Artificial Neural Networks and Deep Learning

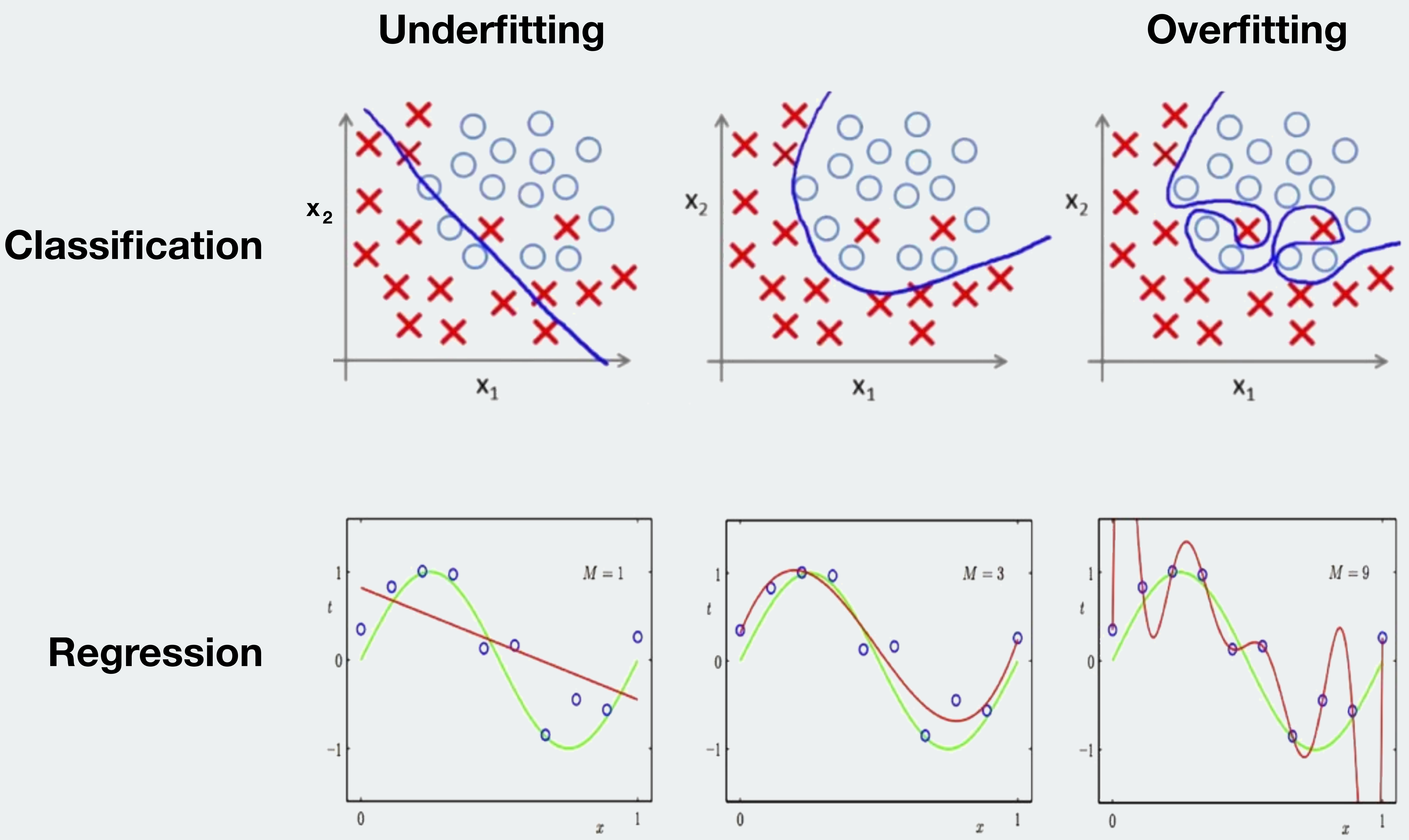
Week 3

Keras, overfitting and regularization

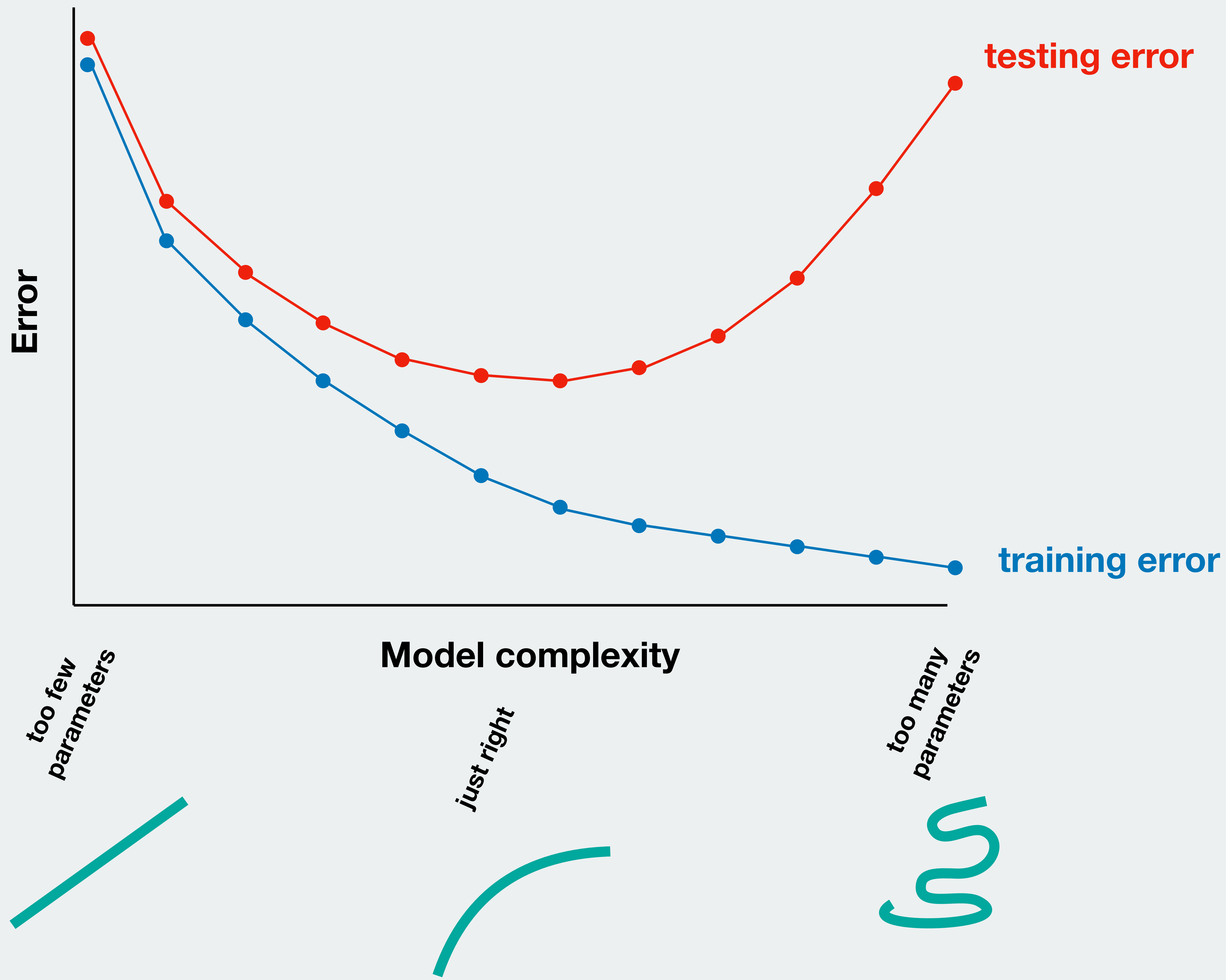
Regularization

Tricks to avoid overfitting

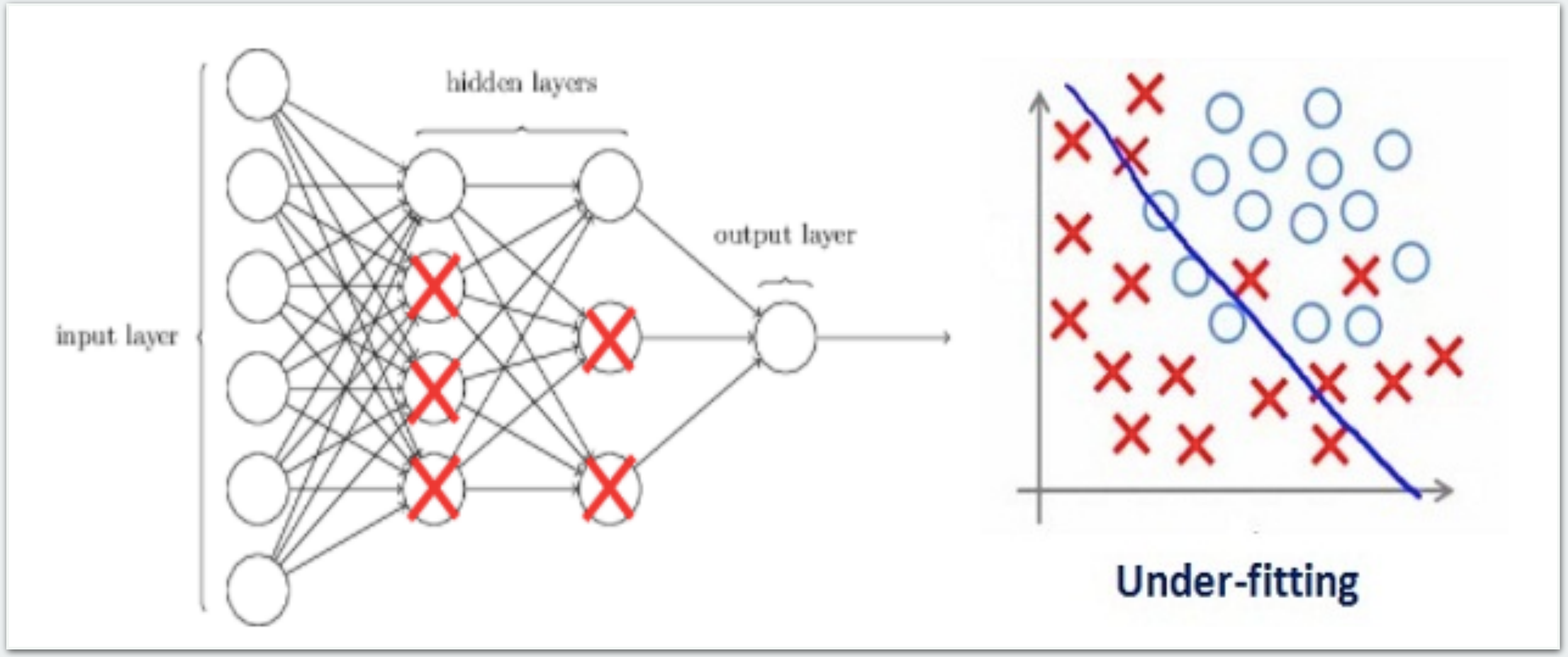
Regularization – underfitting and overfitting



Regularization – underfitting and overfitting



Regularization – *how* does regularization reduce overfitting?



https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Regularization – Different techniques

L-norm regularization: "Introduce a cost for large weights"

$$C = Loss + Regularization\ term$$

Regularization – Different techniques

L-norm regularization: "Introduce a cost for large weights"

$$C = Loss + Regularization\ term$$

L1: $$C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W_l}\|$$

L2: $$C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W_l^2}\|$$

Regularization – Different techniques

L-norm regularization: “Introduce a cost for large weights”

