Recap

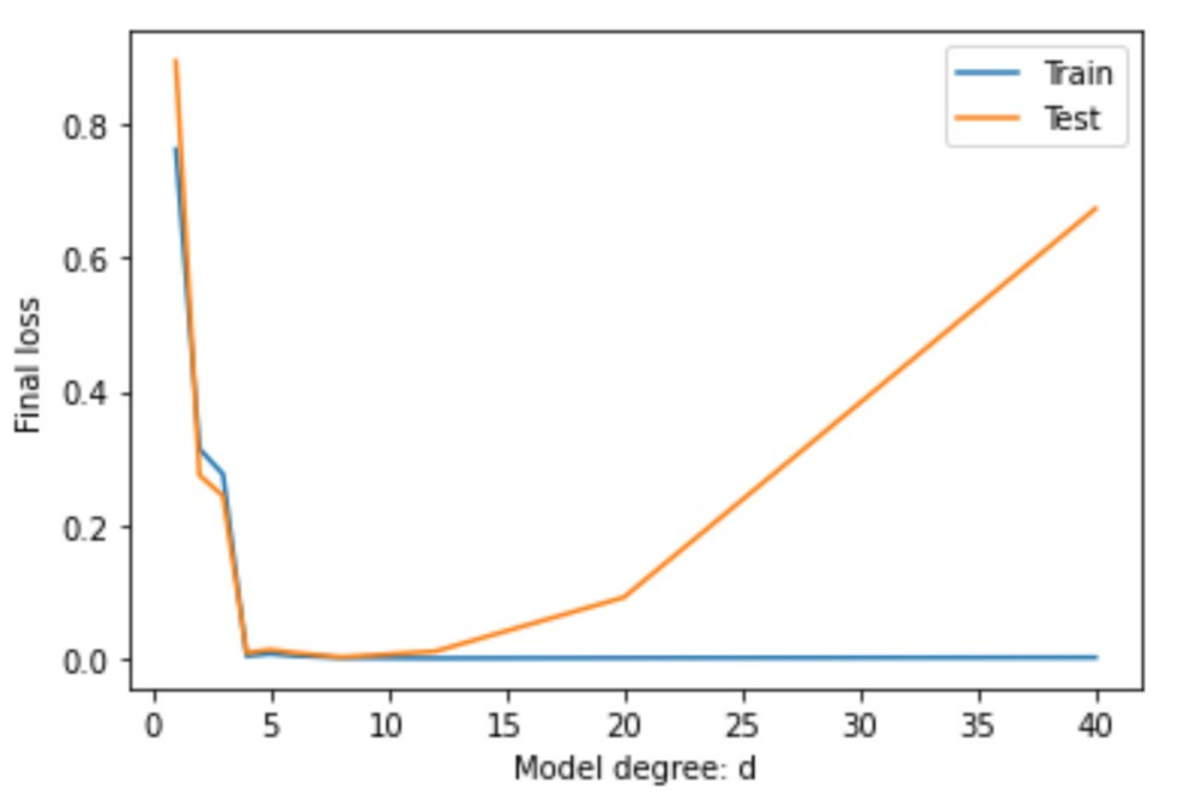
1. Introduction to PyTorch and tensor manipulations

2. Automatic differentiation and gradient-based optimization

3. Advanced optimization methods

Session plan

1. Polynomial regression/fitting
2. Training-testing splits, underfitting/overfitting, model complexity
3. Neural networks: theory and basics

