EE-411: Reproducibility challenge Reconciling modern machine learning practice and the bias-variance trade-off

Defne Culha, Hector M. Ramírez C., Riselda Kodra and Valgerdur Jónsdóttir
EPFL, Switzerland

***Abstract*—A couple of years ago, neural networks in machine learning became a game changer, and now they are often the subject of study and interest. A neural network can be seen as an over-parameterized model, learning hundreds, even thousands, or millions of parameters. However, the theory says that the more complex a model becomes, the less generalization we can achieve, as the model can become over-fitted to the data. This comparison between theory and practice motivated us to reproduce the results from a specific paper. The following discussion and its results establish and confirm the double descent curve many high-parameter modern machine learning applications demonstrate. Three kinds of experiments were conducted to observe this behavior of high-parameter models in the context of both neural networks and ensemble methods. The methods used to reproduce the double descent curve for these models are discussed, and the results are analyzed.**

I. Introduction

In this project, we attempted to reproduce some of the results shown in the following paper: *Reconciling modern machine learning practice and the bias-variance trade-off* [1]. The paper introduces the concept of double descent, a phenomenon that seems to contradict the bias-variance trade-off in classical statistics, namely that having too many parameters, i.e., having a high model complexity, will yield a substantial error. Modern machine learning techniques, such as neural networks, show that increasing the model complexity beyond a certain point of interpolation does, in fact, result in improved performance. The paper provides evidence for the existence of double descent for several models and datasets, and in our reproduction of its results, we will focus on neural network and ensemble methods such as Random Forests.

II. Models and methods

We attempted to reproduce the paper's results for three types of models. In the context of neural networks, we emphasized reproducing the double descent curve for models such as Random features and a fully connected two-layer neural network. In the context of ensemble learning methods, we aimed to reproduce the double descent curve for a Random Forest model. All of these models were trained on the MNIST dataset [2], either on the full dataset or on a subset of it, with feature dimension $d = 784$ and number of classes $K = 10$. We decided to focus on the MNIST dataset, which would provide us with enough complexity to see some promising results while also avoiding needing more memory or computing power. In some experiments, we used Google Colab and a GPU Nvidia GEForce RTX 3060.

Each of the tested models is trained to minimize the squared loss on the training set, and for comparison, we report the zero-one loss or the cross-entropy loss as well.

A. Random Features

The Random Fourier Features model is one of the model introduced in [1] to depict the double descent curve. The intention is to use an approximation of the Reproducing Kernel Hilbert Space (RKHS) corresponding to the Gaussian kernel [3]. In our case, to spare some computational cost and simplify it, we used the Random Features approximation, which is also an approximation of the RKHS, with the difference that in this case, we only have a real part, which was one of the significant advantages in our case. Nevertheless, we can see that the ℓ_2 norm of the parameters is less stable. In this case, the Kernel approximates a kernel and a two-layer neural network, where the first layer is the Random Features Kernel and the second is the classification or regression layer.

We decided to use Scikit-learn to handle the classification and regression problem and make the experiments more agile. In this case, we are not using a regularization because the regularization would have to be used for a particular model in a cross-validation stage, which would increase the complexity of the model, moreover, is different from the aim of the paper.

For the experiment, we use $d \in (1000, 20000)$ parameters, which we used as the upper bound due to computational complexity.

By using Random Features only, we aimed to have a similar result, being aware of the differences between the two Kernel Approximations. In the case of the Random Features, we avoid computing complex numbers, with the difference that the norm of the feature map in our case got bigger.

B. Fully connected two-layer neural network

Besides Random Features, the general multilayer network was also observed in the paper. The working principle of this method consists of two parts: forward and backward propagation. In the forward propagation, versions of SGD can be used and we used it with momentum 0.95, and in the back propagation, partial derivatives are calculated. In order to observe the double descent, a simple two-layer network is implemented, and the number of parameters, which is given with the formula $(d+1) \cdot H + (H+1) \cdot K$, is increased by increasing the number of hidden units, H. In this case, obtaining the double descent curve was challenging since SGD optimization problems are highly dependent on their initialization. According to [1], to observe the double descent, the number of parameters should be bigger than the number of samples multiplied by the number of outputs. Also, to improve the training, the paper suggested first training a smaller network up until the interpolation threshold and then using that network's weights in the bigger network, but this implementation was not executed. In the paper, the number of epochs was much bigger than the one we used due to the limits of computing power. The number of samples used from MNIST was $n = 2 \cdot 10^4$ and testing sampels were $5 \cdot 10^3$.

