

# Data Science: Principles and Practice

## Lecture 4: Deep Learning, Part I

Ekaterina Kochmar<sup>1</sup>

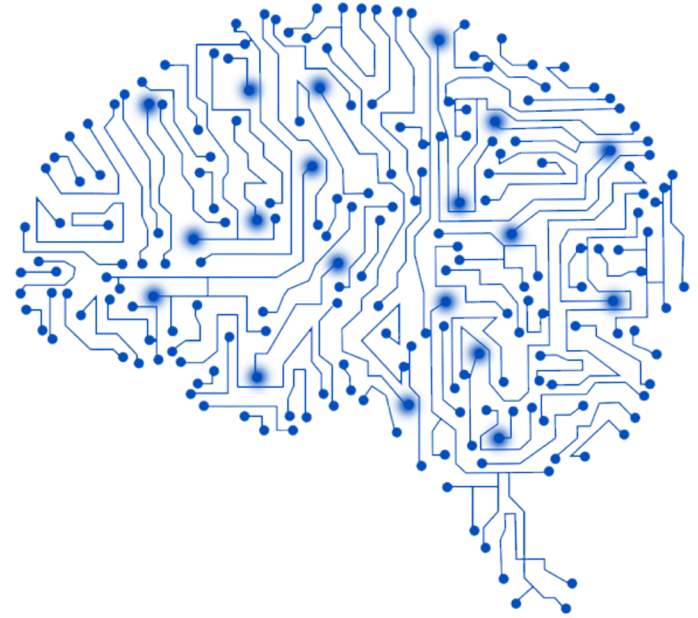


UNIVERSITY OF  
CAMBRIDGE

---

<sup>1</sup> Based on slides by Marek Rei

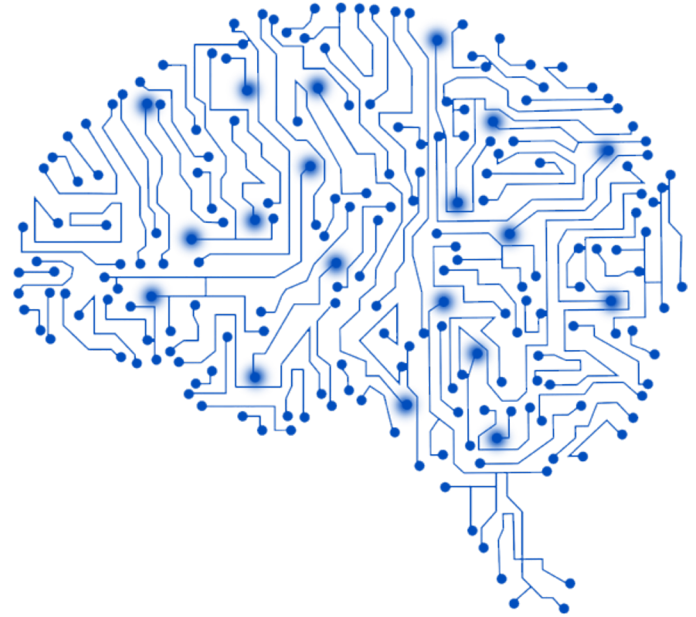
# What is Deep Learning?



# What is Deep Learning?

Deep learning is a class of machine learning algorithms.

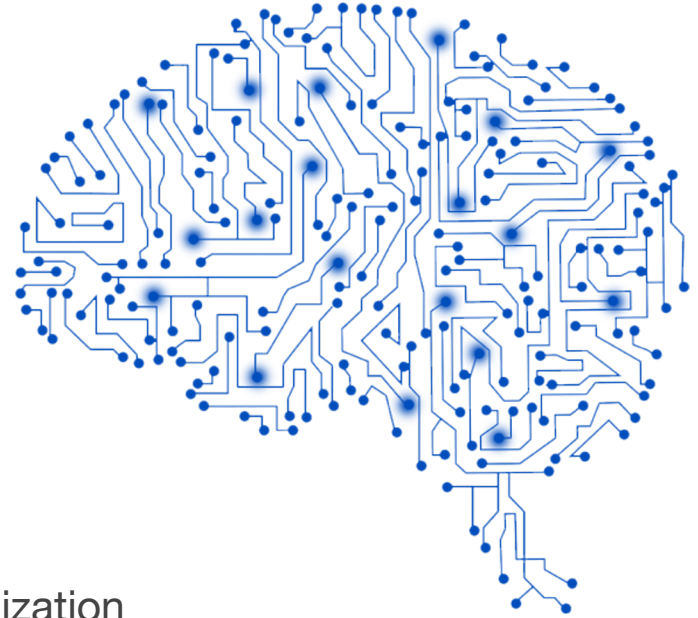
Neural network models with multiple hidden layers.



# What is Deep Learning?

Deep learning is a class of machine learning algorithms.

Neural network models with multiple hidden layers.



**Today:** The basics of neural network models, optimization

**Next lecture:** Implementing models with Tensorflow, network components, practical tips

# Data Science: Principles and Practice

01

Introduction and motivation

02

Fundamentals of Neural Networks

03

Neural Network Optimization

# The Rise of Deep Learning

## Microsoft's voice-recognition tech is now better than even teams of humans at transcribing conversations

 **Matt Weinberger** ✉️ 🐦  
Aug. 21, 2017, 7:30 PM 🔥 667

 FACEBOOK  LINKEDIN  TWITTER  

*Follow Business Insider:*

In October 2016, in a big milestone for artificial intelligence, [Microsoft](#)



# The Rise of Deep Learning

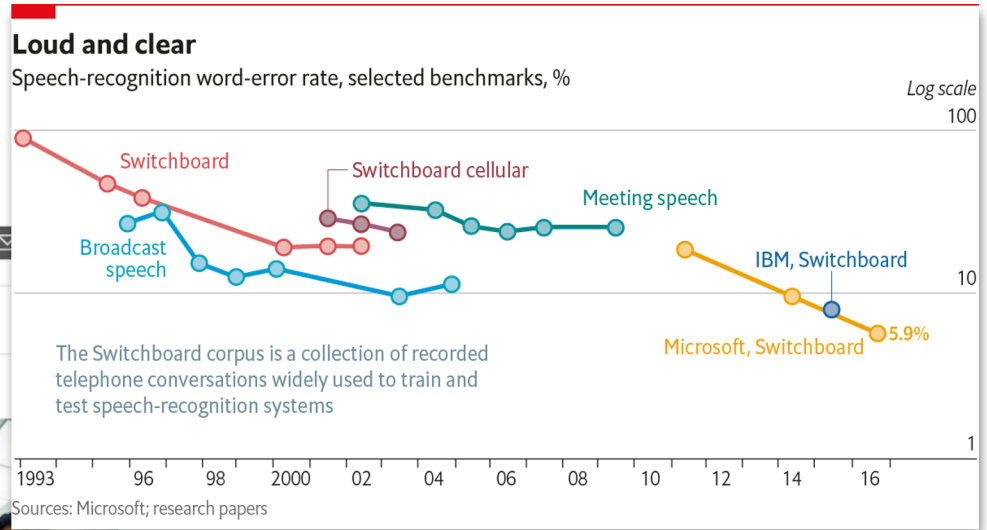
## Microsoft's voice-recognition tech is now better than even teams of humans at transcribing conversations

 **Matt Weinberger**    
Aug. 21, 2017, 7:30 PM  667

 FACEBOOK  LINKEDIN  TWITTER

Follow Business Insider:

In October 2016, in a big milestone for artificial intelligence, Microsoft



# The Rise of Deep Learning

AI

## Google taps neural nets for better offline translation in 59 languages

KHARI JOHNSON @KHARIJOHNSON JUNE 12, 2018 10:16 AM

Above: Google Translate for iOS.  
Image Credit: Jordan Novet / VentureBeat

Google's online translations have been powered by neural machine translation (NMT) since 2016, and today the company is rolling out its neural net-driven approach to more accurate, natural-sounding translations for Google Translate iOS and Android app users to carry out translations offline in 59 languages.

Offline NMT was made by the Translate team in conjunction with the Google Brain team using TensorFlow, Google product manager Julie Cattiau told VentureBeat in a phone interview. Unlike for other Google apps, 95 percent of Google Translate's user base is outside the United States, in countries like India, Brazil, and Indonesia, Cattiau said.



# The Rise of Deep Learning

AI

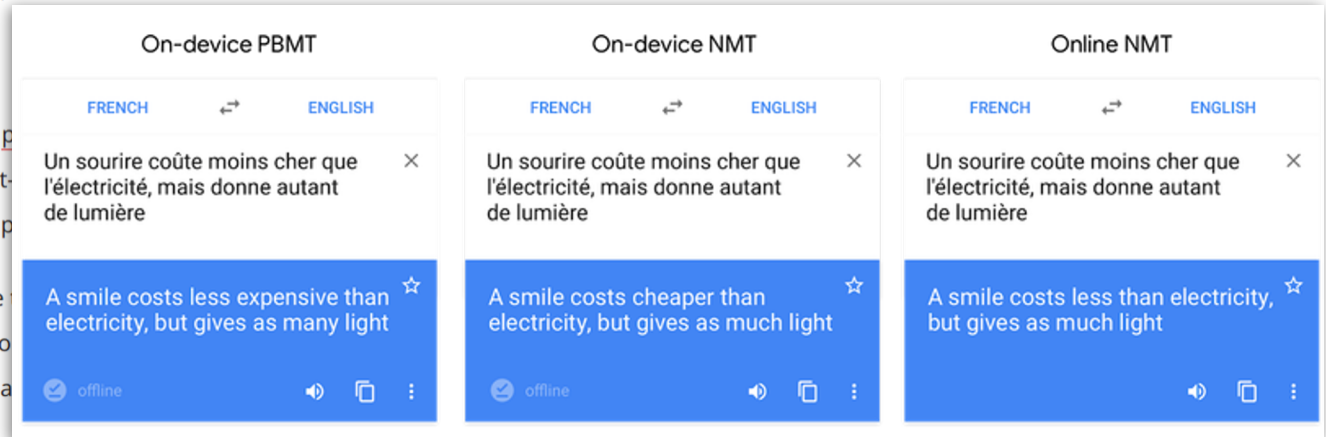
## Google taps neural nets for better offline translation in 59 languages

KHARI JOHNSON @KHARIJOHNSON JUNE 12, 2018 10:16 AM

Above: Google Translate for iOS.  
Image Credit: Jordan Novet / VentureBeat

Google's online translations have been p...  
the company is rolling out its neural net...  
for Google Translate iOS and Android ap...

Offline NMT was made by the Translate...  
Google product manager Julie Cattiau to...  
95 percent of Google Translate's user ba...  
Indonesia, Cattiau said.



# The Rise of Deep Learning

Artificial intelligence / Machine learning

---

## Facebook Creates Software That Matches Faces Almost as Well as You Do

Facebook's new AI research group reports a major improvement in face-processing software.

by **Tom Simonite**

March 17, 2014

---

**Asked whether two unfamiliar photos of faces show the same person, a human being will get it right 97.53 percent of the time. New software developed by researchers at Facebook can score 97.25 percent on the same challenge, regardless of variations in lighting or whether the person in the picture is directly facing the camera.**

<https://www.technologyreview.com/2014/03/17/13822/facebook-creates-software-that-matches-faces-almost-as-well-as-you-do/>

PUBLICATION

## DeepFace: Closing the Gap to Human-Level Performance in Face Verification

Conference on Computer Vision and Pattern Recognition (CVPR)

<https://research.fb.com/publications/deepface-closing-the-gap-to-human-level-performance-in-face-verification/>

# The Rise of Deep Learning



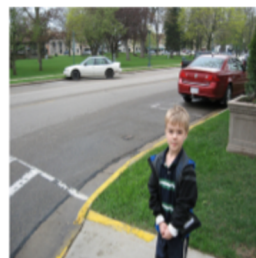
there is a cat sitting on a shelf .



a plate with a fork and a piece of cake .



a black and white photo of a window .



a young boy standing on a parking lot next to cars .



a wooden table and chairs arranged in a room .



a kitchen with stainless steel appliances .



this is a herd of cattle out in the field .



a car is parked in the middle of nowhere .

