# *Deep Learning*: Fundamentos de redes neuronales

**Siham Tabik**

**Dpto. Ciencias de la Computación e I.A.**
**Universidad de Granada**
**siham@ugr.es**

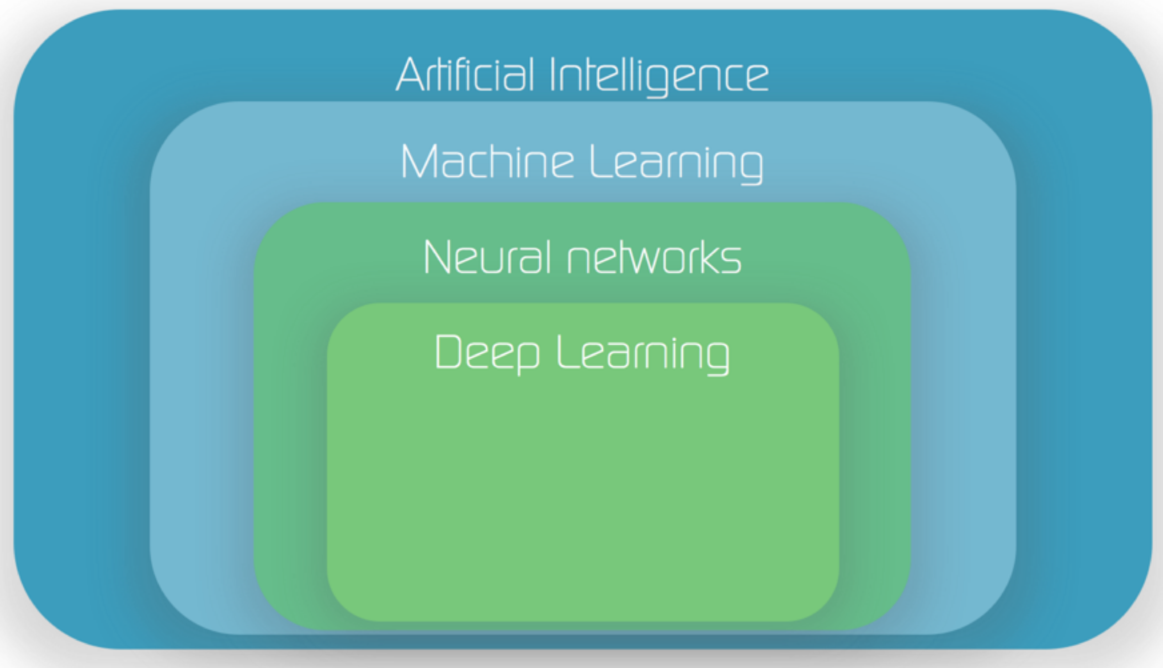
UNIVERSIDAD DE GRANADA

DaSCI

# Outline

- Intro Artificial Neural Networks(ANN)
- How ANN Learn?
- Simple Classifier
- Loss Function
- Gradient Descent
- Backpropagation
- Activation Functions

