Lecture 9: Neural Networks

STAT 1361/2360: Statistical Learning and Data Science

University of Pittsburgh
Prof. Lucas Mentch

Overview

- Neural networks (and the related notion of deep learning) almost certainly one of the most active areas of research and application in machine learning today
- Difficult topic to give a broad survey of: basic idea is very simple but the application in practice and “useful" extensions require much more setup
- Lecture today is intended only to give an overview of the basic idea and discuss some of the most successful recent applications

NN Basics

NN History

- The basic idea(s) behind neural networks (NNs) is actually very simple and dates back (at least) many decades
  - "Threshold logic" (McCulloch and Pitts (1943))
  - Rosenblatt's perceptron (1958) - often considered the first NN
  - Fell out of fashion in the late 1960s, came back around 1980s, and really exploded in 2010s
- Interesting to consider the fact that although many ML researchers would consider these the current "state-of-the-art", their remnants date back decades earlier than alternatives like bagging, boosting, and random forests

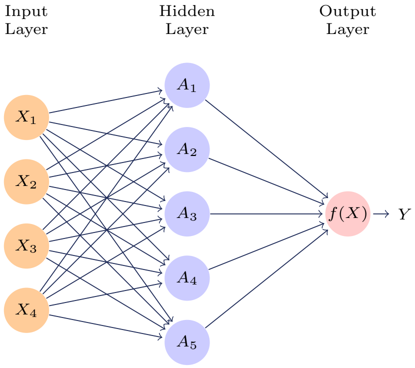
Tuning Parameters

- A basic NN model has the form

$$
\begin{aligned}
f(X) &= \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X) \\
&= \beta_0 + \sum_{k=1}^{K} \beta_k A_k \\
&= \beta_0 + \sum_{k=1}^{K} \beta_k\, g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j\right)
\end{aligned}
$$

- The generic form here should feel very familiar – we're similar building a linear model on a transformation of the original features/predictors $X_1, \ldots, X_p$
- The transformations (new transformed variables) are called activations, $A_1, \ldots, A_K$. The intercept terms $w_{k0}, \ldots, w_{K0}$ are often called the "bias" terms.

NN Basics



ISLR Figure 10.1: A Neural network (NN) with 4 features in orange ($X_1, \ldots, X_4$), one hidden layer in purple with 5 activations ($A_1, \ldots, A_5$), and single response Y.

NN Basics

- Note that the final model for this NN with a single hidden layer is given by

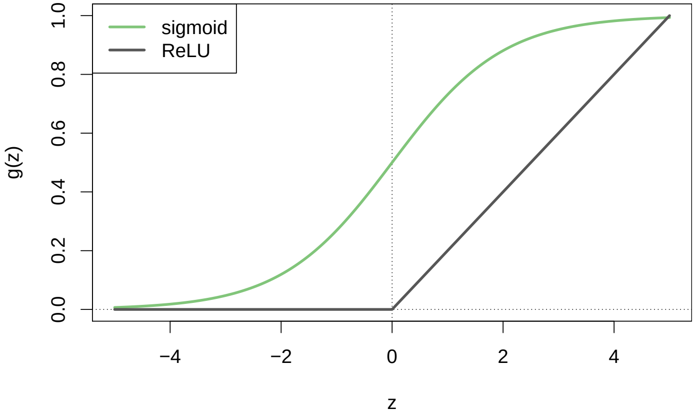
$$Y = \beta_0 + \beta_1 A_1 + \cdots + \beta_K A_K$$

- Should look very familiar – this was the exact setup we discussed with PCA and PLS

- The only real key difference here is that those activations A_k (functions of the original features X) are *non-linear*. Two common choices are the sigmoid (as in logistic regression) and rectified linear unit (ReLU):

$$g_{\text{sig}}(z) = \frac{e^z}{1 + e^z} \qquad g_{\text{ReLU}}(z) = (z)_+ = \max\{z, 0\}$$

NN Basics



ISLR Figure 10.2: Sigmoid vs ReLU activation functions. ReLU are most popular now due to the simplicity of storage. Note that the plot of the ReLU function above is scaled down for ease of comparison with the sigmoid.

