

Neural Network

Yanrong Yang

RSFAS/CBE, Australian National University

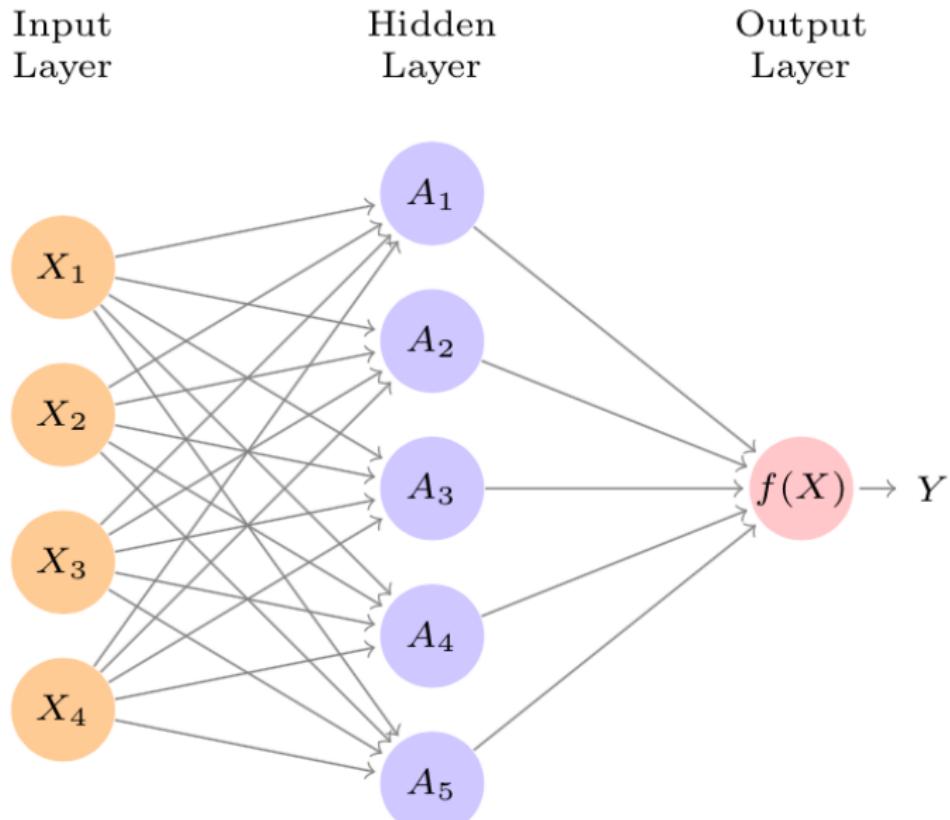
27 September 2022

Contents of this week

- ▶ Basic Neural Network
- ▶ Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN)
- ▶ Double Descent Phenomenon and Interpretation

Basic Neural Network

Single Layer Neural Network



Framework of Feed-Forward Neural Network

- ▶ A neural network takes an input vector of p variables $X = (X_1, X_2, \dots, X_p)$ and builds a nonlinear function $f(X)$ to predict the response Y .
- ▶ The model is

$$f(X) = \beta_0 + \sum_{k=1}^K \beta_k h_k(X) = \beta_0 + \sum_{k=1}^K \beta_k g\left(w_{k0} + \sum_{j=1}^p w_{kj} X_j\right). \quad (1)$$

- ▶ Hidden Layer: K activations $A_k, k = 1, 2, \dots, K$ are

$$A_k = h_k(X) = g\left(w_{k0} + \sum_{j=1}^p w_{kj} X_j\right). \quad (2)$$

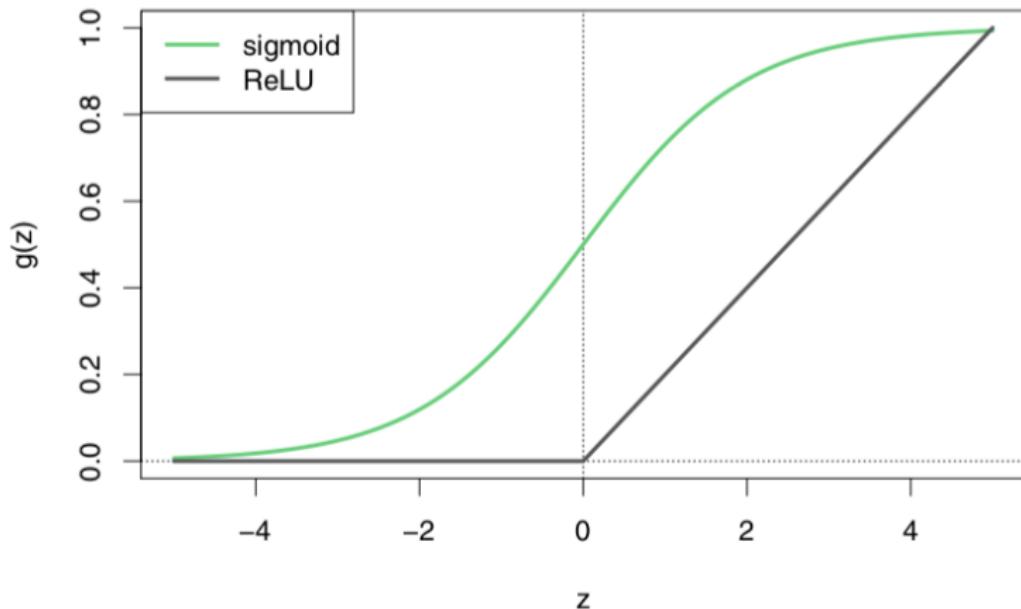
- ▶ The output layer:

$$f(X) = \beta_0 + \sum_{k=1}^K \beta_k A_k. \quad (3)$$

- ▶ In terms of minimizing a loss function, NN recovers the unknown weights parameters. For instance,

$$\sum_{i=1}^n (y_i - f(x_i))^2. \quad (4)$$

Activation Functions



Activation Functions

- ▶ The sigmoid activation function is

$$g(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}. \quad (5)$$

- ▶ The ReLU (rectified linear unit) activation function is

$$g(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases} \quad (6)$$

Nonlinearity of NN

The nonlinearity of NN in the activation function $g(\cdot)$ is essential. Consider an example with $p = 2$ input variables $X = (X_1, X_2)$ and $K = 2$ hidden units $h_1(X)$ and $h_2(X)$ with $g(z) = z^2$.

$$\begin{aligned}\beta_0 &= 0, & \beta_1 &= \frac{1}{4}, & \beta_2 &= -\frac{1}{4}, \\ w_{10} &= 0, & w_{11} &= 1, & w_{12} &= 1, \\ w_{20} &= 0, & w_{21} &= 1, & w_{22} &= -1.\end{aligned}\tag{10.6}$$

