Design of Experiment-based Configuration of Hyperparameters Of An Artificial Neural Network

Rosa Arboretti[1], Riccardo Ceccato[2], Luca Pegoraro[2]
Luigi Salmaso[2]

[1]Department of Civil, Environmental and Architectural Engineering, University of Padova, Padua, Italy
[2]Department of Management and Engineering, University of Padova, Vicenza, Italy

Abstract
This work presents an application of Design of Experiments (DOE) to the choice of best configuration of hyperparameters in a neural network. The example provided uses real data and shows how the use of DOE principles can increase accuracy and reduce the effort required when tuning complex machine learning algorithms. This strategy is particularly useful for practitioners who do not have any particular expertise in this field and can help them understand the relationships that exist between the different hyperparameters and some relevant metrics such as computational time and validation error.

Key Words: Neural network, hyperparameters, DOE.

1. Introduction

In recent years machine learning methods have found application across many fields from science and technology to engineering. The main advantages of such methodologies include their non-parametric nature, meaning that little or no assumption need be made about data, their flexibility, and the possibility to model extremely complex data. However, such advantages come at a cost, namely the effort required to train the algorithms. Such effort is twofold: firstly there is the computational cost required to estimate a rather large number of parameters, such as the weights in a neural network; secondly there are the difficulties faced by the practitioner in choosing the proper hyperparameter setting that governs the algorithm. Additionally, the more the selected machine learning algorithm is complex, the more the effort is onerous. Among the most intricate algorithms are deep neural networks. The term “deep” indicates the number of layers added to the network: as the number of layers increases, the number of parameters to be estimated increases at an even faster rate, and the choice of an appropriate hyperparameter configuration becomes more difficult.
To overcome this burden, several approaches are adopted by practitioners. Two of the most common are manual optimization and grid search.
Manual optimization of hyperparameters is basically a trial-and-error approach which relies on the expertise of the analyst. Model tuning is often regarded more as an art than a science, therefore experience plays a major role in this context. However, not all practitioners are experts, and even the more skilled analysts risk omitting some of the configurations that can lead to best results. Furthermore, the number of hyperparameters can soon become overwhelming, and manual optimization can be very time consuming.

Grid search is an automated strategy for hyperparameter tuning. It consists in choosing a relevant interval and step size for each hyperparameter and trying all possible configurations in search of the one leading to the best results. In this case the computational effort is considerable, and it can become excessively demanding as the number of hyperparameters increases.
Another strategy can speed up the training of an algorithm while investigating a satisfactory number of all possible combinations of hyperparameters and neither increases the effort required by the analyst nor requires excessive expertise. Such an approach consists in using a Design of Experiments (DOE) plan in which the factors are the hyperparameters of the algorithm and the responses are some valuable metrics, such as accuracy in the case of classification, and mean squared error in the case of regression tasks. Another variable of potential interest is the computational time required for training. It is worth noting that this approach has been used by other authors (Packianather, Drake, & Rowlands, 2000; Staelin, 2003; Lasheras, Vilán, Nieto, & del Coz Díaz, 2010; Lujan-Moreno, Howard, Rojas, & Montgomery, 2018), and in this paper the aim is to present an application of such a method in order to show the valuable insights that can be obtained by this approach.

2. Methodology

The procedure of hyperparameter tuning in a machine learning model using DOE is rather simple and consists of the following steps (Lujan-Moreno, Howard, Rojas, & Montgomery, 2018):

1. Choice of the algorithm and response variable
2. Selection of hyperparameters to tune and relative ranges
3. Execution of a screening design and identification of the relevant hyperparameters
4. Reduction of the model and execution of a full or fractional factorial design
5. Fitting of a second-order model using response surface methodology (RSM)
6. Optimization of the RSM model to find the configuration of hyperparameters that leads to the optimization of the response

It is worth noting that if there is reasonable confidence in the list of relevant hyperparameters, and if the dimension of the problem allows it, point 3 can be skipped and a 3-level factorial design can be run directly.
In experimental designs the initial phase of planning the DOE study is crucial, meaning that particular attention should be given to the choice of hyperparameters that are to be tuned and the range of their levels, which should be large enough to allow identification of a relevant effect but not excessively large. Replication should also be performed to consider the stochastic nature of many machine learning algorithms, and randomization should be employed to ensure independence of the experimental runs. Furthermore, the adoption of DOE and RSM allows not only to assess the impact of the main effects on the response variables, but also to investigate interactions among the factors, thus generating useful insights that can help the analyst understand the rationale which drives the performance of the algorithm.
In the case of a deep neural network, which is the algorithm employed in this paper, a list of hyperparameters that should be considered for optimization includes: the number of hidden layers, the number of neurons in each layer, the activation function, the number of iterations used for training, the learning rate and the batch size of data used at each update of the weights. All these hyperparameters are expected to have an impact on the accuracy of the algorithm or on the time required for training, which are the two responses studied in this work.

