# Artificial Neural Networks

Inteligencia Artificial en los Sistemas de Control Autónomo
Máster Universitario en Ingeniería Industrial

Departamento de Automática

Universidad de Alcalá
ISG

## Objectives

1. Describe biological neurons and networks
2. Basics of artifical neurons and networks
3. Understand the role of trainning in ANNs
4. Strengths and weaknesses of ANNs

## Bibliography


- A. Tettamanzi, M. Tomassini. *Soft Computing. Integrating Evolutionary, Neural, and Fuzzy Systems.* Springer-Verlag. 2001
- McCulloch, W. and Pitts, W. (1943). *A logical calculus of the ideas immanent in nervous activity.* Bulletin of Mathematical Biophysics, 7:115 - 133.
- Rosenblatt, Frank. (1958). *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.* Psychological Review, 65:386-408

# Table of Contents

# Introduction

## History

1888 Ramón y Cajal. Discovery of biological neurons

1943 McCulloch & Pitts. First neural network designers

1949 Hebb. First learning rule

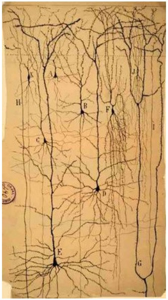
1958 Rosenblatt. Perceptron

1969 Minsky & Papert. Perceptron limitation - Death of ANN

1986 Rumelhart *et al.* Re-emergence of ANN: Backpropagation

201x Convolutional Neural Networks (CNNs) - Deep learning

2014 Goodfellow *et al.* Generative Adversarial Networks (GANs)



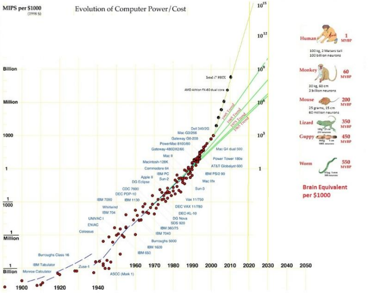
# Introduction

## Structure of neurons (I)

| Animal | Neurons |
|---:|---:|
| Sponge | 0 |
| Roundworm | 302 |
| Jellyfish | 800 |
| Ant | 250,000 |
| Cockroach | 1,000,000 |
| Frog | 16,000,000 |
| Mouse | 71,000,000 |
| Cat | 760,000,000 |
| Macaque | 6,376,000,000 |
| Human | 86,000,000,000 |
| Elephant | 267,000,000,000 |

### Human brain

Neuron switching time: 0.001 s
Synapsis: 10-100 thousand
Scene recognition time: 0.1 s




(Source)

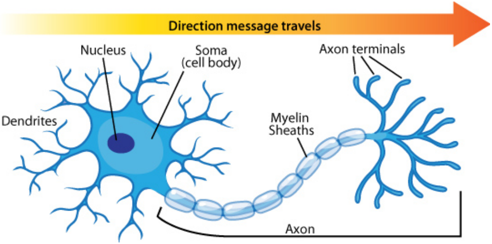
# Introduction

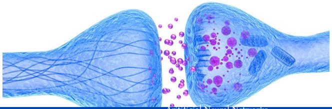
## Structure of neurons (II)

A neuron has a cell body ...

- ... a branching input structure (dendrite) and
- ... a branching output structure (axon)

Axons connect to dendrites via synapses



# Introduction

## Structure of neurons (III)

A neuron only fires if its input signal exceeds a threshold

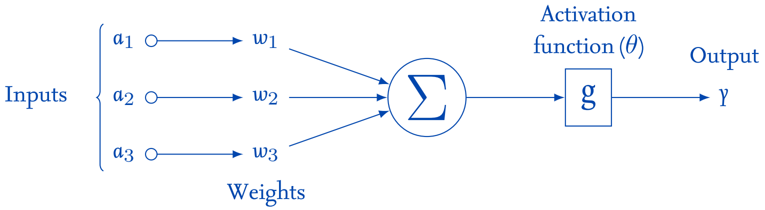
- Good connections allowing a large signal
- Slight connections allowing a weak signal
- Synapses may be either excitatory or inhibitory

Synapses vary in strength

- Biological learning involves setting that strength

# Artificial neurons

## Definition (I)




$a_i$ Normalized input ($0 \leq a_i \leq 1$)