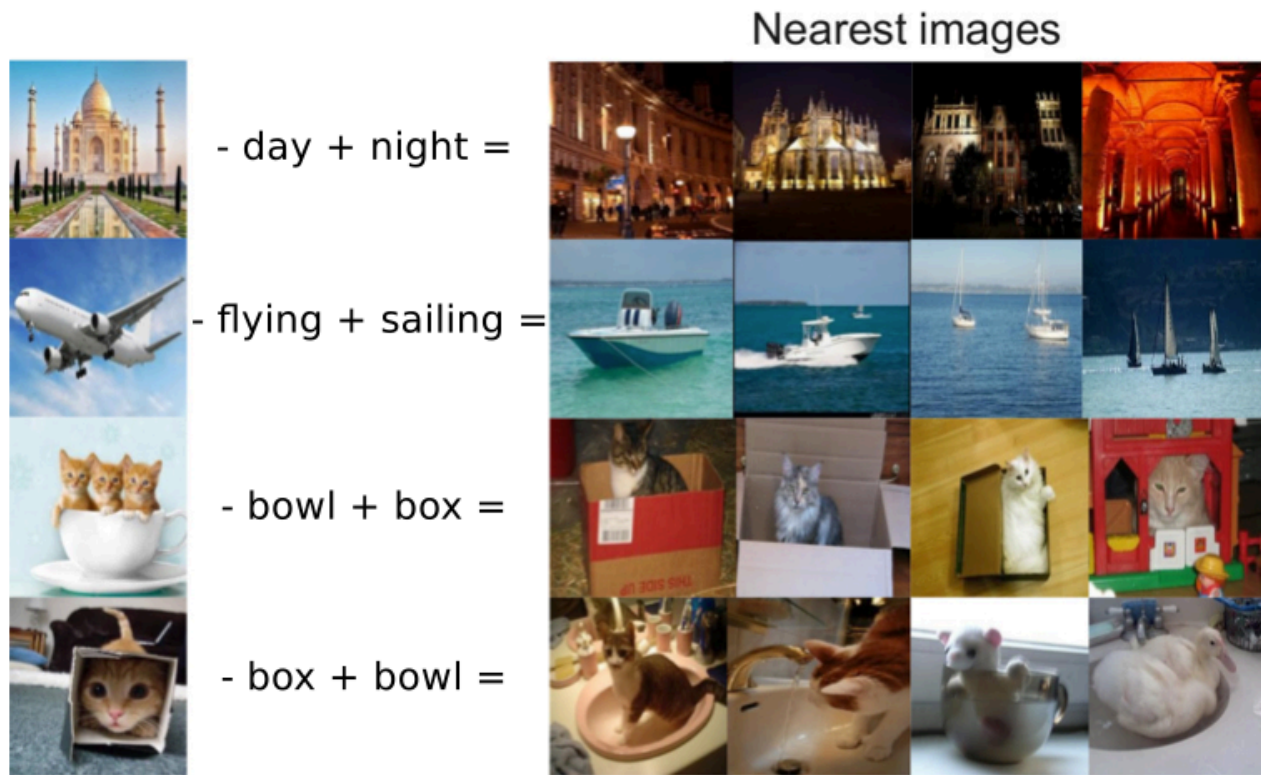


a ferry boat on a marina with a group of people .



a little boy with a bunch of friends on the street .

# The Rise of Deep Learning



# The Rise of Deep Learning



SUBSCRIPTIONS



SIGN IN ▾

TECH —

## Google's AlphaGo AI beats world's best human Go player

Ke Jie tried to use AlphaGo's own moves and lost.

SEBASTIAN ANTHONY - 5/23/2017, 2:20 PM

STR/AFP/Getty Images

[Enlarge](#) / China's 19-year-old Go player Ke Jie (L) prepares to make a move during the first match against Google's artificial intelligence program AlphaGo in Wuzhen, east China's Zhejiang province on May 23, 2017.

DeepMind's AlphaGo AI has defeated Ke Jie in the first round of a best-of-three Go match in China. A video of the match is embedded below. Ke Jie was defeated by just a half a point—the closest margin possible—but scoring versus AlphaGo is a little bit disingenuous: DeepMind's AI doesn't try to win by a large margin; it just plots the surest route to victory, even if it's only by half a point.

Ke Jie is generally considered to be the world's best human Go player, but he wasn't expected to win; AlphaGo defeated the Chinese 19-year-old earlier in the year during [an unbeaten online 60-match victory streak](#).

# The Rise of Deep Learning

ars TECHNICA

SUBSCRIPTIONS



SIGN IN ▾

TECH —

## Google's AlphaGo AI beats world's best human Go player

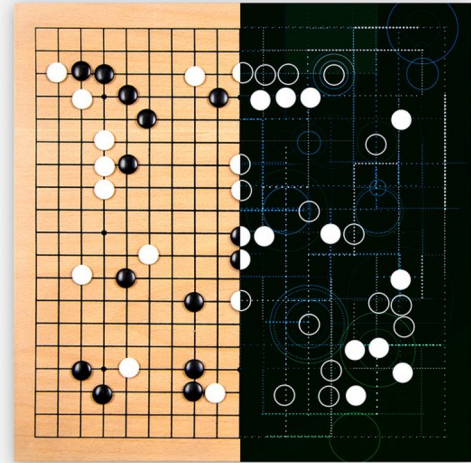
Ke Jie tried to use AlphaGo's own moves and lost.

SEBASTIAN ANTHONY - 5/23/2017, 2:20 PM

**Enlarge** / China's 19-year-old Go player Ke Jie (L) prepares to make a move during the first match against Google's artificial AlphaGo in Wuzhen, east China's Zhejiang province on May 23, 2017.

DeepMind's AlphaGo AI has defeated Ke Jie in the first round of a best-of-three Go match in China. A video is embedded below. Ke Jie was defeated by just a half a point—the closest margin possible—but scoring versus a disingenuous: DeepMind's AI doesn't try to win by a large margin; it just plots the surest route to victory, even if it means a narrow point.

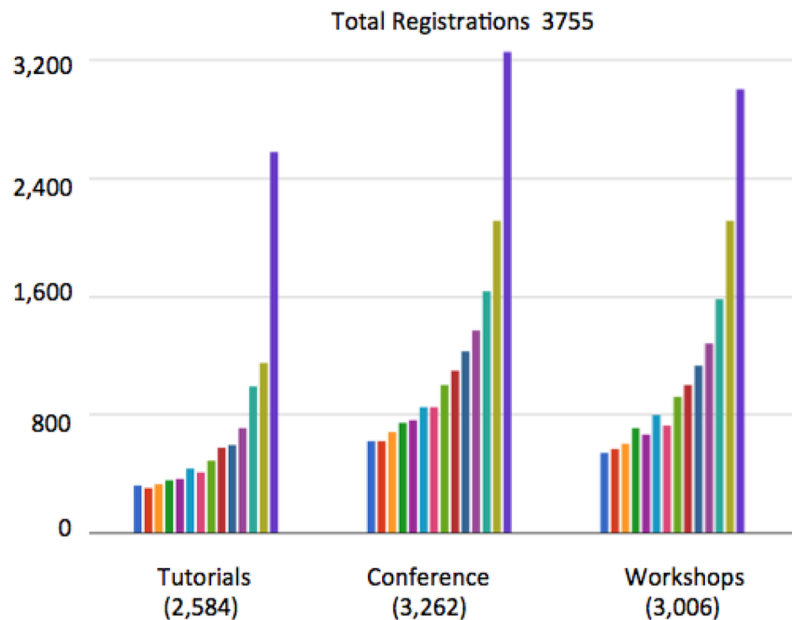
Ke Jie is generally considered to be the world's best human Go player, but he wasn't expected to win; AlphaGo had beaten the Chinese 19-year-old earlier in the year during [an unbeaten online 60-match victory streak](#).



# The Rise of Deep Learning

Conference on Neural Information Processing Systems (NeurIPS, formerly NIPS) – one of the main conferences on deep learning and machine learning.

## NIPS Growth



**NIPS**  
@NipsConference

Following

**#NIPS2018** The main conference sold out in 11 minutes 38 seconds

9:17 AM - 4 Sep 2018

695 Retweets 1,076 Likes



81 695 1.1K

# The Hype Train of Deep Learning



This guy didn't know  
about neural networks  
(a.k.a deep learning)



This guy learned  
about neural networks  
(a.k.a deep learning)

<http://deeplearning.cs.cmu.edu>

- “Deep learning” is often used as a buzzword, even without understanding it.
- Be mindful - it’s a powerful class of machine learning algorithms, but not a magic solution to every problem.
- Deep Learning is particularly successful in solving complex problems: breakthroughs in natural language processing, computer vision, board game programs.



# But Why Now?



2012 - AlexNet wins  
ImageNet, Krizhevsky

2006 - Restricted Boltzmann  
Machine, Hinton

1998 - ConvNets for OCR,  
LeCun

1997 - LSTM, Hochreiter &  
Schmidhuber

1974 - backpropagation,  
Werbos

1958 - perceptrons,  
Rosenblatt

# But Why Now?

2012 - AlexNet wins  
ImageNet, Krizhevsky

2006 - Restricted Boltzmann  
Machine, Hinton

1998 - ConvNets for OCR,  
LeCun

1997 - LSTM, Hochreiter &  
Schmidhuber

1974 - backpropagation,  
Werbos

1958 - perceptrons,  
Rosenblatt

The theory was there before, but the conditions are now better for putting it into action.

## 1. Big Data

- Large datasets for training
- Better methods for storing and managing data



WIKIPEDIA  
The Free Encyclopedia

# But Why Now?

2012 - AlexNet wins ImageNet, Krizhevsky

2006 - Restricted Boltzmann Machine, Hinton

1998 - ConvNets for OCR, LeCun

1997 - LSTM, Hochreiter & Schmidhuber

1974 - backpropagation, Werbos

1958 - perceptrons, Rosenblatt

The theory was there before, but the conditions are now better for putting it into action.

## 1. Big Data

- Large datasets for training
- Better methods for storing and managing data



WIKIPEDIA  
The Free Encyclopedia

## 2. Faster Hardware

- Graphics Processing Units (GPUs)
- Faster CPUs
- More affordable



# But Why Now?

2012 - AlexNet wins ImageNet, Krizhevsky

2006 - Restricted Boltzmann Machine, Hinton

1998 - ConvNets for OCR, LeCun

1997 - LSTM, Hochreiter & Schmidhuber

1974 - backpropagation, Werbos

1958 - perceptrons, Rosenblatt

The theory was there before, but the conditions are now better for putting it into action.

## 1. Big Data

- Large datasets for training
- Better methods for storing and managing data



WIKIPEDIA  
The Free Encyclopedia

## 2. Faster Hardware

- Graphics Processing Units (GPUs)
- Faster CPUs
- More affordable

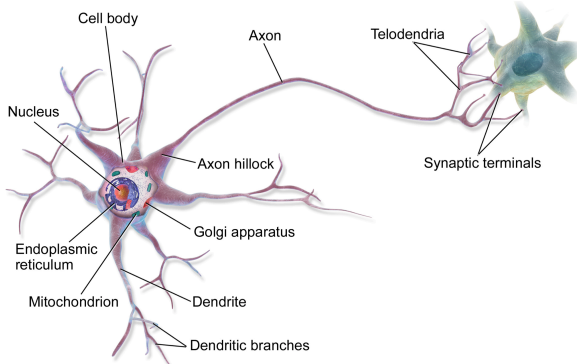
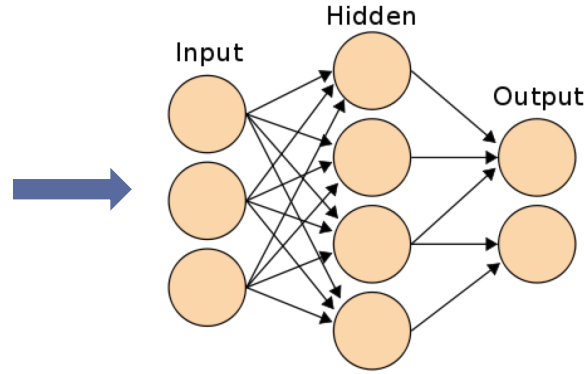
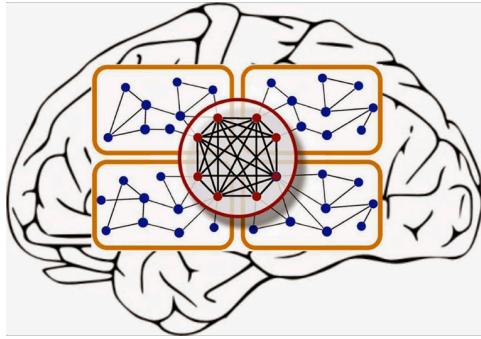