C. Random Forest

As shown in [1], the double descent curve is also established with other prediction methods besides neural networks, such as the Random Forest. The Random Forest classifier is an ensemble learning method that operates by fitting a multitude of decision tree classifiers on samples of the dataset. By doing so, it corrects for the overfitting habit of decision trees. To observe the model's double decent curve, we will increase the parameter of the number of decision trees in the forest when observing the curve beyond the interpolation threshold. To reach the interpolation threshold, we increase the size of the decision tree, i.e., the number of leaves of a tree. Increasing these two parameters implies that we increase the complexity of the model. The number of samples from the MNIST dataset was $n = 10^4$, where 40% of that we define as testing samples. For building the trees, the whole dataset is used instead of bootstrap samples from it.

III. Results

Following our discussion about the models used in our three experiments, we consider the obtained results. It is important to remark that in some cases, the models are not precisely the same, as we adapted them to our available resources. Nevertheless, the results achieve a considerable resemblance to the ones shown in the original paper.

A. Random Features

In the case of the Random Features, we can observe the double descent as in the Fourier Random Features conducted in the original paper, with the difference that the minimum norm solution is not attained. We can observe that the more parameters (over-parametrization) are in the model, the better generalization can be achieved after re-basing a threshold in both cases. The model starts to over-fit the data as the training decreases, but the test error increases for the ϵ values in both cases, which can be observed in Fig. 1. We consider this experiment successful: The generalization error decreases as the parameters increase.

Previously, we stated that the Random Features, as the Fourier Random Features, are an approximation of the kernel that expands the feature space, increasing the number of parameters for the minimum square error and the zero-one loss (cross-entropy) losses for the classification.

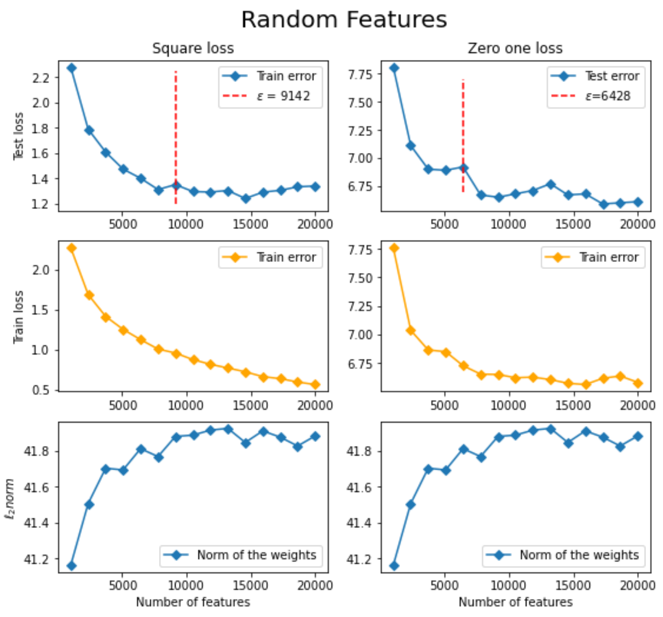

Fig. 1. The double descent curve of a Random Features model.

B. Fully connected two-layer neural network

With the fully connected neural network implementation, we can observe in Fig. 2, that as we increase the number of parameters and the model switches from under-parameterized to over-parameterized, the mean square error and cross entropy loss decrease both for the test and training sets. Moreover, since we use SGD with momentum as the optimizer, it is noted that initialization is important to achieve a solution to the problem. It is concluded that, in neural networks, the number of parameters used should be at least the number of data points times the number of outputs in order to see interpolation. This may lead to using neural networks with very high number of parameters for large data sets. In our experiment, we could not observe this phenomenon very clearly due to the limitations of our computing efficiency. However, the observations would be more apparent with more samples and training with more epochs. The curve that resembles more to the double descent one is obtained for the squared loss. The figure we were trying to replicate from [1] was Figure 9(c), since we did not reuse any weights. Even though we could not obtain a clear view of the double descent, we were able to see that despite over-fitting the data, we get highly accurate models.