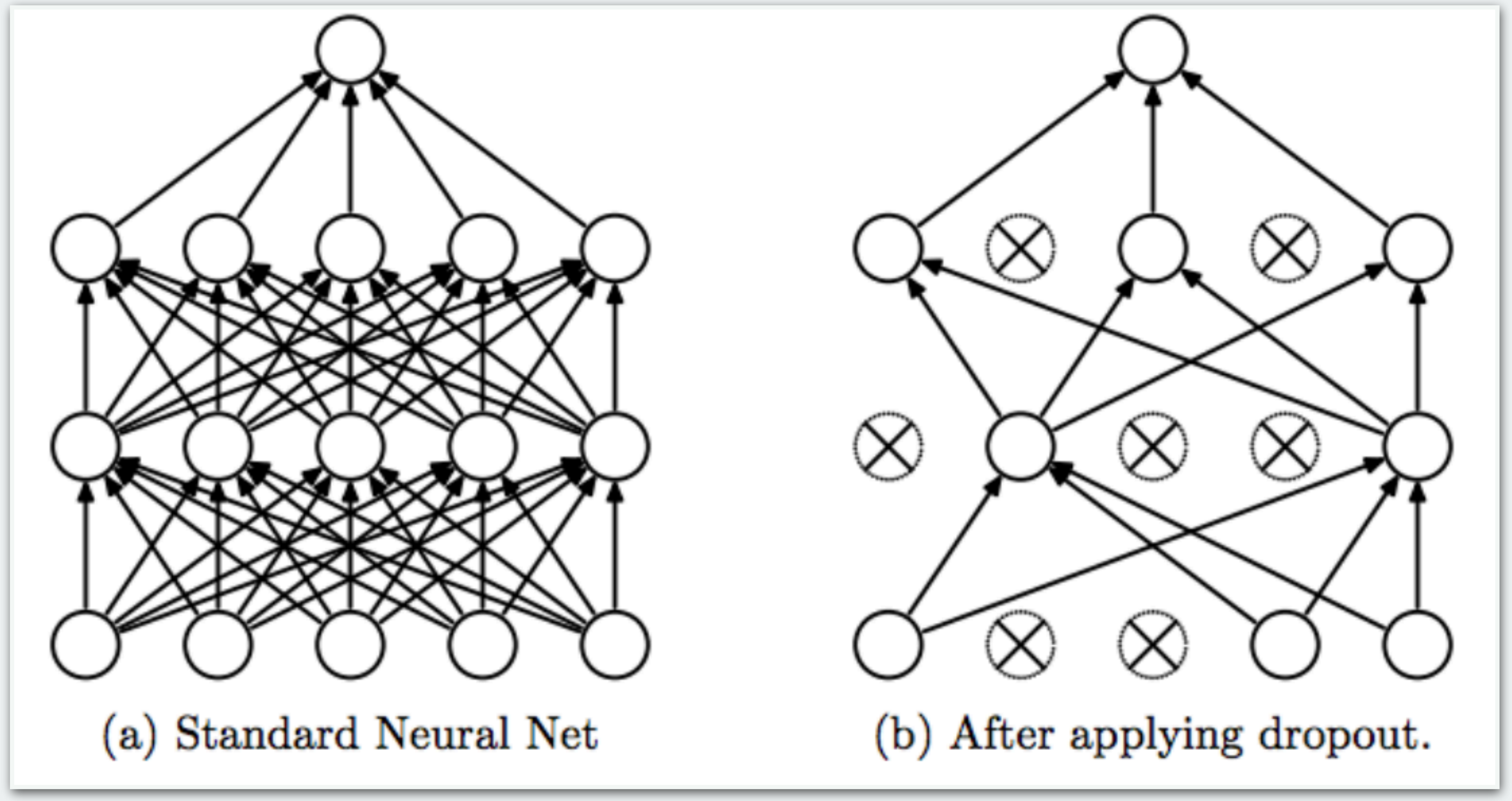
$$C = Loss + Regularization\ term$$

L1: $$C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W_l}\|$$

L2: $$C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W_l^2}\|$$

Dropout:

“In each SGD step, randomly ignore a fraction p of neurons”



Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

- Can select p in wide range. Typical is 0.2 – 0.8, dependent on size of ANN
- Can apply only in specific layers. It is typical to only do dropout in a designated “dropout layer” somewhere close to output.

Regularization – Different techniques

L-norm regularization: “Introduce a cost for large weights”