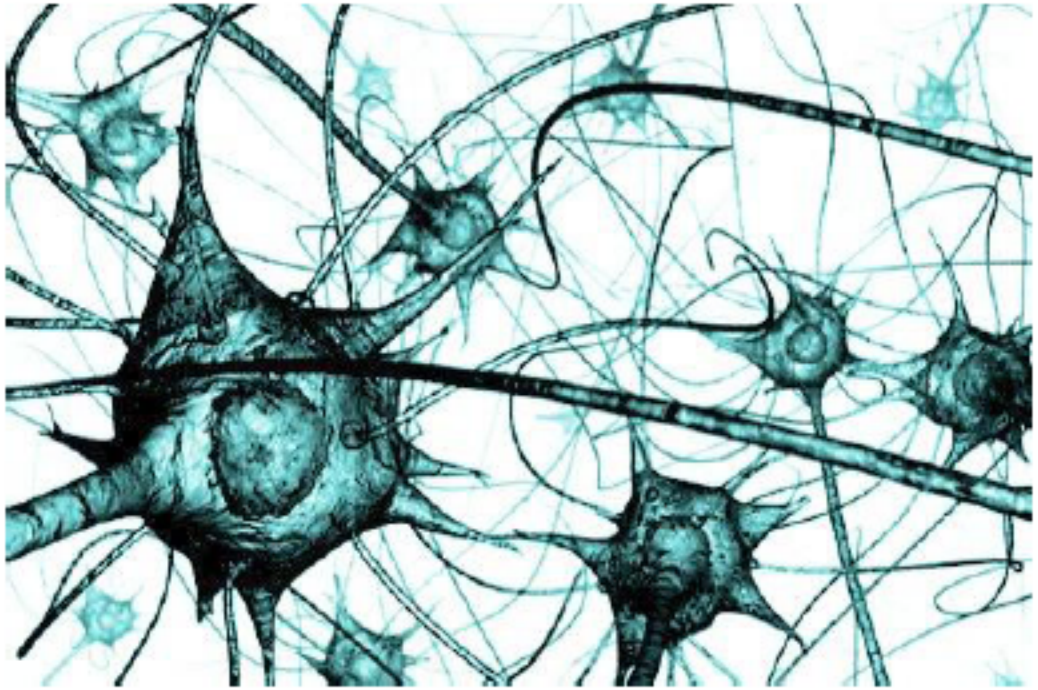
# Artificial Neural Networks

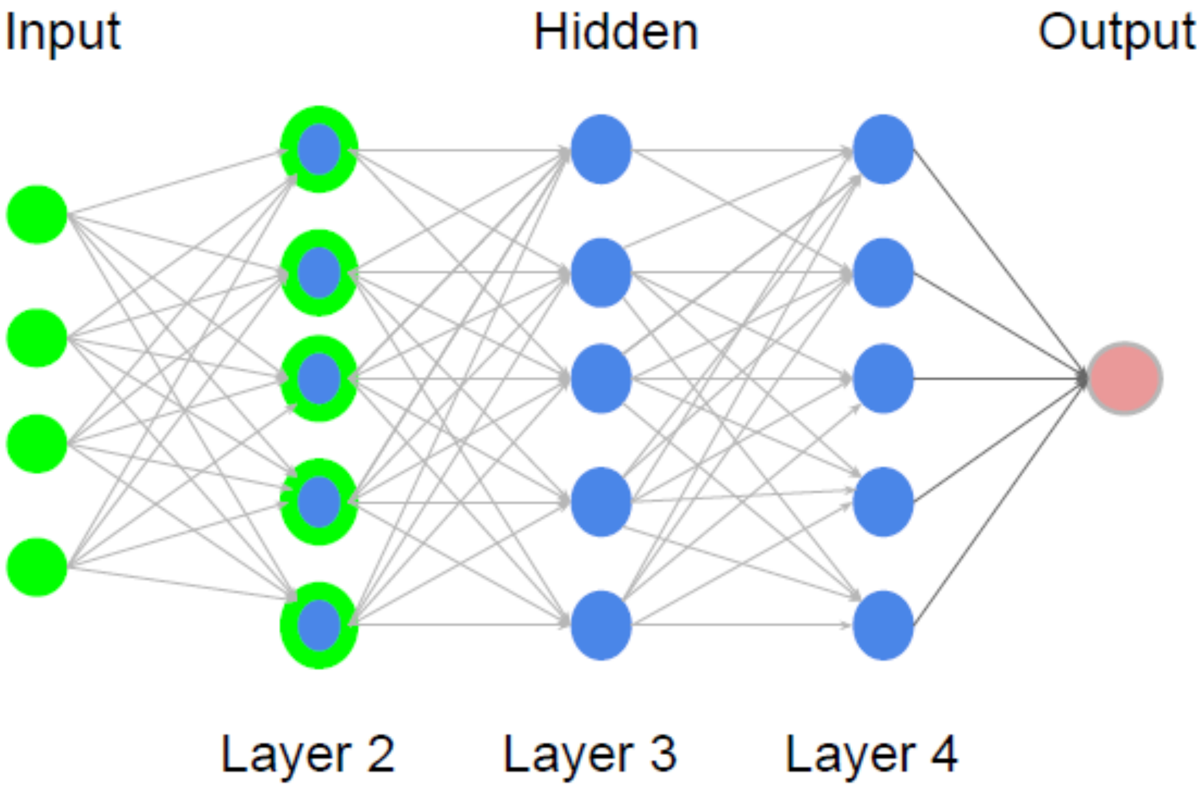


# Artificial Neural Networks

Machine learning algorithms

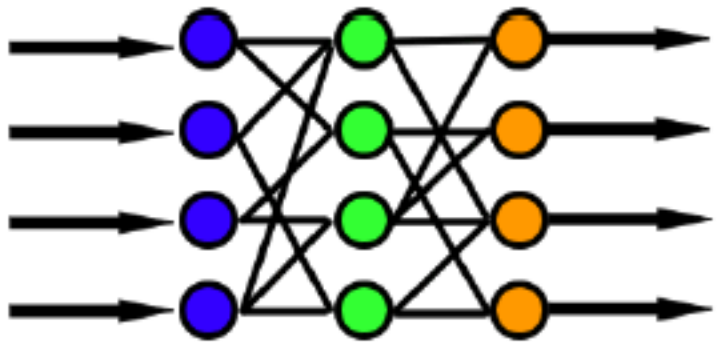
Learn and predict on data



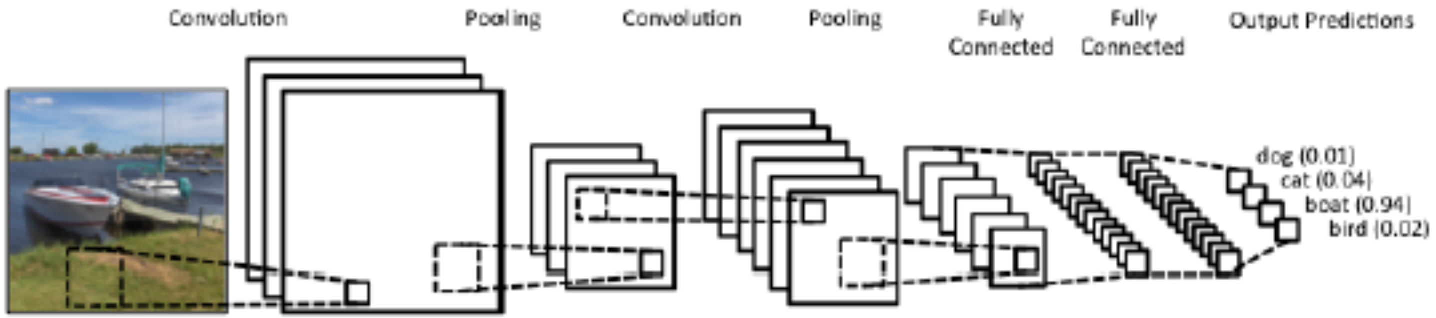


# Artificial Neural Networks

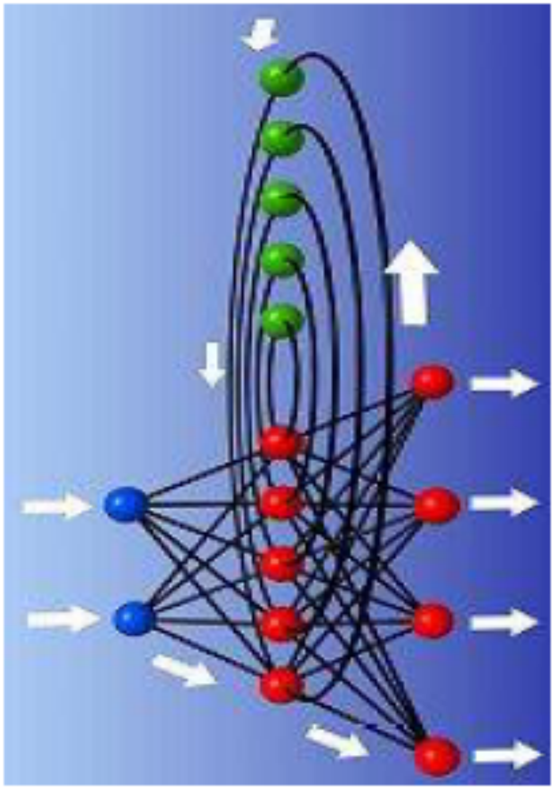
Feed Forward Nets (FNN)



