Theoretical Deep Learning Course - Final Report

Gaspard Berthelier gaspard.berthelier@student-cs.com
Firstname Lastname youremail2@mail.com

February 14, 2024

1 Introduction

We decided to present the paper *A U-turn on Double Descent: Rethinking Parameter Counting in Statistical Learning* by Curth et al. [1]. The theme of this paper is the double descent curve, that can be observed in the plots of the risk of machine learning models with respect to the complexity of the model, measured by the number of parameters.

Indeed, as the number of parameters of the model grows, its complexity increases. We usually obtain a U-shaped curve: we start we few parameters and the model is underfitting, we increase the complexity and the model better interpolates, then we reach a point where there are too many parameters and the model is overfitting on the training data. Overfitting can be observed when the training error decreases but its testing error (representative of the risk) increases. A model that overfits does not generalize well.

However, the arrival of neural networks showed that highly parameterized models could achieve close to zero training error and yet achieve excellent generalization. Recent experiments have shown the there exists an interpolating threshold in the overfitting regime after which the test error decreases. This would imply the over-parameterized regime can actually generalize well if we add enough parameters. Moreover, this stands for neural networks but also for simple models such as linear regression up to random forests and boosted trees.

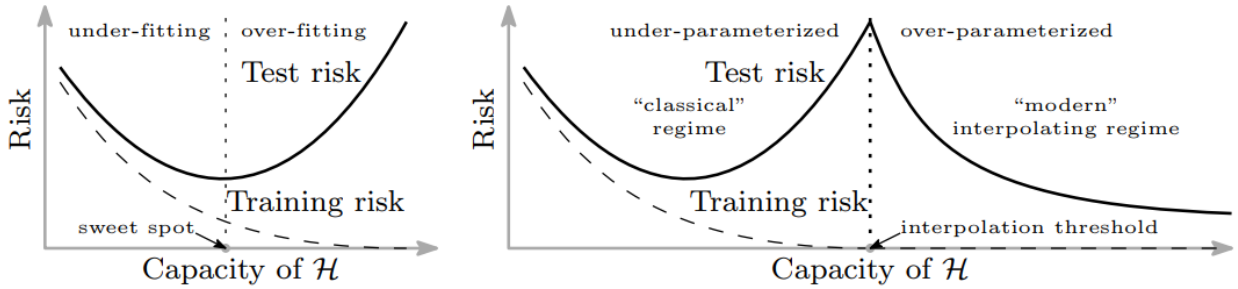

Figure 1: Visualization of double descent. Left: Standard U-curve, Right: Double descent."

Yet, [1] claims double descent is actually only an artefact of how we represent complexity in machine learning models. It corresponds to an unfolding of two separate complexity axis, which once represented correctly yield once again the usual convex error curve. In this report, we will discuss the main theorical contributions of the paper, reproduce the double descent in a linear regression, then observe it on neural network while discussing why this approach doesn't necessarily apply to this type of models.

2 Main theoretical contributions

The paper proposes a new counting method for the number of parameters of a machine learning model - specifically smoother models - which enables to debunk this double descent as complexity grows. The idea is to count the *effective* number of parameters rather that their *raw* number. By doing so, they show that adding more raw parameters in the overparameterized regime does not actually increase the complexity of the model.

First of all, they consider a subset of statistical models called smoother models, where a function f is approximated with $\hat{f}$:

$$\hat{f}(x) = \sum_{i \in I_{train}} \hat{s}^i(x) y_{train,i} = \hat{\mathbf{s}}(x)^T \mathbf{y}_{train} \tag{1}$$

Here, the predicted output for a given input x is a weighted combination of the training outputs $y_{train,i}$. Many models can actually be re-expressed in this form : linear regression, kNN, tree-based methods, etc. In linear regression for instance, we usually train a model to minimize the following quantity:

$$||\phi(\mathbf{X}_{train})\beta - \mathbf{y}_{train}||_2^2 \tag{2}$$

where $\phi(\mathbf{X}_{train})$ is of size $N_{train} \times M$ with M the number of features. ϕ acts as a feature augmentation of the initial X_{train}, usually done via feature engineering but can also be done with Random Fourier Features [2] when no linear relations are known. β is the vector of coefficients to train (M parameters) and $\mathbf{y}_{train,i}$ the values of the function f to approximate at points $\mathbf{X}_{train}[i,:]$.