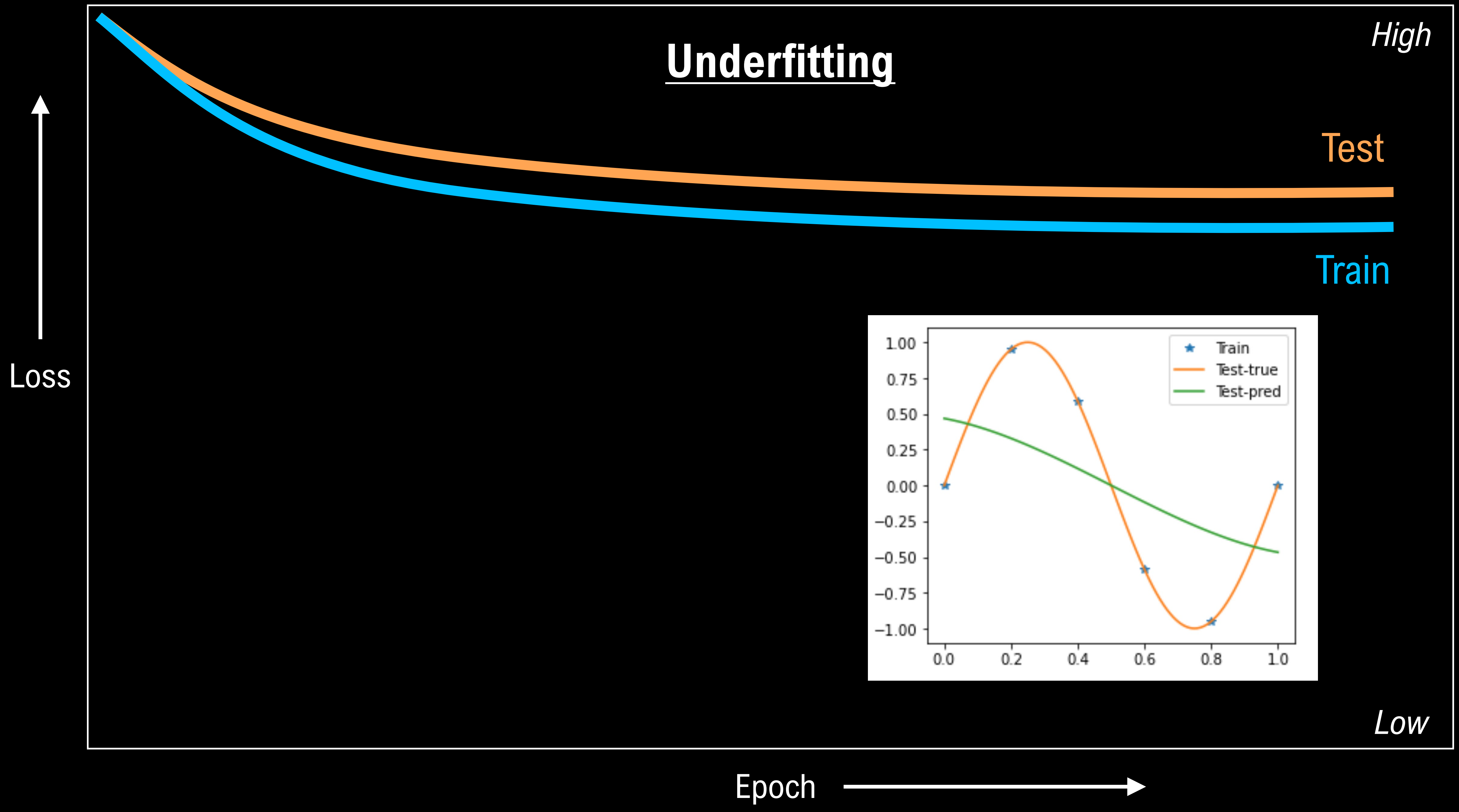


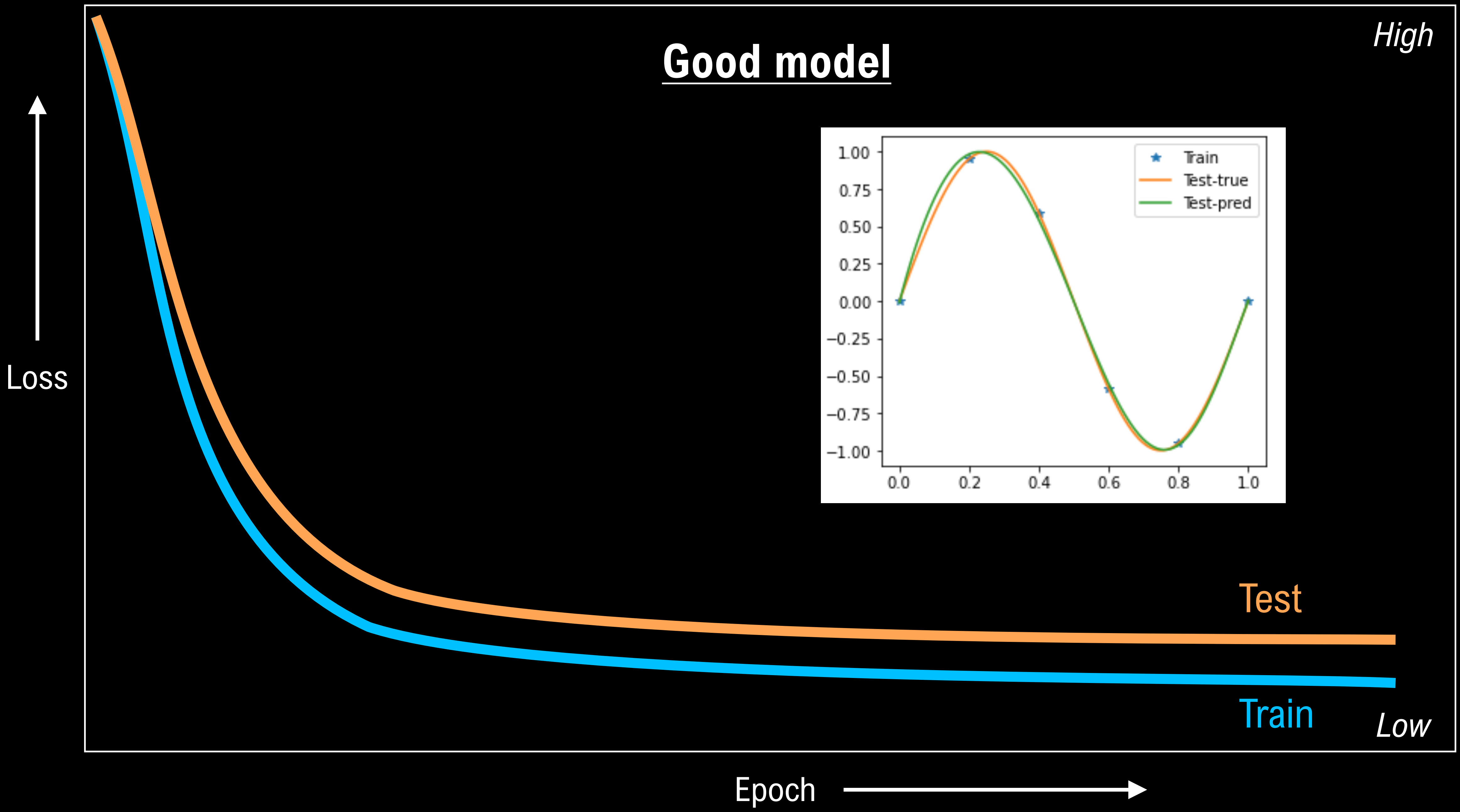
Why did test loss start increasing with model complexity?

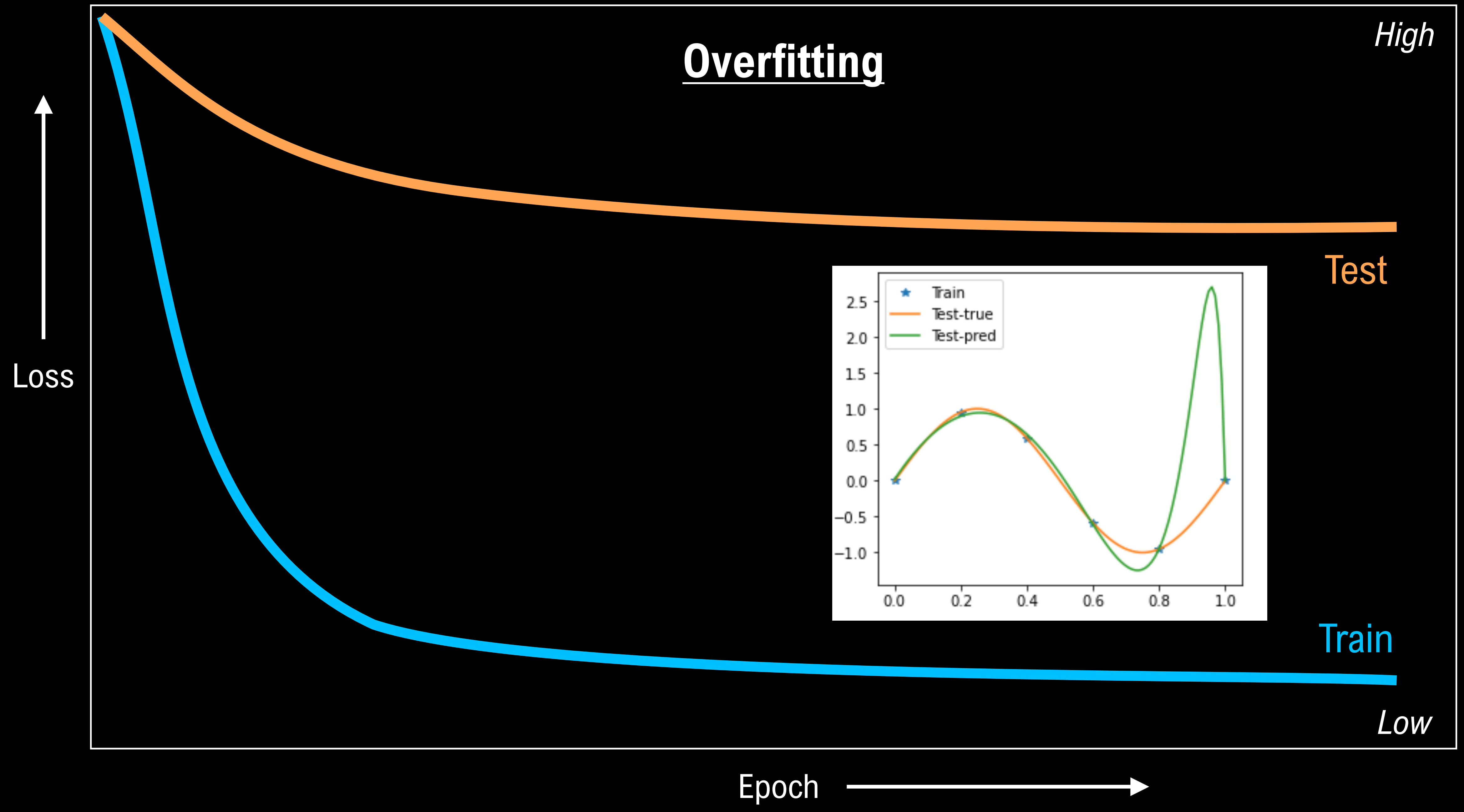
- Because, the model keeps minimizng the train loss at the expense of test data (which it never gets to see)

