Random Matrix Theory and Generalization in Neural Networks

Tanishq Kumar

18.338 Final Project Report

Keywords: neural network generalization, Marchenko-Pasteur, eigengap, frozen subspace, thermodynamic limit, Rademacher complexity, generalization bounds, training error dynamics, high-dimensional kernel regression, random data and covariance matrices, double descent.

1 Introduction and Overview

The goal of this project is to survey and summarize, in detail, some works in modern machine learning theory that exploit tools from random matrix theory. This project aims to be a summary and analysis of two key papers in this intersection. RMT has become a useful tool in machine learning theory for a few reasons: the weights and activations of a neural network before training are random matrices, the data matrix can be modelled as a large random matrix, which then universality guarantees is similar in spectral structure to data matrices found in practice. Covariance matrices often completely characterize learning curves, and we have spectral guarantees as they get bigger. Ideas from RMT in these works are mostly Marcenko-Pasteur and replica theory, but other papers make use of R/S Transforms and free probability. Here are short summaries of the two papers we'll analyse carefully, both very important in the field. The first studies a linear regression task (under gradient flow dynamics) and the second a kernel regression task (which is equivalent to wide neural networks).

1. *High-dimensional dynamics of generalization error in neural networks* Let P, N denote the number of data points and parameters in a network model, respectively. This paper exactly characterizes the generalization error in the limit $P/N = \alpha$ with $P, N \to \infty$ for some constant $\alpha \in \mathbb{R}$ in a variety of network architectures, and uses these conclusions to reason about optimal stopping. In particular, it finds that the eigenvalue distribution of the input correlation matrix has a strong effect on generalization, explaining that $\alpha = 1$ gives a high eigenvalue density at zero and relating this to the tendency of models with $P \approx N$ to fit, and also explaining why both highly under-parameterized (classical statistics) and over-parameterized (modern deep learning) models do better in terms of this eigenvalue distribution. This is a classic paper using ideas from spectral gap and the Marcenko-Pasteur distribution to make conclusions about generalization of neural networks and optimal early stopping and weight regularization.

2. *Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks.* This paper finds a closed-form expression for the generalization error of kernel regression on a dataset (X, y) where $y_i = f^*(x_i)$ deterministically, and $x_i \sim p(x)$ for some distribution p. They find a formally for E^g the test risk averaged over all data sets X, possible target functions f^*, and decompose this into risk induced by each spectral mode of kernel in question. That is, the final E^g can be written as a sum over E_i where i indexes the eigenvalues of the kernel K being used for regression. This is relevant because wide neural networks approximately do kernel regression with their neural tangent kernel (NTK), so these generalization results carry over to generalization results for many types of neural networks as a function of the spectrum of the NTK, which is an object under study in ML theory.

2 Papers

2.1 High-Dimensional Dynamics of Generalization Error

2.1.1 Problem Setup

Here, test risk dynamics are derived for deep linear networks using RMT tools, and insights into implicit regularization result. Of course, the end-time test risk in a linear network is no better than with linear regression, for which closed forms have been known for a long time. The contribution of this paper is studying the *dynamics* of the network's loss as it converges to this test risk over the course of training, and in particular in the thermodynamic limit $P/D \to \infty$ for P data points each of dimension D. These results are then extended to the case of deep linear networks, then to nonlinear networks, but the key is that even the dynamics of the shallow linear network exhibit puzzling properties we see in deep learning – such as double descent, for instance – and so is the focus of our review. Finally, the authors use insights on weight dynamics for such MLPs to give tighter Rademacher complexity-based generalization bounds that explain puzzling generalization results in deep learning.

2.1.2 Main Result

We work in a common setting in statistical mechanics analyses of neural networks: a student-teacher setting where the teacher is of the form $y = \bar{w}X + \epsilon$ for ground truth $\bar{w}$ and noise ϵ. Our dataset is $X \in \mathbb{R}^{N\times P}$ where we have P data points, each in N dimensions. We draw each entry of the data matrix from $x \sim N(0, 1/N)$ so that each data point, which is a row, has unit norm in expectation. We take the thermodynamic limit $P, N \to \infty$ with $P/D = \gamma \ll \infty$ fixed, so this becomes rows having unit norm almost surely due to concentration of measure. The student is also a one-layer network has the form $\hat{y} = wX$ where w is what is learned. The student weights are initialized with variance σ_w^2 and noise with variance σ_ϵ^2. Of course, this is just a very simple linear regression task, and the optimal solution is well known to be of the form $w_\infty = yX^T(XX^T)$. The key is that we are solving for the *dynamics*, in a way that will elucidate the inductive biases of gradient-training of networks when the ground truth (optimal solution) is known. The set $\{\lambda_i\}$ is the eigenvector set of the input correlation $\Sigma = XX^T$.

$$E_g(t) = \frac{1}{N}\sum_i \left[\left(\sigma_w^2 + \left(\sigma_w^0\right)^2\right) e^{-\frac{2\lambda_i t}{\tau}} + \frac{\sigma_\epsilon^2}{\lambda_i}\left(1 - e^{-\frac{\lambda_i t}{\tau}}\right)^2\right] + \sigma_\epsilon^2 \tag{1}$$

Then, we can take an average over all possible data matrices X using RMT to get concentration results for the covariance matrix, which then gives that

$$\left\langle \frac{E_g(t)}{\sigma_w^2} \right\rangle_X = \int \rho^{\mathrm{MP}}(\lambda)\left[(1+\mathrm{INR})e^{-\frac{2\lambda t}{\tau}} + \frac{1}{\lambda\cdot\mathrm{SNR}}\left(1 - e^{-\frac{\lambda t}{\tau}}\right)^2\right] d\lambda + \frac{1}{\mathrm{SNR}} \tag{2}$$

where ρ^{MP} is the Marchenko-Pasteur distribution.