## 3. Better Software

- Better Optimization Algorithms
- Automatic Differentiation Libraries



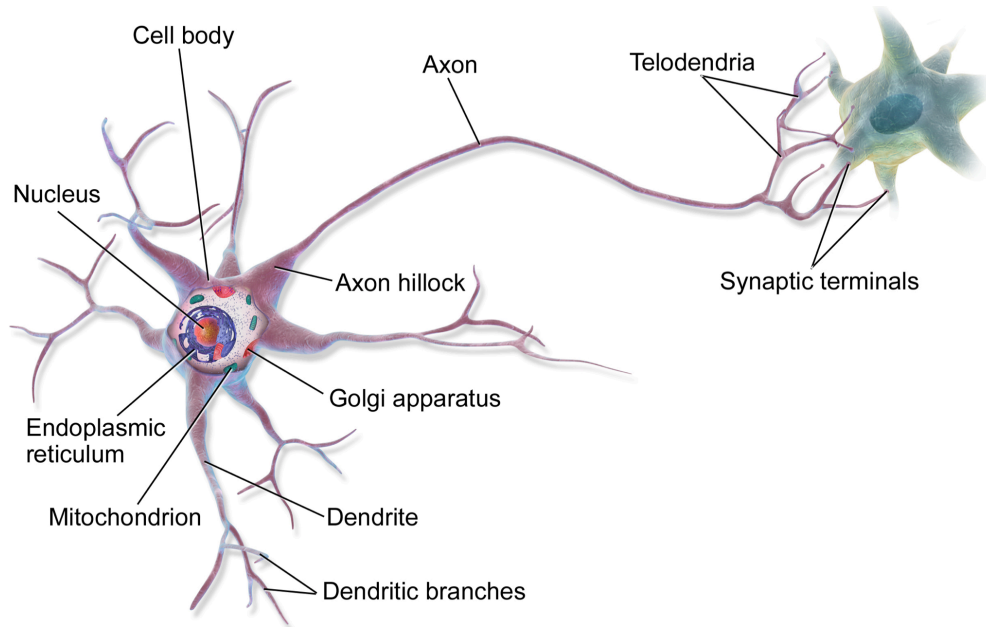
# Biological Inspiration



<https://en.wikipedia.org/wiki/Neuron>

- Often, artificial neural networks (ANNs) are said to be (loosely) based on biological neural networks.
- For instance, Perceptron algorithm was largely inspired by *Hebb's rule*: “Cells that fire together, wire together”.
- Inspiration doesn't mean exact copy: there are notable differences between the two.

# Biological Inspiration



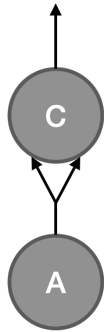
*“If the human brain were  
so simple that we  
could understand it, we would be  
so simple that we couldn’t.”*

*Emerson W. Pugh*

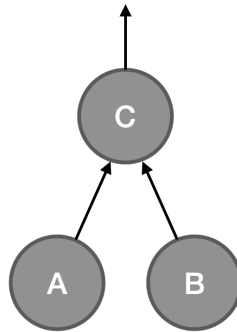
# Fundamentals of Neural Networks

# Simple artificial neuron

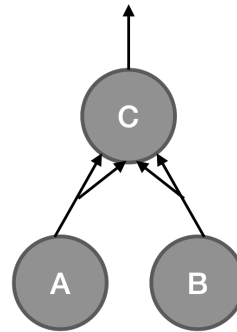
- Threshold Logic Unit, or Linear Threshold Unit, by McCulloch and Pitts<sup>1</sup>
- Has one or more binary (on/off) inputs and one binary output
- Activates its output when more than a certain number of its inputs are active
- Even with such a simplified model it is possible to build a network of artificial neurons that computes any logical proposition



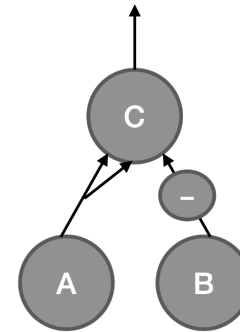
$$C = A$$



$$C = A \wedge B$$



$$C = A \vee B$$



$$C = A \wedge \neg B$$

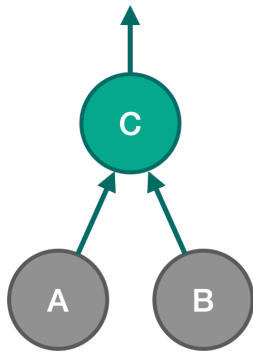


# Simple artificial neuron

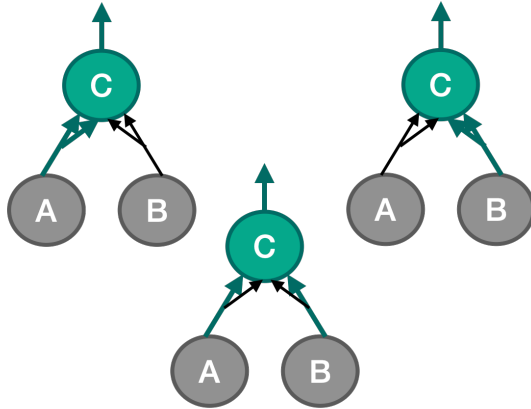
Assume that a neuron is activated when at least two of its inputs are active:



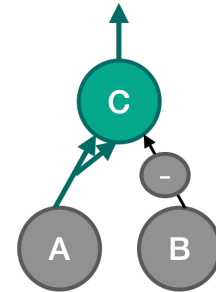
$$C = A$$



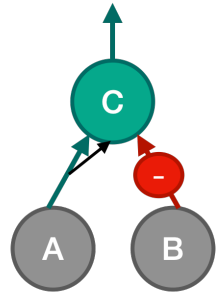
$$C = A \wedge B$$



$$C = A \vee B$$



$$C = A \wedge \neg B$$



# Simple artificial neuron

## Quiz time:

Can you draw an ANN that computes  $A \oplus B$ , where  $\oplus$  is the XOR operation,

$$\text{i.e. } A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)?$$

# Recap: Perceptron

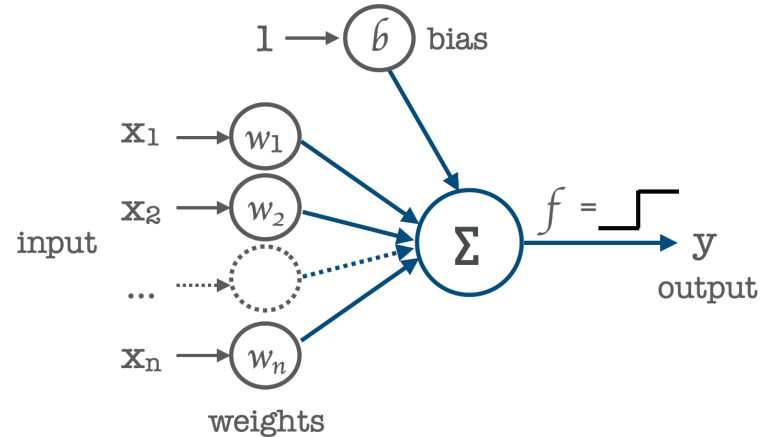
$$\hat{y}^{(i)} = \begin{cases} 1, & \text{if } w \cdot x^{(i)} + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

where:

- $w \cdot x^{(i)}$  is the dot product of the weight vector  $w$  and the feature vector  $x^{(i)}$  for instance  $i$ , i.e.

$$\sum_{j=1}^n w_j x_j^{(i)}$$

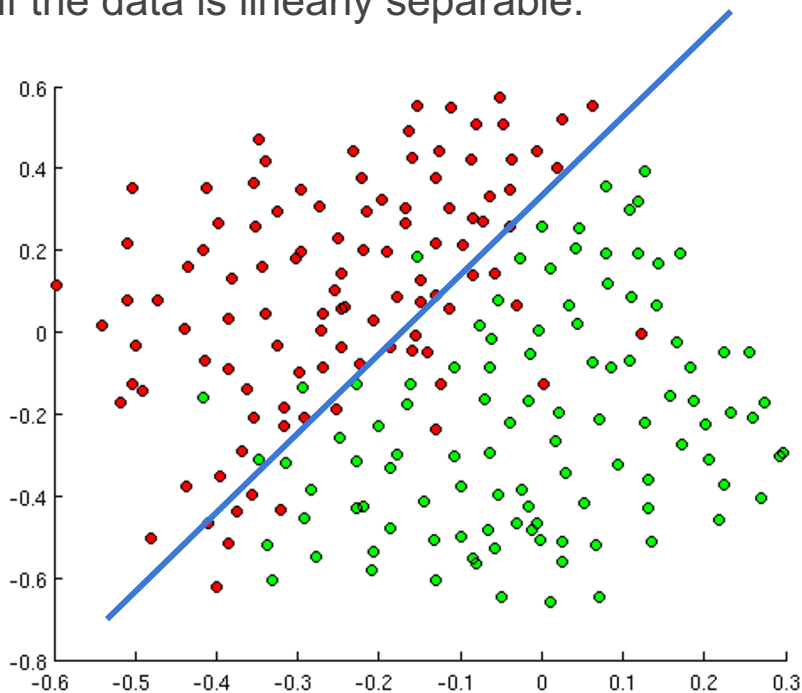
- $b$  is the bias term
- $f$  is a *Heaviside step function*



This is a **Linear Threshold Unit**

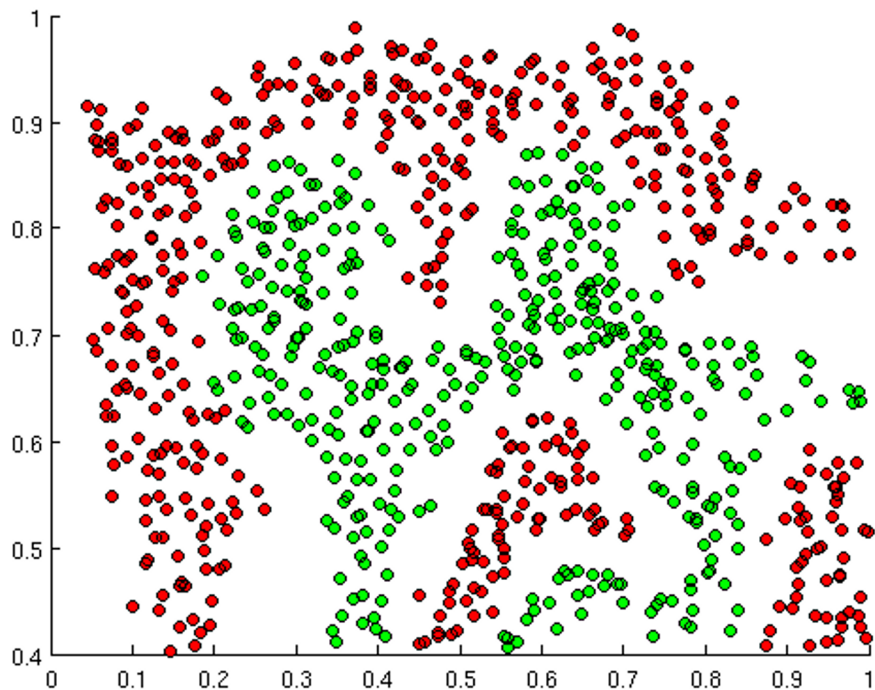
# Linear separability of classes

Linear models are great if the data is linearly separable.



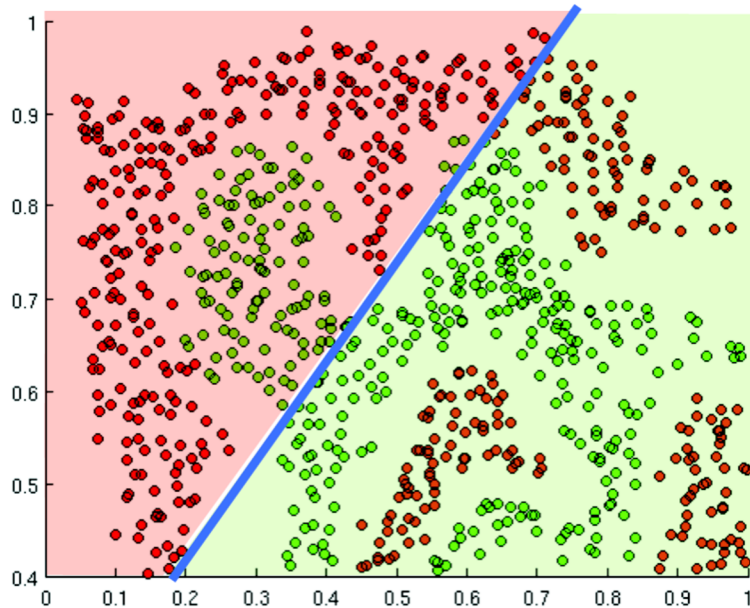
# Linear separability of classes

... but often that is not the case.



