DSGA-1020 Mathematical Statistics Final Project

Reconciling modern machine-learning practice and the classical bias–variance trade-off (Belkin et al., 2019)

Arthur Lok
aml731@nyu.edu

Giulio Duregon
gjd9961@nyu.edu

1 Introduction

In *"Reconciling modern machine learning practice and the bias-variance trade-off"* (Belkin et al., 2019), the authors introduce "double descent", a phenomenon that occurs when methods continue to perform well on unseen testing data even when perfectly fitting training data, contrary to the classical understanding of the bias-variance trade-off. In this paper, we first review the bias-variance trade-off using both results from lecture notes and tying in the narrative from Belkin et al. We then explain double descent and how it implies that traditional machine learning ideas around limiting the function class to prevent overfitting training data may be misguided. Following this, we review some of the key empirical evidence where methods like neural networks and decision tree ensembles follow this behavior as well as the reasons why they do. Lastly, we conduct our own experiment using Random Forest Classifiers to recreate the double descent phenomenon as well as visualize the regularizing effect of ensembling which contributes to improved test performance.

2 Bias-Variance trade-off

The bias-variance trade-off is one of the core tenets of statistics and machine learning theory. The prediction problem for which it is applicable is stated as follows: we have some observations $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$ drawn from an unseen distribution over $R^d \bigotimes R$. We choose a function class $\mathcal{H}$ which is able to model our data appropriately. The predictor $h_n : \mathbb{R}^n \to \mathbb{R}$ is selected from $\mathcal{H}$ via the empirical risk minimization framework (ERM) which tells us to choose h_n such that it minimizes risk over a loss function, $\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(h(x_i), y)$. While we only have access to our observations and select h_n based on its training risk, our goal is to be able to predict y successfully for an unseen x.

A natural loss function for regression problems is the mean squared error (MSE).

$$\mathcal{R}_y(\hat{y}) = \mathbb{E}_y(\hat{y} - y)^2 \tag{1}$$

We see via Proposition 5.6 that the MSE can be decomposed as shown below into bias and variance.

$$\mathcal{R}_y(\hat{y}) = Var_\theta(\hat{y}) + \mathbb{E}^2_y(\hat{y} - y) \tag{2}$$

This is where the bias-variance trade-off arises. Bias clearly decreases as we expand $\mathcal{H}$ while remaining a superset of functions included in smaller function classes. To take an example, we can use linear regression. In linear regression, we aim to fit $\hat{\beta} \in R^d$ such that

$$y = \hat{\beta}^T x + \epsilon \tag{3}$$

where $MSE(y, \hat{y})$ is minimized over our training set and ϵ is Gaussian error where $\mathbb{E}[\epsilon|X] = 0$ and $\sigma^2 = \mathbb{E}[\epsilon^2|X]$. Our function class in this case includes all such models using $\beta \in R^d$. Suppose we only choose to model our data using $d-1$ covariates, excluding the d-th covariate, and limiting our function class $\mathcal{H}'$ appropriately to $\beta' \in R^{d-1}$. Any $h' \in \mathcal{H}'$ can be recovered from if the d-th coordinate of the corresponding set of $h \in \mathcal{H}$ is set to 0. So, increasing our function class increases the explanatory power of our models and decreases bias.

At the same time, variance increases in this setting as the dimensionality of our model increases. By Proposition 7.6, we know that variance satisfies the below equation, and we have shown that higher d returns a higher lower bound on the quantity.

$$\mathbb{E}[||\hat{\beta} - \beta||^2] = \frac{\sigma}{n} tr(\mathbb{E}[\Sigma_X^{-1}]) \tag{4}$$

The tools that we have to control this trade-off include explicitly limiting our function class or implicitly doing so using regularization. Regularization is a method of deliberately increasing bias by introducing a penalty term into our loss function which penalizes higher norm coefficients in our models. This method aligns with an inductive bias that smaller norm solutions are more "natural". Its advantage is also that it allows us to use our full dataset compared to dropping covariates and limiting the dimensionality of our function class.

