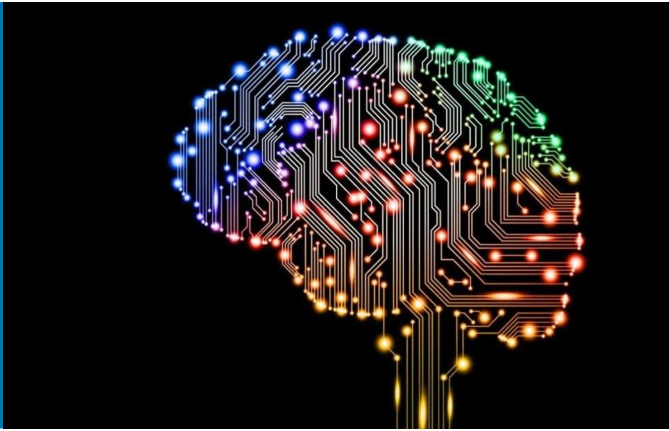
AUG 9, 2019

Deep Learning: Basics



Prasanna Balaprakash

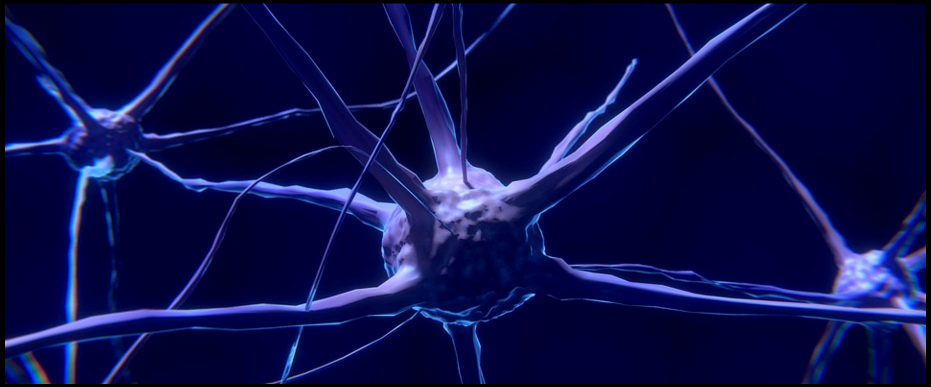
Mathematics and Computer Science Division &

Leadership Computing Facility

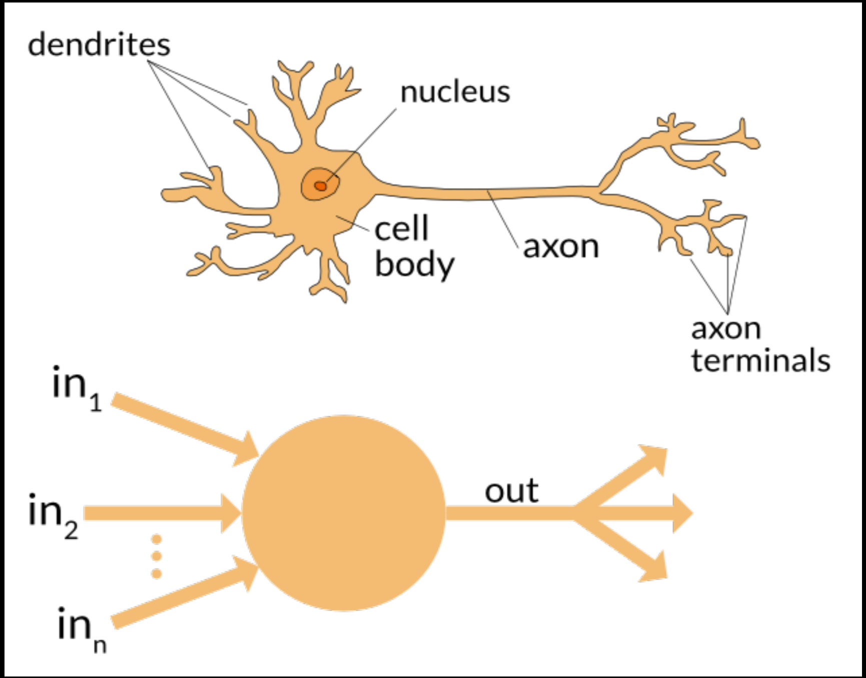
Argonne National Laboratory

What is difficult for a computer?

Brain and neurons



Perceptron



(Loosely) inspired by neurobiology

Frank Rosenblatt, 1952

Supervised deep learning

Inputs **Outputs**

 Cat

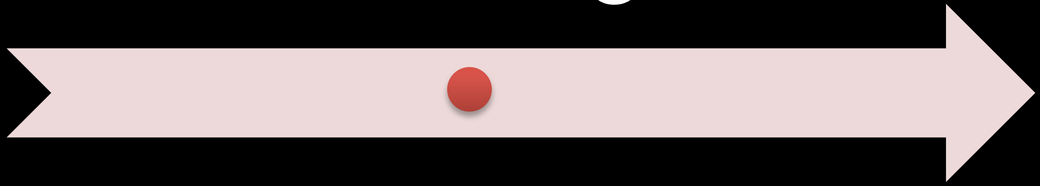
 Dog

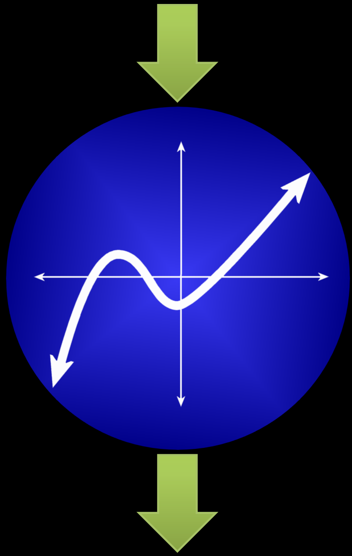
 Horse

 Elephant

 Tiger

Training



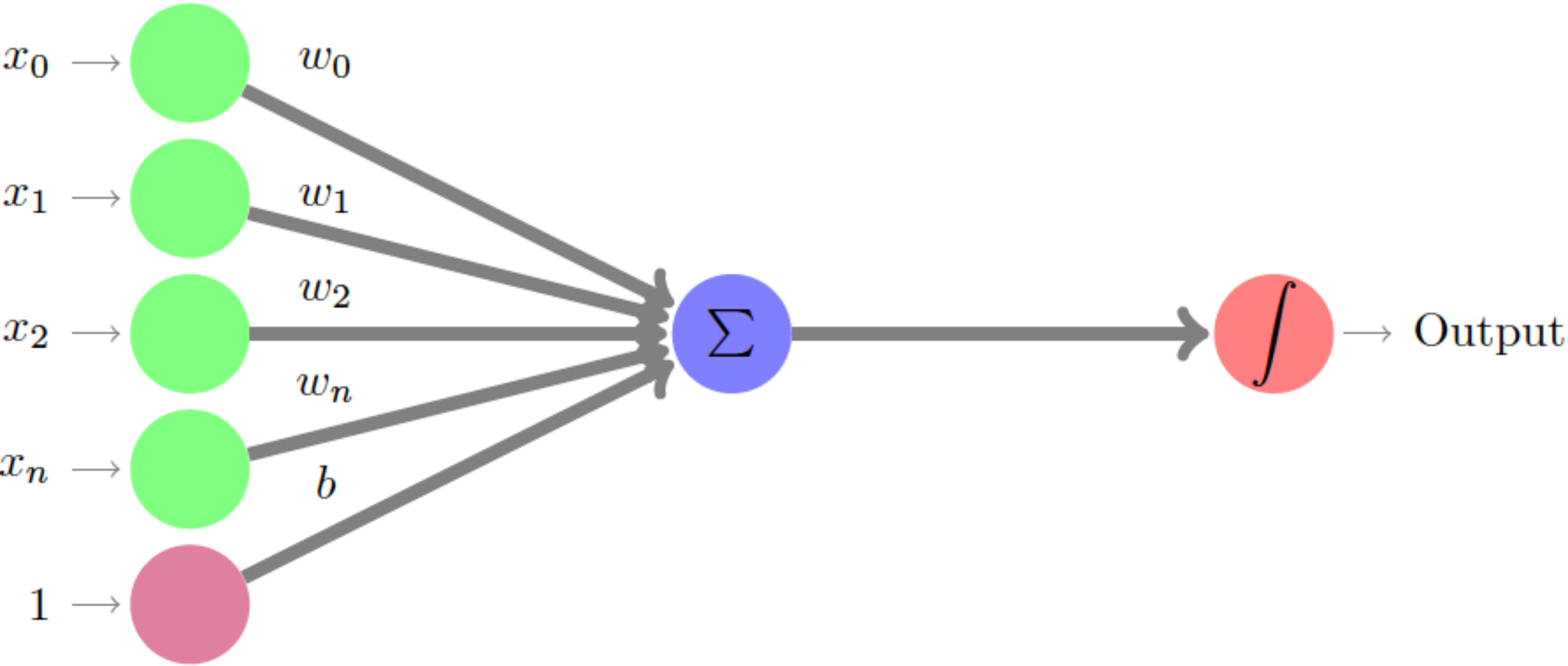


?

