Reconciling modern machine learning practice and the bias-variance trade-off

Mikhail Belkin[a], Daniel Hsu[b], Siyuan Ma[a], and Soumik Mandal[a]

[a]*The Ohio State University, Columbus, OH*
[b]*Columbia University, New York, NY*

September 12, 2019

Abstract

Breakthroughs in machine learning are rapidly changing science and society, yet our fundamental understanding of this technology has lagged far behind. Indeed, one of the central tenets of the field, the bias-variance trade-off, appears to be at odds with the observed behavior of methods used in the modern machine learning practice. The bias-variance trade-off implies that a model should balance under-fitting and over-fitting: rich enough to express underlying structure in data, simple enough to avoid fitting spurious patterns. However, in the modern practice, very rich models such as neural networks are trained to exactly fit (i.e., interpolate) the data. Classically, such models would be considered over-fit, and yet they often obtain high accuracy on test data. This apparent contradiction has raised questions about the mathematical foundations of machine learning and their relevance to practitioners.

In this paper, we reconcile the classical understanding and the modern practice within a unified performance curve. This "double descent" curve subsumes the textbook U-shaped bias-variance trade-off curve by showing how increasing model capacity beyond the point of interpolation results in improved performance. We provide evidence for the existence and ubiquity of double descent for a wide spectrum of models and datasets, and we posit a mechanism for its emergence. This connection between the performance and the structure of machine learning models delineates the limits of classical analyses, and has implications for both the theory and practice of machine learning.

E-mail: mbelkin@cse.ohio-state.edu, djhsu@cs.columbia.edu, masi@cse.ohio-state.edu, mandal.32@osu.edu

1 Introduction

Machine learning has become key to important applications in science, technology and commerce. The focus of machine learning is on the problem of prediction: given a sample of training examples $(x_1, y_1), \ldots, (x_n, y_n)$ from $\mathbb{R}^d \times \mathbb{R}$, we learn a predictor $h_n \colon \mathbb{R}^d \to \mathbb{R}$ that is used to predict the label y of a new point x, unseen in training.

The predictor h_n is commonly chosen from some function class $\mathcal{H}$, such as neural networks with a certain architecture, using *empirical risk minimization (ERM)* and its variants. In ERM, the predictor is taken to be a function $h \in \mathcal{H}$ that minimizes the *empirical (or training) risk* $\frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i)$, where ℓ is a loss function, such as the squared loss $\ell(y', y) = (y' - y)^2$ for regression or zero-one loss $\ell(y', y) = \mathbb{1}_{\{y' \neq y\}}$ for classification.

The goal of machine learning is to find h_n that performs well on new data, unseen in training. To study performance on new data (known as *generalization*) we typically assume the training examples are sampled randomly from a probability distribution P over $\mathbb{R}^d \times \mathbb{R}$, and evaluate h_n on a new test example (x, y) drawn independently from P. The challenge stems from the mismatch between the goals of minimizing the empirical risk (the explicit goal of ERM algorithms, optimization) and minimizing the *true (or test) risk* $\mathbb{E}_{(x,y)\sim P}[\ell(h(x), y)]$ (the goal of machine learning).

Conventional wisdom in machine learning suggests controlling the capacity of the function class $\mathcal{H}$ based on the *bias-variance trade-off* by balancing *under-fitting* and *over-fitting* (cf., [17, 21]):

1. If $\mathcal{H}$ is too small, all predictors in $\mathcal{H}$ may *under-fit* the training data (i.e., have large empirical risk) and hence predict poorly on new data.

2. If $\mathcal{H}$ is too large, the empirical risk minimizer may *over-fit* spurious patterns in the training data resulting in poor accuracy on new examples (small empirical risk but large true risk).

The classical thinking is concerned with finding the "sweet spot" between under-fitting and over-fitting. The control of the function class capacity may be explicit, via the choice of $\mathcal{H}$ (e.g., picking the neural network architecture), or it may be implicit, using regularization (e.g., early stopping). When a suitable balance is achieved, the performance of h_n on the training data is said to *generalize* to the population P. This is summarized in the classical U-shaped risk curve, shown in Figure 1(a) that has been widely used to guide model selection and is even thought to describe aspects of human decision making [18]. The textbook corollary of this curve is that "a model with zero training error is overfit to the training data and will typically generalize poorly" [21, page 221], a view still widely accepted.

Yet, practitioners routinely use modern machine learning methods, such as large neural networks and other non-linear predictors that have very low or zero training risk. In spite of the high function class capacity and near-perfect fit to training data, these predictors often give very accurate predictions on new data. Indeed, this behavior has guided a best practice in deep learning for choosing neural network architectures, specifically that the network should be large enough to permit effortless zero loss training (called *interpolation*) of the training data [34]. Moreover, in direct challenge to the bias-variance trade-off philosophy, recent empirical evidence indicates that neural networks and kernel machines trained to interpolate the training data obtain near-optimal test results even when the training data are corrupted with high levels of noise [42, 4].

The main finding of this work is a pattern for how performance on unseen data depends on model capacity and the mechanism underlying its emergence. This dependence, empirically witnessed with important model classes including neural networks and a range of datasets, is summarized in the "double descent" risk curve shown in Figure 1(b). The curve subsumes the classical U-shaped risk curve from Figure 1(a) by extending it beyond the point of interpolation.

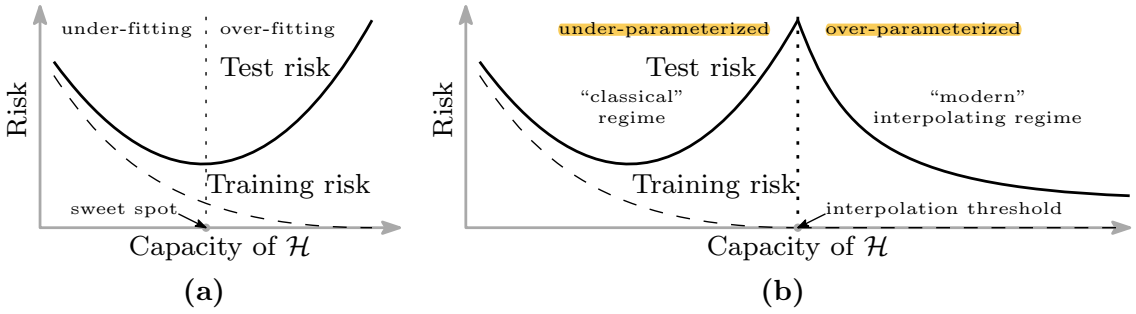

Figure 1: **Curves for training risk (dashed line) and test risk (solid line).** (**a**) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (**b**) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

When function class capacity is below the "interpolation threshold", learned predictors exhibit the classical U-shaped curve from Figure 1(a). (In this paper, function class capacity is identified with the number of parameters needed to specify a function within the class.) The bottom of the U is achieved at the sweet spot which balances the fit to the training data and the susceptibility to over-fitting: to the left of the sweet spot, predictors are under-fit, and immediately to the right, predictors are over-fit. When we increase the function class capacity high enough (e.g., by increasing the number of features or the size of the neural network architecture), the learned predictors achieve (near) perfect fits to the training data—i.e., interpolation. Although the learned predictors obtained at the interpolation threshold typically have high risk, we show that increasing the function class capacity beyond this point leads to decreasing risk, typically going below the risk achieved at the sweet spot in the "classical" regime.

All of the learned predictors to the right of the interpolation threshold fit the training data perfectly and have zero empirical risk. So why should some—in particular, those from richer functions classes—have lower test risk than others? The answer is that the capacity of the function class does not necessarily reflect how well the predictor matches the *inductive bias* appropriate for the problem at hand. For the learning problems we consider (a range of real-world datasets as well as synthetic data), the inductive bias that seems appropriate is the regularity or smoothness of a function as measured by a certain function space norm. Choosing the smoothest function that perfectly fits observed data is a form of Occam's razor: the simplest explanation compatible with the observations should be preferred (cf. [38, 6]). By considering larger function classes, which contain more candidate predictors compatible with the data, we are able to find interpolating functions that have smaller norm and are thus "simpler". Thus increasing function class capacity improves performance of classifiers.

So this is saying that the possible set of large models contains a larger set of possible predictors (possible set of model weights) that are able to find a simpler function to approximate? Isn't this essentially just a better form of weight search/ initialization (or a better form of weight search/ initialization if equivalent to this?)

<-- Can you not just add a smoothness regularizer to your model and get the same results then?

Related ideas have been considered in the context of margins theory [38, 2, 35], where a larger function class $\mathcal{H}$ may permit the discovery of a classifier with a larger margin. While the margins theory can be used to study classification, it does not apply to regression, and also does not predict the second descent beyond the interpolation threshold. Recently, there has been an emerging recognition that certain interpolating predictors (not based on ERM) can indeed be provably statistically optimal or near-optimal [3, 5], which is compatible with our empirical observations in the interpolating regime.