3. Case study

The case study uses data taken from an industrial experiment that dealt with the detection of a fault which typically occurs to refrigeration equipment, namely "condenser fouling" (Salmaso, Pegoraro, Arboretti, Ceccato, Bianchi, Restello & Scarabottolo, 2019). The goal of the experiment was to discover how this fault impacts on the parameters of the refrigeration cycle in order to detect the severity of the failure by reading the measurements of sensors installed on the apparatus. The aim was to discern the case in which the fault affected 25% of the system against when it affected 50%, and a machine learning algorithm using 8 predictors was used to achieve this goal.
The algorithm of choice is a neural network developed using the *mxnet* package in R (Chen, Kou, He & Acharya, 2017), and the relevant response variables, presented in this work, are the computational time required to train the models and the accuracy calculated on a hold-out dataset which includes 25% of the original data. The hyperparameters selected as factors of the DOE study and their ranges are presented in Table 1.

Table 1: Hyperparameters of the neural network selected as factors of the DOE and relative ranges

Factors	*Description*	*Ranges*
n_layers	Is the number of hidden layers in the neural network	1-3
n_neurons	Is the number of neurons in each hidden layer	5-15
activation	Is the activation function of each neuron in the hidden layers	relu, sigmoid, tanh
n_epochs	Is the number of iterations used to train the model	50-150
learning_rate	Is the learning rate used to train the model	0.01-0.1
batch_size	Is the number of training instances used in one iteration	64-256

A replicated 3-level factorial design optimized with the criterion of the D-optimality has been selected. It includes 79 experimental runs that were executed on a personal computer. The response surface methodology has been applied and only the most relevant factors have been included in the final models since backward stepwise regression ($\alpha = 0.05$) is used. For both the responses "Test accuracy" and "Time" the p-value of the ANOVA for the overall models is $p < 0.0001$. The most relevant results are shown in Tables 2 and 3.

Table 2: Results for the response "Test accuracy". The symbol "*" indicates $p < 0.05$.

Term	*Estimate*	*Std Error*	*t Ratio*	*Prob>|t|*
Intercept	0.7486218	0.03271	22.89	<.0001*
n_layers(1,3)	-0.031739	0.007617	-4.17	<.0001*
n_neurons(5,15)	0.0072916	0.007713	0.95	0.3480
activation{sigmoid-relu&tanh}	-0.10478	0.007711	-13.59	<.0001*
n_epochs(50,150)	0.0197046	0.007666	2.57	0.0125*
learning_rate(0.01,0.1)	0.062479	0.007304	8.55	<.0001*
batch_size(64,256)	-0.024859	0.007441	-3.34	0.0014*
n_layers*n_layers	0.0500127	0.021031	2.38	0.0204*

Term	*Estimate*	*Std Error*	*t Ratio*	*Prob>|t|*
n_layers*n_neurons	-0.019243	0.008178	-2.35	0.0217*
n_layers*(activation{sigmoid-relu&tanh}+0.38462)	-0.042763	0.008165	-5.24	<.0001*
n_epochs*n_epochs	0.0716517	0.022361	3.20	0.0021*
(activation{sigmoid-relu&tanh}+0.38462)*learning_rate	0.0378516	0.007759	4.88	<.0001*
learning_rate*learning_rate	-0.057713	0.024222	-2.38	0.0202*
batch_size*batch_size	0.0470985	0.023142	2.04	0.0460*

Table 3: Results for the response "Time". The symbol "*" indicates $p < 0.05$.