$w_i$ Weight of input j

$\theta$ Threshold

g Activation function

**Neuron model (perceptron)**

$$y = g\left(\sum_{i=1}^{n} w_i a_i\right)$$

# Artificial neurons

## Definition (II)

- Each neuron has a threshold value
- Each neuron has weighted inputs
- The input signals form a weighted sum
- If the activation level exceeds the threshold, the neuron activates

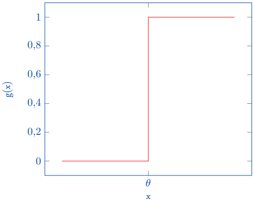
# Artificial neurons

## Definition (III)

The idealized activation function is a step function

$$g(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} w_i x_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

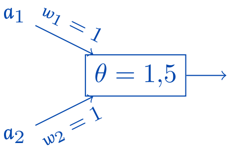
The step function is rarely used in practice



# Artificial neurons

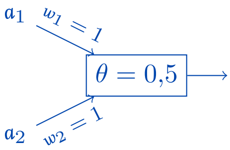
## Logical gates with a neuron

AND



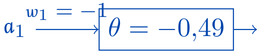

| $a_1$ | $a_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR




| $a_1$ | $a_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

NOT

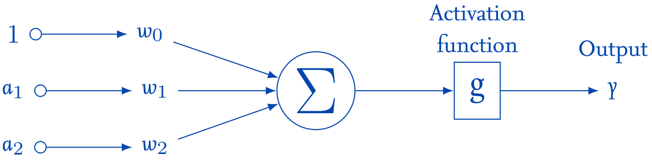



| $a_1$ | $y$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

(A neuron in Excel)

# Artificial neurons

## Definition of neuron (alternative version)




- $a_i$ Normalized input ($0 \leq a_i \leq 1$)
- $w_i$ Weight of input j
- $w_0$ Bias
- g Activation function

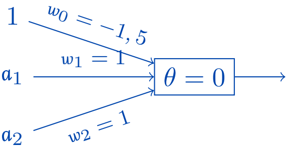
**Neuron model**

$$y = g\left(\sum_{i=0}^{n} w_i a_i\right)$$

# Artificial neurons

## Example of biased neuron

AND logical gate with a biased input

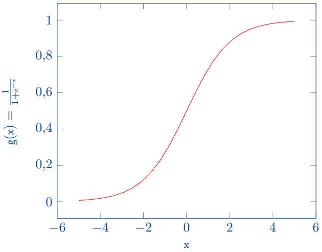



| $a_0$ | $a_1$ | $a_2$ | Output |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Artificial neurons

## Activation functions: Sigmoid function

- Biological motivation
- S-shaped, continuous and everywhere differentiable
- Asymptotically approach saturation points
- Derivative fast computation
- Range $\in [0, 1]$




**Sigmoid function**

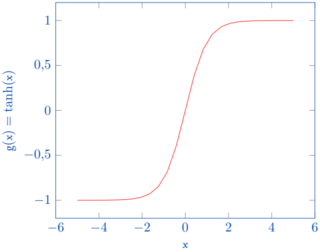
$$g(x) = \frac{1}{1+e^{-x}}$$

$$g'(x) = g(x)(1 - g(x))$$

# Artificial neurons

## Activation functions: Tanh function

- Asymptotically approach saturation points
- Range $\in [-1, 1]$
- Bigger derivative than sigmoid (faster training)




### Tanh function

$$g(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

$$g'(x) = 1 - g(x)^2$$

# Artificial neurons

## Activation functions: Softmax function

- Generalization of the logistic function
- Usually used in the output layer in classification problems
- Asymptotically approach saturation points

### Softmax function

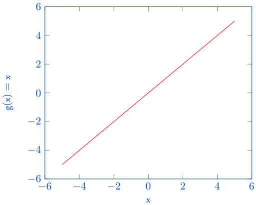
$$g(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1, \ldots, K$$