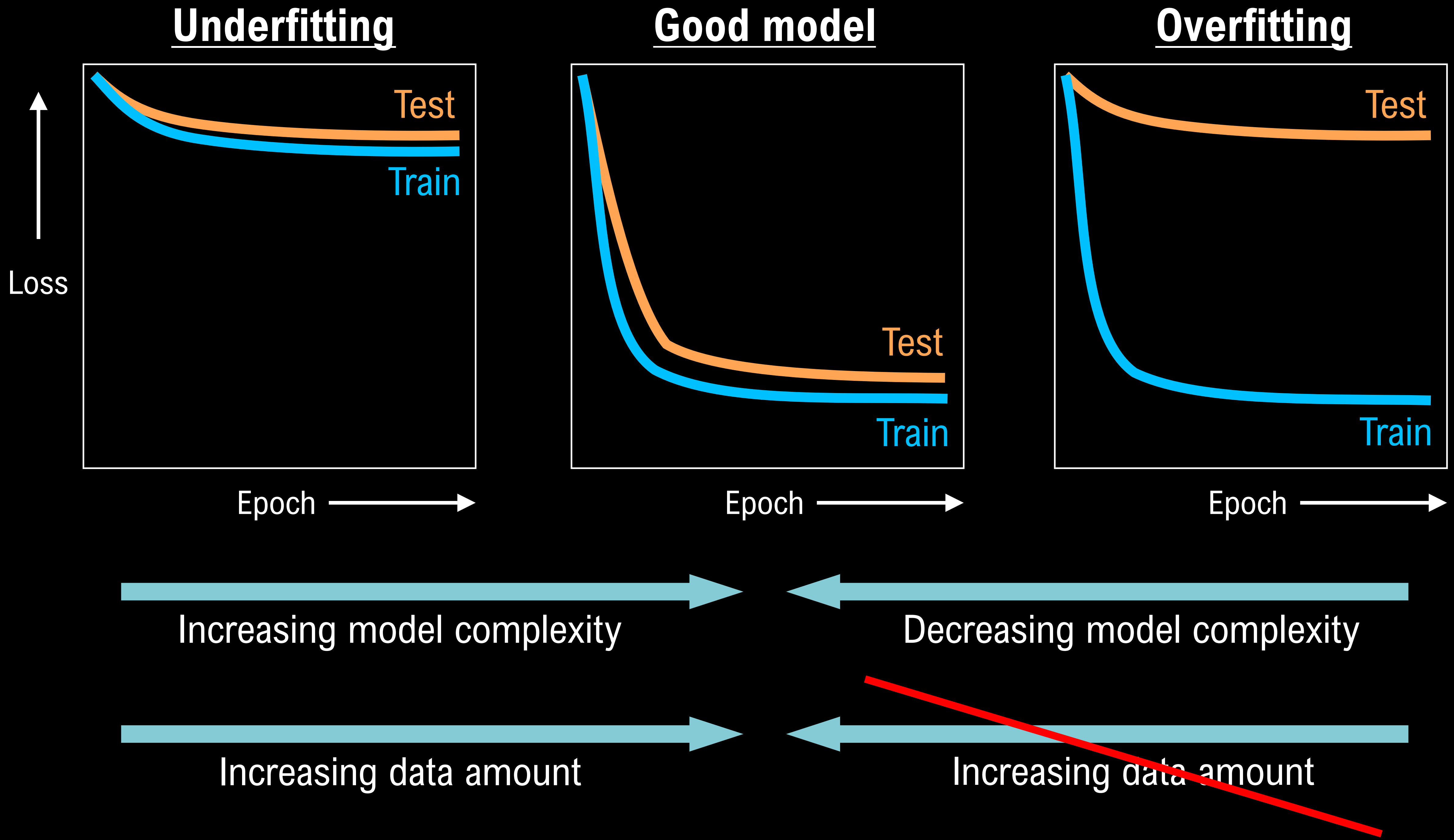
Why increasing data amount won't solve overfitting?

- Because, training loss is already quite small in overfitting, it's the test loss that is high