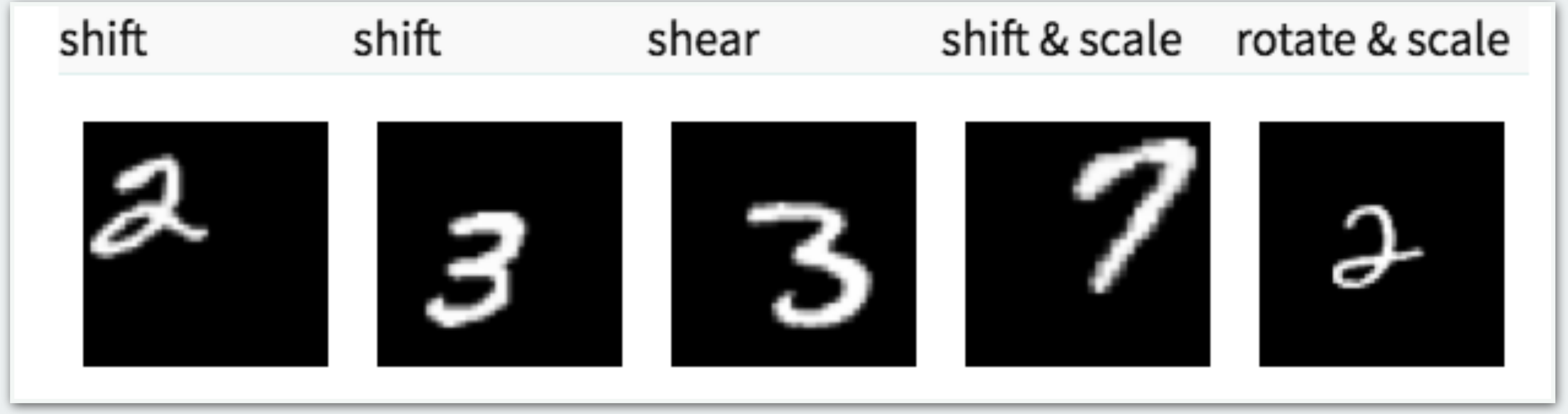
$$C = Loss + Regularization\ term$$

L1: $C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W_l}\|$

L2: $C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W_l^2}\|$

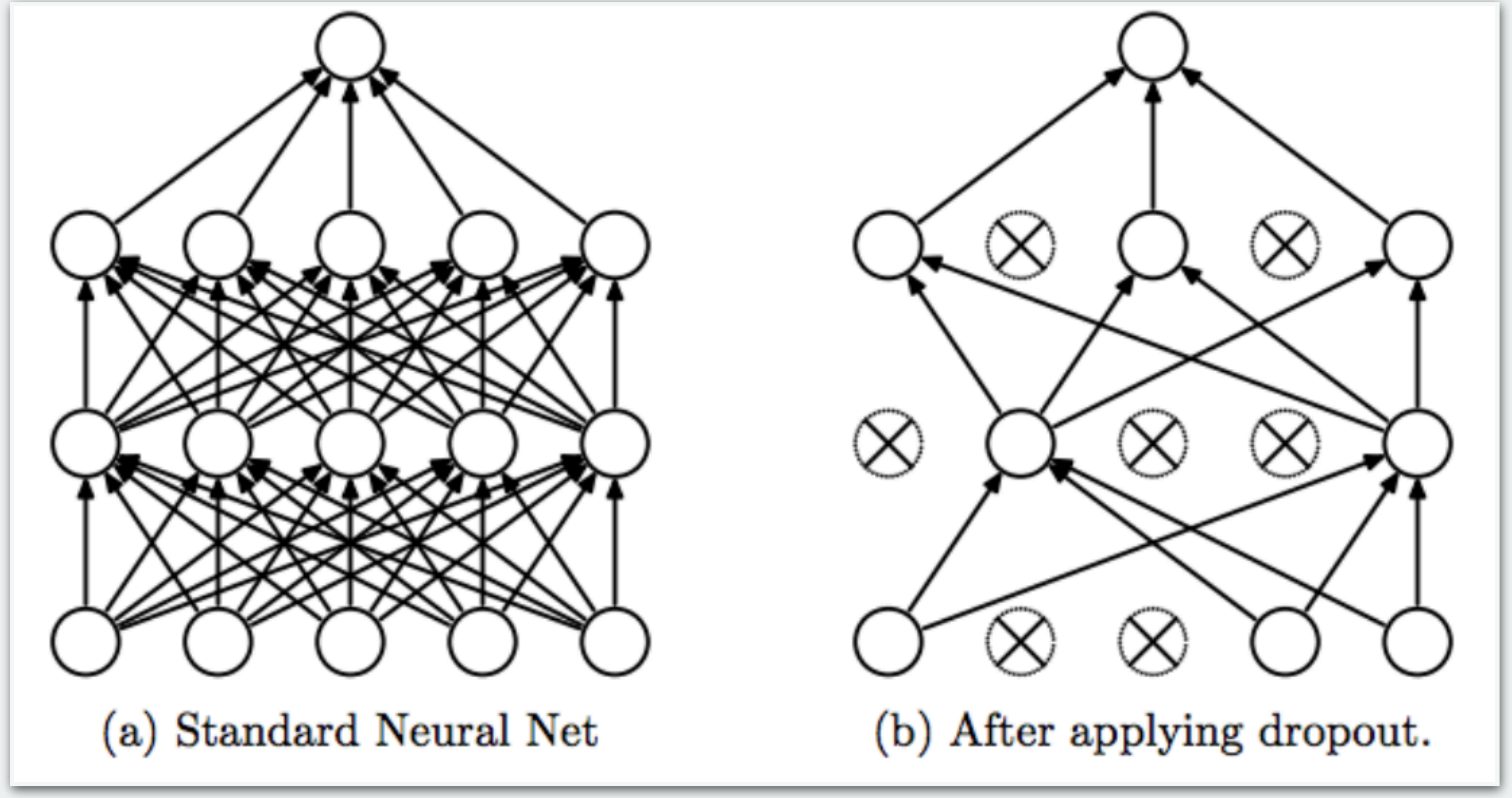
Data augmentation

“Shear, shift, scale and/or rotate input data”



Dropout:

“In each SGD step, randomly ignore a fraction *p* of neurons”



Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

- Can select *p* in wide range. Typical is 0.2 – 0.8, dependent on size of ANN
- Can apply only in specific layers. It is typical to only do dropout in a designated “dropout layer” somewhere close to output.

Regularization – Different techniques

L-norm regularization: “Introduce a cost for large weights”

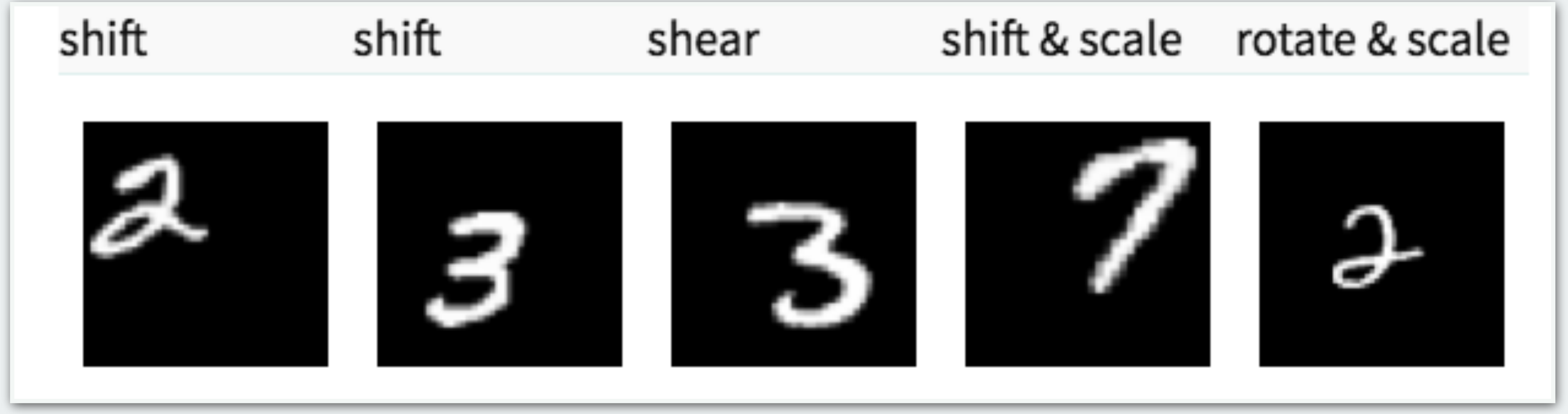
$$C = Loss + Regularization\ term$$

L1: $C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W_l}\|$

L2: $C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W_l^2}\|$

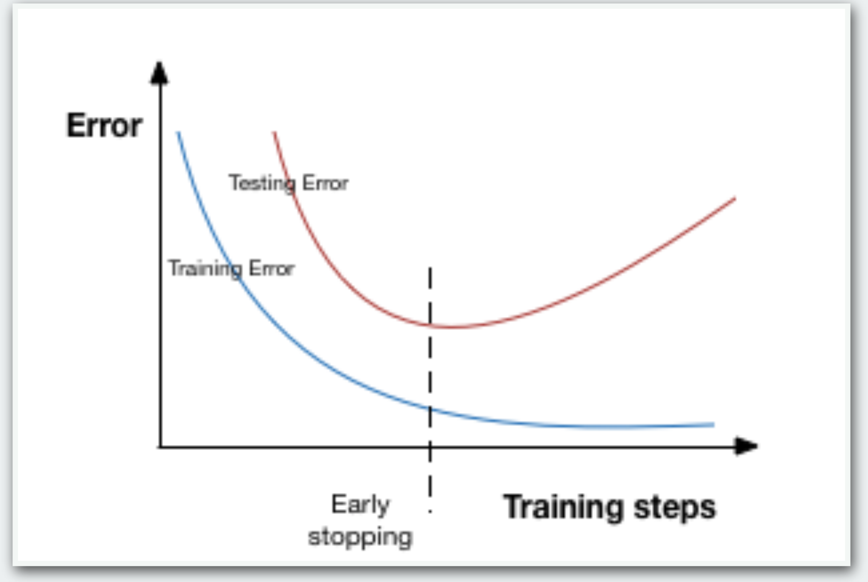
Data augmentation

“Shear, shift, scale and/or rotate input data”