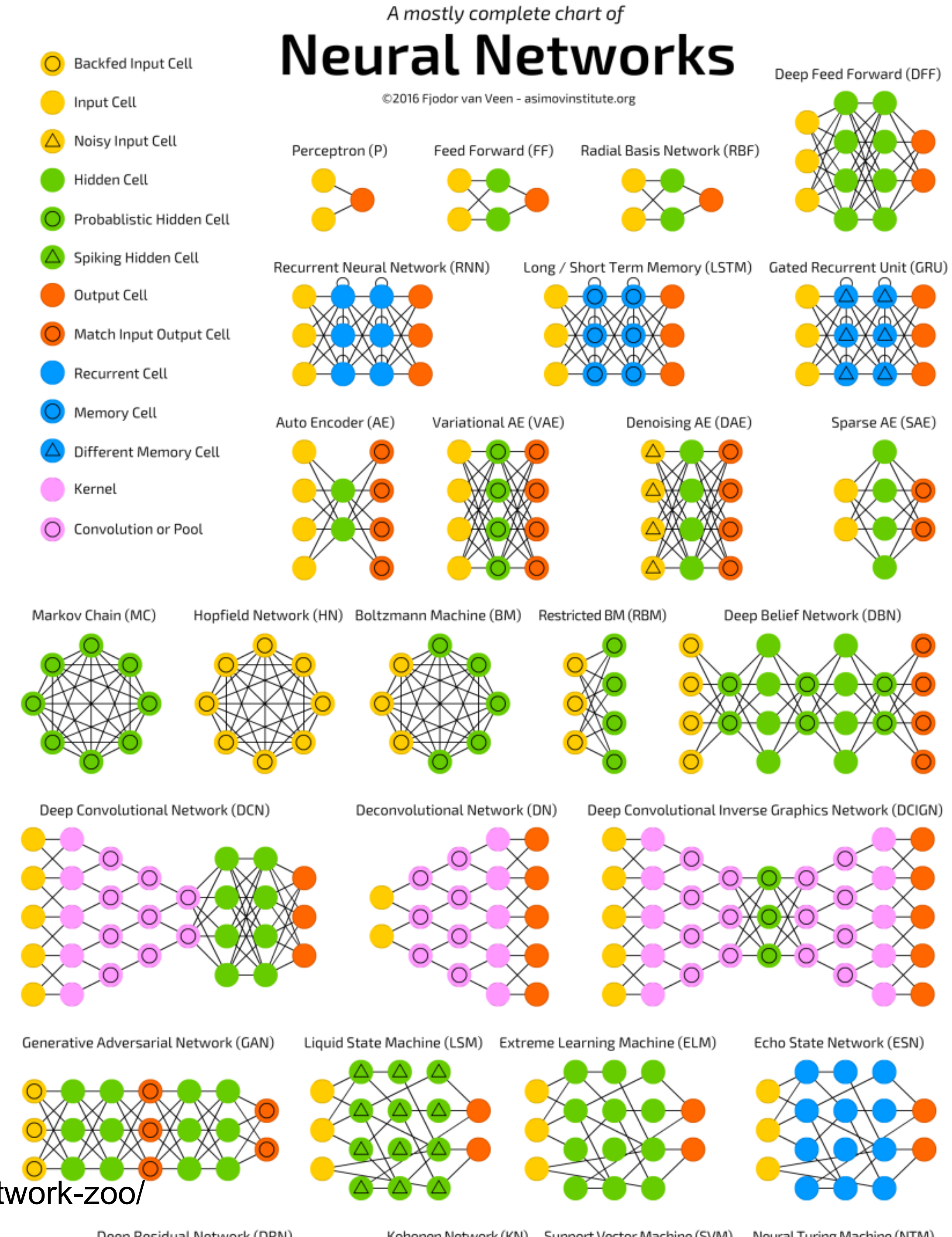
Convolutional Nets (CNN)




Recurrent Nets (RNN)



**... and others**

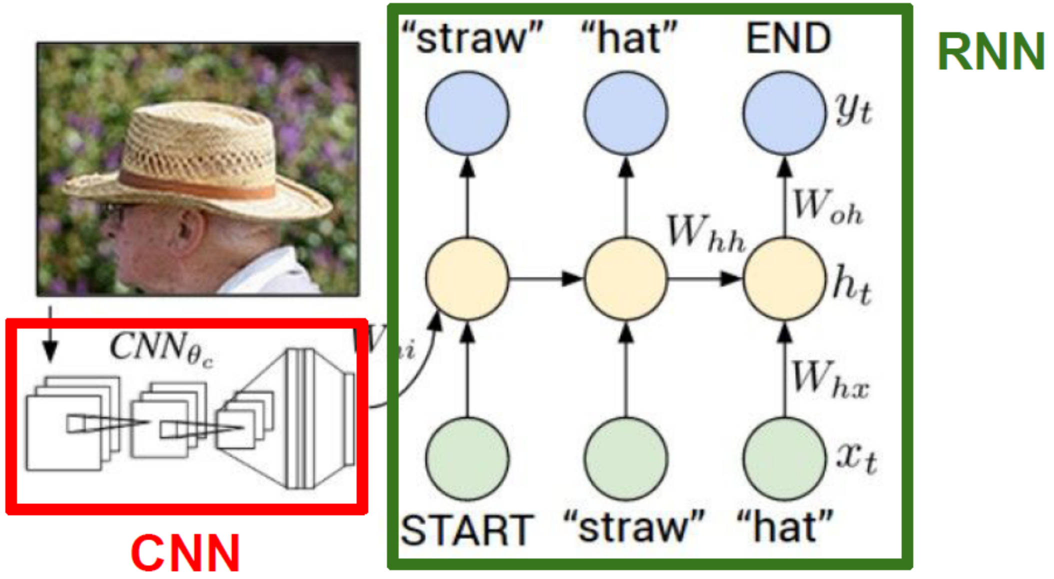



Source:
http://www.asimovinstitute.org/neural-network-zoo/

# Grandes avances del Deep Learning

DeepL


# NN can do more complex tasks, e.g., scene captioning





"two young girls are playing with lego toy."