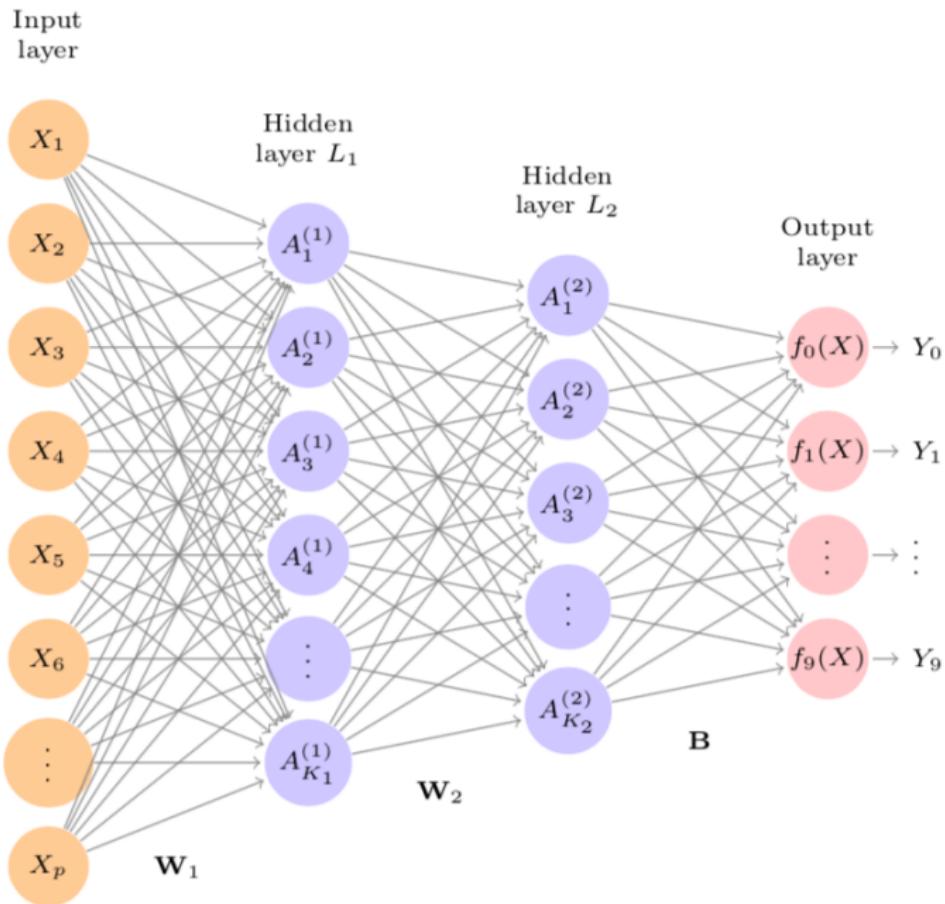
From (10.2), this means that

$$\begin{aligned}h_1(X) &= (0 + X_1 + X_2)^2, \\ h_2(X) &= (0 + X_1 - X_2)^2.\end{aligned}\tag{10.7}$$

Then plugging (10.7) into (10.1), we get

$$\begin{aligned}f(X) &= 0 + \frac{1}{4} \cdot (0 + X_1 + X_2)^2 - \frac{1}{4} \cdot (0 + X_1 - X_2)^2 \\ &= \frac{1}{4} [(X_1 + X_2)^2 - (X_1 - X_2)^2] \\ &= X_1 X_2.\end{aligned}\tag{10.8}$$

Multilayer Neural Network



Formulation of Multilayer NN

- It has two hidden layers L_1 (256 units) and L_2 (128 units) rather than one. Later we will see a network with seven hidden layers.
- It has ten output variables, rather than one. In this case the ten variables really represent a single qualitative variable and so are quite dependent. (We have indexed them by the digit class 0–9 rather than 1–10, for clarity.) More generally, in *multi-task learning* one can predict different responses simultaneously with a single network; they all have a say in the formation of the hidden layers.
- The loss function used for training the network is tailored for the multiclass classification task.

Formulation of Multilayer NN

The first hidden layer is as in (10.2), with

$$\begin{aligned} A_k^{(1)} &= h_k^{(1)}(X) \\ &= g(w_{k0}^{(1)} + \sum_{j=1}^p w_{kj}^{(1)} X_j) \end{aligned} \tag{10.10}$$

for $k = 1, \dots, K_1$. The second hidden layer treats the activations $A_k^{(1)}$ of the first hidden layer as inputs and computes new activations

$$\begin{aligned} A_\ell^{(2)} &= h_\ell^{(2)}(X) \\ &= g(w_{\ell0}^{(2)} + \sum_{k=1}^{K_1} w_{\ell k}^{(2)} A_k^{(1)}) \end{aligned} \tag{10.11}$$

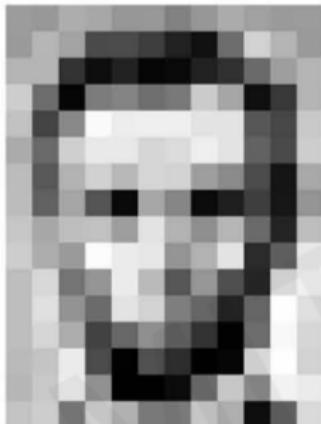
Convolutional Neural Network (CNN)

Aim and Content of CNN

Convolutional Neural Network is a structured neural network and aims to classify images. CNN consists of two parts:

- ▶ Feature Extraction: Convolutional Layers and Pooling Layers
- ▶ Fully Connected Neural Network with extracted features

Representation of Images in Computer



187	163	174	168	160	162	129	151	172	163	155	156
165	182	163	74	76	62	39	17	119	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	239	233	214	220	239	228	98	74	206
188	88	179	209	186	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	144	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	35	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	106	227	210	127	103	36	101	255	224
190	214	173	66	103	143	96	90	2	109	249	215
187	196	226	79	1	81	47	0	6	217	256	211
183	202	237	149	0	0	12	108	200	138	243	236
195	206	123	207	377	121	123	200	176	13	96	218