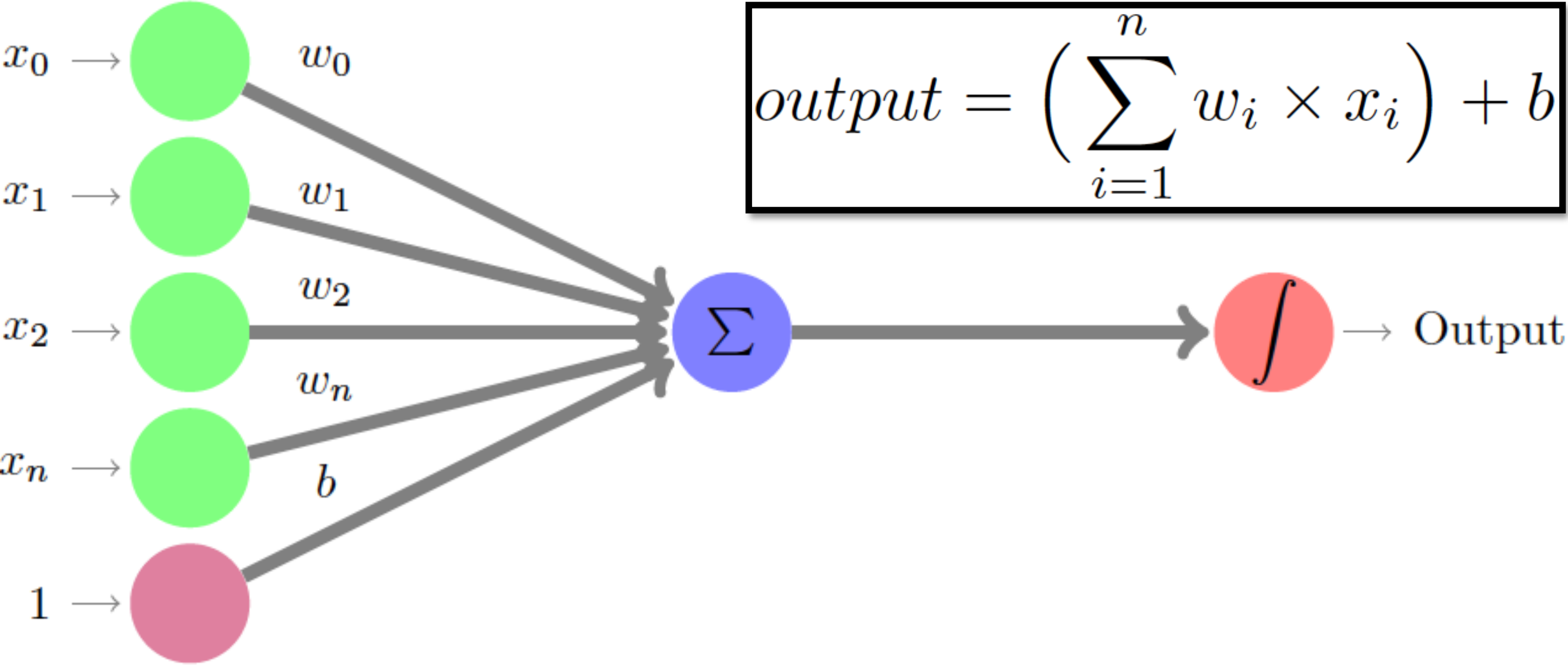
Outline

- Perceptron and deep neural networks
- Training deep neural networks
- Improving training

Perceptron



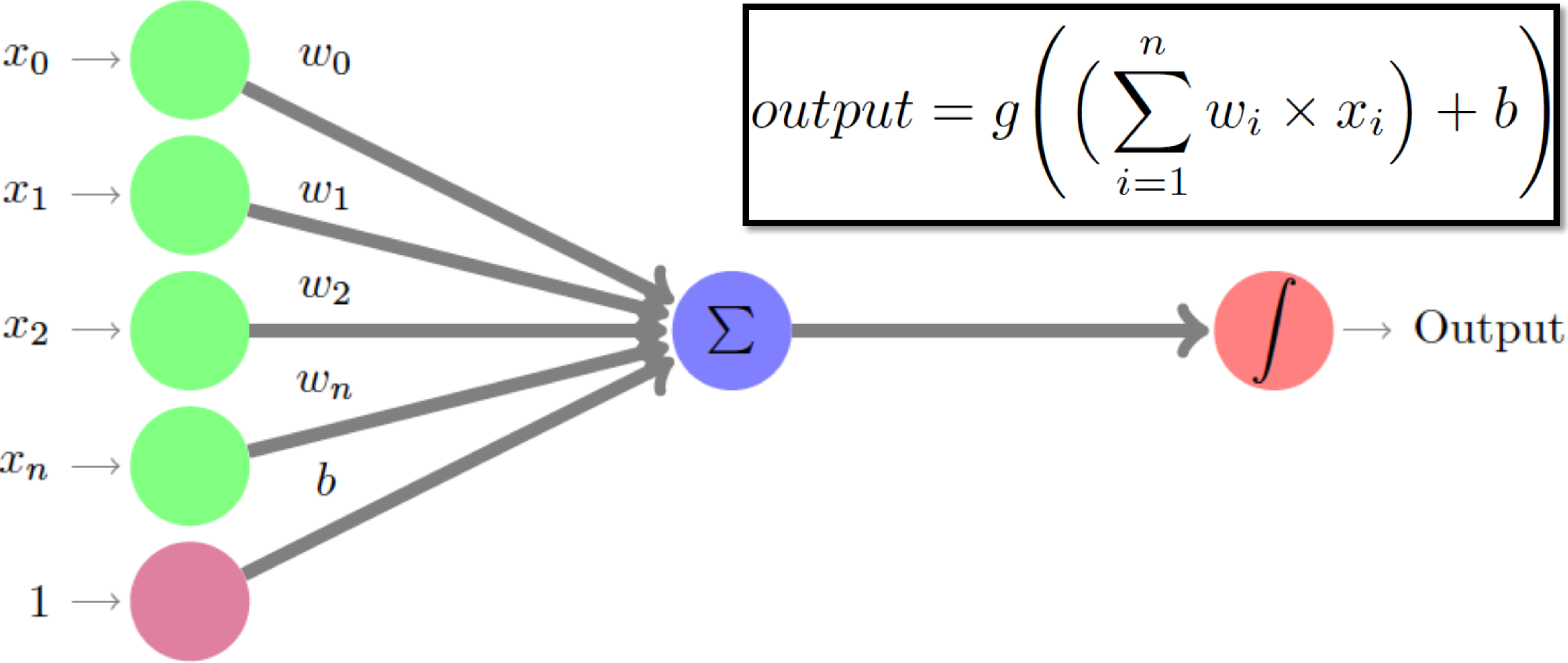
Perceptron



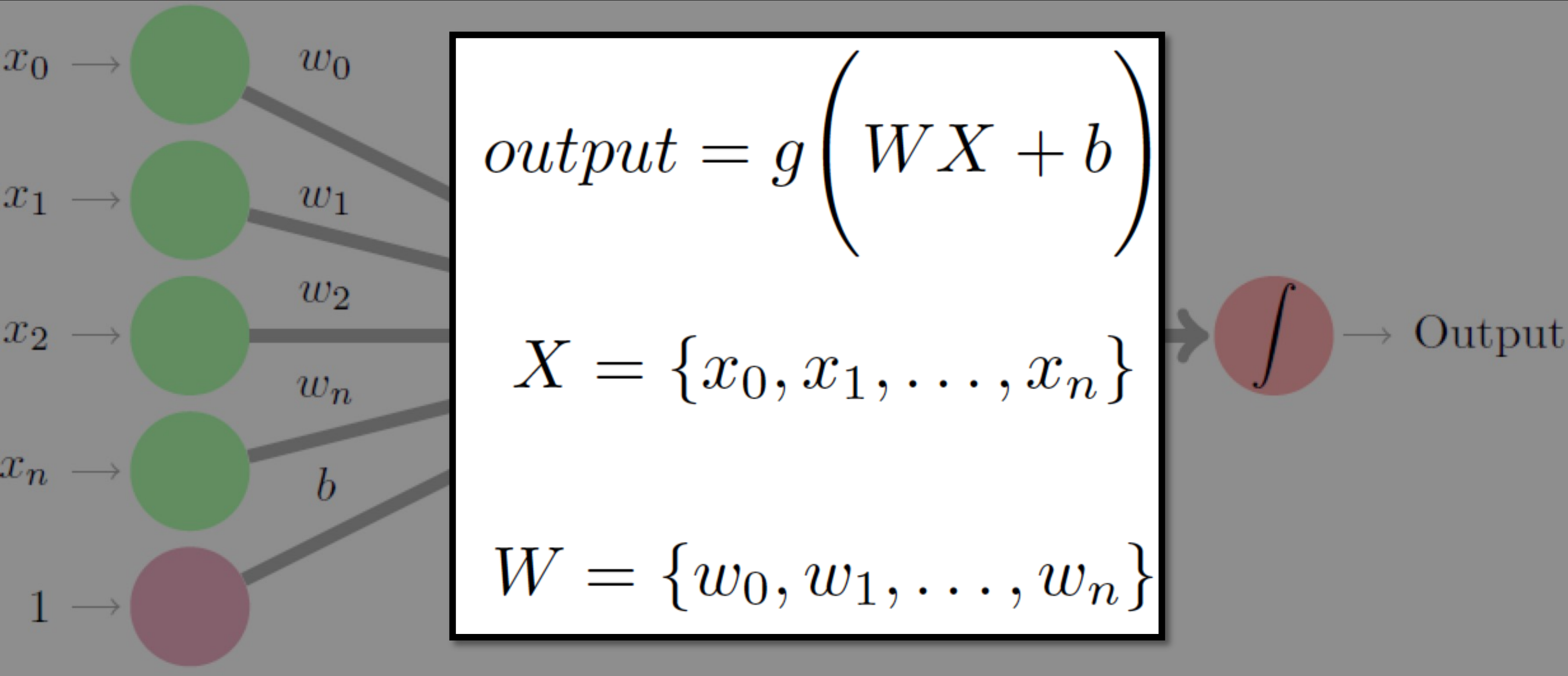
$$\boxed{output = \left(\sum_{i=1}^{n} w_i \times x_i\right) + b}$$