In the remainder of this article, we discuss empirical evidence for the double descent curve, the

mechanism for its emergence and conclude with some final observations and parting thoughts.

2 Neural networks

In this section, we discuss the double descent risk curve in the context of neural networks.

Random Fourier features. We first consider a popular class of non-linear parametric models called *Random Fourier Features* (*RFF*) [30], which can be viewed as a class of two-layer neural networks with fixed weights in the first layer. The RFF model family $\mathcal{H}_N$ with N (complex-valued) parameters consists of functions $h\colon \mathbb{R}^d \to \mathbb{C}$ of the form

$$h(x) = \sum_{k=1}^{N} a_k \phi(x; v_k) \quad \text{where} \quad \phi(x; v) := e^{\sqrt{-1}\langle v, x\rangle},$$

and the vectors $v_1, \ldots, v_N$ are sampled independently from the standard normal distribution in $\mathbb{R}^d$. (We consider $\mathcal{H}_N$ as a class of real-valued functions with $2N$ real-valued parameters by taking real and imaginary parts separately.) Note that $\mathcal{H}_N$ is a randomized function class, but as $N \to \infty$, the function class becomes a closer and closer approximation to the Reproducing Kernel Hilbert Space (RKHS) corresponding to the Gaussian kernel, denoted by $\mathcal{H}_\infty$. While it is possible to directly use $\mathcal{H}_\infty$ (e.g., as is done with kernel machines [8]), the random classes $\mathcal{H}_N$ are computationally attractive to use when the sample size n is large but the number of parameters N is small compared to n.

Our learning procedure using $\mathcal{H}_N$ is as follows. Given data $(x_1, y_1), \ldots, (x_n, y_n)$ from $\mathbb{R}^d \times \mathbb{R}$, we find the predictor $h_{n,N} \in \mathcal{H}_N$ via ERM with squared loss. That is, we minimize the empirical risk objective $\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i)^2$ over all functions $h \in \mathcal{H}_N$. When the minimizer is not unique (as is always the case when $N > n$), we choose the minimizer whose coefficients $(a_1, \ldots, a_N)$ have the minimum ℓ_2 norm. This choice of norm is intended as an approximation to the RKHS norm $\|h\|_{\mathcal{H}_\infty}$, which is generally difficult to compute for arbitrary functions in $\mathcal{H}_N$. For problems with multiple outputs (e.g., multi-class classification), we use functions with vector-valued outputs and sum of the squared losses for each output.

In Figure 2, we show the test risk of the predictors learned using $\mathcal{H}_N$ on a subset of the popular data set of handwritten digits called MNIST. The same figure also shows the ℓ_2 norm of the function coefficients, as well as the training risk. We see that for small values of N, the test risk shows the classical U-shaped curve consistent with the bias-variance trade-off, with a peak occurring at the interpolation threshold $N = n$. Some statistical analyses of RFF suggest choosing $N \propto \sqrt{n}\log n$ to obtain good test risk guarantees [32].

The interpolation regime connected with modern practice is shown to the right of the interpolation threshold, with $N \geq n$. The model class that achieves interpolation with fewest parameters ($N = n$ random features) yields the least accurate predictor. (In fact, it has no predictive ability for classification.) But as the number of features increases beyond n, the accuracy improves dramatically, exceeding that of the predictor corresponding to the bottom of the U-shaped curve. The plot also shows that the predictor $h_{n,\infty}$ obtained from $\mathcal{H}_\infty$ (the kernel machine) out-performs the predictors from $\mathcal{H}_N$ for any finite N.

What structural mechanisms account for the double descent shape? When the number of features is much smaller then the sample size, $N \ll n$, classical statistical arguments imply that the training risk is close to the test risk. Thus, for small N, adding more features yields improvements in both the training and test risks. However, as the number of features approaches n (the interpolation

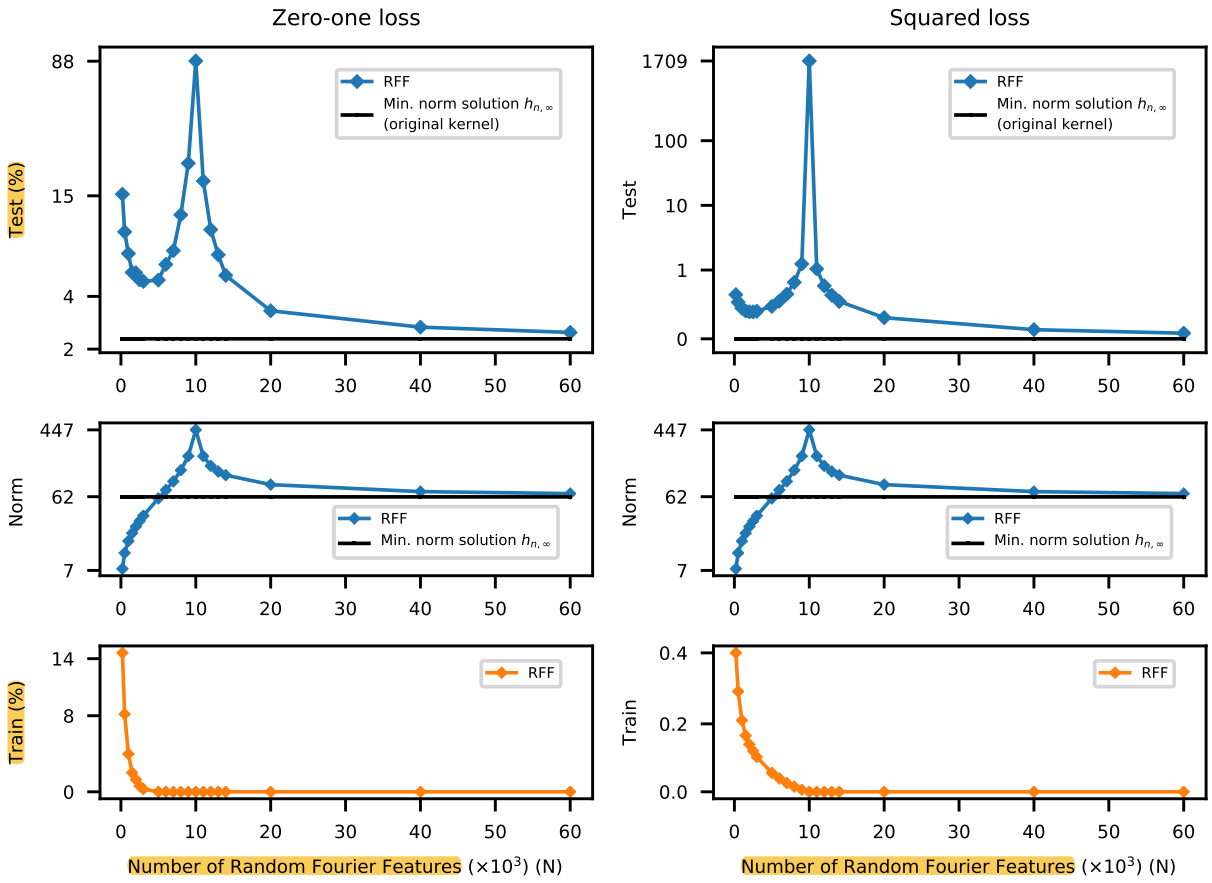

Figure 2: **Double descent risk curve for RFF model on MNIST.** Test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of the RFF model predictors $h_{n,N}$ learned on a subset of MNIST ($n = 10^4$, 10 classes). The interpolation threshold is achieved at $N = 10^4$.