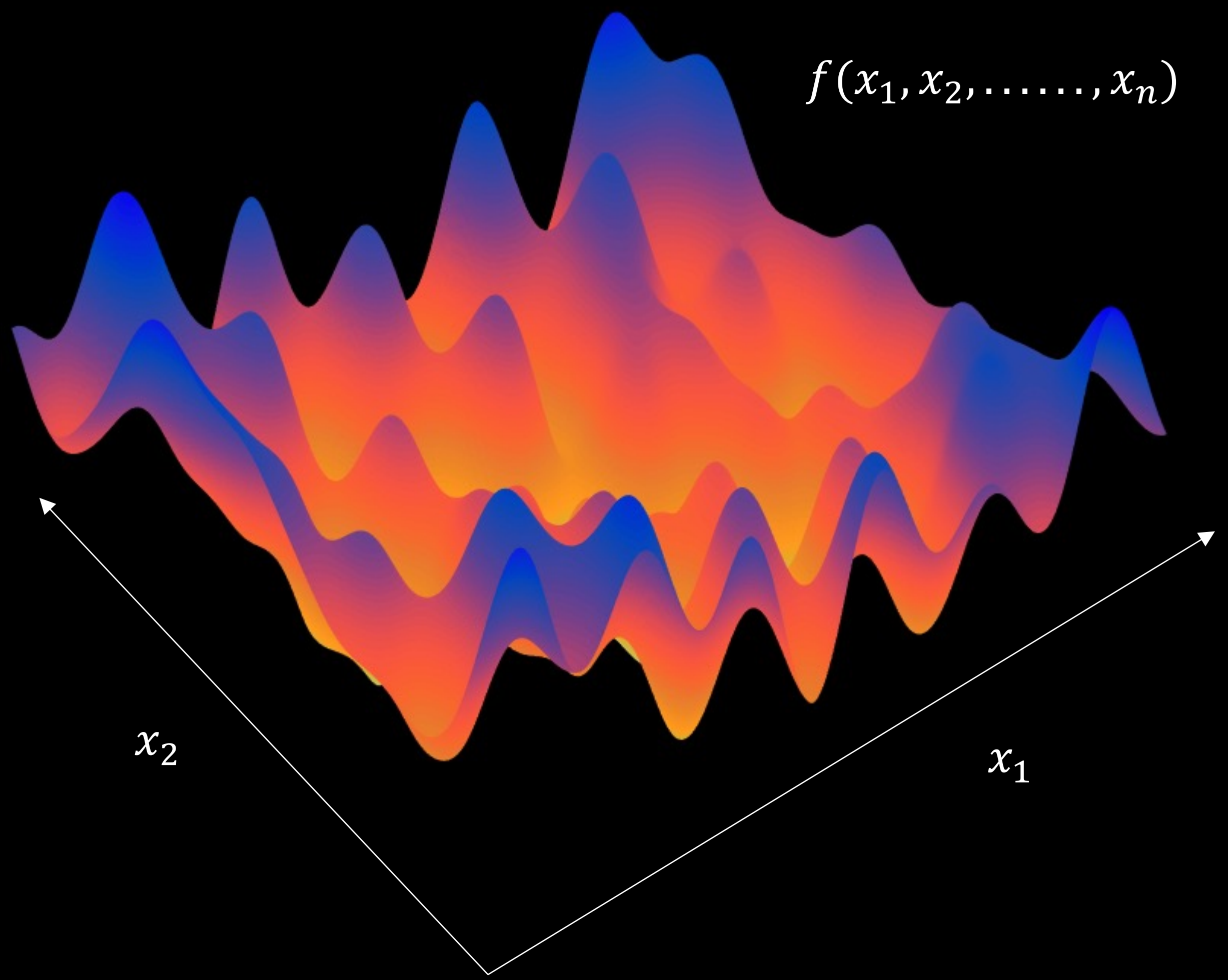

Underfitting
High
Low
Test
Train
Loss
Epoch
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0
Train
Test-true
Test-pred


Good model
High
Loss
Train
Test-true
Test-pred
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0
Test
Train
Low
Epoch


Overfitting
High
Loss
Test
Train
Low
Epoch
Train
Test-true
Test-pred
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0


Underfitting
Good model
Overfitting
Loss
Test
Train
Test
Train
Test
Train
Epoch
Epoch
Epoch
Increasing model complexity
Decreasing model complexity
Increasing data amount
~~Increasing data amount~~

When polynomial regression is not enough?