We can prove using linear algebra that there are two distinct regimes depending on the number of features (complexity of the linear model):

- $M \leq N : \beta_{opt} = (\phi(\mathbf{X}_{train})^T \phi(\mathbf{X}_{train}))^{-1} \phi(\mathbf{X}_{train})^T y_{train} = H_{opt}(X_{train}) y_{train}$
- $M > N : \beta_{min\,norm} = \phi(\mathbf{X}_{train})^T (\phi(\mathbf{X}_{train}) \phi(\mathbf{X}_{train})^T)^{-1} y_{train} = H_{min\,norm}(X_{train}) y_{train}$

where β_{opt} is the unique solution to the problem and $\beta_{min\,norm}$ is one solution among an infinite set of solutions which uniquely minimizes $||\beta||_2^2$. Note that the previous propreties make the assumption that $\phi(\mathbf{X}_{train})^T \phi(\mathbf{X}_{train})$ or $\phi(\mathbf{X}_{train}) \phi(\mathbf{X}_{train})^T$ are invertible, which is true if the matrix $\phi(X_{train})$ is of either column or row full rank, ie. rank($\phi(X_{train})$)=M or rank($\phi(X_{train})$=N. This assumption is usually held to be true in machine learning, but we keep in mind that theoretical results may not be as straightforward in practice depending on the dataset.

Since the output of a given x is predicted as $\phi(x)^T \beta$, we get the smoother expressions:

$$\hat{s}(x) = \begin{cases} \mathrm{H}_{opt}(X_{train})x & \text{if } \mathrm{M} \leq N \\ \mathrm{H}_{min\,norm}(X_{train})x & \text{if } \mathrm{M} > \mathrm{N} \end{cases} \tag{3}$$

The paper's proposed method to count the effective number of parameters is the following:

$$M_{eff}(I) = \frac{|I_{train}|}{|I|} \sum_{i \in I} ||\hat{\mathbf{s}}(x_i)||_2^2 \tag{4}$$

We can verify that $M_{eff} = M$ when applying the definition to I_{train} with linear regression, since $\hat{s}(x_i) = 1$. We expect this value to be different on unseen data though. We will verify this through the experiments in Sec. 3.

The reasons for this definition of the effective number of parameters is pretty straight forward: since the prediction is a weighted sum of the $y_{train,i}$, it is only natural to consider the sum of their importances (averaged on the test dataset). Also, with outputs generated with homoskedastic variance σ^2, we would have $V(f(x)) = ||\hat{\mathbf{s}}||^2\sigma^2$, which implies $\frac{1}{|I|}\sum_{i\in I} V(\hat{f}(x_i)) = \frac{\sigma^2}{|I_{train}|}M_{eff}(I)$. This shows M_{eff} represents some sort of expected average variance for unseen data. Its main advantage is that it can be computed on any set of set of inputs I.

Of course, the formula for $\hat{s}(x)$ must be known in advance in order to measure M_{eff} and it differs for different families of models (linear, trees, ...). Also, its expression usually involves multiple parameters : the projection matrices for linear regression (H_{opt} & $H_{min\,norm}$), the number of leaves & the number of trees for forests, etc. As it turns out, plotting the error for one parameter while keeping others fixed yield convex curves with no double descent. The cause of the double descent is that augmenting the *raw* number of parameters usually modifies the other *hidden* parameters, reducing in practice the value of M_{eff} (aka the complexity) or at least keeping it constant while improving the quality of the effective parameters. For instance, in linear regression with the full rank hypothesis : the rank of $\phi(X_{train})$ is equal to M to start with and then to N in the interpolation regime. When applying a Singular Value Decomposition, we get N singular vectors and $M - N$ redundant vectors. Augmenting M only adds to the redundant vectors. M_{eff} remains close to M in practice, but the M singular vectors are more representative. The proof this phenomenon is provided in the appendixes of [1].

3 Experiments

We reproduced the paper's main experiment : observing the double descent on linear regression and reobserving the convex error curve when changing the axis to the effective number of parameters. We then experimented on double descent for neural networks. We observed the phenomenon but found no direct relation between the interpolation threshold and the number of training samples. Finally, we discuss why counting the number of effective parameters in neural networks is challenging.

3.1 Reproducing double descent in linear regression

In this experiment, we follow the method provided by the article : we use Random Fourier Features to control the number of features of the input. We experiment with 1000 samples from a generated synthetic dataset and using scikit-learn library [3] for the linear regression. We can see on the left picture of Fig. 2 that there is indeed a double descent curve and an interpolation threshold at $M = N_{train}$.