Perceptron

$$output = g\left(\left(\sum_{i=1}^{n} w_i \times x_i\right) + b\right)$$



Perceptron



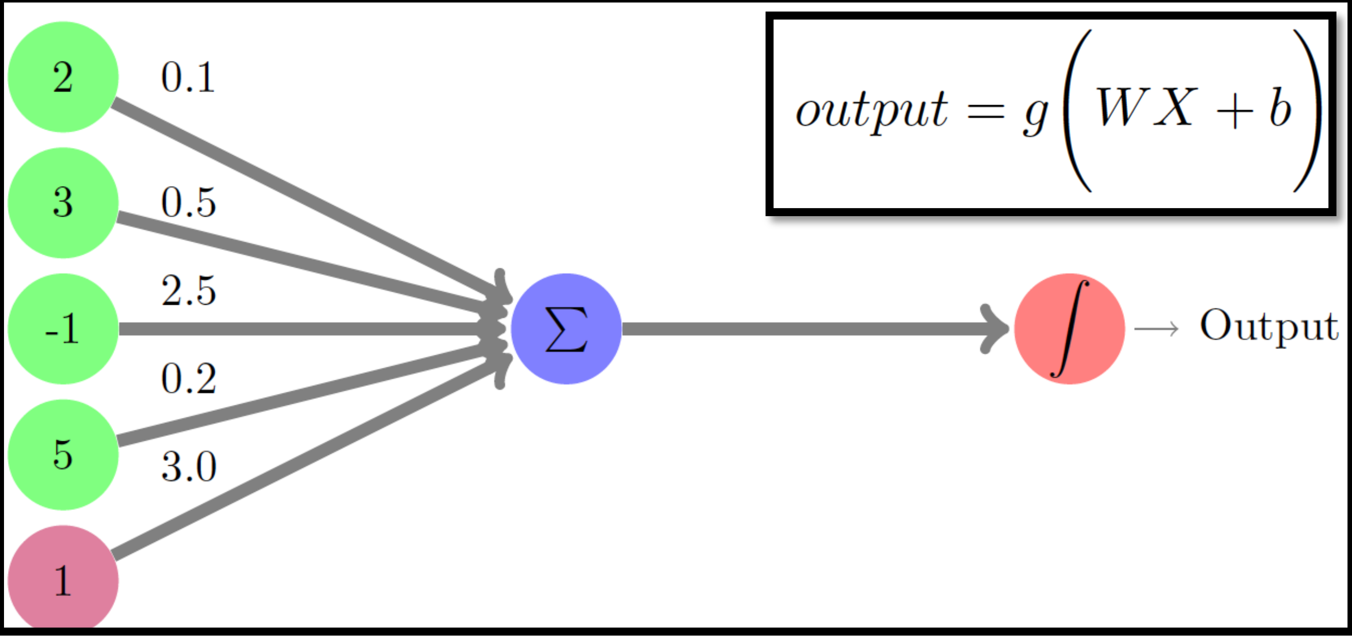
$$output = g\left(WX + b \right)$$

$$X = \{x_0, x_1, \ldots, x_n\}$$

$$W = \{w_0, w_1, \ldots, w_n\}$$

Perceptron

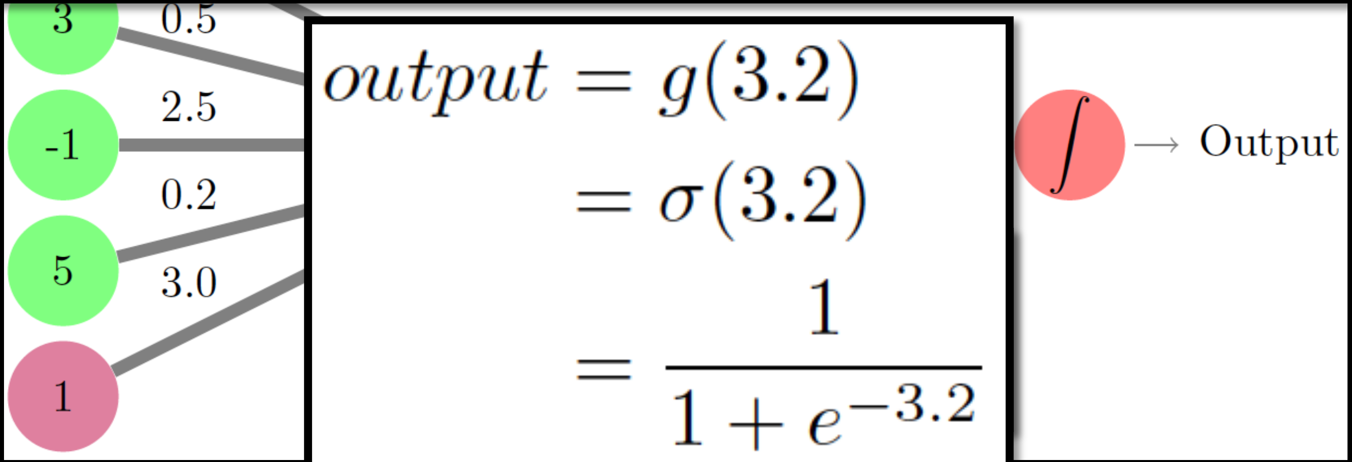
$$\boxed{output = g\left(WX + b\right)}$$



$$output = g\left(\left(2 * 0.1 + 3 * 0.5 + -1 * 2.5 + 5 * 0.2\right) + 1 * 3.0\right)$$

Perceptron

$$output = g\left(\left(2 * 0.1 + 3 * 0.5 + -1 * 2.5 + 5 * 0.2\right) + 1 * 3.0\right)$$

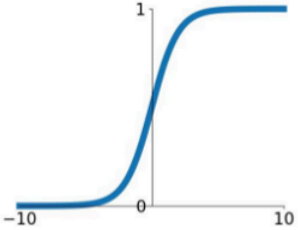


$$\begin{aligned} output &= g(3.2) \\ &= \sigma(3.2) \\ &= \frac{1}{1 + e^{-3.2}} \\ &= 0.96 \end{aligned}$$

Common activation functions

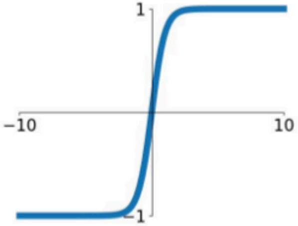
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



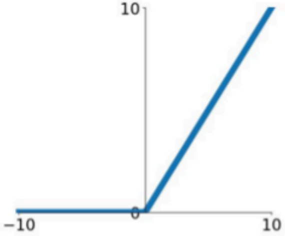
tanh

$\tanh(x)$



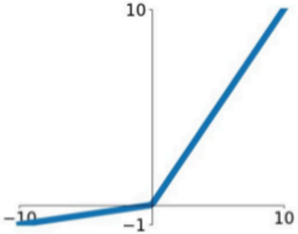
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

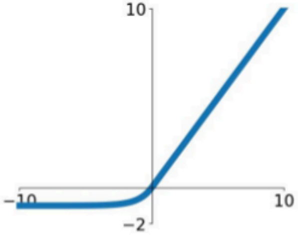


Maxout

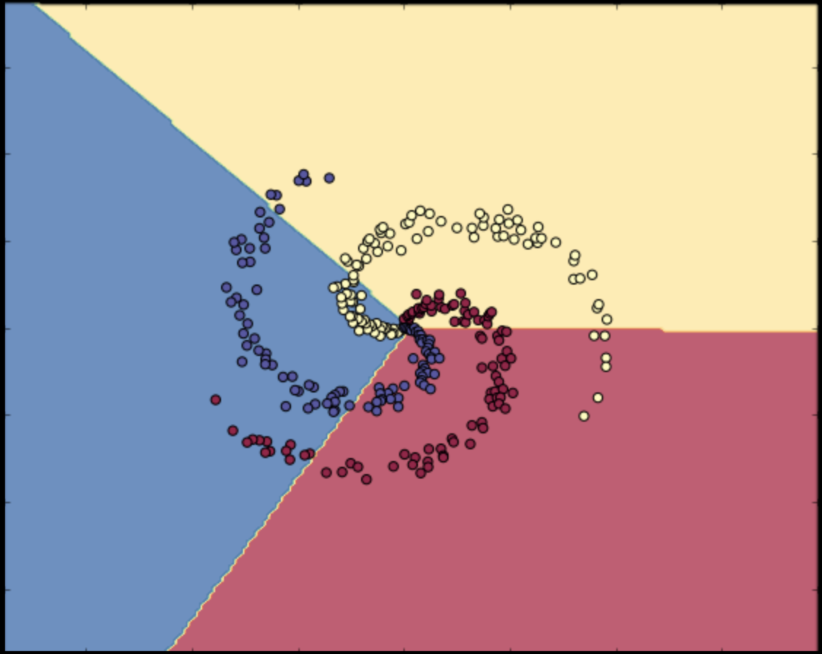
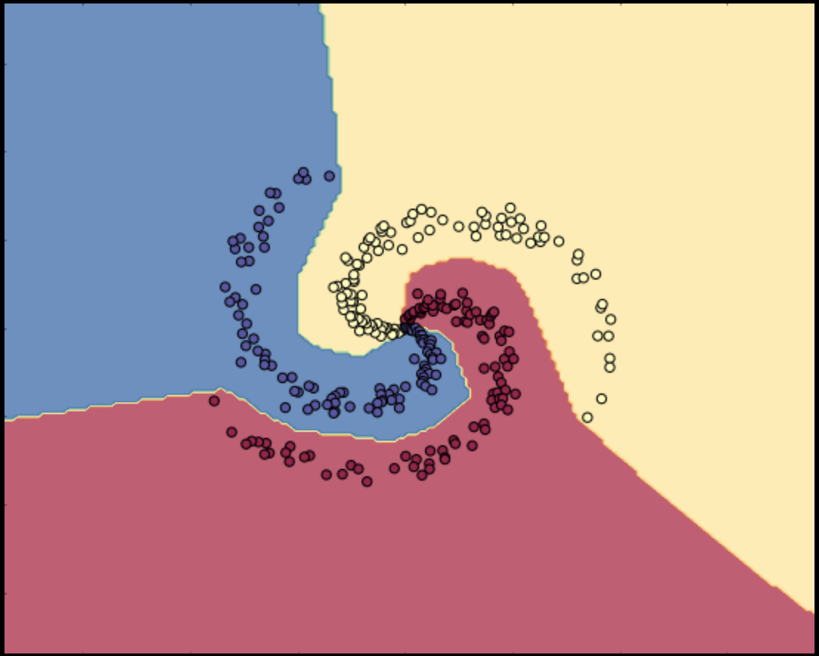
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

