Deep Learning

Siham Tabik

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

DaSCI

UNIVERSIDAD DE GRANADA

Schedule

26/02/2025 15:30-18:00	Fundamentos de redes neuronales	Ejemplos de NN con `Pytorch`
27/02/2025 18:00-20:30	Redes neuronales convolucionales: CNN	Ejemplo de CNNs con `Pytorch`
05/03/2025 15:30-18:00	Self-supevised Learning, Few-shot learning	SimCLR con `Pytorch`

Deep Learning: Fundamentos de redes neuronales

Siham Tabik

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

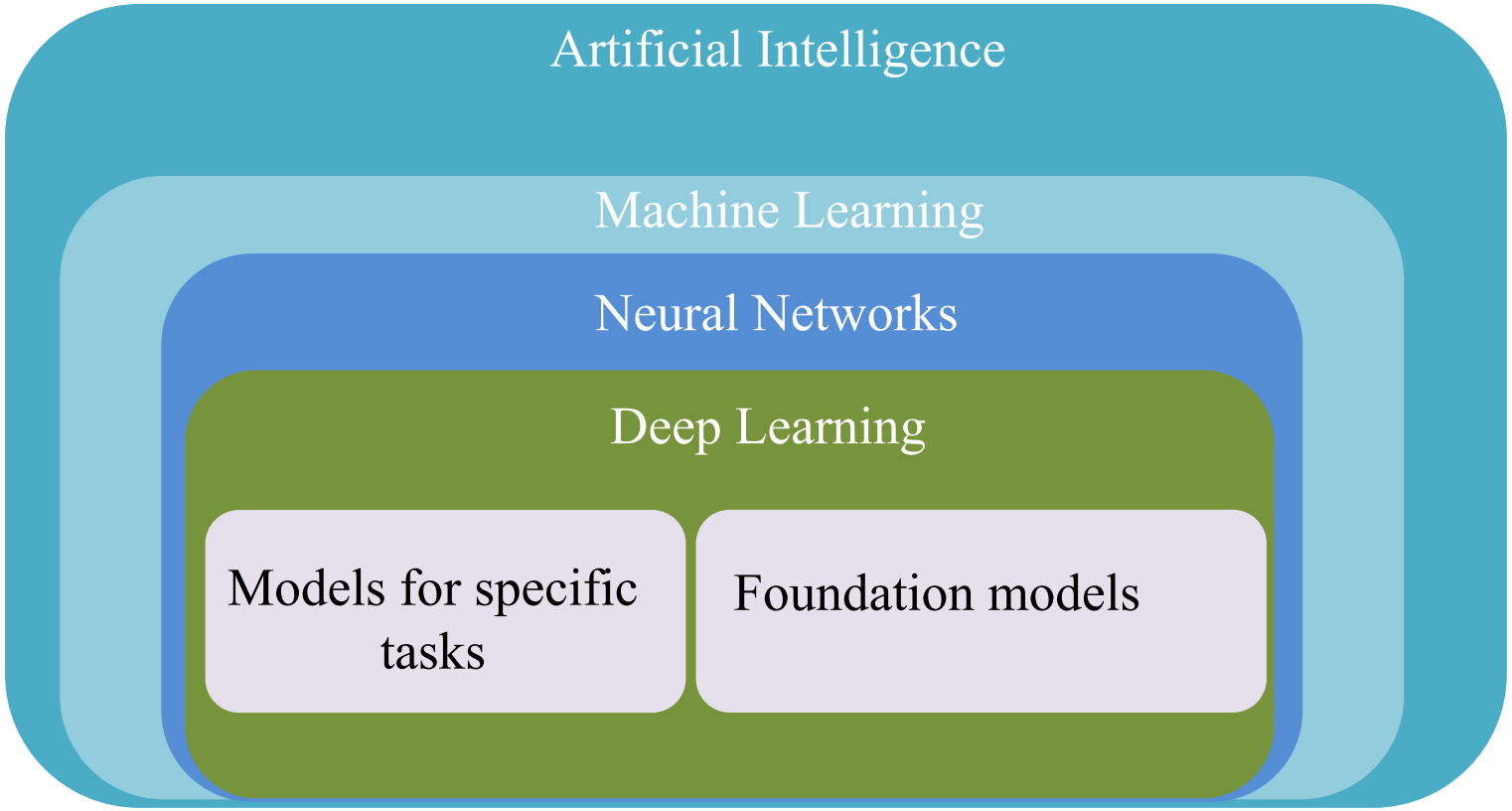
DaSCI

UNIVERSIDAD DE GRANADA

Outline

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

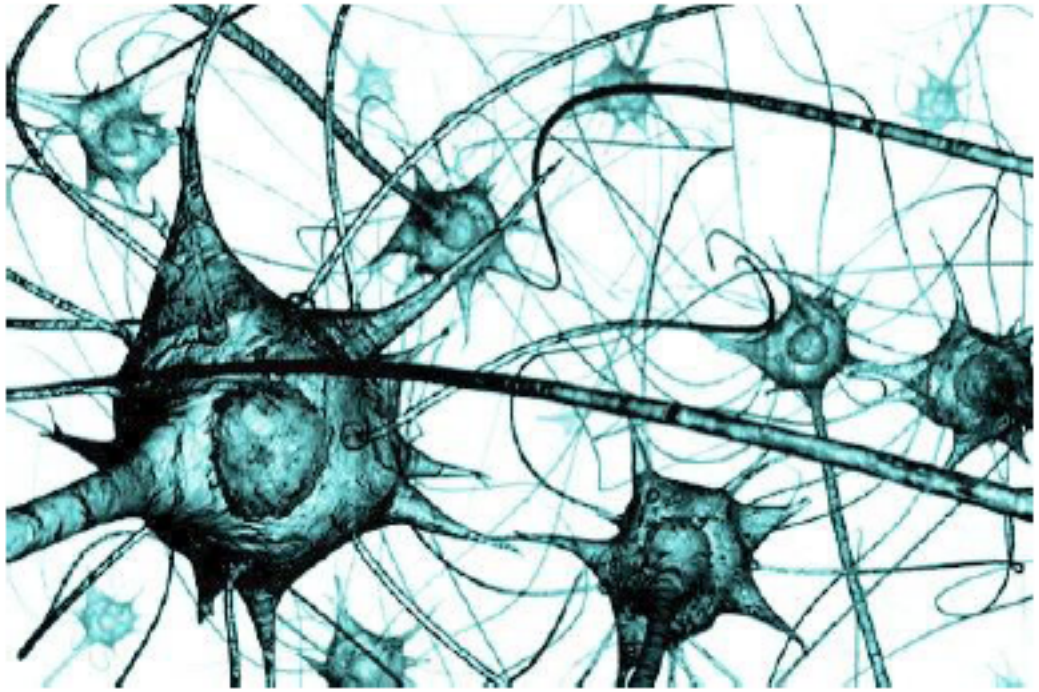
Artificial Neural Networks

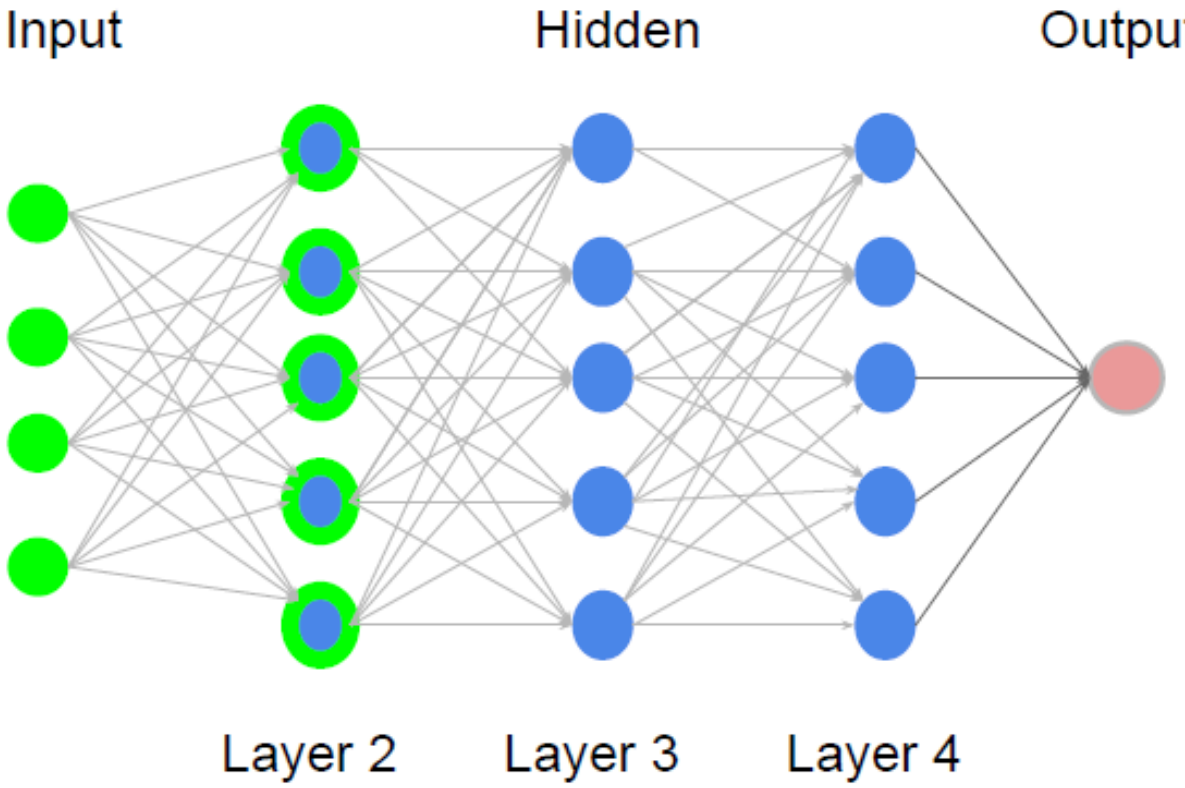


Artificial Neural Networks

Machine learning algorithms

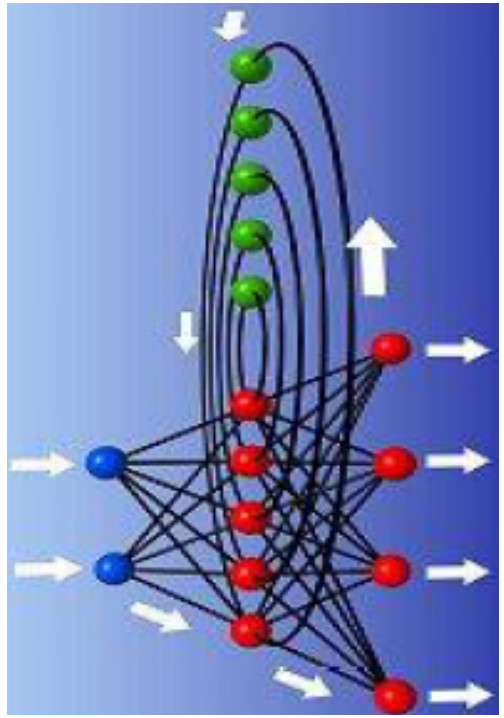
Learn and predict on data