NN Basics

- Note that NN get their name because this kind of network is a rough/basic form of how we think about (real) *biological* neural networks – neurons receive some kind of input/signal and depending on how much signal each receives, that signal might be passed on to other connected neurons.

- The combination of neurons that make it to the end produce a particular kind of output or result

NN Basics

- What's one big difference you should notice right away with NNs compared with traditional linear models or even linear models on transformed features like PCA and PLS?

  Think in terms of model complexity ...

NN Basics

- What's one big difference you should notice right away with NNs compared with traditional linear models or even linear models on transformed features like PCA and PLS?

  Think in terms of model complexity ...

- These are *really* complex – all parameters $\beta_0, \ldots, \beta_K$ and $w_{10}, \ldots, w_{Kp}$ need to be estimated: $(K+1) + K \cdot (p+1)$ total. For example, if we begin with a 3-feature model (X_1, X_2, X_3) and build a NN with 3 activations, we need to estimate:

$$\begin{gathered}\beta_0, \beta_1, \beta_2, \beta_3 \\ w_{10}, w_{11}, w_{12}, w_{13} \\ w_{20}, w_{21}, w_{22}, w_{23} \\ w_{30}, w_{31}, w_{32}, w_{33}\end{gathered}$$

Multilayer NNs

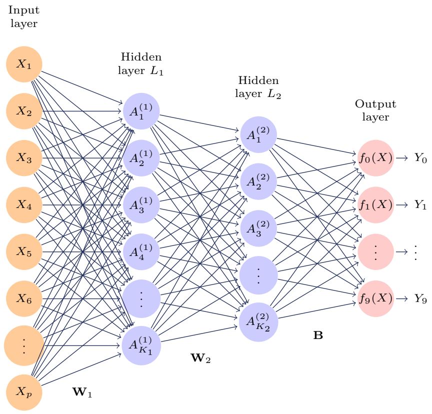
Multilayer NNs

- But while these might be quite complex models relative to most others we've discussed in this course, these *single-layer* networks are actually some of the simplest forms of NNs
- Most NNs used for modern tasks are *multilayer* NNs that contain multiple hidden layers with many nodes per layer
- In these cases, the activations from the previous layer are treated as the inputs for the following layer:

$$A_k^{(1)} = h_k^{(1)}(X) = g(\boldsymbol{w}_k^{(1)}, X)$$
$$A_k^{(2)} = h_k^{(2)}(X) = g(\boldsymbol{w}_k^{(2)}, A_k^{(1)}) = g(\boldsymbol{w}_k^{(2)}, g(\boldsymbol{w}_k^{(1)}, X))$$

- Importantly, note that this is just a composition of functions – the "end" functions are ultimately still just functions of the original X. This is one baffling aspect of NNs – why does something like something like $a(b(c(X)))$ seem to be so much better than something like $d(X)$?

Multilayer NNs



ISLR Figure 10.4: A multilayer NN with two hidden layers and multiple outputs (corresponds to the popular MNIST handwriting data). This NN contains a total of **235,146** (!) parameters.

Multilayer NNs

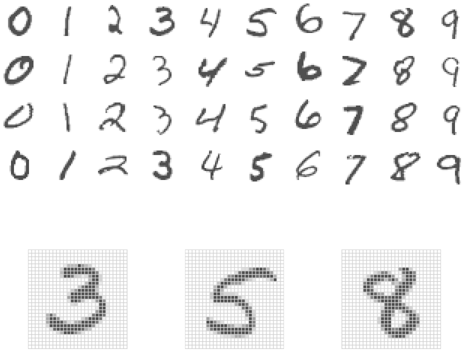
- Previous figure looks a bit strange because there are multiple outputs ... this is just an example of *one-hot encoding* where the true response is categorical but we code the “true value" as a vector of indicator functions

- This particular example is from the `MNIST` dataset – 60,000 training images and 10,000 test images of handwritten digits 0-9 where the response is just the actual number that is written