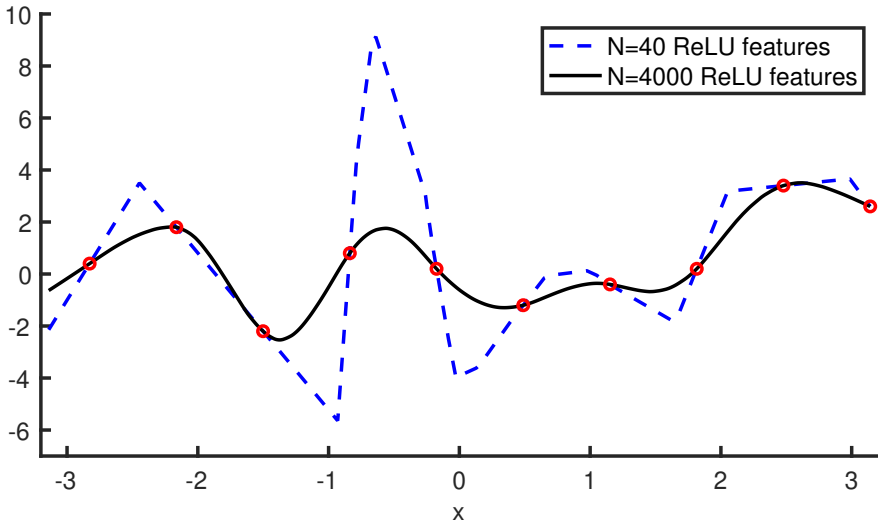

Figure 3: Plot of two univariate functions fitted to 10 data points using Random ReLU features $\phi(x;(v_1,v_2)) := \max(v_1 x + v_2, 0)$. The data points are shown in red circles. The fitted function with $N = 40$ Random ReLU features is the blue dashed line; the coefficient vector's norm (scaled by $\sqrt{N}$) is ≈ 695. The fitted function with $N = 4000$ Random ReLU features is the black solid line; the coefficient vector's norm is ≈ 159.

threshold), features not present or only weakly present in the data are forced to fit the training data nearly perfectly. This results in classical over-fitting as predicted by the bias-variance trade-off and prominently manifested at the peak of the curve, where the fit becomes exact.

To the right of the interpolation threshold, all function classes are rich enough to achieve zero training risk. For the classes $\mathcal{H}_N$ that we consider, there is no guarantee that the most regular, smallest norm predictor consistent with training data (namely $h_{n,\infty}$, which is in $\mathcal{H}_\infty$) is contained in the class $\mathcal{H}_N$ for any finite N. But increasing N allows us to construct progressively better approximations to that smallest norm function. Thus we expect to have learned predictors with largest norm at the interpolation threshold and for the norm of $h_{n,N}$ to decrease monotonically as N increases thus explaining the second descent segment of the curve. This is what we observe in Figure 2, and indeed $h_{n,\infty}$ has better accuracy than all $h_{n,N}$ for any finite N. Favoring small norm interpolating predictors turns out to be a powerful inductive bias on MNIST and other real and synthetic data sets [4]. For noiseless data, we make this claim mathematically precise in Appendix A.

Additional empirical evidence for the same double descent behavior using other data sets is presented in Appendix C.1. For instance, we demonstrate double descent for rectified linear unit (ReLU) random feature models, a class of ReLU neural networks with a setting similar to that of RFF. The inductive bias corresponding to the larger number of features can be readily observed in a one-dimensional example in Figure 3. Although the fitted function is non-smooth (piecewise linear) for any number of Random ReLU features, it appears smoother—with smaller norm—as the number of features is increased.

Finally, in Appendix C.4, we also describe a simple synthetic model, which can be regarded as a one-dimensional version of the RFF model, where we observe the same double descent behavior.

Neural networks and backpropagation. In general multilayer neural networks (beyond RFF or ReLU random feature models), a learning algorithm will tune all of the weights to fit the training

data, typically using versions of stochastic gradient descent (SGD), with backpropagation to compute partial derivatives. This flexibility increases the representational power of neural networks, but also makes ERM generally more difficult to implement. Nevertheless, as shown in Figure 4, we observe that increasing the number of parameters in fully connected two-layer neural networks leads to a risk curve qualitatively similar to that observed with RFF models. That the test risk improves beyond the interpolation threshold is compatible with the conjectured "small norm" inductive biases of the common training algorithms for neural networks [20, 25]. We note that this transition from under- to over-parameterized regimes for neural networks was also previously observed by [7, 1, 27, 37]. In particular, [37] draws a connection to the physical phenomenon of "jamming" in particle systems.

The computational complexity of ERM with neural networks makes the double descent risk curve difficult to observe. Indeed, in the classical under-parametrized regime ($N \ll n$), the non-convexity of the ERM optimization problem causes the behavior of local search-based heuristics, like SGD, to be highly sensitive to their initialization. Thus, if only suboptimal solutions are found for the ERM optimization problems, increasing the size of a neural network architecture may not always lead to a corresponding decrease in the training risk. This suboptimal behavior can lead to high variability in both the training and test risks that masks the double descent curve.

It is common to use neural networks with extremely large number of parameters [11]. But to achieve interpolation for a single output (regression or two class classification) one expects to need at least as many parameters as there are data points. Moreover, if the prediction problem has more than one output (as in multi-class classification), then the number of parameters needed should be multiplied by the number of outputs. This is indeed the case empirically for neural networks shown in Figure 4. Thus, for instance, data sets as large as ImageNet [33], which has $\sim 10^6$ examples and $\sim 10^3$ classes, may require networks with $\sim 10^9$ parameters to achieve interpolation; this is larger than many neural network models for ImageNet [11]. In such cases, the classical regime of the U-shaped risk curve is more appropriate to understand generalization. For smaller data sets, these large neural networks would be firmly in the over-parametrized regime, and simply training to obtain zero training risk often results in good test performance [42].

Additional results with neural networks are given in Appendix C.3.

3 Decision trees and ensemble methods

Does the double descent risk curve manifest with other prediction methods besides neural networks? We give empirical evidence that the families of functions explored by boosting with decision trees and Random Forests also show similar generalization behavior as neural nets, both before and after the interpolation threshold.

AdaBoost and Random Forests have recently been investigated in the interpolation regime by [41] for classification. In particular, they give empirical evidence that, when AdaBoost and Random Forests are used with maximally large (interpolating) decision trees, the flexibility of the fitting methods yield interpolating predictors that are more robust to noise in the training data than the predictors produced by rigid, non-interpolating methods (e.g., AdaBoost or Random Forests with shallow trees). This in turn is said to yield better generalization. The averaging of the (near) interpolating trees ensures that the resulting function is substantially smoother than any individual tree, which aligns with an inductive bias that is compatible with many real world problems.

We can understand these flexible fitting methods in the context of the double descent risk curve. Observe that the size of a decision tree (controlled by the number of leaves) is a natural way to parametrize the function class capacity: trees with only two leaves correspond to two-piecewise

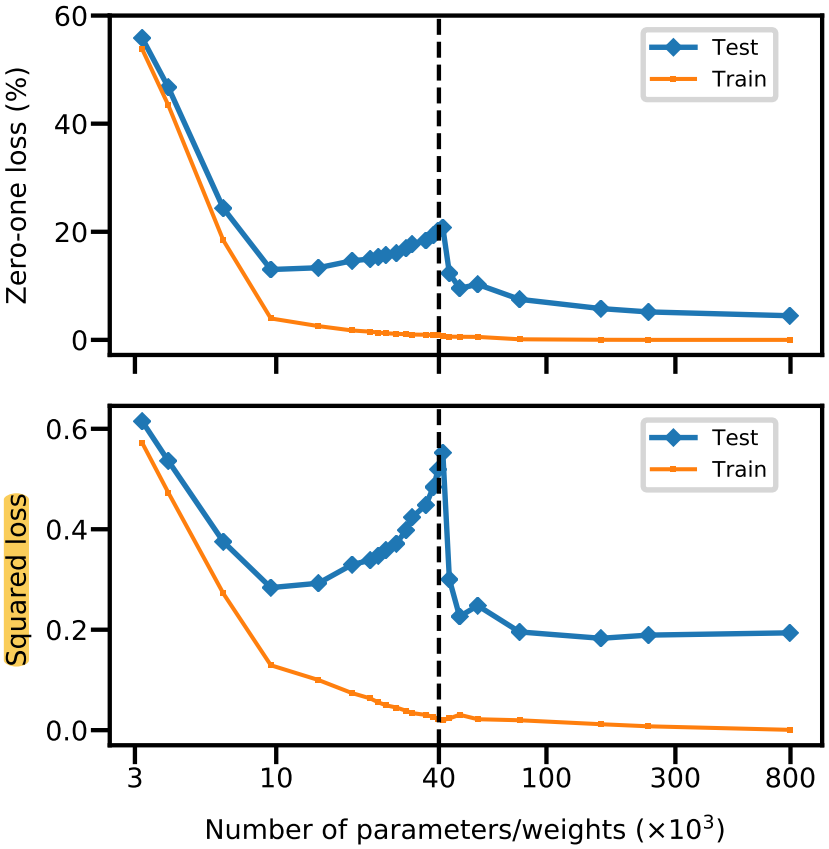

Figure 4: **Double descent risk curve for fully connected neural network on MNIST.** Training and test risks of network with a single layer of H hidden units, learned on a subset of MNIST ($n = 4 \cdot 10^3$, $d = 784$, $K = 10$ classes). The number of parameters is $(d+1) \cdot H + (H+1) \cdot K$. The interpolation threshold (black dotted line) is observed at $n \cdot K$.