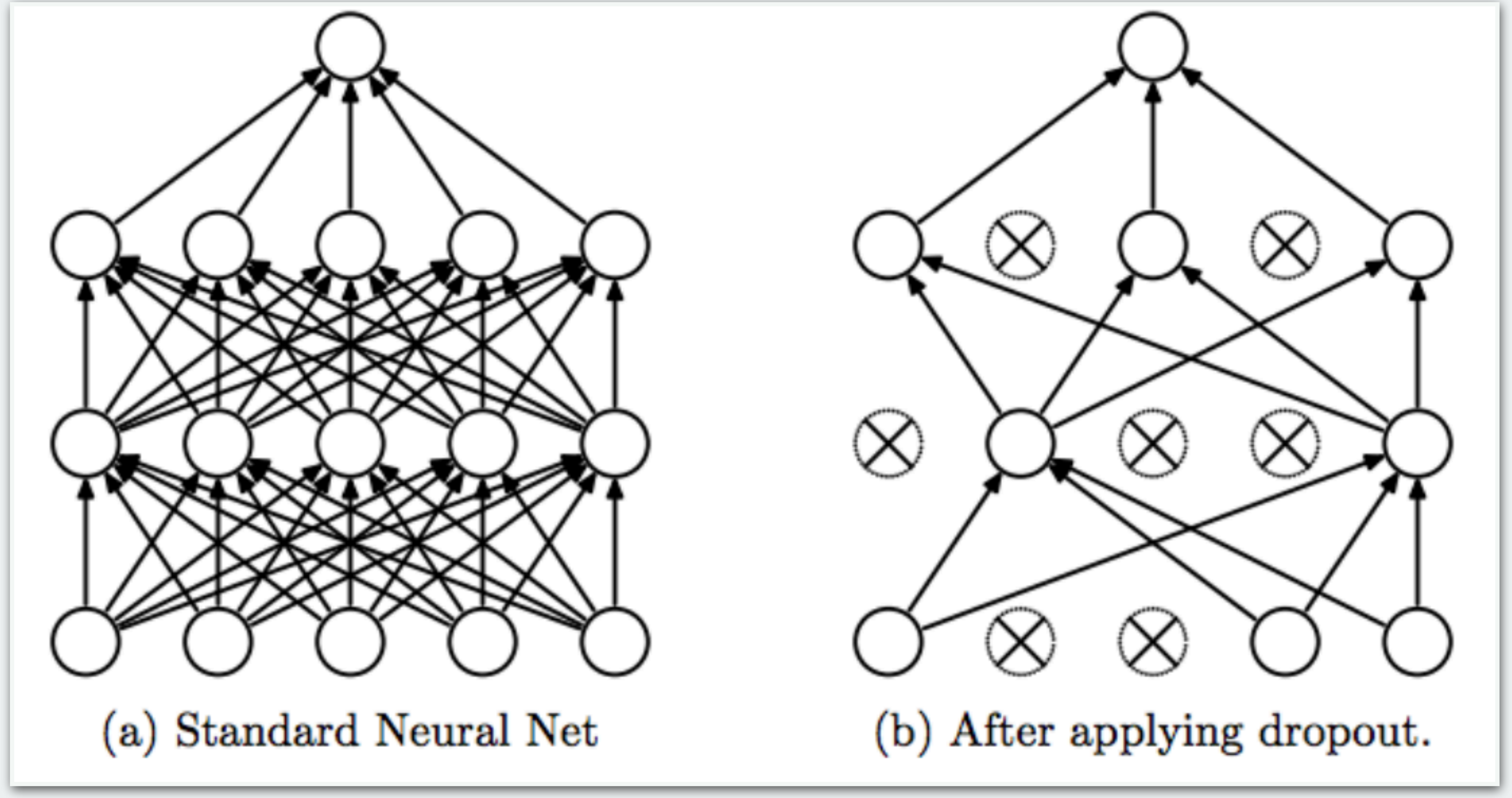
Early stopping

“Stop training when performance on validation dataset starts worsening”



Dropout:

“In each SGD step, randomly ignore a fraction p of neurons”



Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

- Can select p in wide range. Typical is 0.2 – 0.8, dependent on size of ANN
- Can apply only in specific layers. It is typical to only do dropout in a designated “dropout layer” somewhere close to output.

A quick word on:

The Vanishing Gradient Problem

Vanishing gradients – A problem in *deep* neural nets

Problem:

- Gradients closer and closer to the input tend to get smaller and smaller
- Leads to smaller weight updates near input and larger weight updates near output
- Bad because layers near input take part in recognizing “simple” patterns, which are important to learning

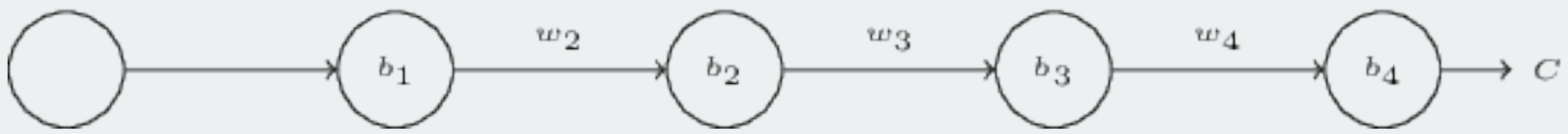
$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$

