

# Neural Networks

Bùi Tiến Lên

2021



KHOA CÔNG NGHỆ THÔNG TIN  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

# Contents

---



1. Artificial Neural Network
2. Perceptron: The simplest ANN (Revisited)
3. Multilayer Perceptrons (MLP)
4. Gradient-Based Learning
5. Advanced Gradient-Based Learning
6. Back-propagation Learning



# Notation

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning  
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

symbol	meaning	operator	meaning
$a, b, c, N \dots$	scalar number	$w^T$	transpose
$w, v, x, y \dots$	column vector	$X Y$	matrix
$X, Y \dots$	matrix	$X^{-1}$	multiplication
$\mathbb{R}$	set of real numbers	$X \odot Y$	inverse
$\mathbb{Z}$	set of integer numbers		an element-wise
$\mathbb{N}$	set of natural numbers		matrix-vector
$\mathbb{R}^D$	set of vectors		multiplication
$\mathcal{X}, \mathcal{Y}, \dots$	set		
$\mathcal{A}$	algorithm		



Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Previous Lessons

---

- Ensemble
- Uncertainty
- Feature representation
  - Feature transformation
  - Hidden variables



# Deep Learning vs. Classical Machine Learning

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

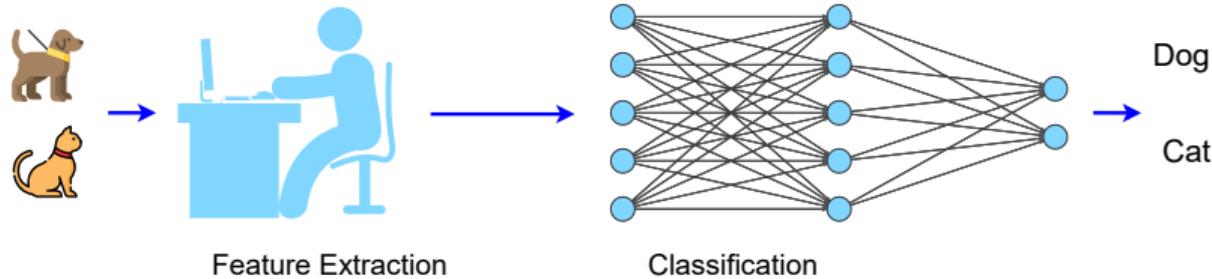
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

## Classical Machine Learning

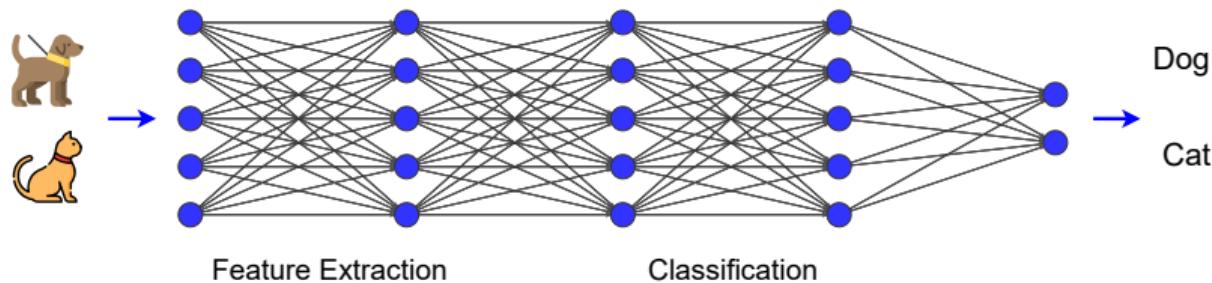


Feature Extraction

Classification

Dog  
Cat

## Deep Learning



Feature Extraction

Classification

Dog  
Cat



# Artificial Neural Network



# What is a neural network?

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

- Gradient Descent
- Gradient Ascent
- Min-max optimization

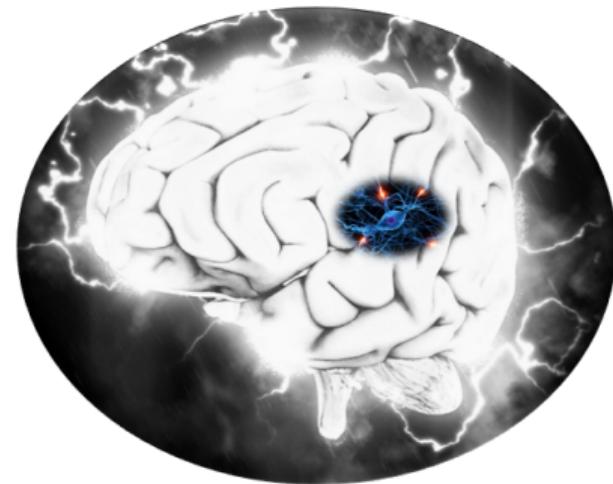
Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

**Neural network** is a reasoning model based on the human brain, including billions of neurons and trillion connections between them.

The biological brain is considered as a highly complex, nonlinear and parallel information-processing system.

- Number of neurons  $10^{10}$
- Connections per neuron  $10^{4-5}$
- Neuron switching time .001 second
- Scene recognition time .1 second





# Artificial neural networks

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

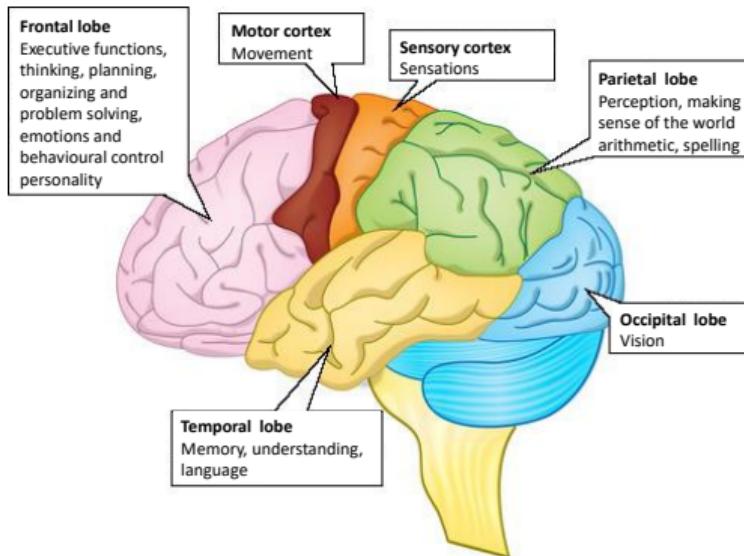
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- **Artificial neural networks** (ANNs) resembles the human brain in **terms of learning mechanisms**.
- An ANN consists of a number of very simple and highly interconnected processors (or **neurons**), arranging in a **hierarchy of layers**.





# Neuron

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

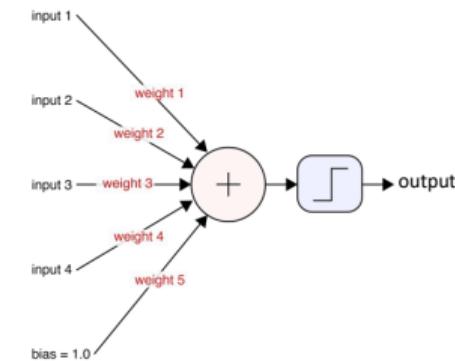
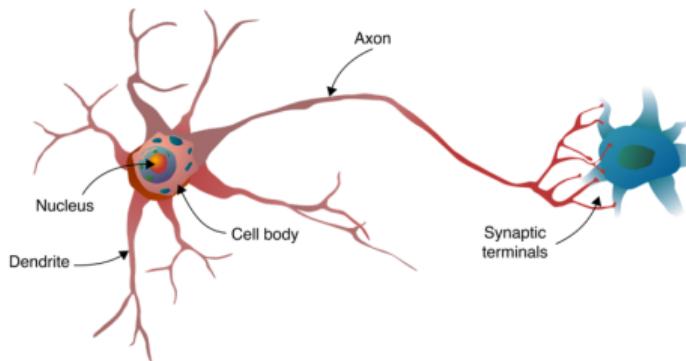
Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning  
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- Each **neuron** is an elementary information-processing unit.
- The neurons are connected by **links** passing signals from one neuron to another.
- Each neuron receives a number of **input signals** through its connections and produces at most a single **output signal**.
- Each link associates with a **numerical weight** expressing the strength of the neuron input → basic means of long-term memory in ANNs
- ANNs “learn” through repeated adjustments of these weights





# The network architecture

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

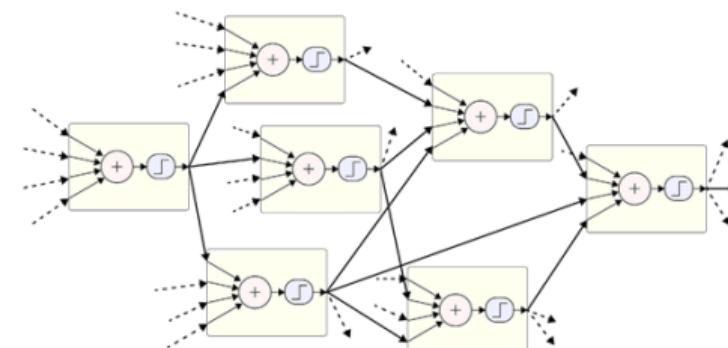
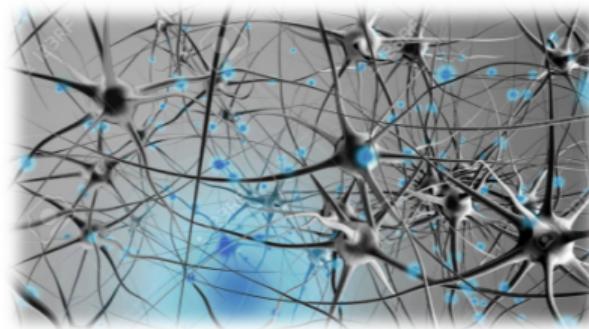
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- The network architecture includes
  - How many neurons are to be used?
  - How the neurons are to be connected to form a network?