- Each image contains 28×28 pixels (784 features), each of which takes a value between 0-255 (greyscale)

- In this case a response of “3" would be coded as $(0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$

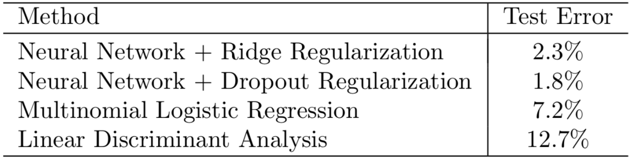
Multilayer NNs



ISLR Figure 10.3: Examples of images from the MNIST dataset.

Multilayer NNs

Performance comparisons from ISLR of NNs with $p = 784$ inputs (features), $K_1 = 256$ nodes in the first hidden layer, and $K_2 = 128$ nodes in the second hidden layer ...



Method	Test Error
Neural Network + Ridge Regularization	2.3%
Neural Network + Dropout Regularization	1.8%
Multinomial Logistic Regression	7.2%
Linear Discriminant Analysis	12.7%

TABLE 10.1. *Test error rate on the* `MNIST` *data, for neural networks with two forms of regularization, as well as multinomial logistic regression and linear discriminant analysis. In this example, the extra complexity of the neural network leads to a marked improvement in test error.*

Multilayer NNs

- Note that even with a very large training set (n =60,000) images, the NNs here still contain more than 235,000 parameters, so we're in a high-dimensional ($p > n$) setting, and thus some form of regularization is needed

- Note also that the (standard) logistic regression model fit here will contain only 7065 terms ... 785 inputs (including the intercept) $\times\ 10 - 1 = 9$ classes. So the NNs contain about 33× as many parameters.

- Worth pausing here for a moment to let that sink in ... consider just how vastly more complicated these NNs are than their more traditional statistical model counterparts

Advanced NNs

Advanced NNs

- Modern data can be very complex and advanced; for some of these data types specific kinds of NNs have been developed

- **Disclaimer:** The next several slides are merely a *very* rough summary of the book's very rough summary. These are intended only to provide an intuitive overview of these kinds of NNs. The book provides some additional details, but understand that there are entire graduate-level courses devoted to going through just this topic alone.

Convolutional NNs

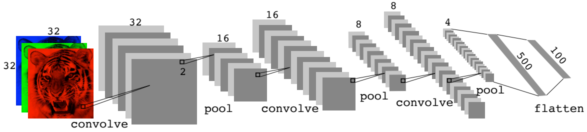
Convolutional NNs (CNNs)

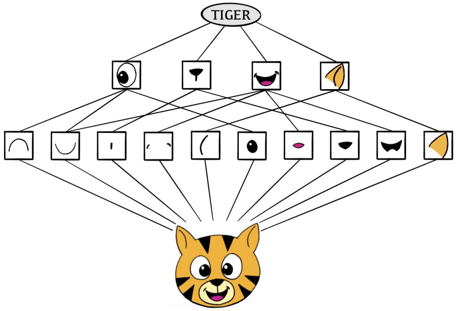
- Commonly used for image classification
- Book gives example of `CIFAR100` dataset – each input is a color image with a 32×32 resolution and 3 values (RGB) per pixel – even in the most basic form, that's $32 \times 32 \times 3 = 3072$ initial features
- CNNs are generally characterized by going back and forth between *convolution* and *pooling* layers. These have precise mathematical meanings, but can broadly summarized as:

  convolution: look for small-scale spatial patterns

  pooling: organize previous patterns into natural categories

Multilayer NNs





ISLR Figures 10.8 (top) and 10.6 (bottom)

Recurrent NNs

Recurrent NNs (RNNs)

- RNNs are used when the features have some kind of natural ordering. For example ...
  - Document/Musical classification – if we wanted to do some kind of sentiment analysis, for example, the ordering of the features (words) matters a great deal. If we want to predict “genre" of music, the order of notes or sequence of rhythms matters.
  - Time Series – if we’re looking at how something changes over time, then ordering by time is important for prediction (same applies to spatial data)
- In RNNs, features are added to the model in sequential order, “activated", and then added as features at the next step