Methods of reducing our overall risk in this class have assumed that we must necessarily increase bias to reduce our variance. In high dimensionality, that is often a trade-off we want to make. In section 8.2, we have shown that in ridge regression (Eq. 5), bias and variance can be upper-bounded respectively by Equation 6 and Equation 7:

$$\underset{\beta \in \mathbb{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left(x_i^T \beta - y_i\right)^2 + \lambda \left\|\beta\right\|^2 \tag{5}$$

$$\left\|\mathbb{E}_\beta \hat{\beta}_\lambda - \beta\right\|^2 \leq \left\|\beta\right\|^2 \tag{6}$$

$$\mathbb{E}[\left\|\hat{\beta}_\lambda - \beta\right\|^2 | x_1, ..., x_n] \leq \frac{\sigma^2}{n} \frac{d}{\lambda} \tag{7}$$

If λ is not introduced, we will have an unbiased estimator of β and no bias in the standard linear regression setup. With λ we are able to upper bound variance at the cost of incurring bias upper bounded by the squared norm of β. However, this may be a worthy trade-off with high d or σ.

In the traditional understanding of our learning curve in Figure 1, using regularization would help when we are approaching the interpolation threshold from the left. A highly rich function class allows expression of complex patterns in the data but may also increase variance by learning spurious relationships in the sample, increasing our test risk. Without knowledge of the test risk beyond interpolation, practitioners are commonly recommended to minimize test risk by involve incurring some amount of training risk to reach the trough of the first curve.

3 Double Descent

Belkin et al. show that modern machine learning methods often show a different relationship between $\mathcal{H}$ and risk in practice, namely the "double descent" curve. Large neural networks have shown that they can both generalize and achieve near zero training risk. The point of interpolation, or where an estimator achieves zero training risk, is traditionally understood where test risk is the highest. As we showed in section 2, training risk is typically minimized with a larger function class that allows for richer expression of underlying structure in the data. However, it also increases the tendencies to overfit patterns in training data that may not generalize.

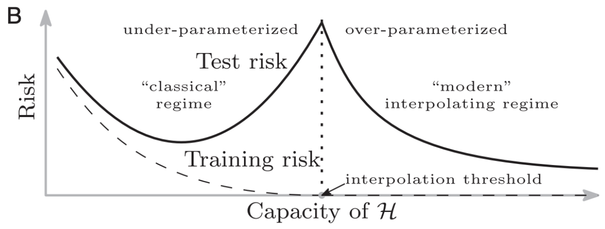

Figure 1: Illustration of "double descent" risk curve taken from the original paper (Belkin et al., 2019)

The part that seems to be missing in classical understanding of machine learning theory is what if we continue to expand $\mathcal{H}$ after this point. With methods like neural networks, ensemble methods of decision trees, and other non-linear predictors, we see that test performance in practice actually decreases after this point as well. The idea presented behind the phenomenon is that at the interpolation threshold, there is no guarantee that $\mathcal{H}$ contains the smallest norm and most regular interpolating predictor $h^*(x)$. If we continue to expand $\mathcal{H}$, we can get closer to approximating $h^*(x)$, explaining the second descent curve. In our experiment in section 5, we aim to recreate this phenomenon and also examine whether the properties of h at different size function classes hold true to this idea.