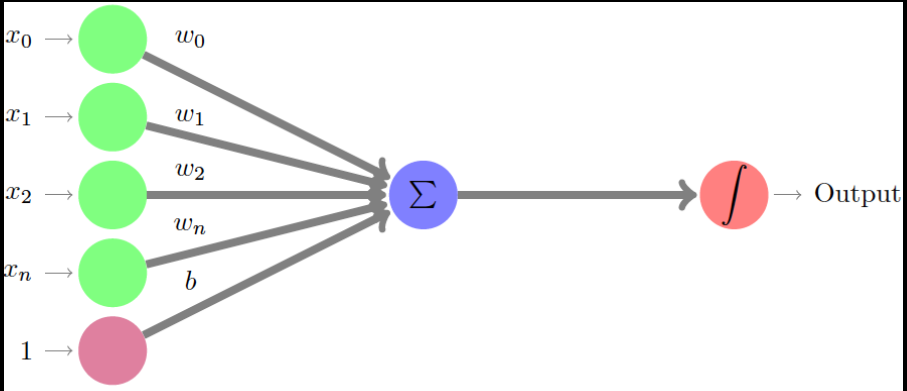
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

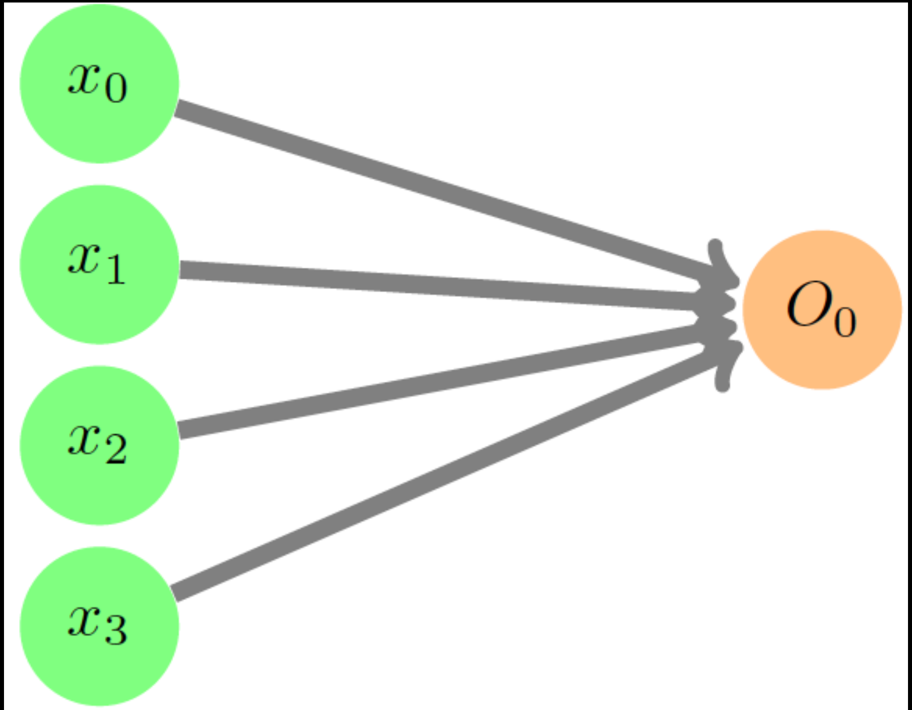


Importance of nonlinear activations

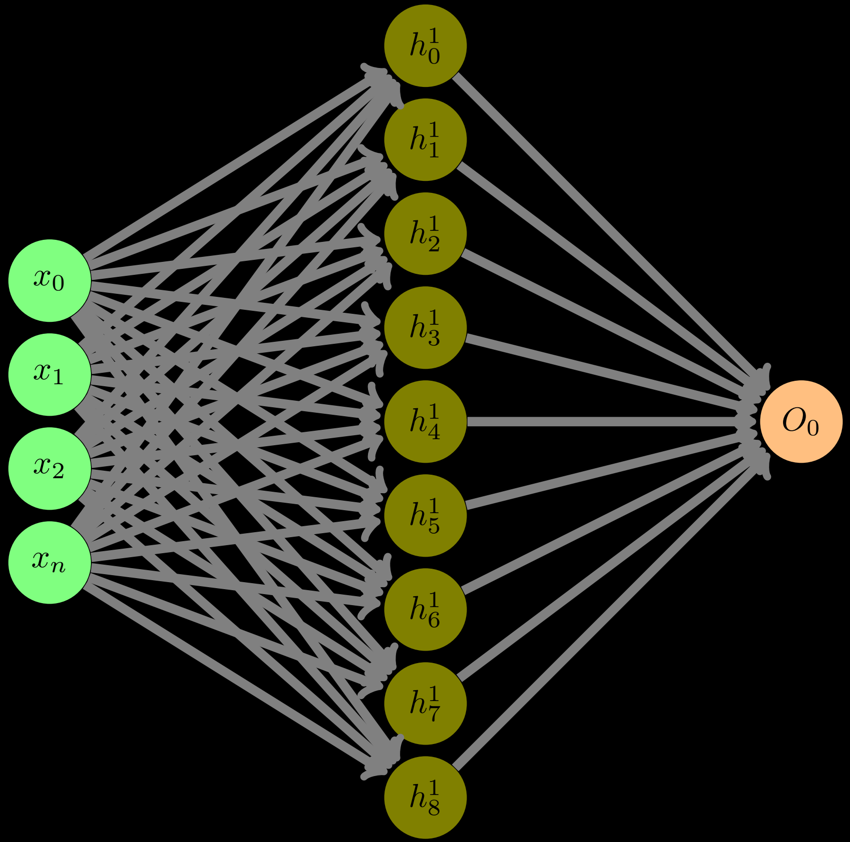
 

Perceptron simplified

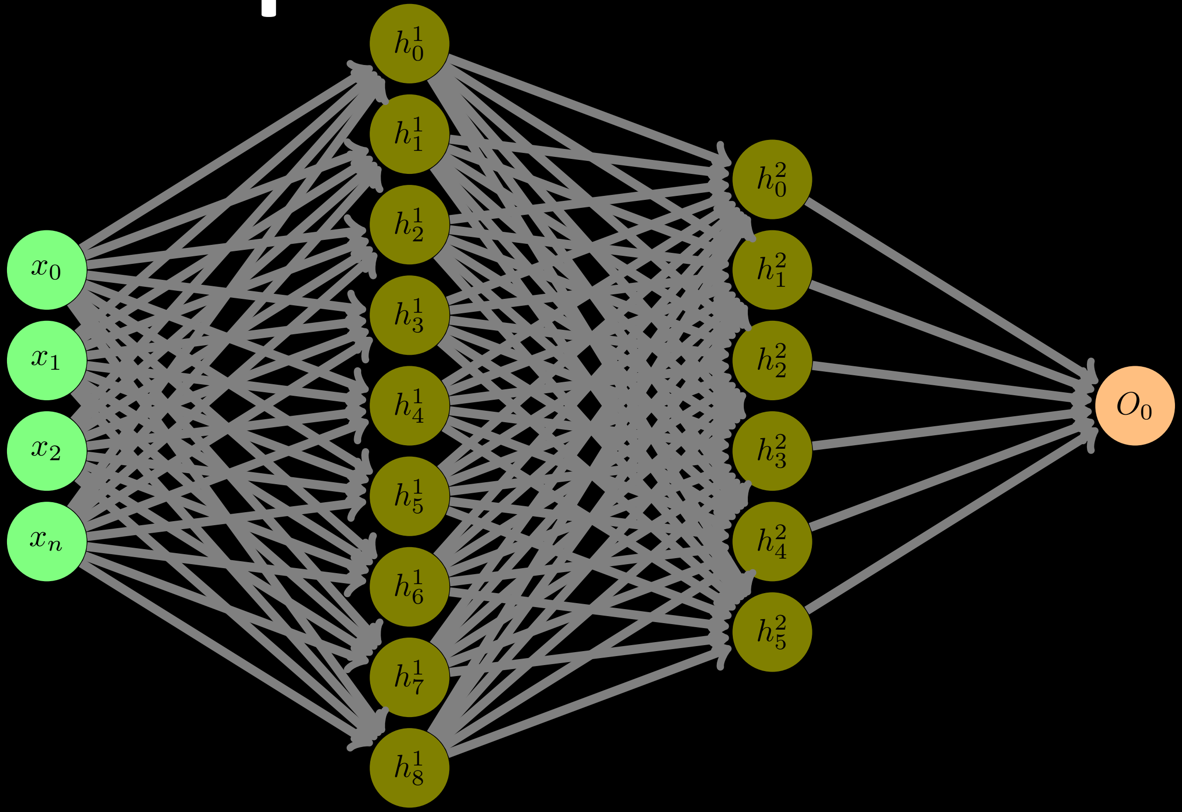




Multi-layer perceptron



Deep neural network



Number of hidden layers > 1

Outline

- Perceptron and deep neural networks
- Training deep neural networks
- Improving training