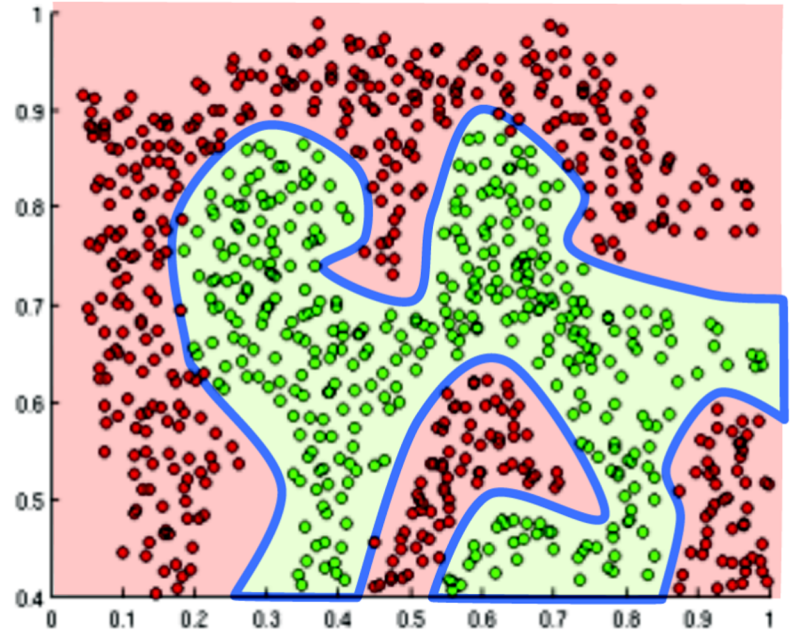
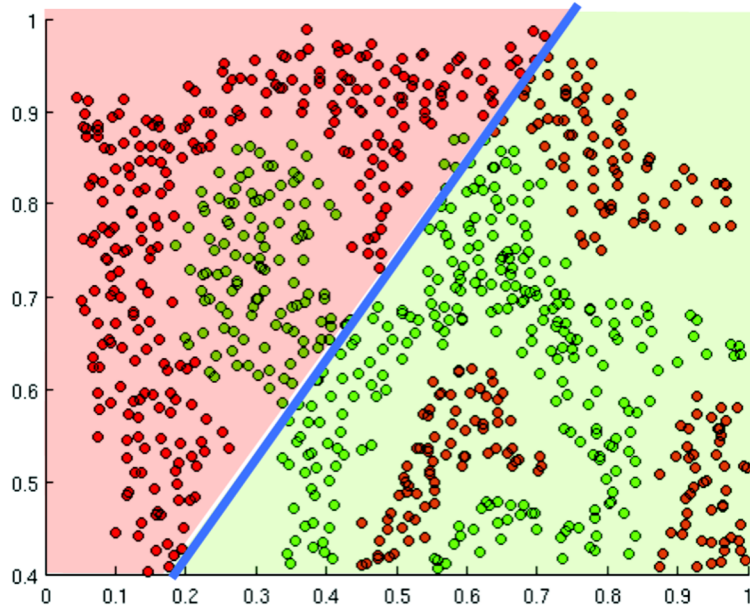
# Linear separability of classes

Linear models are not able to capture complex patterns in the data.



# Linear separability of classes

Linear models are not able to capture complex patterns in the data.

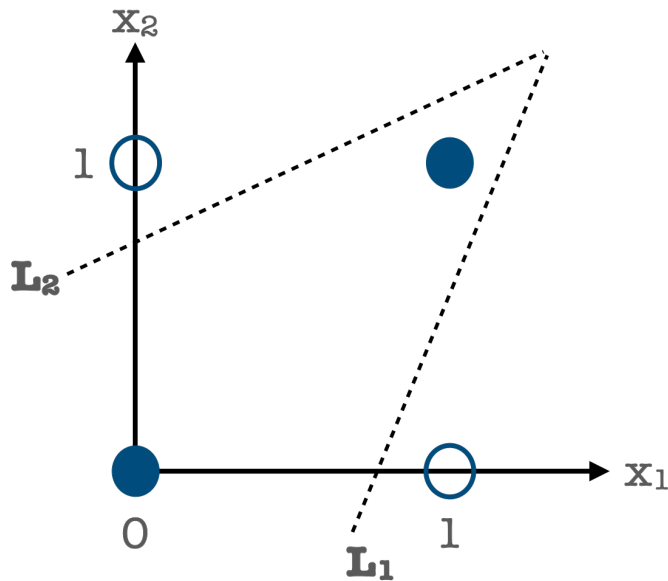


# Recap: Non-linearly separable data

Consider the following classic example of the XOR problem  $y = x_1 \oplus x_2$

$x_1$	$x_2$	$y$
0	0	0
0	1	<b>1</b>
1	0	<b>1</b>
1	1	0

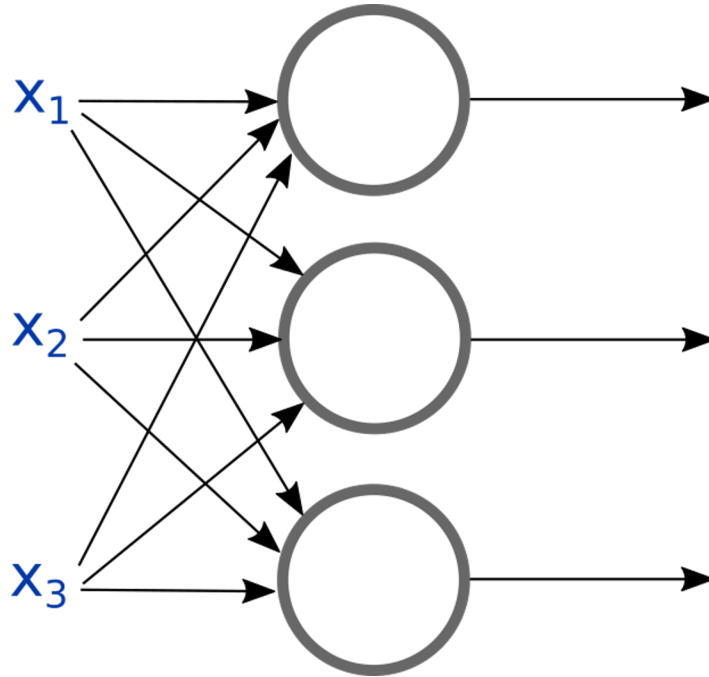
$$y = x_1 \oplus x_2$$





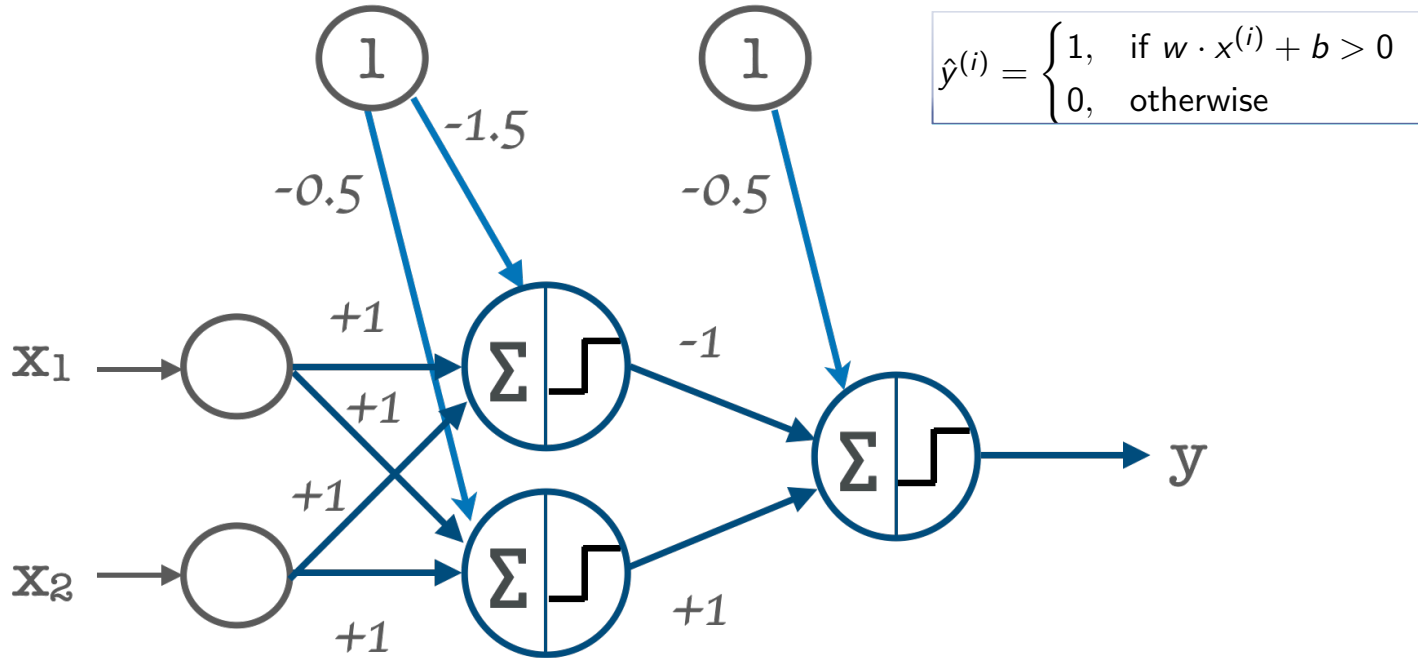
# Connecting the neurons

We can connect multiple neurons in parallel – each one will learn to detect something different.

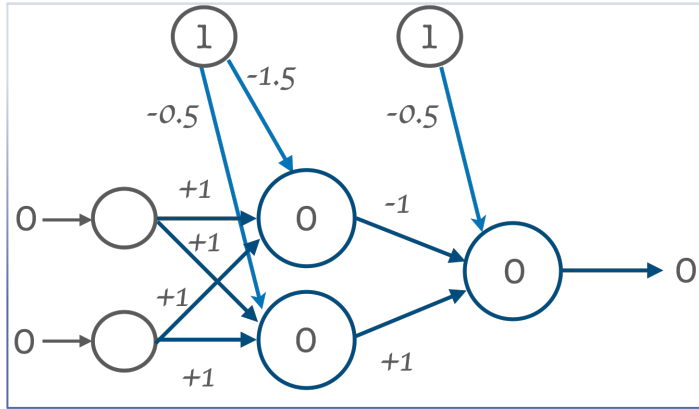


# Connecting the neurons

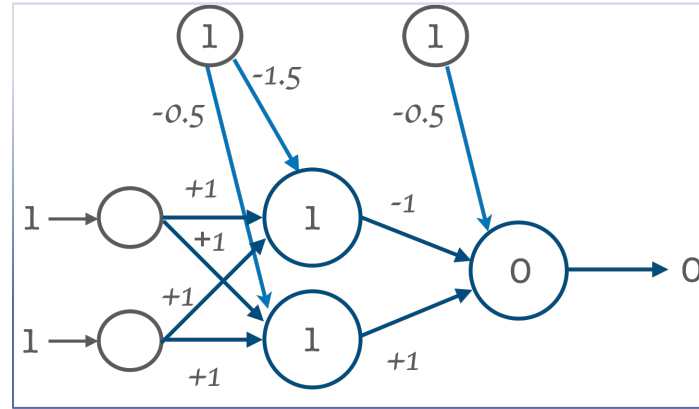
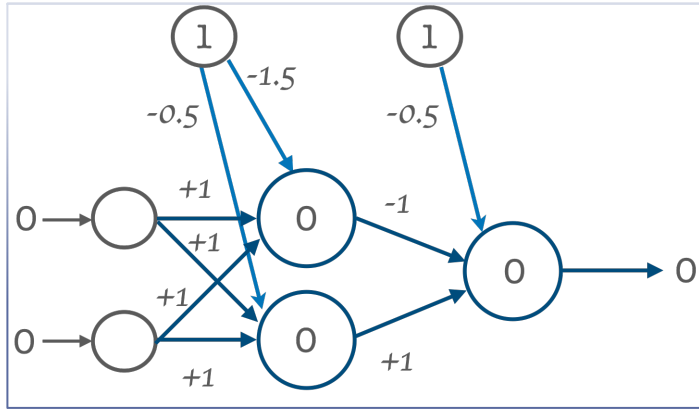
Each node will learn to detect something different: e.g., one hidden node – whether at least one input is 1, and another – whether both are 1