with $\mathbf{z}$ a K-dimensional vector

# Artificial neurons

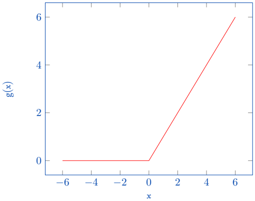
## Other activation functions

### Linear function



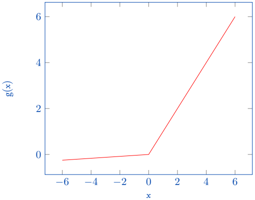

- Used in regression

### Rectified Linear (ReLU)




- Faster derivate
- Popular in DL

### Leaky ReLU

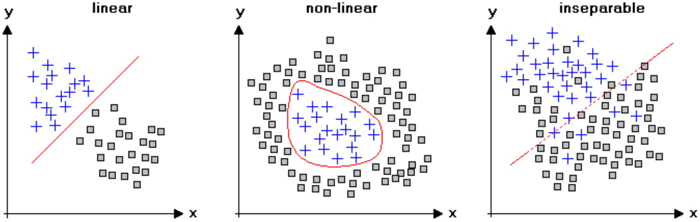



- More informative derivate
- Popular in DL

The lack of non-linear activation function makes a network a simple linear regression

# Artificial neurons

## Learning limits (I)

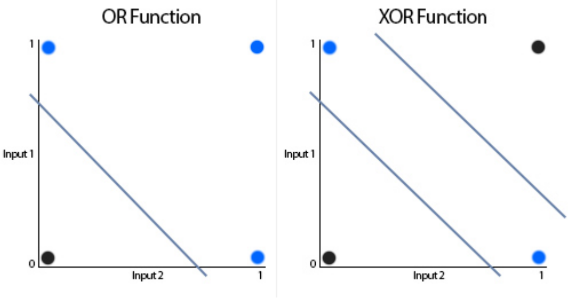



Problem: A single neuron only can solve linearly separable problems

# Artificial neurons

## Learning limits (II)

XOR cannot be implemented with a neuron




Solution: Neuronal networks

# Artificial Neural Networks

## Definition (II)

ANN properties

- Noise tolerance
- General function approximator

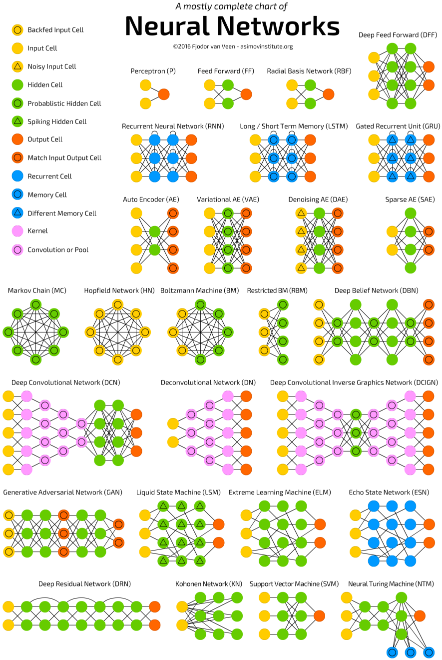
Machine Learning tasks

- Supervised learning (classification and regression)
- Unsupervised learning (known as self-organizing maps in ANN terminology)

Many topologies

- Acyclic, recurrent (cyclic), modular, etc
- Feed Forward networks (MLPs)

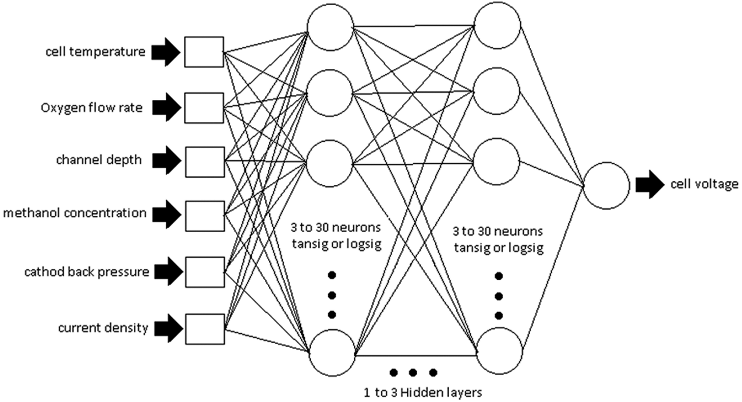
*Human readability less important than performance*


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

(More info)