We will then use and interpret the above equations to understand optimal early stopping, and the mechanistic role that early stopping plays compared to weight-based regularization.

2.1.3 Proof

First, we derive (2), deriving (1) en route. Consider the student-teacher setup as in the previous section. Suppose we train our student weights with gradient flow, a continuous time version of gradient descent that allows us to write a differential equation from the loss. Taking MSE loss, we have

$$E_t(w(t)) = \frac{1}{P}\sum_{\mu=1}^{P} \|y^\mu - \hat{y}^\mu\|_2^2$$

as our loss signal, and noting that $dw \propto \frac{dE_t}{dw}$, we plug in the form of our estimator $\hat{y} = wX$ for our trainable student weights w to get the gradient flow equation updating our weights takes the form

$$\tau \dot{w} = wX - \bar{w}XX^T \tag{3}$$

for some constant τ that represents speed of learning (and eats up the factor of two from the derivative of E_t. Recall our goal here is to find the dynamics of E_g, so we seek $E_G(w(t))$ as a function of time, which we can write as $E_g(t) = \left\langle (y - \hat{y})^2 \right\rangle_{x,\epsilon}$ averaged over possible draws of data and noise.

But for now we fix X, ϵ as nonrandom samples. We will take expectations later, and make use of the tools of RMT to get general results. Before we do the obvious thing and solve (3) to compute weight dynamics, observe that the dynamics of entry w_i for $1 \leq i \leq n$ will involves terms from w_j for $i \neq j$, coupling the equations making them simultaneous. However, this can be gotten around by decorrelating in the following way. Consider the eigendecomposition $\Sigma = V\Lambda V^T$, noting that since Σ is a covariance matrix, it is positive definite, thus symmetric, and so its eigenvectors form an orthonormal set, thus V is unitary. This means $VV^T = V^TV = I$ and $V^T = V^{-1}$, we will make use of this property from now on. Then proceed write the input-output correlation as $\Sigma^{yx} = yX^T = \tilde{s}V^T$ for some $\tilde{s}$ we will determine later. We do this so that we can always work in terms of the basis of the data covariance matrix Σ. Now notice every term in (3) has a V^T term, so define $w = zV^T$ and $\bar{w} = \bar{z}V^T$ to conveniently be able to write (3) in the new variable z to see that the equation we really want to solve, without coupling, is

$$\tau \dot{z}(t) = \tilde{s} - z\Lambda \tag{4}$$

where we can observe matching terms that

$$\tilde{s}V^T \equiv yX^T = \bar{w}XX^T + \epsilon X^T = \bar{w}V\Lambda V^T + \epsilon\Lambda^{1/2}V^T = (\bar{z}\Lambda + \epsilon\Lambda^{1/2})V^T$$

we have $\tilde{s} = (\bar{z}\Lambda + \epsilon\Lambda^{1/2})$. In the final equality in the chain above, note that we make use of the fact $\bar{w}V = \bar{z}V^TV = \bar{z}$ since V is unitary. Thus our equation (22) of interest *can now be written in terms of its components* to get the entry-wise system of (decoupled) equations

$$\tau \dot{z}_i = (\bar{z}_i - z_i)\,\lambda_i + \epsilon_i\sqrt{\lambda_i}, \quad i = 1, \cdots, N \tag{5}$$

Where we think of the modes z_i as what we are "really learning" (up to a rotation V) when we learn weights $w = zV^T$ under gradient descent. We can also see the exact role that data has on learning dynamics: the speed of learning under gradient flow in mode i *is exactly* λ_i, the i-th eigenvalue of the data covariance $\Sigma = XX^T$. Crucially, in this basis, we see that the learning speed of each mode i is independent of every other mode j because (5) is an equation in one variable. Also, we can see that error guiding the learning of the modes z_i is the ground truth from the labels $\bar{z}_i$ up to a factor of noise. The solution to this differential equation is

$$\bar{z}_i - z_i = (\bar{z}_i - z_i(0))\,e^{-\frac{\lambda_i t}{\tau}} - \frac{\tilde{\epsilon}_i}{\sqrt{\lambda_i}}\left(1 - e^{-\frac{\lambda_i t}{\tau}}\right) \tag{6}$$

Now recall that

$$
\begin{aligned}
E_g(t) &= \left\langle [y-\hat{y}]^2 \right\rangle_{x,\epsilon} = \left\langle [(z-\bar{z})^2 X X^T + \epsilon]^2 \right\rangle_{x,\epsilon} \\
&= \left\langle (z-\bar{z})^2 \right\rangle_{x,\epsilon} + \sigma_\epsilon^2 = \frac{1}{N} \sum_i \left\langle (\bar{z}_i - z_i)^2 \right\rangle + \sigma_\epsilon^2 \\
&= \frac{1}{N} \sum_i \left[\left(\sigma_w^2 + \left(\sigma_w^0\right)^2 \right) e^{-\frac{2\lambda_i t}{\tau}} + \frac{\sigma_\epsilon^2}{\lambda_i} \left(1 - e^{-\frac{\lambda_i t}{\tau}} \right)^2 \right] + \sigma_\epsilon^2
\end{aligned}
\tag{7}
$$

Where the last equality is just obtained by plugging in (6). Thus we have the test error dynamics we initially wanted in (2) for our toy model. Before we take another expectation to get (3) – this time over the covariance eigenspectrum – we stop to interpret (7) to see what a shallow linear network learns under these dynamics. We make a couple of observations.