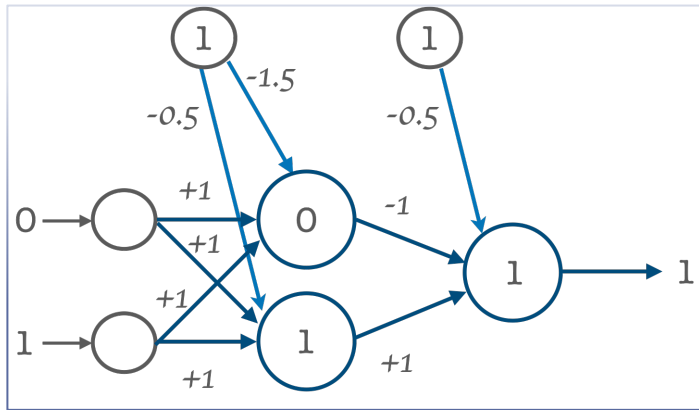
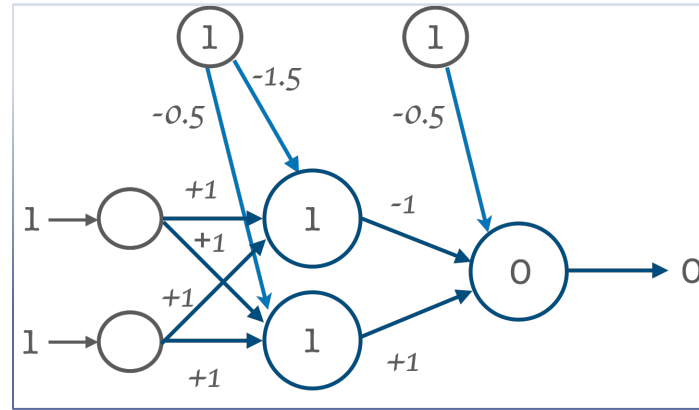
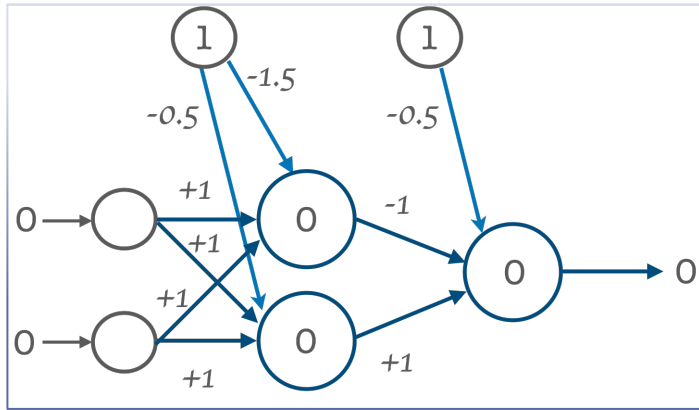
# Multilayer Perceptron (MLP) on XOR problem



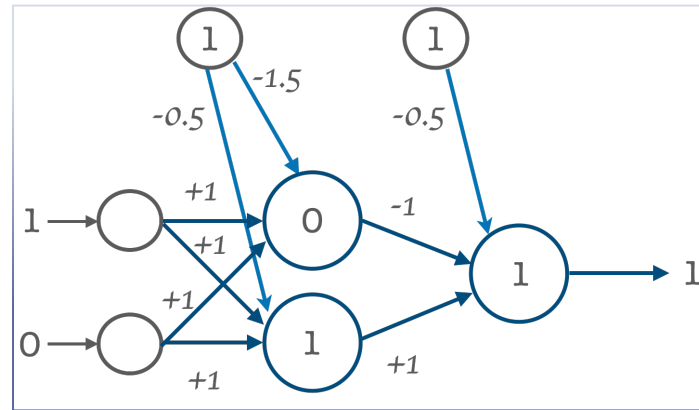
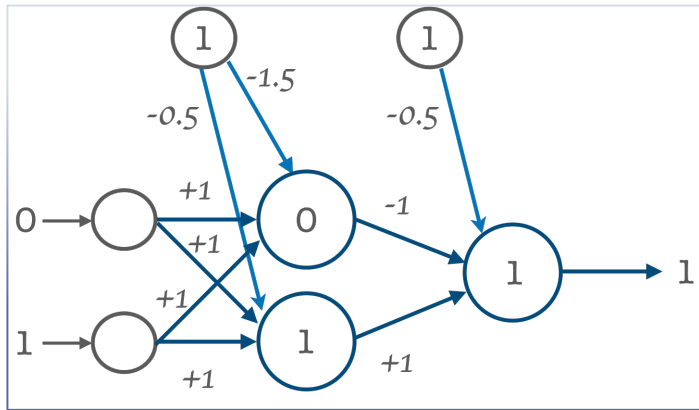
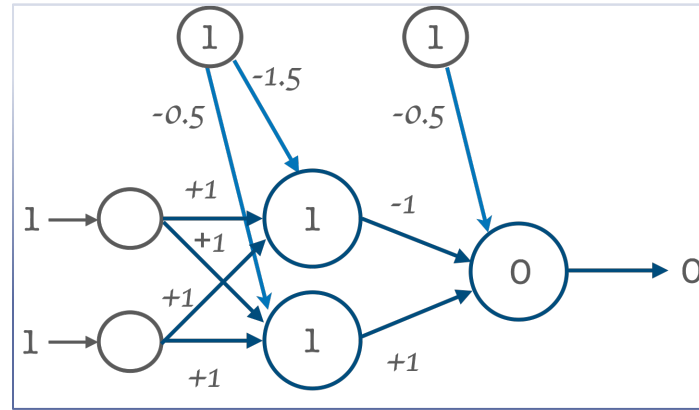
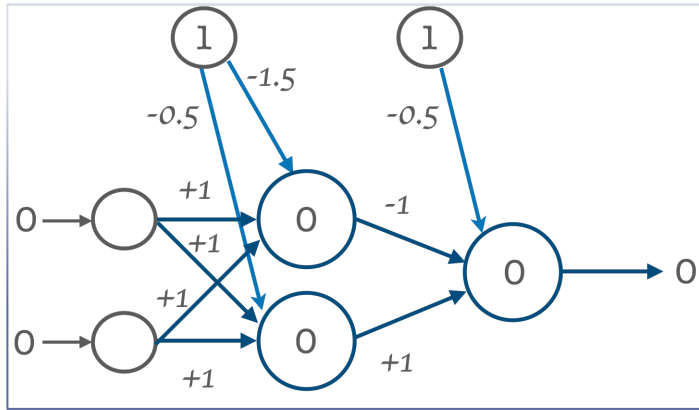
# Multilayer Perceptron (MLP) on XOR problem



# Multilayer Perceptron (MLP) on XOR problem



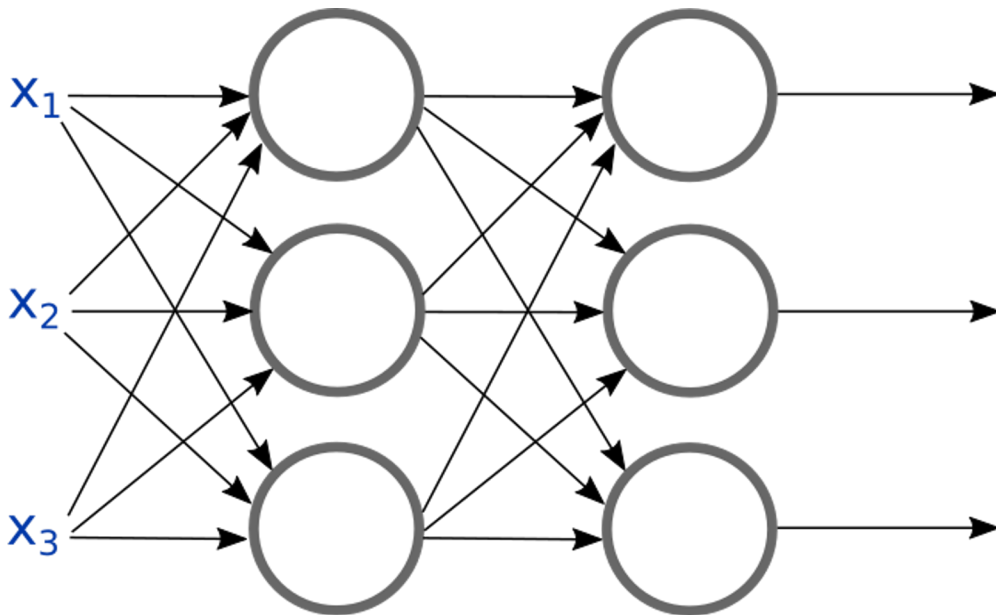
# Multilayer Perceptron (MLP) on XOR problem



# Multilayer Perceptron

← Not actually a  
perceptron

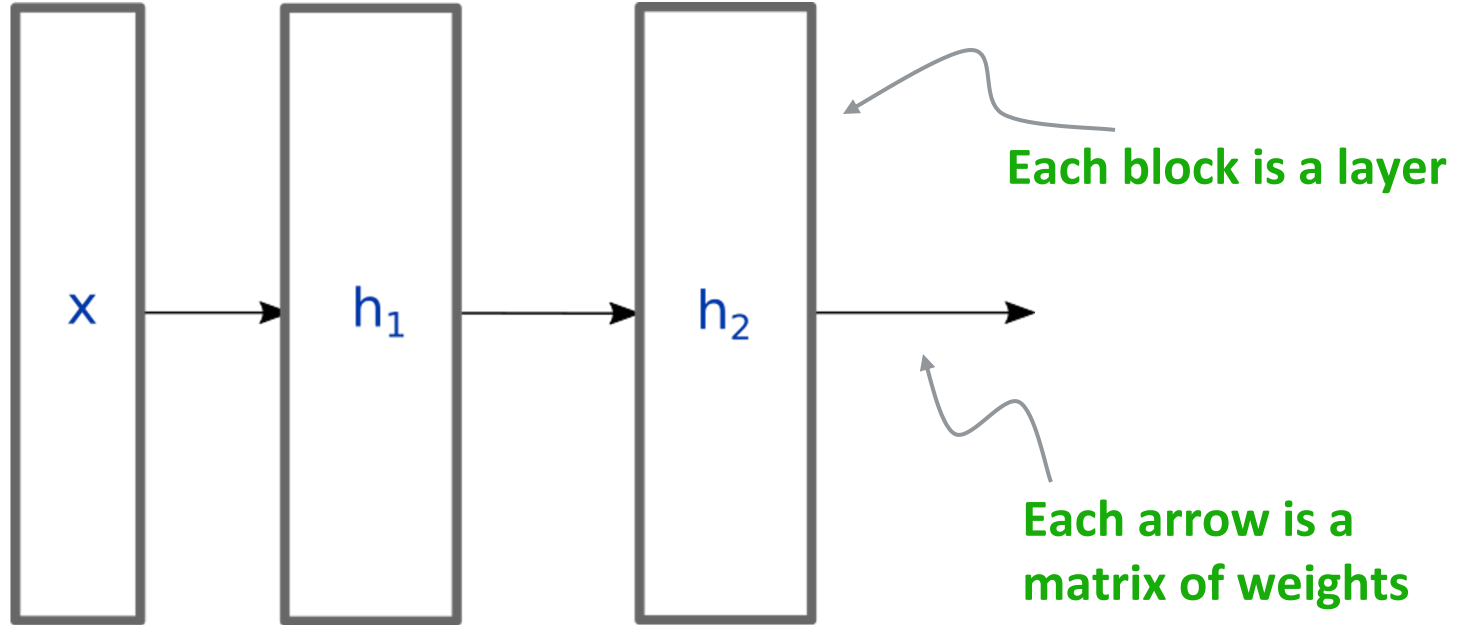
We can connect neurons in sequence in order to learn from higher-order features.



An MLP with sufficient number of neurons can theoretically model an arbitrary function over an input.