Term	*Estimate*	*Std Error*	*t Ratio*	*Prob>|t|*
Intercept	7.7057013	0.267865	28.77	<.0001*
n_layers(1,3)	1.2463965	0.287636	4.33	<.0001*
n_epochs(50,150)	3.9439517	0.28607	13.79	<.0001*
batch_size(64,256)	-4.194953	0.288679	-14.53	<.0001*
n_layers*batch_size	-0.805178	0.307284	-2.62	0.0107*
n_epochs*batch_size	-2.027308	0.300882	-6.74	<.0001*

The results displayed in Table 2 show that all the main effects are relevant with respect to the performance of the model in classifying the fault on the hold-out test set. It is worth noting that a significant difference exists when the activation function is a sigmoid function rather than a rectified linear or hyperbolic tangent function, however no significant difference is found when the activation function is changed from the rectified linear function to the hyperbolic tangent. Better accuracy is obtained when the activation function is rectified linear or hyperbolic tangent. Another rather surprising result is that accuracy seems to decrease when the number of layers in the network increases, meaning that the underlying function describing data is rather simple and does not need a great number of parameters to be captured. Given that the results reported are the errors on a validation set, a decrease in accuracy could also indicate that some overfitting is occurring when the number of layers is excessively large. A large effect is also detected for the learning rate, confirming it to be one of the hyperparameters which cannot be overlooked when training a neural network (Bengio, 2012). Several interactions and quadratic effects are significant as well. Figure 1 describes the relationships that govern the interaction effects. Some interesting results include: adding neurons to the layers has a positive impact on accuracy only when the number of hidden layers is one; it has a negative impact on the accuracy as the number of layers increases; increasing the number of layers degrades the accuracy far more when the sigmoid function is chosen as an activation function.

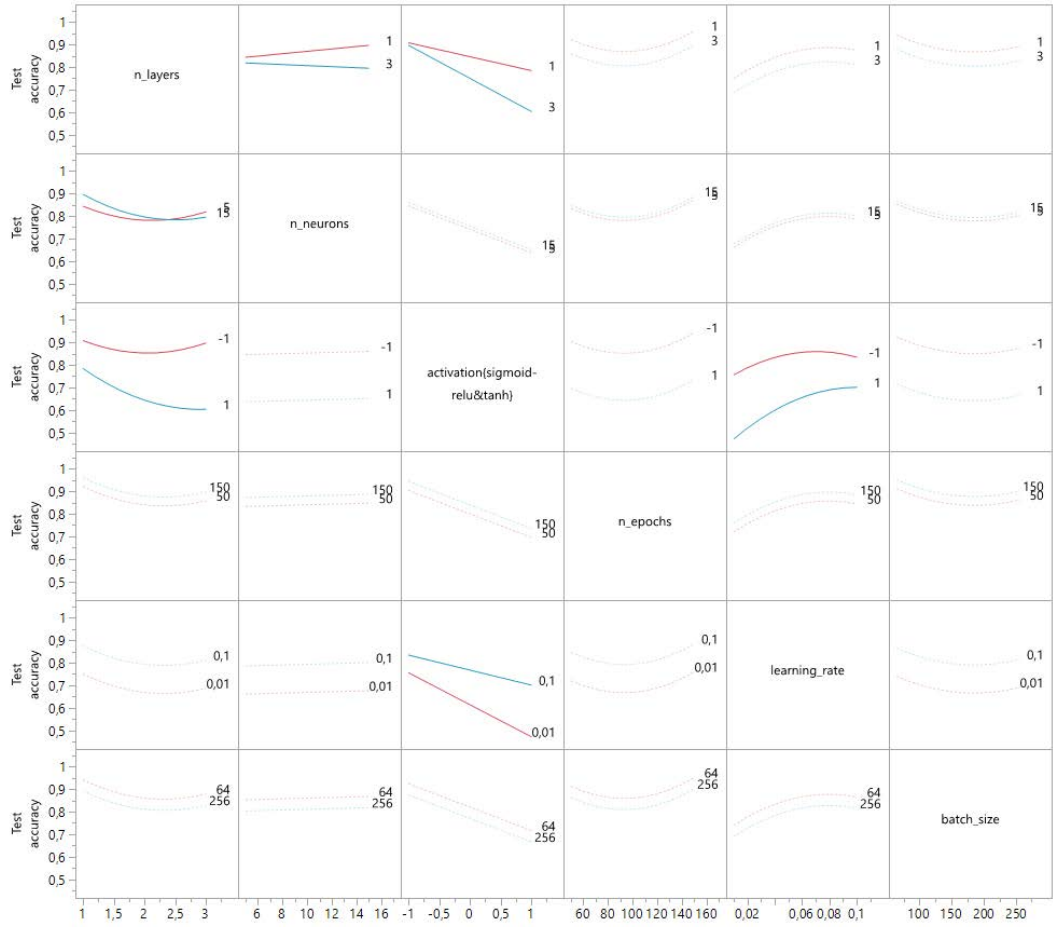

Figure 1: Interaction plots for the response "Test accuracy". Dashed lines indicate interactions which are not statistically significant.