- Immediately, we see cannot do better in expectation than the noise allows us to. This makes sense, as we can't get something for nothing.
- Looking at the first term inside the sum, we see it decays quickly with time, and represents a difference in scale between initialization desired features. The scale of our solution grows exponentially fast in time over the course of our training, but of course if our initialization scale is way off, that means that this error is higher for longer.
- The second term saturates exponentially fast to $\frac{\lambda_\epsilon^2}{\lambda_i}$. Note that this will blow up for small, nonzero eigenvalues. This means that if the covariance matrix has a few such eigenvalues, test risk will be high. We will see a regime where this happens often in expectation in terms of N, P via RMT soon, giving some unexpectedly shaped learning curves. One can think of this as a measure of "overfitting" along mode i.
- From (6) itself we can note that there is no movement along modes with zero eigenvalue over time. For these modes, initialization matters a lot because that determines their end-time values z_i. We call the modes for which λ_i as comprising a *frozen subspace* in which no learning occurs under gradient descent. We will find this is a form of implicit regularization: all the w_i are moving, but they are moving along *at most* a P-dimensional subspace of weight space, because many z_i are frozen because $\lambda_i = 0$.
- Note that timescales for convergence are $O(\frac{\tau}{\lambda_{\min}})$ where $\lambda_{\min}$ is the smallest *nonzero* eigenvalue. This tells us something important: the *eigengap* of the covariance matrix determines time to convergence. We will see this quantity is important in other ways as well, and reason about it using RMT.

When we examine some of the empirics in the paper that simulate deeper networks, we will find many of the phenomena above remain. This is why this toy model is so informative. It tells us about 1) how double descent can occur due to the overfitting term, and the implicit regularization effect of 2) frozen subspaces of weights and 3) the covariance eigengap. These are all important aspects of deep learning, and it seems they have explanations rooted in something as pedestrian as a shallow linear network.

Now we proceed to do what we set out to: derive equation (2). In fact, this is just one step away from equation (7), we just need to take the average over all possible eigenvalues weighted by the eigenvalue distribution of the covariance matrix. But what is this distribution? For large $P, N \to \infty$ with $P/N = \alpha$ it is exactly the Marchenko-Pasteur, whose spectral statistics are well understood. So we pass into exactly this limit, and observe that by LOTUS (Law of the Unconscious Statistician), we can find the expectation of (7) by just integrating with respect to the MP distribution:

$$
\langle E_g \rangle_X = \int d\rho^{\mathrm{MP}}(\lambda) \left(\frac{1}{N} \sum_i \left[\left(\sigma_w^2 + \left(\sigma_w^0\right)^2 \right) e^{-\frac{2\lambda_i t}{\tau}} + \frac{\sigma_\epsilon^2}{\lambda_i} \left(1 - e^{-\frac{\lambda_i t}{\tau}} \right)^2 \right] + \sigma_\epsilon^2 \right) \tag{8}
$$

Of course, this can be simplified and made more interpretable. In particular, we can divide through by σ_w^2 so that $\frac{(\sigma_w^0)^2}{\sigma^2} \equiv INR$ and $\frac{\sigma_\epsilon^2}{\sigma_w^2} \equiv$ SNR becomes the Initialization to Noise Ratio and Signal To Noise Ratio, respectively. The first is a measure of how large our initialization scale is, and the second a measure of how much teacher weights contribute to labels compared to noise. With these defined, the *predicted typical generalization error* (over datasets and noise) becomes

$$\frac{E_g(t)}{\sigma_w^2} = \int \rho^{\mathrm{MP}}(\lambda) \left[(1+\mathrm{INR}) e^{-\frac{2\lambda t}{\tau}} + \frac{1}{\lambda \cdot \mathrm{SNR}} \left(1 - e^{-\frac{\lambda t}{\tau}}\right)^2 \right] d\lambda + \frac{1}{\mathrm{SNR}} \tag{9}$$

which is exactly (3), as we wanted. With our cake in front of us now, we're going to stop and extract some insights from this, with an eye toward understanding early stopping optimally, and how early stopping can play a role similar with L_2 regularization.

Early Stopping and Regularization

Crucially, we've now solved for *dynamics* of generalization error, which is stronger than just end-time test risk. In particular, generalization error, as one knows if one has ever trained a neural network, is not always monotonically decreasing over the course of training (in the offline setting). This is why early stopping is sometimes used as a form of regularization, to take the best parameters from throughout training, as opposed to just at the end. So, with our dynamics in hand, what t should we pick to minimize generalization error? Well, we could simply take the derivative of (9) and set to zero and solve for t. Indeed, this would work. Instead, we'll use a known analogous form for optimal L_2 *regularization*, set it equal to (9), and see what t must be to match its test risk. This works to find t^{opt} because it is known that if ridge γ matches noise σ_ϵ^2, that is the best we can do in terms of minimizing test risk. From [3], by the same authors, we have that this test risk, for the same task we're considering but now regularized, is given by

$$\frac{E_g^{L2}(\gamma)}{\sigma_w^2} = \int \rho^{\mathrm{MP}}(\lambda) \left[\frac{\gamma^2}{(\lambda+\gamma)^2} + \frac{\lambda^2}{\lambda \cdot \mathrm{SNR}(\lambda+\gamma)^2} \right] d\lambda + \frac{1}{\mathrm{SNR}} \tag{10}$$

Which is minimized by setting $\gamma = \frac{1}{\mathrm{SNR}}$. This makes sense – we regularize less (upweight the data more) as we have more signal from the data. Thus, to find optimal stopping time t^{opt}, note that we can just equate the terms in the square brackets in (10) with those in (9), plugging in $\gamma = \frac{1}{\mathrm{SNR}}$, and solve out for t^{opt}? Doing so gives $t^{\mathrm{opt}} = \frac{\tau}{\gamma} \log(\mathrm{SNR} \cdot \lambda + 1)$.