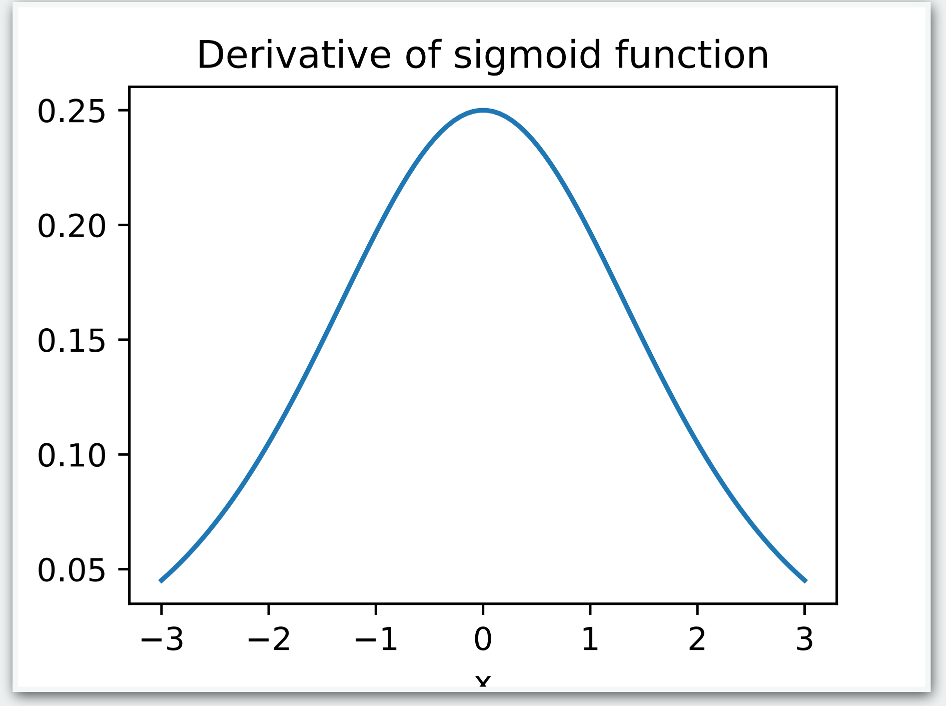
Vanishing gradients – A problem in *deep* neural nets

Problem:

- Gradients closer and closer to the input tend to get smaller and smaller
- Leads to smaller weight updates near input and larger weight updates near output
- Bad because layers near input take part in recognizing "simple" patterns, which are important to learning

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



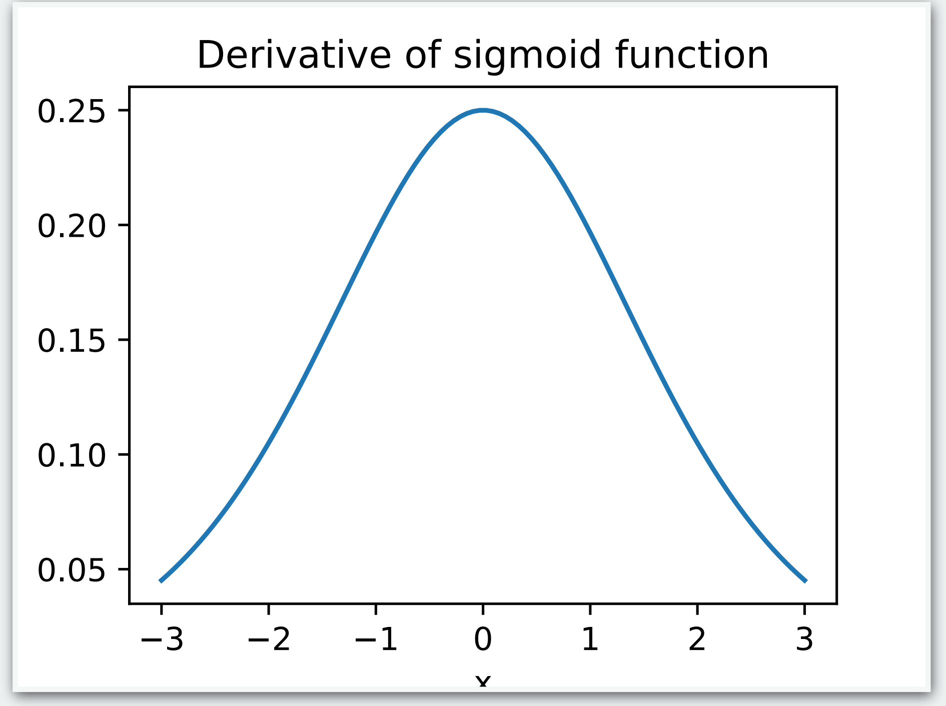


Vanishing gradients – A problem in *deep* neural nets

Problem:

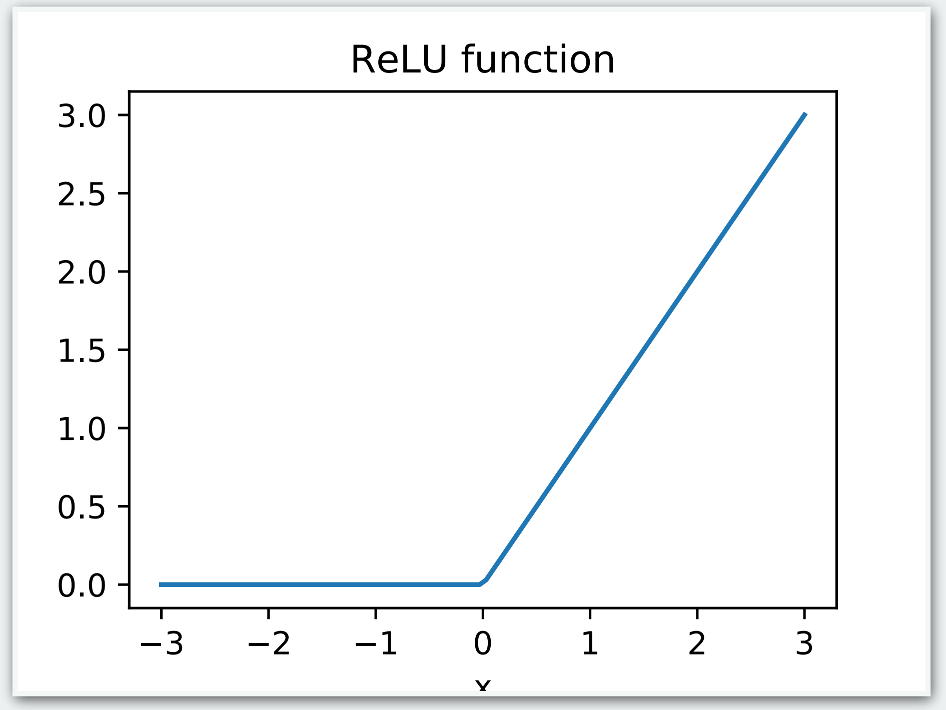
- Gradients closer and closer to the input tend to get smaller and smaller
- Leads to smaller weight updates near input and larger weight updates near output
- Bad because layers near input take part in recognizing "simple" patterns, which are important to learning

