Master 1 Statistique & Data Science, Ingénierie Mathématique

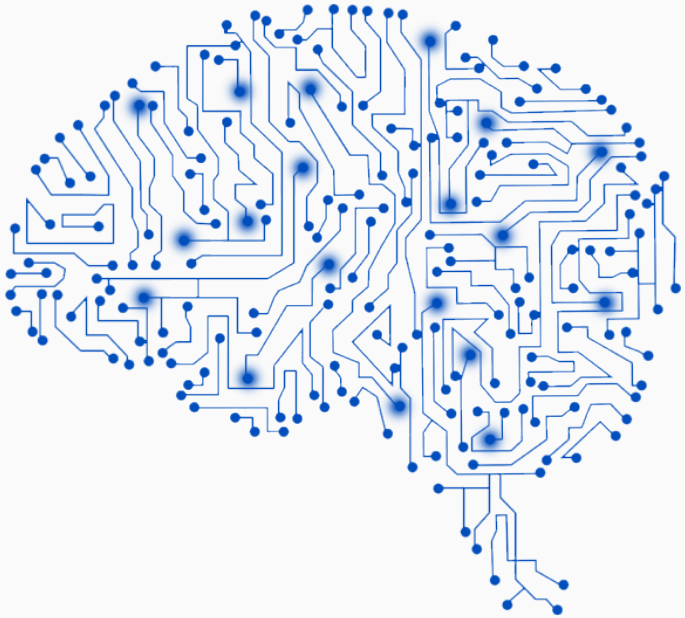
Apprentissage pour l'image
Machine learning for image processing

Course I – Introduction to Artificial Neural Networks: Multiclass logistic classification

---

Emile Pierret

Jeudi 20 mars 2025



Credits

Most of the slides from **Charles Deledalle's** course "UCSD ECE285 Machine learning for image processing" (30 $\times$ 50 minutes course)

`www.charles-deledalle.fr/`

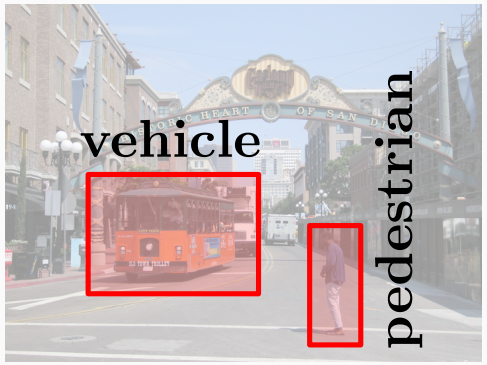
`https://www.charles-deledalle.fr/pages/teaching.php#learning`

Programme du cours d'aujourd'hui

1. Introduction aux capacités des réseaux de neurones
2. Introductions aux "datasets"
3. Introduction aux réseaux de neurones
4. Introduction aux coûts
5. Etude de la regression logistique

Computer Vision and Machine Learning

Image

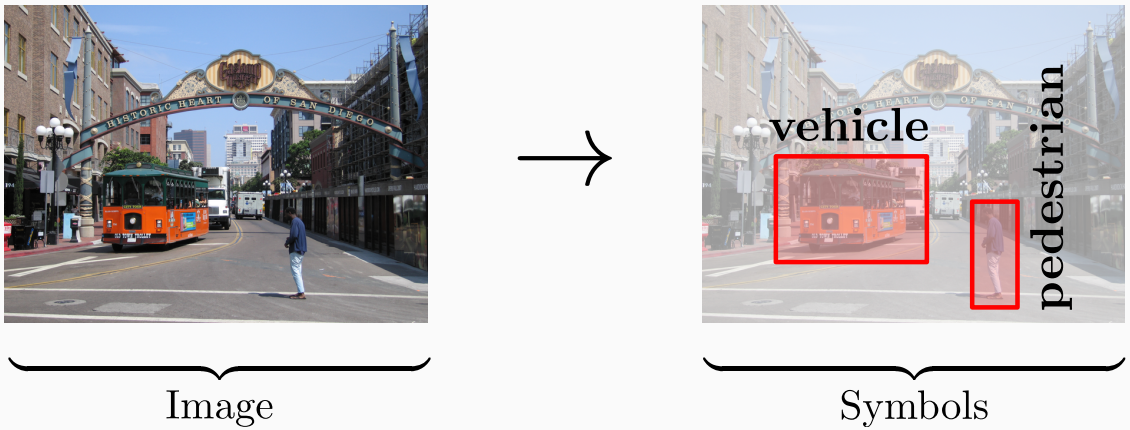


Symbols

Computer vision – Artificial Intelligence – Machine Learning

Definition (The British Machine Vision Association)

Computer vision (CV) consiste en l'extraction, analyse et compréhension d'information utile à partir d'une image ou d'un ensemble d'image.



CV is a subfield of Artificial Intelligence.

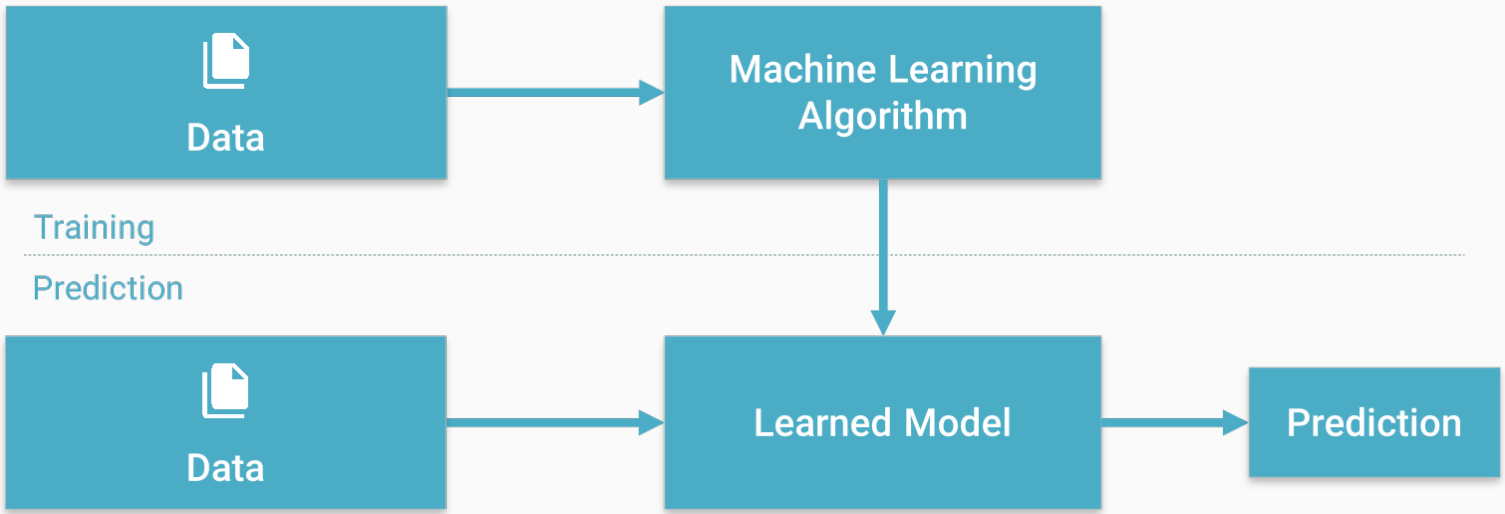
Definition (Oxford dictionary)

Artificial Intelligence, *noun*: La théorie et le développement de système informatique capable d'effectuer des tâches qui requièrent normalement l'intelligence humaine comme la perception visuelle, la reconnaissance de parole, la prise de décision ou la traduction.

Definition

Machine Learning, *noun*: Type d'intelligence artificielle qui premet à un système informatique d'apprendre une tâche sans être explicitement programmé pour.

L'objectif est de faire des prédictions à partir de données :



(Source: Lucas Masuch)

Deep learning: Academic actors

- Popularized by Hinton in 2006 with Restricted Boltzmann Machines

Geoffrey Hinton: University of Toronto & Google

- Developed by different actors:

Yann LeCun: New York University & Facebook

Andrew Ng: Stanford & Baidu

Yoshua Bengio: University of Montreal

Jürgen Schmidhuber: Swiss AI Lab & NNAISENSE

and many others...

Deep learning: Academic actors

- Popularized by Hinton in 2006 with Restricted Boltzmann Machines

Geoffrey Hinton: University of Toronto & Google

- Developed by different actors:

Yann LeCun: New York University & Facebook

Andrew Ng: Stanford & Baidu

Yoshua Bengio: University of Montreal

Jürgen Schmidhuber: Swiss AI Lab & NNAISENSE

and many others...