What does this tell us? Because of the λ term, it tells us that optimal stopping could match L_2 regularization's test risk (the best possible for this noisy task) if and only if there was one constant value for λ. This would happen is $\rho^{\mathrm{MP}}(\lambda) = \delta_\lambda$ was a δ function. In reality, $\rho^{\mathrm{MP}}(\lambda)$ does not concentrate this way, so we know that no matter where when we early stop during training dynamics, we will never (in expectation over data and noise) be able to do optimally, in the sense of regularization. However, we can come close. Simulations show that using the stopping time above works very well in practice even if not reaching the theoretical optimum.

Training Error Dynamics

Now we derive dynamics for the train error. In particular, in doing so, we want to see what makes the train error 1) monotone decreasing (where the test error may not be) and 2) what makes neural networks able to memorize noisy labels for $N > P$ with only slightly more training time. Since our data matrix is not square, consider its SVD, which is guaranteed to exist $X = U\Lambda^{1/2}V^T$ and see that the train error becomes

$$E_t(w(t)) = \frac{1}{P}\|y - w(t)X\|_2^2 = \frac{1}{P}\left\|y - z(t)\Lambda^{1/2}U^T\right\|_2^2 \tag{11}$$

where we use the same convention that $w(t) \equiv z(t)V^T$ as before. Assume U is unitary or can be extended to a unitary matrix $\tilde{U}$ so that

$$E_t(w(t)) = \frac{1}{P}\left\|y\tilde{U} - z(t)\Lambda^{1/2}U^T\tilde{U}\right\|_2^2 = \frac{1}{P}\left\|\tilde{\epsilon} + (\bar{z} - z(t))\Lambda^{1/2}\right\|_2^2 + \left\|\epsilon U^\perp\right\|_2^2 \tag{12}$$

then rearranging (6) into the form

$$\tilde{\epsilon}_i + \sqrt{\lambda_i}\left(\bar{z}_i - z_i(t)\right) = \left(\sqrt{\lambda_i}\left(\bar{z}_i - z_i(0)\right) + \tilde{\epsilon}_i\right) e^{-\frac{\lambda_i t}{\tau}} \tag{13}$$

and plugging this into the interior of (12) gives that

$$E_t(w(t)) = \frac{1}{P}\left(\sum_{i=1}^{N}\left(\sqrt{\lambda_i}\left(\bar{z}_i - z_i(0)\right) + \tilde{\epsilon}_i\right)^2 e^{-\frac{2\lambda_i t}{\tau}} + \sum_{j=1}^{P-N}\left(\tilde{\epsilon}_j^{\perp}\right)^2\right) \tag{14}$$

where then averaging over the data, noise, and weights gives the final expression

$$\langle E_t(w(t)\rangle = \frac{1}{\alpha}\int \rho^{MP}(\lambda)\left(\lambda\left(\sigma_w^2 + \left(\sigma_w^0\right)^2\right) + \sigma_\epsilon^2\right) e^{-\frac{2\lambda t}{\tau}} d\lambda + \left(1 - \frac{1}{\alpha}\right)\sigma_\epsilon^2 \mathbf{1}[\alpha > 1] \tag{15}$$

From which we can extract a few insights.

- We can in theory interpolate full only with $N > P$ because the second term is otherwise nonzero.
- The first term (the average over eigenvalues) vanishes as $\alpha \to \infty$ we approach the online limit of infinite data. It also vanishes over time anyways because of the exponential at the end, but with timescale $O(\frac{\tau}{\lambda_{\min}})$ as usual.
- This equation explains the generalization puzzle in [4] that asks why networks find it so easy to fit random noise with only slightly more training steps. Any setting in which noise can be memorized is $\alpha < 1$ (the common, overparameterized regime in modern deep learning). In this setting, consider the end-time solution to (15). The exponential term with $\lambda = \lambda_{\min}$ dominates, so we can take what is called "the Saddle point approximation" of the integral as its largest term to say that

  $$E_t(w(t)) \propto \left(\lambda_{\min} + \frac{1}{\text{SNR}}\right) e^{-\frac{2\lambda_{\min} t}{\tau}} \tag{16}$$