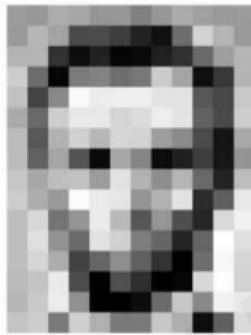
What the computer sees

187	163	174	168	160	162	129	151	172	161	155	156
156	182	163	74	76	62	39	17	119	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	239	233	214	220	239	228	98	74	206
188	88	179	209	186	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	35	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	106	227	210	127	103	36	101	255	224
190	214	173	66	103	143	96	90	2	109	249	215
187	196	226	79	1	81	47	0	6	217	256	211
183	202	237	149	0	0	12	108	200	138	243	236
195	206	123	207	377	121	123	200	176	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

from MIT 6.S191: Introduction to Deep Learning

Computer Task for Images



Input Image



157	159	174	166	160	152	129	181	172	161	158	196
150	182	143	74	76	62	33	17	110	210	180	154
180	180	90	14	94	6	10	33	48	106	158	181
206	109	6	126	131	111	120	204	166	15	56	180
154	68	137	261	237	239	239	228	227	87	71	201
172	105	207	233	233	214	223	230	228	98	74	205
188	66	179	209	195	216	211	159	76	38	169	169
189	87	165	64	56	168	134	11	31	62	22	140
199	168	191	189	198	227	119	142	182	106	36	190
206	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	46	150	79	38	218	241
190	224	147	106	227	218	127	102	36	101	220	224
190	214	173	66	110	143	96	50	2	109	248	216
187	196	236	76	1	61	47	0	6	217	295	211
189	203	237	149	0	0	12	108	200	138	243	236
195	206	129	207	177	121	123	200	175	12	96	218

Pixel Representation

classification

Lincoln

0.8

Washington

0.1

Jefferson

0.05

Obama

0.05

- **Regression:** output variable takes continuous value
- **Classification:** output variable takes class label. Can produce probability of belonging to a particular class

from MIT 6.S191: Introduction to Deep Learning

Feature Detection

The major challenge of image processing is feature detection.

Let's identify key features in each image category



Nose,
Eyes,
Mouth



Wheels,
License Plate,
Headlights



Door,
Windows,
Steps

Manual Feature Extraction



from MIT 6.S191: Introduction to Deep Learning

Manual Feature Extraction

The essential challenge of feature learning is to find features that are able to classify images.

Domain knowledge

Define features

Detect features
to classify

Viewpoint variation



Scale variation



Deformation



Occlusion



Illumination conditions



Background clutter

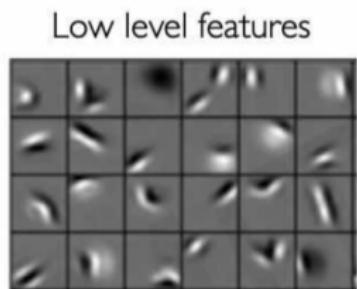


Intra-class variation

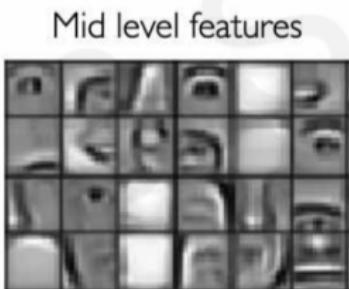


Feature Learning

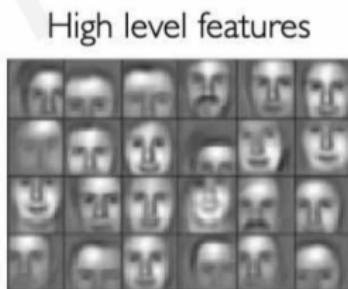
Can we learn a **hierarchy of features** directly from the data instead of hand engineering?



Edges, dark spots



Eyes, ears, nose

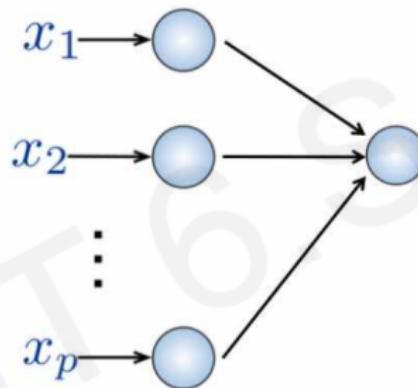


Facial structure

from MIT 6.S191: Introduction to Deep Learning

Failure of Fully Connected Neural Network

- Input:**
- 2D image
 - Vector of pixel values



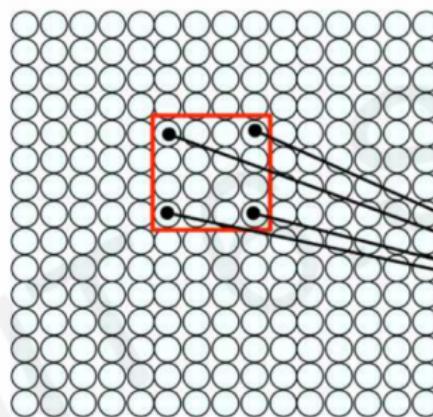
- Fully Connected:**
- Connect neuron in hidden layer to all neurons in input layer
 - No spatial information!
 - And many, many parameters!

How can we use **spatial structure** in the input to inform the architecture of the network?

from MIT 6.S191: Introduction to Deep Learning

CNN: Imposing Spatial Structure

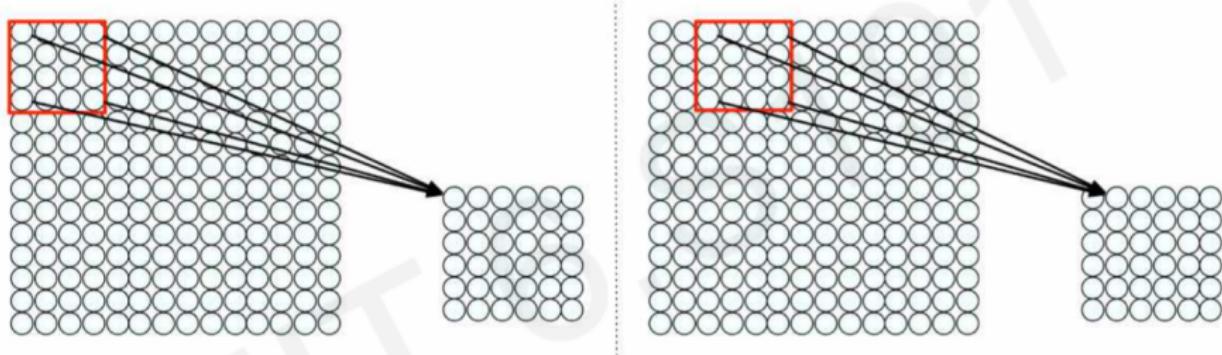
Input: 2D image.
Array of pixel values



Idea: connect patches of input
to neurons in hidden layer.
Neuron connected to region of
input. Only "sees" these values.

from MIT 6.S191: Introduction to Deep Learning

CNN: Imposing Spatial Structure



Connect patch in input layer to a single neuron in subsequent layer.