# Multilayer Perceptron

We can connect neurons in sequence in order to learn from higher-order features.



An MLP with sufficient number of neurons can theoretically model an arbitrary function over an input.



# Mathematical Definition

$$h_i^{(1)} = \phi^{(1)}(\sum_j w_{i,j}^{(1)} x_j + b_i^{(1)})$$

$$h_i^{(2)} = \phi^{(2)}(\sum_j w_{i,j}^{(2)} h_j^{(1)} + b_i^{(2)})$$

$$y_i = \phi^{(3)}(\sum_j w_{i,j}^{(3)} h_j^{(2)} + b_i^{(3)})$$

- Each unit in the first hidden layer  $h_i^{(1)}$  receives activations from each input unit  $x_j$  multiplied with the weight relevant for this pair of units  $w_{i,j}^{(1)}$  plus unit's own bias  $b_i^{(1)}$

Fully connected network

# Mathematical Definition

$$h_i^{(1)} = \phi^{(1)}(\sum_j w_{i,j}^{(1)} x_j + b_i^{(1)})$$

$$h_i^{(2)} = \phi^{(2)}(\sum_j w_{i,j}^{(2)} h_j^{(1)} + b_i^{(2)})$$

$$y_i = \phi^{(3)}(\sum_j w_{i,j}^{(3)} h_j^{(2)} + b_i^{(3)})$$

- Each unit in the first hidden layer  $h_i^{(1)}$  receives activations from each input unit  $x_j$  multiplied with the weight relevant for this pair of units  $w_{i,j}^{(1)}$  plus unit's own bias  $b_i^{(1)}$
- Second layer  $h_i^{(2)}$  units receive activations from the first layer units  $h_j^{(1)}$

Fully connected network

# Mathematical Definition

$$h_i^{(1)} = \phi^{(1)}(\sum_j w_{i,j}^{(1)} x_j + b_i^{(1)})$$

$$h_i^{(2)} = \phi^{(2)}(\sum_j w_{i,j}^{(2)} h_j^{(1)} + b_i^{(2)})$$

$$y_i = \phi^{(3)}(\sum_j w_{i,j}^{(3)} h_j^{(2)} + b_i^{(3)})$$

- Each unit in the first hidden layer  $h_i^{(1)}$  receives activations from each input unit  $x_j$  multiplied with the weight relevant for this pair of units  $w_{i,j}^{(1)}$  plus unit's own bias  $b_i^{(1)}$
- Second layer  $h_i^{(2)}$  units receive activations from the first layer units  $h_j^{(1)}$
- Different activation functions  $\phi^{(1)} \dots \phi^{(3)}$  may be used at each level

Fully connected network

# Non-linear Activation Functions

- **Logistic function:**

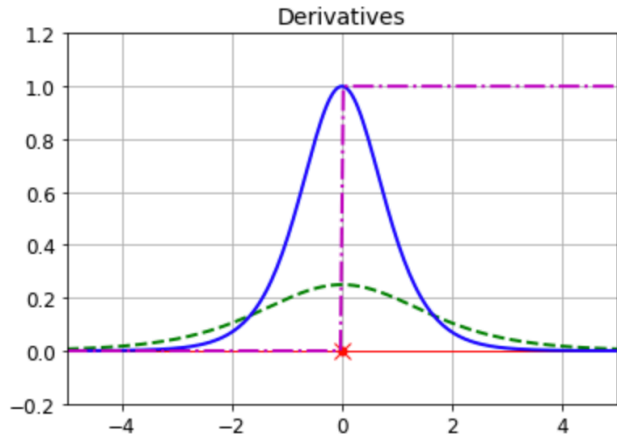
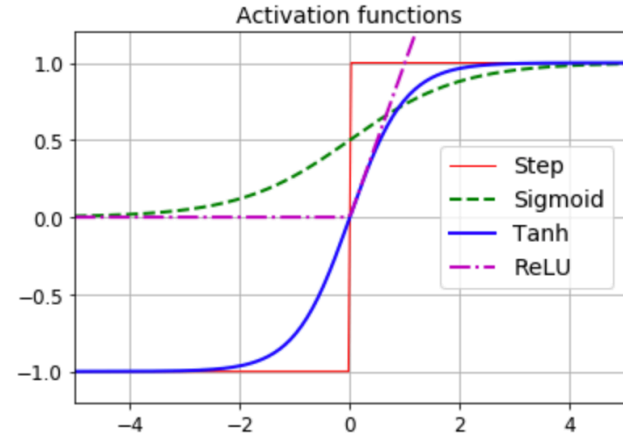
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

- **Hyperbolic tangent function:**

$$\tanh(z) = 2\sigma(2z) - 1$$

- **Rectified linear unit (ReLU) function:**

$$\text{ReLU}(z) = \max(0, z)$$

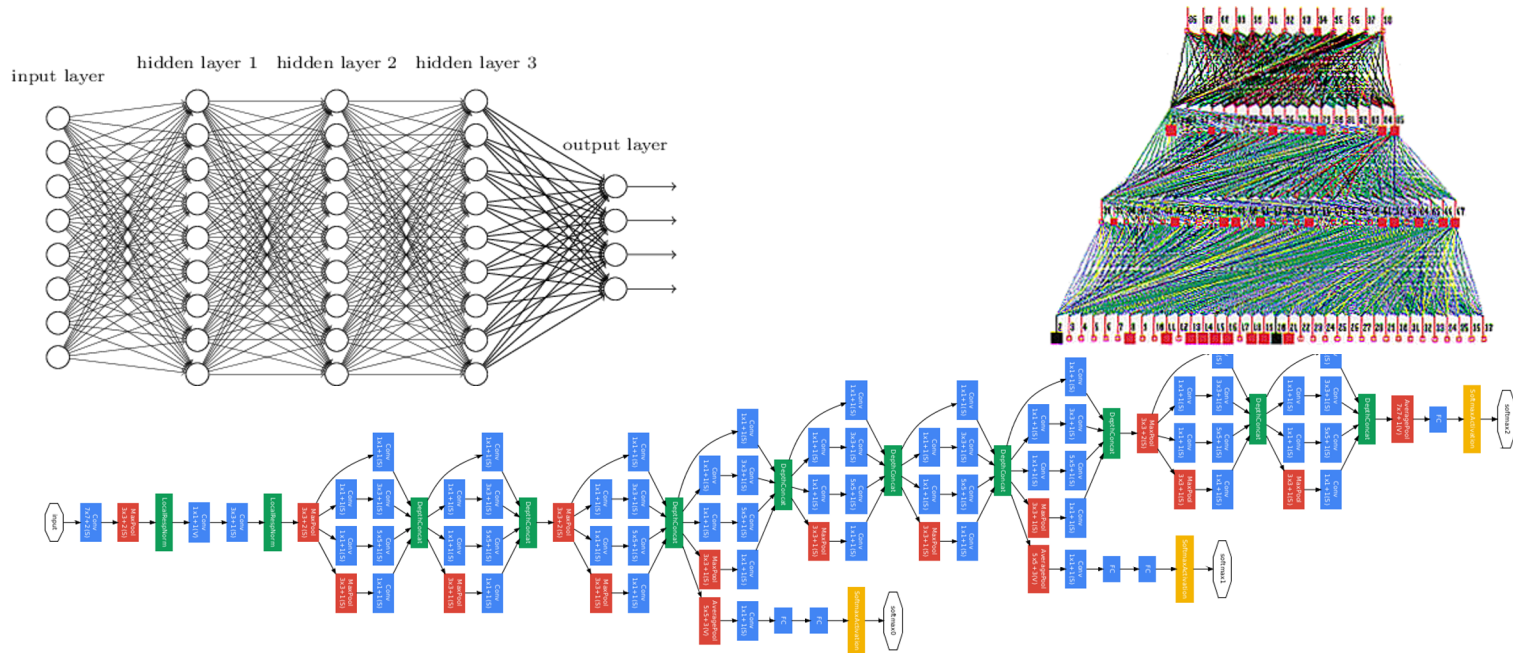


# Neural Network Hyperparameters

- **Depth** – number of hidden layers
- **Width** – number of units per hidden layer
- **Activation functions**

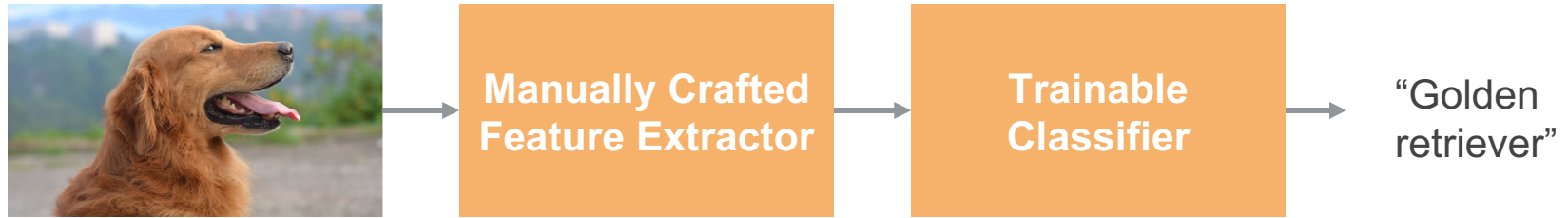
# Deep Neural Networks

In practice we train neural networks with thousands of neurons and millions (or billions) of trainable weights.

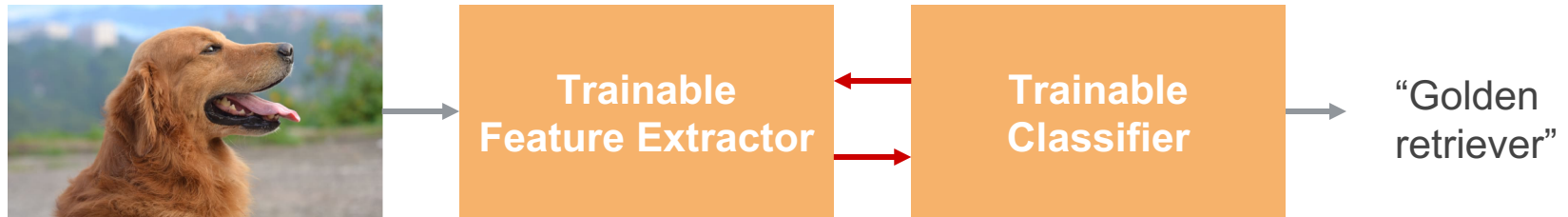


# Learning Representations & Features

Traditional pattern recognition

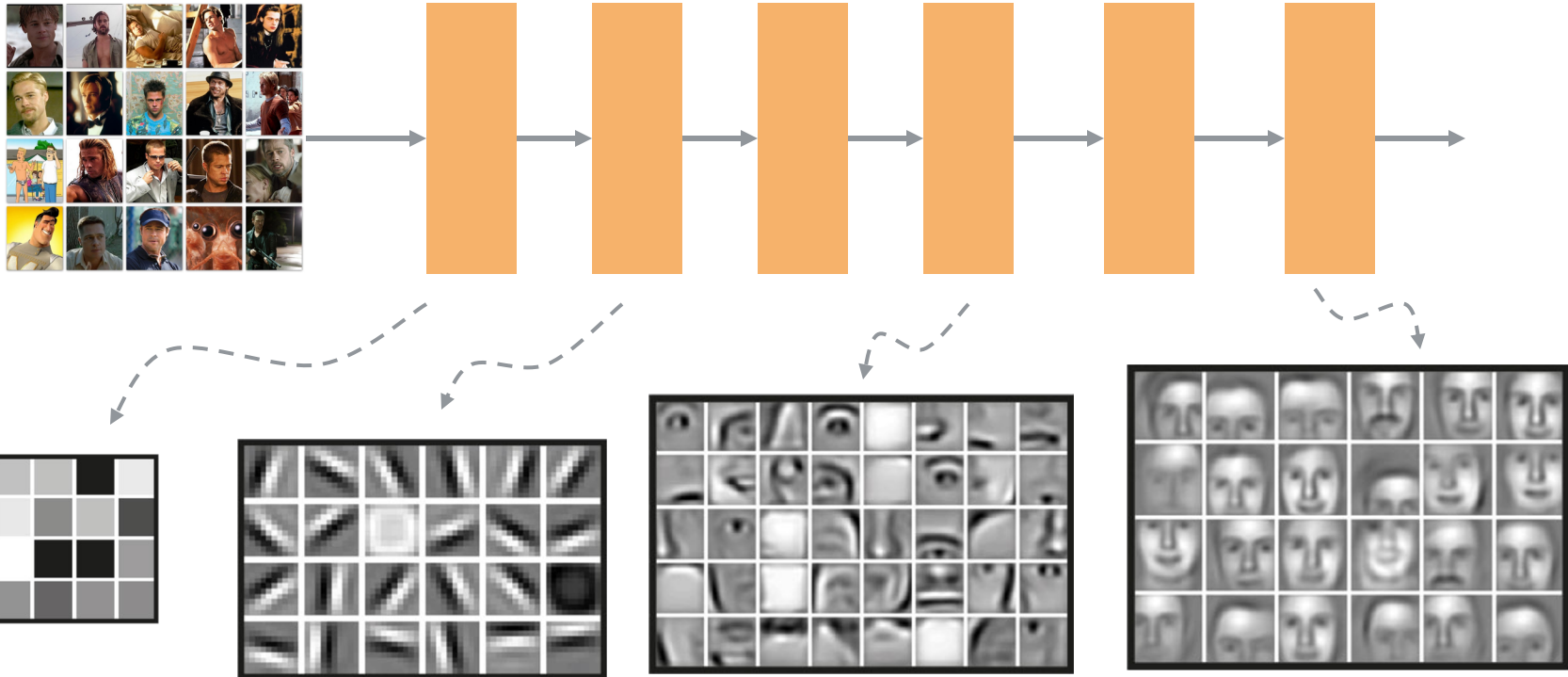


End-to-end training: Learn useful features also from the data



# Learning Representations & Features

Automatically learning increasingly more complex feature detectors from the data.