# Some architectures

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

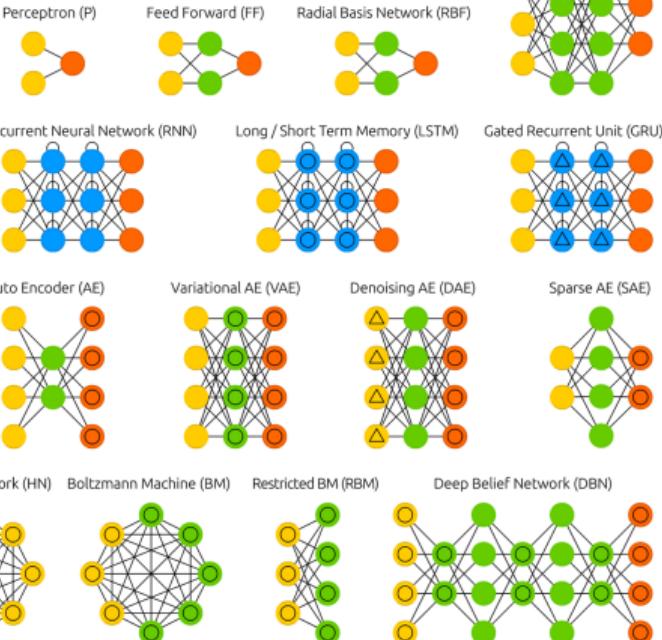
Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- Input Cell
- Backfed Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Gated Memory Cell
- Kernel
- Convolution or Pool

## A mostly complete chart of Neural Networks

©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org





# Some architectures (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

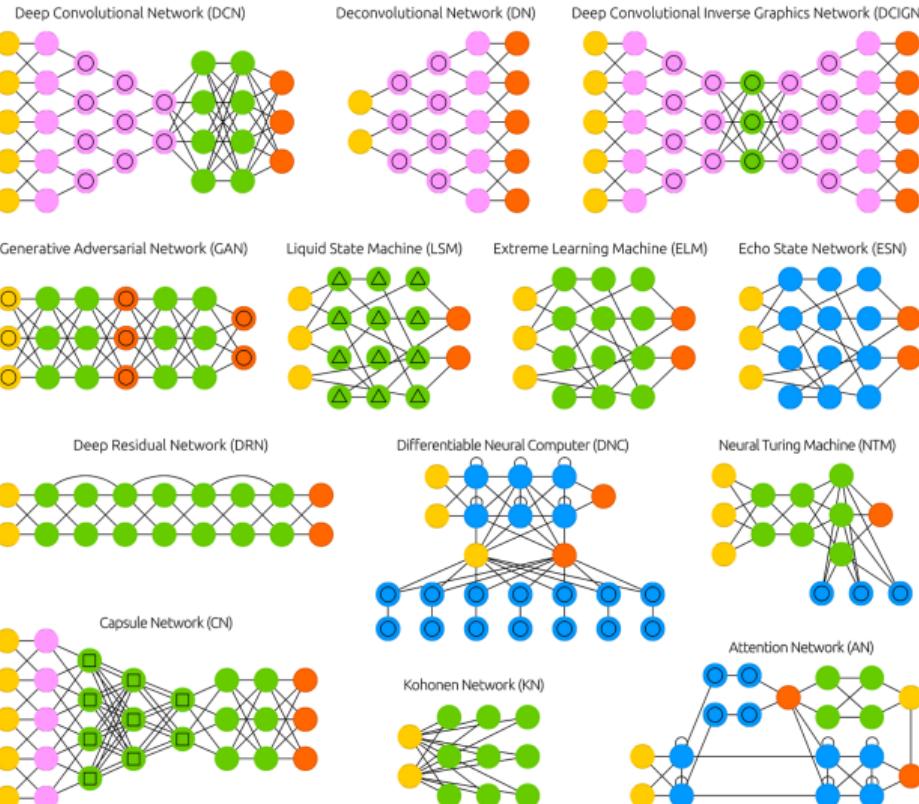
Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning





# Perceptron: The simplest ANN (Revisited)



# Perceptron Model

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

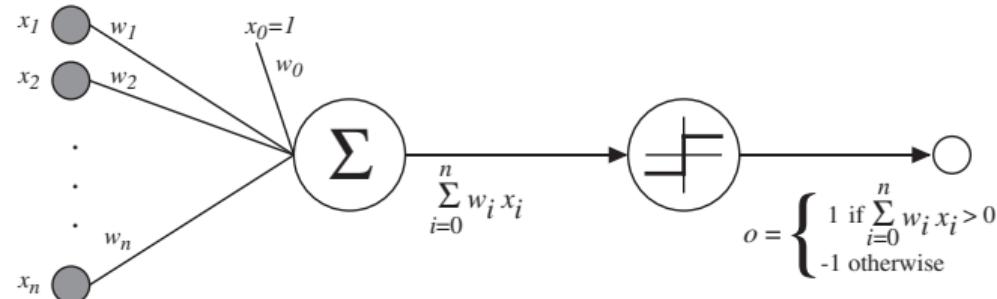
Back-  
propagation  
Learning

- A **perceptron** proposed by Minsky-Papert, more general than McCulloch-Pitts neuron, consists of  $n$  inputs  $x_1, \dots, x_n$  and 1 output  $o$
- First, it computes a **weighted sum of its inputs**

$$s = \sum_{i=0}^n w_i x_i = w_0 + w_1 x_1 + \dots + w_n x_n \quad (1)$$

- Then it applies an **activation function**  $g(s)$  (*sign, sigmoid etc.*)

$$o(x_1, \dots, x_n) = g(s) = \begin{cases} 1 & \text{if } s \geq 0 \\ -1 & \text{otherwise.} \end{cases} \quad (2)$$





# Representation

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

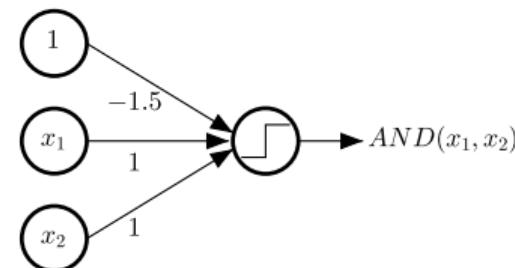
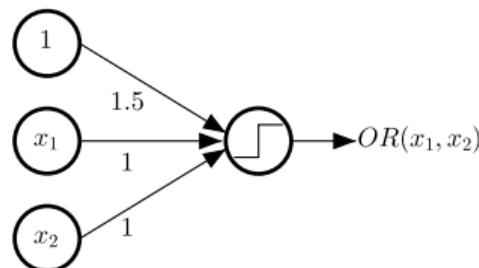
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- Perceptron can represent some useful functions



$x_1$	$x_2$	$OR(x_1, x_2)$
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

$x_1$	$x_2$	$AND(x_1, x_2)$
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

- Perceptron cannot represent  $XOR(x_1, x_2)$ 
  - Solution is to combine perceptrons → multilayer perceptrons



# Multilayer Perceptrons (MLP)

- Learning Model
- Design Neural Network
- Neural Network Learning



# Multilayer Perceptrons

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model

Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

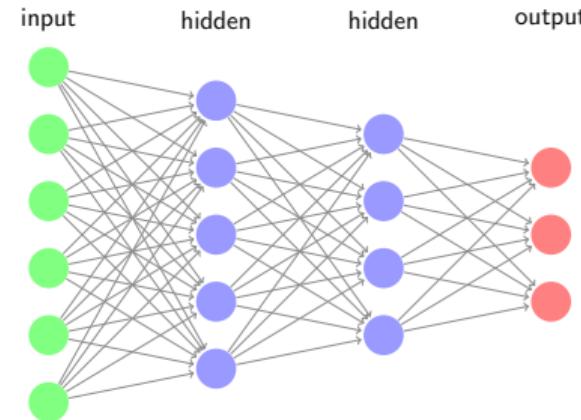
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

Multilayer perceptrons (neural networks) is a DAG graph that includes many layers.

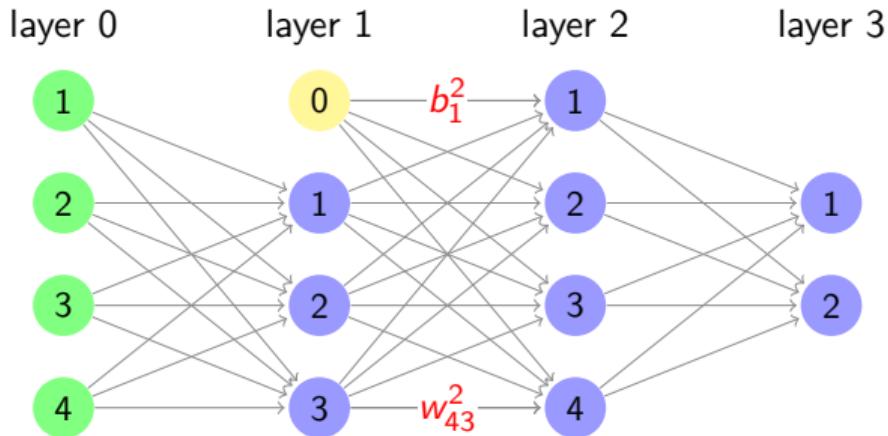
- The first layer is called **input layer**
- The last layer is called **output layer**
- The other are called **hidden layers**





# Multilayer Perceptrons (cont.)

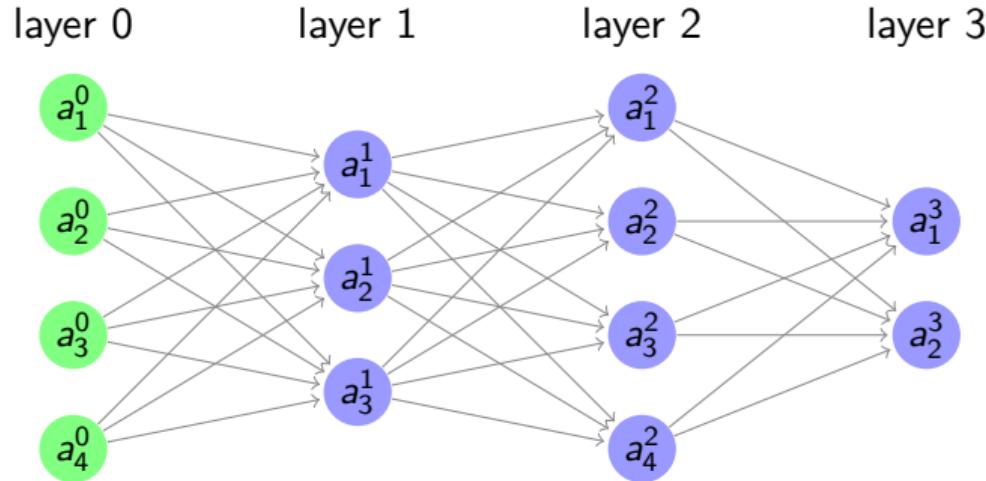
- Each layers composed of **nodes** (neurons)
- Neurons are connected by directed **links** to propagate the **activation**.  $w_{jk}^l$  is the weight of the link from the  $k$ -th neuron in the  $(l - 1)$ -th layer to the  $j$ -th neuron in the  $l$ -th layer
- Each layer has **weight matrix  $w^l$**  and a **bias vector  $b^l$**