Use a sliding window to define connections.

How can we **weight** the patch to detect particular features?

from MIT 6.S191: Introduction to Deep Learning

Feature Learning in CNN

- 1) Apply a set of weights – a filter – to extract **local features**
- 2) Use **multiple filters** to extract different features
- 3) Spatially **share** parameters of each filter
(features that matter in one part of the input should matter elsewhere)

from MIT 6.S191: Introduction to Deep Learning

Illustration of Convolution Operation

Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter:

The diagram shows a 5x5 input image and a 3x3 filter. The image has values 1, 1, 1, 0, 0; 0, 1, 1, 1, 0; 0, 0, 1, 1, 1; 0, 0, 1, 1, 0; 0, 1, 1, 0, 0. The filter has values 1, 0, 1; 0, 1, 0; 1, 0, 1. A circled multiplication symbol (\otimes) indicates element-wise multiplication between the overlapping regions of the image and the filter as it slides across the image.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

\otimes

1	0	1
0	1	0
1	0	1

filter

image

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs...

from MIT 6.S191: Introduction to Deep Learning

Illustration of Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:

The diagram illustrates a convolution operation. On the left is an input image represented as a 5x5 grid of numbers. The first three columns of the input are highlighted in orange, and the last two columns are green. The first three rows of the input are highlighted in orange, and the last two rows are green. The last row and column of the input are entirely green. To the right of the input is a multiplication symbol (\otimes). Next to it is a 3x3 grid labeled "filter", which contains the numbers 1, 0, 1; 0, 1, 0; and 1, 0, 1. To the right of the filter is an equals sign (=). Finally, to the right of the equals sign is a 3x3 grid labeled "feature map". Only the top-left cell of the feature map is filled with the number 4, while all other cells are empty.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

\otimes

1	0	1
0	1	0
1	0	1

=

4		

filter

feature map

from MIT 6.S191: Introduction to Deep Learning

Illustration of Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:

The diagram illustrates a convolution operation. On the left is an input image represented as a 5x5 grid of green cells. Some cells contain numerical values (1, 0, 1, 0, 0) and some contain smaller yellow cells with multipliers (e.g., $\times 1$, $\times 0$). In the center is a 3x3 filter represented as a grid of yellow cells containing the values 1, 0, 1; 0, 1, 0; and 1, 0, 1. To the right of the filter is an equals sign followed by a 3x3 feature map. The feature map has two red cells containing the values 4 and 3, and three white cells.

1	1×1	1×0	0	$\times 1$	0
0	1×0	1×1	1×0	0	
0	0	$\times 1$	1×0	1×1	1
0	0	1	1	0	
0	1	1	0	0	

\otimes filter = feature map

from MIT 6.S191: Introduction to Deep Learning

Illustration of Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:

The diagram illustrates a convolution operation. On the left is an input image represented as a 5x5 grid of green squares. The values in the grid are: Row 1: 1, 1, 1, 0, 0; Row 2: 0, 1, 1, 1, 0; Row 3: 0, 0, 1, 1, 1; Row 4: 0, 0, 1, 1, 0; Row 5: 0, 1, 1, 0, 0. The value 1 is highlighted in red in the third row, second column. The value 0 is highlighted in red in the fourth row, first column. The value 1 is highlighted in red in the fifth row, second column. The value 1 is highlighted in red in the third row, third column. The value 1 is highlighted in red in the fourth row, second column. The value 0 is highlighted in red in the fifth row, third column. In the center is a 3x3 filter represented as a grid of yellow squares. The values in the filter are: Row 1: 1, 0, 1; Row 2: 0, 1, 0; Row 3: 1, 0, 1. All values in the filter are highlighted in red. To the right of the filter is an equals sign (=). To the right of the equals sign is a 3x3 feature map represented as a grid of pink squares. The values in the feature map are: Row 1: 4, 3, 4; Row 2: 2, 4, 3; Row 3: 2, 3, 4. All values in the feature map are highlighted in red.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

