architecture with different batch sizes.

Then we have a look at the behaviour of the different architectures, in particular the simpler models. In Figure 5 are showed the learning curves of two different model sizes. We can see that with smaller architectures (5a and 5b) the validation error remains stationary after some epochs. The bigger architecture (Figure 4b) instead has a better learning curve and is able to reach a lower error.



(a) 3x10, $\eta = 0.0001$



(b) 2x50, $\eta = 0.001$.

Figure 5: Learning curves for the MLP with two different architectures.

Finally we try to apply L2 regularization to the larger model. We find that only with small values of λ the model reaches the lowest validation error and also in this case the non-regularized model performs better, in Figure 6 is showed the learning curve of the 5x500 architecture with L2 regularization over 5000 epochs of training. In Table 3 are reported the parameters that we choose as final model.

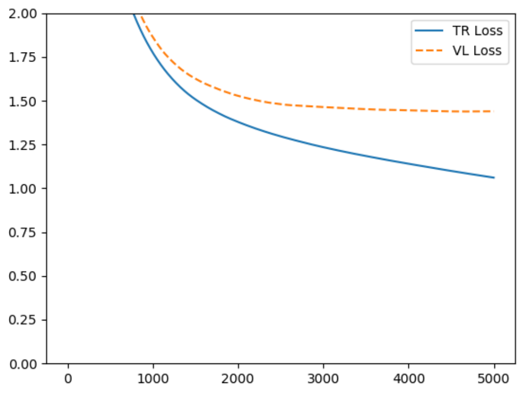


Figure 6: Learning curve with L2 regularization, $\lambda = 0.0001$.

Model	η	ν	**Stopping**	**Validation loss**	**MEE**
5x500, complete batch	0.0001	0.9	3000 epochs	1.520	1.249

Table 3: Final model for the MLP.

3.3 SVM

The SVM was implemented with scikit-learn through the use of a SVR with two different kernels: Radial Basis Function (RBF) and polynomial. All the grid searches are performed with 3-fold cross validation.

3.3.1 RBF kernel

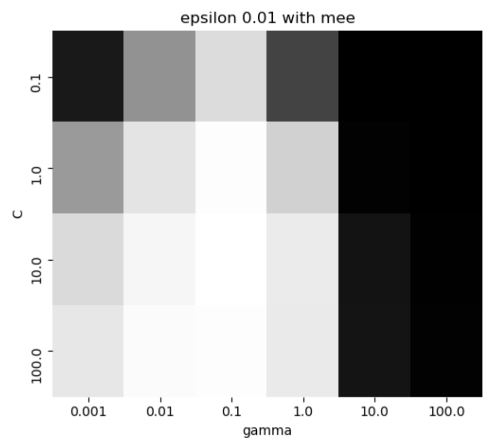
The RBF kernel is defined as:

$$k_{RBF}(x, x_i) = e^{-\gamma ||x - x_i||^2}$$

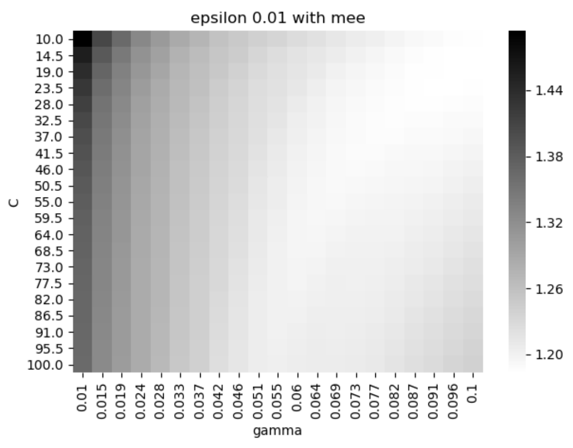
where $\gamma = \frac{1}{2\sigma^2}$.

To find the best combination of hyperparameters we perform a grid search over some orders of magnitude of C and γ and also over two different ϵ. The search extends from 10^{-1} to 10^2 for C, from 10^{-3} to 10^2 for γ and with $\epsilon = 0.001$ and $\epsilon = 0.01$. The results with the two different values of ϵ are the very same, so we choose to use $\epsilon = 0.01$ for all the following experiments. As shown in the heatmap in Figure 7a, the model performs better in the range $0.01 \leq \gamma \leq 0.1$ and $10 \leq C \leq 100$.

We use these intervals to perform a finer grid search over more specific values of this parameters. As can be seen in Figure 7b, the area with $0.82 \leq \gamma \leq 0.1$ and with $19 \leq C \leq 28$ is where the model reach the lowest MEE values.