"boy is doing backflip on wakeboard."


"young girl in pink shirt is swinging on swing."

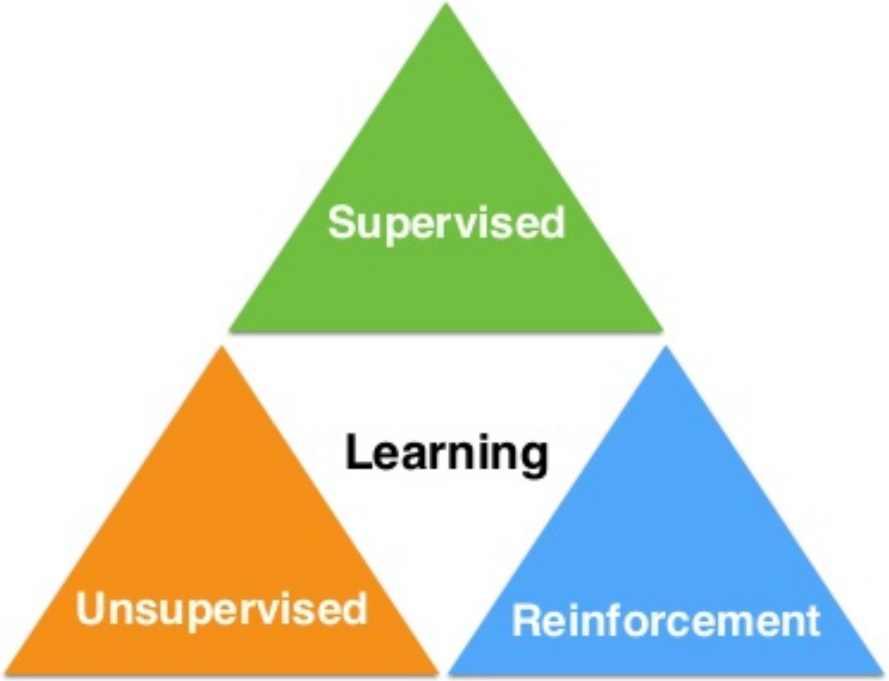

"man in blue wetsuit is surfing on wave."

# Outline

- Intro de Artificial Neural Networks(ANN)
- **How ANN Learn?**
- Linear Classifier
- Loss Function
- Gradient Descent
- Backpropagation
- Activation Functions

# Neural Networks Learning

- Labeled data
- Direct feedback
- Predict outcome/future




- No labels
- No feedback
- “Find hidden structure”

- Decision process
- Reward system
- Learn series of actions

# Image classification of MNIST

**Training stage**
(60.000 images)

0
1
2
3
4
5
6
7
8
9

→ Supervised Learning of the NN → Predictive Model

**Test stage**
(10.000 images)

→ Predictive Model → 3

# Outline

- Intro de Artificial Neural Networks(ANN)
- How ANN Learn?
- **Linear Classifier**
- Loss Function
- Gradient Descent
- Backpropagation
- Activation Functions
- The training of a simple ANN step by step

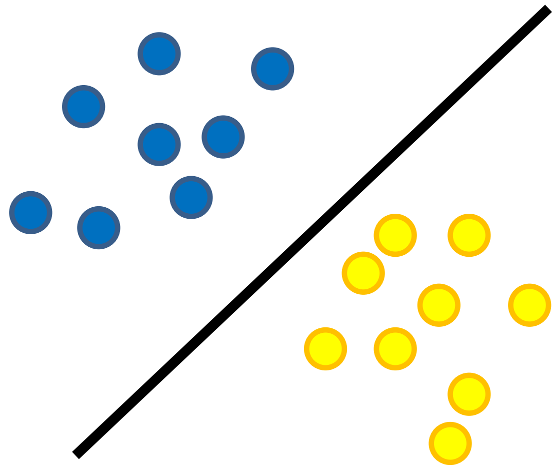
# Building Neural Network

Building block

NN developer

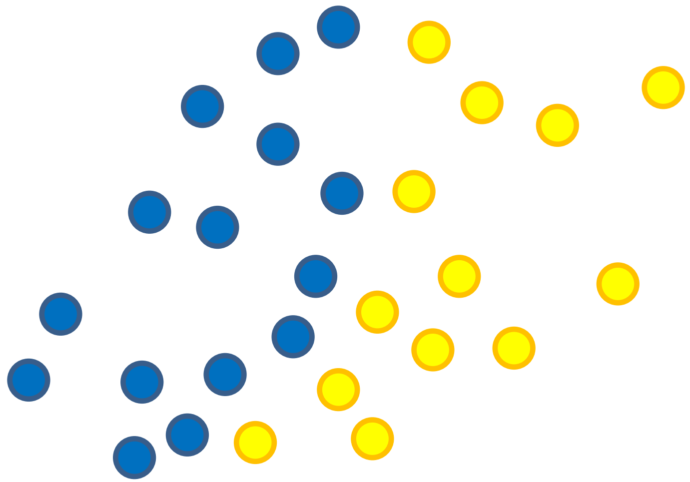
# Classifying with Neural Networks

Simple dataset → simple linear classifier



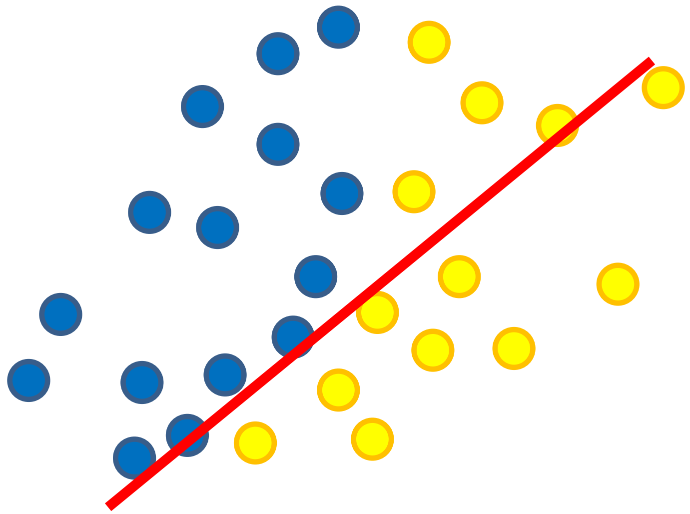
# Classifying with Neural Networks

Complex dataset → ?