The situation is rather different for the response "Time". Only the number of hidden layers, number of iterations and the batch size seem to influence the time required to train the algorithms. The results are in the main as expected: the number of iterations has the largest positive impact on the computational effort, and a negative effect is detected between the batch size and the time required for training (Bengio, 2012). A network with a larger number of layers requires more time to train, and the interactions, including the batch size, are also significant (Figure 2). It is interesting to note that for large batch sizes the computational time only slightly increases when the number of layers or number of iterations increases. Although in this case the dimension of the problem is rather limited, batch size appears to greatly impact reduction of the computational time required to train a neural network.

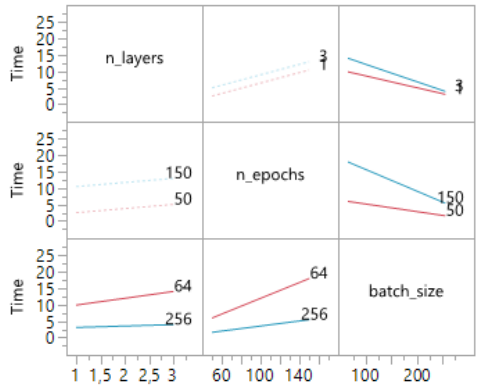

Figure 2: Interaction plots for the response "Time". Dashed lines indicate interactions which are not significant.

The next step is to identify the configuration of hyperparameters that leads to optimization of the responses. In this case the aim is to maximize test accuracy while minimizing the time required to fit the neural network. An optimizer is used, and the results are shown in Figure 3.

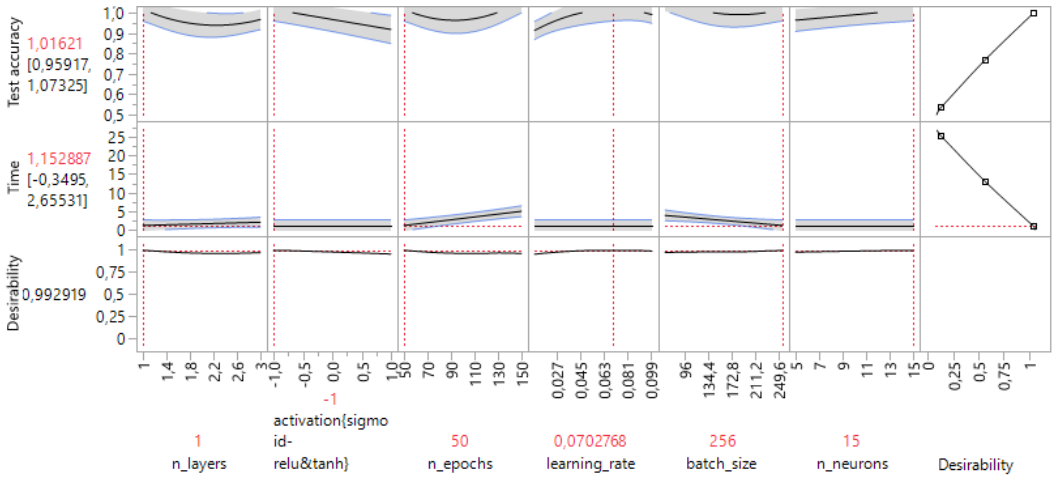

Figure 3: Optimal configuration of hyperparameters that maximizes "Test accuracy" while minimizing "Time".

The optimal configuration of hyperparameters is achieved for a neural network with 1 hidden layer and 15 neurons, either $relu$ or $tanh$ activation functions, and the training procedure should be executed for 50 iterations with a learning rate of 0.0703 and a batch size of 256 training instances. Note that at this point an additional DOE study could be carried out to investigate the region where n_epochs < 50 , batch_size > 256 and n_neurons> 15 since the optimal configuration is achieved for these extreme values of the factor ranges. However, since the accuracy has already achieved optimality and the predicted training time is around 1 second, the choice was made to stop and accept this configuration as the optimal one.
The optimal configuration has not been previously tested therefore 10 neural networks have been trained for validation. The results are shown in Table 4: both "Test accuracy" and "Time" values are inside the confidence intervals predicted by the DOE model for all

the new instances, although the time required for training is on average lower than the predicted value.

