The problem is, the outputs range from 0 to 1, meaning the activations must have nonzero mean. The clever solution (which was standard practice for neural net training pre-ReLU) was to replace the activation function with tanh:

$$\tanh(z) = \frac{\exp(2z) - 1}{\exp(2z) + 1}.$$

It can be shown that tanh can be obtained from the logistic function through affine transformations of inputs and outputs, implying that networks with logistic and tanh activations are equally expressive. But the tanh function ranges from -1 to 1, so its outputs aren't *necessarily* uncentered. Therefore, tanh networks can in practice train much faster than logistic networks. Based on this line of reasoning, there were various attempts to find ways to exactly center (and sometimes rescale) the activations. The first one to become widely adopted was batch norm (discussed in detail in Chapter 5).

Interestingly, ReLU suffers from the same uncentering problem as the logistic function, but no solution analogous to tanh has been adopted. Batch norm was invented shortly after ReLU became popular, so evidently it was timed just right to meet the newfound need for activation centering!

4 Double Descent

We introduced the phenomenon of "double descent" in the Introduction. This phenomenon happens for plain (unregularized) linear regression as well as neural nets. Ideally, we'd like to use a linear regression model to understand why increasing model capacity past the interpolation threshold can improve generalization. Unfortunately, there isn't a clean analogue of "increasing the number of parameters" of a linear regression model, since the number of parameters is the same as the input dimension D, and adding input dimensions makes more information available to the learner (unlike adding more hidden units). Instead, let's look at the behavior for fixed D as the number of data points N is varied. We set $D = 50$, sample the input dimensions i.i.d. $x_j \sim \mathcal{N}(0, 1)$, compute the true labels as $t = \mathbf{w}^\top \mathbf{x}$ for $\mathbf{w} \sim \mathcal{N}(0, 0.1)$, and observe noisy versions of the targets as $\hat{t} \sim \mathcal{N}(t, 1)$. The training set consists of $\{(\mathbf{x}^{(i)}, \hat{t}^{(i)})\}_{i=1}^N$. We use the raw inputs (i.e. $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$) and we exactly compute the stationary (i.e. min-norm) weights $\mathbf{w}^{(\infty)}$ for the training set. Figure 4(a) shows the training and test error as N is varied from 1 to 200. We observe a double descent effect, whereby the test error peaks at $N = D$, a point called the **interpolation threshold**. This is perhaps even more surprising than the original double descent effect (where the number of parameters was varied), since we wouldn't expect *adding more data* to hurt the performance.

Hastie et al. (2019) analyze the limit for random linear regression problems as $D \to \infty$, $N \to \infty$, and $D/N \to \gamma$. Analyzing the asymptotic risk as γ is varied gives a more direct analogue of the double descent effect for neural nets. But this construction is a bit too involved for an introductory lecture.

The term *interpolation threshold* refers to the idea that when $D > N$, the model is basically interpolating the training data. The regime where $D > N$ is known as the *interpolation regime*.

What's going on? For $N > D$, with high probability the matrix $\mathbf{X}^\top \mathbf{X}$ is invertible, so $\mathbf{w}^{(\infty)}$ is the unique global minimum of the cost function. As $N \to \infty$, the test error decreases because there's more information and less overfitting, just as classical learning theory predicts. But when $N < D$, the model is **overparameterized**, so $\mathbf{w}^{(\infty)}$ is the minimum norm solution. Intuitively, when $N \approx D$, it's just barely possible to fit all the training data, i.e. this requires a large norm for $\mathbf{w}^{(\infty)}$. When N is much smaller than D, it's possible to fit the training data using a much smaller weight norm. Smaller norm weight vectors are more robust to overfitting (hence the motivation for ℓ_2 regularization), so they might generalize better.

To understand this mathematically, consider the closed-form expression for the min-norm weights, $\mathbf{w}^{(\infty)} = \mathbf{X}^\dagger \mathbf{t}$ (see Eqn. 16). Roughly speaking,

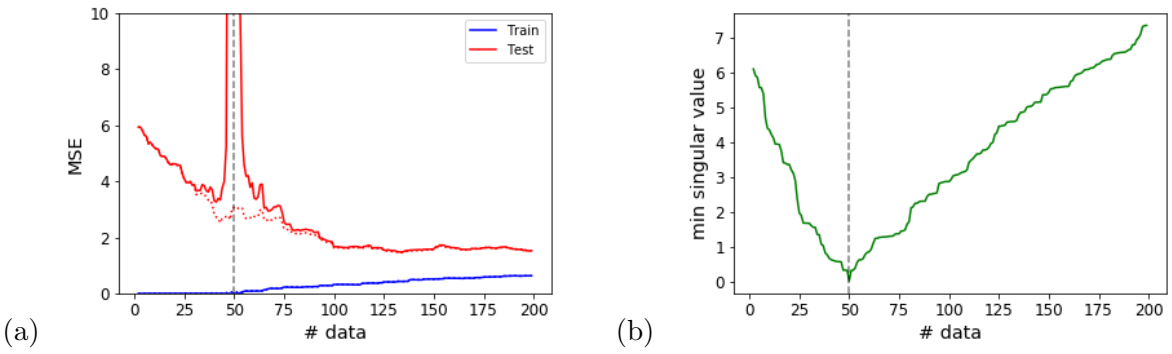

Figure 4: Double descent effect for linear regression. **(a)** Training and test error as a function of N, the number of training examples. The solid line is the unregularized model, and the dotted line uses ℓ_2 regularization with $\lambda = 1$. (The regularized and unregularized training set curves are nearly indistinguishable.) The dashed line indicates the interpolation threshold, $N = D$. **(b)** The minimum nonzero singular value of $\mathbf{X}$, as a function of N.

$\mathbf{w}^{(\infty)}$ will be large when $\mathbf{X}^\dagger$ is large. Since $\mathbf{X}^\dagger$ is defined by inverting the nonzero singular values of $\mathbf{X}$, it will be large when $\mathbf{X}$ has small nonzero singular values. Figure 4(b) plots the smallest nonzero singular value of $\mathbf{X}$ as a function of D. The minimum singular value gets closest to zero at the interpolation threshold. Explaining why this happens is beyond the scope of this course, but it follows from a basic result of a field of mathematics called random matrix theory.

The observation that performance can get worse as more data is added indicates that the double descent effect is somehow pathological. Adding more information shouldn't hurt. Indeed, the pathology can be basically eliminated (in our regression example) by adding a small amount of ℓ_2 regularization ($\lambda = 1$), without noticeably hurting the training or test error outside the pathological regime, as shown in Figure 4(b). The fact that double descent still happens even for modern, well-tuned neural nets is a reflection of how we haven't found a regularizer for neural nets that's as principled and robust as ℓ_2 regularization is for linear regression.

See Nakkiran et al. (2019) for lots of empirical demonstrations of various forms of double descent in modern neural nets.

The above discussion only scratches the surface of what's known about double descent. Hastie et al. (2019) gave basically a complete characterization of double descent for linear regression. They broke the generalization risk down into bias and variance terms, and determined the asymptotic behavior of these terms under a variety of assumptions about the data distribution. It all boils down to random matrix theory.

5 Beyond Linear Regression: Strongly Convex Optimization

We've done a lot of work analyzing one particular system: gradient descent for linear regression. What about other cost functions? Gradient descent dynamics are hard to analyze in general, since a lot of different things can happen. General cost functions can have saddle points and local optima.