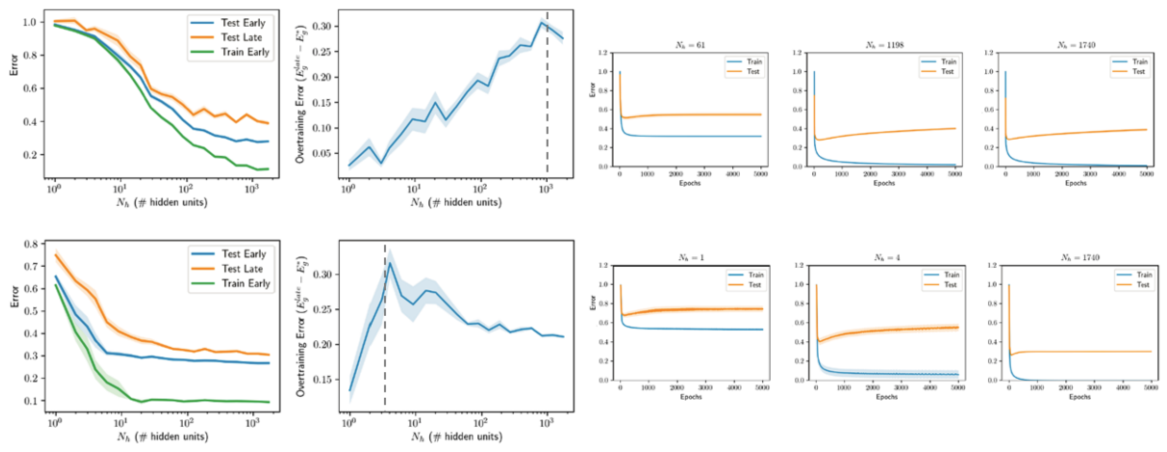
  Which tells us that the time t required to dip below a train error of δ is roughly

  $$t \approx \frac{\tau}{2\lambda_{\min}} \log\left(\frac{\lambda_{\min} + \frac{1}{\text{SNR}}}{\delta}\right) \tag{17}$$

  This tells us that timescales to memorize are logarithmic in noise! Thus memorizing completely random noise should be possible given 1) $N > P$ and 2) slightly longer time scales than if SNR was high (because of the log term in above). This explains a big generalization puzzle in [4]!

Deep, Nonlinear Networks

How does our analysis of the shallow, linear network dynamics extend to an actual deep learning setting? Well, in fact. Below are plots taken from the paper where a deep, nonlinear network was trained on MNIST. We can see several properties from our simple model are preserved, including double descent, the importance of eigengap at the interpolation threshold $P/N = 1$, and the fact that larger models can still generalize well. The paper gives some formal treatment of deep linear networks and only simulations for nonlinearities (which have since been treated mathematically in more recent work in this subfield). The key complication in the depth setting is that we cannot uncouple dynamics into individual differential equations in each z_i, so we have to solved a nonlinear dynamical system in a lot of variables, which is difficult. Of course, if it were easy, we would not need deep learning (where SGD on neural network loss landscape is implicitly solving such equations)!



2.1.4 Other Results

Stronger Rademacher Generalization Bounds

Part of the reason that deep learning is so surprising is because it violates the classical statistical lore that bigger models can overfit. This lore is made formal by generalization bounds using various measures of complexity. Formally, these bounds take the form $E_g - E_G \leq f(C(H))$ where $C(H)$ is some complexity measure of a hypothesis class H. One famous measure of complexity is $C(H) = R(H)$ the Rademacher complexity, defined as

$$\mathcal{R}(H) = \left\langle \sup_{h \in H} \frac{1}{P} \sum_{\mu=1}^{P} \sigma^\mu h\left(x^\mu\right) \right\rangle_{\sigma, x}$$

where μ index P data points and σ_i are random signs. Intuitively, this measures the capacity of the class H to memorize random labels, and uses it as a measure of complexity of the class, and $R(H) \in [0,1]$ for any H. One famous generalization bound on P samples is that, with probability at least $1-\delta$, we have

$$E_g - E_t \leq 2R(H) + \sqrt{\frac{\log \frac{1}{\delta}}{2P}}$$

Thus we see that classically, if $R(H)$ is increasing in neural network size, then we should expect the generalization gap to widen, and network generalization for larger models to be better. The paradox for machine learning is that they are counterexamples to such bounds. Or so it seems.

One important bound from [5] on the Rademacher complexity of neural networks is

$$\mathcal{R}(H) \leq \frac{B_2 B_1 C \sqrt{N_h}}{\sqrt{P}} \tag{18}$$

which was derived for one-hidden layer ReLU networks. B_2 is the norm of the output weights, B_1 is the max weight of input neurons, and C is the max norm of the input data. Thus we can see that bounding the norm of the readout weights suffices to control the complexity of such a network. As early stopping stops the weights (that grow to match the task scale, when initialized small), it is indeed a regularize in the sense of bounding complexity. But we'll now see that both the *frozen subspace* effect and *eigengap* play an important role in constraining readout weight norm, allowing us to use our analysis above to provide tighter bounds than (11), that will actually lead us to see that Rademacher complexity is *not* monotone increasing in model size, which will tell us that deep learning is *not* actually at odds with classical computational learning theory.

Consider the one hidden-layer ReLU. If we fix the first layer, then the architecture $\hat{y}(x) = \sum_{a=1}^{N_h} \frac{w_a}{\sqrt{N_h}} \phi_a(x)$ can be see as a version of our initial linear model, characterized by (3), with "data matrix" $X_h^{a\mu} = \frac{1}{\sqrt{N_h}} \phi_a(W_a^1 \cdot x^\mu)$. Then we can write the eigendecomposition of this new "data matrix" as $X_h X_h^T = V_h \Lambda_h V_h^T$. Recall that the solution fo the weight dynamics in our linear model (6) in terms of this basis gives us that the norm of the weights takes the form

$$\|w(t)\|^2 = \sum_i z_i^2(t) = \sum_i \left(\frac{\left\|\tilde{z}_i^h\right\|^2}{\left(\lambda_i^h\right)^2} \left(1 - e^{-\frac{\lambda_i^h t}{\tau}}\right)^2 + \left\|z_i^h(0)\right\|^2 e^{-2\frac{\lambda_i^h t}{\tau}} \right) \tag{19}$$