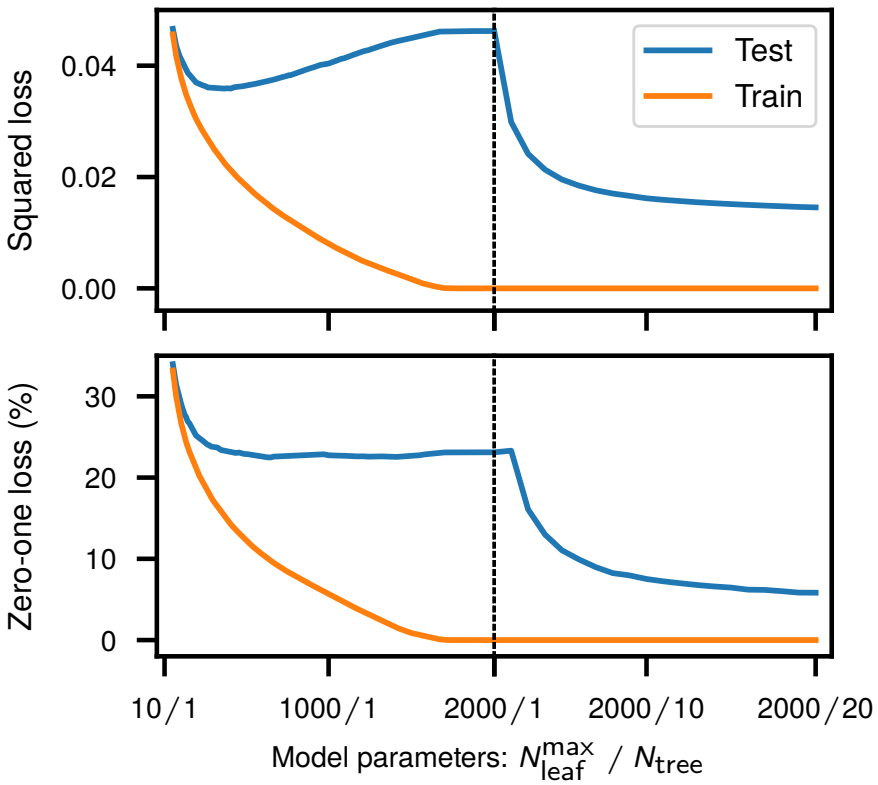

Figure 5: **Double descent risk curve for random forests on MNIST.** The double descent risk curve is observed for random forests with increasing model complexity trained on a subset of MNIST ($n = 10^4, 10$ classes). Its complexity is controlled by the number of trees N_{tree} and the maximum number of leaves allowed for each tree $N_{\text{leaf}}^{\text{max}}$.

constant functions with axis-aligned boundary, while trees with n leaves can interpolate n training examples. It is a classical observation that the U-shaped bias-variance trade-off curve manifests in many problems when the class capacity is considered this way [21]. (The interpolation threshold may be reached with fewer than n leaves in many cases, but n is clearly an upper bound.) To further enlarge the function class, we consider ensembles (averages) of several interpolating trees.[1] So, beyond the interpolation threshold, we use the number of such trees to index the class capacity. When we view the risk curve as a function of class capacity defined in this hybrid fashion, we see the double descent curve appear just as with neural networks; see Figure 5 and Appendix D. We observe a similar phenomenon using L_2-boosting [15, 10], another popular ensemble method; the results are reported in Appendix E.

4 Concluding thoughts

The double descent risk curve introduced in this paper reconciles the U-shaped curve predicted by the bias-variance trade-off and the observed behavior of rich models used in modern machine learning practice. The posited mechanism that underlies its emergence is based on common inductive biases, and hence can explain its appearance (and, we argue, ubiquity) in machine learning applications.

We conclude with some final remarks.

Historical absence. The double descent behavior may have been historically overlooked on account of several cultural and practical barriers. Observing the double descent curve requires a parametric family of spaces with functions of arbitrary complexity. The linear settings studied extensively in classical statistics usually assume a small, fixed set of features and hence fixed fitting capacity. Richer families of function classes are typically used in the context of non-parametric statistics, where smoothing and regularization are almost always employed [39]. Regularization, of all forms, can both prevent interpolation and change the effective capacity of the function class, thus attenuating or masking the interpolation peak.

The RFF model is a popular and flexible parametric family. However, these models were originally proposed as computationally favorable alternative to kernel machines. This computational advantage over traditional kernel methods holds only for $N \ll n$, and hence models at or beyond the interpolation threshold are typically not considered.

The situation with general multilayer neural networks, is slightly different and more involved. Due to the non-convexity of the ERM optimization problem, solutions in the classical under-parametrized regime are highly sensitive to initialization. Moreover, as we have seen, the peak at the interpolation threshold is observed within a narrow range of parameters. Sampling of the parameter space that misses that range may lead to the misleading impression that increasing the size of the network simply improves performance. Finally, in practice, training of neural networks is typically stopped as soon as (an estimate of) the test risk fails to improve. This *early stopping* has a strong regularizing effect that, as discussed above, makes it difficult to observe the interpolation peak.

Inductive bias. In this paper, we have dealt with several types of methods for choosing interpolating solutions. For Random Fourier and Random ReLU features, solutions are constructed explicitly by minimum norm linear regression in the feature space. As the number of features tends

[1] These trees are trained in the way proposed in Random Forest except without bootstrap re-sampling. This is similar to the PERT method of [14].

to infinity they approach the minimum functional norm solution in the Reproducing Kernel Hilbert Space, a solution which maximizes functional smoothness subject to the interpolation constraints. For neural networks, the inductive bias owes to the specific training procedure used, which is typically SGD. When all but the final layer of the network are fixed (as in RFF models), SGD initialized at zero also converges to the minimum norm solution. While the behavior of SGD for more general neural networks is not fully understood, there is significant empirical and some theoretical evidence (e.g., [20]) that a similar minimum norm inductive bias is present. Yet another type of inductive bias related to averaging is used in random forests. Averaging potentially non-smooth interpolating trees leads to an interpolating solution with a higher degree of smoothness; this averaged solution performs better than any individual interpolating tree.

Remarkably, for kernel machines all three methods lead to the same minimum norm solution. Indeed, the minimum norm interpolating classifier, $h_{n,\infty}$, can be obtained directly by explicit norm minimization (solving an explicit system of linear equations), through SGD or by averaging trajectories of Gaussian processes (computing the posterior mean [31]).

Optimization and practical considerations. In our experiments, appropriately chosen "modern" models usually outperform the optimal "classical" model on the test set. But another important practical advantage of over-parametrized models is in optimization. There is a growing understanding that larger models are "easy" to optimize as local methods, such as SGD, converge to global minima of the training risk in over-parametrized regimes (e.g., [36]). Thus, large interpolating models can have low test risk and be easy to optimize at the same time, in particular with SGD [26]. It is likely that the models to the left of the interpolation peak have optimization properties qualitatively different from those to the right, a distinction of significant practical import.

Outlook. The classical U-shaped bias-variance trade-off curve has shaped our view of model selection and directed applications of learning algorithms in practice. The understanding of model performance developed in this work delineates the limits of classical analyses and opens new lines of enquiry to study and compare computational, statistical, and mathematical properties of the classical and modern regimes in machine learning. We hope that this perspective, in turn, will help practitioners choose models and algorithms for optimal performance.

Acknowledgments

We thank Peter Bickel for editing the PNAS submission, and the anonymous reviewers for their helpful feedback. Mikhail Belkin, Siyuan Ma and Soumik Mandal were supported by NSF RI-1815697. Daniel Hsu was supported by NSF CCF-1740833 and Sloan Research Fellowship. We thank Nvidia for donating GPUs used for this research.

References

[1] Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

[2] Peter L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

[3] Mikhail Belkin, Daniel Hsu, and Partha Mitra. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In *Advances in Neural Information Processing Systems*, pages 2306–2317, 2018.

[4] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 541–549, 2018.

[5] Mikhail Belkin, Alexander Rakhlin, and Alexandre B Tsybakov. https://arxiv.org/abs/1806.09471, 2018.

[6] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's razor. *Information processing letters*, 24(6):377–380, 1987.

[7] Siegfried Bös and Manfred Opper. Dynamics of training. In *Advances in Neural Information Processing Systems*, pages 141–147, 1997.

[8] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[9] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] Peter Bühlmann and Bin Yu. Boosting with the l_2 loss: regression and classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

[11] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[12] Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pages 342–350, 2009.

[13] François Chollet et al. Keras. https://keras.io, 2015.

[14] Adele Cutler and Guohua Zhao. Pert-perfect random tree ensembles. *Computing Science and Statistics*, 33:490–497, 2001.

[15] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[16] John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. *NIST speech disc*, 1-1.1, 1993.