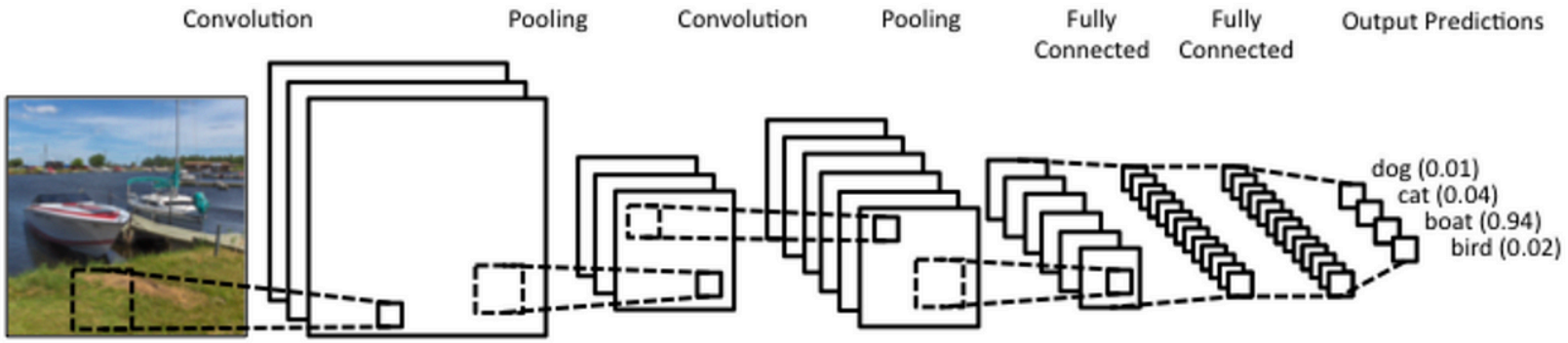


Specialized Artificial Neural Networks

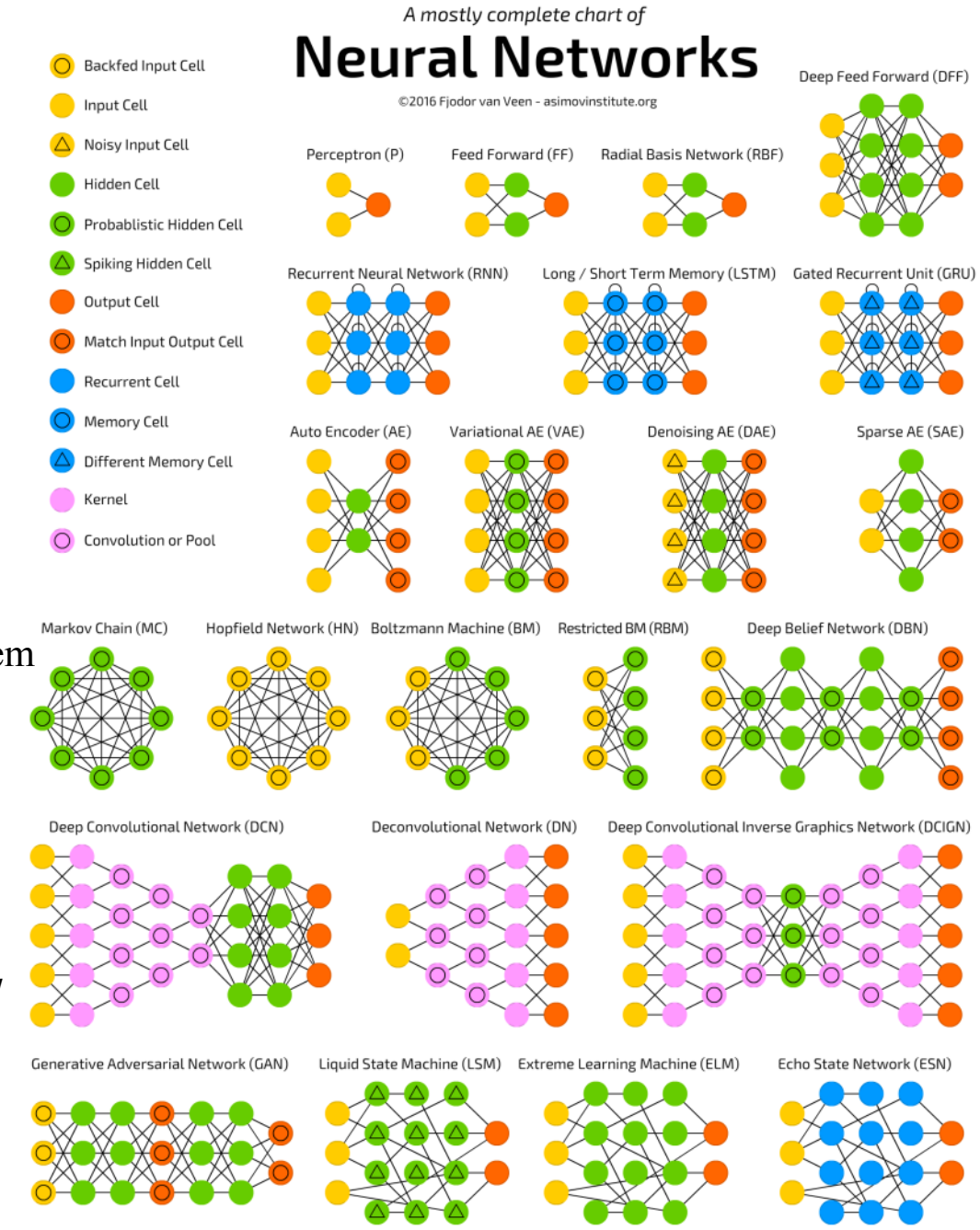
Recurrent Neural Network



Convolutional Neural Network

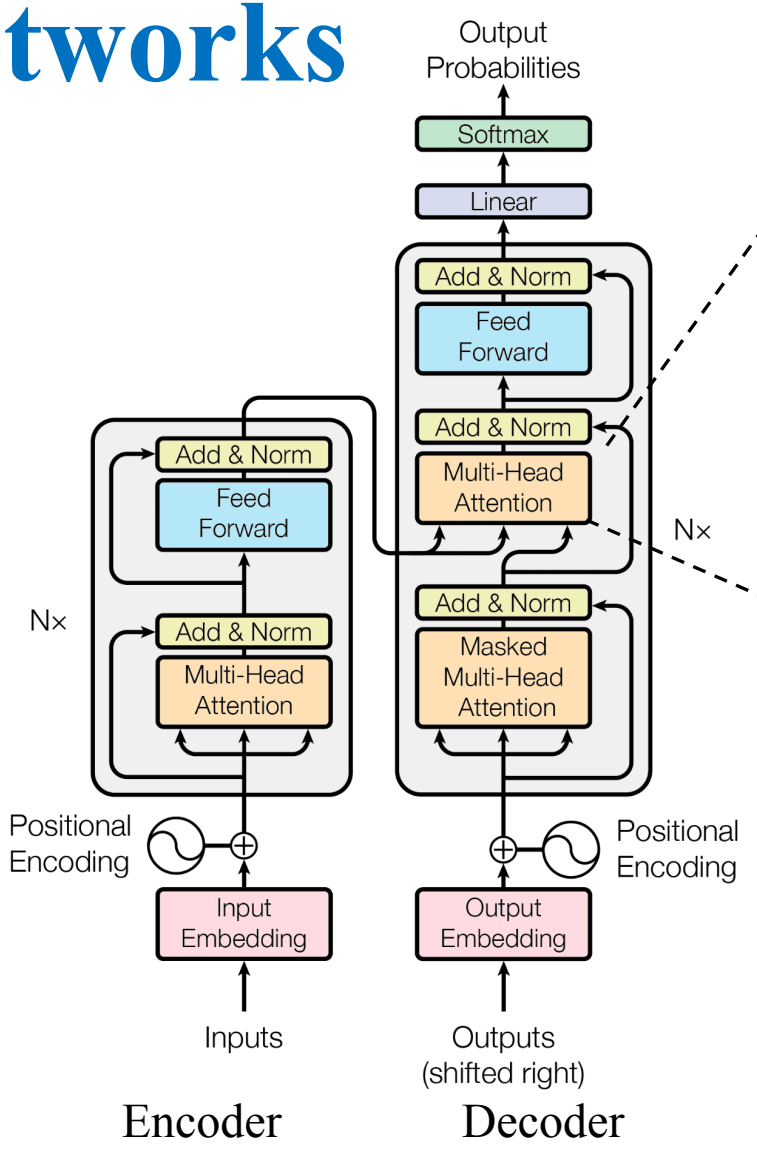


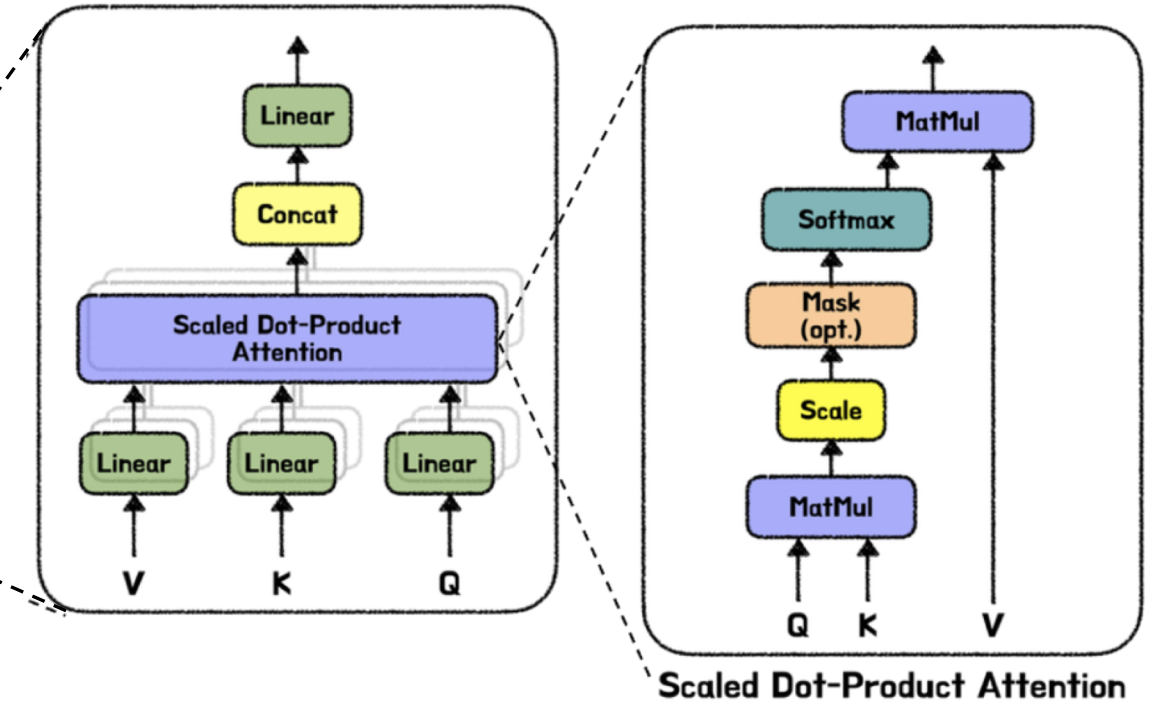
- This are few examples of NN architectures
  - Different nodes
  - Different topology
- We are still learning based on trial and error which architectures are good for each problem