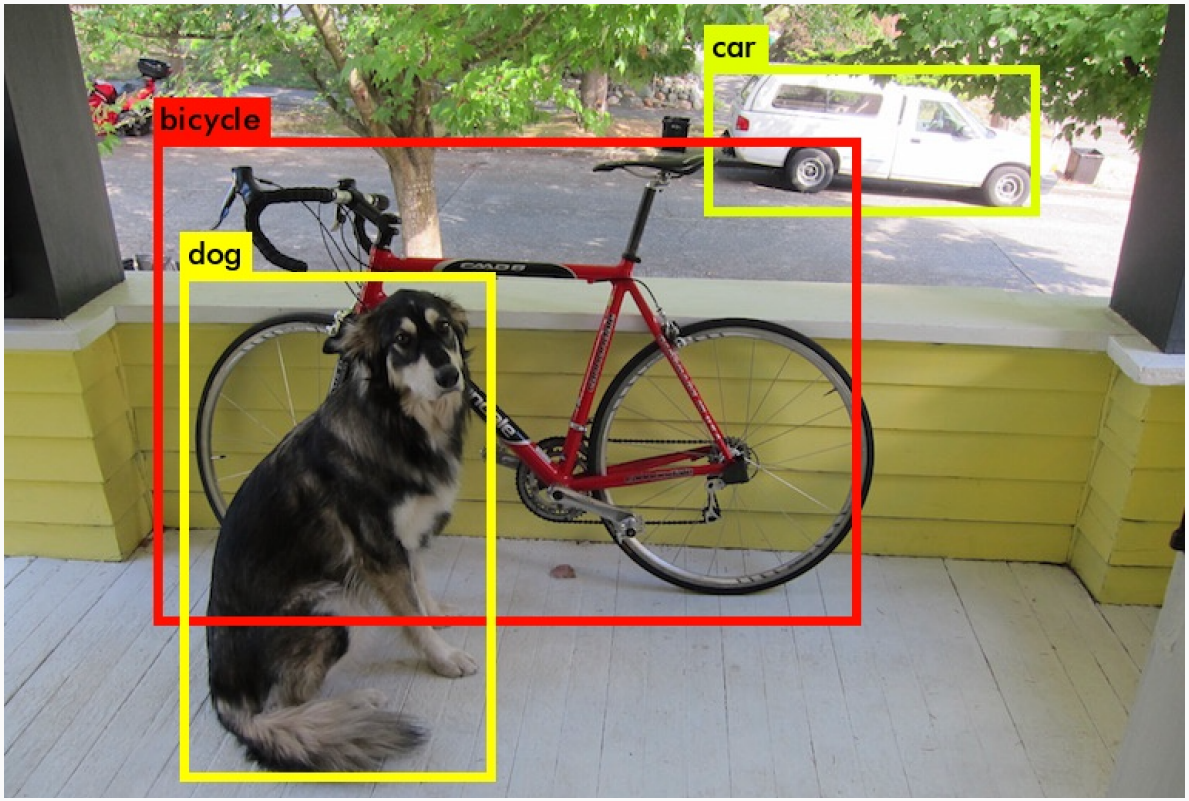
- Yoshua Bengio, Geoffrey Hinton, and Yann LeCun recipients of the 2018 ACM A.M. Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.

Actors and applications

- Very active technology adopted by big actors

- Success story for many different academic problems
  - Image processing
  - Computer vision
  - Speech recognition
  - Natural language processing
  - Translation
  - etc
- Today all industries wonder if Machine learning can improve their process.

Computer vision – Object detection



(Source: Joseph Redmon)

Goal: to detect instances of objects of a certain class (such as human).

Computer vision – Image segmentation

(Source: Abhijit Kundu)

Goal: to partition an image into multiple segments such that pixels in a same segment share certain characteristics (color, texture or semantic).

Computer vision – Image captioning

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

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

"man in blue wetsuit is surfing on wave."

"little girl is eating piece of cake."

"baseball player is throwing ball in game."

"woman is holding bunch of bananas."

"black cat is sitting on top of suitcase."

(Karpathy, Fei-Fei, CVPR, 2015)

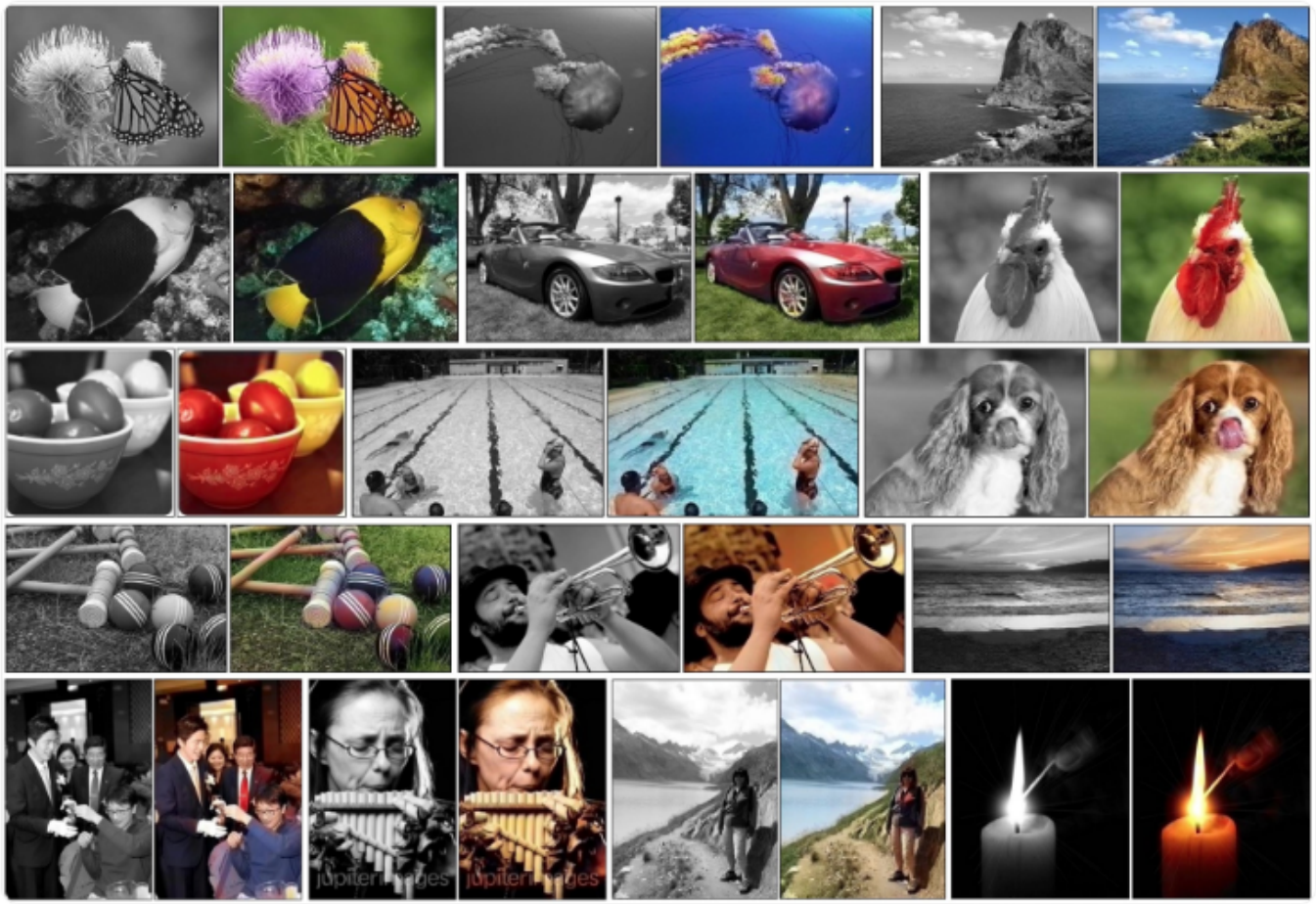
Goal: to write a sentence that describes what is happening.

Computer vision – Depth estimation

(Stereo-vision: from two images acquired with different views.)

Goal: to estimate a depth map from one, two or several frames.

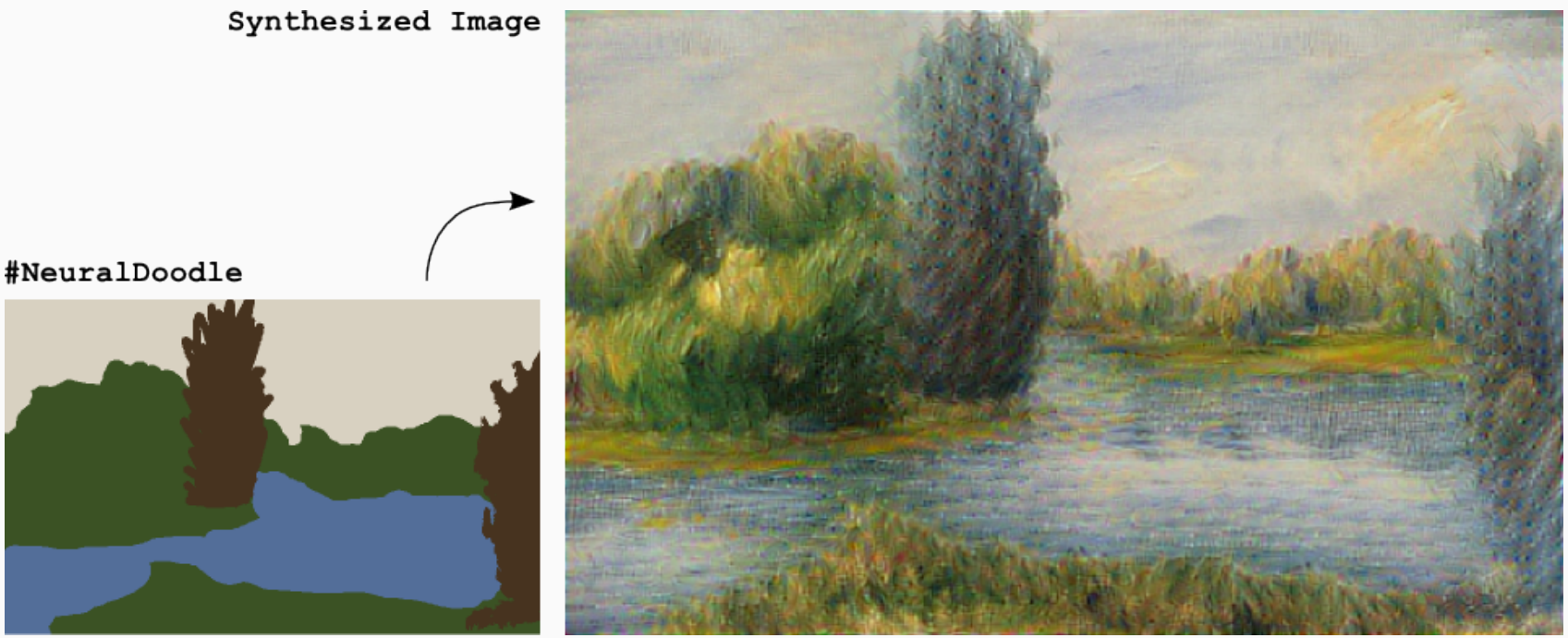
Image colorization



(Source: Richard Zhang, Phillip Isola and Alexei A. Efros, 2016)

Goal: to add color to grayscale photographs.