In Example 8.5, we showed that ridge regression is equivalent to

$$\underset{\beta \in \mathbb{R}^p, ||\beta|| \leq \tau}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left(x_i^T \beta - Y_i\right)^2 \tag{8}$$

In other words, optimization under regularization is equivalent to optimizing within a smaller norm parameter space. With the methods where

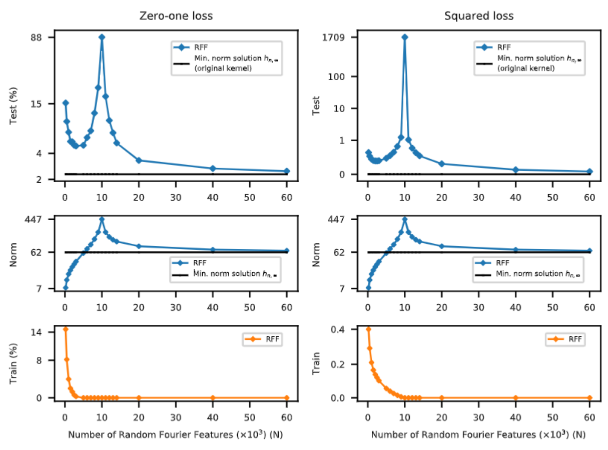

Figure 2: Risk curves for RFF model on MNIST showing double descent. The interpolation threshold is reached at $N = 10^4$ from (Belkin et al., 2019)

we observe the double descent phenomenon, the training algorithms that are used exhibit a similar regularizing effect. When there are multiple minimizing functions, we see the smoother function or the function with smaller norm coefficients being chosen even as the function space increases. Belkin et al. notes that the inductive bias of the training algorithms to find smaller or smoother predictors appears to be successful on many real world problems, explaining why the double descent phenomenon occurs.

4 Empirical Evidence

The authors give empirical evidence of the double descent curve with different methods. They first consider the model family of random Fourier features, $\mathcal{H}^N$ consisting of functions $h : \mathbb{R}^d \to \mathbb{C}$, of the form

$$h(x) = \sum_{k=1}^{N} a_k \phi(x; v_k) \tag{9}$$

$$\text{where } \phi(x; v) := \exp \sqrt{-1}\langle v_k, x\rangle$$

where the vectors $v_1, ..., v_n$ are sampled from independently from the standard normal distribution in $\mathcal{R}^d$. When performing ERM over H_N while $N >> n$, there does not exist a unique minimizer, and therefore the minimizer of the squared loss with the minimum l_2 norm of the coefficients $a_1, ..., a_n$ was chosen. After the method described was performed on a subset of the MNIST dataset, the results showed the double descent phenomenon, with the interpolation threshold at $N = n$.

RFFs can be viewed as a class of 2-layer neural networks, but the authors showed that the double descent behavior holds for general multilayer neural networks as well. By increasing the number of parameters, they achieved an interpolation threshold at nk parameters for a k-class classification problem. Past this point, test performance continues to improve.

Belkin et al. also experimented with ensembles of decision trees. In this setup, the function class capacity is defined by both the number of leaves N and by the number of trees once $N > n$. Once again, by averaging an increasing number of interpolating trees, we are able to observe the double descent curve. The authors note that the averaging of interpolating trees allow for more flexibility and smoothness of the resulting function at the same time. In the below section, we aim to recreate a similar experiment using a Random Forest function class as well as provide a perspective on why test performance improves past the interpolation threshold.

5 Experiment

5.1 Dataset

As done in (Belkin et al., 2019), for our training and test data we used the MNIST (Lecun et al., 1998) dataset. The dataset is comprised of 70,000 images of hand-written digits, each 28×28 pixels. This dataset is high dimensional, as each observation in our dataset, x, is represented by a feature vector $x \in \mathbb{R}^{784}$. We split the dataset into 60,000 training observations, and saved 10,000 in a test set for cross-validation purposes.