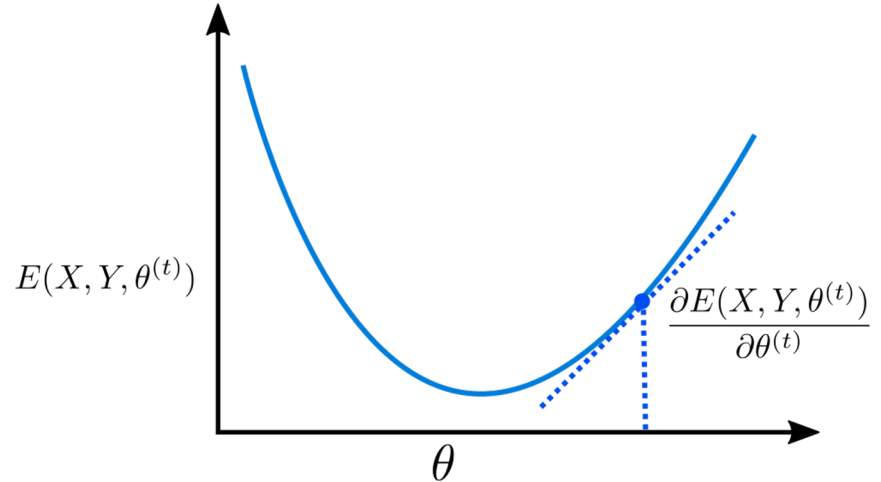
# Neural Network Optimization

# Optimizing Neural Networks

Define a **loss function** that we want to minimize

Update the parameters using **gradient descent**, taking small steps in the direction of the gradient (going downhill on the slope).

All the operations in the network need to be **differentiable**.

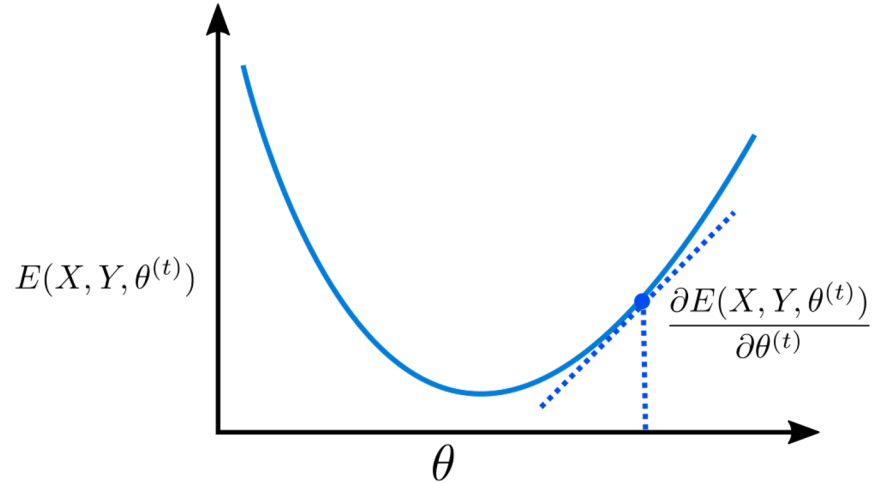


$$\theta_i^{(t+1)} = \theta_i^{(t)} - \alpha \frac{\partial E}{\partial \theta_i^{(t)}}$$

# Gradient Descent

## Algorithm

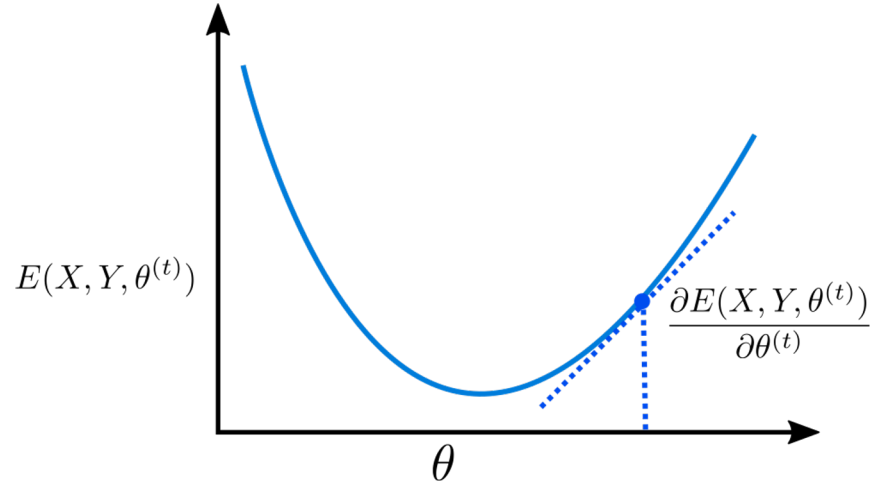
1. Initialize weights randomly
2. Loop until convergence:
3.     Compute gradient based on the whole dataset
4.     Update weights
5. Return weights



# Gradient Descent

## Algorithm

1. Initialize weights randomly
2. Loop until convergence:
3.     Compute gradient based on the whole dataset
4.     Update weights
5. Return weights

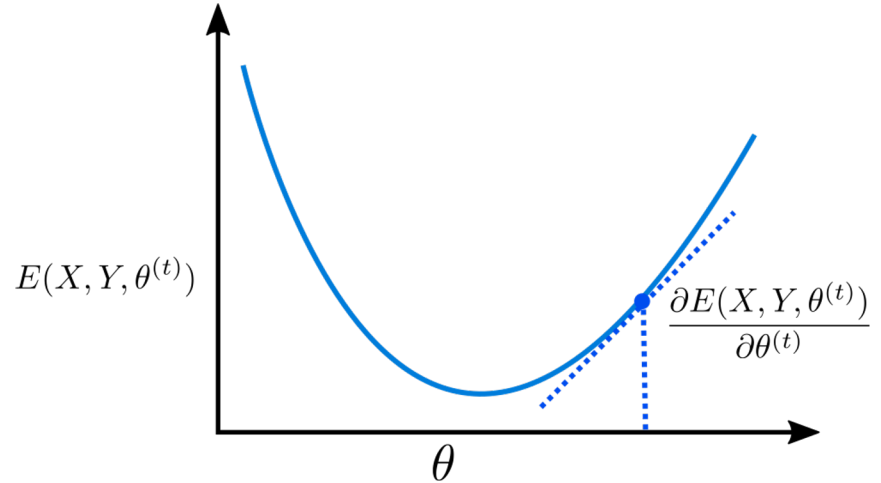


**In practice, datasets are often too big for this**

# Stochastic Gradient Descent

## Algorithm

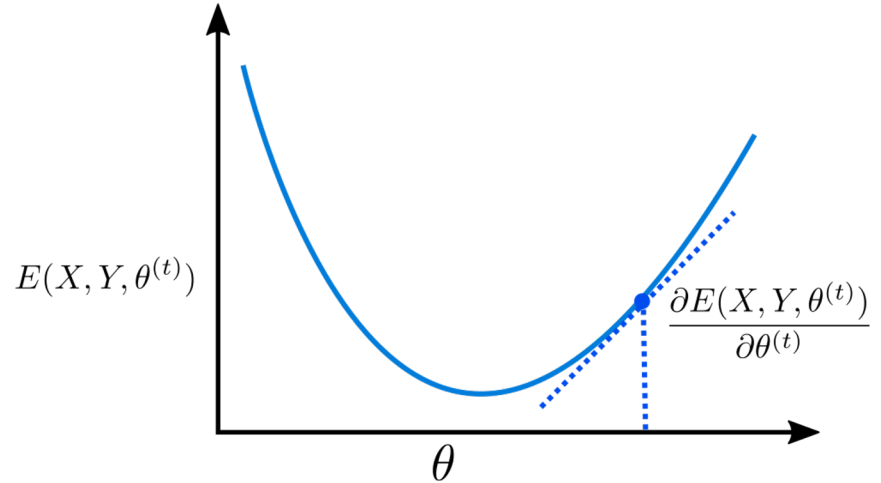
1. Initialize weights randomly
2. Loop until convergence:
  3. Loop over **each datapoint**:
    4. Compute gradient based on the datapoint
    5. Update weights
6. Return weights



# Stochastic Gradient Descent

## Algorithm

1. Initialize weights randomly
2. Loop until convergence:
  3. Loop over **each datapoint**:
    4. Compute gradient based on the datapoint
    5. Update weights
6. Return weights

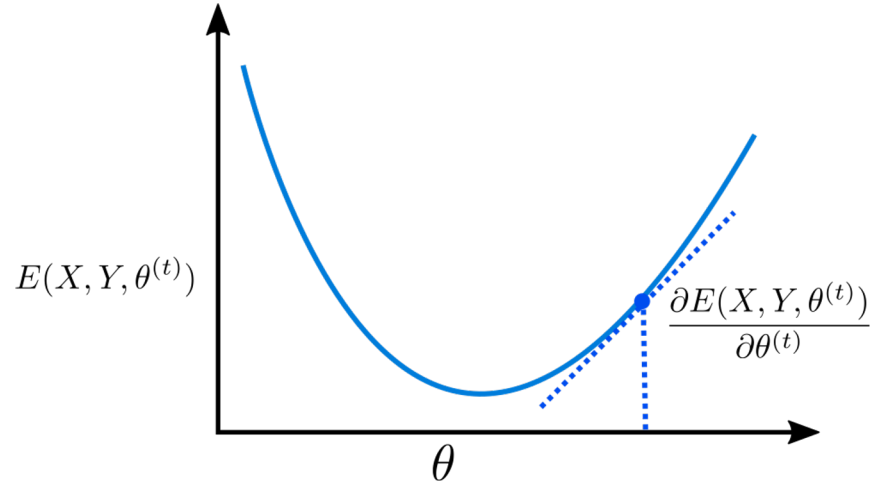


**Very noisy to take steps  
based only on a single  
datapoint**

# Mini-batch Gradient Descent

## Algorithm

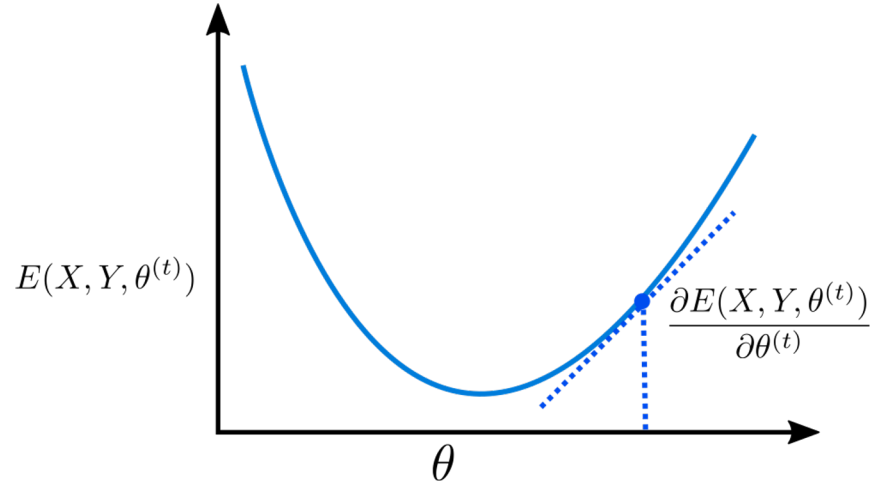
1. Initialize weights randomly
2. Loop until convergence:
  3. Loop over **batches of datapoints**:
    4. Compute gradient based on the batch
    5. Update weights
6. Return weights