# Multilayer Perceptrons (cont.)

- The  $j$ -th neuron in the  $l$ -th layer has the **activation**  $a_j^l$





# Multilayer Perceptrons (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model

Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

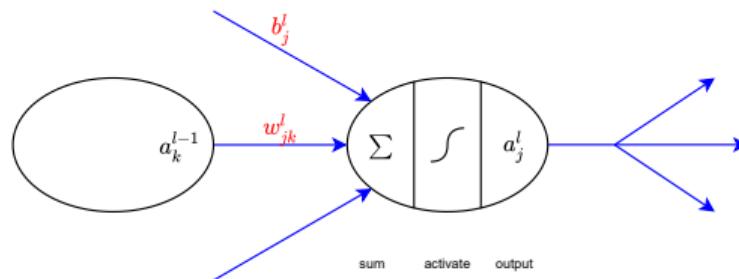
Back-  
propagation  
Learning

- The  $j$ -th neuron in the  $l$ -th layer first computes a weighted sum of its inputs

$$in = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \quad (3)$$

- Then it applies an **activation function**  $g$  to this sum to derive the output:

$$a_j^l = g(in) = g\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right) \quad (4)$$





# Activation functions

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning  
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- Sign function

$$a = \text{sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \quad (5)$$

- Step function

$$a = \text{step}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (8)$$

- Sigmoid function

$$a = \sigma(x) = \frac{1}{1 + \exp(-x)} \quad (6)$$

- Tanh function

$$a = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (9)$$

- Linear function

$$a = x \quad (7)$$

- ReLU (rectified linear unit)

$$a = \text{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (10)$$



# Activation functions (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

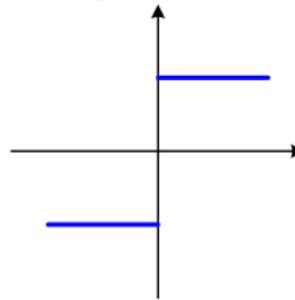
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

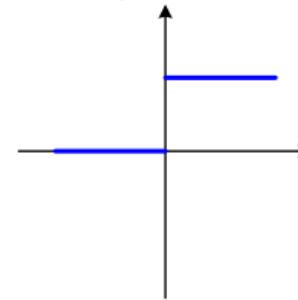
Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

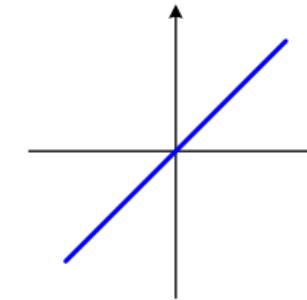
Sign function



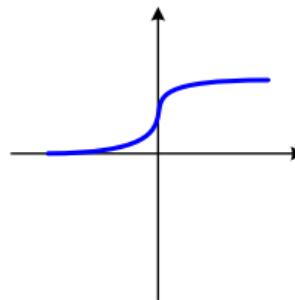
Step function



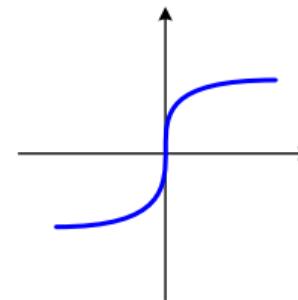
Linear function



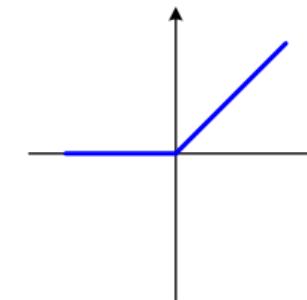
Sigmoid function



Tanh function



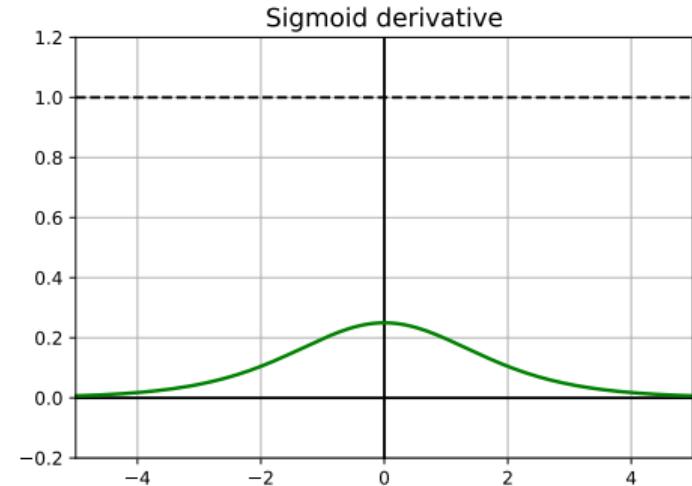
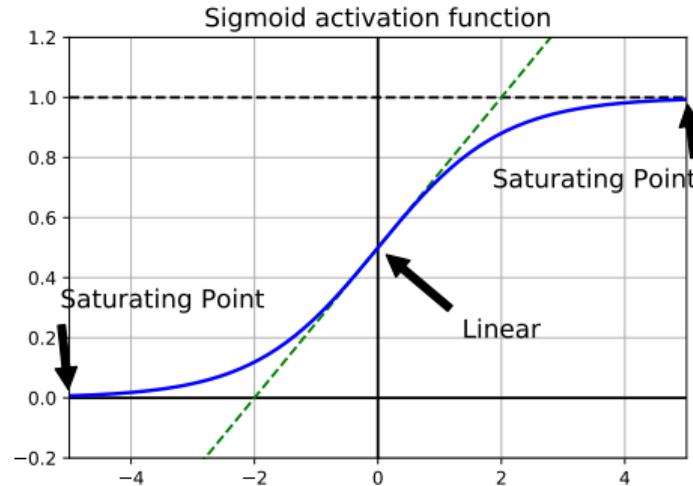
ReLU function





# Activation functions (cont.)

- Sigmoid function





# Activation functions (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

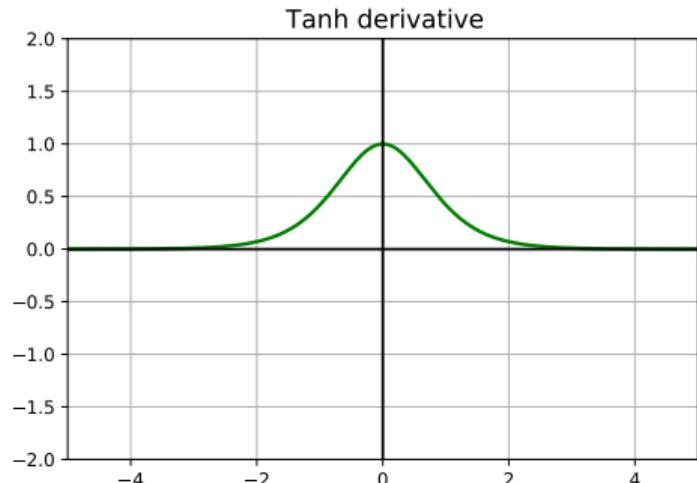
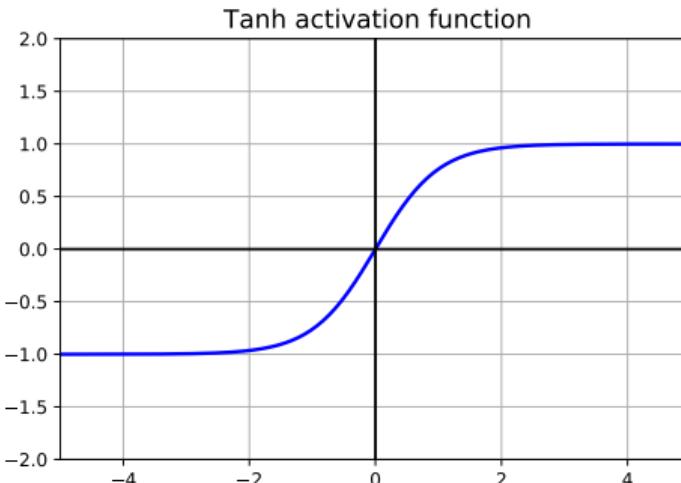
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- Tanh function





# Softmax Gate

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model

Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

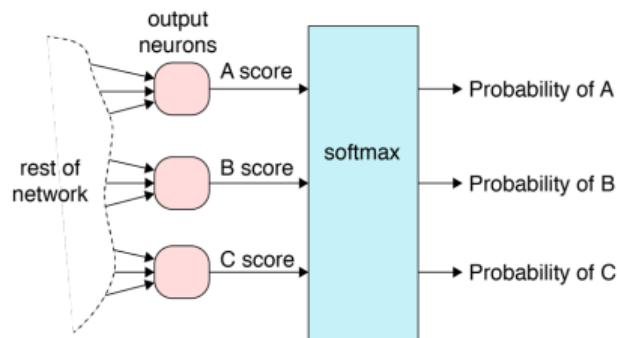
Back-  
propagation  
Learning

- The softmax function takes all the network's outputs and modifies them simultaneously. The result is that the scores are turned into probabilities (this technique is not a normal activation function)

$$f_t(\mathbf{x}) = f_t(x_1, x_2, \dots, x_k) = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}}, \text{ for } t = 1, \dots, k \quad (11)$$

$$f_t(\mathbf{x}) = f_t(x_1, x_2, \dots, x_k) = \frac{e^{x_i - M}}{\sum_{j=1}^k e^{x_j - M}}, \text{ stable version} \quad (12)$$

where  $M = \max(x_1, x_2, \dots, x_k)$





# Universal Approximation Theorem

## Theorem 1

### *Expressive Capabilities of MLP*

- ***Boolean functions***

- *Every boolean function can be represented by network with **single hidden layer** but might require exponential (in number of inputs) hidden units*

- ***Continuous functions***

- *Every bounded continuous function can be approximated with arbitrarily small error, by network with **one hidden layer***
- *Any function can be approximated to arbitrary accuracy by a network with **two hidden layers***



# Learning Model

Perceptron:  
The simplest  
ANN  
(Revisited)

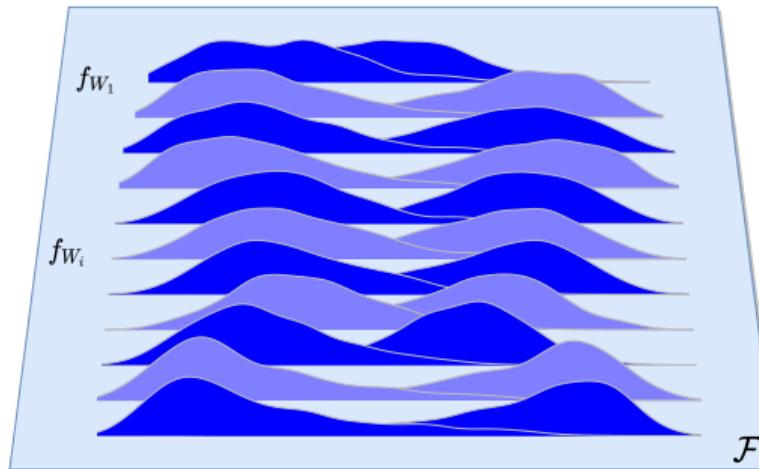
Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning  
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning



## 2. Learning algorithm:

- Gradient-based algorithms
- Delta learning rule
- ...