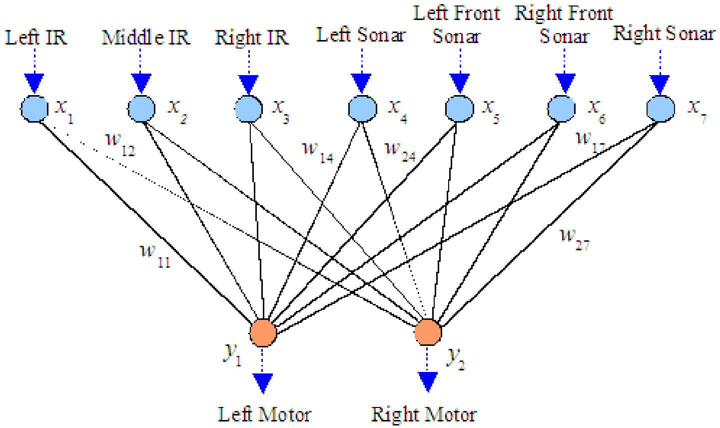
# Artificial Neural Networks

## Application examples (I)



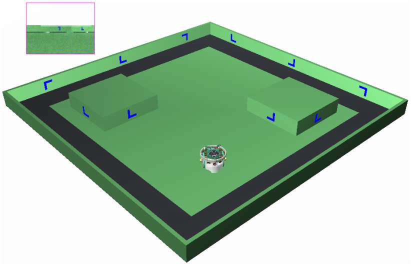
# Artificial Neural Networks

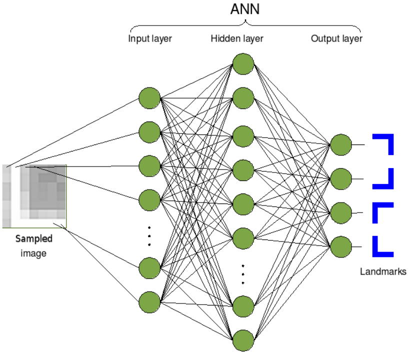
## Application examples (II)



# Artificial Neural Networks

## Application examples (III)

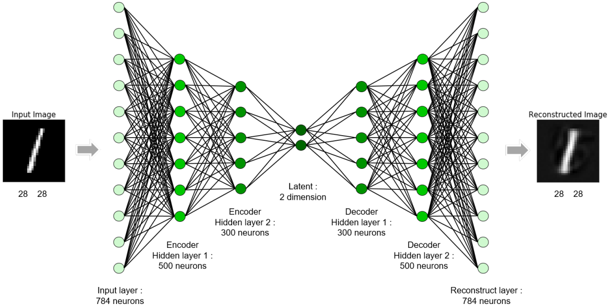


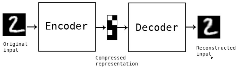



(Source)

# Artificial Neural Networks

## Application examples (IV)




(Source)

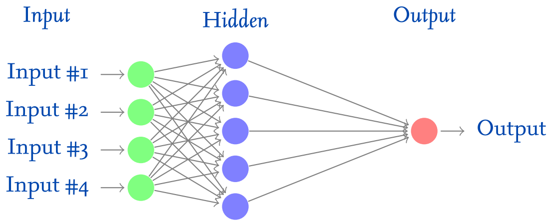
# Feedforward networks

## Definition (I)

Neurons are arranged in layers

- *Input* Which consists of any normalized data
- *Output* Which are the net outcome
- *Hidden* (Optional) No direct interaction

Also known as **multilayer perceptron** (MLP)



# Feedforward networks

## Definition (II)

The input layer

- Introduces input values into the network
- No activation function or other processing

The hidden layer(s)

- Perform classification of features
- Two hidden layers are sufficient to solve any problem
- Features imply more layers may be better

The output layer

- Functionally just like the hidden layers
- Outputs are passed on to the world outside the neural network

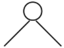
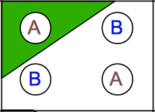
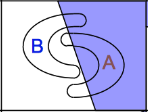
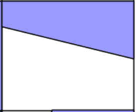
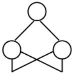
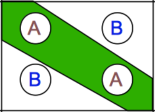
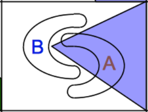
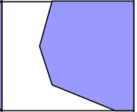
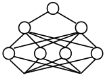
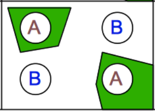
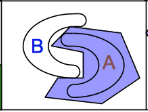
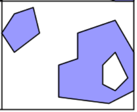
# Feedforward Networks

## Demo

(Online demo)