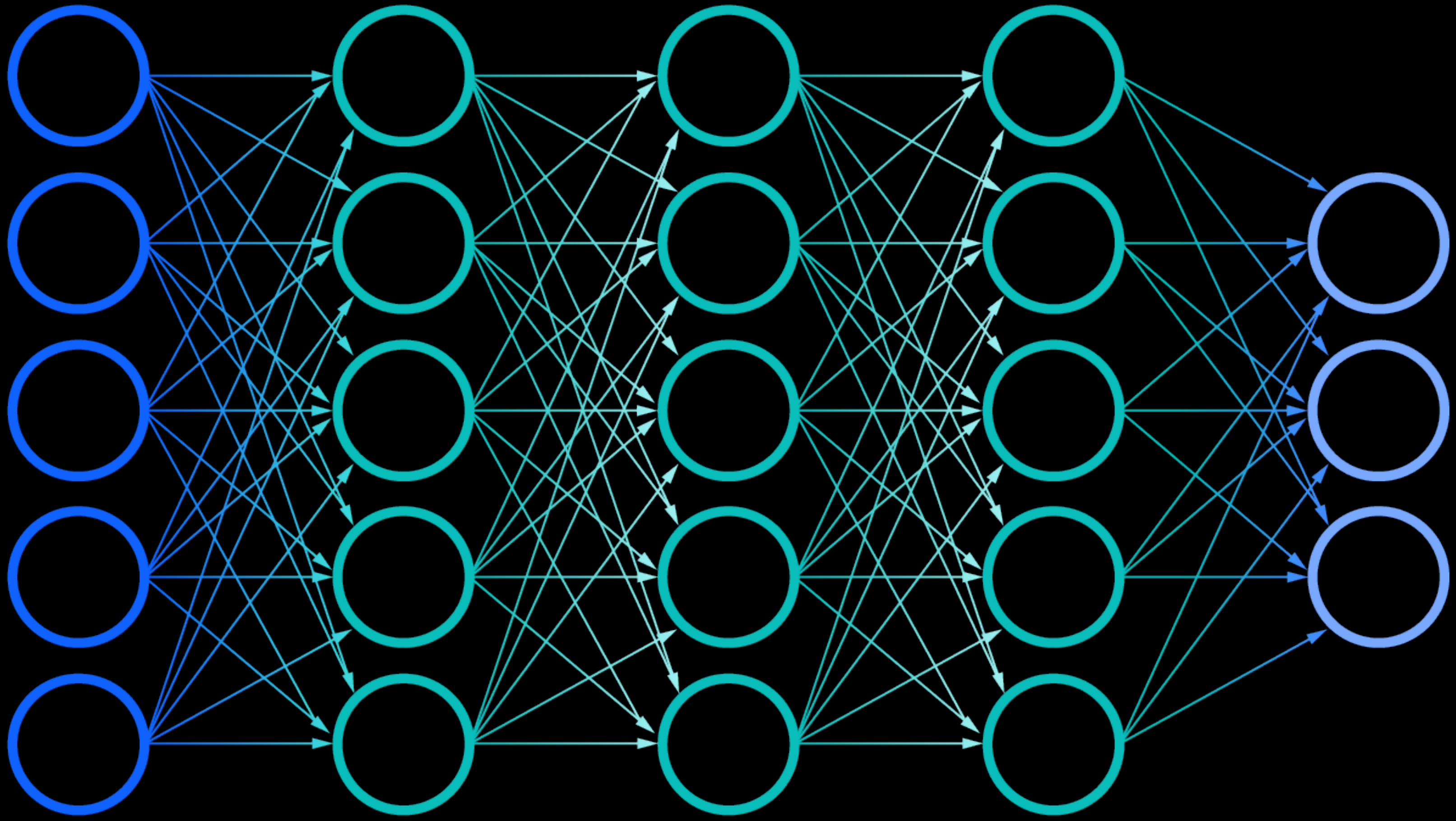
Highly nonlinear data

. . . even when $d = 100$ may not be enough

High-dimensional data

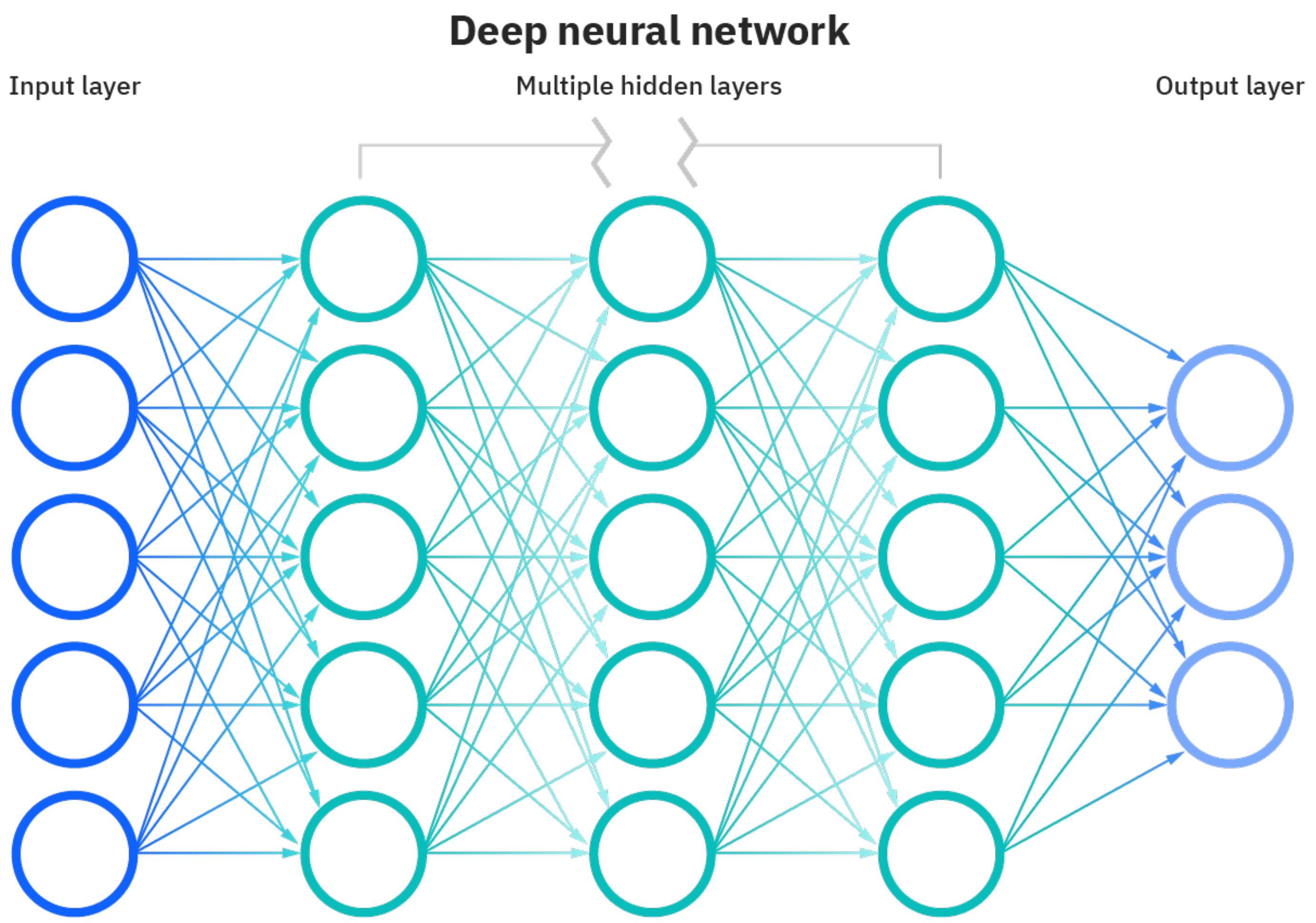
. . . n^d terms in polynomial (computational intractable)

Neural networks to the rescue . . .



Quick 5-minute break

Neural networks to the rescue . . .



Data

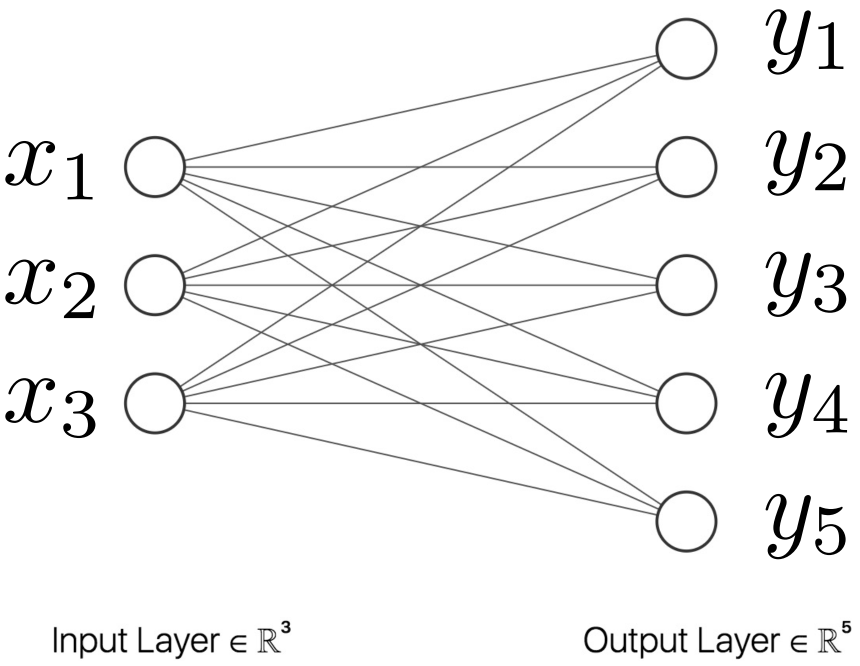
x_1	x_2	x_3	…
.	.	.	
.			
.			

Input / features

y_1	y_2	y_3	…
.	.	.	
.			
.			