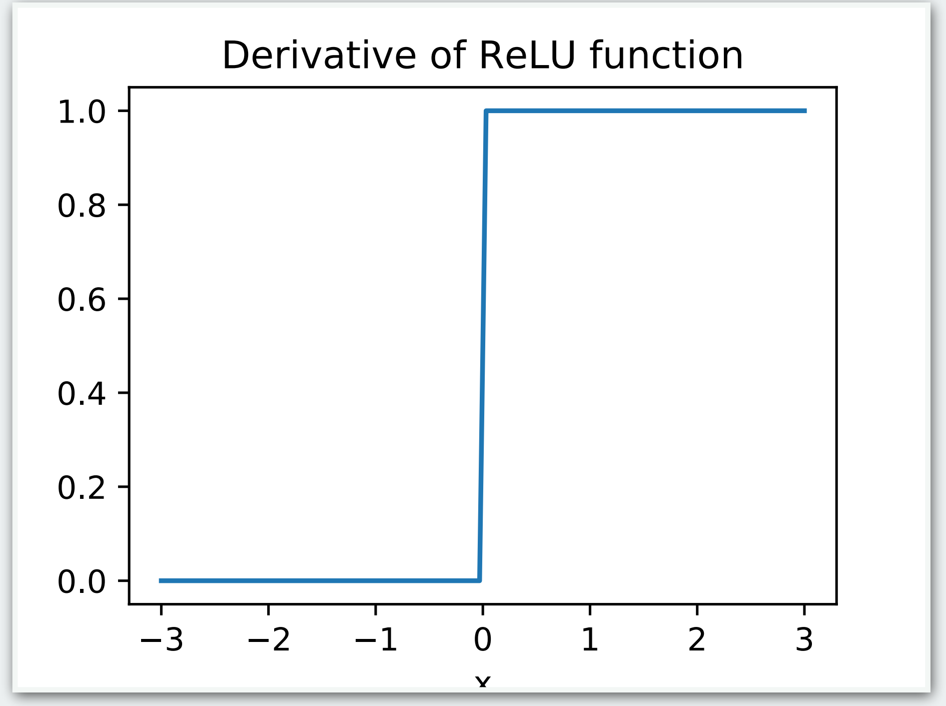
$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



Solution:

- Use an activation function without small gradient for high values
- Candidate activation function: ReLU



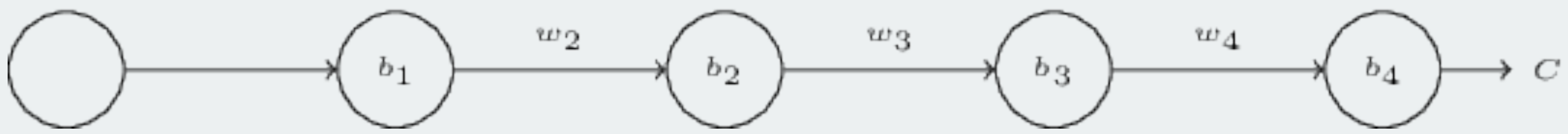


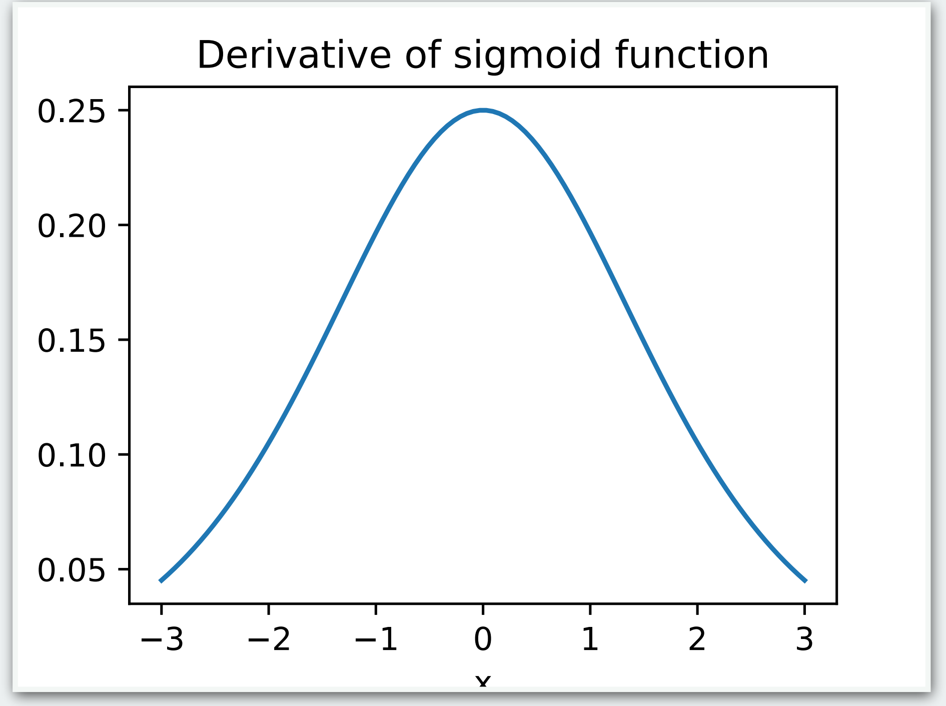
Vanishing gradients – A problem in *deep* neural nets

Problem:

- Gradients closer and closer to the input tend to get smaller and smaller
- Leads to smaller weight updates near input and larger weight updates near output
- Bad because layers near input take part in recognizing "simple" patterns, which are important to learning

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$





Solution:

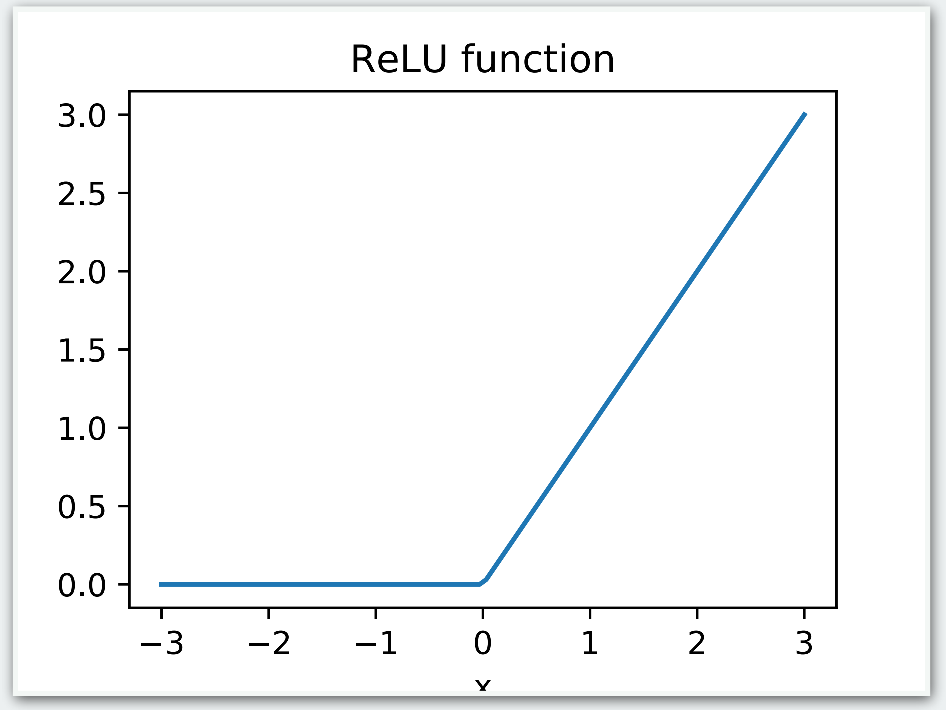
- Use an activation function without small gradient for high values
- Candidate activation function: ReLU

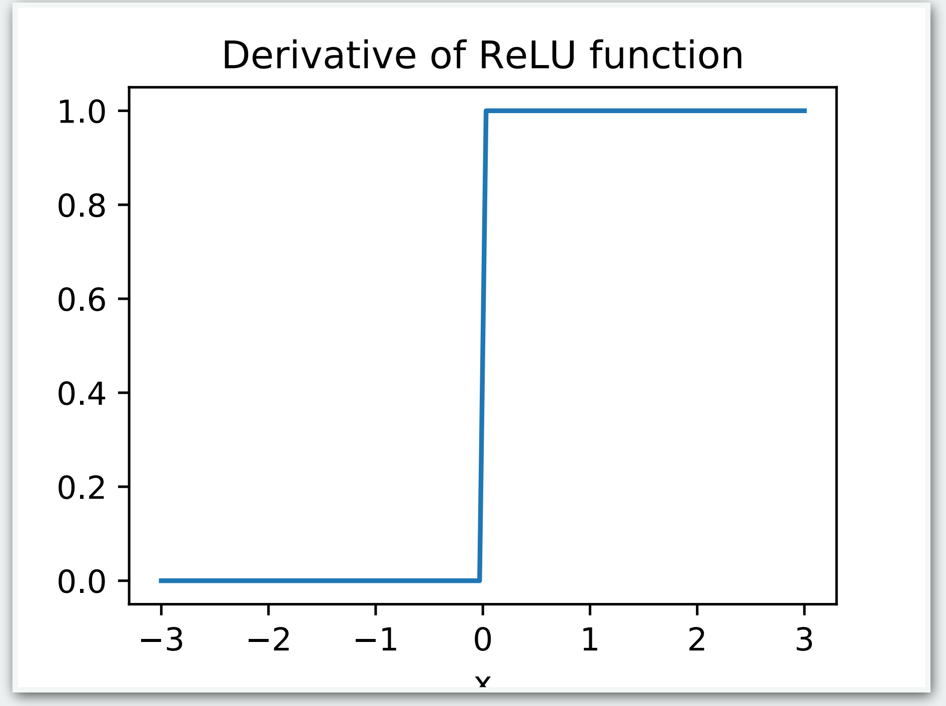
Problems with ReLU:

- Exploding gradients!

Solution:

- Batch normalization, gradient clipping, weight regularization



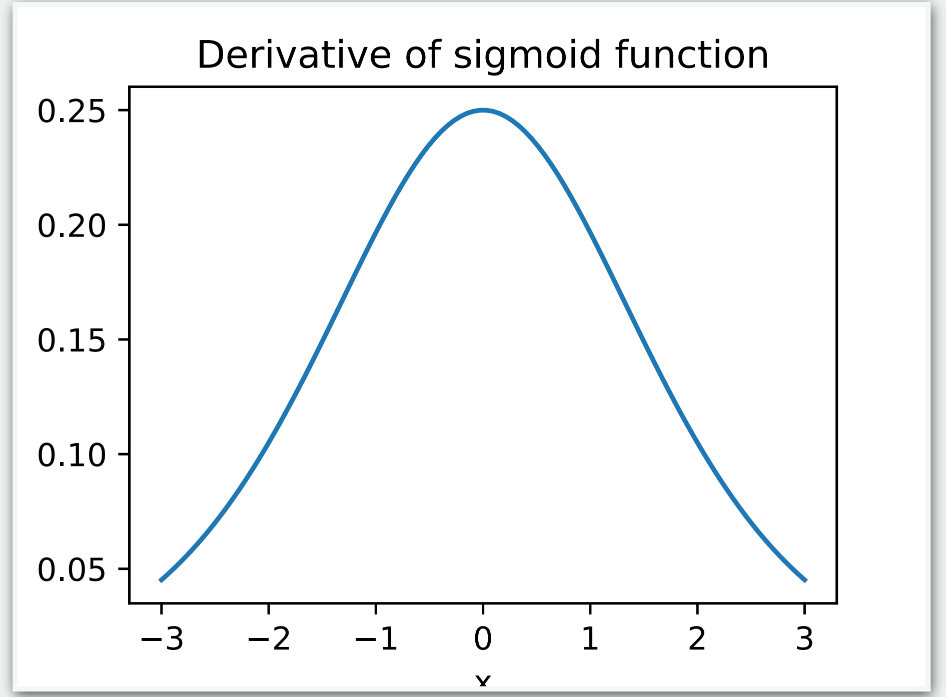


Vanishing gradients – A problem in *deep* neural nets

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$

Problem:

- Gradients closer and closer to the input tend to get smaller and smaller
- Leads to smaller weight updates near input and larger weight updates near output
- Bad because layers near input take part in recognizing "simple" patterns, which are important to learning



Solution:

- Use an activation function without small gradient for high values
- Candidate activation function: ReLU

Problems with ReLU:

- Exploding gradients!

Solution:

- Batch normalization, gradient clipping, weight regularization