# Design Neural Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
**Design Neural Network**

Neural Network  
Learning

Gradient-Based  
Learning

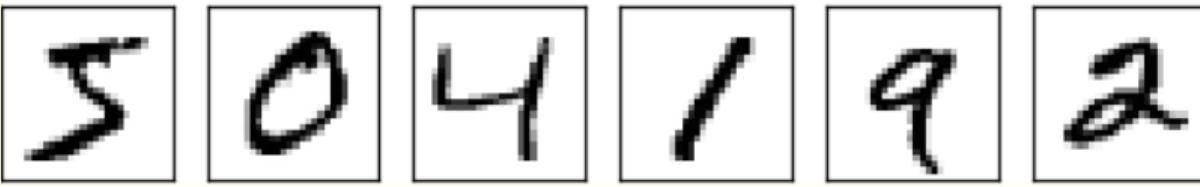
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

## Example 1

A simple network to classify handwritten digits





Artificial Neural Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model

Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Design Neural Network (cont.)

---

- The input layer of the network contains neurons encoding the values of the input pixels. Our training data for the network will consist of many 28 by 28 pixel images of scanned handwritten digits, and so the input layer contains  $784 = 28 \times 28$  neurons
- The second layer of the network is a hidden layer
- The output layer of the network contains 10 neurons. If the first neuron fires, i.e., has an output  $\approx 1$ , then that will indicate that the network thinks the digit is a 0. If the second neuron fires then that will indicate that the network thinks the digit is a 1. And so on



# Design Neural Network (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model

Design Neural Network

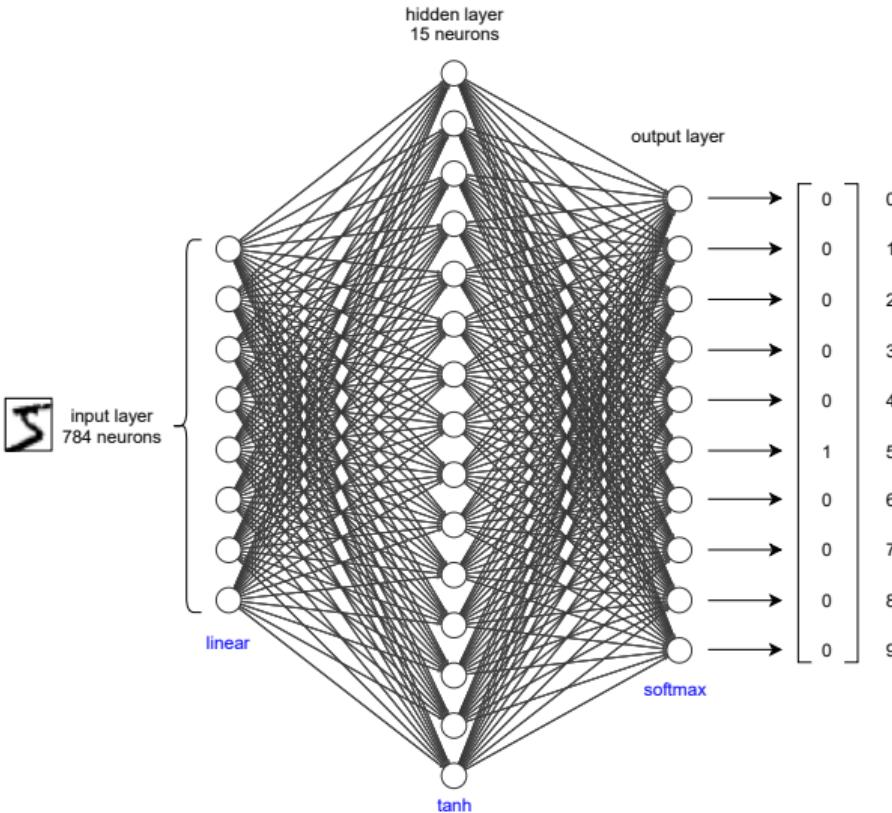
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning





Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
**Design Neural Network**

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Design Neural Network (cont.)

## Example 2

A simple object classification



Pedestrian



Car



Motorcycle



Truck



# Design Neural Network (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
**Design Neural Network**

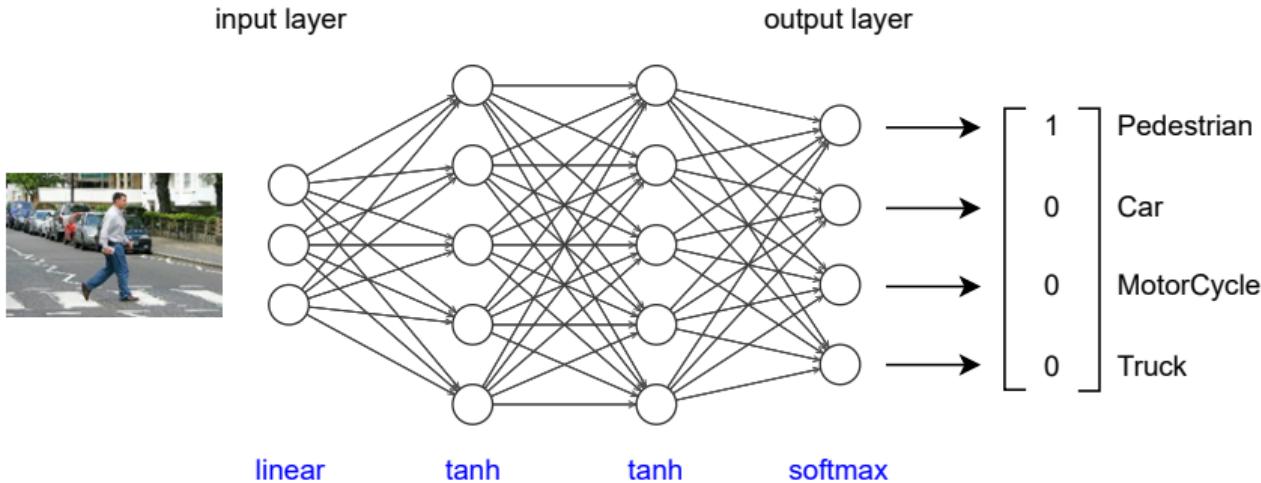
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning





# Neural Network Learning

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

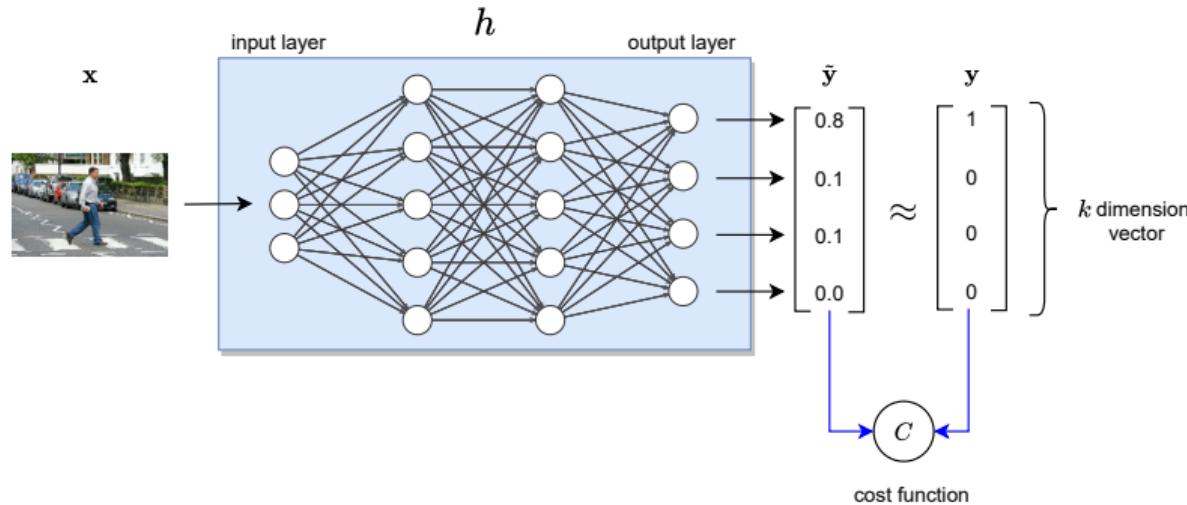
Neural Network  
Learning

Gradient-Based  
Learning  
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- In training step, we **fit** the network to the data based on a cost function





Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Cost Function

---

- In the neural network learning, the selection of **output layer** depends on the learning problems.
  - Classification: sigmoid, softmax .etc
  - Regression: linear etc.
- **Cost function** (lost function) can be derived from many methods, the most common of them
  - Mean Square Error
  - Cross Entropy



Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model

Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Cost Function (cont.)

## Concept 1 (Mean Square Error)

Let  $\mathbf{y}$  denotes the desired label of data  $\mathbf{x}$ , and  $\tilde{\mathbf{y}} = h(\mathbf{x})$  as the prediction. The cost function  $C_{MSE}$  is defined by

$$C_{MSE} = \frac{1}{k} \sum_{i=1}^k (y_i - \tilde{y}_i)^2 \quad (13)$$



Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Cost Function (cont.)