Recurrent NNs

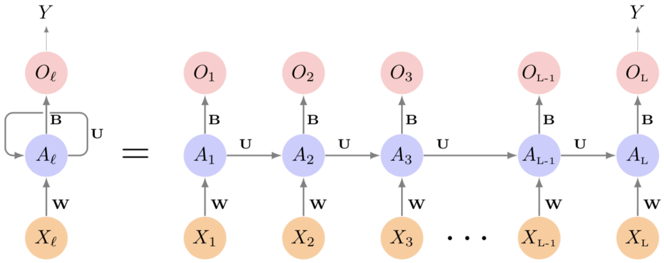


FIGURE 10.12. *Schematic of a simple recurrent neural network. The input is a sequence of vectors* $\{X_\ell\}_1^L$*, and here the target is a single response. The network processes the input sequence* X *sequentially; each* X_ℓ *feeds into the hidden layer, which also has as input the activation vector* $A_{\ell-1}$ *from the previous element in the sequence, and produces the current activation vector* A_ℓ*. The same collections of weights* $\mathbf{W}$, $\mathbf{U}$ *and* $\mathbf{B}$ *are used as each element of the sequence is processed. The output layer produces a sequence of predictions* O_ℓ *from the current activation* A_ℓ*, but typically only the last of these,* O_L*, is of relevance. To the left of the equal sign is a concise representation of the network, which is* unrolled *into a more explicit version on the right.*

NN Performance and Success

NN Performance and Success

- Within ML circles, you'll often hear NNs being spoken about as if they are some miracle cure for all of data analysis (same was true of RFs before that, and boosting before that, etc.)

NN Performance and Success

- Within ML circles, you'll often hear NNs being spoken about as if they are some miracle cure for all of data analysis (same was true of RFs before that, and boosting before that, etc.)
- This is simply not true – NNs do ***not*** always work well

NN Performance and Success

- Within ML circles, you'll often hear NNs being spoken about as if they are some miracle cure for all of data analysis (same was true of RFs before that, and boosting before that, etc.)

- This is simply not true – NNs do ***not*** always work well

- **But ...** they have been *incredibly* successful on some problems, far surpassing previous benchmarks.

NN Performance and Success

Let's consider why ...

- What did we say earlier in the lecture about how these NN models compared to most of the other models we've considered in this course?

NN Performance and Success

Let's consider why ...

- What did we say earlier in the lecture about how these NN models compared to most of the other models we've considered in this course? *They seem to be far more complex.*

NN Performance and Success

Let's consider why ...

- What did we say earlier in the lecture about how these NN models compared to most of the other models we've considered in this course? *They seem to be far more complex.*

- So when would we expect them (NN) to work well? Here's what the books says:

  "Typically we expect deep learning to be an attractive choice when the sample size of the training set is extremely large, and when interpretability of the model is not a high priority."

NN Performance and Success

- But they're leaving out probably the most important thing ... what?

NN Performance and Success

- But they're leaving out probably the most important thing ... what?
- What do all of the problems we've looked at today have in common that is *unlike* many of the other problems we've considered in this course?

NN Performance and Success

- But they’re leaving out probably the most important thing ... what?
- What do all of the problems we’ve looked at today have in common that is *unlike* many of the other problems we’ve considered in this course?
- They’re all *very high* signal (high SNR) – in essence, these are “easy" problems if we could only extract everything possible from the inputs/features. Rarely is it difficult, for example, to determine which number is written down or whether a photo contains a picture of a lion or an elephant.