Image stylization



(Source: Neural Doodle, Champandard, 2016)

Goal: to create stylized images from rough sketches.

Style transfer

(Source: Gatys, Ecker and Bethge, 2015)

Goal: transfer the style of an image into another one.

Image generation

Generated images of bedrooms (Source: Alec Radford, Luke Metz, Soumith Chintala, 2015)

Goal: to automatically create realistic pictures of a given category.

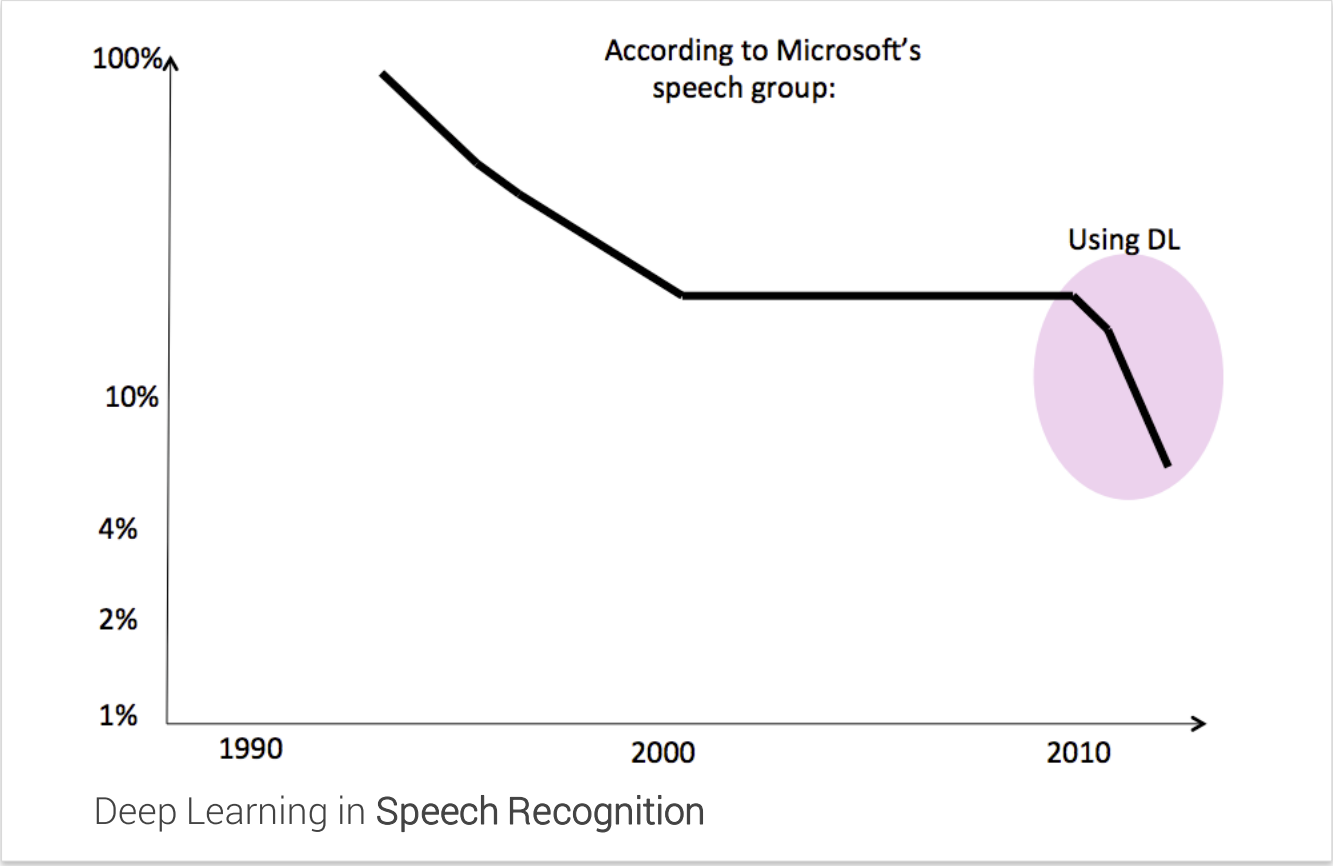
Success stories

Face generation (Style GAN, (Karras et al., 2018) (NVIDIA)):

These people do not exist.

Success stories

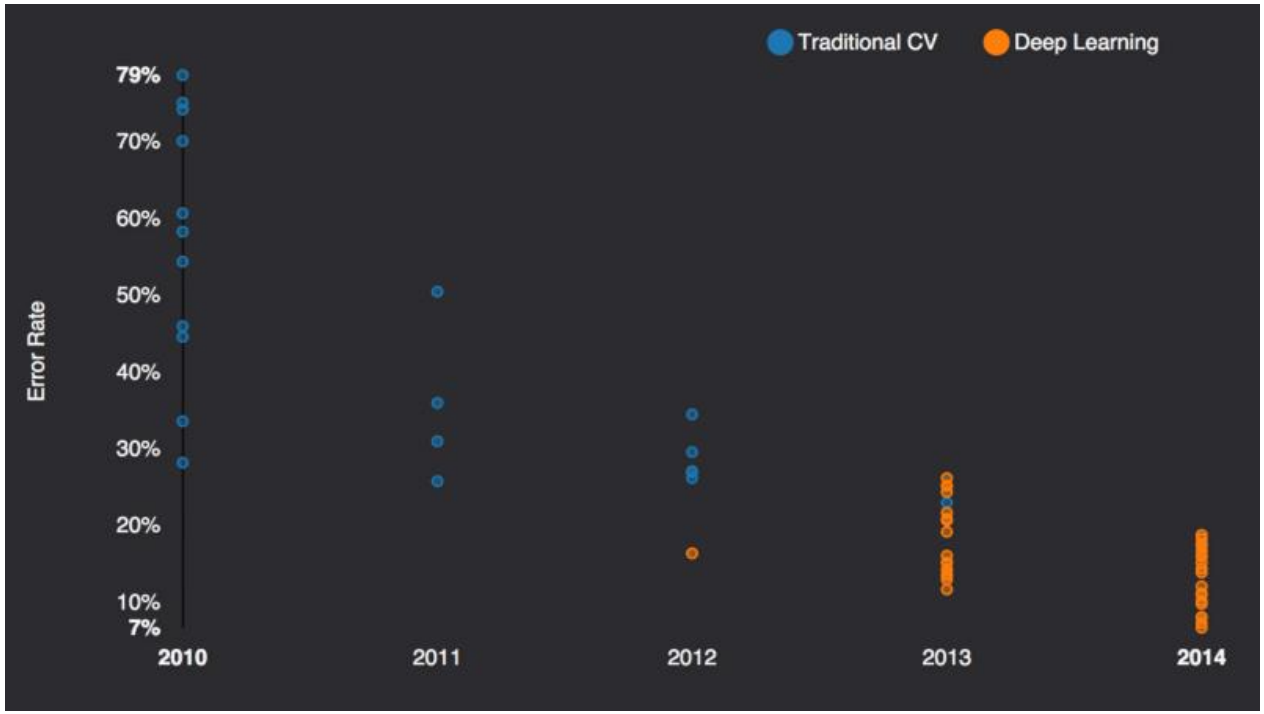
Speech recognition



Same improvements in other fields: automatic translation, Go, ...

Success stories

AlexNet wins the ImageNet classification challenge (Krizhevsky, 2012)

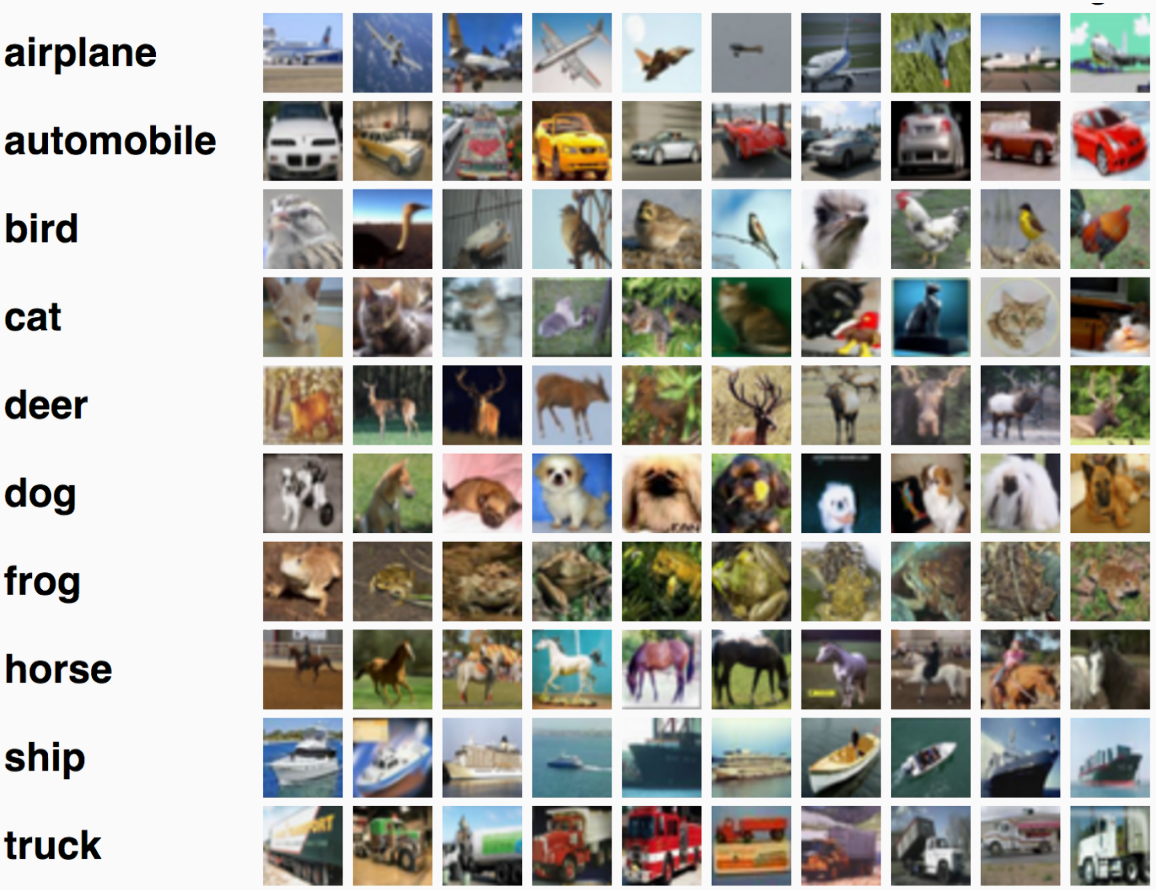


ImageNet: The "computer vision World Cup"

Since, Deep Learning has been the standard and achieved remarkable results.