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

Towards a general Artificial Neural Networks



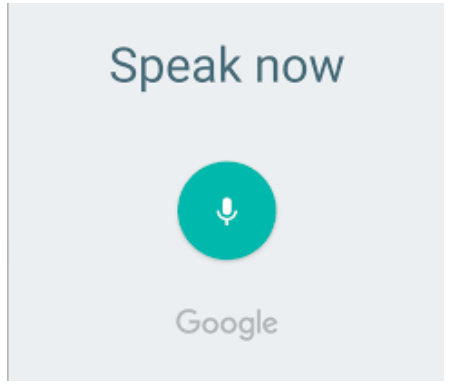


Transformer Network

Google Brain, "Attention is all you need". *NIPS 2017*

Used in BERT, GPT

Deep Learning in specific tasks



DeepL

Outline

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

Neural Networks Learning

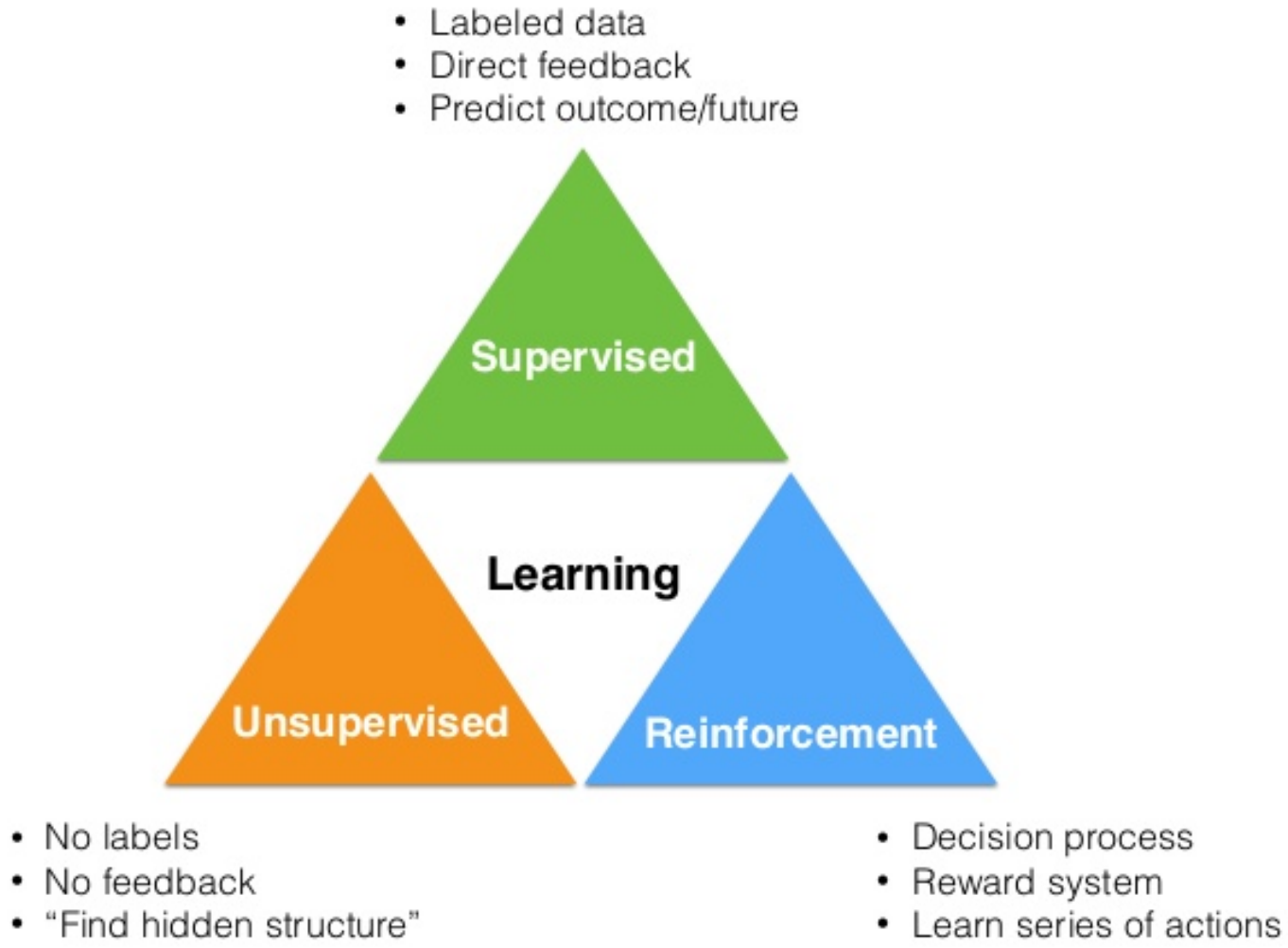


Image classification of MNIST

Training stage
(60.000 images)

Supervised Learning of the NN

Predictive Model

Test stage
(10.000 images)