NN Performance and Success

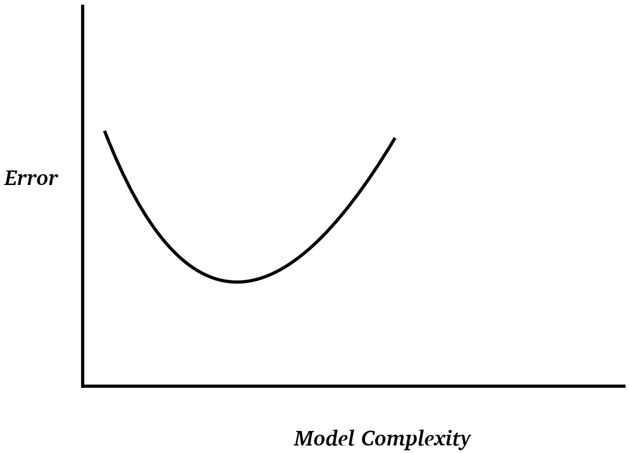
- So we would expect NN – deep NN in particular (many layers/nodes per layer) – to do well when we have both (1) a large sample size and (2) a high amount of signal in the data (high SNR)
- In addition, it's also important to have a *representative* sample. Imagine what a NN would be "learning" if we only included pictures of tigers with their mouths open, or from the left side, for example.
- But there's another reason as well - let's rewind a bit and once again revisit arguably the most fundamental topic in the entire course ... the bias-variance tradeoff

The Double Descent

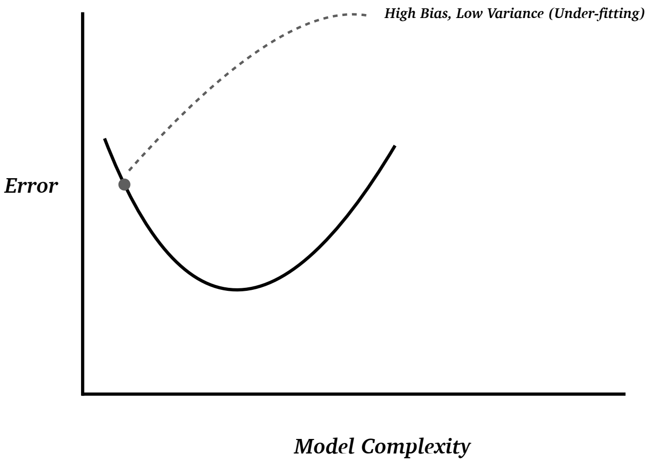
Double Descent

- Throughout the entire course, we've talked about the bias-variance tradeoff. The idea that to get the best fit, we needed to find the right balance between overfitting and underfitting.
- We thought of this is as a kind of “goldilocks" problem – *this model has too much variance, this one has too little, this one is just right*
- In pictures ...