\otimes

1	0	1
0	1	0
1	0	1

=

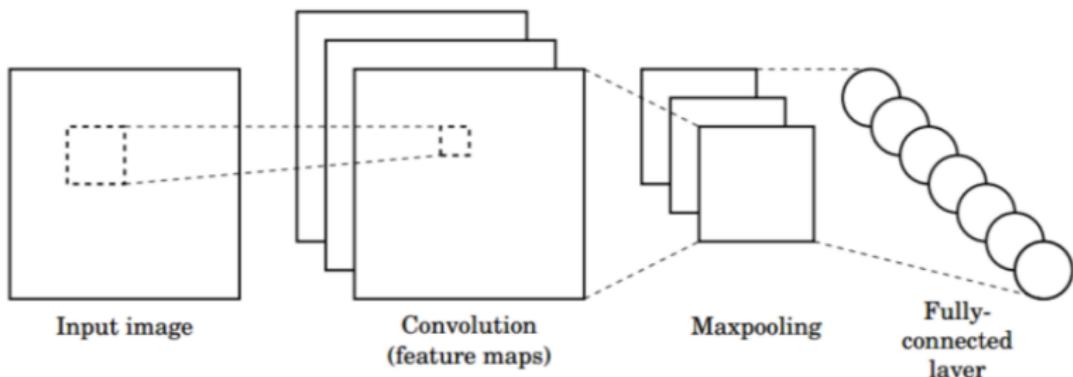
4	3	4
2	4	3
2	3	4

filter

feature map

from MIT 6.S191: Introduction to Deep Learning

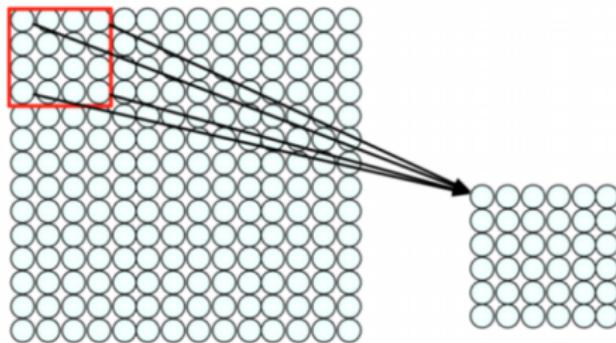
Summary of Feature Learning in CNN



- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

**Train model with image data.
Learn weights of filters in convolutional layers.**

Formula of Convolution Operation



4x4 filter: matrix
of weights θ_{ij}

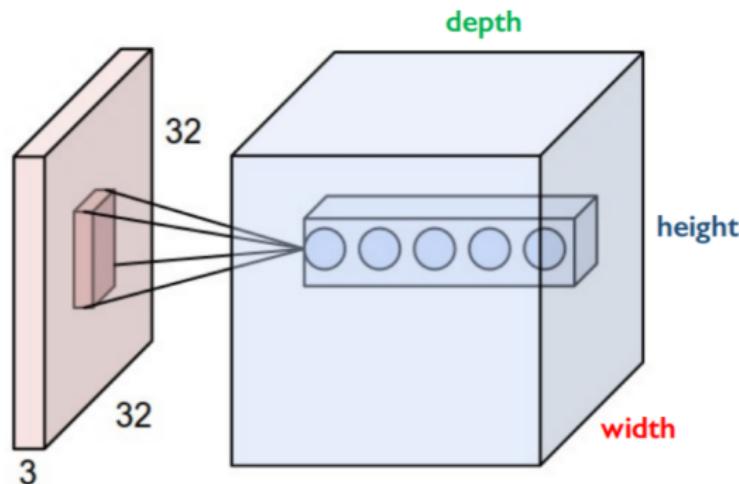
$$\sum_{i=1}^4 \sum_{j=1}^4 \theta_{ij} x_{i+p, j+q} + b$$

for neuron (p,q) in hidden layer

applying a window of weights
computing linear combinations
activating with non-linear function

from MIT 6.S191: Introduction to Deep Learning

Spatial Positions of Neurons



Layer Dimensions:

$h \times w \times d$

where h and w are spatial dimensions
 d (depth) = number of filters

Stride:

Filter step size

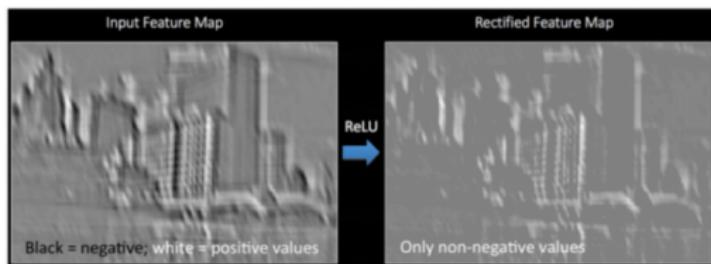
Receptive Field:

Locations in input image that
a node is path connected to

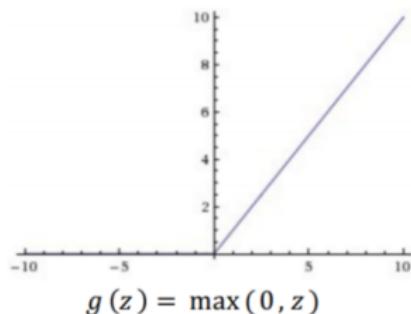
from MIT 6.S191: Introduction to Deep Learning

Nonlinearity via Activation Function

- Apply after every convolution operation (i.e., after convolutional layers)
- ReLU: pixel-by-pixel operation that replaces all negative values by zero. **Non-linear operation**

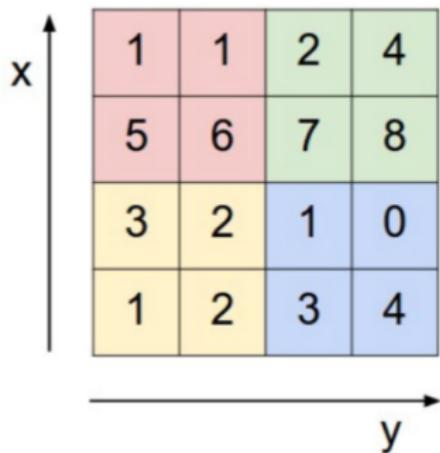


Rectified Linear Unit (ReLU)



from MIT 6.S191: Introduction to Deep Learning

Max Pooling



max pool with 2x2 filters
and stride 2

A 2x2 output grid resulting from max pooling with 2x2 filters and stride 2. The grid contains the following values:

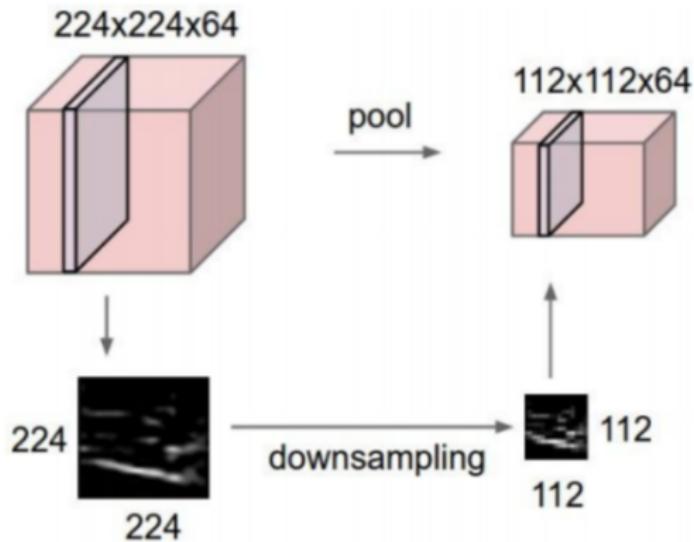
x\y	1	2
1	6	8
2	3	4

- 1) Reduced dimensionality
- 2) Spatial invariance

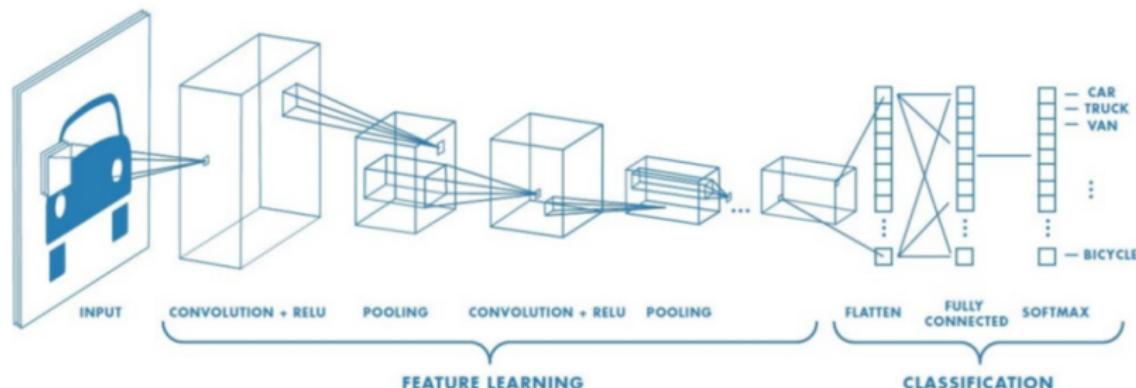
How else can we downsample and preserve spatial invariance?