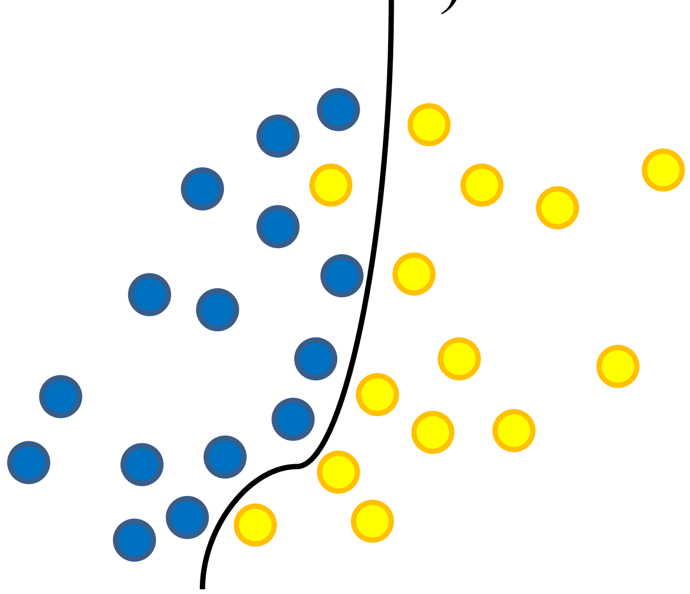
# Classifying with Neural Networks

Complex data → ?



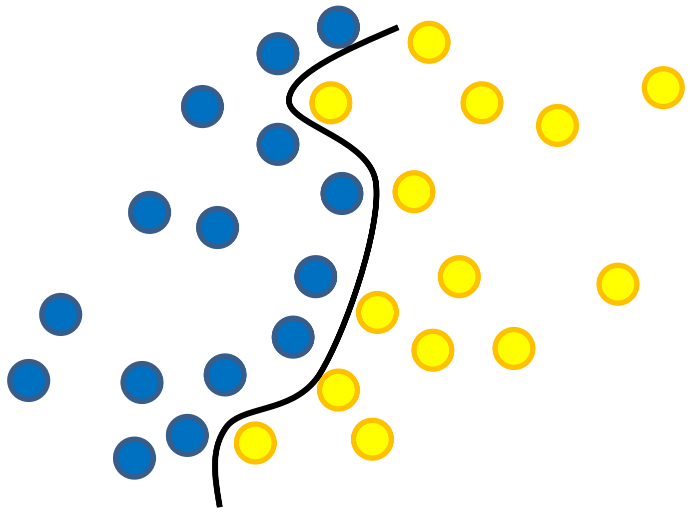
# Classifying with Neural Networks

Complex data→ Non-linear classifier = a sequence of one or more (linear classifier+activation functions)



# Classifying with Neural Networks

Complex data→ Non-linear classifier = linear classifier+activation functions



# The math behind

**Learning step**

- Given N examples ($x_i$ , $y_i$) , where $x_i$ is the image and $y_i$ the corresponding label find function **F()** such as:

$$F(W,x,b)=y$$

**Test step**

- How well is the prediction on a new set

28x28 pixels

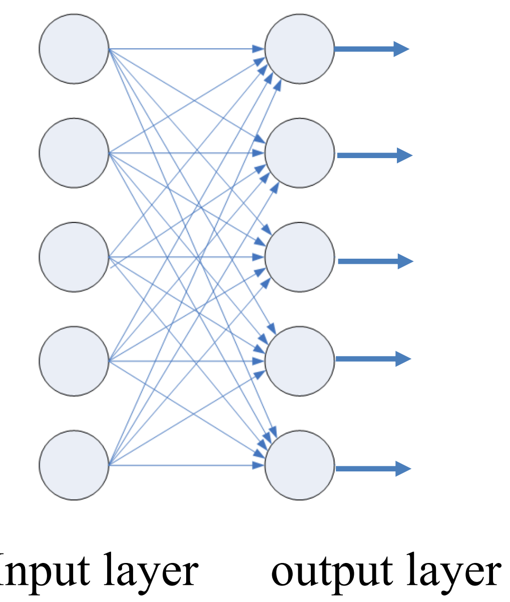
→ **F(W,x,b)** → **3**

# Simple Linear classifier

weights bias

$$F(W, x, b) = y$$

data



# How does a Linear Classifier learn?

For MNIST dataset

- Each image is 28x28=784 pixels
- 10 classes

Array of [28,28] pixels
(**784** numbers total)