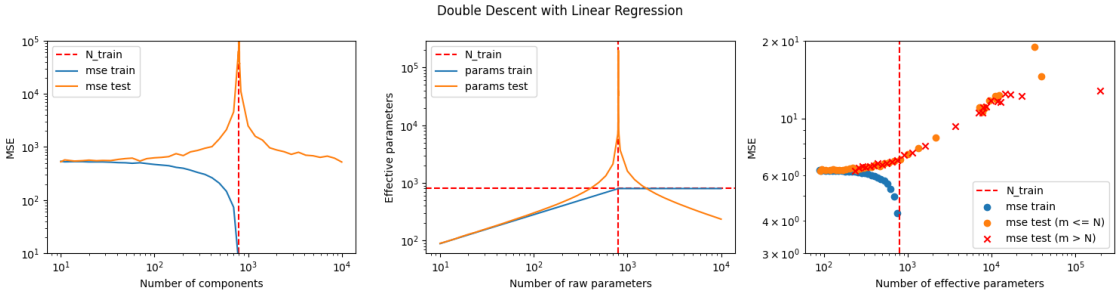

Figure 2: Visualization of double descent in linear regression on a synthetic dataset.

The curve in the middle is the *effective* number of parameters with respect to the *raw* number of parameters, computed using Eq. 3. We see that the number of effective parameters on test data grows exponentially as it gets closer to the number of training samples, but then decreases after this interpolation threshold. If we now plot the loss as a function of the effective number of parameters, we get the curve on the far right which is convex once again. Fig. 4 in appendix shows similar results on the MNIST dataset.

3.2 Observing double descent in neural networks

We now want to visualize double descent in a neural network. We experimented with one and two layer neural networks, with ReLU activation and Adam optimizer, trained on the MNIST dataset. To increase the number of parameters, we increased the size of the first layer, since the total number of parameters is $(N_{pixels} + 1) \times N_0 + (N_0 + 1) \times N_1 + N_1 \times 10$, with N_0 and N_1 the number of neurons in the first and second hidden layers. We trained the models for around 30 epochs in order to reach a satisfactory loss while preventing overfitting. We also averaged the results of a few different random states. Fig. 3 shows our results for the 1-layer neural network.

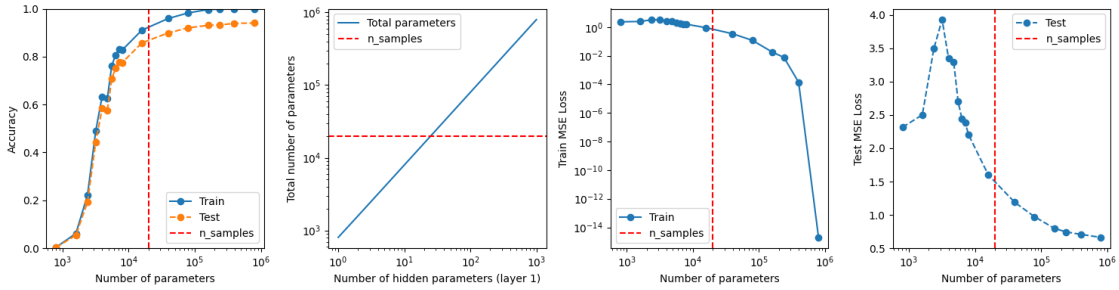

Figure 3: 1-layer NN. Double-descent can be observed on the test loss on the far right.

We can indeed observe an interpolation threshold, which happens before reaching $M = N_{train}$ though. Similar behaviors happen when adding a second layer and varying its size, as well as varying the number of training samples (see Fig. 5 and Fig. 6 in appendix).

3.3 Discussion on the article's model

As stated in Sec. 2, the article considers a subset of statistical models called smoother models that approximate a function f with $\hat{f}(x) = \hat{\mathbf{s}}(x)^T \mathbf{y}_{train}$. Linear models fall in this category, however the ReLU activation in the 1-layer neural network introduces non-linearities. Therefore, the method introduced in [1] to count the number of effective parameters cannot be directly applied.

However, Neural Tangent Kernels [4] are another family of models which approximate the behavior of usual neural network, with an infinite width. It turns out these models can be expressed as smoothers. Studying double descent in NTK models may be a promising direction to understand double descent in neural networks and to find a complexity measure which enables us to find a convex error curve once again.

4 Conclusion

The paper *A U-turn on Double Descent: Rethinking Parameter Counting in Statistical Learning* by Curth et al. discusses traditional understanding of the U-shaped curve in the testing error with respect to the number of parameters, which has been challenged by the discovery of a double descent phenomenon, particularly in the overparameterized regime.