Predictive Model

3

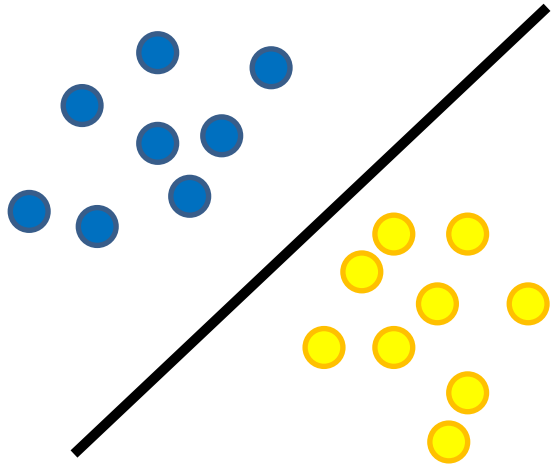
Building Neural Network

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

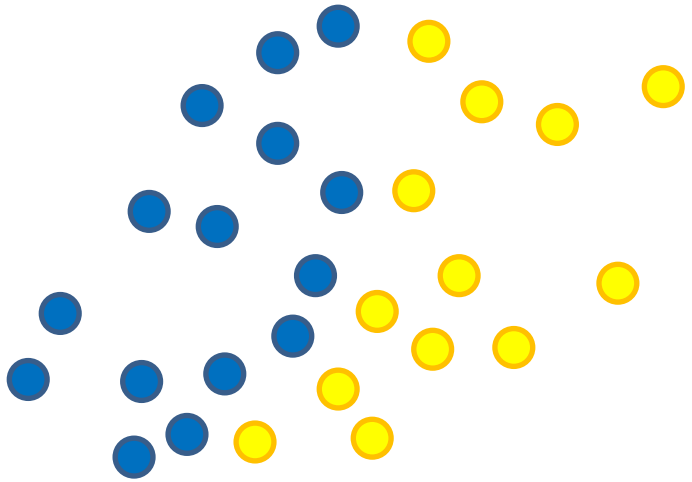
Classifying with Neural Networks

Simple dataset → simple linear classifier



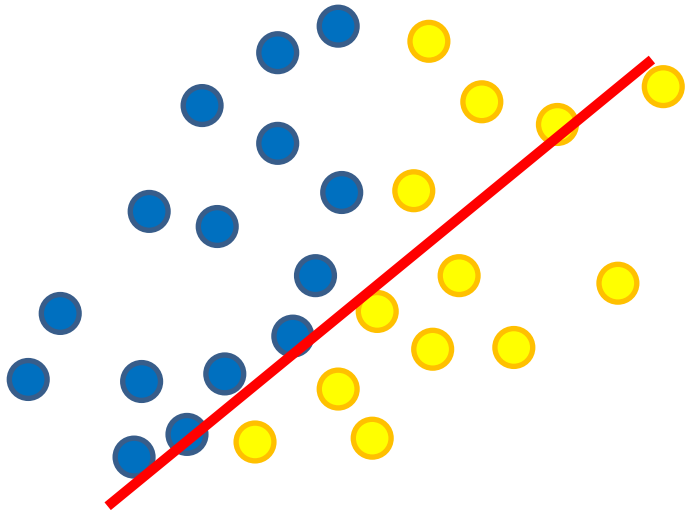
Classifying with Neural Networks

Complex dataset → ?



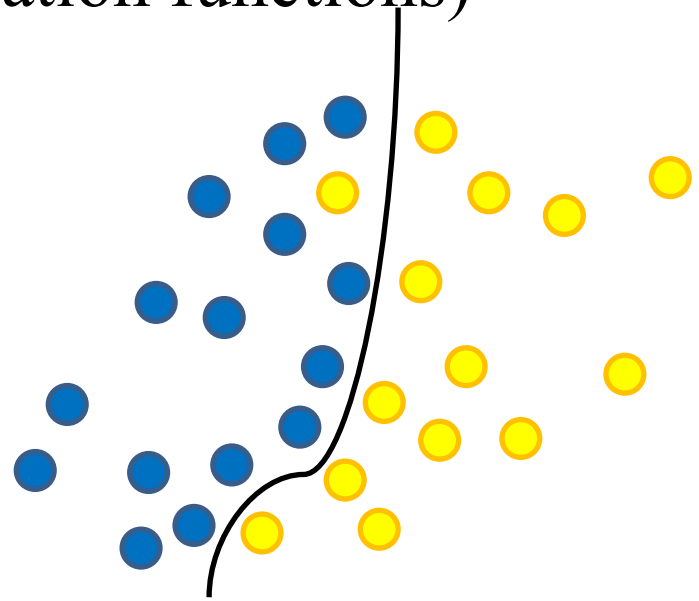
Classifying with Neural Networks

Complex data → ?



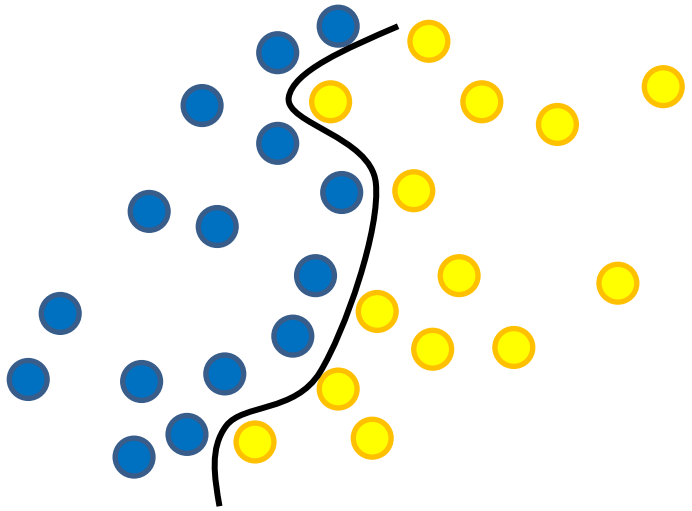
Classifying with Neural Networks

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:

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

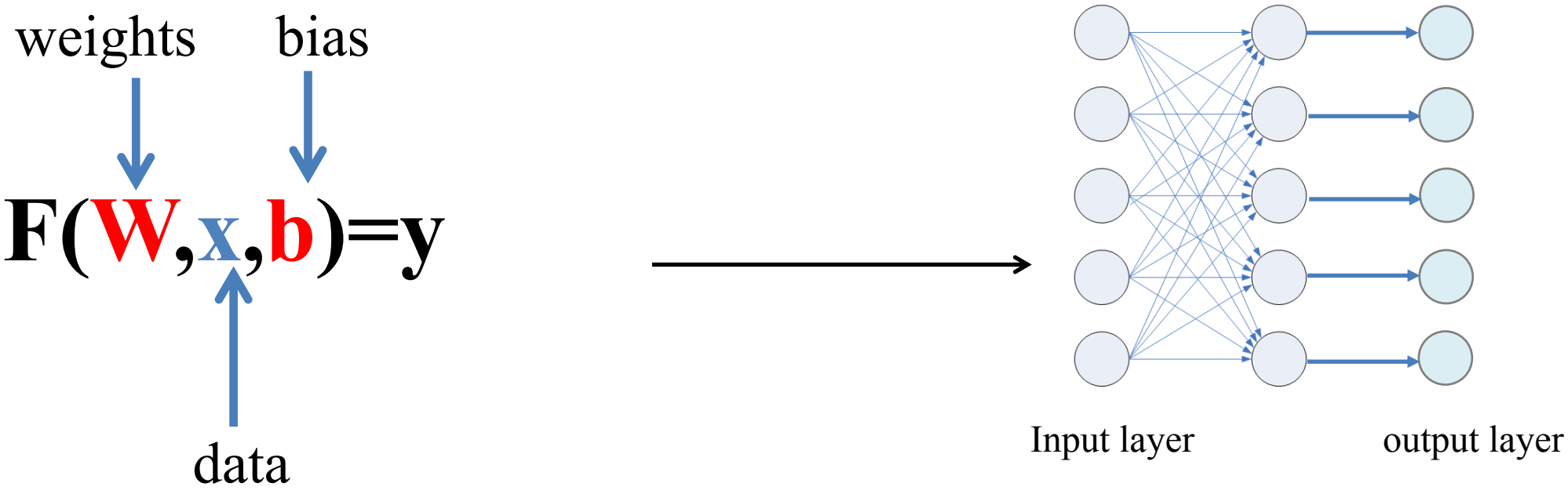
Test step

- How well is the prediction on a new set

28x28 pixels

F(W,x,b) → **3**

Simple Linear classifier

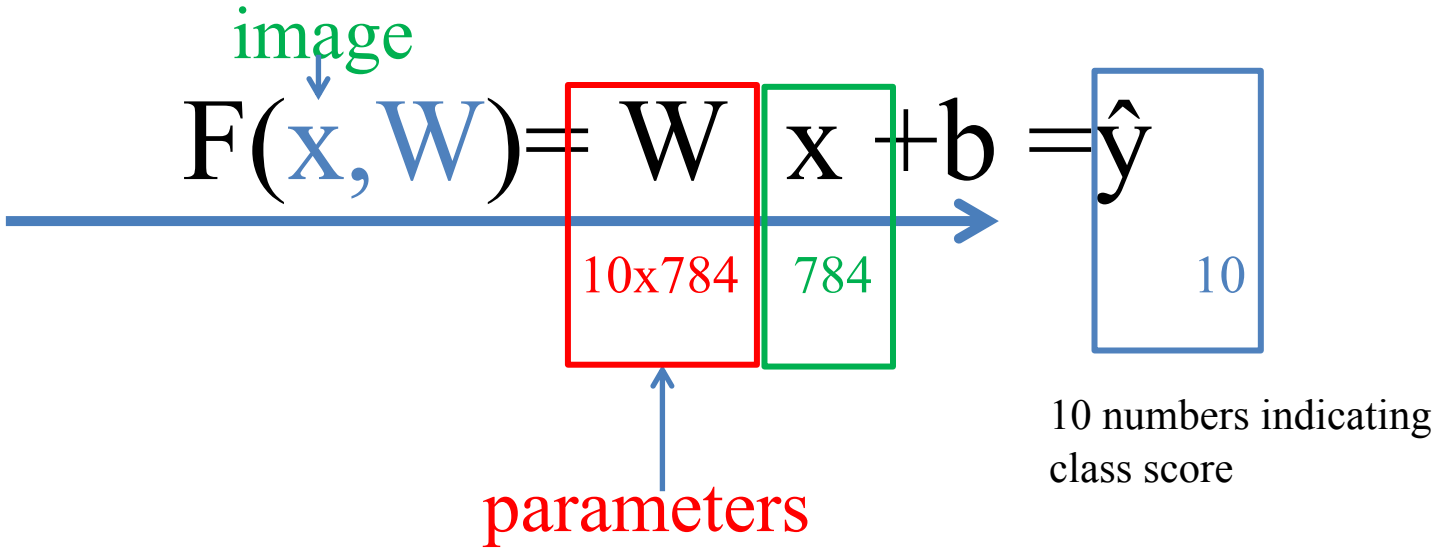


How does a Linear Classifier learn?

For MNIST dataset

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

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



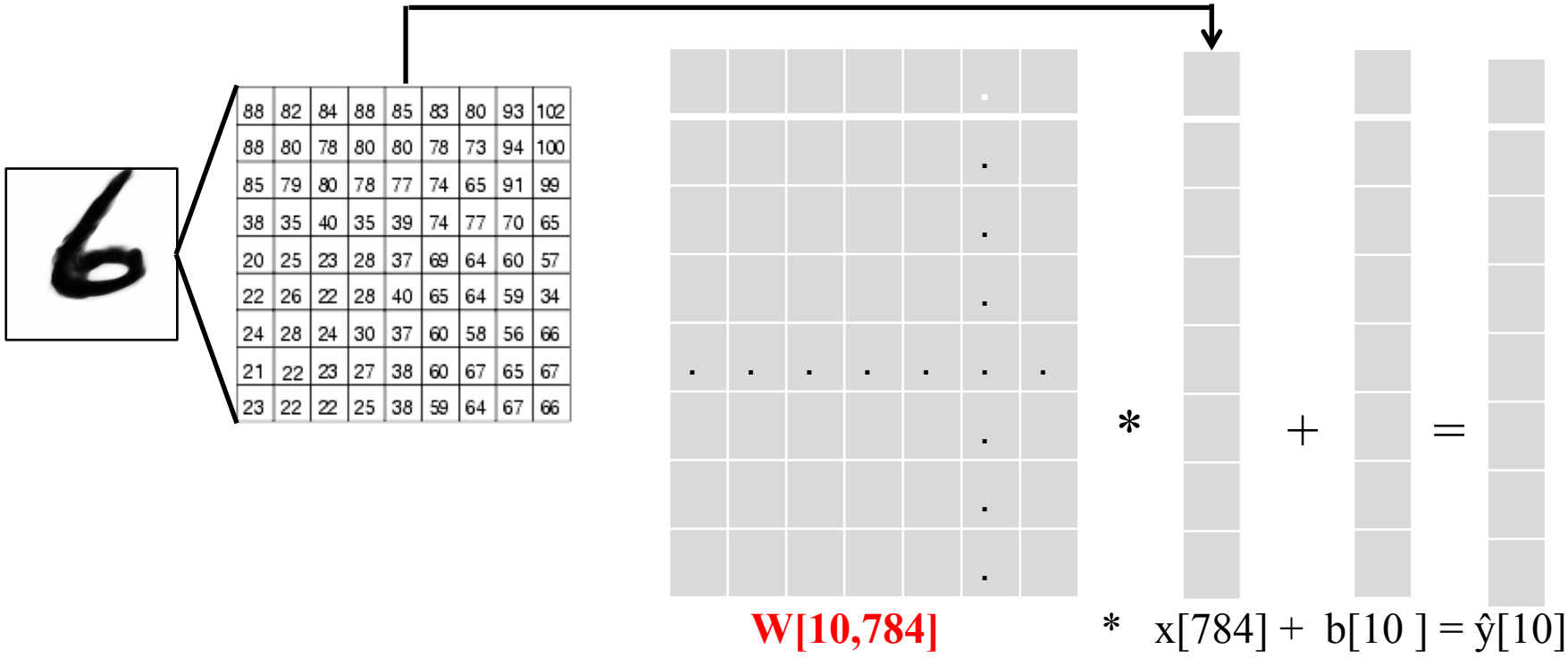
An optimization problem:

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?

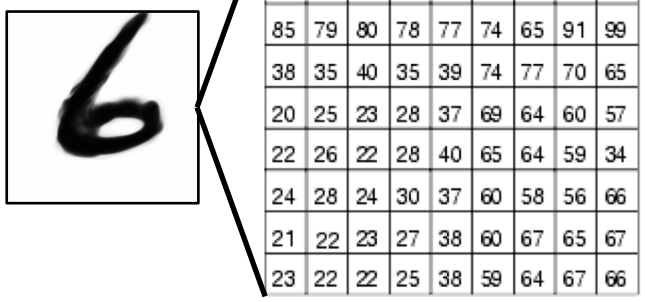
$$F(x,W) = \mathbf{W}\,x + b = \hat{y}$$