Our goal: Image classification

Computer vision – Image classification



Goal: to assign a given image into one of the predefined classes.

Plan of the course: How to learn from examples

Plan of the course: How to learn from examples

3 main ingredients

1. Training set / examples (Données d'entraînement):

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$$

Plan of the course: How to learn from examples

3 main ingredients

1. Training set / examples (Données d'entraînement):

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$$

2. Machine or model (Modèle):

$$\boldsymbol{x} \rightarrow \underbrace{f(\boldsymbol{x}; \theta)}_{\text{function / algorithm}} \rightarrow \underbrace{\boldsymbol{y}}_{\text{prediction}}$$

θ: parameters of the model

Plan of the course: How to learn from examples

3 main ingredients

1. Training set / examples (Données d'entraînement):

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$$

2. Machine or model (Modèle):

$$\boldsymbol{x} \rightarrow \underbrace{f(\boldsymbol{x}; \theta)}_{\text{function / algorithm}} \rightarrow \underbrace{\boldsymbol{y}}_{\text{prediction}}$$

θ: parameters of the model

3. Loss, cost, objective function / energy (Fonction de coût):

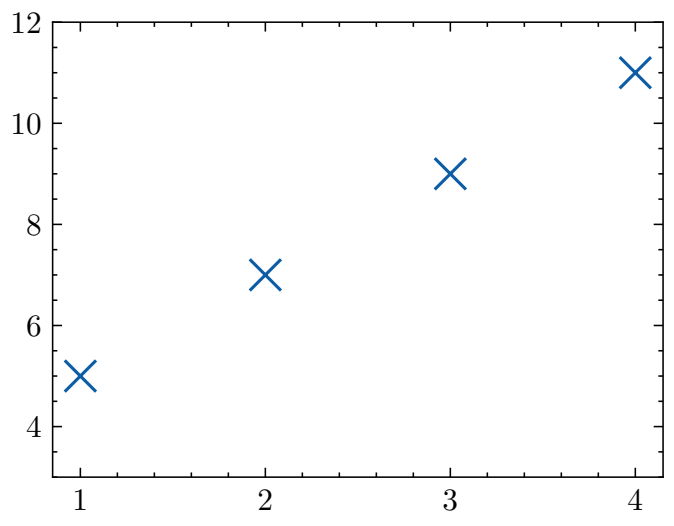
$$\underset{\theta}{\operatorname{argmin}}\ E(\theta; \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$$

Exemple de la regression linéaire:

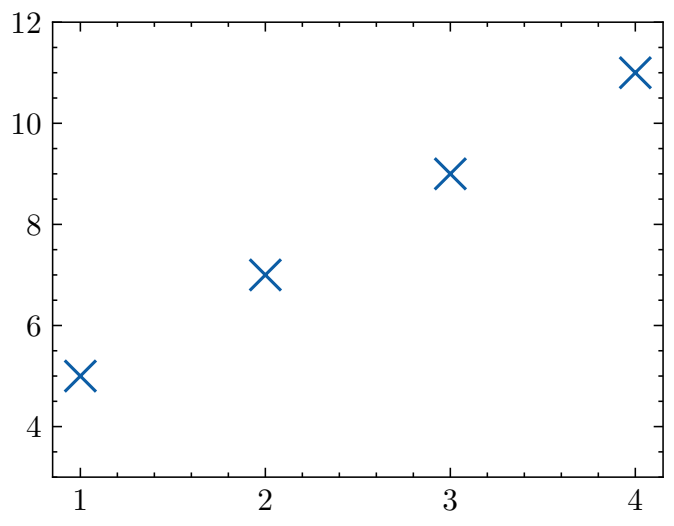
- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.

Exemple de la regression linéaire:

- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.



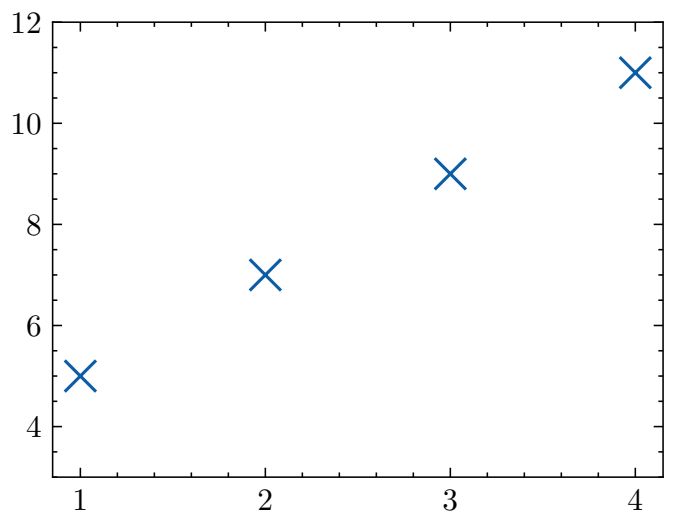
Exemple de la regression linéaire:



- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

Exemple de la regression linéaire:

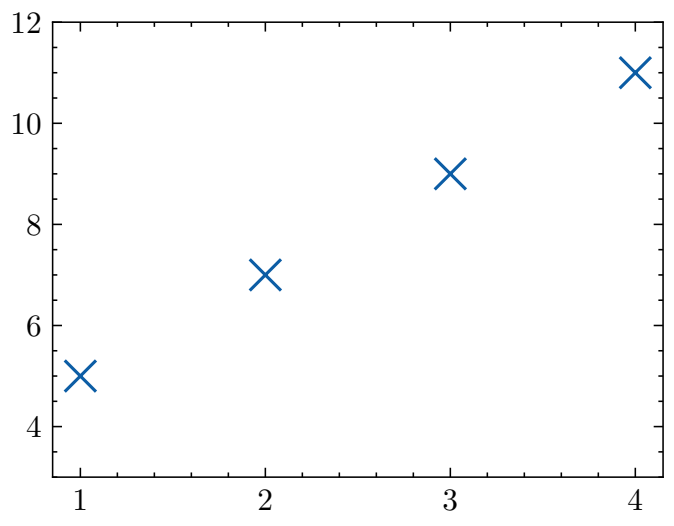


- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

- Modèle retenu :
  $\{f_\theta : x \mapsto ax + b\}_{\theta=(a,b)\in\mathbb{R}^2}$

Exemple de la regression linéaire:

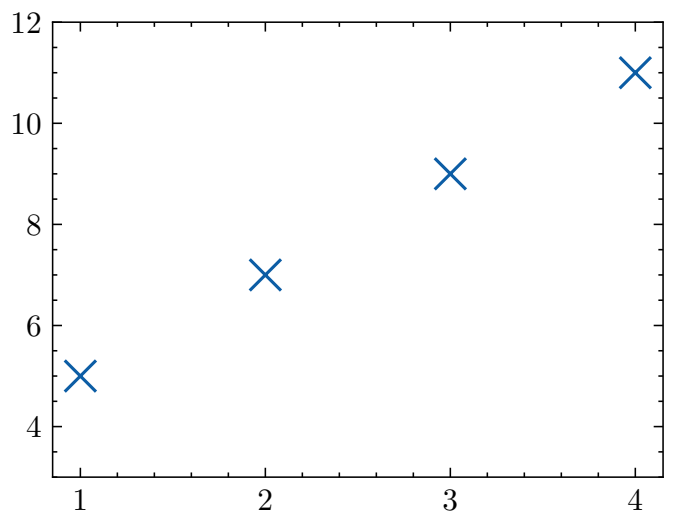


- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

- Modèle retenu :
  $\{f_\theta : x \mapsto ax + b\}_{\theta=(a,b)\in\mathbb{R}^2}$
- Loss :
  $$E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right) = \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$$

Exemple de la regression linéaire:

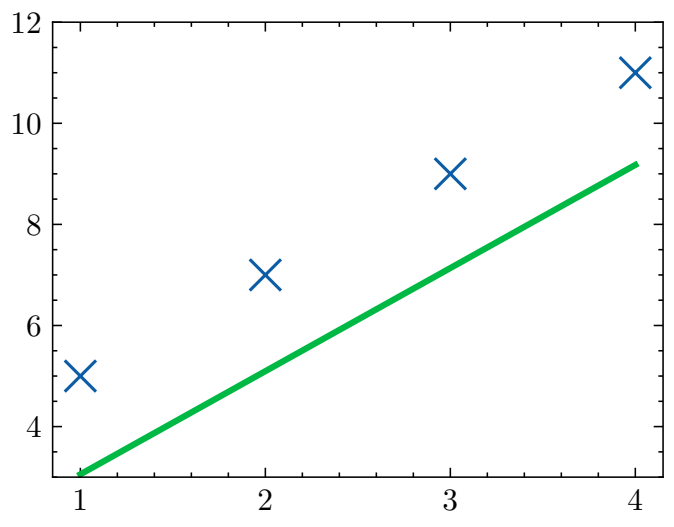


- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

- Modèle retenu :
  $\{f_\theta : x \mapsto ax + b\}_{\theta=(a,b)\in\mathbb{R}^2}$
- Loss :
  $E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right) = \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$
- Loss minimization:
  $\theta^\star = \text{argmin}_{\theta\in\mathbb{R}^2} E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right)$