Set initial weights to approximately zero so the second term vanishes then we get that

$$\frac{\|w\|}{\sqrt{N_h}} \leq \sqrt{\frac{\max_i \left\|\tilde{z}_i^h\right\|^2}{\min_{i,\lambda_i^h>0} \left(\lambda_i^h\right)^2} \frac{\min\left(P, N_h\right)}{N_h}} = B_2 \tag{20}$$

which, plugged back into our generalization bound, give

$$\mathcal{R}(H) \leq B_1 C \sqrt{\frac{\max_i \left\|\tilde{z}_i^h\right\|^2}{\min_{i,\lambda_i^h>0} \left(\lambda_i^h\right)^2} \frac{\min\left(P, N_h\right)}{P}} \tag{21}$$

Which makes use of information about 1) the frozen subspace property in the form of $\min(P, N_h)$ and 2) the eigengap $\min_i(\lambda_i^h)^2$. Notice, crucially, that this tells us 1) that increasing the model size in terms of P, N_h while the other stays fixed does not increase the complexity upper bound and thus doesn't raise the bound on the generalization gap, and 2) the generalization gap is upper bounded by something decreasing in the eigengap. Notice that the eigengap is *decreasing in model size*, so that the bound on our complexity is slightly smaller for larger models! This means that *larger models have slightly lower Rademacher complexity based generalization bounds*, so that the fact that larger neural networks generalize is not a contradiction to classical statistics! Hoorah!

2.2 Spectral Bias and Task-Model Alignment

2.2.1 Problem Setup

Here we provide some background on kernels and kernel regression, without diving into too many of the details. First, a kernel $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ can roughly be thought of as a similarity function between two data points. Kernel regression is then outputting a prediction $y(x) = \sum_i K(x, x_i) y_i$ as a weighted combination of points in our data set. One special case of this is the Kernel Ridge Regression estimator $\hat{y}_{\mathrm{KRR}}(x) = K(X, x)(K(X, X) + \lambda I)^{-1} y(X)$ where X is the train set and x is a test point we want to estimate the label for. Note that this is the posterior mean given the data in a Bayesian interpretation.

The set of functions we can reach by regressing this way (given some regularity conditions on our choice of kernel, K) forms a subset of the Hilbert Space L_2 of functions, call this subset $\mathcal{H}$, often referred to as an RKHS ("Reproducing Kernel Hilbert Space"). We note K is itself a member of this space in either of its arguments, and thus can be decomposed in terms of an orthogonal eigenbasis of this space (as can any element of this vector space of functions). This decomposition is called the Mercer Decomposition, and is given by $K(x, x') = \sum_{i=1}^{\infty} \eta_i \phi_i(x) \phi_i(x')$ for $\{\phi_i\}_{i=1}^{\infty}$ an orthonormal eigenbasis of the space. Formally, we should show things like existence of such a basis, proof that indeed $K \in \mathcal{H}$, a precise definition of a RKHS, but in the spirit of brevity we omit those here.

We now note something important: wide neural networks are equivalent to kernel regression at inference-time. In particular, let inference with a wide neural network with weights θ be given by f_θ. Then, it turns out that $f_\theta = KRR(NTK_\theta)$ where KRR stands for the Kernel Ridge Regression estimator with kernel being the "Neural Tangent Kernel" defined as $NTK(x, x') = \frac{df_\theta}{d\theta}(x) \cdot \frac{df_\theta}{d\theta}(x')$. Though this fact is stated here concisely, is not obvious, and realizing that neural networks behave essentially like kernel methods (which, in turn, are generalized linear models) was a big achievement of ML theory. This is important because it means that the generalization error derived below for kernel regression also applies to wide neural networks with the kernel being the NTK, so it means we can predict generalization of a given network architecture (on average over all Gaussian data sets) by just looking it (and its induced NTK) and in particular without looking at any data. This generalization error is not equal to its exact generalisation properties on a *fixed* data set, but is a useful proxy.

TL;DR of this paper. This paper derived a closed form for the high-dimensional asymptotic mean (in $P, D \to \infty$ while $P/D = \alpha \ll \infty$) for kernel ridge regression (KRR) in terms of the number of training samples, training distribution, and kernel eigenspectrum. This is done via a replica calculation. Importantly, while the first paper in this report treats test risk *dynamics*, this just computes the static *end-time* test risk in a kernel setting.

2.2.2 Main Result

Suppose our choice of kernel is K, with eigenvalues η_ρ, and we are regressing a target function $\bar{f} \in \mathcal{H}$ which can be eigendecomposed as $\bar{f} = \sum_\rho \bar{w}_\rho \phi_\rho$. P is the number of data points, and data is generated according to $y^\mu(x) = \bar{f}(x^\mu) + \epsilon^\mu$ where $\mu \in [P]$ and $\epsilon \sim N(0, \tilde{\sigma}^2 I)$ is noise. The ridge parameter is given by λ. The kernel regression problem is then

$$f^* = \underset{f \in \mathcal{H}}{\arg\min} \frac{1}{2\lambda} \sum_{i=1}^{P} \left(f\left(\mathbf{x}^\mu\right) - y^\mu\right)^2 + \frac{1}{2} \langle f, f \rangle_{\mathcal{H}}$$

and the resulting generalization error is then

$$E_g = \left\langle \int d\mathbf{x} p(\mathbf{x}) \left(f^*(\mathbf{x}) - \bar{f}(\mathbf{x})\right)^2 \right\rangle_{\mathcal{D}}$$