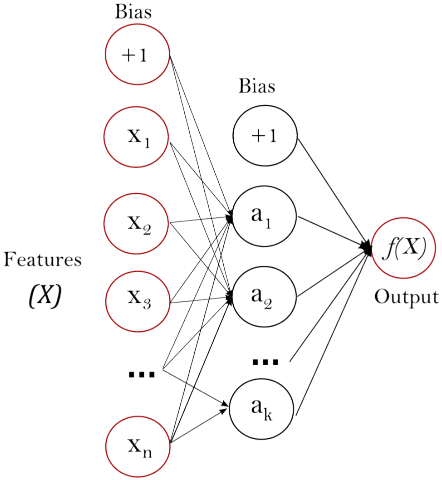
# Feedforward Networks

## Separability

| Structure | Types of Decision Regions | Exclusive-OR Problem | Classes with Meshed regions | Most General Region Shapes |
|---|---|---|---|---|
| *Single-Layer*  | *Half Plane Bounded By Hyperplane* |  |  |  |
| *Two-Layer*  | *Convex Open Or Closed Regions* |  |  |  |
| *Three-Layer*  | Arbitrary (Complexity Limited by No. of Nodes) |  |  |  |

# Feedforward Networks

## Bias in a MLP




(Source)

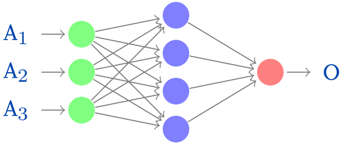
# Training algorithms

## Problem statement (I)

ANNs can perform different tasks

- Classification, regression, others

Classification (or supervised learning) uses a *training set*




| $A_1$ | $A_2$ | $A_3$ | O | Y |
|---|---|---|---|---|
| 1,1 | 2,5 | 4,5 | 0,2 | $-0{,}1$ |
| 0,9 | 2,4 | 1,2 | 0,5 | 0,4 |
| 1,0 | 2,0 | 9,9 | 0,4 | 1,2 |

Toss function: Measure of the error

- Y and O are the desired and observed outputs
- Usually mean squared error (MSE): $f(w) = E = \frac{1}{2}(y - o)^2$

# Training algorithms

## Problem statement (II)

Problem: Determine $\vec{w}$ that minimize $f(\vec{w})$

- Remember, $\vec{w}$ is our network
- This is a classical optimization problem
- Any optimization algorithm can be used
- ... in AI, optimization means search

We do know anatically $f(\vec{w}) \Rightarrow$ Optimization based on gradients

# Training algorithms

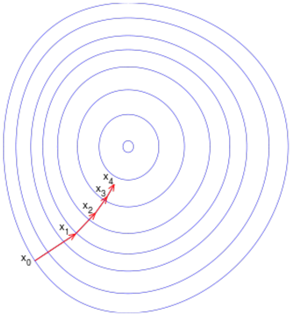
## Gradient Descent (I)

Calculate the gradient of the loss function with respect weights

- Adjust weights along gradient direction
- Gradient provides the direction
- $\alpha$ is the learning rate ($|\alpha| < 1$)

### Gradient descent

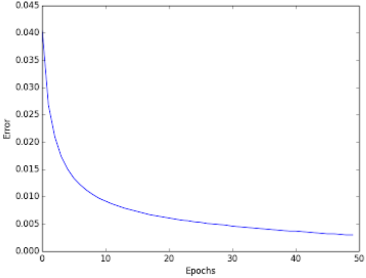
1: $\vec{w} \leftarrow random()$
2: **while** Not converged **do**
3: **for all** $w_i \in \vec{w}$ **do**
4: $w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} f(\vec{w})$
5: **end for**
6: **end while**



# Training algorithms

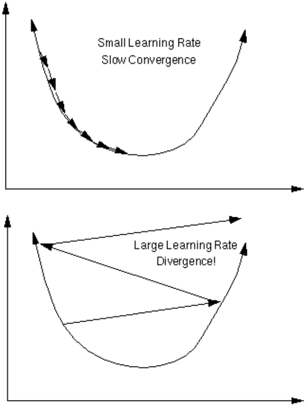
## Gradient Descent (II)

Each iteration is named epoch




(Source)

### Learning rate




(Source)

# Training algorithms

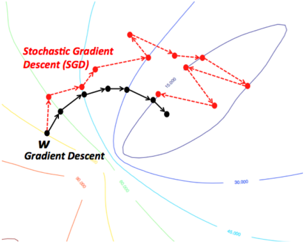
## Stochastic Gradient Descent (I)

SDG approximates the gradient sampling the dataset

| | |
|---|---|
| On-line | One sample |
| Mini-batch | Several samples |
| Batch | All the samples (Gradient Descent) |

Computations are faster ...