Exemple de la regression linéaire:

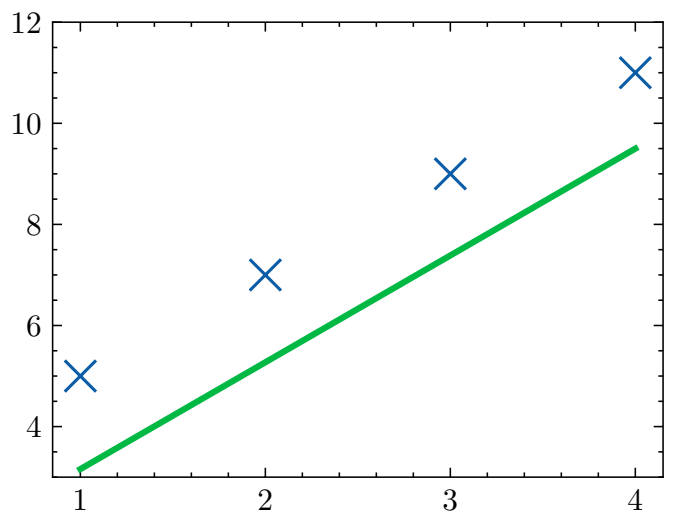


- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

- Modèle retenu :
  $\{f_\theta : x \mapsto ax + b\}_{\theta=(a,b)\in\mathbb{R}^2}$
- Loss :
  $E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right) = \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$
- Loss minimization:
  $\theta^\star = \text{argmin}_{\theta\in\mathbb{R}^2} E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right)$

Exemple de la regression linéaire:

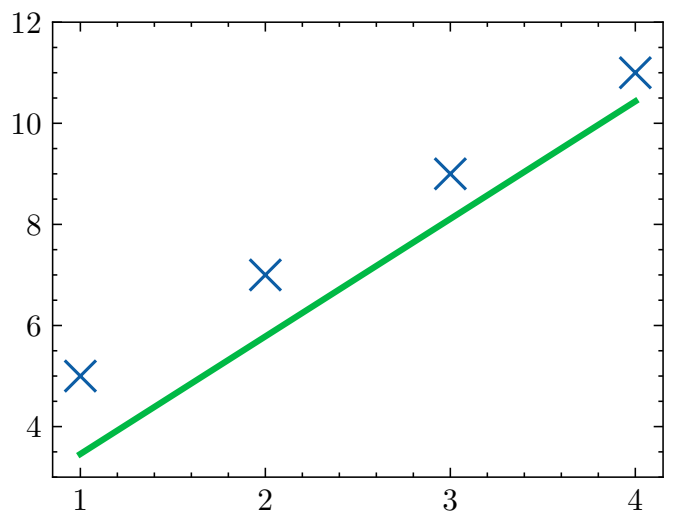


- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

- Modèle retenu :
  $\{f_\theta : x \mapsto ax + b\}_{\theta=(a,b)\in\mathbb{R}^2}$
- Loss :
  $E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right) = \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$
- Loss minimization:
  $\theta^\star = \text{argmin}_{\theta\in\mathbb{R}^2}\, E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right)$

Exemple de la regression linéaire:

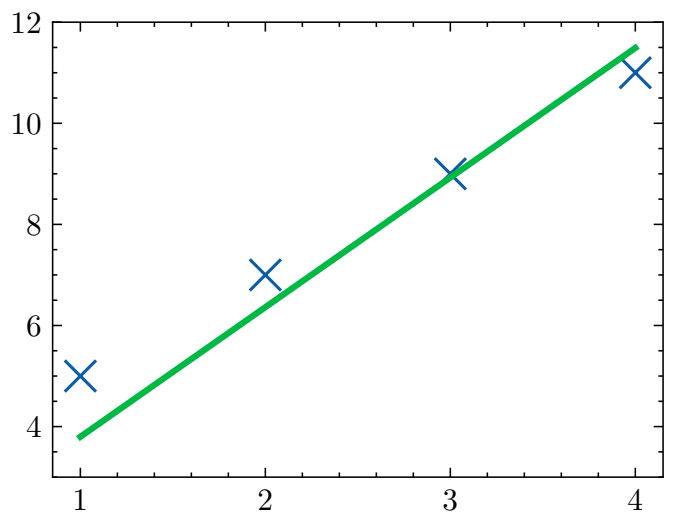


- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

- Modèle retenu :
  $\{f_\theta : x \mapsto ax + b\}_{\theta=(a,b)\in\mathbb{R}^2}$
- Loss :
  $E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right) = \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$
- Loss minimization:
  $\theta^\star = \text{argmin}_{\theta\in\mathbb{R}^2} E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right)$

Exemple de la regression linéaire:

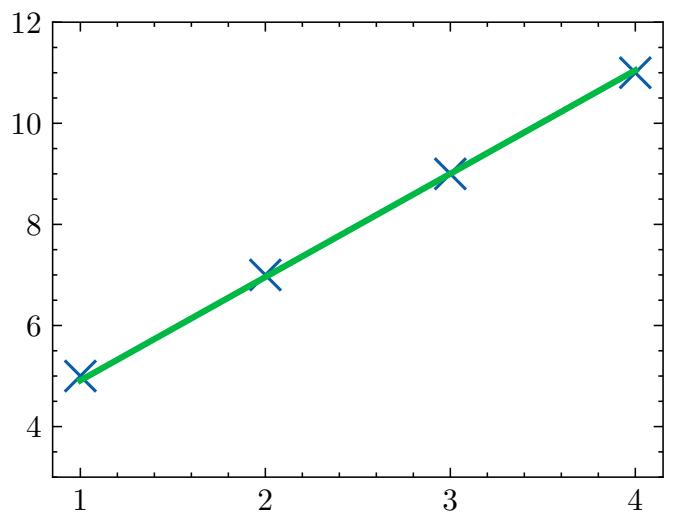


- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

- Modèle retenu :
  $\{f_\theta : x \mapsto ax + b\}_{\theta=(a,b)\in\mathbb{R}^2}$
- Loss :
  $E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right) = \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$
- Loss minimization:
  $\theta^\star = \text{argmin}_{\theta\in\mathbb{R}^2} E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right)$

Exemple de la regression linéaire:

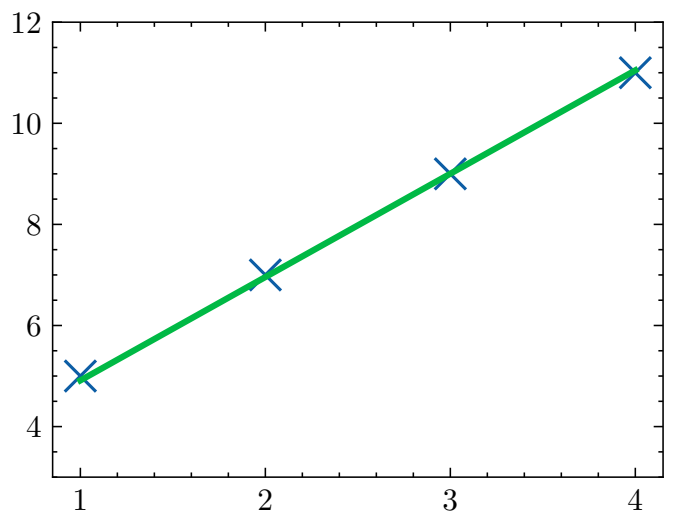


- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

- Modèle retenu :
  $\{f_\theta : x \mapsto ax + b\}_{\theta=(a,b)\in\mathbb{R}^2}$
- Loss :
  $E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right) = \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$
- Loss minimization:
  $\theta^\star = \text{argmin}_{\theta\in\mathbb{R}^2} E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right)$

Exemple de la regression linéaire:

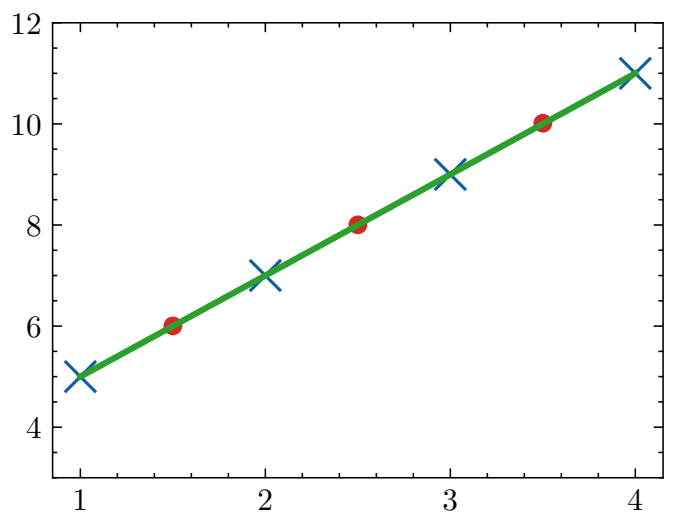


- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

- Modèle retenu :
  $\{f_\theta : x \mapsto ax + b\}_{\theta=(a,b)\in\mathbb{R}^2}$
- Loss :
  $E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right) = \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$
- Loss minimization:
  $\theta^\star = \text{argmin}_{\theta\in\mathbb{R}^2} E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right)$
- Puis, phase de test.