Table 3: Results of "Test accuracy" and "Time" for 10 neural networks trained with the optimal configuration of hyperparameters.

Instance ID	*Test accuracy*	*Time [s]*
1	1	0.85367
2	1	0.893476
3	0.996558	0.894027
4	0.996558	0.954077
5	0.998279	0.864067
6	0.977625	0.921698
7	0.996558	0.947628
8	0.996558	0.845608
9	0.982788	0.883035
10	0.998279	0.908921

Other useful insights can be derived from the investigation of the contour plots, which show how the response varies when two predictors are considered. As an example, in Figures 3 and 4 we show how the responses "Test accuracy" and "Time" vary when the number of layers together with other predictors are varying. In order to investigate a neighbourhood of optimal conditions, the variables which are not included in the plots are fixed according to the results of Figure 3.
The contour plots highlight some results that were previously only partially evident. The best results for test accuracy are achieved either when the number of layers is small and the number of neurons is large or when the number of layers is large and the number of neurons is small, meaning that two different strategies are available to the analyst. A saddle point is present in the surface plot that associates the learning rate and the number of layers, implying that the best results are reached when the learning rate is around the optimum and the number of layers is either 1 or 3. When batch size or number of training iterations are associated with the number of layers, "Test accuracy" is maximized on the extreme areas of the experimental region. Figure 5 shows that the computational effort is minimized when both the number of layers and number of epochs is minimized or when the number of layers is small, and the batch size is large.

3. Conclusions

This article shows the advantages of the application of DOE for optimization of hyperparameters of a neural network. Although the presented results are specific to the problem at hand and cannot be generalized, the adopted framework proved to be valuable for practitioners who desire to understand the rationale driving the performance of the algorithms. Among the main advantages of such an approach are: (i) a significant combination of hyperparameter levels is studied, reducing the risk of omitting some relevant configurations, (ii) not only the main effects are studied, but also the interactions between the hyperparameters, (iii) the optimal setting is sought inside the whole experimental region, not only for the combinations actually tested, (iv) the time required to tune the algorithms can effectively be reduced.