Output / labels

A neural network layer *. . . is just a linear transform*

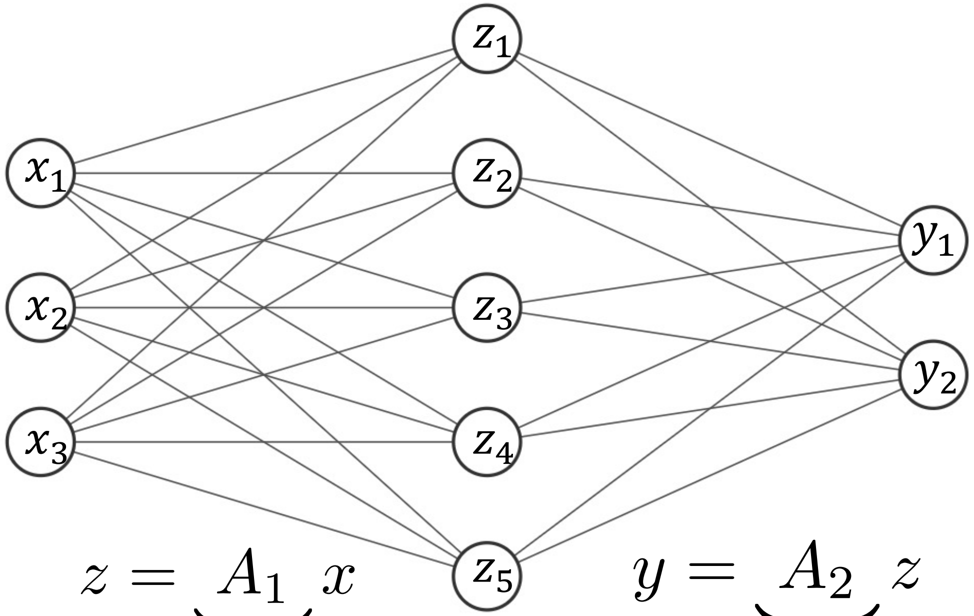


$$y = Ax$$

- For each output node, there are 3 incoming edges
- Each edge denotes entries of transformation matrix A : 5x3 matrix

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \\ A_{51} & A_{52} & A_{53} \end{pmatrix}$$

Layers of linear transformations

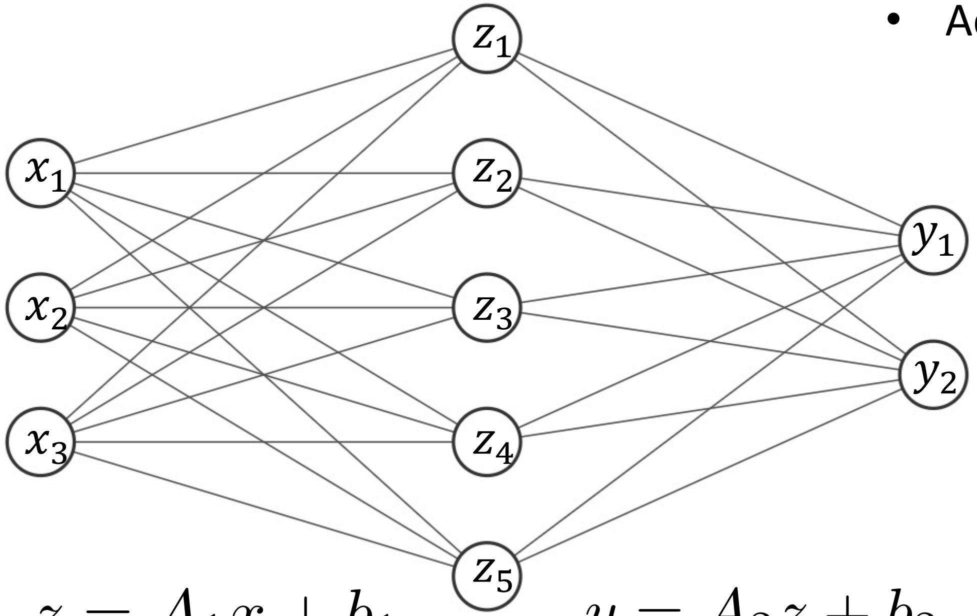


$$z = \underbrace{A_1}_{5\times 3} x \qquad y = \underbrace{A_2}_{2\times 5} z$$

$$y = \underbrace{A_2}_{2\times 5}\underbrace{A_1}_{5\times 3} x$$

Layers of linear transformations

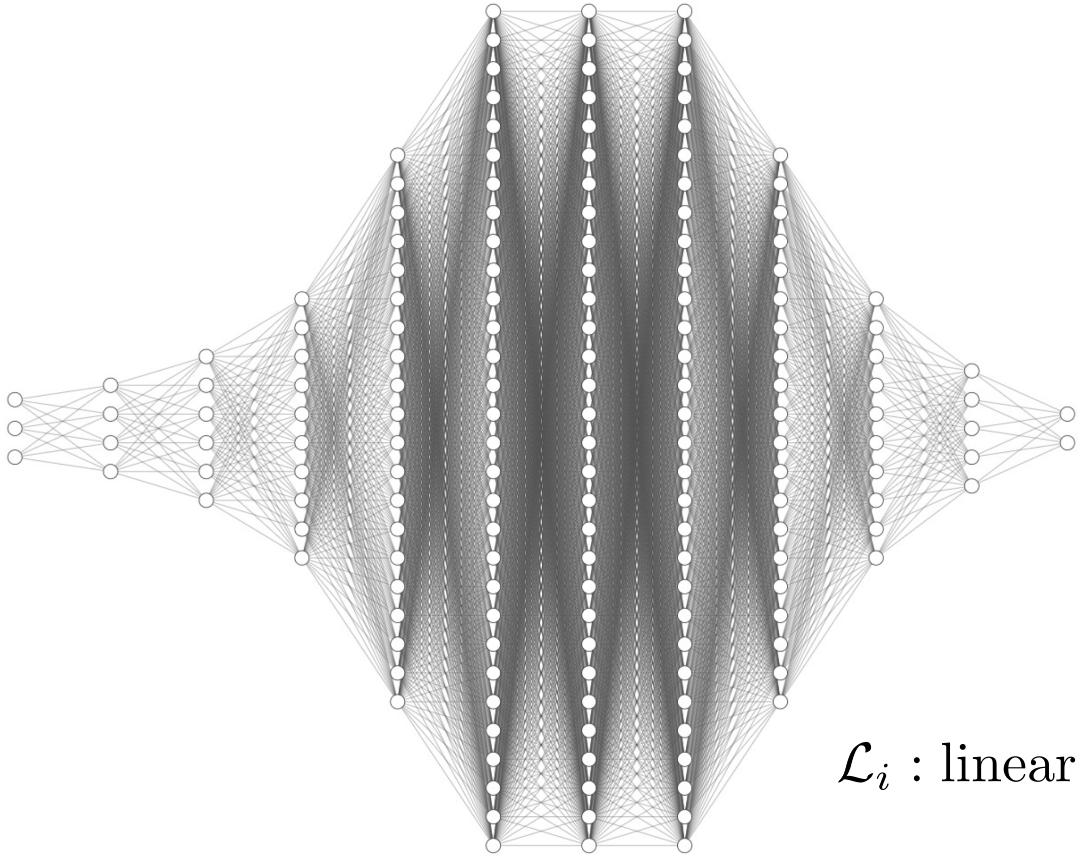
- Add biases to linear transformations



$$z = A_1 x + b_1 \qquad y = A_2 z + b_2$$

$$y = A_2(A_1 x + b_1) + b_2$$

Deep neural networks . . . (getting there)