Exemple de la regression linéaire:



- Données d'entraînement: $(x_i, y_i)_{1 \leqslant i \leqslant N}$.
- Objectif : Trouver f_θ tel que:

$$y_i = f_\theta(x_i), \quad 1 \leqslant i \leqslant N$$

- Modèle retenu :
  $\{f_\theta : x \mapsto ax + b\}_{\theta=(a,b)\in\mathbb{R}^2}$
- Loss :
  $E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right) = \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$
- Loss minimization:
  $\theta^\star = \text{argmin}_{\theta\in\mathbb{R}^2} E\left(\theta, (x_i, y_i)_{1 \leqslant i \leqslant N}\right)$
- Puis, phase de test.

Terminology

Echantillon (Sample/Observation/Data): Elément à traiter (*e.g.*, classify).
Example: un individu, un document, **une image**, *un son, une vidéo. . .*

Terminology

Echantillon (Sample/Observation/Data): Elément à traiter (*e.g.*, classify). *Example: un individu, un document,* **une image**, *un son, une vidéo. . .*

Features (Input): Ensemble de traits distincts qui peuvent être utilisés pour décrire chaque échantillon de manière quantitative. Habituellement représenté comme un vecteur multidimensionnel désigné par $\boldsymbol{x}$. *Exemple: taille, poids, nationalité, rayon . . .*

Terminology

Echantillon (Sample/Observation/Data): Elément à traiter (*e.g.*, classify). *Example: un individu, un document,* **une image**, *un son, une vidéo. . .*

Features (Input): Ensemble de traits distincts qui peuvent être utilisés pour décrire chaque échantillon de manière quantitative. Habituellement représenté comme un vecteur multidimensionnel désigné par $\boldsymbol{x}$. *Exemple: taille, poids, nationalité, rayon . . .*

Training set: Ensemble de données utilisé pour découvrir des relations potentiellement prédictives.

Terminology

Echantillon (Sample/Observation/Data): Elément à traiter (*e.g.*, classify). *Example: un individu, un document,* **une image**, *un son, une vidéo. . .*

Features (Input): Ensemble de traits distincts qui peuvent être utilisés pour décrire chaque échantillon de manière quantitative. Habituellement représenté comme un vecteur multidimensionnel désigné par $\boldsymbol{x}$. *Exemple: taille, poids, nationalité, rayon . . .*

Training set: Ensemble de données utilisé pour découvrir des relations potentiellement prédictives.

Validation set: Ensemble utilisé pour ajuster les hyperparamètres du modèle (à utiliser comme feedback).

Terminology

Echantillon (Sample/Observation/Data): Elément à traiter (*e.g.*, classify). *Example: un individu, un document,* **une image**, *un son, une vidéo. . .*

Features (Input): Ensemble de traits distincts qui peuvent être utilisés pour décrire chaque échantillon de manière quantitative. Habituellement représenté comme un vecteur multidimensionnel désigné par $\boldsymbol{x}$. *Exemple: taille, poids, nationalité, rayon . . .*

Training set: Ensemble de données utilisé pour découvrir des relations potentiellement prédictives.

Validation set: Ensemble utilisé pour ajuster les hyperparamètres du modèle (à utiliser comme feedback).

Testing set: Ensemble utilisé pour tester les performances du modèles.

Terminology

Echantillon (Sample/Observation/Data): Elément à traiter (*e.g.*, classify). *Example: un individu, un document,* **une image**, *un son, une vidéo. . .*

Features (Input): Ensemble de traits distincts qui peuvent être utilisés pour décrire chaque échantillon de manière quantitative. Habituellement représenté comme un vecteur multidimensionnel désigné par $\boldsymbol{x}$. *Exemple: taille, poids, nationalité, rayon . . .*

Training set: Ensemble de données utilisé pour découvrir des relations potentiellement prédictives.

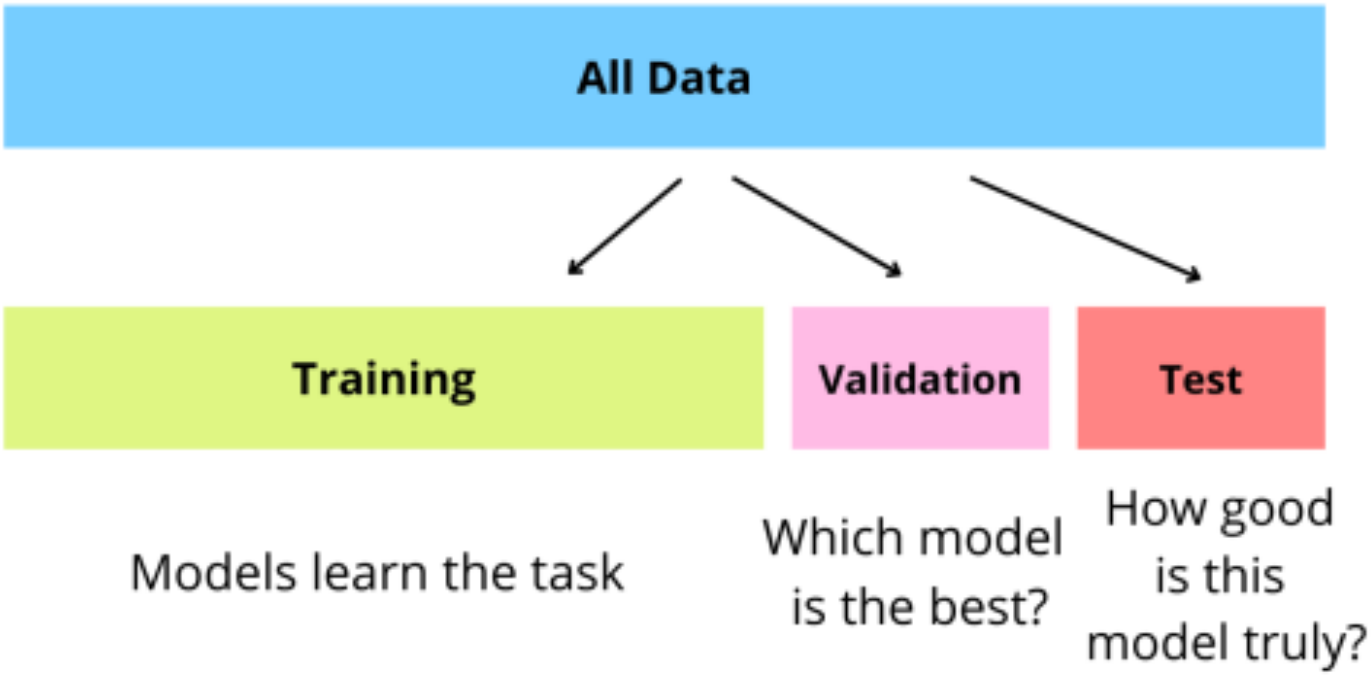
Validation set: Ensemble utilisé pour ajuster les hyperparamètres du modèle (à utiliser comme feedback).

Testing set: Ensemble utilisé pour tester les performances du modèles.

Label (Output):

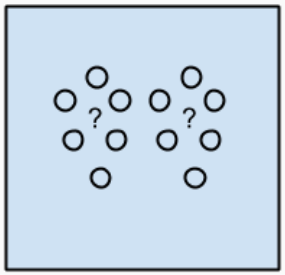
Classe ou résultat attribué à un échantillon. La prédiction réelle est souvent

Training set/Test set

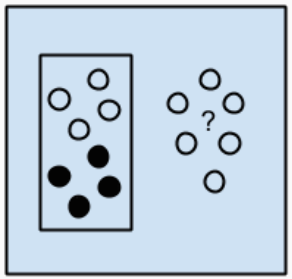


In our examples, there will be not validation dataset.

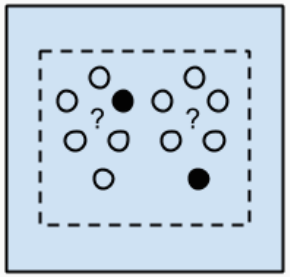
Plusieurs façons d'entraîner une fonction



Unsupervised Learning Algorithms

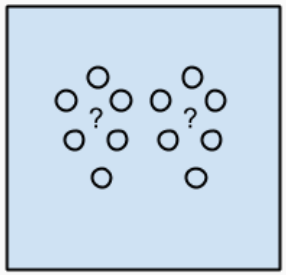


Supervised Learning Algorithms

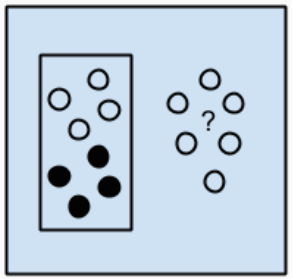


Semi-supervised Learning Algorithms

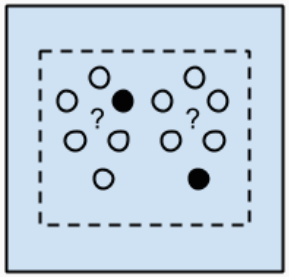
Plusieurs façons d'entraîner une fonction



Unsupervised Learning Algorithms



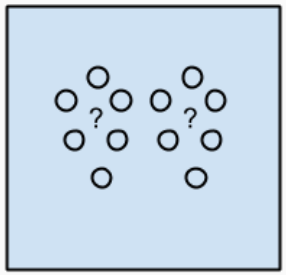
Supervised Learning Algorithms



Semi-supervised Learning Algorithms

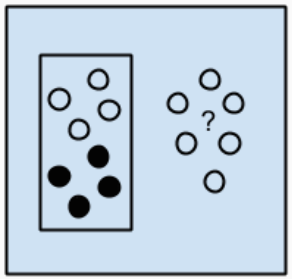
Apprentissage non supervisé: En cherchant à différencier les données sans qu'elles soient explicitement étiquetées.
Example: cluster similar documents based on the text content.