Pooling Layers

- makes the representations smaller and more manageable
- operates over each activation map independently:



Flowchart of CNN



Learn weights for convolutional filters and fully connected layers

Backpropagation: cross-entropy loss

$$J(\theta) = \sum_i y^{(i)} \log(\hat{y}^{(i)})$$

from MIT 6.S191: Introduction to Deep Learning

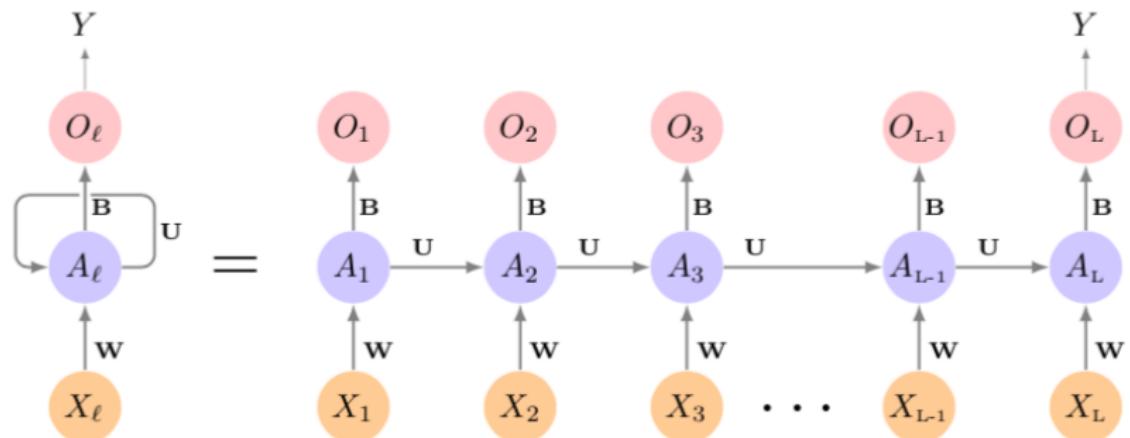
Recurrent Neural Network (RNN)

Motivation of RNN

RNN is designed for data being sequential in nature.

- ▶ **Documents** such as book and movie reviews, newspaper articles. The sequence of words can be exploited in tasks such as topic classification, sentiment analysis and language translation.
- ▶ **Time Series** of temperature, rainfall and so on. The aim is to forecast the weather several days ahead.
- ▶ **Financial Time Series**: market indices, trading volumes, stock and bond prices, and exchange rates. Accurate prediction is difficult for financial time series.
- ▶ **Sound Recordings** like recorded speech and musical recording. Aims may be to give a text transcription of a speech, or assess the quality of a piece of music.
- ▶ **Handwriting**, such as doctors' notes, and handwritten digits such as zip codes. Aim is to turn the handwriting into digital text, or read the digits.

Flowchart of RNN



Framework of RNN

- ▶ The input X is a sequence, i.e. $X = \{X_1, X_2, \dots, X_L\}$. Each X_ℓ is a vector, i.e. $X_\ell = (X_{\ell 1}, X_{\ell 2}, \dots, X_{\ell p})^\top$.
- ▶ The hidden layer consists of K units
 $A_\ell = (A_{\ell 1}, A_{\ell 2}, \dots, A_{\ell K})^\top$.
- ▶ Input-layer Weights: $\mathbf{W} = (w_{kj})_{K \times (p+1)}$; Hidden-to-hidden layer weights: $\mathbf{U} = (u_{ks})_{K \times K}$; the output layer:
 $\mathbf{B} = (\beta_1, \beta_2, \dots, \beta_{K+1})^\top$.
- ▶ The hidden layer:
$$A_{\ell k} = g \left(w_{k0} + \sum_{j=1}^p w_{kj} X_{\ell j} + \sum_{s=1}^K u_{ks} A_{\ell-1, s} \right).$$
- ▶ The output O_ℓ is $O_\ell = \beta_0 + \sum_{k=1}^K \beta_k A_{\ell k}$.
- ▶ The loss function is $(Y - O_L)^2$.

Optimization in RNN

With n input sequence pairs (x_i, y_i) , the shared parameters $\mathbf{W}, \mathbf{U}, \mathbf{B}$ are found by minimizing the sum of squares

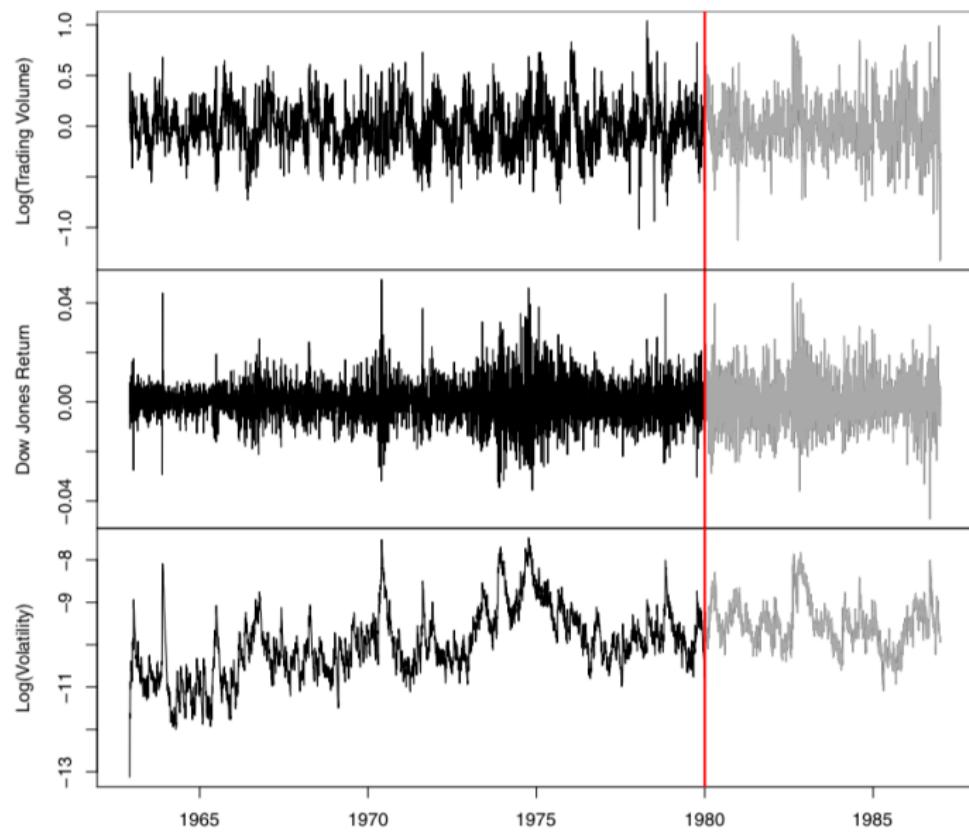
$$\sum_{i=1}^n (y_i - o_{iL})^2 = \sum_{i=1}^n \left(y_i - \left(\beta_0 + \sum_{k=1}^K \beta_k g \left(w_{k0} + \sum_{j=1}^p w_{kj} x_{iLj} + \sum_{s=1}^K u_{ks} a_{i,L-1,s} \right) \right) \right)^2.$$