(a) Orders of magnitude



(b) Close view

Figure 7: Heatmaps of the MEE for the grid searches over C and γ.

In Table 4 are reported the performances of the model in that area of the grid, the MEE is very similar for all the combinations but the model perform better when $\gamma = 0.1$ and $C = 23.5$ so we choose that as final model (Table 5).

C \ γ	**0.082**	**0.087**	**0.091**	**0.096**	**0.1**
19.0	1.2225	1.2200	1.2184	1.2163	1.2144
23.5	1.2206	1.2185	1.2164	1.2144	1.2140
28.0	1.2201	1.2180	1.2166	1.2165	1.2163

Table 4: MEE of the SVR with RBF kernel over the parameters γ and C.

Kernel	ϵ	C	γ	**Validation error**
RBF	0.01	23.5	0.1	1.214

Table 5: Final model for the SVM with RBF kernel.

3.3.2 Polynomial kernel

The second kernel that we tried is the polynomial which is defined as:

$$k_{POLY}(x, x_i) = (\gamma \cdot x^T x + r)^p$$

We first perform a grid search over C and the polynomial degree p. From the Figure 8 we can see that the model performs better with a polynomial degree of 5 and $C = 1$.

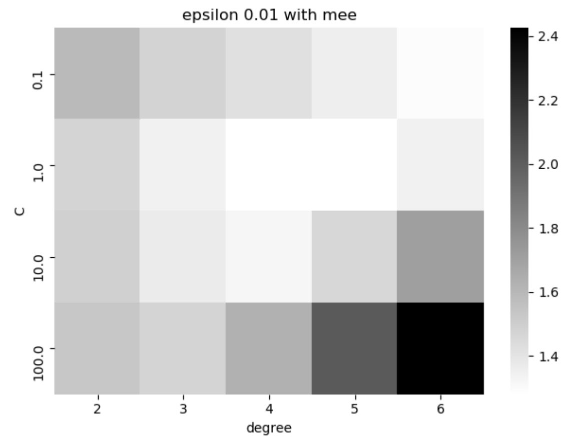


Figure 8: Heatmap of the MEE for the grid search over C and the degree p.

Then we want to investigate how the model behaves if we change the γ and the polynomial coefficient r, so we perform another grid search over this two parameters keeping fixed $C = 1$ and the polynomial degree at 5. The results are reported in Table 6. The lowest MEE error is achieved with $\gamma = 0.1$, while for different values of γ the model underfits. We choose as final model the combination of hyperparamters which is able to reach the lowest MEE validation error (Table 7).

		r			
		0	**0.5**	**1**	**2**
γ	**0.01**	8.550	2.724	1.580	1.447
	0.1	3.381	1.339	1.324	1.279
	1	10.859	4.724	4.302	3.953

Table 6: MEE for the polynomial kernel over the parameters γ and r.

Kernel	p	ϵ	C	γ	r	**Validation error**
Polynomial	5	0.01	1	0.1	1	1.279

Table 7: Final model for the SVM with polynomial kernel.

3.4 Random forest

The third model that we try is the Random forest with two different learning algorithms, first the ordinary Random Forest Regressor (RFR) [1] and then the Extremely Randomized Trees (ETR) [2], both implemented with scikit-learn. The models are very similar, they have only few differences:

- in ERT each tree is trained using the whole dataset, while in RFR to building trees are used bootstrap samples.
- in ERT the cut-point for the splitting in the tree, for each feature, is randomly selected.

For these reasons we use the same parameter's values to perform a grid search over: the maximum depth of the trees (*Max depth*), the number of trees in the forest (*N. estimators*), the minimum number of samples required to split an internal node (*Min sample split*) and the maximum number of feature to consider in order to find the best split (*Max features*).

Some 'good' values of these parameters can be found in the original papers ([1], [2]). For the RFR the paper suggest *N. estimators* = 100, *Min. sample split* = 5 and *Max. features* = $\lfloor \frac{n}{3} \rfloor$. For the ETR default parameters are *N. estimators* = 100, *Min. sample split* = 5 and *Max. features* = n. However the best values can depend also on the specific problem, so we explore some more values:

- Max. depth = $[1, 10]$
- N. estimators = $10, 50, 100, 500, 1000$
- Min. sample split = $[1, 5]$
- Max. features = n, $\lfloor \frac{n}{3} \rfloor$

where n is the number of features of the input data.

In Table 8 and in Table 9 are showed the the top 5 combination of the parameters for both the Random forest models.