Plusieurs façons d'entraîner une fonction



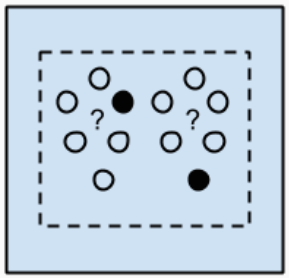
Unsupervised Learning Algorithms

Apprentissage non supervisé: En cherchant à différencier les données sans qu'elles soient explicitement étiquetées. *Example: cluster similar documents based on the text content.*



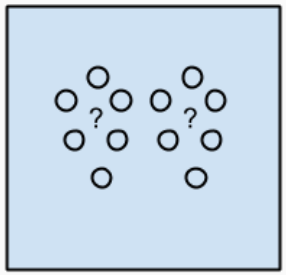
Supervised Learning Algorithms

Apprentissage supervisé: En apprenant à différencier les données à partir de données déjà étiquetées. labeled training set. *Example: email spam detector with training set of already labeled emails.*



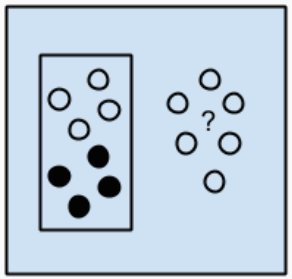
Semi-supervised Learning Algorithms

Plusieurs façons d'entraîner une fonction



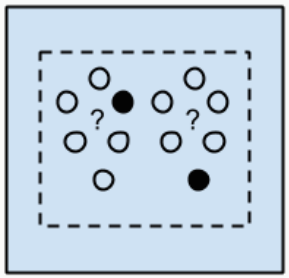
Unsupervised Learning Algorithms

Apprentissage non supervisé: En cherchant à différencier les données sans qu'elles soient explicitement étiquetées. *Example: cluster similar documents based on the text content.*



Supervised Learning Algorithms

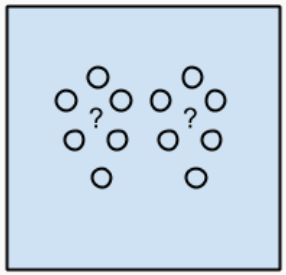
Apprentissage supervisé: En apprenant à différencier les données à partir de données déjà étiquetées. labeled training set. *Example: email spam detector with training set of already labeled emails.*



Semi-supervised Learning Algorithms

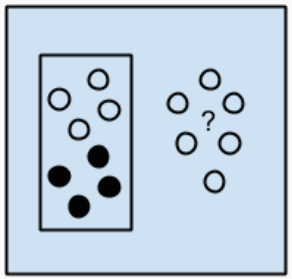
Apprentissage semi-supervisé: En apprenant avec une petite quantité de données étiquetées.. *Example: web content and protein sequence classifications.*

Plusieurs façons d'entraîner une fonction



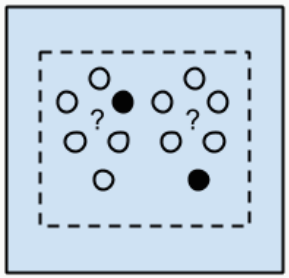
Unsupervised Learning Algorithms

Apprentissage non supervisé: En cherchant à différencier les données sans qu'elles soient explicitement étiquetées. *Example: cluster similar documents based on the text content.*



Supervised Learning Algorithms

Apprentissage supervisé: En apprenant à différencier les données à partir de données déjà étiquetées. labeled training set. *Example: email spam detector with training set of already labeled emails.*



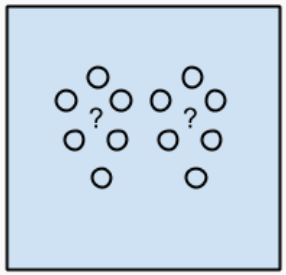
Semi-supervised Learning Algorithms

Apprentissage semi-supervisé: En apprenant avec une petite quantité de données étiquetées.. *Example: web content and protein sequence classifications.*

Reinforcement learning: Learning based on feedback or reward. *Example: learn to play chess by winning or losing.*

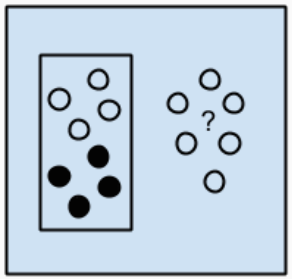
(Source: Jason Brownlee and Lucas Masuch)

Plusieurs façons d'entraîner une fonction



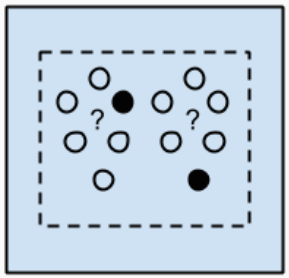
Unsupervised Learning Algorithms

Apprentissage non supervisé: En cherchant à différencier les données sans qu'elles soient explicitement étiquetées. *Example: cluster similar documents based on the text content.*



Supervised Learning Algorithms

Apprentissage supervisé: En apprenant à différencier les données à partir de données déjà étiquetées. labeled training set. *Example: email spam detector with training set of already labeled emails.*



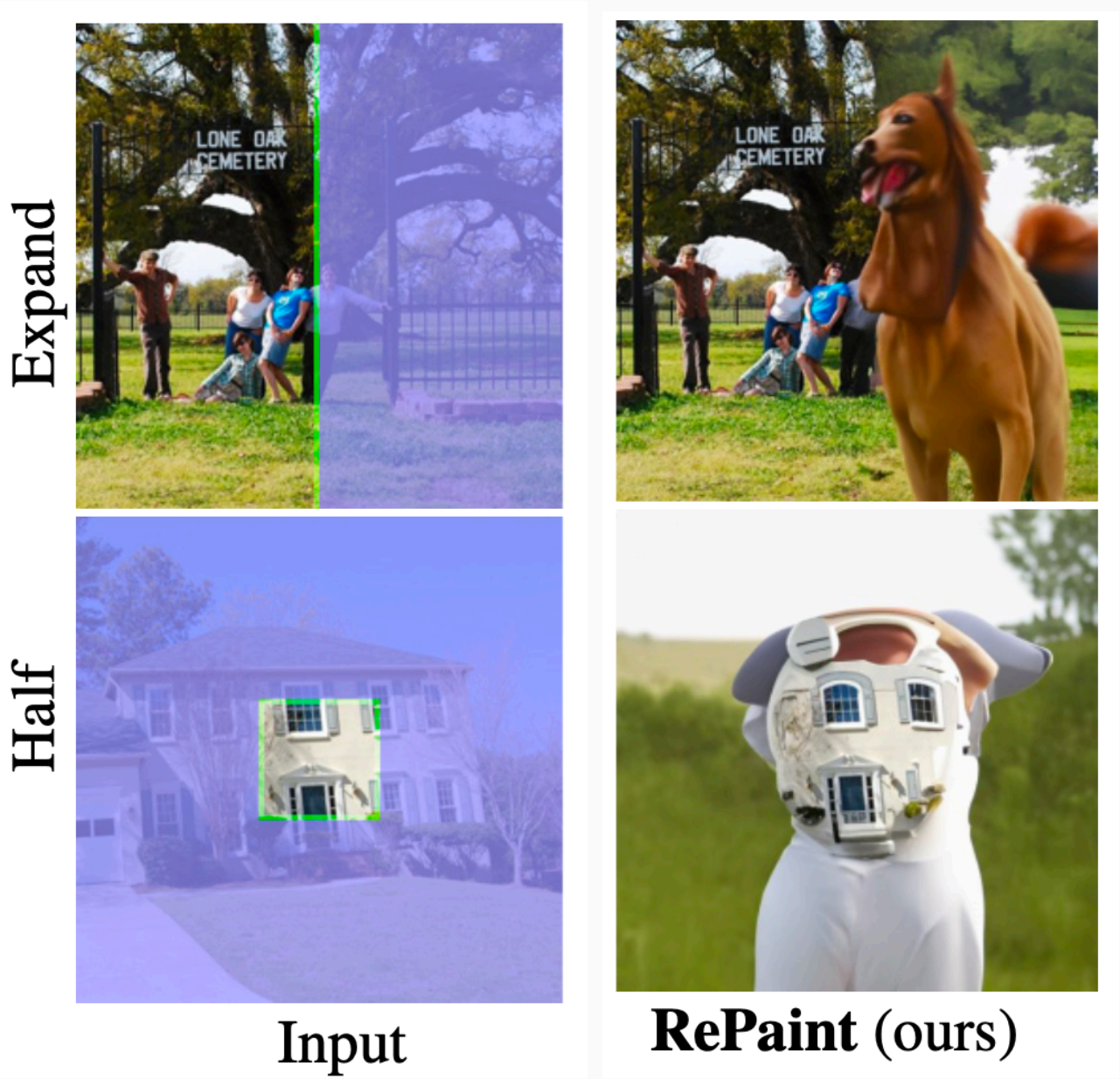
Semi-supervised Learning Algorithms

Apprentissage semi-supervisé: En apprenant avec une petite quantité de données étiquetées.. *Example: web content and protein sequence classifications.*