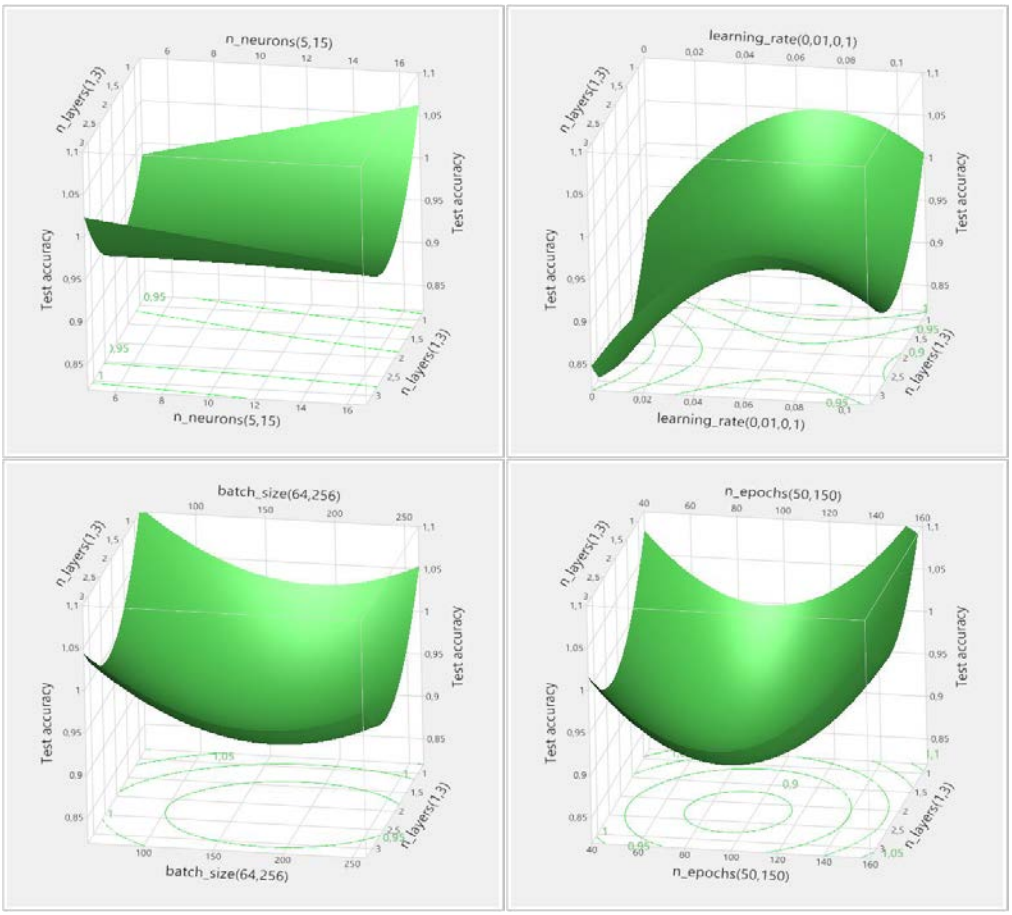

Figure 4: Surface and contour plots of the response "Test accuracy" in a neighbourhood of the optimal setting of predictors.

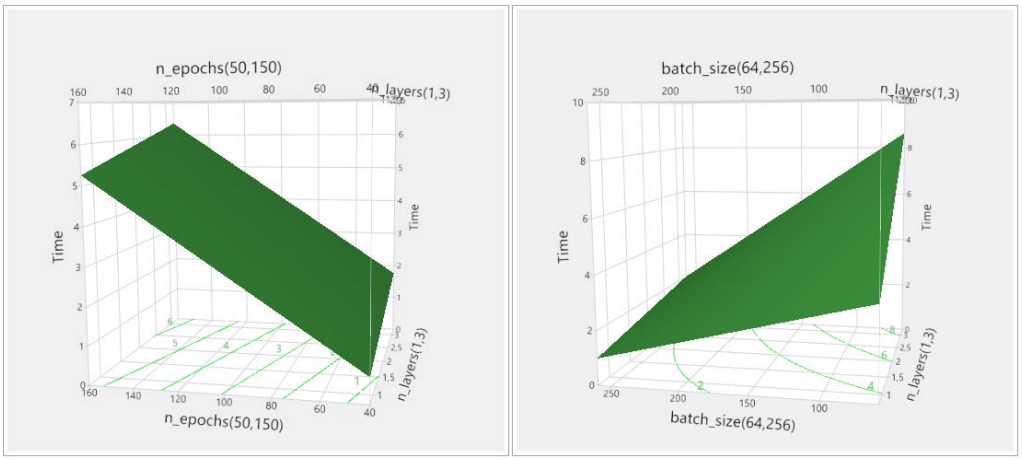

Figure 5: Surface and contour plots of the response "Time" in a neighbourhood of the optimal setting of predictors.

References

Bengio, Y. (2012) Practical Recommendations for Gradient-Based Training of Deep Architectures. In: Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_26

Chen, T., Kou, Q., He, T., & Acharya, A. (2017). mxnet: MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. R package version 1.5.0. https://github.com/apache/incubator-mxnet/tree/master/R-package

Lasheras, F. S., Vilán, J. V., Nieto, P. G., & del Coz Díaz, J. (2010). The use of design of experiments to improve a neural network model in order to predict the thickness of the chromium layer in a hard chromium plating process. Mathematical and Computer Modelling, 52 (7-8), 1169-1176.

Lujan-Moreno, G. A., Howard, P. R., Rojas, O. G., & Montgomery, D. C. (2018). Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study. Expert Systems with Applications, 109, 195-205.

Packianather, M. S., Drake, P.R. & Rowlands, H. (2000). Optimizing the parameters of multilayered feedforward neural networks through Taguchi design of experiments. Quality and Reliability Engineering International, 16, 461-473.

Salmaso, L., Pegoraro, L., Arboretti, R., Ceccato, R., Bianchi, A., Restello, S., & Scarabottolo, D. (2019). Design of experiments and machine learning to improve robustness of predictive maintenance with application to a real case study, Communications in Statistics - Simulation and Computation, DOI: 10.1080/03610918.2019.1656740

Staelin, C. (2003). Parameter selection for support vector machines. Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1, 1.