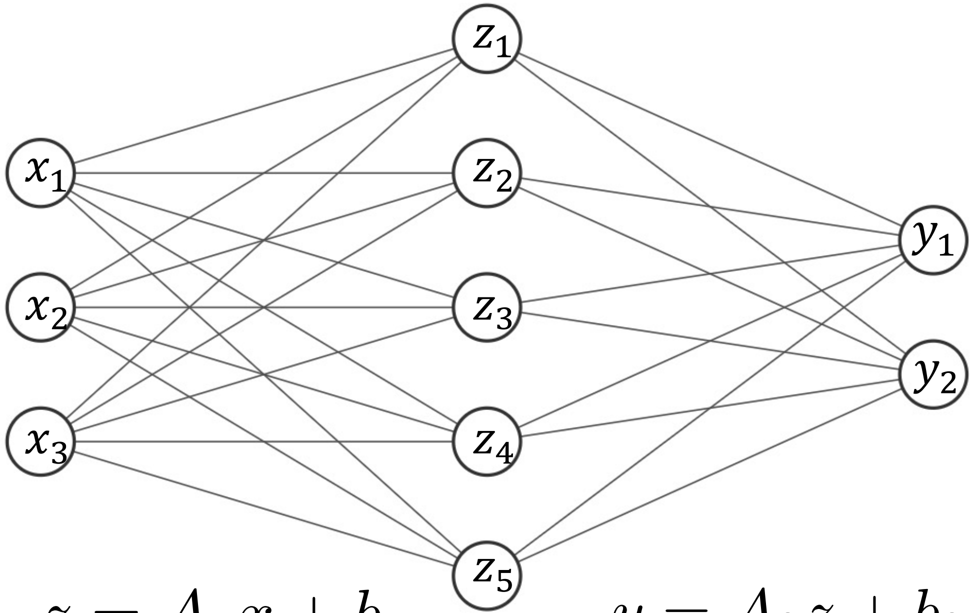
Linear function of linear function is still a linear function

Can we introduce non-linearity in our approximation?

$\mathcal{L}_i$: linear transformation

$$y = \mathcal{L}_n \circ \mathcal{L}_{n-1} \circ \cdots \circ \mathcal{L}_2 \circ \mathcal{L}_1 x$$

Deep neural networks . . . (getting there)

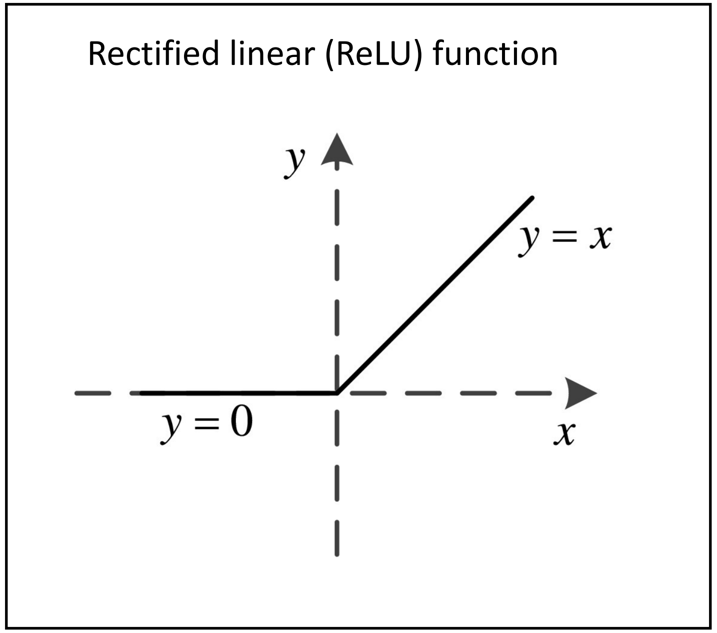


$$z = A_1 x + b_1 \qquad\qquad y = A_2 z + b_2$$

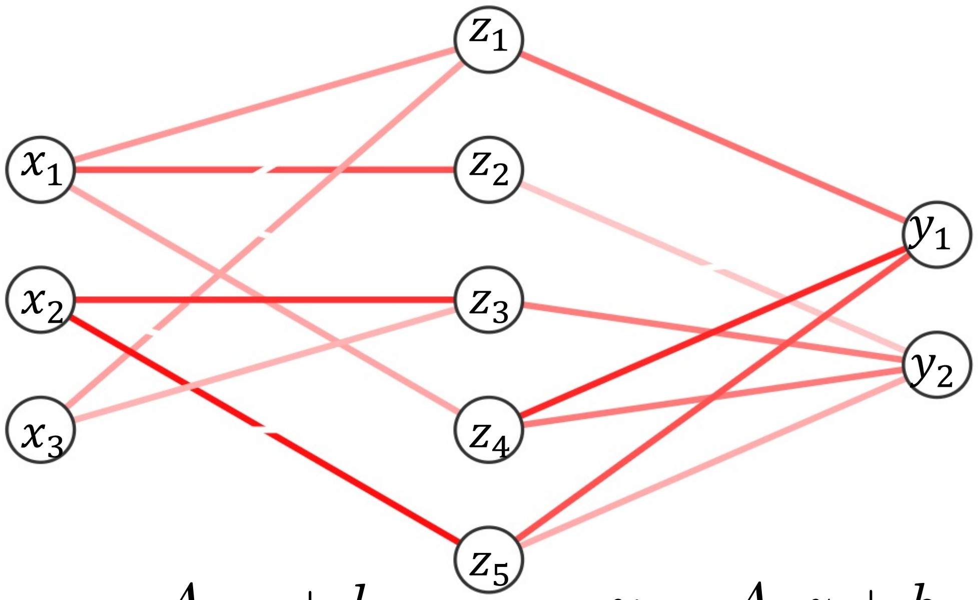
$$\Big\downarrow \qquad\qquad\qquad\qquad \Big\downarrow$$

$$z = \sigma(A_1 x + b_1) \qquad\qquad y = \sigma(A_2 z + b_2)$$

σ : Activation function
(Element-wise non-linear function)



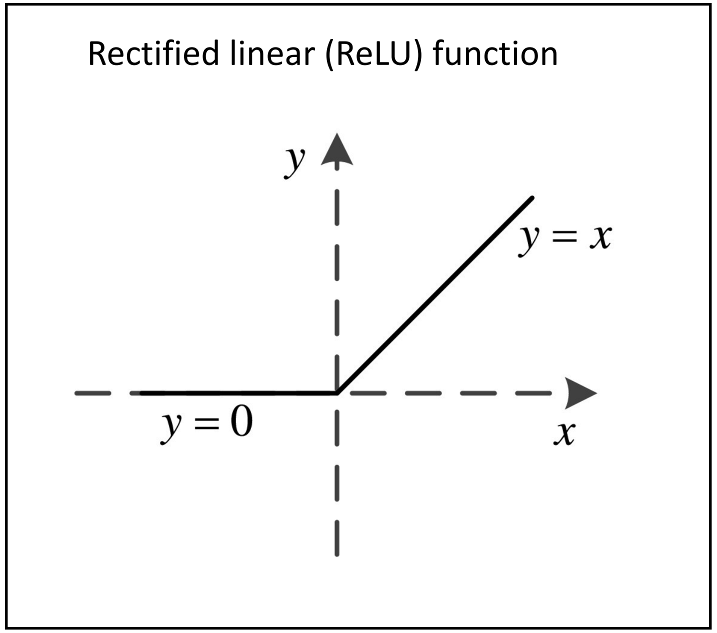
Deep neural networks . . . (getting there)



$$z = A_1 x + b_1 \quad\longrightarrow\quad z = \sigma(A_1 x + b_1)$$

$$y = A_2 z + b_2 \quad\longrightarrow\quad y = \sigma(A_2 z + b_2)$$

σ : Activation function
(Element-wise non-linear function)