# Mini-batch Gradient Descent

## Algorithm

1. Initialize weights randomly
2. Loop until convergence:
  3. Loop over **batches of datapoints**:
    4. Compute gradient based on the batch
    5. Update weights
6. Return weights

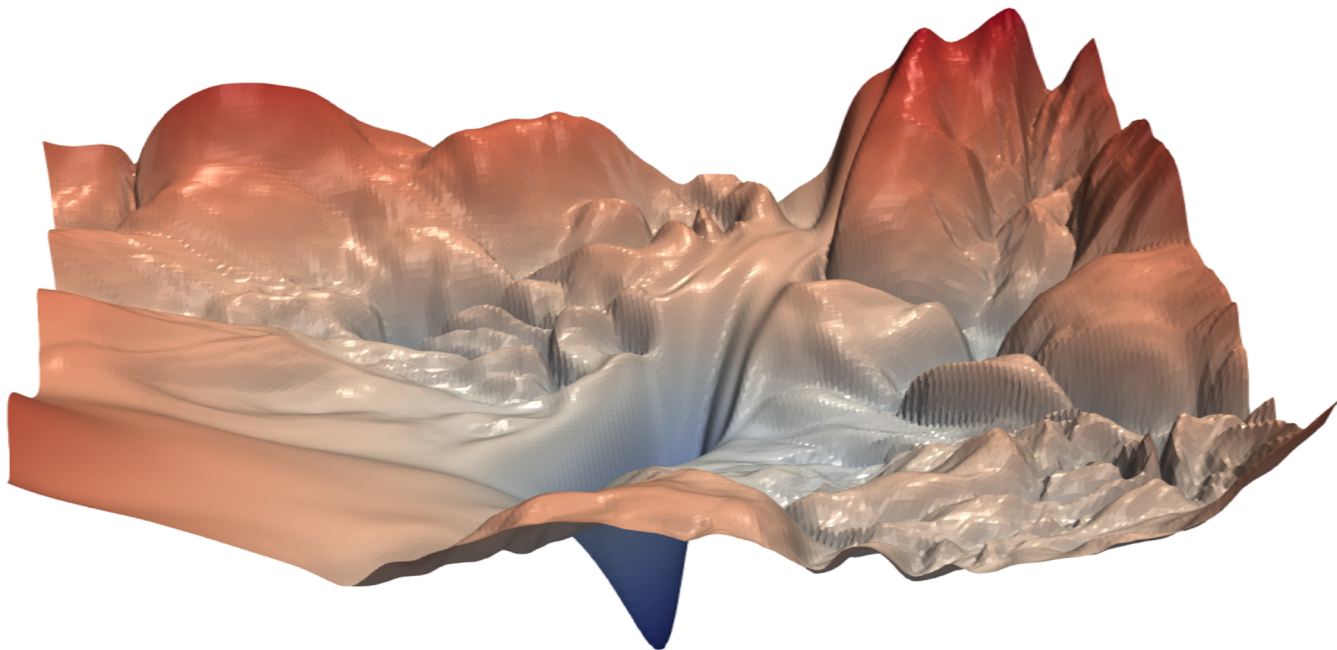


**This is what we  
mostly use in  
practice**



# Optimizing Neural Networks

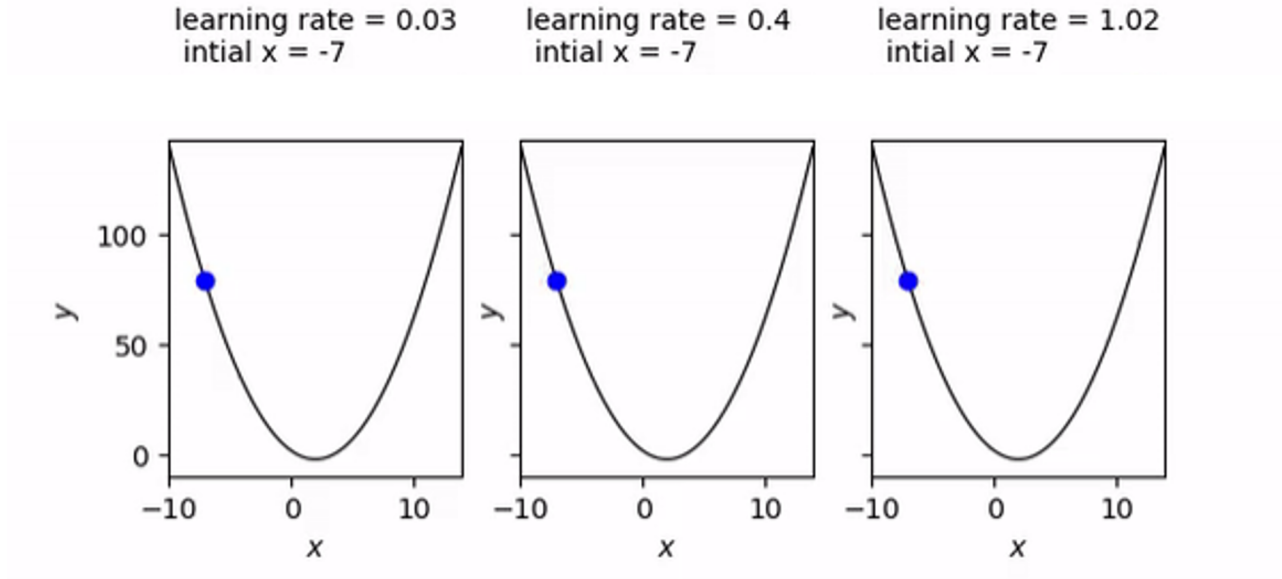
Neural networks have very complex loss surfaces and finding the optimum is difficult.



# The Importance of the Learning Rate

If the learning rate is too low, the model will take forever to converge.

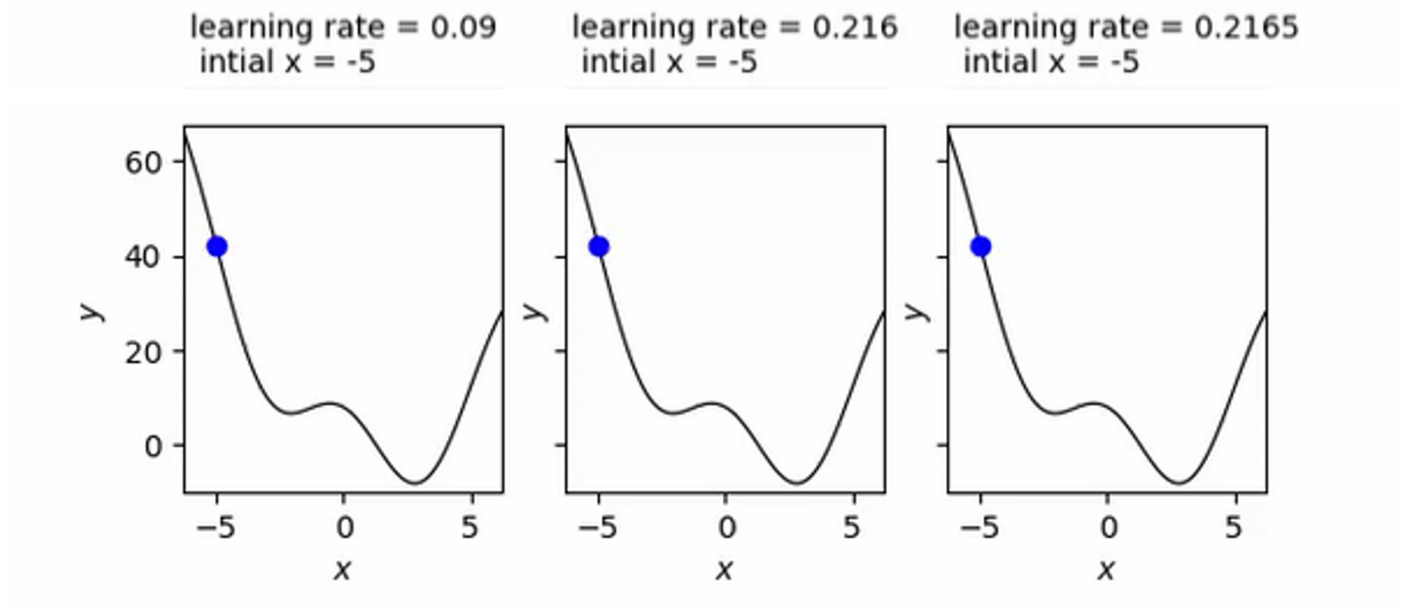
If the learning rate is too high, we will just keep stepping over the optimum values.



# The Importance of the Learning Rate

A small learning rate can get the model stuck in local minima.

A bigger learning rate can help the model converge better (if it doesn't overshoot).



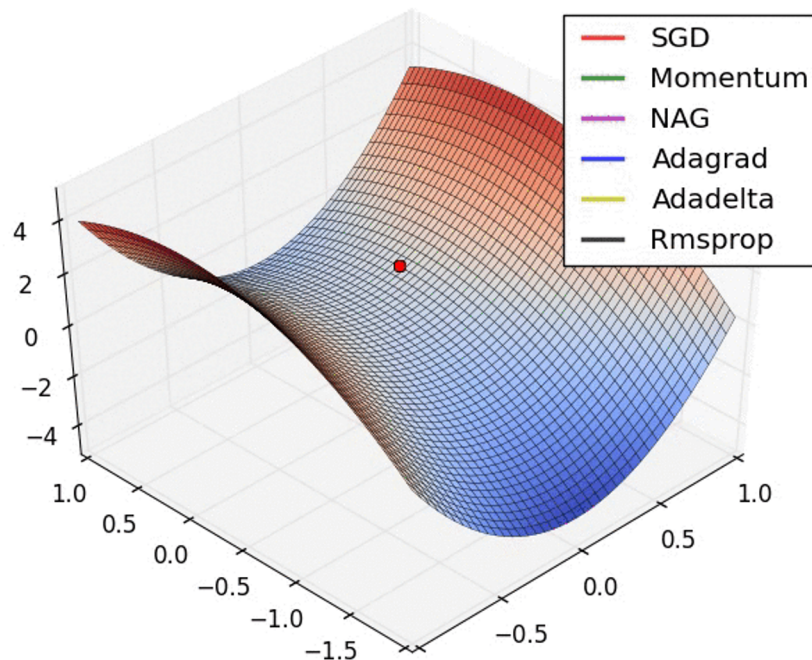
# Adaptive Learning Rates

## Intuition:

Have a different learning rate for each parameter.

Take bigger steps if a parameter has not been updated much recently.

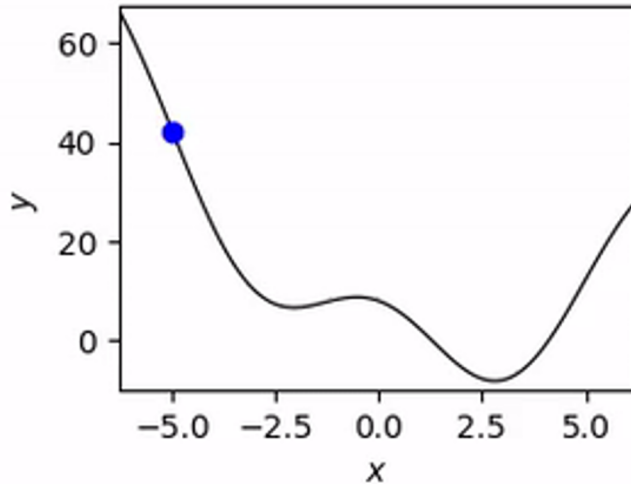
Take smaller steps if a parameter has been getting many big updates.



# Random initialization Matters

All other things being equal, just starting from a different location can lead to a different result.

learning rate = 0.07  
initial x = -5



learning rate = 0.07  
initial x = 5