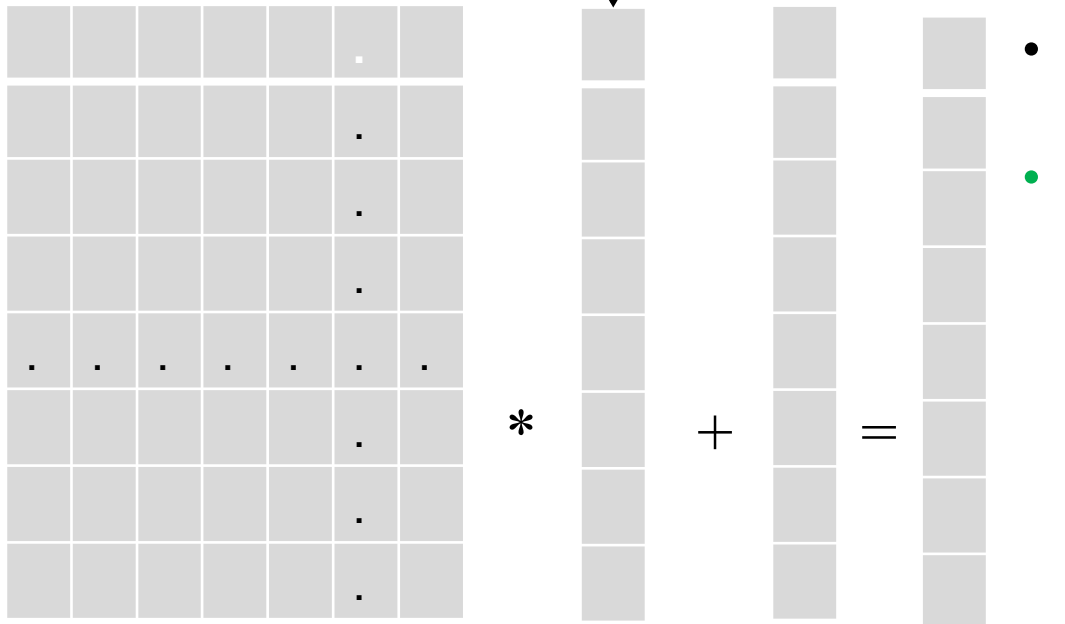
- W_0 = random
- Define a loss function (cost function)
  Loss=(Predicted-desired)

How does a Linear Classifier learn?

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



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[10,784] * x[784] + b[10] = out[10]

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

How good is W?

Loss Function

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

$$Loss = crossentropy = -\frac{1}{N}\left(\sum_{i=1}^{N} y_i * \log(\hat{y}_i) + (1 - yi) * \log(1 - \hat{y}i)\right)$$

Where $\hat{y}_i = softmax(\hat{y}i)$

Our aim is to minimize the Loss Function

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?

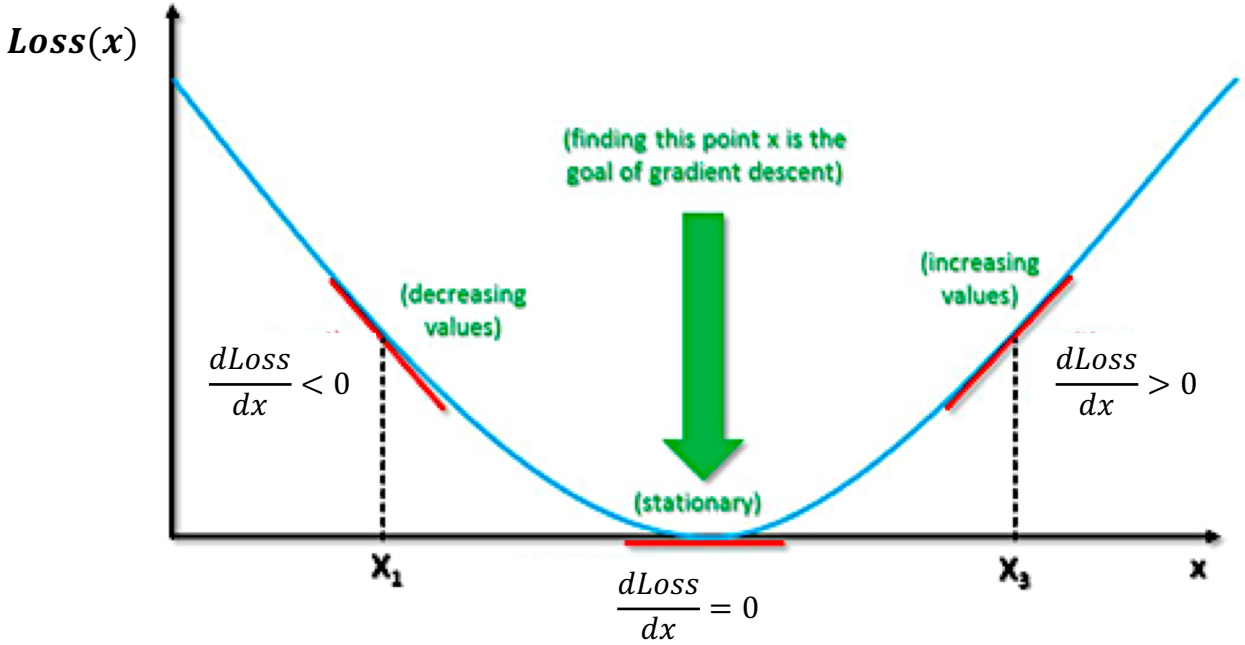
Gradient descent

How to find the best W?

Gradient descent

How to detect the direction that minimize the error?

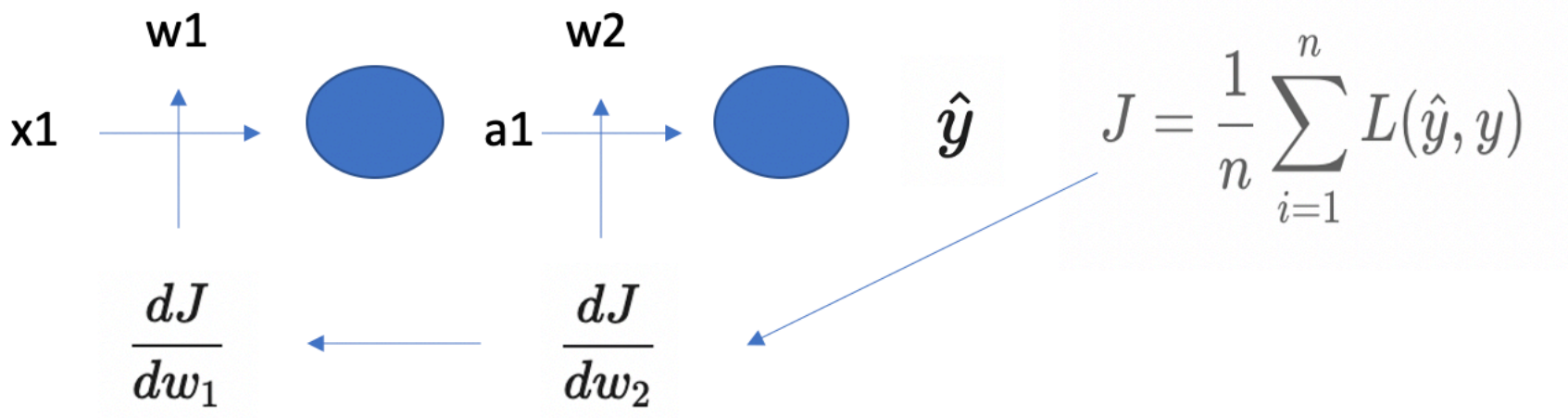
In 1-dimension the derivative $\frac{dloss(x)}{dx} = \lim_{h \to 0} \frac{loss(x+h) - loss(x)}{dh}$



How to find the best W?