The main theoretical contribution of the paper is to introduce a novel counting method for the effective number of parameters in smoother models. By redefining complexity in terms of effective parameters, the authors show that increasing the raw parameter count in the overparameterized regime does not necessarily increase model complexity.

We reproduced the article's experiments and found that for linear regression, one can indeed observe an interpolation regime when parameters exceed the number of samples in the train dataset. When plotting the error with respect to the effective number of parameters, we observe a more traditional U-shaped curve with respect to complexity.

While we observed that there also exists a double descent in neural networks, the paper's proposed counting method of parameters doesn't directly apply to this kind of models. Finding revelant complexity axes to count the number of effective parameters remains an open problem.

References

[1] A. Curth, A. Jeffares, and M. van der Schaar, "A u-turn on double descent: Rethinking parameter counting in statistical learning," 2023.

[2] G. Gundersen, "Random fourier features," 2019.

[3] "Scikit learn linear regression,"

[4] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *CoRR*, vol. abs/1806.07572, 2018.

5 Appendix

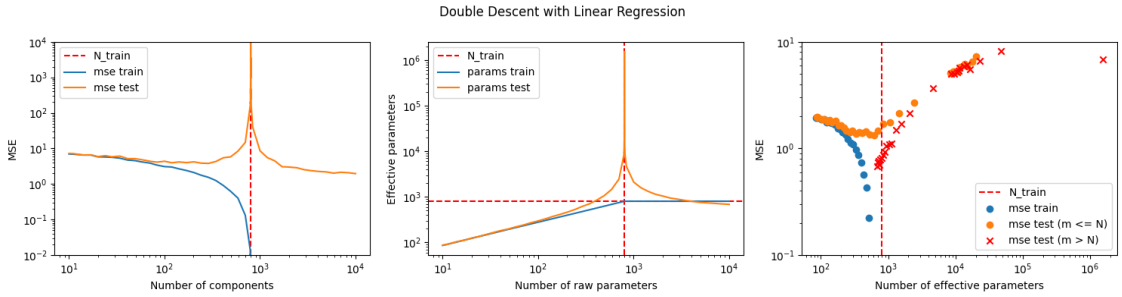

Figure 4: Double Descent with linear regression on MNIST dataset

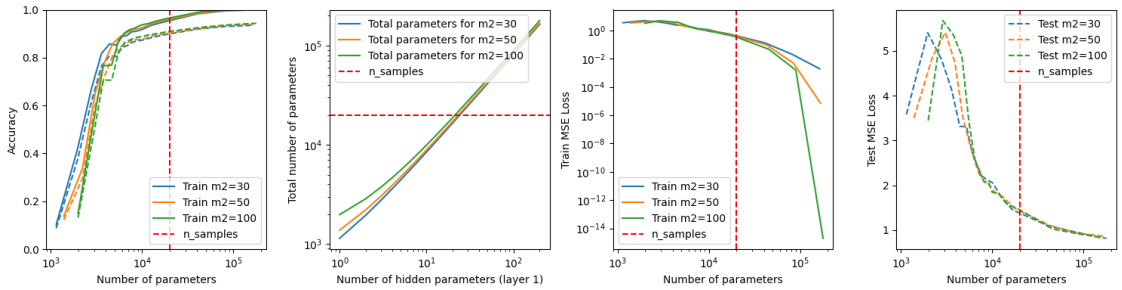

Figure 5: 2-layer NN : varying the size of the second hidden layer

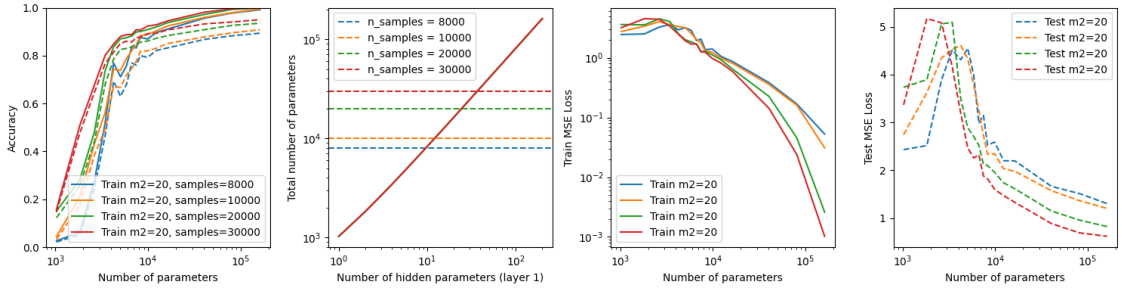

Figure 6: 2-layer NN : varying the number of training samples