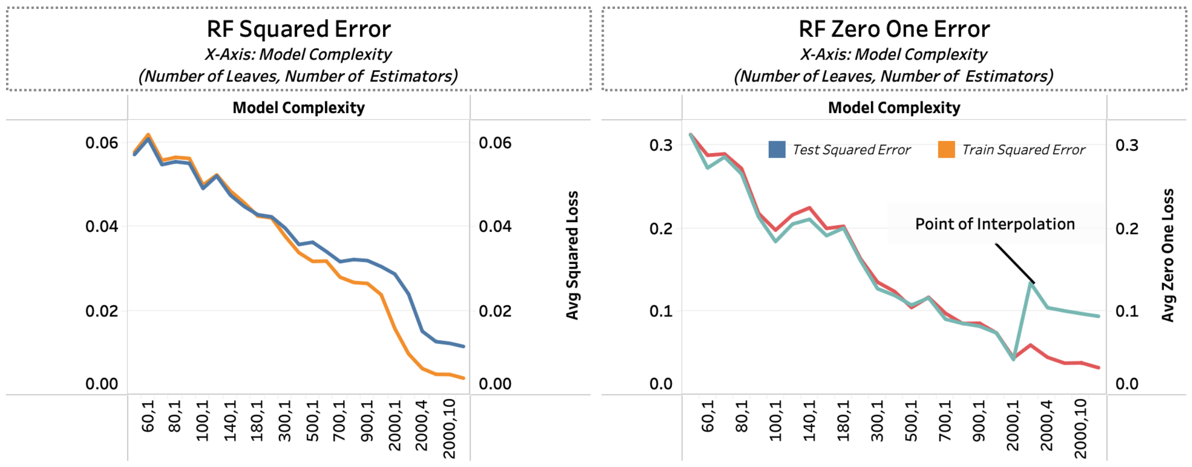

Figure 3: Authors recreation of experiments visualized in Figure 2

5.2 Software

Seeking to recreate the authors results, we leveraged Scikit-learn's (Pedregosa et al., 2011) ensemble package to recreate a Random Forest Classifier. The Random Forest Classifier implementation in Scikit-learn has two methods for prediction post training. The first prediction method is to return the model's most confident prediction for input x_i, which returns $\hat{y_i} \in \{0, \ldots, k\}$ where k is the number of classes. The second prediction method returns all of the models predicted probabilities for each class label, $\hat{y_i}' \in \mathbb{R}^k$. The former is normally called when evaluating the zero-one loss of a model, while the latter is traditionally used for mean-squared-error loss metrics as described in Equation 10.

5.3 Random Forest Classifier

A random forest classifier is an ensemble of decision trees each trained on a bootstrapped sample of data and subset of features from said sample. At time of inference, each tree's decision path is evaluated to predict a classification label, with majority vote of the trees resulting in the finalized predicted class.

The foundation of the Random Forest algorithm is the bootstrap method. This method involves sampling with replacement from a dataset to create synthetic data subsets, called bootstrapped samples. A subset of each bootstrapped samples features is taken and then used by a decision tree to fit a model.

The bootstrap method has been observed to reduce variance without increasing bias, proving to be an invaluable tool when constructing any form of decision trees as they are low bias, high variance models. Regularization through limiting the number of leaf nodes a tree could have acts as an important form of regularization. Intuitively, this makes sense, as if left unbounded, any decision tree can create an arbitrary (and potentially infinite) set of decisions to perfectly fit any dataset.

For a random forest, this regularization is achieved by altering two main hyper-parameters: the maximum number of leaves it can have, and the amount of trees that comprise its ensemble.

5.4 Recreation of Double Descent

We created several different implementations of a random forest model, with each implementation increasing in model complexity. Each tasked with problematically predicting a label, $k; \in \{0, 1, \ldots, 9\}$, with loss evaluated using the mean squared error and zero-one loss as done in (Belkin et al., 2019):

$$\begin{aligned} MSE &:= \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} \|\hat{y}_i - y_i\|^2\right) \\ \textit{Zero-One Loss} &:= \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\hat{y}_i \neq y_i}\right) \end{aligned} \tag{10}$$

We were successful in recreating the double descent phenomenon as described in (Belkin et al., 2019), visualized in Figure 3. For the settings used in the paper, the double descent phenomenon is not that apparent for the squared error. However, looking at the zero-one loss, the double descent curve is apparent. For zero-one loss, the inflection point occurring at 2,000 leafs with 1 tree, and the