Gradient descent

$$z_1 = w_1 x_1 \rightarrow a_1 = \sigma(z_1) \rightarrow z_2 = w_2 a_1 \rightarrow \hat{y} = \sigma(z_2) \rightarrow J = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}, y)$$

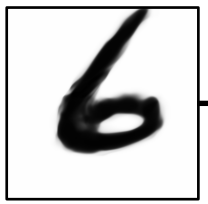


Outline

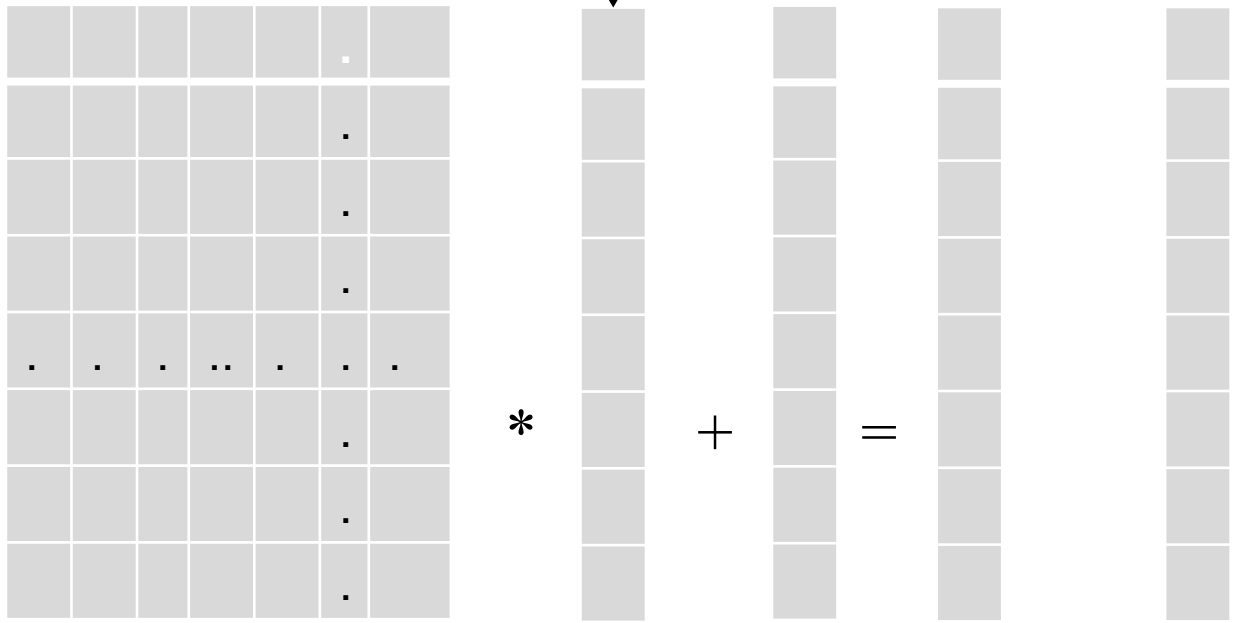
- Intro de Artificial Neural Networks(ANN)
- How ANN Learn?
- Linear Classifier
- Loss Function
- Gradient Descent
- **Backpropagation**
- Activation Functions
- The training of a simple ANN step by step

How does a Linear classifier learn?

Backpropagation



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[10,784] * **x[784] + b[10] =ŷ[10]** **y[10]**

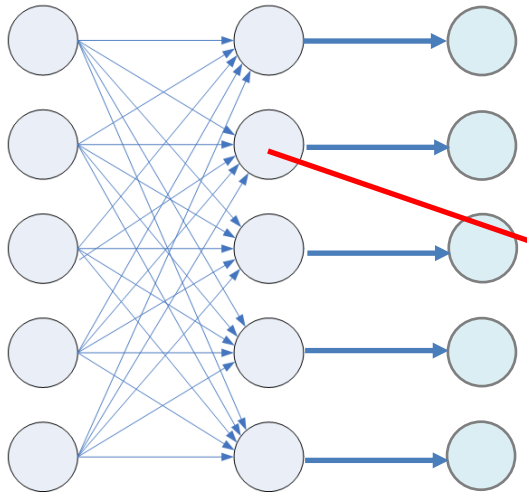
Crossentropy(ŷ,y,w_i) → GD → **Δw_i**

Outline

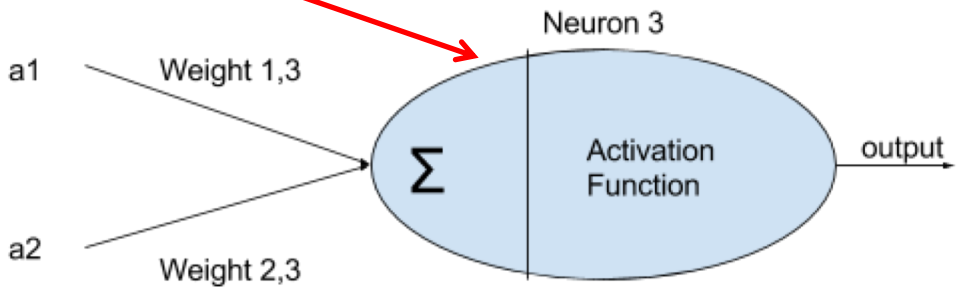
- Intro de Artificial Neural Networks(ANN)
- How ANN Learn?
- Linear Classifier
- Loss Function
- Gradient Descent
- Backpropagation
- **Activation Functions**
- The training of a simple ANN step by step

Activation functions

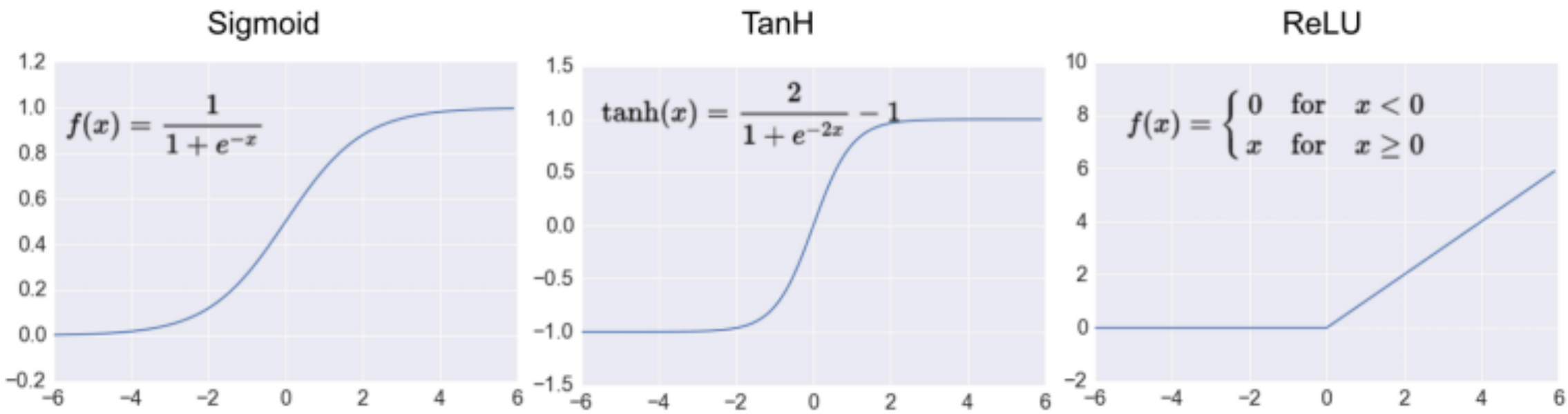
The linear classifier $W x + b = \hat{y}$



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

Backward Propagation

References

“Deep Learning Tuning Playbook” Deep Learning Tuning Playbook https://github.com/google-research/tuning_playbook/blob/main/README.md