Supervised deep learning

Inputs Outputs


Cat


Dog


Horse

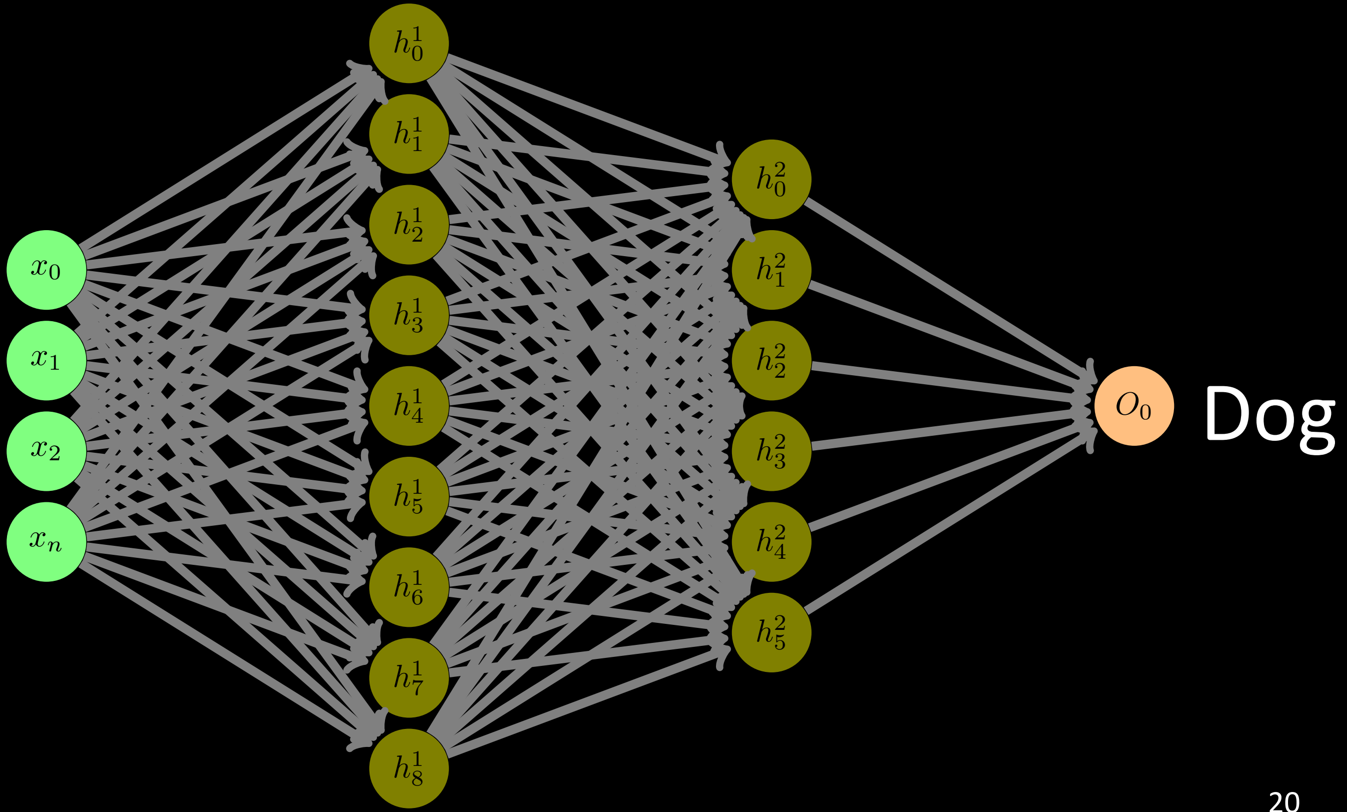

Elephant


Tiger

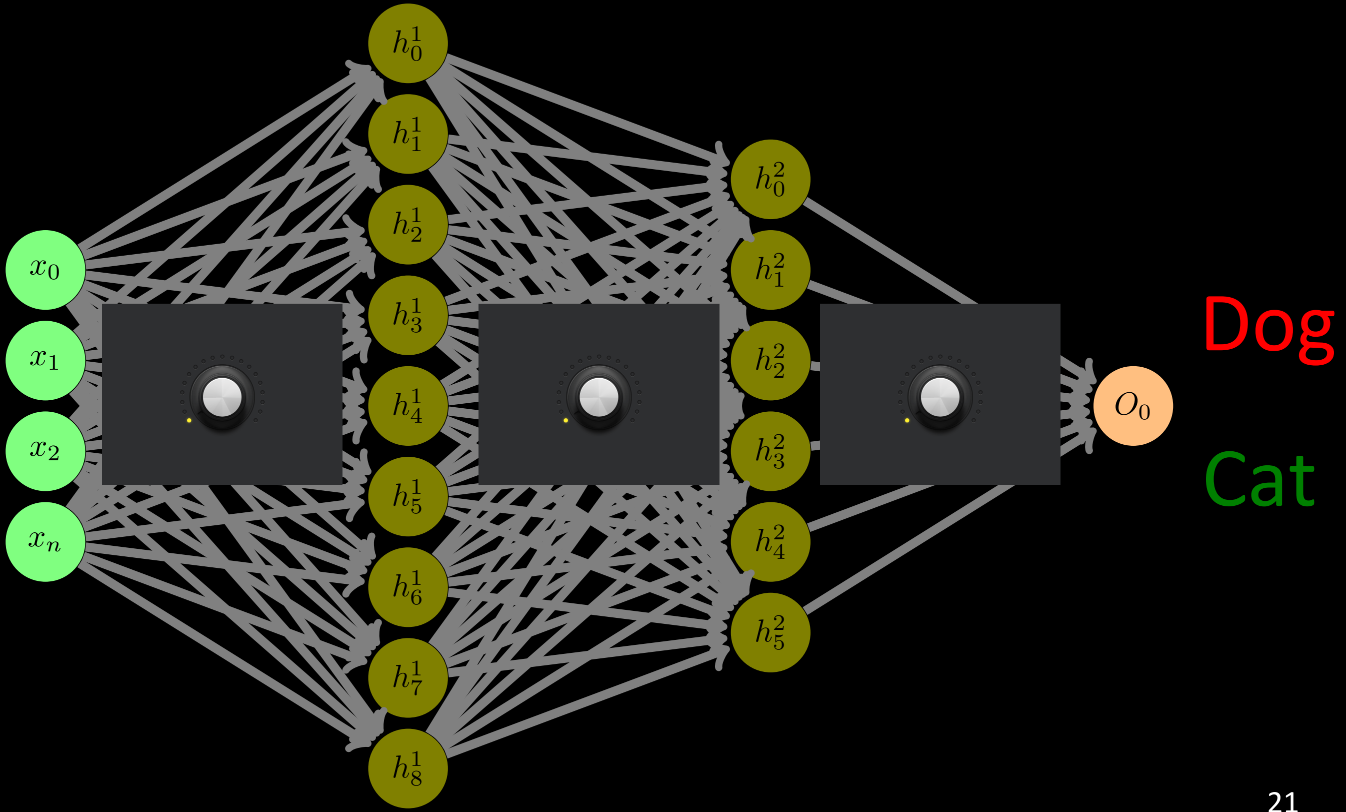
Training

?

Training: forward pass



Training: backward pass



Quantifying error (loss)

i^{th} training instance (e.g. cat image)

network parameters

$$loss(\underbrace{f(x^{(i)};\theta)}_{\text{Predicted (Dog)}}, \underbrace{y^{(i)}}_{\text{Observed (Cat)}})$$

Predicted
(Dog)

Observed
(Cat)

Quantifying error (loss)

network parameters

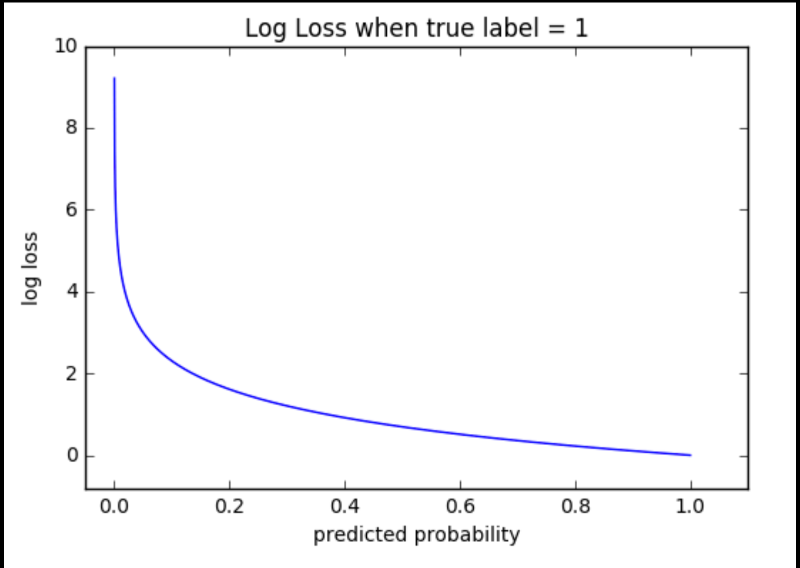
total training instances

i^{th} training instance

$$total\ loss := J(\theta) = \frac{1}{N}\sum_{i=1}^{N} loss(f(\underbrace{x^{(i)}}_{\text{Predicted}};\theta), \underbrace{y^{(i)}}_{\text{Observed}})$$

Cross entropy loss

Measure loss of ***a classification model*** whose ***output is a probability value between 0 and 1***