[17] Stuart Geman, Elie Bienenstock, and Ren Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992. doi: 10.1162/neco.1992.4.1.1. URL `https://doi.org/10.1162/neco.1992.4.1.1`.

[18] Gerd Gigerenzer and Henry Brighton. Homo heuristicus: Why biased minds make better inferences. *Topics in cognitive science*, 1(1):107–143, 2009.

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[20] Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pages 6151–6159, 2017.

[21] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, volume 1. Springer, 2001.

[22] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[23] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings*, pages 331–339. Elsevier, 1995.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[25] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 2–47, 06–09 Jul 2018.

[26] Siyuan Ma, Raef Bassily, and Mikhail Belkin. The power of interpolation: Understanding the effectiveness of SGD in modern over-parametrized learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3325–3334, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/ma18a.html`.

[27] Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018.

[28] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop*, volume 2011, page 4, 2011.

[29] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532–1543, 2014.

[30] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008.

[31] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

[32] Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, pages 3215–3225, 2017.

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[34] Ruslan Salakhutdinov. Deep learning tutorial at the Simons Institute, Berkeley, https://simons.berkeley.edu/talks/ruslan-salakhutdinov-01-26-2017-1, 2017.

[35] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Statist.*, 26(5):1651–1686, 1998.

[36] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 2018.

[37] Stefano Spigler, Mario Geiger, Stéphane d'Ascoli, Levent Sagun, Giulio Biroli, and Matthieu Wyart. A jamming transition from under-to over-parametrization affects loss landscape and generalization. *arXiv preprint arXiv:1810.09665*, 2018.

[38] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995. ISBN 0-387-94559-8.

[39] Larry Wasserman. *All of Nonparametric Statistics*. Springer, 2006.

[40] Holger Wendland. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004. doi: 10.1017/CBO9780511617539.

[41] Abraham J Wyner, Matthew Olson, Justin Bleich, and David Mease. Explaining the success of adaboost and random forests as interpolating classifiers. *Journal of Machine Learning Research*, 18(48):1–33, 2017.

[42] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

A Approximation theorem

Suppose the training data $(x_1, y_1), \ldots, (x_n, y_n)$ are sampled independently by drawing x_i uniformly from a compact domain in $\mathbb{R}^d$, and assigning the label $y_i = h^*(x_i)$ using a target function $h^* \in \mathcal{H}_\infty$. Let $h \in \mathcal{H}_\infty$ be another hypothesis that interpolates the training data $(x_1, y_1), \ldots, (x_n, y_n)$. The following theorem bounds the error of h in approximating h^*.

Theorem 1. *Fix any $h^* \in \mathcal{H}_\infty$. Let $(x_1, y_1), \ldots, (x_n, y_n)$ be independent and identically distributed random variables, where x_i is drawn uniformly at random from a compact cube[2] $\Omega \subset \mathbb{R}^d$, and $y_i = h^*(x_i)$ for all i. There exists absolute constants $A, B > 0$ such that, for any interpolating $h \in \mathcal{H}_\infty$ (i.e., $h(x_i) = y_i$ for all i), so that with high probability*

$$\sup_{x \in \Omega} |h(x) - h^*(x)| < A e^{-B(n/\log n)^{1/d}} \left(\|h^*\|_{\mathcal{H}_\infty} + \|h\|_{\mathcal{H}_\infty} \right).$$

Proof sketch. Recall that *the fill* κ_n of the set of points $x_1, \ldots, x_n$ in Ω is a measure of how well these points cover Ω: $\kappa_n = \max_{x \in \Omega} \min_{x_j \in \{x_1, \ldots, x_n\}} \|x - x_j\|$. It is easy to verify (e.g., by taking an

[2] Same argument can be used for more general domains and probability distributions.

appropriate grid partition of the cube Ω and applying the union bound) that with high probability $\kappa_n = O(n/\log n)^{-1/d}$.

Consider now a function $f(x) := h(x) - h^*(x)$. We observe that $f(x_i) = 0$ and by the triangle inequality $\|f\|_{\mathcal{H}_\infty} \leq \|h^*\|_{\mathcal{H}_\infty} + \|h\|_{\mathcal{H}_\infty}$. Applying Theorem 11.22 in [40] to f yields the result. □

The minimum norm interpolating function $h_{n,\infty}$ has norm no larger than that of h^* (by definition) and hence achieves the smallest bound in Theorem 1. While these bounds apply only in the noiseless setting, they provide a justification for the inductive bias based on choosing a solution with a small norm. Indeed, there is significant empirical evidence that minimum norm interpolating solutions generalize well on a variety of datasets, even in the presence of large amounts of label noise [4].

B Experimental setup

To demonstrate the double descent risk curve, we train a number of representative models including neural networks, kernel machines and ensemble methods on several widely used datasets that involve images, speech, and text.

Datasets. Table 1 describes the datasets we use in our experiments. These datasets are for classification problems with more than two classes, so we adopt the one-versus-rest strategy that maps a multi-class label to a binary label vector (one-hot encoding). For the image datasets—namely MNIST [24], CIFAR-10 [22], and SVHN [28]—color images are first transformed to grayscale images, and then the maximum range of each feature is scaled to the interval $[0, 1]$. For the speech dataset TIMIT [16], we normalize each feature by its z-score. For the text dataset 20-Newsgroups [23], we transform each sparse feature vector (bag of words) into a dense feature vector by summing up its corresponding word embeddings obtained from [29].

For each dataset, we subsample a training set (of size n) uniformly at random without replacement. For the 20-Newsgroups dataset, which does not have a test set provided, we randomly pick 1/8 of the full dataset for use as a test set.

Model training. Each model is trained to minimize the squared loss on the given training set. Without regularization, such model is able to interpolate the training set when its capacity surpasses certain threshold (interpolation threshold). For comparison, we report the test/train risk for zero-one and squared loss. In experiments for neural networks and ensemble methods, we repeat the same experiment five times and report the mean for the risks. RFF and Random ReLU experimental results were reported based on a single run as the results were empirically highly consistent.

Table 1: Descriptions of datasets. In experiments, we use subsets to reduce the computational cost.

Dataset	Size of full training set	Feature dimension (d)	Number of classes
CIFAR-10	$5 \cdot 10^4$	1024	10
MNIST	$6 \cdot 10^4$	784	10
SVHN	$7.3 \cdot 10^4$	1024	10
TIMIT	$1.1 \cdot 10^6$	440	48
20-Newsgroups	$1.6 \cdot 10^4$	100	20

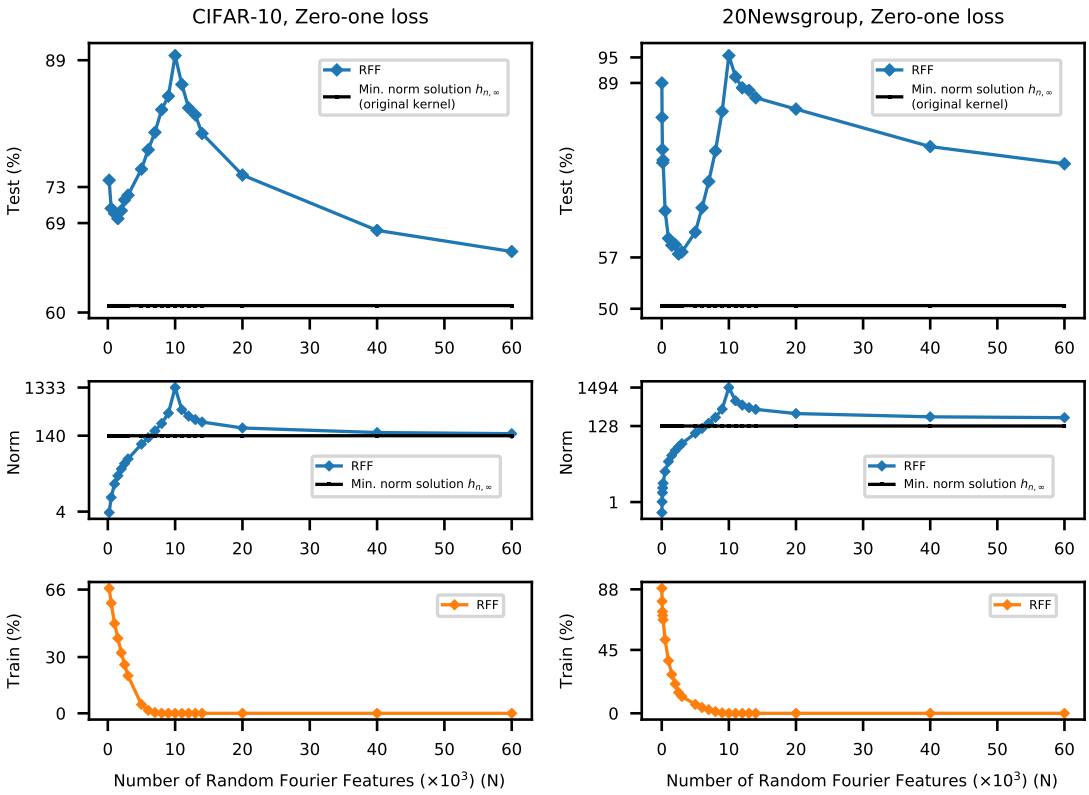

Figure 6: **Double descent risk curve for RFF model.** Test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of the RFF model predictors $h_{n,N}$ learned on subsets of CIFAR-10 and 20Newsgroups ($n = 10^4$). The interpolation threshold is achieved at $N = 10^4$.