## Concept 2 (Cross Entropy)

The cost function defined by Cross Entropy is

$$C_{CE} = \sum_{i=1}^k y_i \ln \tilde{y}_i \quad (14)$$



# Cross Entropy vs. MSE

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

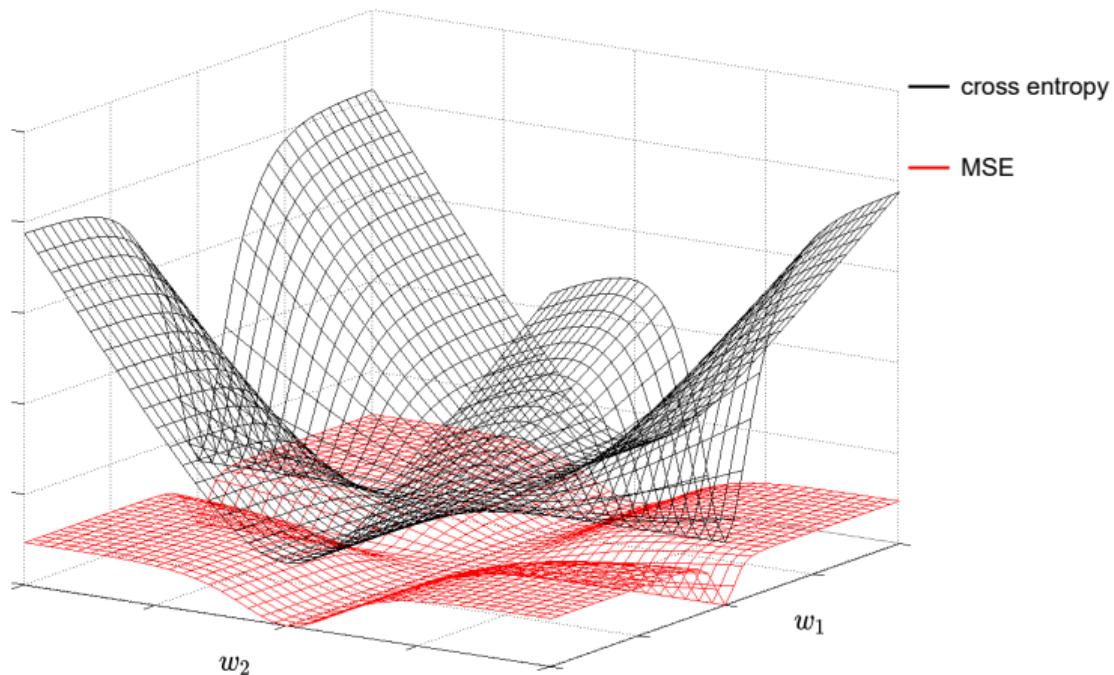
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning





# Output Types

Output Type	Output Distribution	Output Layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary cross-entropy
Discrete	Multinoulli	Softmax	Discrete cross-entropy
Continuous	Gaussian	Linear	Gaussian cross-entropy (MSE)
Continuous	Mixture of Gaussian	Mixture Density	Cross-entropy



# Regularization

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

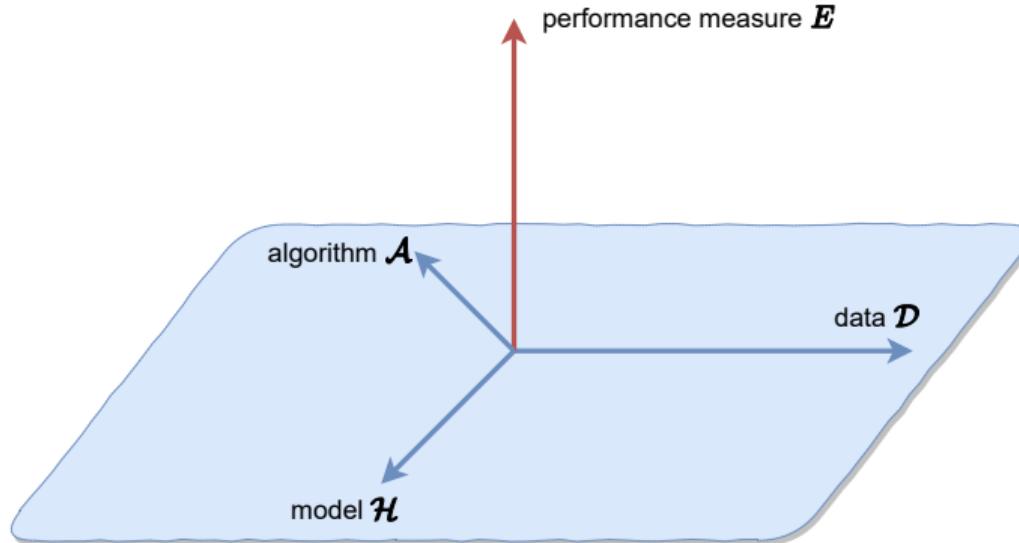
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

*“Regularization is any modification we make to a learning model that is intended to reduce its generalization error but not its training error.”*





Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

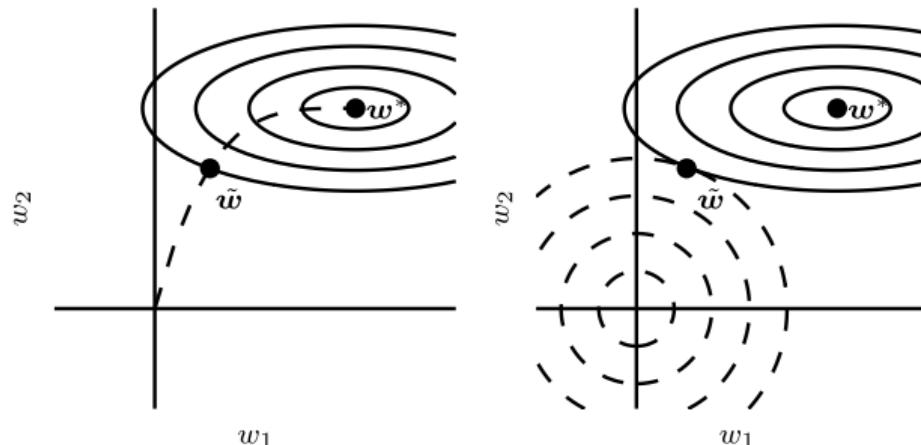
# Regularization (cont.)

- To avoid overfitting, we can use the regularized cost function (weight decay)

$$C_{reg}(\mathbf{w}) = C(\mathbf{w}) + \lambda \text{regularizer}(\mathbf{w}) \quad (15)$$

where  $\lambda$  is the regularization coefficient (*hyper-parameter*) that controls the relative importance of the data-dependent error  $C(\mathbf{w})$  and the regularization term

$$\text{regularizer}(\mathbf{w}) = \mathbf{w}^T \mathbf{w} \quad (16)$$





# Learning

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

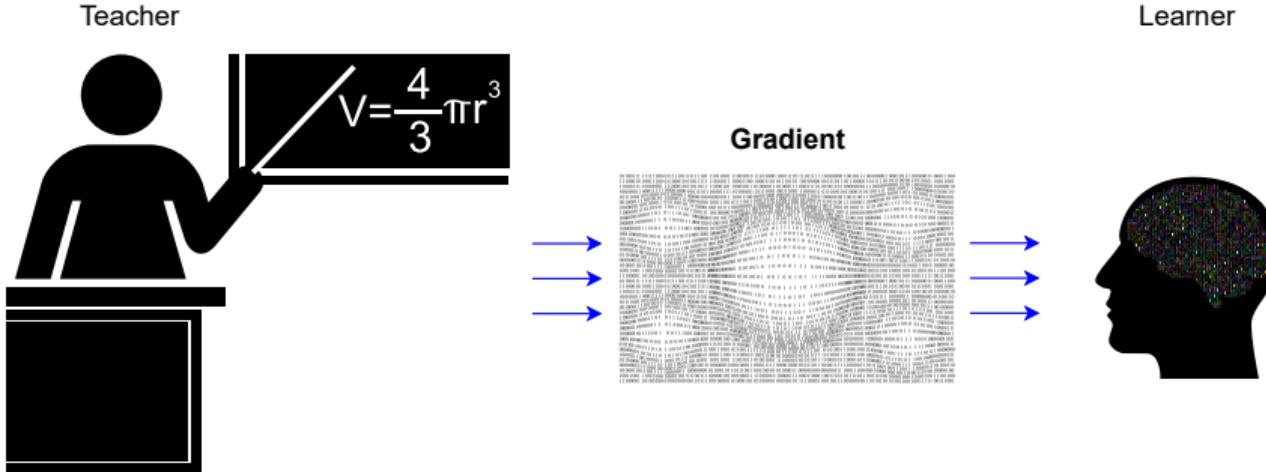
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- **Learning goal:** find  $h_w$  minimize the cost function  $C$ .
- **Solution:** use gradient-based techniques





# Gradient-Based Learning

- Gradient Descent
- Gradient Ascent
- Min-max optimization



# Gradient Descent

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

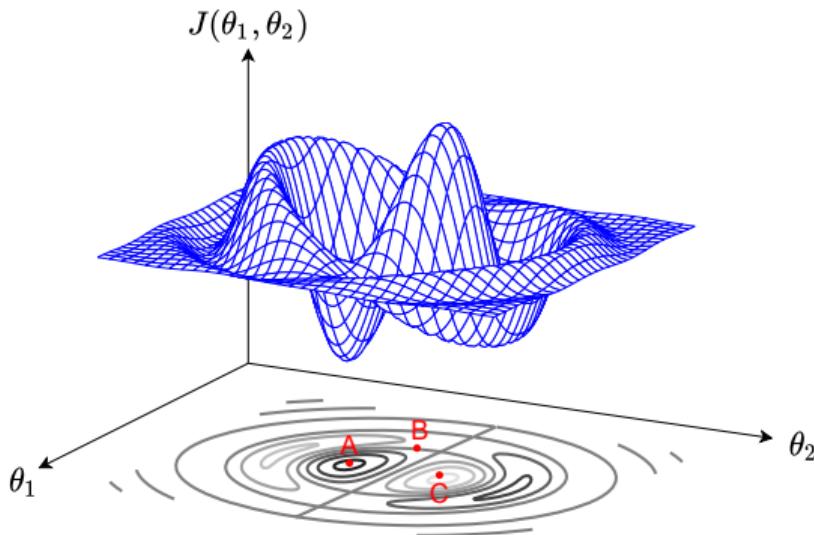
Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- **Objective:** minimize a objective function (cost function)

$$J(\theta) = J(\theta_1, \theta_2, \dots, \theta_D)$$

$$\hat{\theta} = \arg \min_{\theta} J(\theta) \quad (17)$$





# Gradient Descent (cont.)

- **Gradient descent method:** an iterative procedure where, at each iteration  $i$  we modify the parameters  $\theta$

1. Init  $\theta^{(0)}$

2. Do until satisfied

update  $\theta$  to reduce  $J(\theta)$

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_{\theta} J \quad (18)$$

where  $\eta$  is learning rate ( $0 < \eta < 1$ ) and

$$\nabla_{\theta} J = \left[ \frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}, \dots, \frac{\partial J}{\partial \theta_D} \right]^T \quad (19)$$