image

$$F(x, W) = W x + b = out$$

W: 10x784 — parameters

x: 784

out: 10 — 10 numbers indicating class score

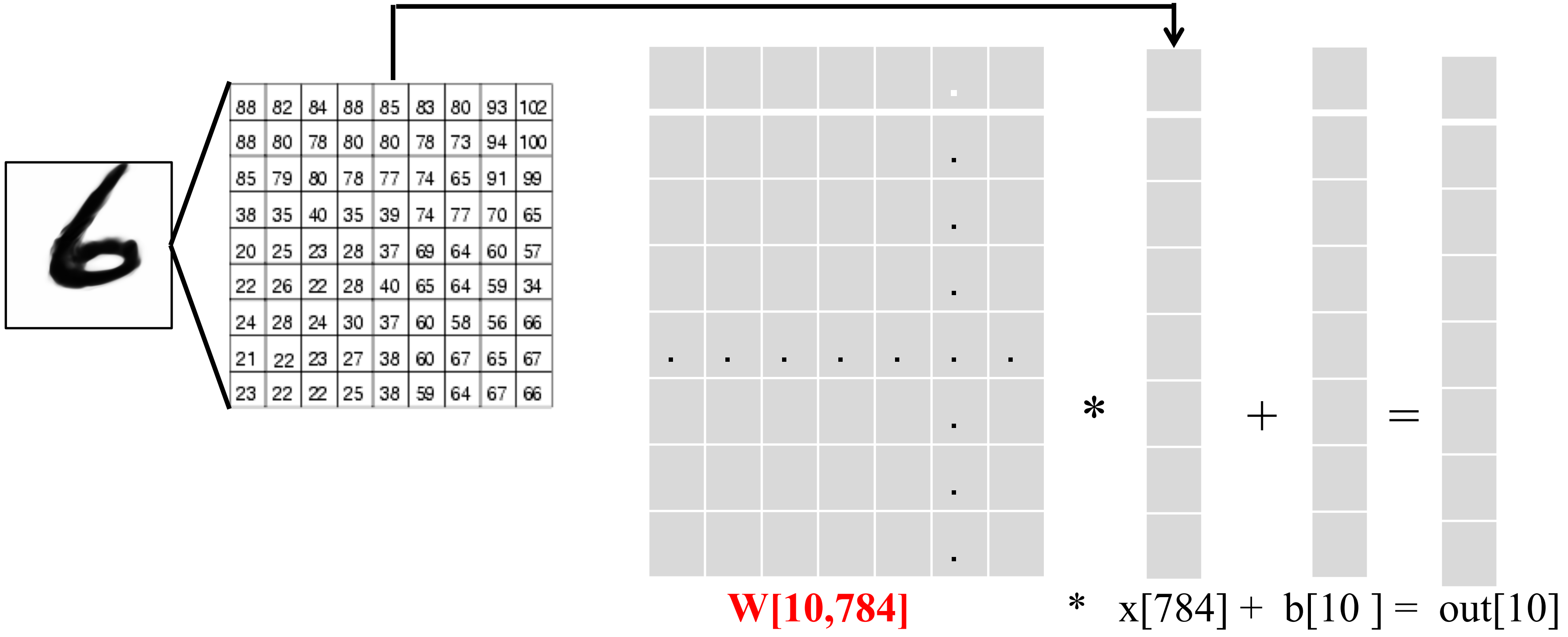
An optimization problem:

# Outline

- Intro de Artificial Neural Networks(ANN)
- How ANN Learn?
- Linear Classifier
- **Loss Function**
- Gradient Descent
- Backpropagation
- Activation Functions
- The training of a simple ANN step by step

# How does a Linear Classifier learn?

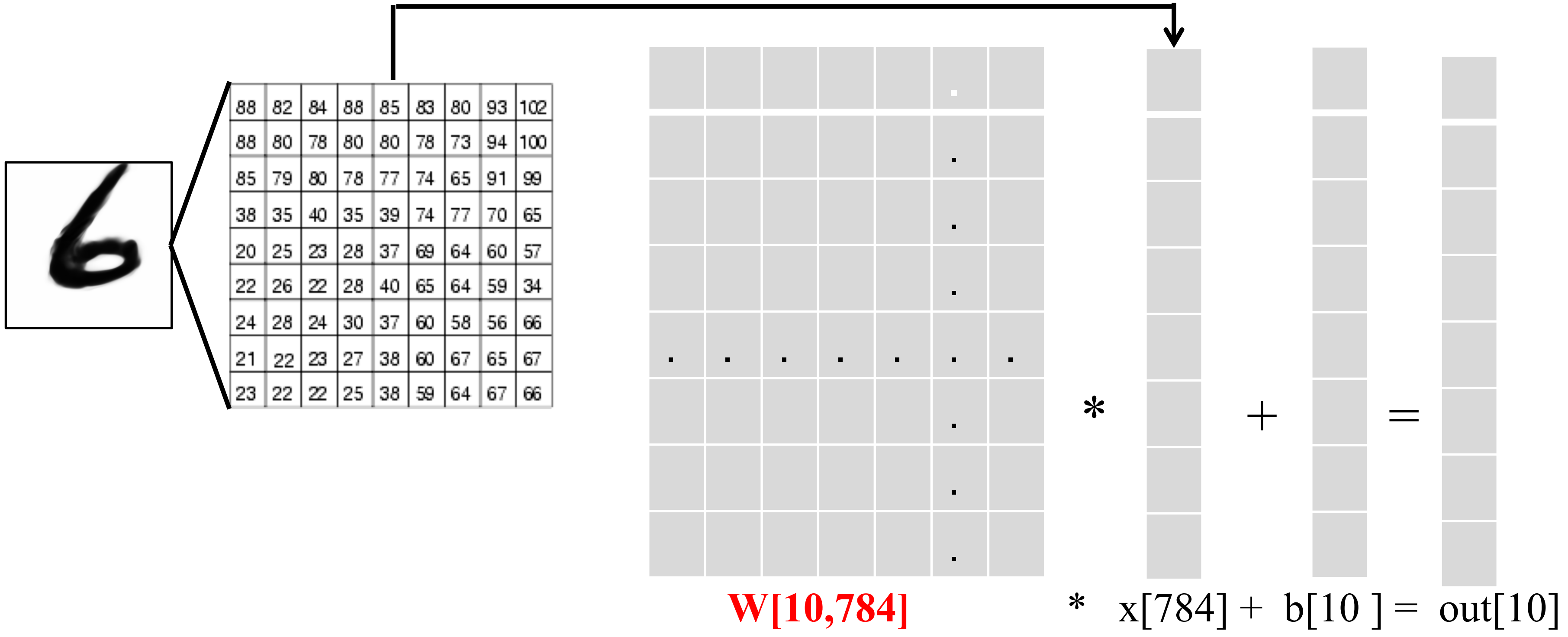
$$F(x, W) = \mathbf{W}\, x + b = out$$




- $W_0$ = random
- Define a loss function (cost function)
  **Loss=(Predicted-desired)**