- ... but gradient estimation looses accuracy




(Source)

# Training algorithms

## Stochastic Gradient Descent (II)

Usually, a momentum is introduced as

$$w^{k+1} = w^{k} - \alpha z^{k+1}$$

with

$$z^{k+1} = \beta z^{k} + \nabla g(\mathrm{in})$$

where ...

- $\alpha$ is the learning rate
- $\beta$ is the momentum strength
- If $\beta = 0$ then gradient descent

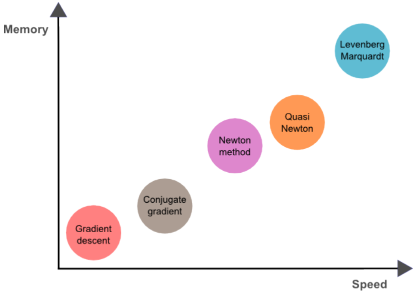
(On-line demo)

# Training algorithms

## Other optimization algorithms (I)

Other second derivative-based optimization algorithms

- Newton's method
- Quasi-Newton's method
- Levenberg-Marquardt method
- Conjugate Gradient




(Source)

# Training algorithms

## Other optimization algorithms (II)

Learning reate / momentum adaptative methods

- AdaGrad - Adaptative Gradient Algorithm
- RMSProp - Root Mean Square Propagation
- Adam - Adaptive Moment Estimation

# Training algorithms

## Backpropagation

Backpropagation is an efficient algorithm to compute gradients

- It applyes the chain rule to propagate errors
- Implicit in optimization algorithms

### Backpropagation algorithm

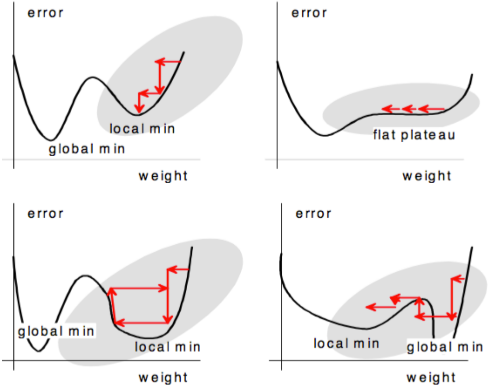
1. Compute output
2. Compute error
3. For each layer, repeat the following steps
   3.1 Propagate errors backwards
   3.2 Update weights between two layers

# Training algorithms

## Learning problems

Potential problems

- Local minima
- Flat plateau
- Oscillation
- Missing good minima



# Training algorithms

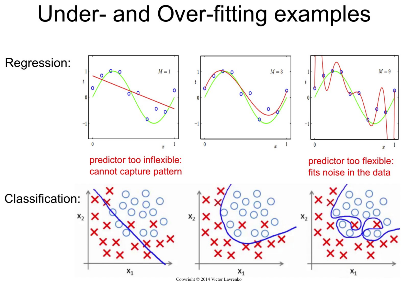
## Learning problems: Under and overfitting (I)

Underfitting: Does not learn

- Topology too simple

Overfitting: Memorizes samples

- Topology too complex
- Perhaps, the most serious concern in ML
- The net fails when exposed to new data



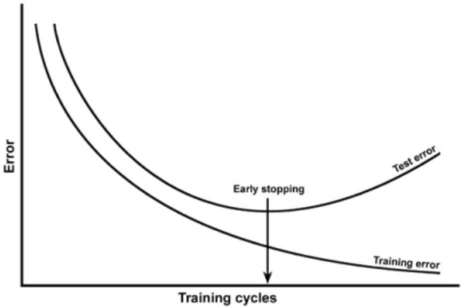

(Source)

# Training algorithms

## Learning problems: Under and overfitting (II)

Solution: Evaluate generalization capabilities

- Split training and validation sets and measure errors




(Source)

# Acknowlegements


- Jesus Aguilar Ruiz, Pablo de Olavide, Seville, Spain
- Daniel Rodríguez, Universidad de Alcalá, Alcalá de Henares, Spain