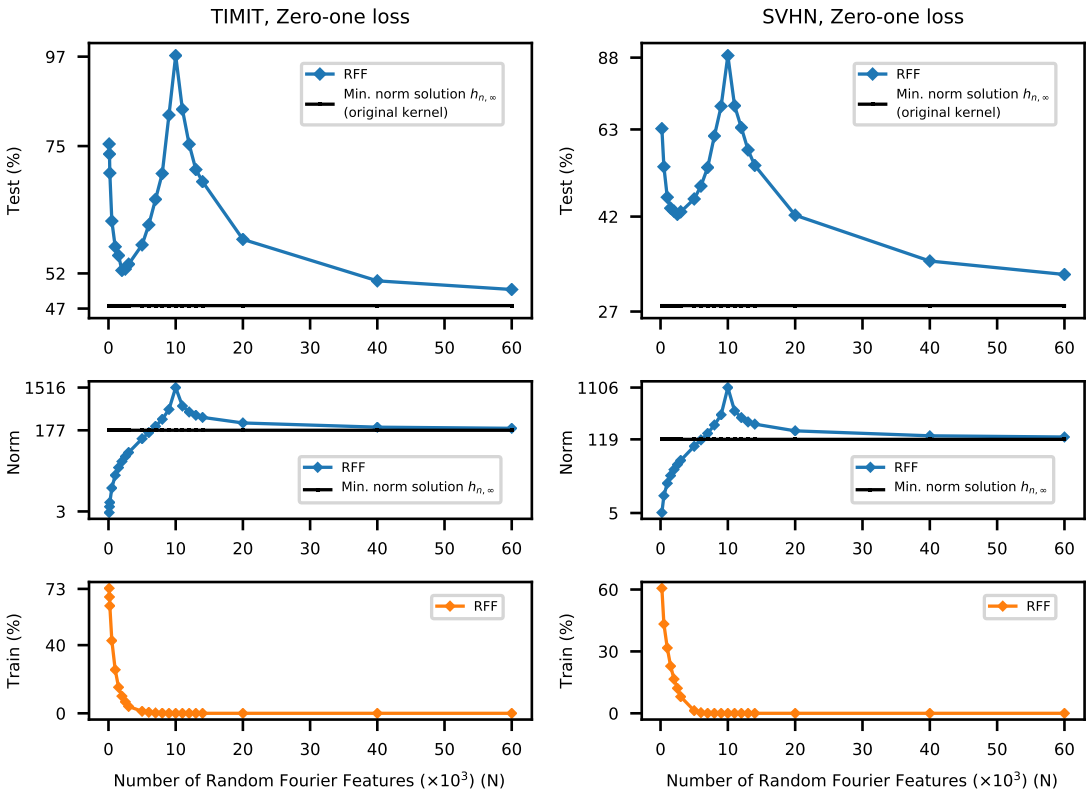

Figure 7: **Double descent risk curve for RFF model.** Test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of RFF model predictors $h_{n,N}$ learned on subsets of TIMIT and SVHN ($n = 10^4$). The interpolation threshold is achieved at $N = 10^4$.

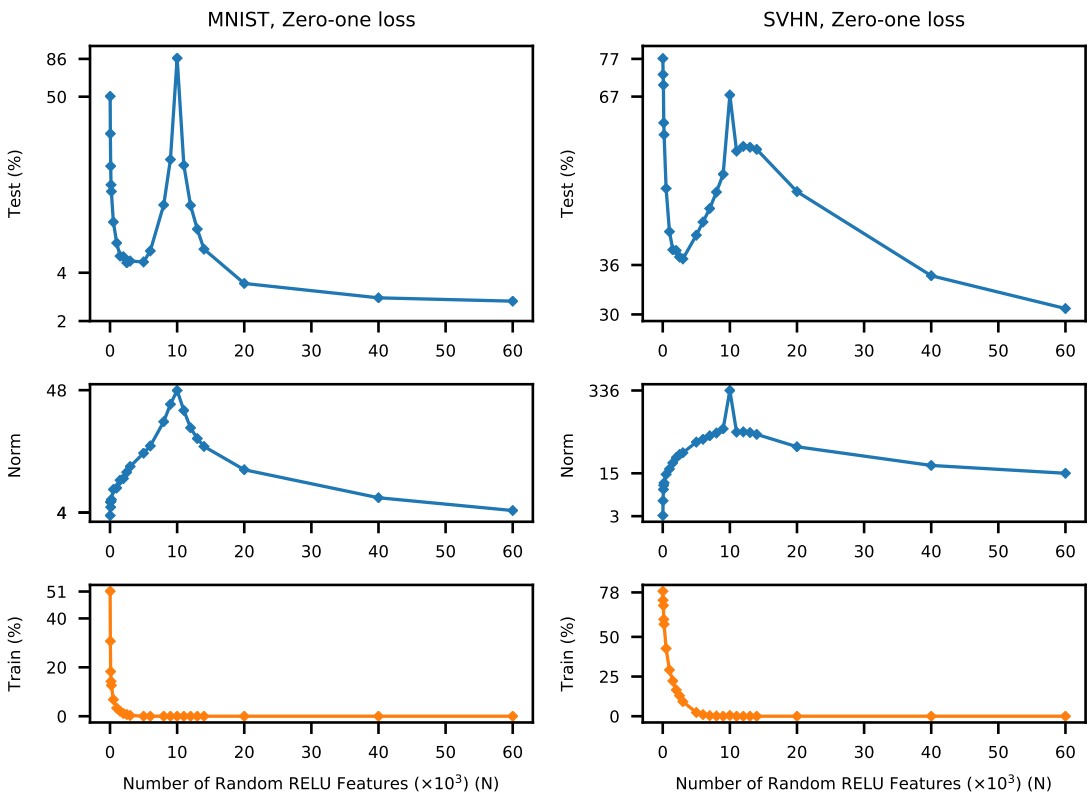

Figure 8: **Double descent risk curve for Random ReLU model.** Test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of the Random ReLU Features model predictors $h_{n,N}$ learned on subsets of MNIST and SVHN data ($n = 10^4$). The interpolation threshold is achieved at $N = 10^4$. Regularization of $4 \cdot 10^{-6}$ is added for SVHN to ensure numerical stability near interpolation threshold.

C Additional experimental results for neural networks

C.1 Random Fourier Feature models

We provide additional experimental results for several real-world datasets. Figure 6 illustrates double descent behavior for CIFAR-10 and 20Newsgroup. Figure 7 shows similar curves of zero-one loss for TIMIT and SVHN. The random feature vectors $v_1, \ldots, v_N$ are sampled independently from $\mathcal{N}(0, \sigma^{-2} \cdot I)$, the mean-zero normal distribution in $\mathbb{R}^d$ with covariance $\sigma^{-2} \cdot I$. The bandwidth parameter σ is set to 5, 5, 5, 0.1, and 16 for MNIST, SVHN, CIFAR-10, 20-Newsgroup, and TIMIT, respectively.

C.2 Random ReLU Feature models

We show that the double descent risk curve also appears with Random ReLU feature networks [12]. Such networks are similar to the RFF models, except that they use the ReLU transfer function. Specifically, the Random ReLU features model family $\mathcal{H}_N$ with N parameters consists of functions

$h\colon \mathbb{R}^d \to \mathbb{R}$ of the form

$$h(x) = \sum_{k=1}^{N} a_k \phi(x; v_k) \quad \text{where} \quad \phi(x; v) := \max(\langle v, x \rangle, 0).$$

The vectors $v_1, \ldots, v_N$ are sampled independently from uniform distribution over surface of unit sphere in $\mathbb{R}^d$. The coefficients a_k are learned using linear regression. Figure 8 illustrates zero-one loss with Random ReLU features for MNIST and SVHN data. Ridge regularization with parameter $\lambda := 4 \cdot 10^{-6}$ is added in SVHN experiments to ensure numerical stability near the interpolation threshold. For MNIST experiments, no regularization is added. We observe that the resulting risk curves and the norm curves are very similar to those for RFF.