# How does a Linear Classifier learn?

$$F(x, W) = \mathbf{W}\,x + b = out$$




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 88 | 82 | 84 | 88 | 85 | 83 | 80 | 93 | 102 |
| 88 | 80 | 78 | 80 | 80 | 78 | 73 | 94 | 100 |
| 85 | 79 | 80 | 78 | 77 | 74 | 65 | 91 | 99 |
| 38 | 35 | 40 | 35 | 39 | 74 | 77 | 70 | 65 |
| 20 | 25 | 23 | 28 | 37 | 69 | 64 | 60 | 57 |
| 22 | 26 | 22 | 28 | 40 | 65 | 64 | 59 | 34 |
| 24 | 28 | 24 | 30 | 37 | 60 | 58 | 56 | 66 |
| 21 | 22 | 23 | 27 | 38 | 60 | 67 | 65 | 67 |
| 23 | 22 | 22 | 25 | 38 | 59 | 64 | 67 | 66 |

- $W_0$ = random
- Define a loss function (cost function)

  **Loss=(Predicted-desired)**

**How good is W?**

# Loss Function

Quantifies how good is our W($w_i$,…,$w_N$) in each iteration:

$$crossentropy = -\frac{1}{N}(\sum_{i=1}^{N} y_i * \log(out_i) + (1 - yi) * \log(1 - outi))$$

where $out_i = softmax(outi)$

# Outline

- Intro de Artificial Neural Networks(ANN)
- How ANN Learn?
- Linear Classifier
- Loss Function
- **Gradient Descent**
- Backpropagation
- Activation Functions

# How to find the best W?

## Gradient descent

# How to find the best W?

## Gradient descent

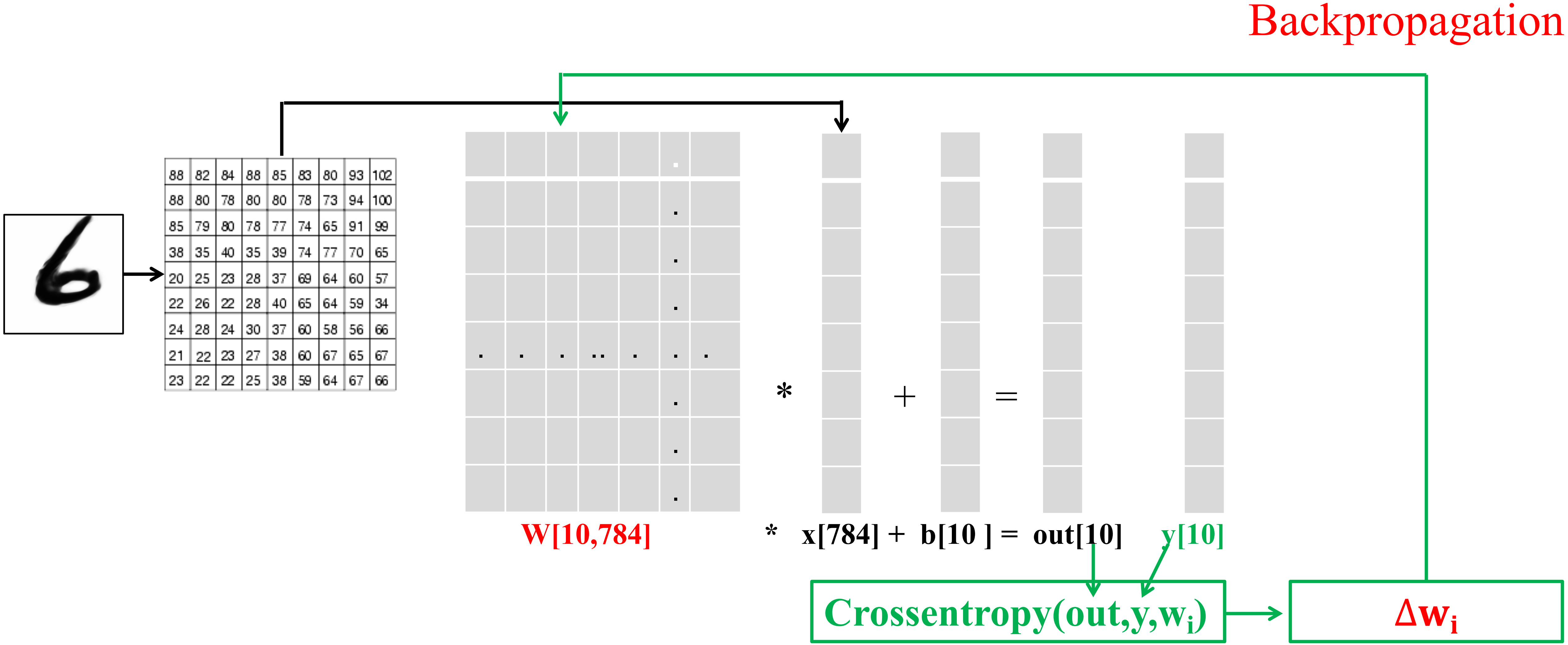
- How to detect the direction that minimize the error?
- The slop. In 1-dimension the derivative is

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{dh}$$

# Outline

- Intro de Artificial Neural Networks(ANN)
- How ANN Learn?
- Linear Classifier
- Loss Function
- Gradient Descent
- **Backpropagation**
- Activation Functions
- The training of a simple ANN step by step