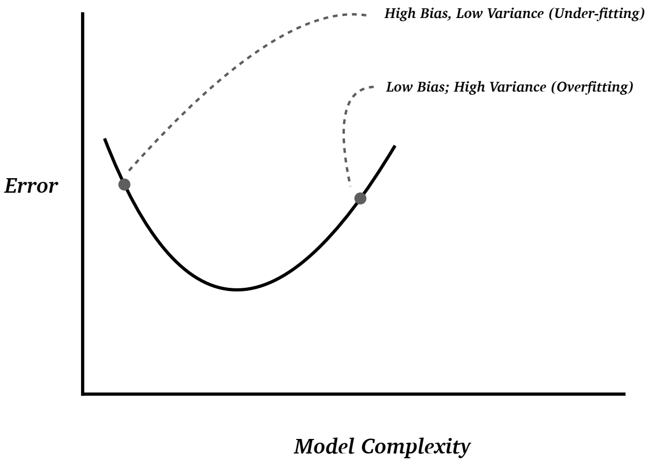
Double Descent



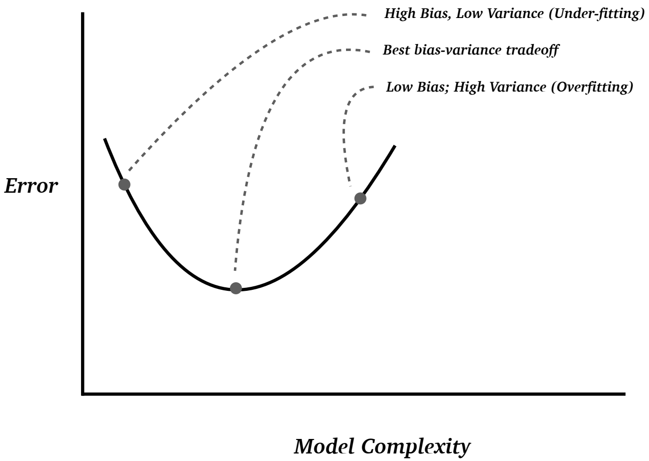
Double Descent



Double Descent


High Bias, Low Variance (Under-fitting)
Low Bias; High Variance (Overfitting)
Error
Model Complexity

Double Descent

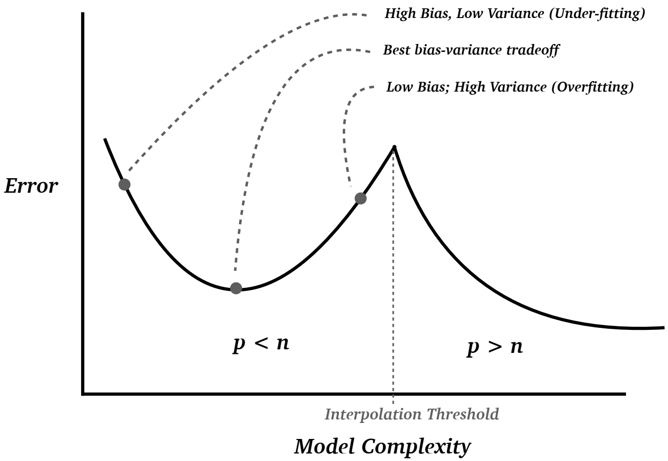


Double Descent

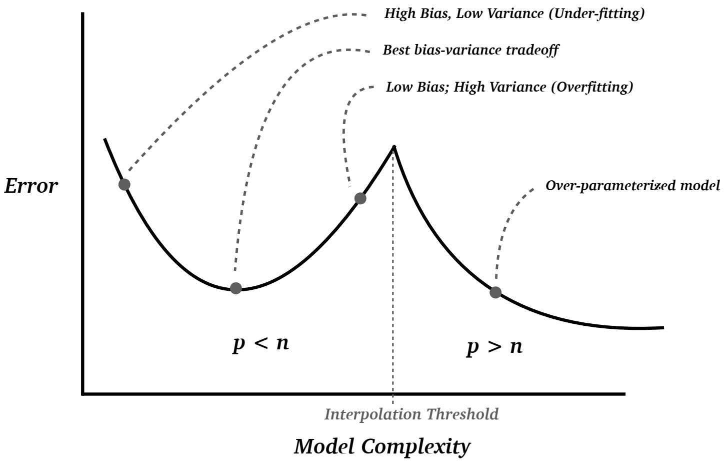
- But recent (very recent; 2019+) research has revealed that *sometimes*, something very interesting can happen beyond the interpolation threshold – the error can begin to come down (*descend*) again

- *Interpolation* just means the model fits the training data perfectly, so the *interpolation threshold* is simply the complexity point at which this begins to happen (think $p = n$)

Double Descent


High Bias, Low Variance (Under-fitting)
Best bias-variance tradeoff
Low Bias; High Variance (Overfitting)
Error
$p < n$
$p > n$
Interpolation Threshold
Model Complexity

Double Descent


High Bias, Low Variance (Under-fitting)
Best bias-variance tradeoff
Low Bias; High Variance (Overfitting)
Over-parameterized model
Error
$p < n$
$p > n$
Interpolation Threshold
Model Complexity