C.3 Fully connected neural networks

In our experiments, we use fully connected neural networks with a single hidden layer. To control the capacity of function class, we vary the number of hidden units. We use stochastic gradient descent (SGD) to solve the ERM optimization problem in this setting.

The ERM optimization problem in this setting is generally more difficult than that for RFF and ReLU feature models due to a lack of analytical solutions and non-convexity of the problem. Consequently, SGD is known to be sensitive to initialization. To mitigate this sensitivity, we use a "weight reuse" scheme with SGD in the under-parametrized regime ($N < n$), where the parameters obtained from training a smaller neural network are used as initialization for training larger networks. This procedure, detailed below, ensures decreasing training risk as the number of parameters increases. In the over-parametrized regime ($N \geq n$), we use standard (random) initialization, as typically there is no difficulty in obtaining near-zero training risk.

Additional experimental results for neural networks are shown in Figure 9. Results for MNIST and CIFAR-10 with weight reuse are reported in Figure 9(a) and Figure 9(b). Results for MNIST without weight reuse are reported in Figure 9(c). In this setting, all models are randomly initialized. While the variance is significantly larger, and the training loss is not monotonically decreasing, the double descent behavior is still clearly discernible.

We now provide specific details below. We use SGD with standard momentum (parameter value 0.95) implemented in [13] for training. In the weight reuse scheme, we assume that we have already trained a smaller network with H_1 hidden units. To train a larger network with $H_2 > H_1$ hidden units, we initialize the first H_1 hidden units of the larger network to the weights learned in the smaller network. The remaining weights are initialized with normally distributed random numbers (mean 0 and variance 0.01). The smallest network is initialized using standard Glorot-uniform distribution [19]. For networks smaller than the interpolation threshold, we decay the step size by 10% after each of 500 epochs, where an epoch denotes a pass through the training data. For these networks, training is stopped after classification error reached zero or 6000 epochs, whichever happens earlier. For networks larger than interpolation threshold, fixed step size is used, and training is stopped after 6000 epochs.

C.4 Synthetic model

We now discuss the nature of the double descent risk curve in the context of a simple synthetic model, which can be viewed as a version of RFF for functions on the one-dimensional circle. Consider the class $\mathcal{H}$ of periodic complex-valued functions on the interval $[0, 2\pi]$, and let

$$e_k(x) := \exp(\sqrt{-1}(k-1)x)$$

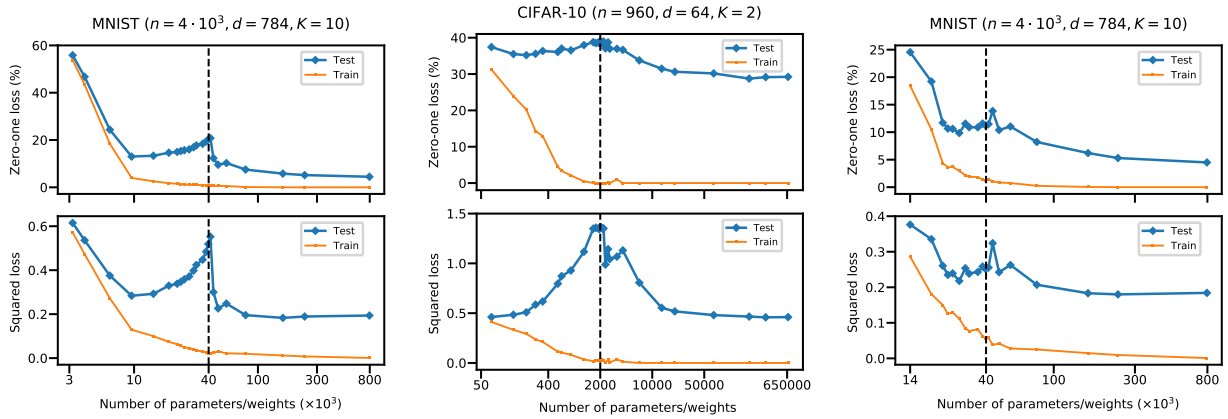

Figure 9: **Double descent risk curve for fully connected neural networks.** In each plot, we use a dataset with n subsamples of d dimension and K classes for training. We use networks with a single hidden layer. For network with H hidden units, its number of parameters is $(d+1) \cdot H + (H+1) \cdot K$. The interpolation threshold is observed at $n \cdot K$ and is marked by black dotted line in figures. (**a**) Weight reuse before interpolation threshold and random initialization after it on MNIST. (**b**) Same, on a subset of CIFAR-10 with 2 classes (cat, dog) and downsampled image features (8×8). (**c**) No weight reuse (random initialization for all ranges of parameters).

for positive integers k. Fix a probability distribution $p = (p_1, p_2, \ldots)$ on the positive integers. For each integer N, we generate a random function class $\mathcal{H}_N$ by (i) sampling independently from p until N distinct indices $k_1, \ldots, k_N$ are chosen, and then (ii) let $\mathcal{H}_N$ be the linear span of $e_{k_1}, \ldots, e_{k_N}$. Here, N is the number of parameters to specify a function in $\mathcal{H}_N$ and also reflects the capacity of $\mathcal{H}_N$.

We generate data from the following model:

$$y_i = h^*(x_i) + \varepsilon_i$$

where the target function $h^* = \sum_k \alpha_k^* e_k$ is in the span of the e_k, and $\varepsilon_1, \ldots, \varepsilon_n$ are independent zero-mean normal random variables with variance σ^2. The $x_1, \ldots, x_n$ themselves are drawn uniformly at random from $\{2\pi j/M : j = 0, \ldots, M-1\}$ for $M := 4096$. We also let $\alpha_k^* := p_k$ for all k, with $p_k \propto 1/k^2$. The signal-to-noise ratio (SNR) is $\mathbb{E}[h^*(x_i)^2]/\sigma^2$.

Given data $(x_1, y_1), \ldots, (x_n, y_n) \in [0, 2\pi] \times \mathbb{R}$, we learn a function from the function class $\mathcal{H}_N$ using empirical risk minimization, which is equivalent to ordinary least squares over an N-dimensional space. Interpolation is achieved when $N \geq n$, so in this regime, we choose the interpolating function $h = \sum_{j=1}^N \alpha_{k_j} e_{k_j}$ of smallest (squared) norm $\|h\|_{\mathcal{H}}^2 = \sum_k \alpha_k^2 / p_k$.

Our simulations were carried out for a variety of sample sizes ($n \in \{2^6, 2^7, \ldots, 2^{11}\}$) and are all repeated independently 20 times; our plots show averages over the 20 trials. The results confirm our hypothesized double descent risk curve, as shown in Figure 10 for $n = 256$; the results are similar for other n. The peak occurs at $N = n$, and the right endpoint of the curve is lower than the bottom of the U curve. The norm of the learned function also peaks at $N = n$ and decreases for $N > n$.

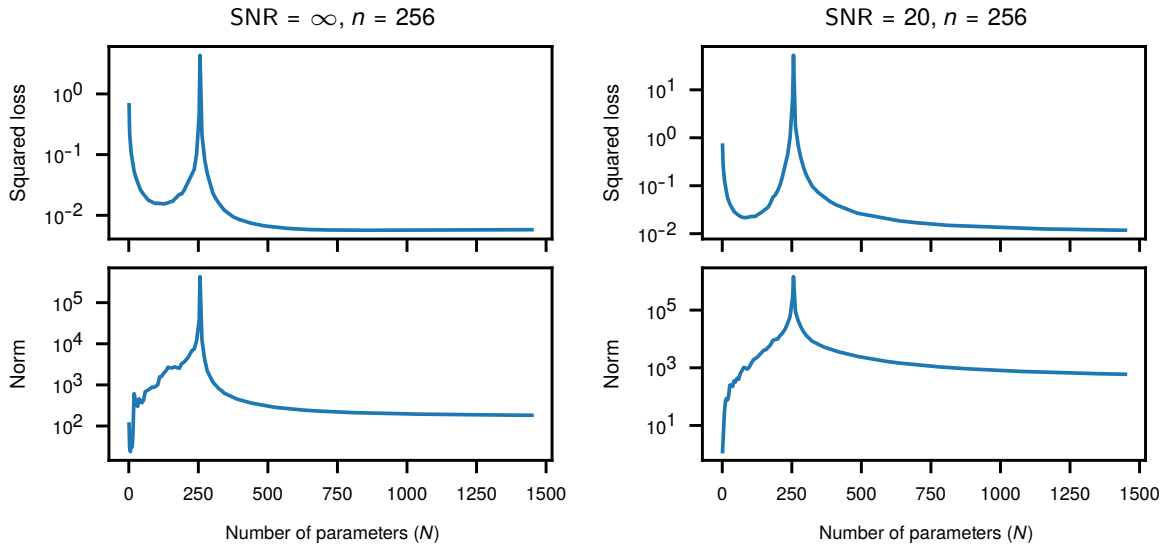

Figure 10: **Results from the synthetic model at SNR $= \infty$ and SNR $= 20$.** Top: excess test risk under squared loss of learned function. Bottom: norm of learned function $\|h\|_{\mathcal{H}_\infty}$. For $n = 256$ training samples, the interpolation threshold is reached at $N = 256$.