# How does a Linear classifier learn?

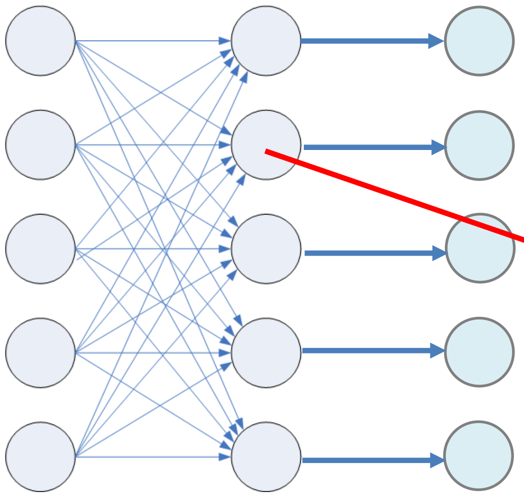


# Outline

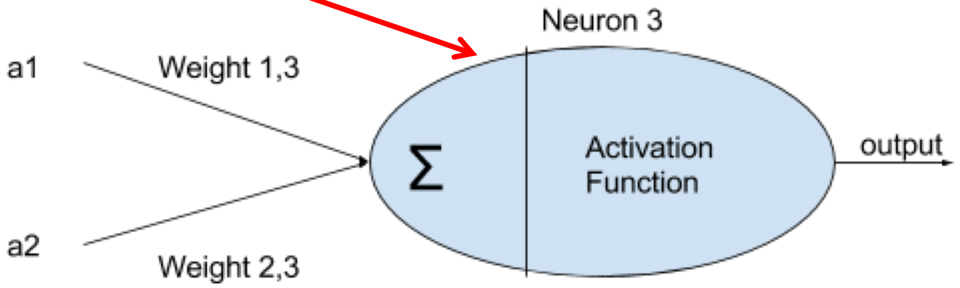
- Intro de Artificial Neural Networks(ANN)
- How ANN Learn?
- Linear Classifier
- Loss Function
- Gradient Descent
- Backpropagation
- **Activation Functions**
- The training of a simple ANN step by step

# Activation functions

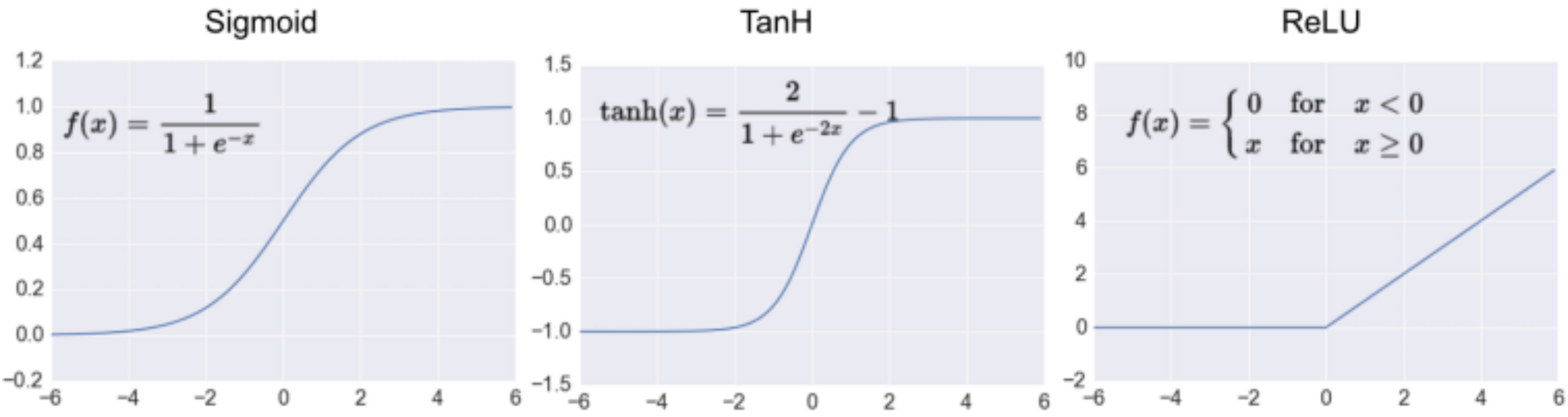
The linear classifier $W x + b = out$



$x W + b = out$

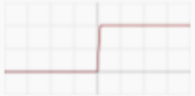
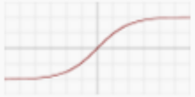
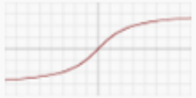
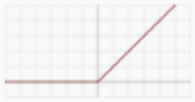
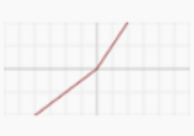
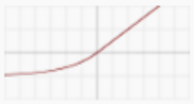
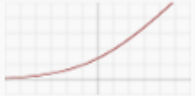


# Activation functions



# Activation functions

These function are easy to derive

| Name | Plot | Equation | Derivative |
|---|---|---|---|
| Identity |  | $f(x) = x$ | $f'(x) = 1$ |
| Binary step |  | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$ |
| Logistic (a.k.a Soft step) |  | $f(x) = \frac{1}{1+e^{-x}}$ | $f'(x) = f(x)(1 - f(x))$ |
| TanH |  | $f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$ | $f'(x) = 1 - f(x)^2$ |
| ArcTan |  | $f(x) = \tan^{-1}(x)$ | $f'(x) = \frac{1}{x^2+1}$ |
| Rectified Linear Unit (ReLU) |  | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| Parameteric Rectified Linear Unit (PReLU) [2] |  | $f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| Exponential Linear Unit (ELU) [3] |  | $f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| SoftPlus |  | $f(x) = \log_e(1+e^x)$ | $f'(x) = \frac{1}{1+e^{-x}}$ |

Source: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

# Recapitulate

Training iteration of a neural network with a specific architecture is two steps:

Forward Propagation

**Initial Weights** → **Training** → **Prediction** → **Error**

Backward Propagation

**Error** → **Calculate gradients** → **Update weights**

# References


- Prerequisite: https://docs.python.org/3/tutorial/
- Textbook "Deep Learning". Ian Goodfellow, Yoshua Bengio, and Aaron Courville". Free Link.
- "Deep Learning with Tensorflow" Free link:

https://drive.google.com/file/d/1M0lSk7zeC_KoqwHUnpKqvhppkY5AlWUo/view?usp=sharing