where this is *averaged over all possible random train data sets drawn from the population.* This is the quantity that the paper is interested in computing in the limit $P, N \to \infty, P/N \ll \infty$. Notationally, note the physicists' shorthand for averages

$$\langle f \rangle_{X \sim p(x)} = E_X[f] = \int_{\mathbb{R}} p(x) f(x) dx$$

With this notation established, the main result is then:

$$\begin{aligned} E_g &= \frac{1}{1-\gamma} \sum_\rho \frac{\eta_\rho}{(\kappa + P\eta_\rho)^2} \left(\kappa^2 \bar{w}_\rho^2 + \tilde{\sigma}^2 P \eta_\rho\right) \\ \kappa &= \lambda + \sum_\rho \frac{\kappa \eta_\rho}{\kappa + P\eta_\rho}, \quad \gamma = \sum_\rho \frac{P\eta_\rho^2}{(\kappa + P\eta_\rho)^2}. \end{aligned} \tag{22}$$

So we have our desired formula as a pure function of the number of data points, the eigenvalues of our chosen kernel, and the eigencoefficients of the target function $\bar{f}$ as well as the noise level in our labels.

2.2.3 Proof Sketch

The generalization error is an average over all possible datasets, D. A useful trick invented in statistical physics to deal with such averages is called the *replica trick*. We briefly introduce the replica trick in generality, and then we'll skim how it is applied specifically to compute equation (22).

The Replica Trick

The key idea is to think of the mean-squared error of a kernel regression problem as the Hamiltonian of a statistical mechanics problem, construct a partition function, take the limit $P, N \to \infty, P/N \ll \infty$, and use the tools of statistical mechanics to compute the partition function (this is the part that makes use of the "replica trick"). The only fact from stat mech we use is that physical systems tend to minimize their energy (Hamiltonian).

Motivation

In CS and ML, we often care about minimizating a function of the form $H(x; D)$ where x are parameters we minimize over (such as weights of a neural network or regression parameters) and D is a set of random variables characterizing the cost function we want to minimize (such as the training dataset which characterizes the loss function). In particular, we care about knowing some statistic about the minimizing solution, $O(x)$. In our case, $O(x)$ could be the generalization error (over all datasets) of the parameters that minimize KRR error. Let S be the set/surface of minimizing parameters $S \subset \mathcal{X}$ for the parameter space $\mathcal{X}$. Thus, we want to compute

$$E_D E_{x \in S} O(x; D) = \left\langle \int p(x) O(x; D) dx \right\rangle_D$$

where $p(x)$ is a uniform measure over S. We can use the replica method to compute this when a few assumptions are (approximately) satisfied:

1. Self-averaging: denote $f(D) = \int p(x) O(x; D) dx$ be a function of the dataset. This is the function that takes in a random dataset and outputs the "typical value of the observable (eg. error)" for that dataset over all minimizers. Then we require $f(D)$ to concentrate in the sense of being close to its expected value. In particular, we need $\frac{f(D)}{E_D[f]} \to 1$ as $N \to \infty$. If we let $S = \sum_{i=1}^N X_i$ for X_i coin flips, then $\frac{S}{ES} \to 1$ as $N \to \infty$, so S is self-averaging. However, for instance, 2^S is not self-averaging. This is an instance of the general fact that while many summary statistics of random variables may not self-average, their logarithms do, so studying $\langle \log S \rangle$ often suffices to understand S (this motivates the form of the coming replica trick).

2. Thermodynamic limit: we take $P, N \to \infty$ while $P/N \ll \infty$.

3. Loss function: we assume H is not too complicated, in a sense that can be made precise.

The key steps of any replica calculation

1. We seek $\left\langle \int p(x)O(x;D)dx \right\rangle_D$.
2. Reduce problem to finding $\langle \log Z\rangle_D$ for appropriate choice of partition function. In our case, construction the partition function
$$Z(D,J) = \int_S \exp(-\beta H(x;D) + JO(x;D))dx$$
for a dummy term J gives us that
$$\lim_{\beta\to\infty} \frac{d}{dJ}\langle \log Z(D,J)\rangle_D \bigg|_{J=0} = \left\langle \int p(x)O(x;D)dx \right\rangle_D$$
for our choice of Z by construction. Importantly, this is because p_β characterized by the Gibbs measure converges to the uniform measure inside the expectation, but we used the Gibbs measure because the tools of stat mech are in terms of that. The two averages (over minimizing x then over D, are called the "thermal" and "quenched" averages, respectively).
3. Use replica trick to reduce $\langle \log Z_\beta(D)\rangle_D = \lim_{n\to 0} \frac{1}{n}\log\langle Z_\beta(D)^n\rangle_D$.
Assuming the identity above, we now note that
$$\langle Z^n\rangle_D = \left\langle \int_{\vec{x}\in S^n} \exp(-\beta\sum_{i=1}^n H(x_i;D) + JO(x_i;D))dx \right\rangle_D$$
4. Reparameterize with order parameters (overlap matrix) $\langle Z_\beta(D)^n\rangle_D = \int \exp(-nN\mathcal{F}(Q))dQ$. We have guarantees that say that there exists a "nice" function $\mathcal{F}$ that makes this true.
5. Saddle-point approximation for this integral (special case of Laplace's method, a standard trick for computing exponential integrals) tells us $\langle Z^n\rangle_D \approx \exp(-nN\mathcal{F}(Q^*))$, so that original quantity via replica trick is now
$$\langle \log Z_\beta(D)\rangle_D = \lim_{n\to 0}\frac{1}{n}\log\langle Z_\beta(D)^n\rangle_D = \lim_{n\to 0}\frac{1}{n}(-nN\mathcal{F}(Q^*)) = -N\mathcal{F}(Q^*)$$
6. Assume solution Q^* takes form $Q^* = (q-q_0)\delta_{ab} + q_0$. This conjectured form for the solution is called the replica symmetric ansatz. This is reasonable because we are taking $P\to\infty$, so this reflects that $||x_i|| \approx q$ and $x_i \cdot x_j = q_0 \approx 0$ is very small in high dimension (random high dimensional vectors are almost orthogonal). Recall that $Q_{ab} = \frac{1}{N}(x_a \cdot x_b)$ is a correlation matrix of sorts. The fact we can "guess" a solution of this sort in high dimension is at the heart of the replica method, since finding an optimal Q via FOCs on q, q_0 is what makes this difficult (high-dimensional) marginalization "easier" (two-dimensional).
7. Suffices to optimize order parameters q, q_0 via first order conditions to get $\langle \log Z\rangle_D = \max_{q,q_0} -N\mathcal{F}(Q^*)$ for Q^* of the form above.
8. Recall $\langle \log Z\rangle_D$ can be used to compute desired quantity $\left\langle \int p(x)O(x;D)dx \right\rangle_D$, and conclude that we reduced our initial, difficult problem of finding the "typical value of our observable $O(x;D)$ over random data D" to the easy first-order conditions of solving $\frac{d\mathcal{F}}{dq} = 0$, $\frac{d\mathcal{F}}{dq_0} = 0$. This is the power of the replica trick, which relies on convergence and concentration as $P, N \to \infty$.