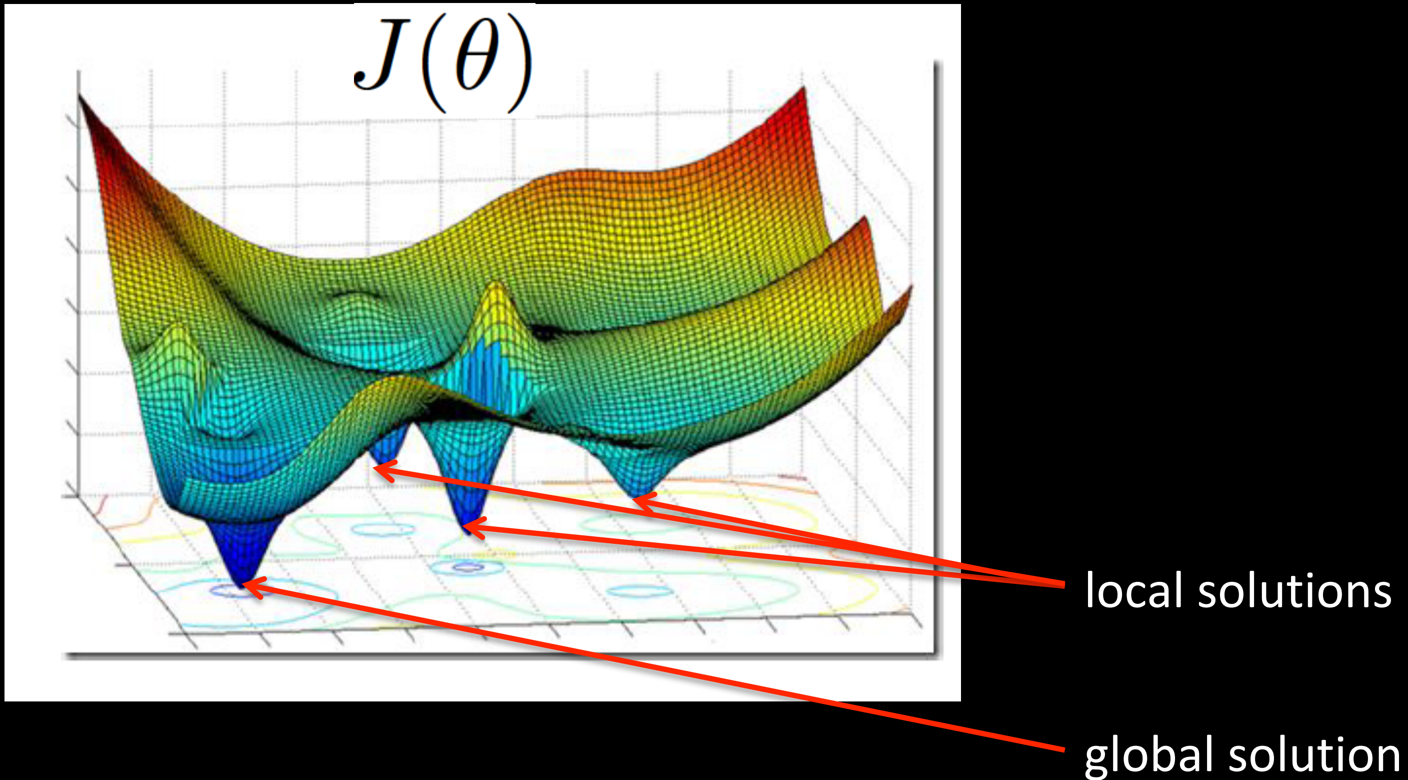
$$Cross\ Entropy(\theta) := J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y^{(i)}\ log(f(x^{(i)};\theta) + (1-y^{(i)})\ log(1-f(x^{(i)};\theta))$$

Training neural networks: objective

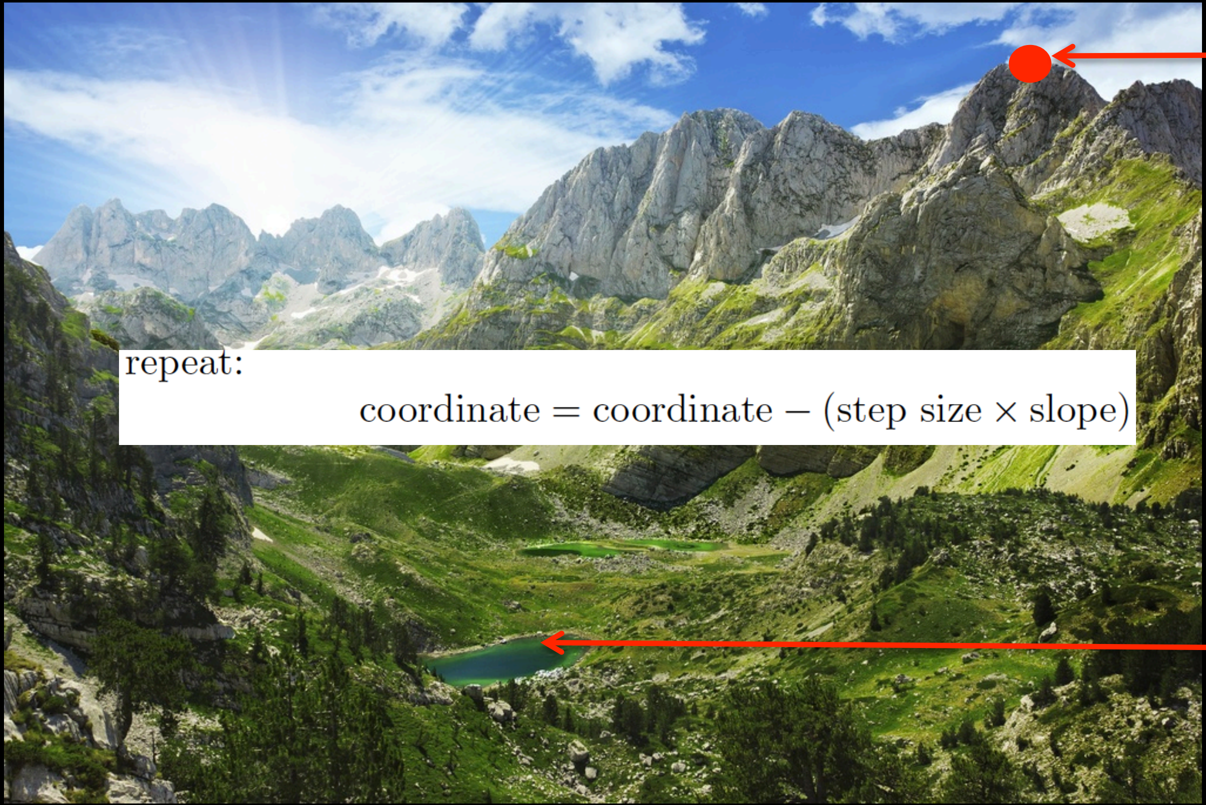
$$\underbrace{\arg_\theta \min \frac{1}{N} \sum_{i=1}^{N} loss(f(x^{(i)};\theta), y^{(i)})}_{J(\theta)}$$

How to minimize? $J(\theta)$ $\theta = W_1, W_2, \ldots, W_n$

Training neural networks: objective



Gradient descent



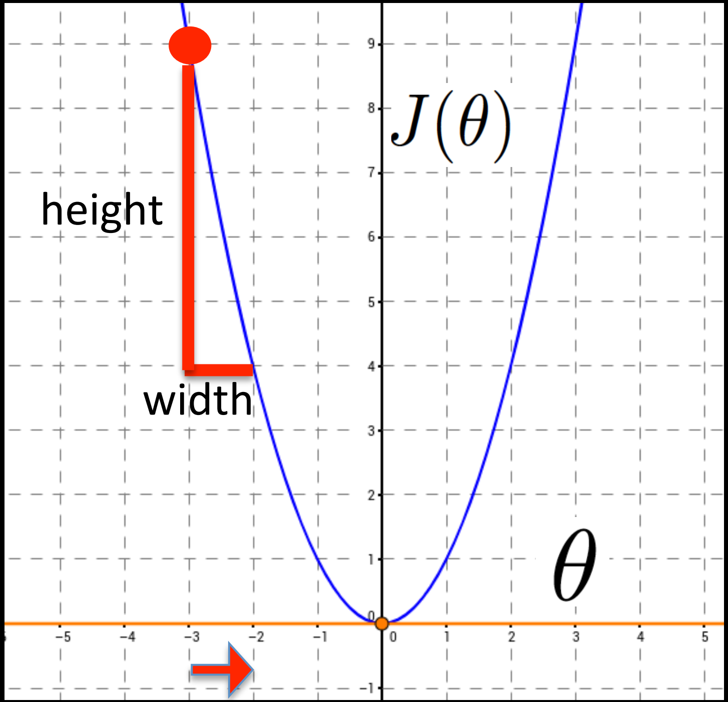
repeat:

$$\text{coordinate} = \text{coordinate} - (\text{step size} \times \text{slope})$$

You are here!

You want to go here!

Gradient descent



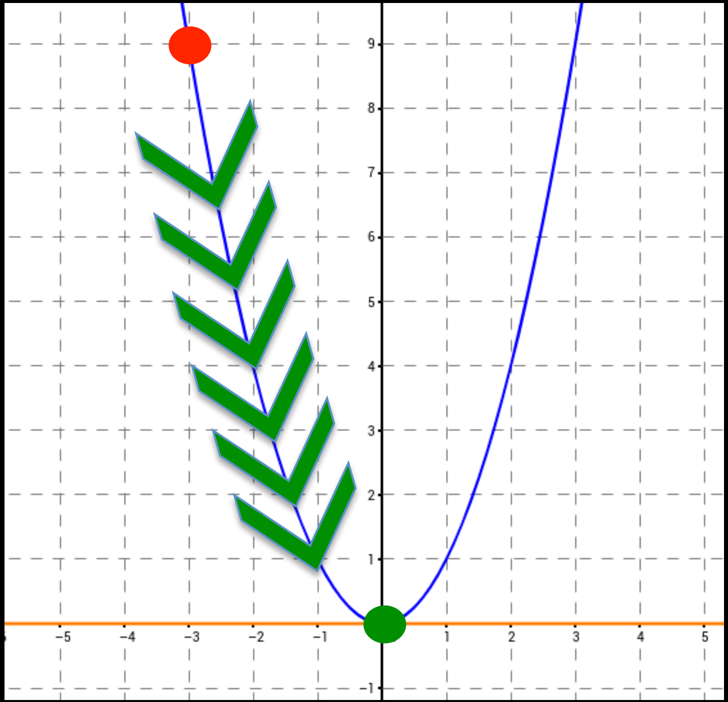
repeat:

$$\text{coordinate} = \text{coordinate} - (\text{step size} \times \text{slope})$$

$$\frac{\partial J(\theta)}{\partial \theta} = \text{slope} = \frac{\text{height}}{\text{width}}$$

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

Gradient descent



repeat:

$$\text{coordinate} = \text{coordinate} - (\text{step size} \times \text{slope})$$