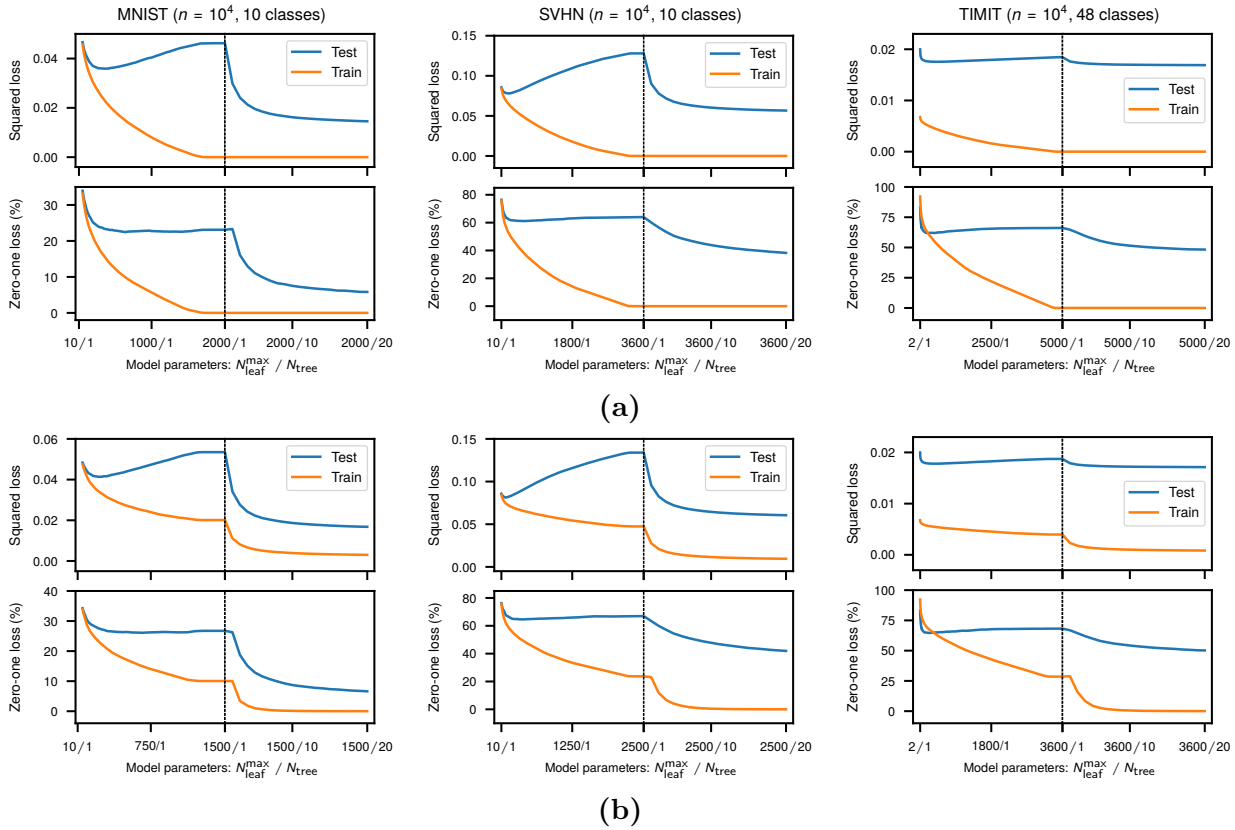

Figure 11: **Double descent risk curve for random forests.** In all plots, the double descent risk curve is observed for random forest with increasing model complexity on regression tasks. Its complexity is controlled by the number of trees N_{tree} and the maximum number of leaves allowed for each tree $N_{\mathsf{leaf}}^{\mathsf{max}}$. (**a**) Without bootstrap re-sampling, a single tree can interpolate the training data. (**b**) With bootstrap re-sampling, multiple trees are needed to interpolate the data.

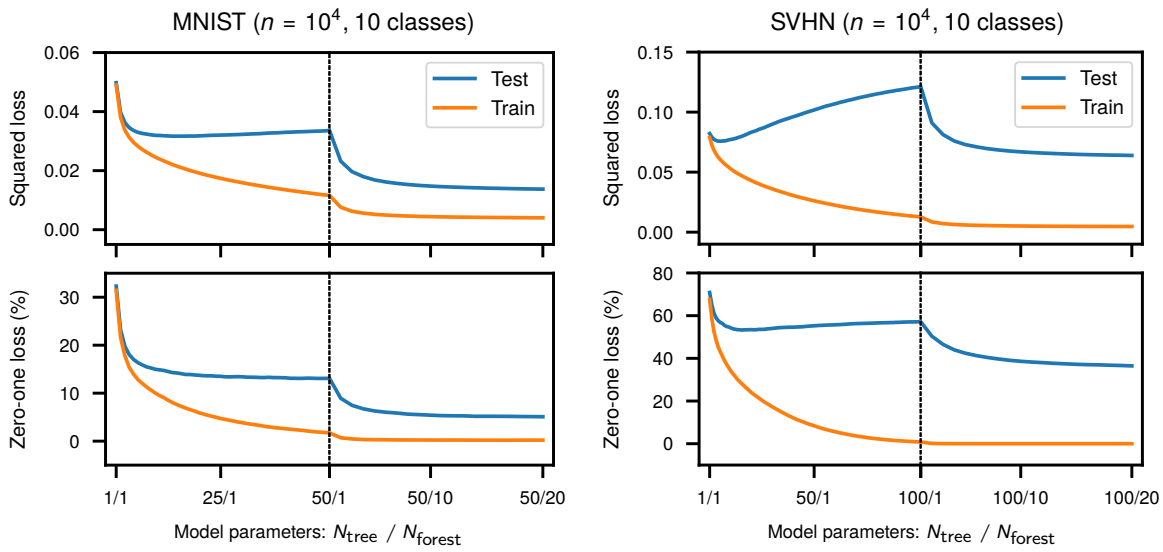

Figure 12: **Double descent risk curve for L_2-boosting trees.** In both plots, we increase the model complexity by first increasing the number of boosting (random) trees (N_{tree}) which form a forest, then averaging several such forests (N_{forest}). Each tree is constrained to have no more than 10 leaves. For fast interpolation, the gradient boosting is applied with low shrinkage parameter (0.85).

D Additional results with Random Forests

We train standard random forests introduced by Breiman [9] for regression problems. When splitting a node, we randomly select a subset of features whose number is the square root of the number of the total features, a setting which is widely used in mainstream implementations of random forest. We control the capacity of the model class by choosing the number of trees (N_{tree}) and limiting the maximum number of leaves in each tree ($N_{\text{leaf}}^{\max}$). We put minimum constraints on the growth of each tree: there is no limit for the tree depth and we split each tree node whenever it is possible.

To interpolate the training data, we disable the bootstrap re-sampling for results in Figure 11(a), which has been investigated under the name "Perfect random tree ensembles" by Cutler et al. [14]. We see clear double decent risk curve (with both squared loss and zero-one loss) as we increase the capacity of the model class (although the U-shaped curve is less apparent with zero-one loss). In Figure 11(b), we run the same experiments with bootstrap re-sampling enabled, which show similar double decent risk curves.

E Results with L_2-boosting

We now show double descent risk curve for L_2-boosting (random) trees introduced by Friedman [15]. When splitting a node in a tree, we randomly select a subset of features whose number is the square root of the number of the total features. We constrain each tree to have a small number of leaves (no more than 10). As the number of trees increases, the boosted trees gradually interpolate the training data and form a forest. To quickly reach interpolation, we adopt low shrinkage (parameter value 0.85) for gradient boosting. To go beyond the interpolation threshold, we average the predictions of several such forests which are randomly constructed and trained with exactly same

hyper-parameters. The capacity of our model is hence controlled by the number of forests (N_{forest}) and the number of trees (N_{tree}) in each forest.

Figure 12 shows the change of train and test risk as the model capacity increases. We see the double descent risk curve for both squared loss and zero-one loss. We also observe strong overfitting under squared loss before the interpolation threshold. For similar experiments with high shrinkage (parameter value 0.1), the double descent risk curve becomes less apparent due to the regularization effect of high shrinkage [21].