In the paper, these steps are applied to our kernel regression problem. We omit the details of this specific calculation because it comprises 12 pages of the paper Appendix.

2.2.4 Interpreting (22)

By looking at the form of (22), we can extract a few insights.

- *Spectral Bias.* This is an important way in which high-dimensional kernel regression (and by extension, any neural network that behaves like a kernel method, for instance an infinitely wide on) is implicitly biased for efficient learning. In short, marginal new data points reduce error in higher eigenfunctions of the target function more than they do in lower eigenfunctions.

  More specifically, kernel eigenfunctions ϕ_ρ with large eigenvalues η_ρ can be estimated with kernel regression using a smaller number of samples. We can see this by noting that the generalization error can be written as a sum of modal errors arising from the estimation of the coefficient for eigenfunction ψ_ρ as $E_g = \sum_\rho \eta_\rho \bar{w}_\rho^2 E_\rho$ wherewe have that

  $$E_\rho = \frac{1}{\bar{w}_\rho^2}\left\langle \left(w_\rho^* - \bar{w}_\rho\right)^2\right\rangle_{\mathcal{D}} = \frac{1}{1-\gamma}\frac{\kappa^2}{\left(\kappa + P\eta_\rho\right)^2}.$$

  We can use this mode error equation alongside the fact that $\eta_\rho > \eta_{\rho'}$, and noting that $\kappa'(P) = -\frac{\kappa\gamma}{P(1+\gamma)} < 0$ since $\kappa, \gamma, P > 0$, we have

  $$\frac{d}{dP}\log\left(\frac{E_\rho}{E_{\rho'}}\right) = -2\left[\frac{\kappa'(P)+\eta_\rho}{\kappa + P\eta_\rho} - \frac{\kappa'(P)+\eta_{\rho'}}{\kappa + P\eta_{\rho'}}\right] < 0$$

  This is enough to get that

  $$\frac{d}{dP}\log E_\rho < \frac{d}{dP}\log E_{\rho'} \implies \frac{1}{E_\rho}\frac{dE_\rho}{dP} < \frac{1}{E_{\rho'}}\frac{dE_{\rho'}}{dP}.$$

  This gives that $E_\rho < E_{\rho'}$ for all P, so we get that the mode errors have the opposite ordering of eigenvalues: the generalization error falls faster for the eigenmodes with larger eigenvalues!

  Task-Model Alignment: Target functions with most of their power in top kernel eigenfunctions can be estimated efficiently and are compatible with the chosen kernel. We introduce cumulative power distribution, $C(\rho)$, as defined in Eq. (6), as a measure off this alignment. *Non-monotonicity:* Generalization error may be non-monotonic with dataset size in the presence of noise, or when the target function is not expressible by the kernel (not in the RKHS). This shouldn't be surprising given we saw double descent emerge in a linear model under gradient flow in the first paper in this report, and this paper instead studied kernel methods (essentially linear models in some projected feature space). However, it is interesting that even portions of the target that are out of the span of the kernel eigenfunctions act as effective "noise" for the purposes of double descent. Thus, while the noise in the first paper contributed to an overfitting term of the form $\frac{\sigma_\epsilon^2}{\lambda_i}$, we don't need explicit label noise to see double descent in kernel regression, because if the target function is out of RKHS, it is sufficiently hard to learn (compared to a simple linear model in the previous paper) that the portions of it that are unreachable by our choice of kernel can themselves induce non-monotonicity!

References

[1] Advani, M. S., Saxe, A. M., Sompolinsky, H. (2020). High-dimensional dynamics of generalization error in neural networks. Neural Networks, 132,

[2] Canatar, A., Bordelon, B., Pehlevan, C. (2021). Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. Nature Communications, 12(1), 2914.

[3] Saxe, A. M., McClelland, J. L., Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv preprint arXiv:1312.6120.

[4] Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530.

[5] Bartlett, P. L., Mendelson, S. (2002). Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. Journal of Machine Learning Research, 3, 463-482.