# Gradient Descent (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

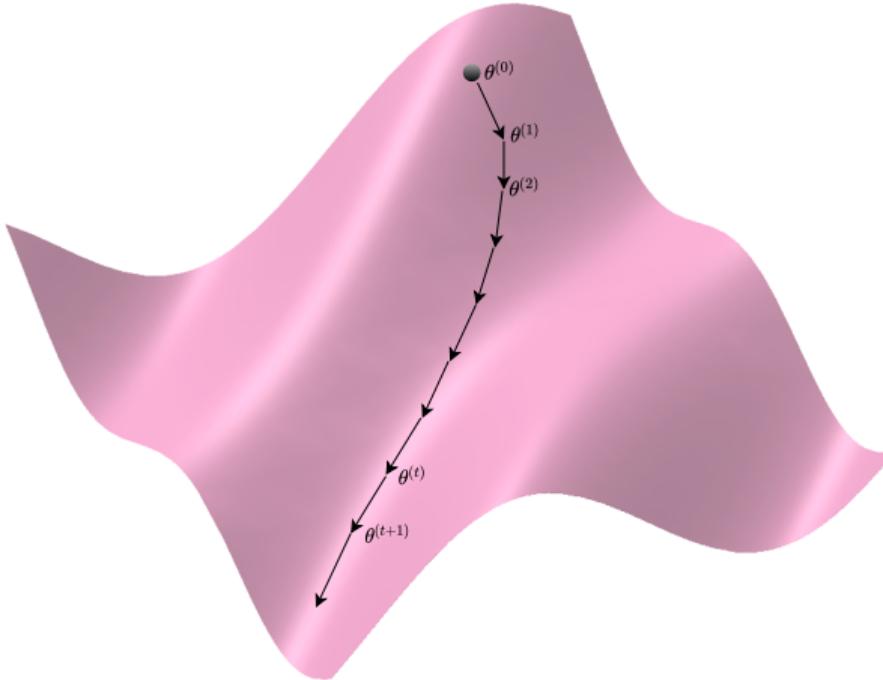
Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning





Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Gradient descent variants

---

There are three variants of gradient descent. Depending on the amount of data, we make a trade-off between the accuracy of the parameter update and the time it takes to perform an update.

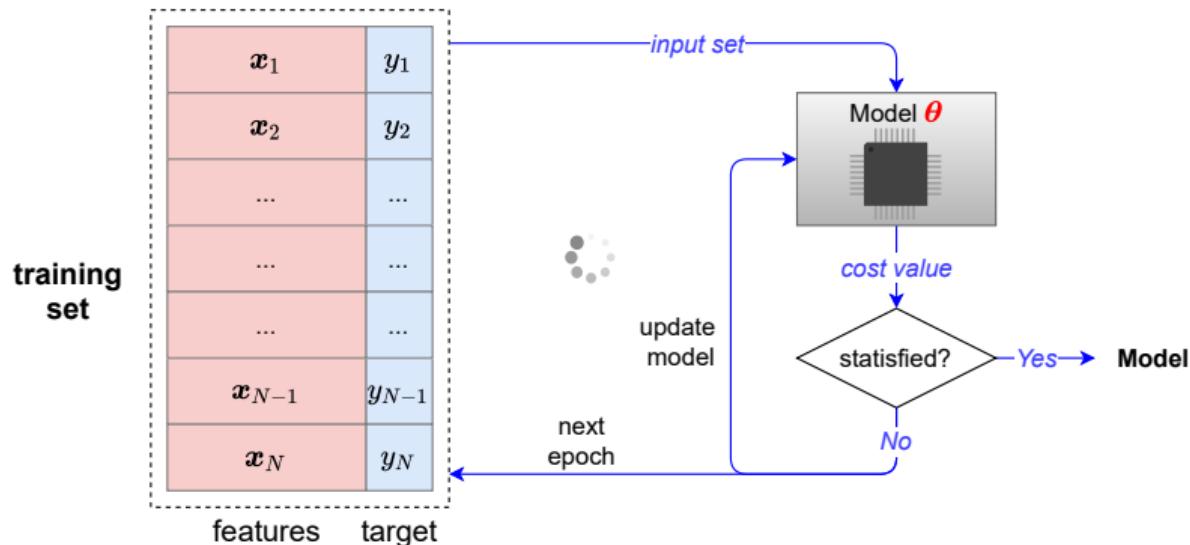
- Batch gradient descent
- Stochastic gradient descent
- Mini-batch gradient descent



# Batch gradient descent

- Batch gradient descent (vanilla gradient descent) computes the gradient of the cost function w.r.t. to the parameters  $\theta$  for the entire training dataset  $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_N, y_N)\}$

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_{\theta} J(\theta_t \mid \mathcal{D}) \quad (20)$$





Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Batch gradient descent (cont.)

---

## Cons

- We need to calculate the gradients for the whole dataset to perform just *one* update
- It can be very slow and is intractable for datasets that do not fit in memory.
- It also does not allow us to update our model *online*, i.e. with new examples on-the-fly.



# Stochastic gradient descent

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

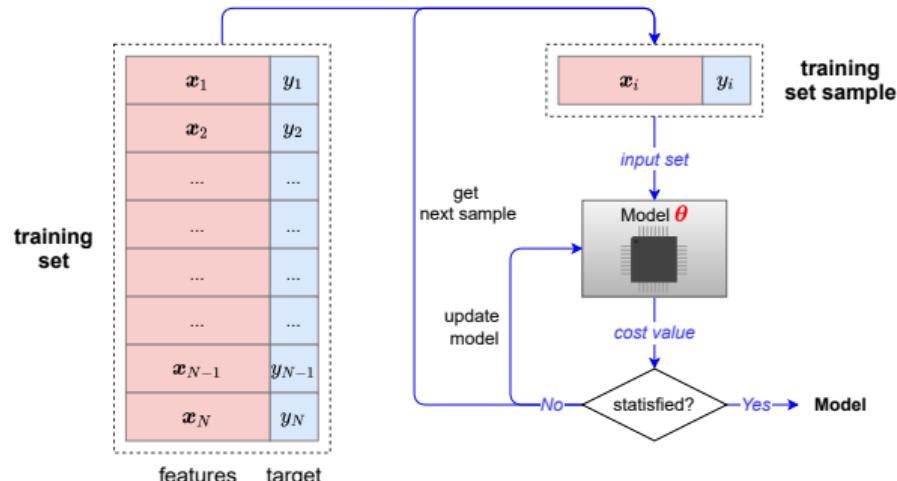
Back-  
propagation  
Learning

- Stochastic gradient descent (SGD) in contrast performs a parameter update for each training example  $\mathbf{x}_i$  and label  $y_i$  of the training dataset

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_N, y_N)\}$$

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_{\theta} J(\theta_t \mid \mathbf{x}_i, y_i) \quad (21)$$

- Note that we should **shuffle** the training data at every epoch





# Stochastic gradient descent (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

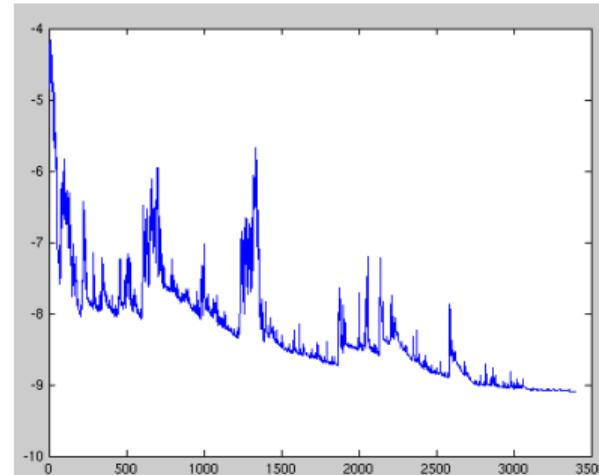
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

## Cons

- SGD performs frequent updates with a high variance that cause the objective function to fluctuate heavily

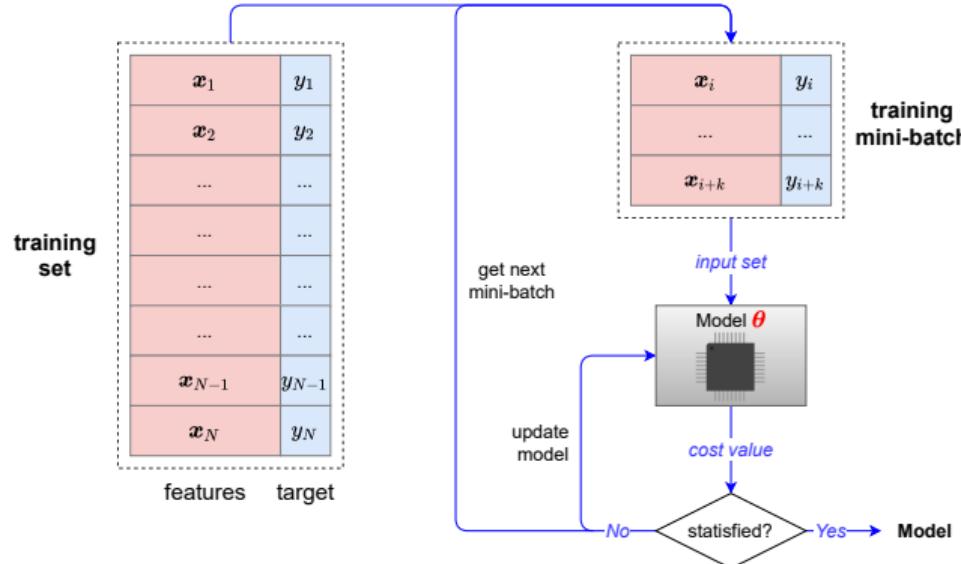




# Mini-batch gradient descent

- Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of  $k$  training examples of the training dataset  $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_N, y_N)\}$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t \mid \mathbf{x}_{i:i+k}, y_{i:i+k}) \quad (22)$$





Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Mini-batch gradient descent (cont.)

---

## Pros

- It reduces the variance of the parameter updates, which can lead to more stable convergence
- It can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient.