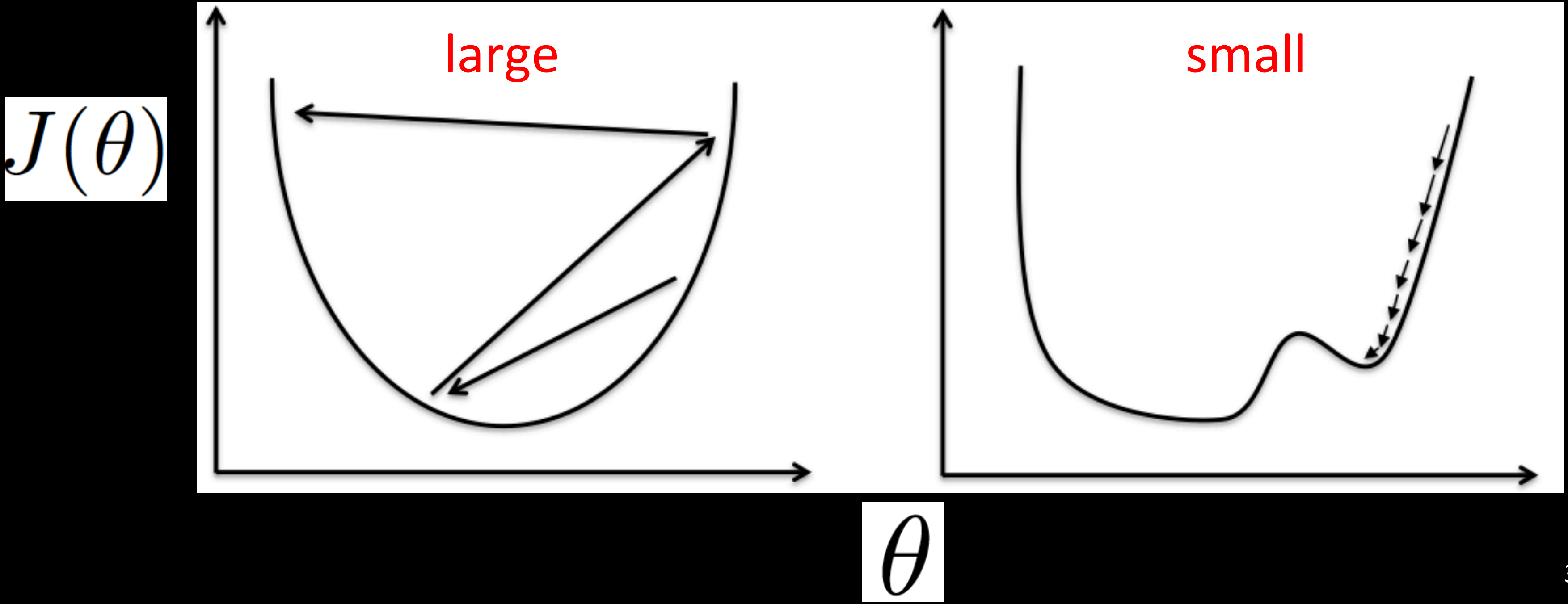
$$\frac{\partial J(\theta)}{\partial \theta} = \text{slope} = \frac{\text{height}}{\text{width}}$$

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

Gradient descent

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

learning rate



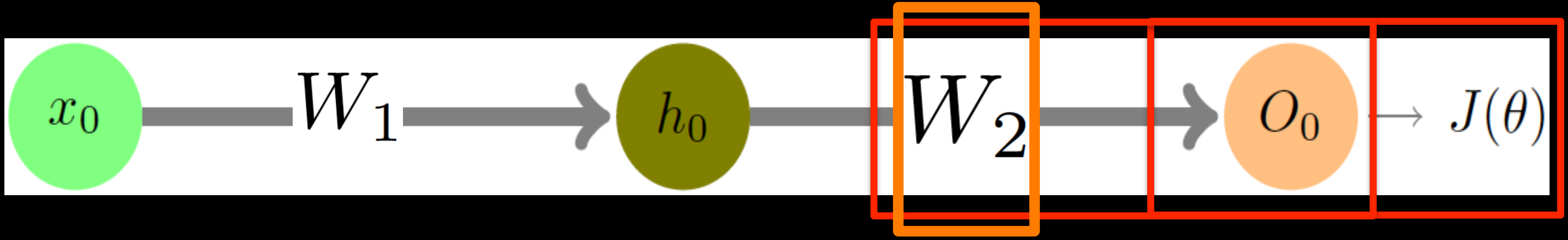
Stochastic gradient descent

- Initialize θ randomly
- For N epochs
  - For each training example (x, y):
    - compute loss gradient: $\frac{\partial J(\theta)}{\partial \theta}$
    - update θ with update rule:
      $\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$

Mini-batch gradient descent

- Initialize θ randomly
- For N epochs
  - For each batch of training examples $\{(x_0, y_0), \ldots, (x_b, y_b)\}$:
    - compute loss gradient: $\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{B} \sum_{i=1}^{N} \frac{\partial J^i(\theta)}{\partial \theta}$
    - update θ with update rule:
      $\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$

Backpropagation



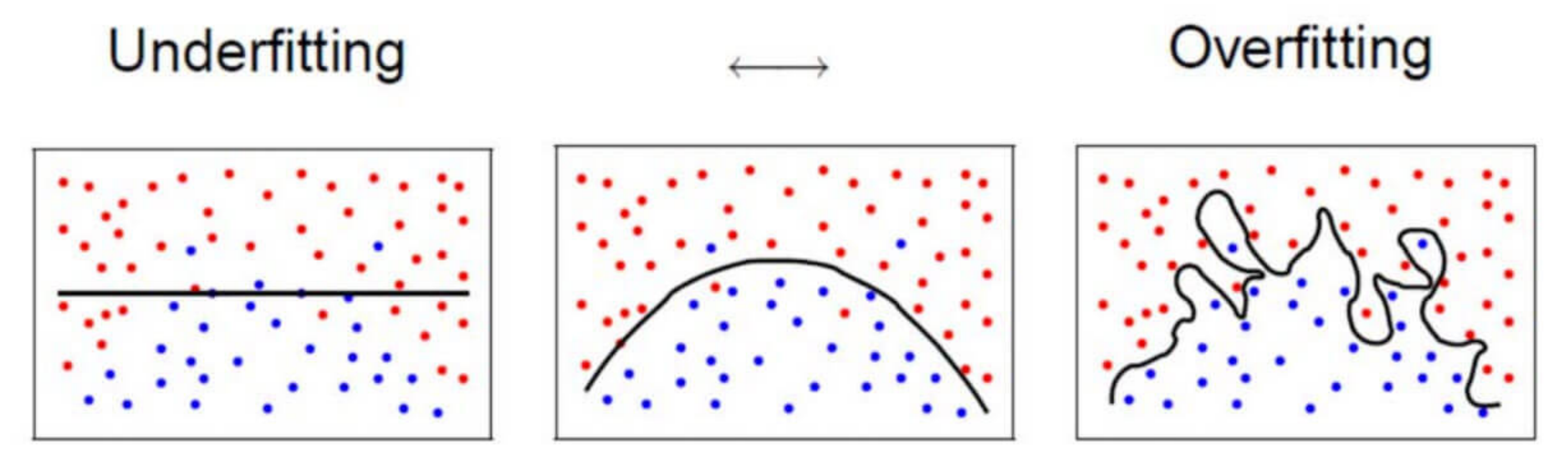
Chain rule

$$\frac{\partial J(\theta)}{\partial W_2} = \boxed{\frac{\partial J(\theta)}{\partial o_0}} \times \boxed{\frac{\partial o_0}{\partial W_2}}$$

Outline

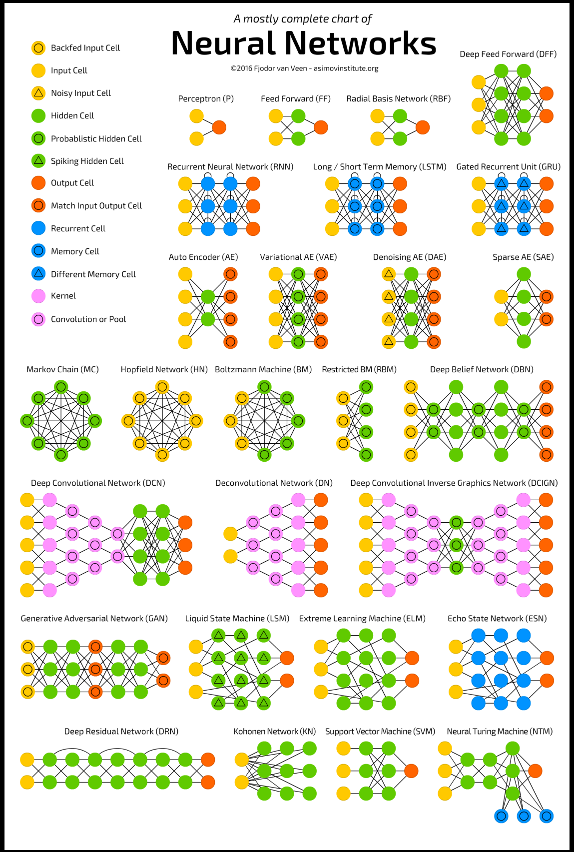
- Perceptron and deep neural networks
- Training deep neural networks
- Improving training

Improving training



How to avoid under fitting?

- Increase the number of units/layer and hidden layers
- Use appropriate network
  - Convolutional network for images
  - Recurrent network for sequences
- Within the same network
  - Experiment with hyper parameters
    - Activation functions, optimizers, batch size

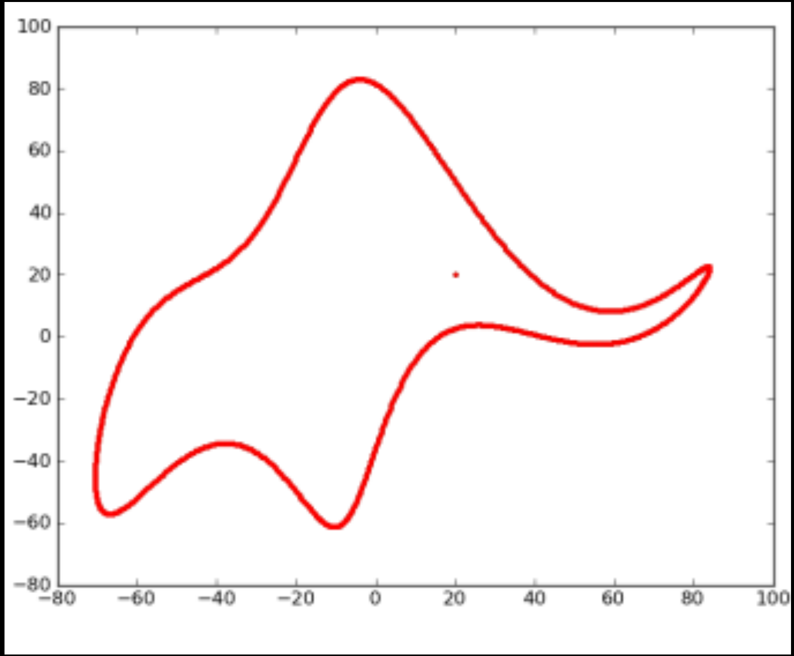


How to avoid over fitting?