Example: Financial Time Series Forecasting

Figure 10.14 shows historical trading statistics from the New York Stock Exchange. Shown are three daily time series covering the period December 3, 1962 to December 31, 1986:¹⁹

- **Log trading volume.** This is the fraction of all outstanding shares that are traded on that day, relative to a 100-day moving average of past turnover, on the log scale.
- **Dow Jones return.** This is the difference between the log of the Dow Jones Industrial Index on consecutive trading days.
- **Log volatility.** This is based on the absolute values of daily price movements.

Example: Financial Time Series Forecasting (Figure 10.14)



RNN Forecaster

- ▶ On day t , v_t : log trading volume; r_t : Dow Jones return; z_t : log volatility.
- ▶ Data: a total of $T = 6051$ such triples.
- ▶ Aim: to predict a value v_t from past values v_{t-1}, v_{t-2}, \dots , and past values of the other series r_{t-1}, r_{t-2}, \dots and z_{t-1}, z_{t-2}, \dots
- ▶ The input sequence $X = \{X_1, X_2, \dots, X_L\}$ and output Y are defined as follows.

$$X_1 = \begin{pmatrix} v_{t-L} \\ r_{t-L} \\ z_{t-L} \end{pmatrix}, X_2 = \begin{pmatrix} v_{t-L+1} \\ r_{t-L+1} \\ z_{t-L+1} \end{pmatrix}, \dots, X_L = \begin{pmatrix} v_{t-1} \\ r_{t-1} \\ z_{t-1} \end{pmatrix}, \text{ and } Y = v_t.$$

- ▶ Here, $L = 5$, $K = 12$ hidden units using 4281 training data and then forecast the 1770 values of v_t . We achieve $R^2 = 0.42$. The baseline model can attain $R^2 = 0.18$.

Example: RNN Forecast

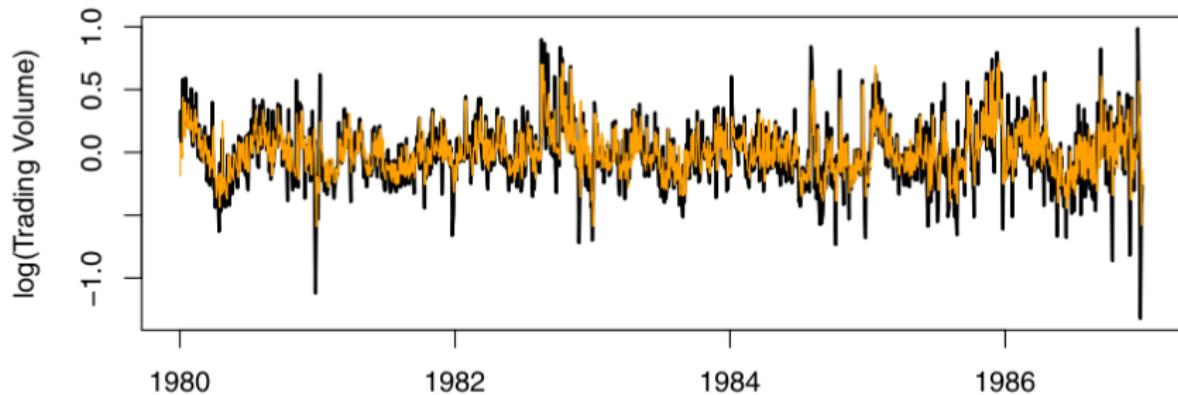
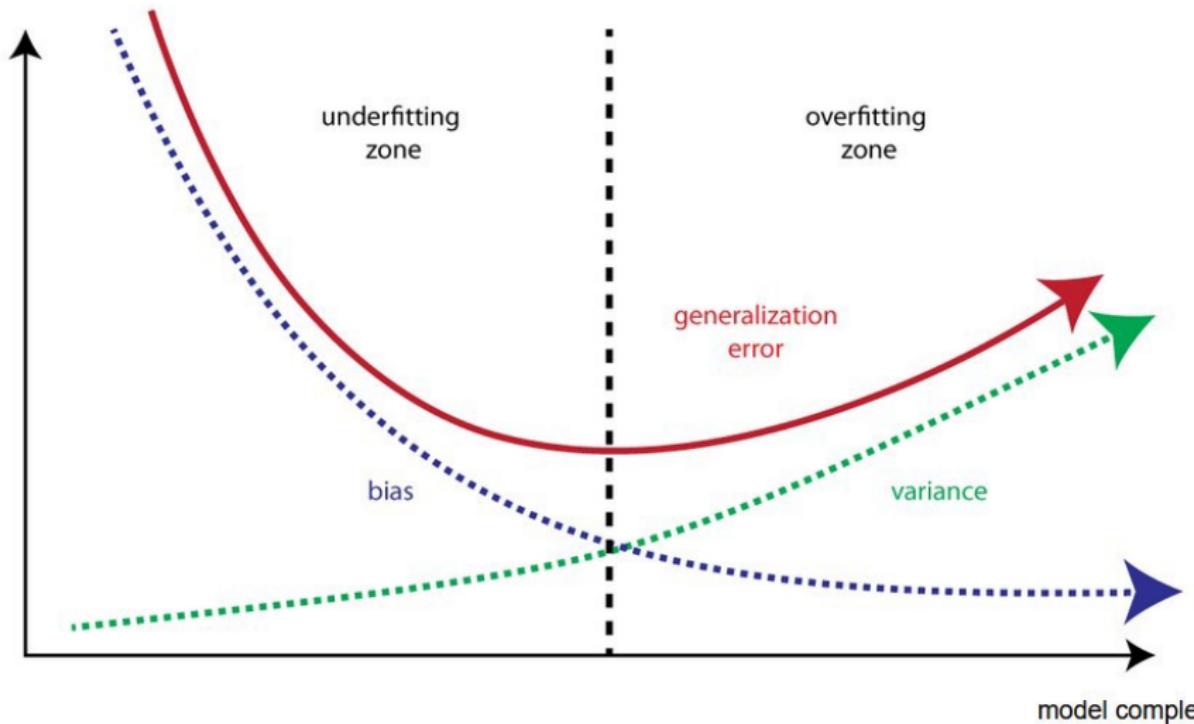


FIGURE 10.16. RNN forecast of `log_volume` on the NYSE test data. The black lines are the true volumes, and the superimposed orange the forecasts. The forecasted series accounts for 42% of the variance of `log_volume`.