# Gradient Ascent

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- **Objective:** maximize a objective function (cost function)

$$J(\theta) = J(\theta_1, \theta_2, \dots, \theta_D)$$

$$\hat{\theta} = \arg \max_{\theta} J(\theta) \quad (23)$$

1. **Init**  $\theta^{(0)}$
2. **Do until satisfied**  
update  $\theta$  to increase  $J(\theta)$

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \eta \nabla_{\theta} J \quad (24)$$

where  $\eta$  is learning rate ( $0 < \eta < 1$ ) and

$$\nabla_{\theta} J = \left[ \frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}, \dots, \frac{\partial J}{\partial \theta_D} \right]^T \quad (25)$$



# Min-max

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

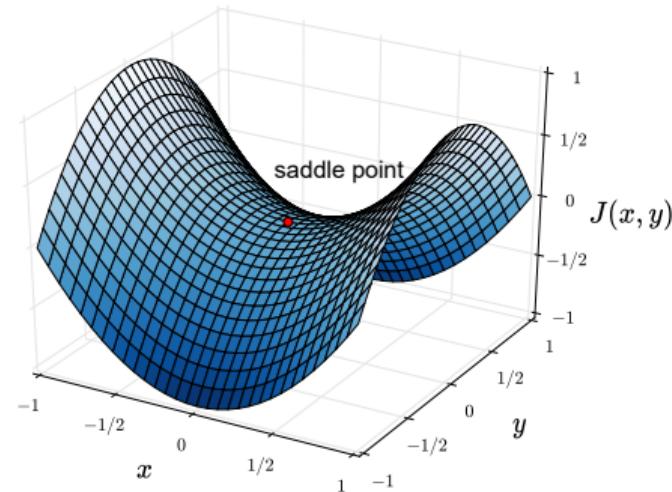
Gradient-Based  
Learning  
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- **Objective:** Given a objective function (cost function)  $J(x, y)$

$$\hat{x}, \hat{y} = \arg \min_x \max_y J(x, y) \quad (26)$$





Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

## Min-max (cont.)

- GAN optimization (original version), does not always converge

1. **Init**  $x^{(0)}, y^{(0)}$

2. **Do until satisfied**  
update  $x, y$

$$x^{(t+1)} \leftarrow x^{(t)} - \rho \nabla_x J \quad (27)$$

$$y^{(t+1)} \leftarrow y^{(t)} + \eta \nabla_y J \quad (28)$$



# Advanced Gradient-Based Learning



# Gradient descent: Local minima

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network

Neural Network  
Learning

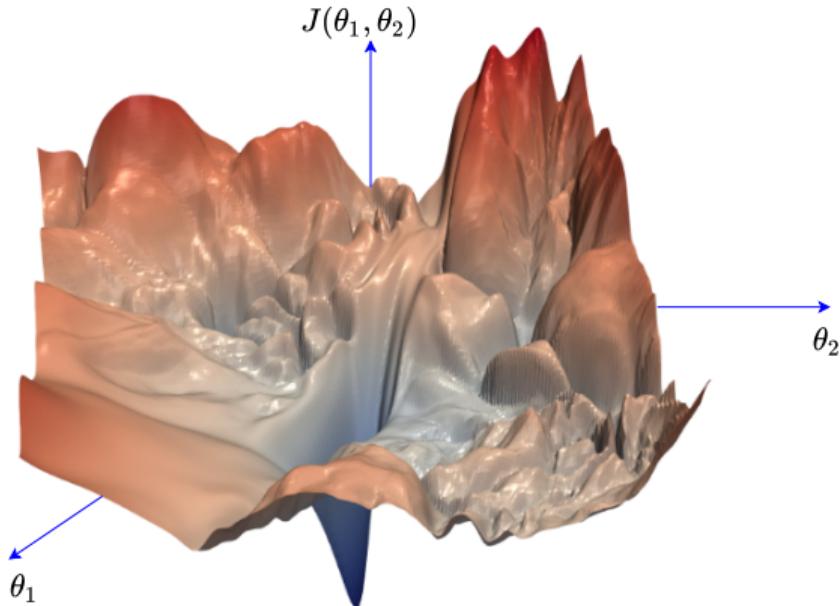
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

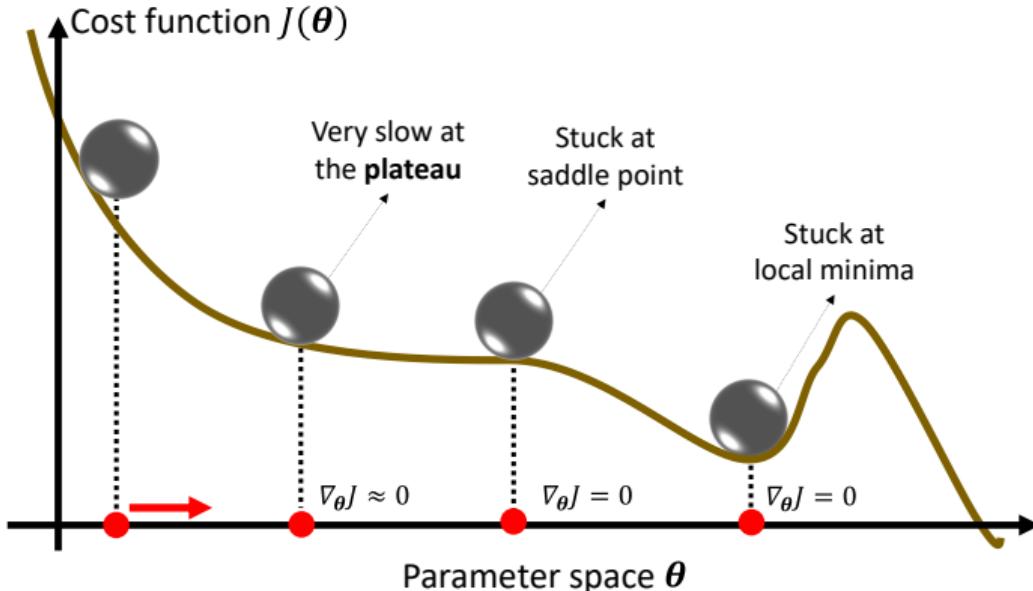
- Gradient descent never guarantee **global minima**
  - Different initial point  $\theta^{(0)}$  → reach different minima, so different results





# More issues of gradient descent

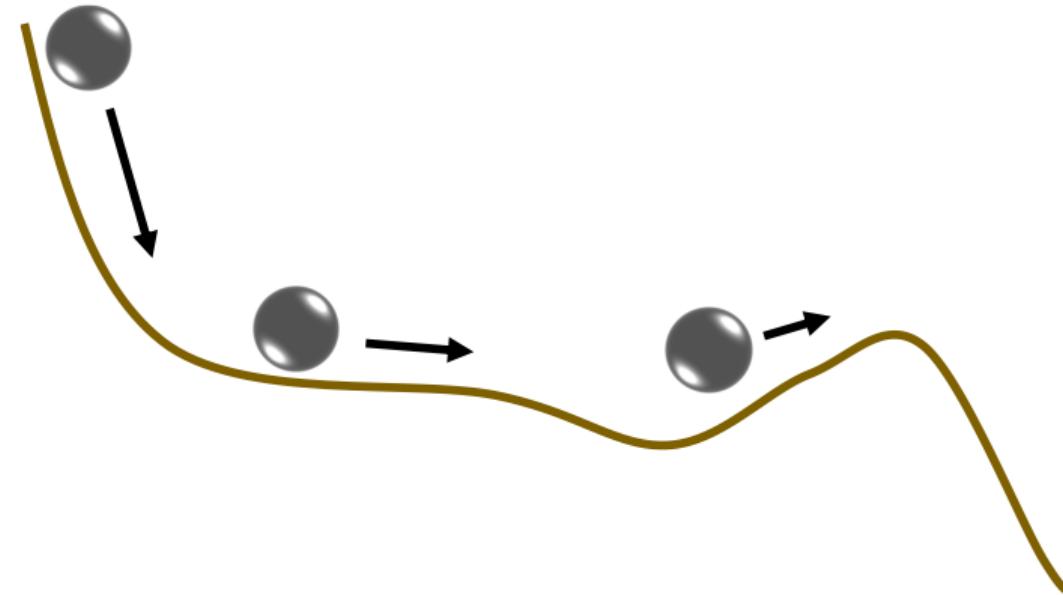
- It also has issues at **plateau** and **saddle point**





# Momentum in physical world

- How about put this phenomenon in gradient descent?





# Momentum

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

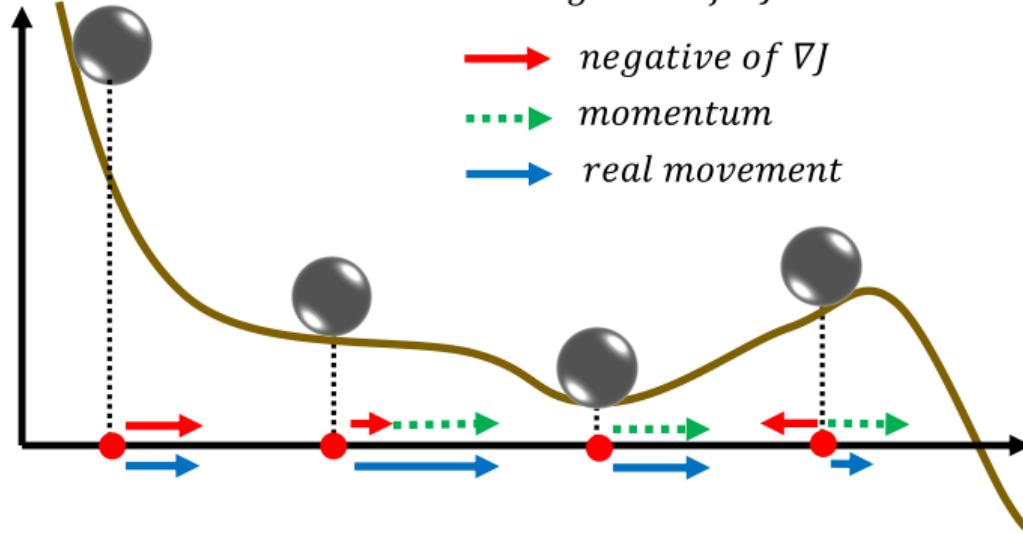
Gradient-Based  
Learning  
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- Still not guarantee reaching global minima, but give some hope ...

*movement = negative of  $\nabla J$  + momentum*





# Momentum

- **Momentum** is a method that helps accelerate SGD in the relevant direction and dampens oscillations
- It does this by adding a fraction  $\gamma$  of the update vector of the past time step to the current update vector

$$\mathbf{v}_0 = \mathbf{0}$$

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \eta \nabla_{\theta} J(\theta_{t-1}) \quad (29)$$

$$\theta_t = \theta_{t-1} - \mathbf{v}_t$$

- The momentum term  $\gamma$  is usually set to 0.9 or a similar value.



Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning  
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Nesterov accelerated gradient (NAG)

- A ball that rolls down a hill, blindly following the slope, is highly unsatisfactory. We would like to have a smarter ball, a ball that has a notion of where it is going so that it knows to slow down before the hill slopes up again.
- We can now effectively look ahead by calculating the gradient not w.r.t. to our current parameters  $\theta$  but w.r.t. the approximate future position of our parameters:

$$\mathbf{v}_0 = \mathbf{0}$$

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \eta \nabla_{\theta} J(\theta_{t-1} - \gamma \mathbf{v}_{t-1}) \quad (30)$$

$$\theta_t = \theta_{t-1} - \mathbf{v}_t$$



# Nesterov accelerated gradient (NAG) (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

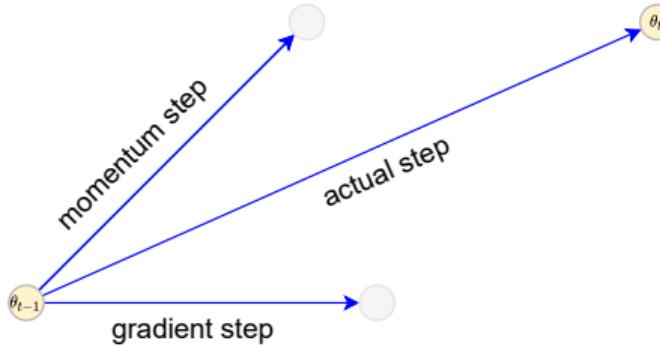
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

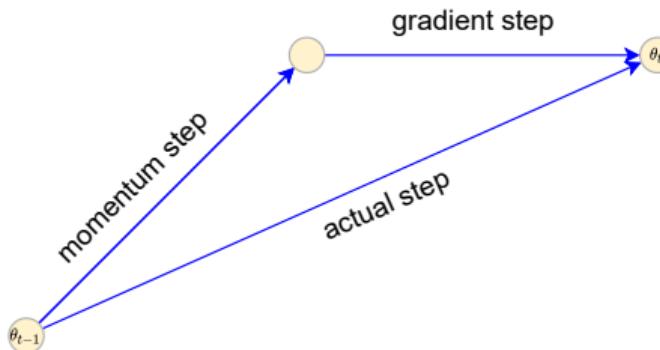
Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

Momentum update



Nestorov update





# Adagrad

- Previously, we performed an update for all parameters  $\theta$  at once as every parameter  $\theta_i$  used the **same learning rate**  $\eta$ . As Adagrad (Duchi, 2011) uses a **different learning rate** for every parameter  $\theta_i$  at every time step  $t$

$$\theta_{t+1,i} = \theta_{t,i} - \eta_{t,i} \cdot g_{t,i} \quad (31)$$

where

$$g_{t,i} = \frac{\partial J}{\partial \theta_{t,i}} \quad (32)$$

and

$$\eta_{t,i} = \frac{\eta}{\sqrt{\sum_{k=0}^t g_{k,i}^2}}$$



Artificial Neural  
Network

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning  
Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

# Adagrad (cont.)

---

## Pros

- One of Adagrad's main benefits is that it eliminates the need to manually tune the learning rate  $\eta$ . Most implementations use a default value of 0.01 and leave it at that.

## Cons

- Since every added term is positive, the accumulated sum keeps growing during training. This in turn causes the learning rate to shrink and eventually become infinitesimally small, at which point the algorithm is no longer able to acquire additional knowledge.



# Other Learning Methods rather than Adagrad

- **Adadelta** (Zeiler, 2012) is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size  $w$ .
- **RMSprop** is an adaptive learning rate method proposed by Geoff Hinton.
- **Adadelta**
- **Adam**
- **AdaMax**
- **Nadam**



# Back-propagation Learning



# Computational Graph

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model

Design Neural Network  
Neural Network  
Learning

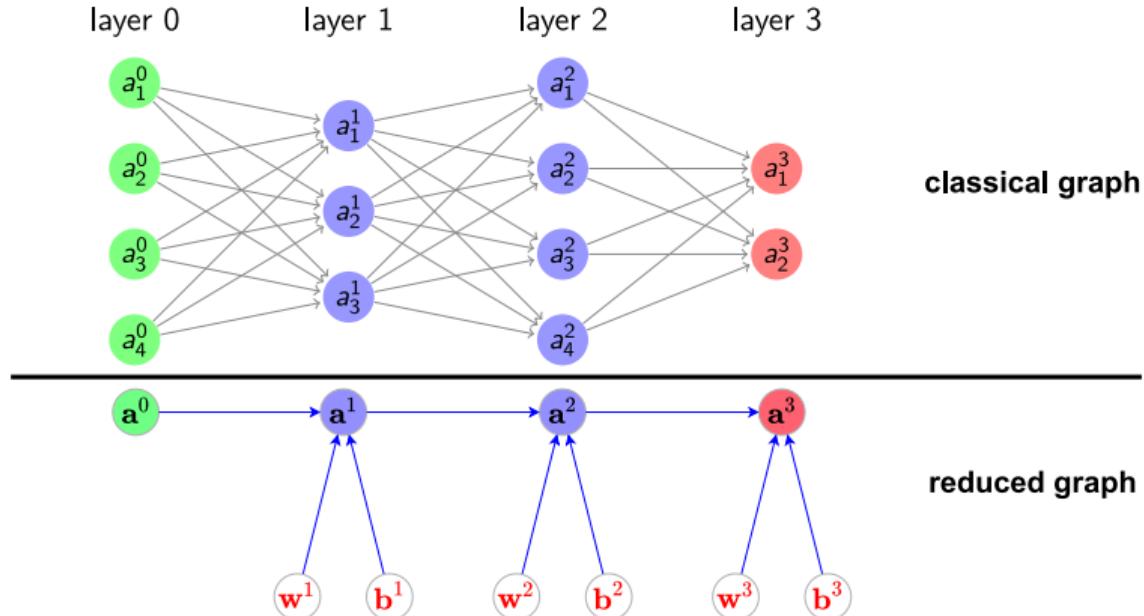
Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- The **classical graph** of neural network is cumbersome at first, and it does take some work to master
- We use the another representation, **reduced graph (computational graph)**





# The back-propagation algorithm

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

- Proposed by Bryson and Ho, 1969 → **most popular** among over a hundred different learning algorithms available.
- The algorithm provide us with a way of computing the gradient of the cost function  $C(\mathbf{w}, \mathbf{b})$ .
- **Input**  $(\mathbf{x}, \mathbf{y})$ : Set the corresponding activation  $\mathbf{a}^0$  for the input layer.

## Feedforward:

1. Init  $\mathbf{a}^0 = \mathbf{x}$
2. For each  $l = 1, 2, \dots, L$  compute

$$\mathbf{z}^l = \mathbf{w}^l \mathbf{a}^{l-1} + \mathbf{b}^l \quad \text{and} \quad \mathbf{a}^l = \sigma(\mathbf{z}^l) \quad (33)$$

where  $\sigma$  is an activation function

3. Compute the cost value  $C$



# The back-propagation algorithm (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning

## Backpropagate:

1. Compute **the error vector**  $\delta^L$

$$\delta^L = \sigma'(\mathbf{z}^L) \odot \nabla C \quad (34)$$

2. For each  $l = L - 1, L - 2, \dots, 1$ , compute **the error vector**  $\delta^l$

$$\delta^l = ((\mathbf{w}^{l+1})^\top \delta^{l+1}) \odot \sigma'(\mathbf{z}^l) \quad (35)$$

3. Compute the partial gradient of the cost function with respect to parameters

$$\frac{\partial C}{\partial \mathbf{w}_{jk}^l} = a_k^{l-1} \delta_j^l \quad \text{and} \quad \frac{\partial C}{\partial b_j^l} = \delta_j^l \quad (36)$$



# The back-propagation algorithm (cont.)

Perceptron:  
The simplest  
ANN  
(Revisited)

Multilayer  
Perceptrons  
(MLP)

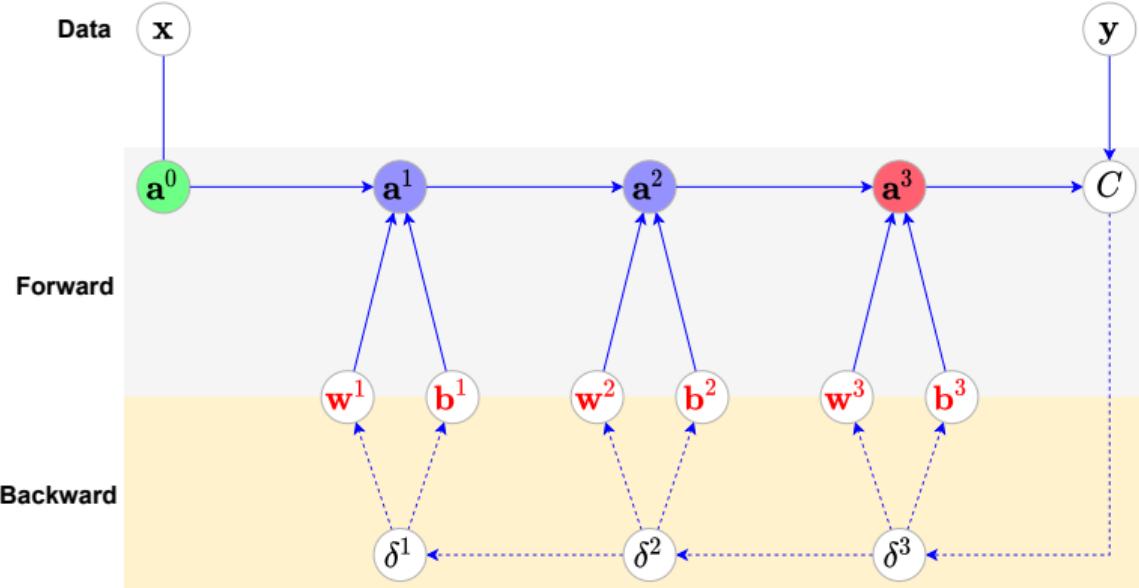
Learning Model  
Design Neural Network  
Neural Network  
Learning

Gradient-Based  
Learning

Gradient Descent  
Gradient Ascent  
Min-max optimization

Advanced  
Gradient-Based  
Learning

Back-  
propagation  
Learning





## References

---

- ❑ Goodfellow, I., Bengio, Y., and Courville, A. (2016).  
*Deep learning.*  
MIT press.
- ❑ Lê, B. and Tô, V. (2014).  
*Cở sở trí tuệ nhân tạo.*  
Nhà xuất bản Khoa học và Kỹ thuật.
- ❑ Nguyen, T. (2018).  
Artificial intelligence slides.  
Technical report, HCMC University of Sciences.
- ❑ Russell, S. and Norvig, P. (2016).  
*Artificial intelligence: a modern approach.*  
Pearson Education Limited.