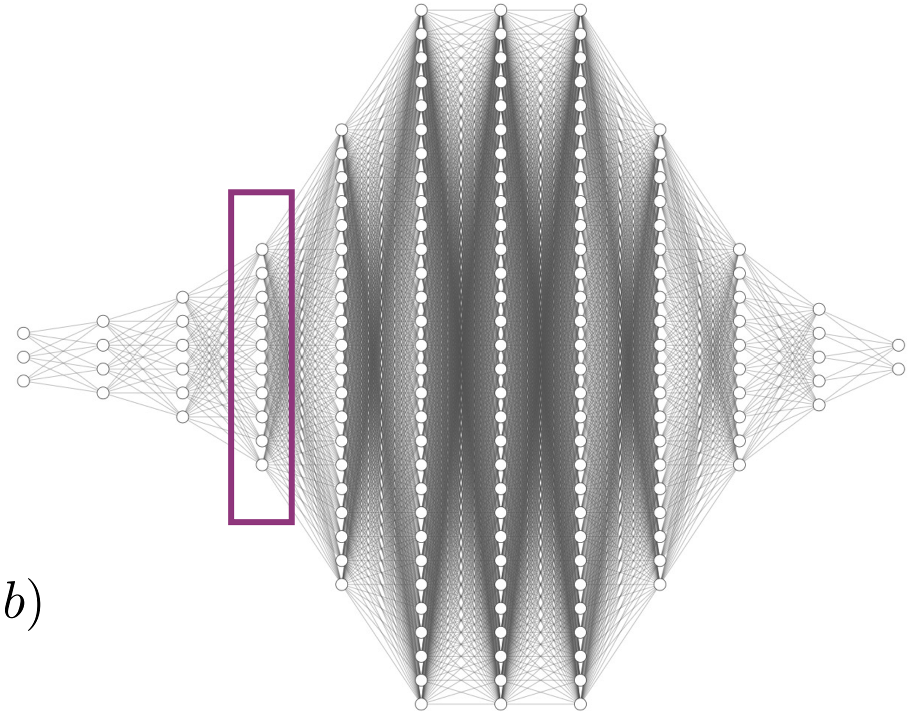


<u>Only some connections are activated, similar to neurons firing in the brain</u>

Deep neural networks

In each layer:

$$\text{Input: } x$$
$$\downarrow$$
$$Ax$$
$$\downarrow$$
$$Ax + b$$
$$\downarrow$$
$$\text{Output: } \sigma(Ax + b)$$



$$y = \mathcal{R}_n \circ \cdots \circ \mathcal{R}_1 x, \qquad \mathcal{R}_i = \sigma \circ \mathcal{L}_i$$

$$\text{Unknowns:} \quad \{A_i, b_i \mid i = 1, \ldots, n\}$$

(similar to coefficients in polynomials)

Repeated linear transforms and activations

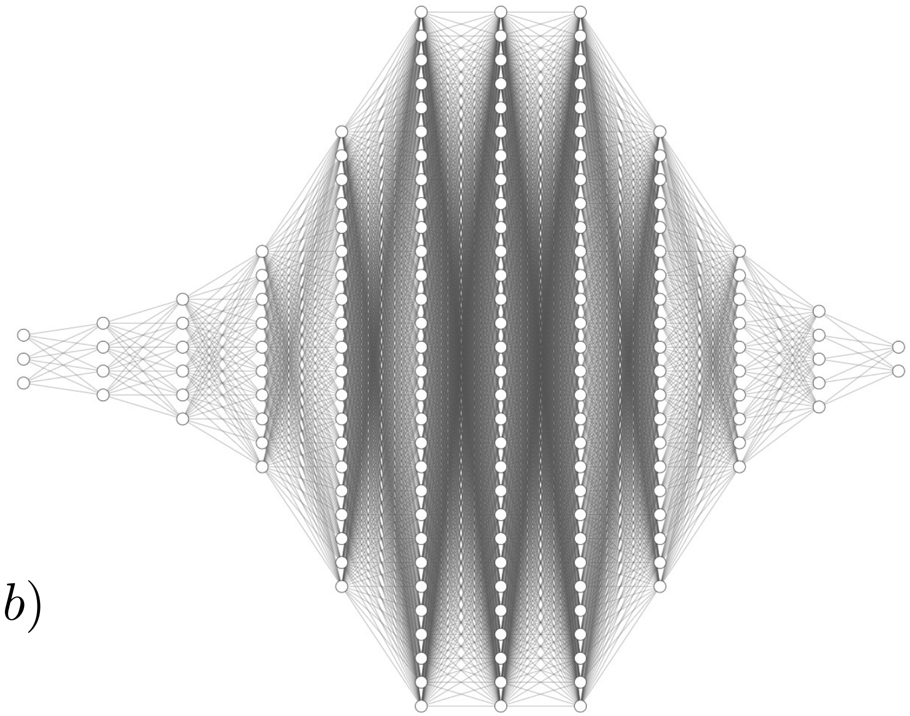
Much higher nonlinearity than polynomials

These types of NNs are called feed-forward NNs

Deep neural networks

In each layer:

$$\begin{array}{c}\text{Input: } x \\ \downarrow \\ Ax \\ \downarrow \\ Ax + b \\ \downarrow \\ \text{Output: } \sigma(Ax+b)\end{array}$$



Repeated linear transforms and activations

⇓

Much higher nonlinearity than polynomials

$$y = \mathcal{R}_n \circ \cdots \circ \mathcal{R}_1 x, \qquad \mathcal{R}_i = \sigma \circ \mathcal{L}_i$$

$$\text{Unknowns:} \quad \{A_i, b_i \mid i = 1, \ldots, n\}$$

(similar to coefficients in polynomials)

These types of NNs are called feed-forward NNs

Recap

1. Polynomial regression/fitting

2. Training-testing splits, underfitting/overfitting, model complexity

3. Neural networks: theory and basics

Follow-ups

1. Start doing group lab assignments *progressively (don't wait until last session)*

2. Solution notebooks from today's session and related lab assignment on Brightspace by today end-of-day

3. Review the neural network codes;
   related advanced contents in next session