Double Descent Phenomenon

Model Flexibility and Prediction Accuracy

the bias vs. variance trade-off



Insights from Bias-Variance Tradeoff

- ▶ This trade-off indicates that statistical learning methods tend to perform the best, in terms of test error, for an intermediate level of model complexity.
- ▶ So one implication of the bias-variance trade-off is that it is generally not a good idea to interpolate the training data, that is, to get zero training error will result in very high test error.

Example: Basis Expansion

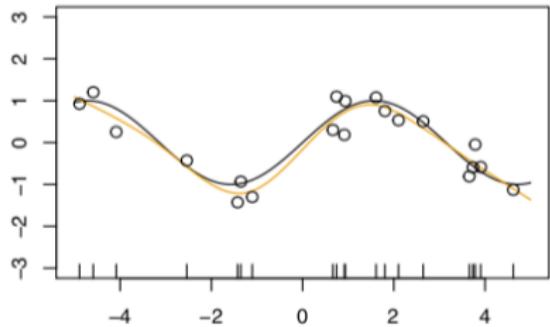
- ▶ The regression model

$$y_i = \sin(x_i) + \varepsilon_i, \quad i = 1, 2, \dots, n. \quad (8)$$

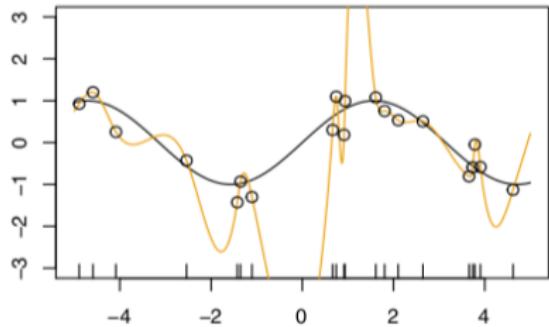
- ▶ The sample size $n = 20$.
- ▶ Predictors $x_i \sim U[-5, 5]$ and errors $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = 0.3$.
- ▶ Fit a natural spline with d basis functions $B_\ell(x_i), \ell = 1, 2, \dots, d$.
- ▶ The minimum-norm estimator for the linear coefficient in Basis Expansion is $(\mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B}\mathbf{Y}$, where $\mathbf{B} = (B_\ell(x_i))_{d \times n}$ and $\mathbf{Y} = (y_1, y_2, \dots, y_n)^\top$.

Fitted Functions

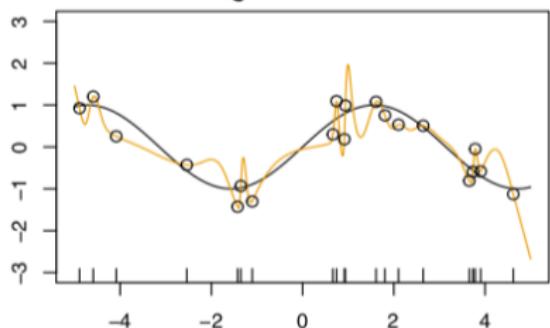
8 Degrees of Freedom



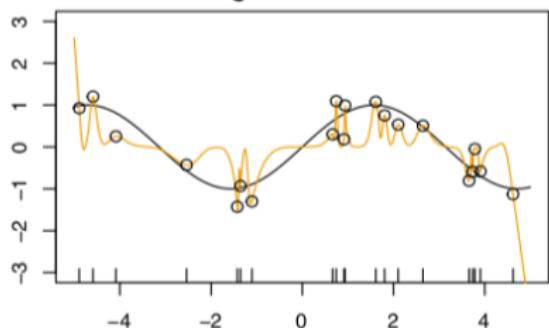
20 Degrees of Freedom



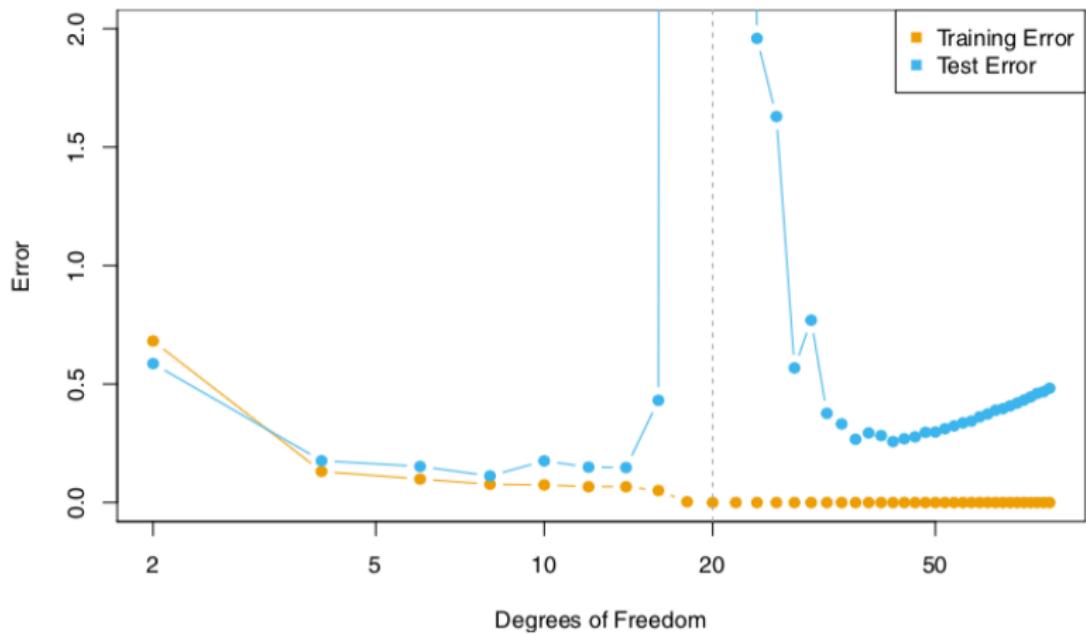
42 Degrees of Freedom



80 Degrees of Freedom



Double Descent Phenomenon



Insights from Double Descent

- ▶ Interpolating the training data perform well, or at least, better than a slightly less complex model that does not quite interpolate the data.
- ▶ Double descent gets its name from the fact that the test error has a U-shape before the interpolation threshold is reached, and then it descends again as an increasingly flexible model is fit.
- ▶ The double descent phenomenon does not contradict the bias-variance trade-off.