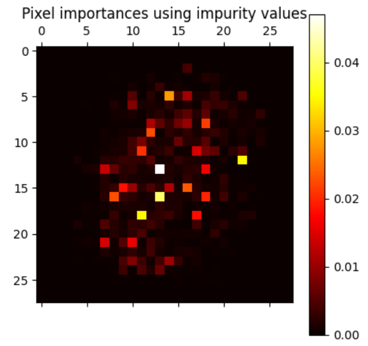

Figure 4: Feature importance from Point of Inflection

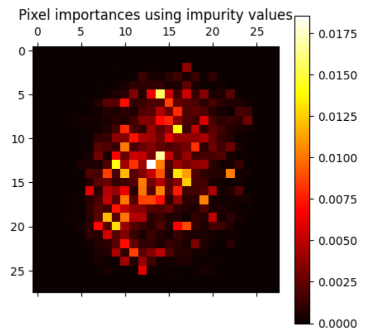

Figure 5: Feature importance from Point of Interpolation

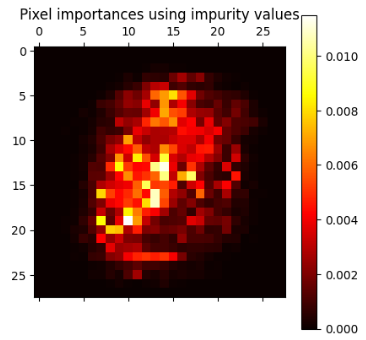

Figure 6: Feature importance from best performance

point of interpolation occurred at 2,000 leafs with 2 trees.

Intrigued by this phenomenon, we investigated the relevant importance of each pixel in our decision trees at the following configurations:

1. Point of Inflection, as seen in Figure 4 (2000 leafs, 1 tree)

2. Point of Interpolation, as seen in Figure 5 (2000 leafs, 2 tree)

3. Point of best performance, as seen in Figure 6 (2000 leafs, 20 tree)

The relevant importance of each pixel, 784 in total as each image is a 28×28 grid of pixels, is plotted in Figures 4-6. Relative importance is determined by a combination of fraction of samples as well as the decrease of impurity by splitting on the feature. The values on the plot are all positive and sum up to 1.

Belkin et al. noted that allowing for the individual trees to interpolate would allow for a more flexible fit than limiting the tree size. Increasing the number of leaves in each tree past interpolation overall results in a smoother function after ensembling the trees to obtain the final predictor compared to functions prior to the interpolation threshold. While it is difficult to visualize a decision boundary in high-dimensional space, the relative importance plots corroborate the ideas presented by the authors. We can see that the functions consisting of interpolating trees take advantage of many more pixels in its predictions. It is also interesting that the best performing tree past interpolation not only places importance on more pixels compared to the point of interpolation, but also values more pixels at a similar level of relative importance. This evidence supports the claim that smoother functions can be found past interpolation which perform better in terms of test risk.

6 Conclusion

Belkin et al. describe the ”double descent” curve in their paper which indicates a different relationship between the capacity of the function class and test risk than was previously understood in the traditional machine learning regime. To approach this topic, we first reviewed our treatment of the bias-variance trade-off from class. In the empirical risk minimization framework, several of our results show that we expect to pay in bias if we want to reduce the variance that is associated with high dimensional data. This inspires the traditional advice given to practitioners that recommends that training risk must be incurred to avoid overfitting and achieve the best out of sample performance. However, the author show that with several modern methods, predictor can both interpolate and achieve even greater out of sample performance than the optimal non-interpolating predictor. If we continue to expand our function class in ERM and the training algorithm exhibits a regularizing effect, we can find a smoother or smaller norm predictor that performs better. Finally, we recreated some results from the paper using MNIST and Random Forest Classifiers. We performed our own analysis of the properties of the functions at critical points to examine how they differed in smoothness and expressiveness.

References

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. 2019. Reconciling modern machine-learning practice and the classical bias–variance trade-off.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.