N.estimators	**Max.depth**	**Min.sampleSplit**	**Max.feature**	**Validation loss**	**MEE**
100	9	4	10	1.2940	1.1108
500	9	3	10	1.2985	1.1164
500	10	2	10	1.2987	1.1174
1000	10	3	3	1.1869	1.1190
1000	9	2	10	1.3028	1.1204

Table 8: The results of the grid search for the RFR model.

N.estimators	**Max.depth**	**Min.sampleSplit**	**Max.feature**	**Validation loss**	**MEE**
500	10	2	10	1.0988	1.0616
1000	10	4	10	1.1053	1.0640
1000	10	2	10	1.1082	1.0646
1000	10	3	10	1.1086	1.0648
500	10	4	10	1.1081	1.0662

Table 9: The results of the grid search for the ETR model.

As a final model we choose the ETR algorithm that was able to reach the lowest MSE error.

N.estimators	**Max.depth**	**Min.sample split**	**Max.feature**	**Validation loss**	**MEE**
500	10	2	10	1.0988	1.0616

Table 10: Final model for ETR.

3.5 Final model selection

We select as final model to use for the CUP, the one with the lowest MEE error, using 3-fold Cross-Validation on our training set. In Table 11 are reported the performances of the three models (see Table 10, 5 and 3 for the parameters). We choose ETR for the CUP, see the *BarBer˙ML-CUP18-TS.csv* file for the results. These are generated from the script Test.py. For more information on how to perform this and other script, please look at the ReadMe.md file attached to the project.

Model	MEE
ETR	1.061
SVM	1.214
MLP	1.249

Table 11: Model selection results.

4 Test results and conclusions

From the Table 12 we can see that the ETR and the SVM are the fastest models and also they have the best performances. The MLP is very much slower and does not perform so well. Finally we want to remark some aspects that we encountered during the experiments.The SVM and Random Forest models are very much easier to study due to the fact that we can run very large grid searches and do much more experiments. On the other side training large Neural Networks takes much more time, and do an exhaustive search over a huge hypotesis space can be hard.

Model	Test error	Computing time
ETR	0.870	0.810s
SVM	1.000	0.176s
MLP	1.208	111.277s

Table 12: Model assessment results. Computing times refers to a pc with an i5 6600K CPU @ 3.5GHz

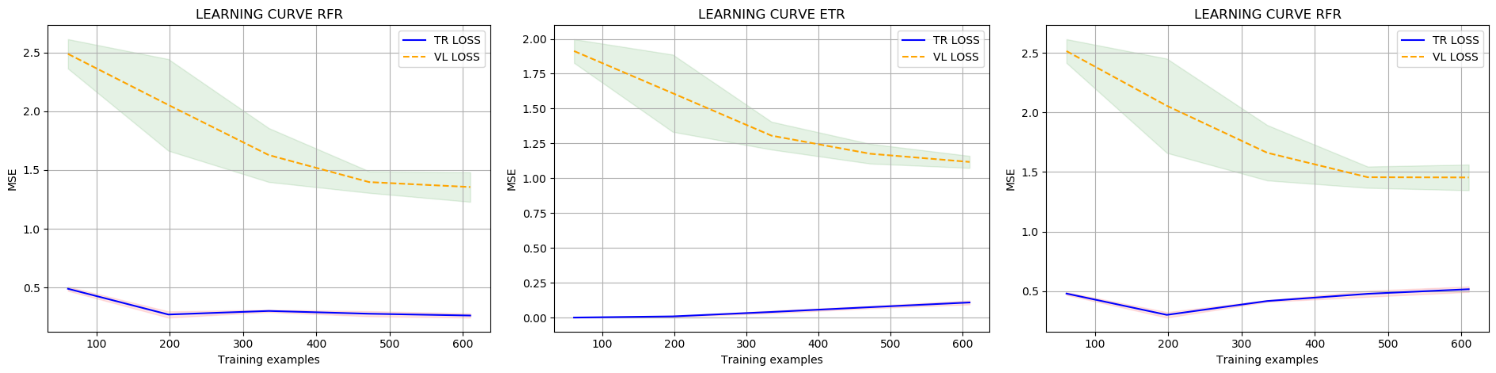
References

[1] Tin Kam Ho: 'Random decision forests', Proc. Int. Conference on Document Analysis and Recognition, *IEEE*, **1995**

[2] P. Geurts, D. Ernst, L. Wehenkel: 'Extremely randomized trees', Machine learning, *Kluwer Academic Publishers*, Vol. 63, pp 3-42, https://doi.org/10.1007/s10994-006-6226-1, **2006**

Appendices

A Random forest plots



(a) RTR (b) ETR (c) RTR with Max leaf node = 40

Figure 9: Learning curves of the random forest models (MSE).