C. Random Forest

The Random Forest model shows clear evidence of a descent, which can be observed in Fig. 3. As the model's complexity increases with the increase of number of trees and maximum number of leaves allowed for each each tree, both the squared test loss and the zero-one test loss drop substantially. However, the test loss drop and increase below

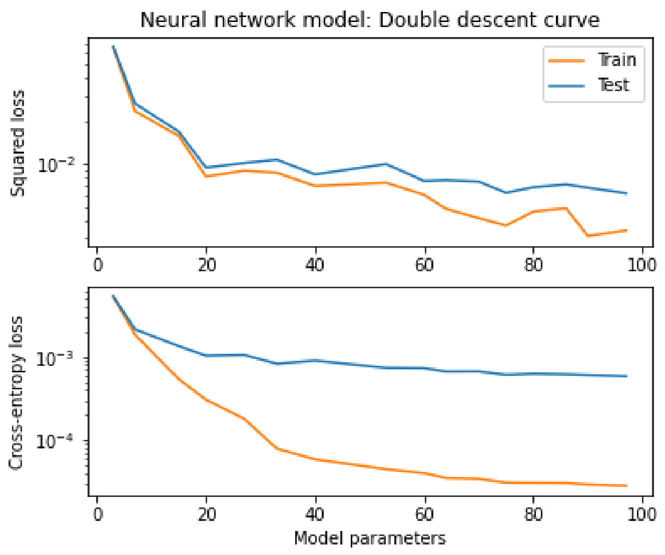

Fig. 2. The double descent curve of a fully connected two-layer neural network. Graph in log-lin scale.

the interpolation threshold, i.e., the first descent in the double descent curve, is not as evident in Fig. 3 as in many double descent curves introduced in [1]. One could argue that this Random Forest model does not display the double descent curve, however, it is evident that the test loss does not increase after the descent in the over-parameterized region with more parameters, as in the textbook U-shaped bias-variance trade-off curve. Therefore, we can conclude that it actually depicts the double descent.

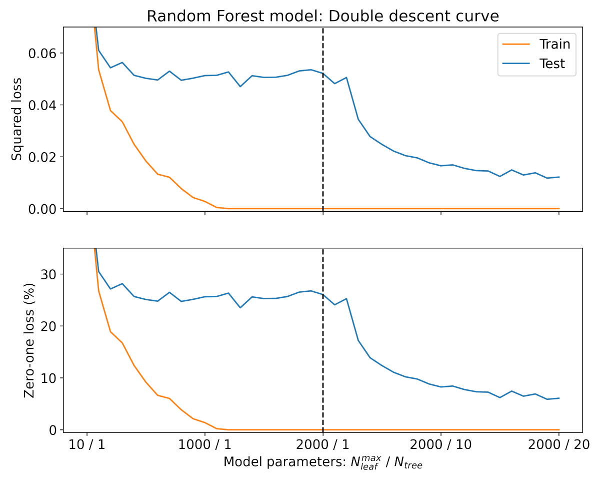

Fig. 3. The double descent curve of a Random Forest model.

IV. Discussion

The reproduction of these three images from [1] clearly shows the double descent phenomenon on the MNIST dataset. However, due to differences in implementation between [1] and our project, mainly because of hardware constraints, the figures are not identical to the ones in the paper. Neverthe-less, we could observe the impact that the augmentation of the number of parameters can have over the generalization error, something that we can also see in the convolutional neural networks such as VGGNet, GoogLeNet, and ResNet, to mention a few, where there are even millions of parameters that have an excellent performance, comparing it with the first architectures of CNN, that for easy tasks as MNIST are enough but for more complex images might not behave as good as the over-parameterized architectures. For improving our results in the fully connected two-layer neural network, the weight-reuse scheme mentioned in II, might have allowed us to do so. In the Random Forest model, increasing the complexity of the model clearly shows the double descent phenomenon, as increasing the number of leaves and trees result in an increase of parameters of the model (more complex model), increasing the trees or the depth of the trees is, as we could see in the results a big augmentation in the parameter set, thus better generalization is achieved.

V. Summary

The reproduction aimed to explore three different types of models with different amount of parameters. In order to observe that when raising the amount of parameters in a machine learning model, we are able to beat the over-fitting behaviour that classical statistical models depict and attain a good generalization error. We can approximate a more complex function that allows us to explain the data much better, which is the principle of the kernel trick and, intuitively, the reason why the neural networks work so well as the number of parameters increase. In an oversimplified way, the neural networks are learning the kernel that help us to make a better classification at the end with the proper loss function.

References

[1] M. Belkin, D. Hsu, S. Ma, and S. Mandal, “Reconciling modern machine-learning practice and the classical bias–variance trade-off,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, jul 2019. [Online]. Available: https://doi.org/10.1073%2Fpnas.1903070116

[2] L. Deng, “The mnist database of handwritten digit images for machine learning research,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[3] A. Rahimi and B. Recht., “Random features for large-scale kernel machines,” *Advances in Neural Information Processing Systems*, p. 1177–1184, - 2008. [Online]. Available: https://people.eecs.berkeley.edu/~brecht/papers/07.rah.rec.nips.pdf