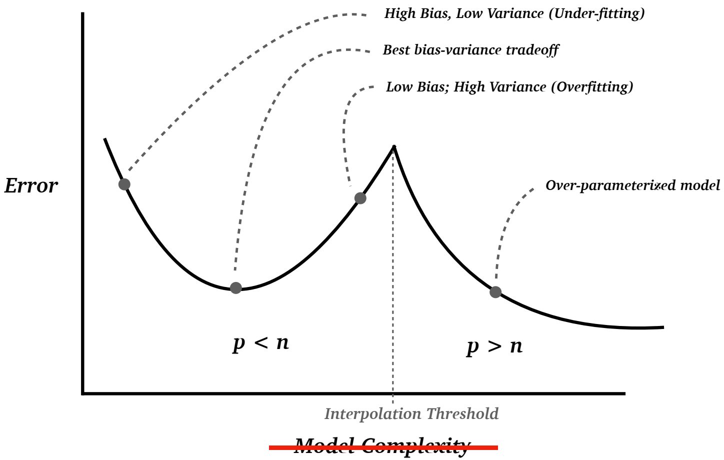
Double Descent

- So what's going on here?

Double Descent

- So what's going on here?
- Throughout the course we've equated "degrees of freedom" with "model complexity". But this was just based on intuition, and this intuition breaks down once the model begins fitting perfectly on the training data (which is all dof looks at).
- Remember: once we get into $p > n$ settings, there are *infinitely many* models that fit well – not just one. So what is shown in plots like this is the "min norm" solution – among all perfect-fitting models, we choose the one with the least variance.
  - As it turns out, the "minimum-variance" model is sometimes less complex than the single unique solution we get in the low-dimensional $p < n$ setting

Double Descent



High Bias, Low Variance (Under-fitting)
Best bias-variance tradeoff
Low Bias; High Variance (Overfitting)
Over-parameterized model
Error
$p < n$
$p > n$
Interpolation Threshold
~~Model Complexity~~

Number of parameters/terms

Double Descent

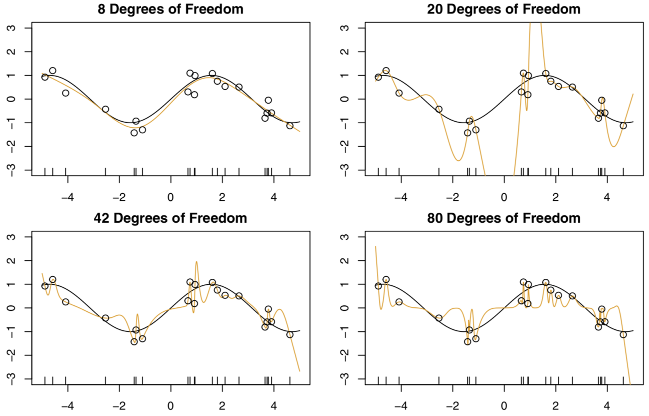


FIGURE 10.21. *Fitted functions $\hat{f}_d(X)$ (orange), true function $f(X)$ (black) and the observed 20 training data points. A different value of d (degrees of freedom) is used in each panel. For $d \geq 20$ the orange curves all interpolate the training points, and hence the training error is zero.*

Double Descent

Very Important: Repeat after me:

Double Descent

Very Important: Repeat after me:

This

Double Descent

Very Important: Repeat after me:

This does

Double Descent

Very Important: Repeat after me:

This does not

Double Descent

Very Important: Repeat after me:

This does not violate

Double Descent

Very Important: Repeat after me:

This does not violate the

Double Descent

Very Important: Repeat after me:

This does not violate the bias-

Double Descent

Very Important: Repeat after me:

This does not violate the bias-variance

Double Descent

Very Important: Repeat after me:

This does not violate the bias-variance tradeoff.

Double Descent

- The bias-variance tradeoff is a mathematical result – not an empirical one. So if this is violated, then either every statistician has gotten the math wrong for the past 100+ years, or there's a hole in the universe. (This does not stop this claim from being made in numerous CS/ML papers.)
- This does not happen with very many kinds of models and does not explain the robust success of other kinds of ML procedures like random forests. (This does not stop this claim from being made in numerous CS/ML papers.)
- But it is very interesting nonetheless and helps explain why NN sometimes achieve very good success – "overfitting, yes, but not by as much as it may seem"
- Still a very early/active research area – much remains to be said