"With four parameters I can fit an elephant, and with five I can make him wiggle his trunk"

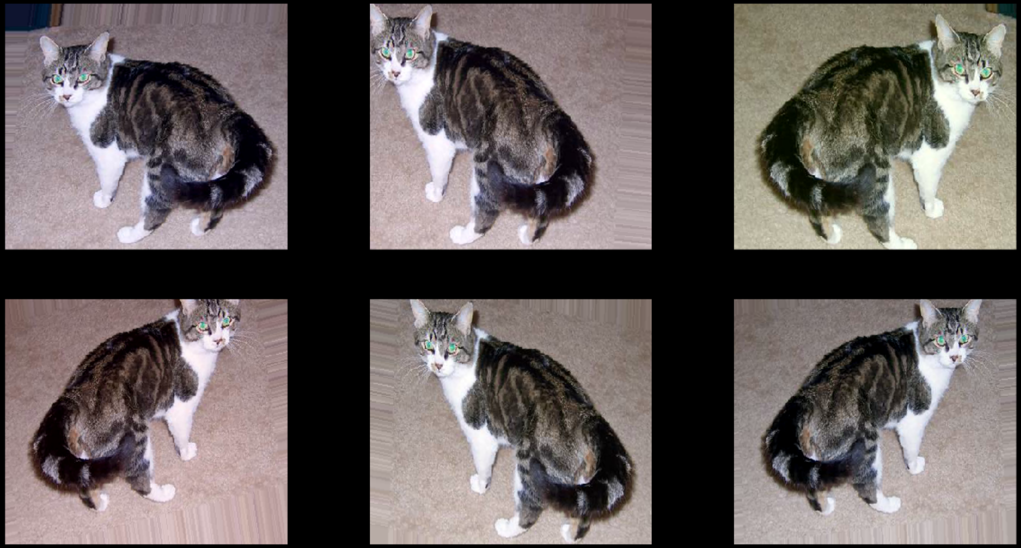
---Enrico Fermi

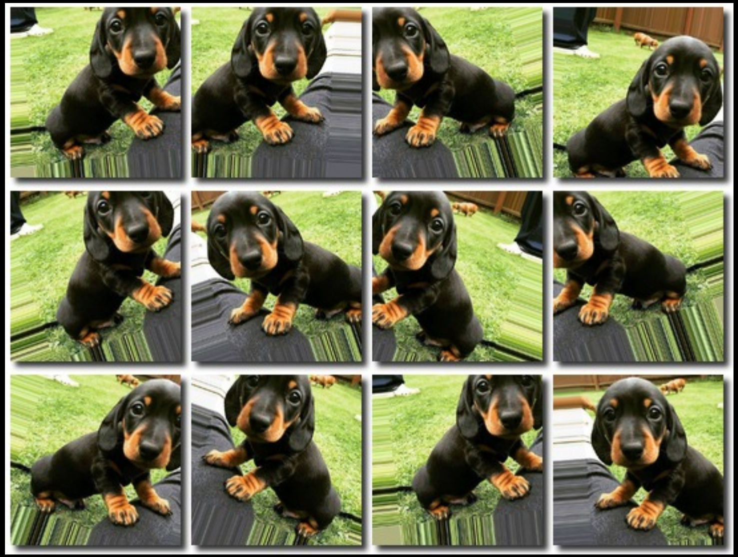
Deep neural networks: 10^6 to 10^9 parameters!



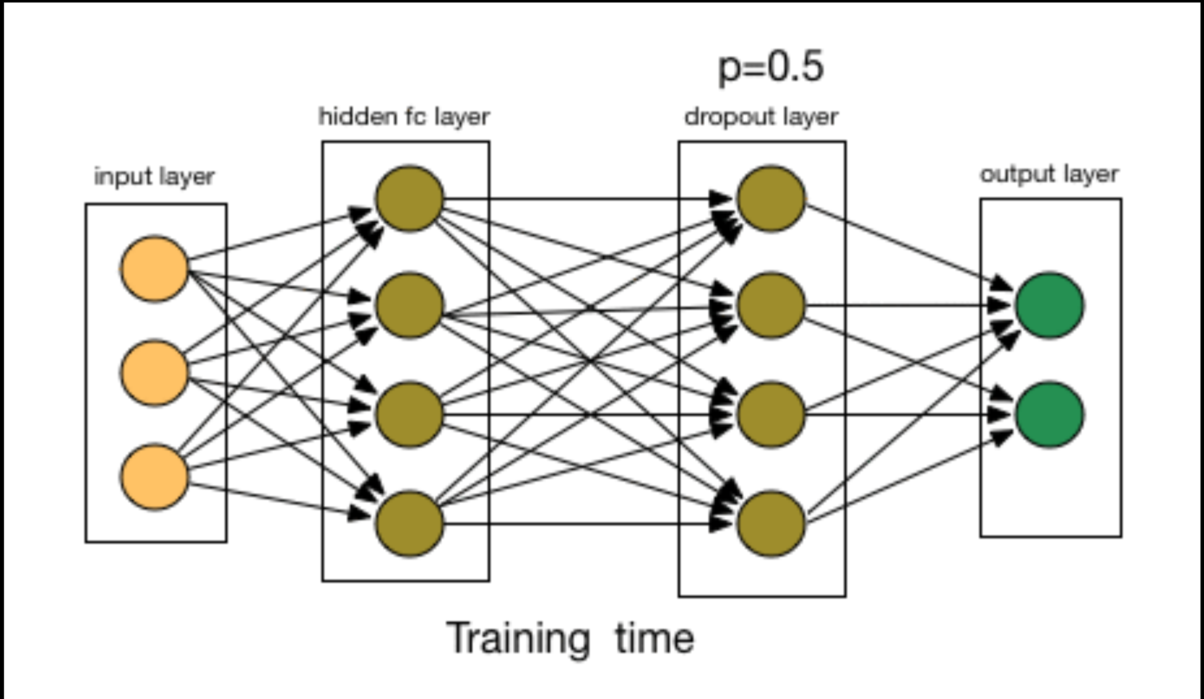
Mayer et. al., *Drawing an elephant with four complex parameters*, Am. J. Phys. 78, 648 (2010)

Data augmentation

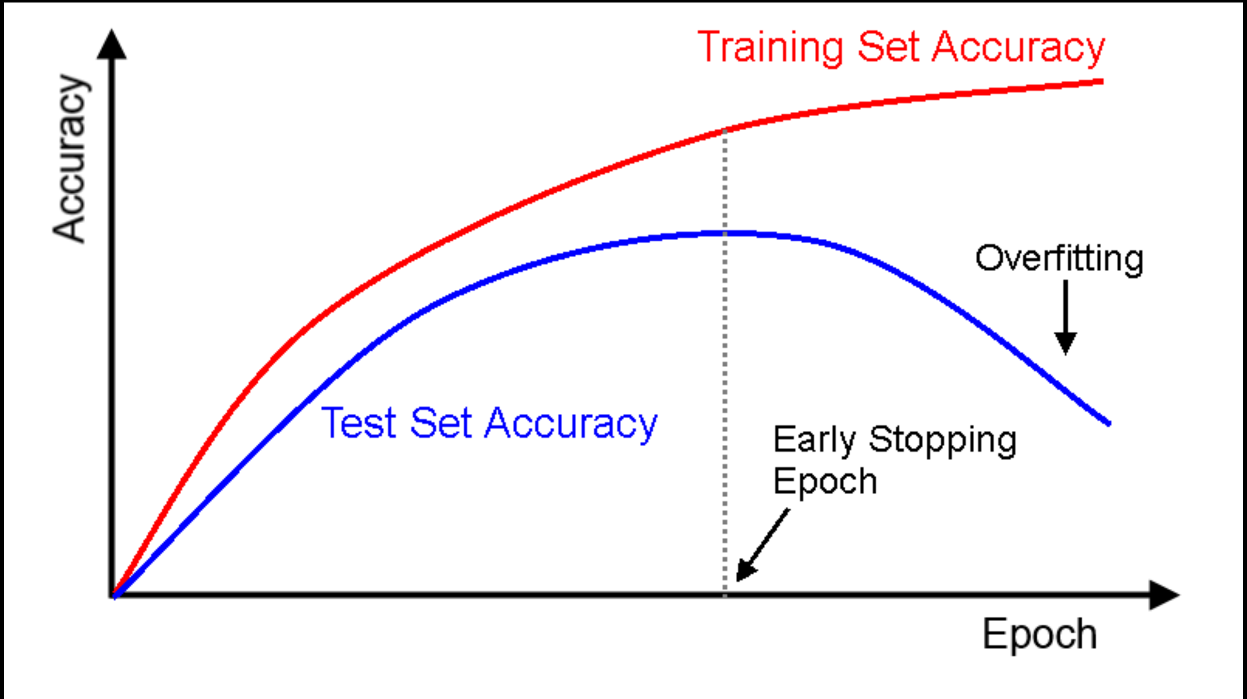




Dropout


p=0.5
hidden fc layer
dropout layer
input layer
output layer
Training time

Early stopping



Regularization

$$\frac{1}{N}\sum_{i=1}^{N} loss(f(x^{(i)};\theta), y^{(i)}) + \lambda \sum_{m=1}^{p} (\theta_i)^2$$

Summary

- Perceptron
- Perceptron to neural networks
- Forward pass
- Backward propagation with gradient descent
- Under fitting and over fitting

Questions

https://imgs.xkcd.com/comics/machine_learning.png