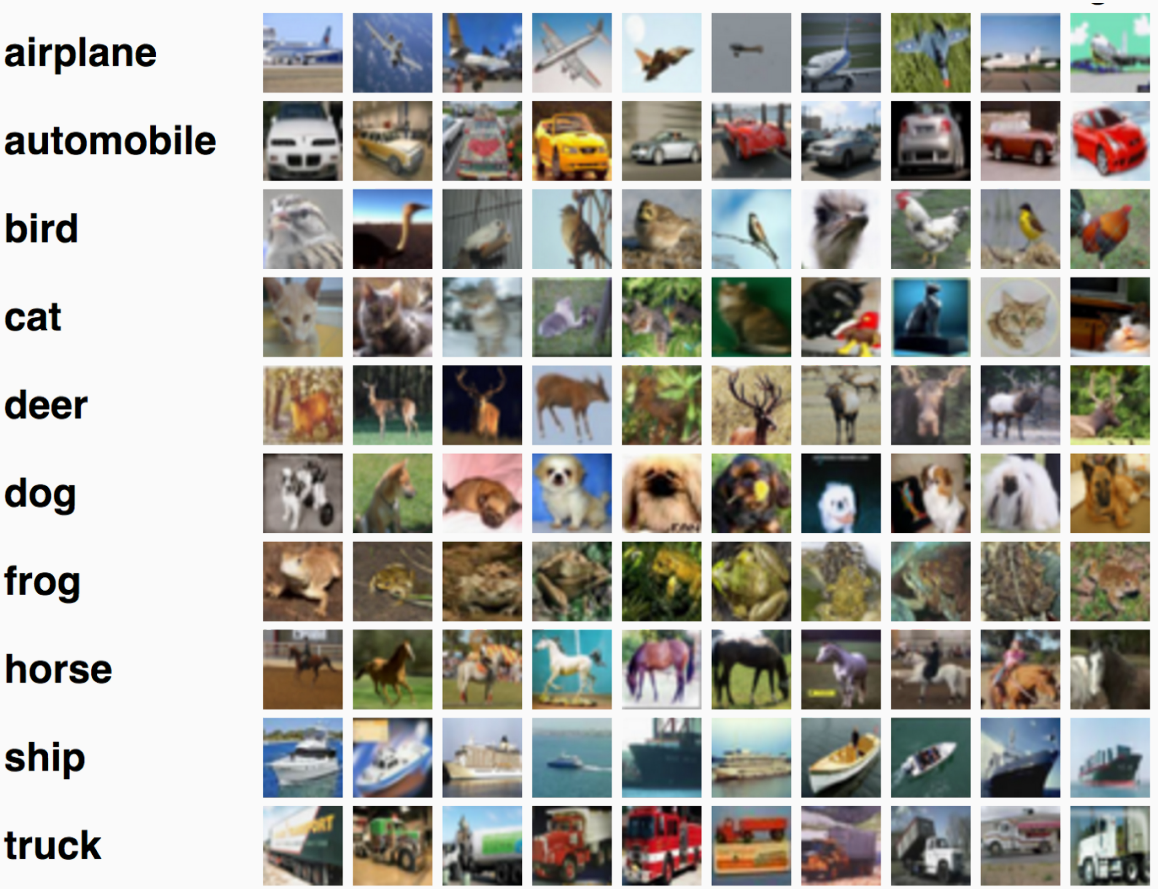
Reinforcement learning: Learning based on feedback or reward. *Example: learn to play chess by winning or losing.*

(Source: Jason Brownlee and Lucas Masuch)

Dataset problem



Classification example



Dataset: $(x_i, d_i)_{1 \leqslant i \leqslant N}$ where for $1 \leqslant i \leqslant N$, x_i is an image and x_i is in class d_i.

Learning from examples

3 main ingredients

1. Training set / examples:

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$$

Learning from examples

3 main ingredients

1. Training set / examples:

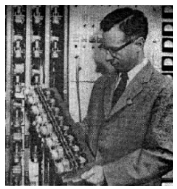
$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$$

2. Machine or model:

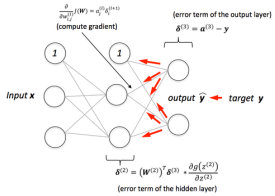
$$\boldsymbol{x} \rightarrow \underbrace{f(\boldsymbol{x}; \theta)}_{\text{function / algorithm}} \rightarrow \underbrace{\boldsymbol{y}}_{\text{prediction}}$$

θ: parameters of the model

Timeline of (deep) learning



1958 Perceptron



1974 Backpropagation

Convolution Neural Networks for Handwritten Recognition

1998

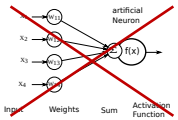
Google Brain Project on 16k Cores

2012

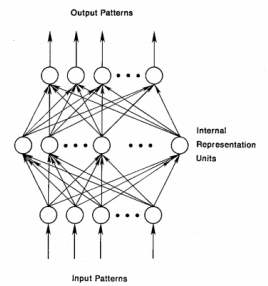
awkward silence (AI winter)

1969 Perceptrons book

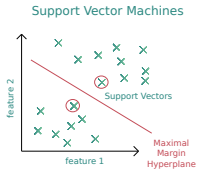
Perceptron criticized



~1980 Multilayer network



1995 SVM reigns



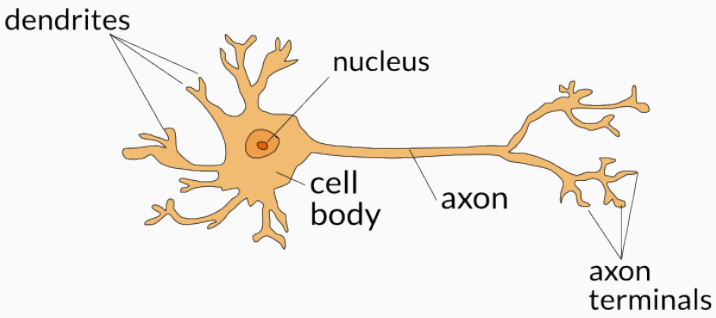
2006 Restricted Boltzmann Machine

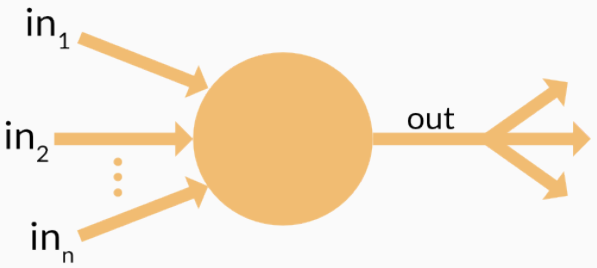
2012 AlexNet wins ImageNet

IMAGENET

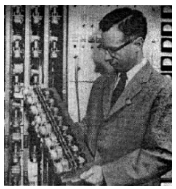
(Source: Lucas Masuch & Vincent Lepetit)

Perceptron


dendrites
nucleus
cell
body
axon
axon
terminals


in1
in2
inn
out

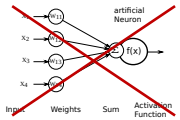
Perceptron



1958 Perceptron

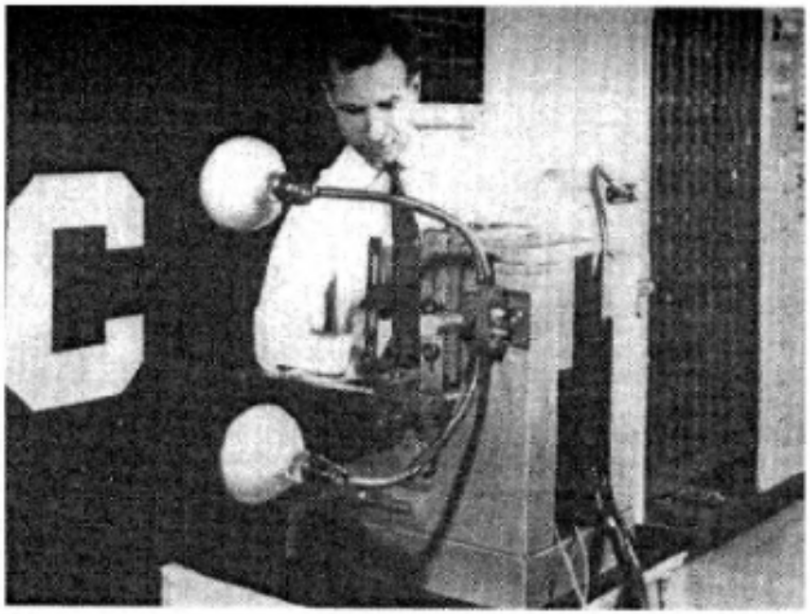
1969
Perceptrons
book

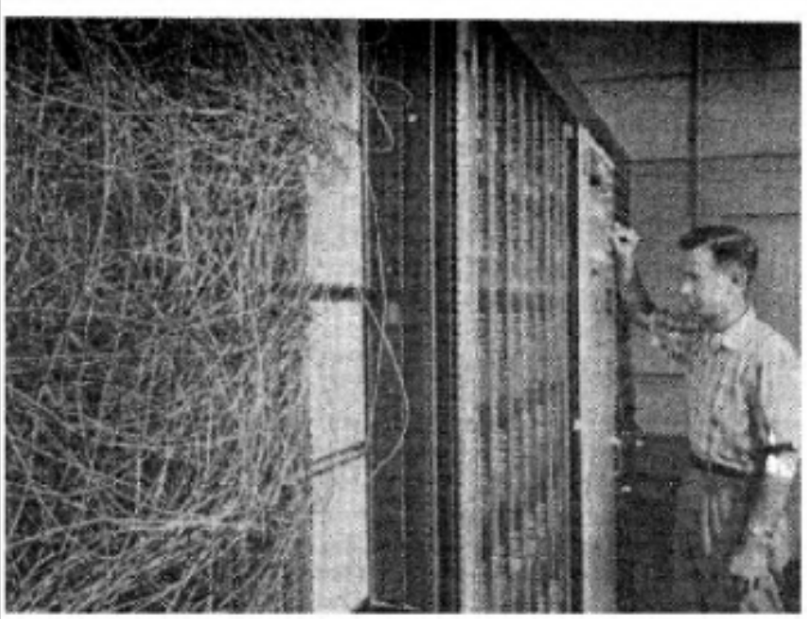
Perceptron criticized



(Source: Lucas Masuch & Vincent Lepetit)

Perceptron (Frank Rosenblatt, 1958)



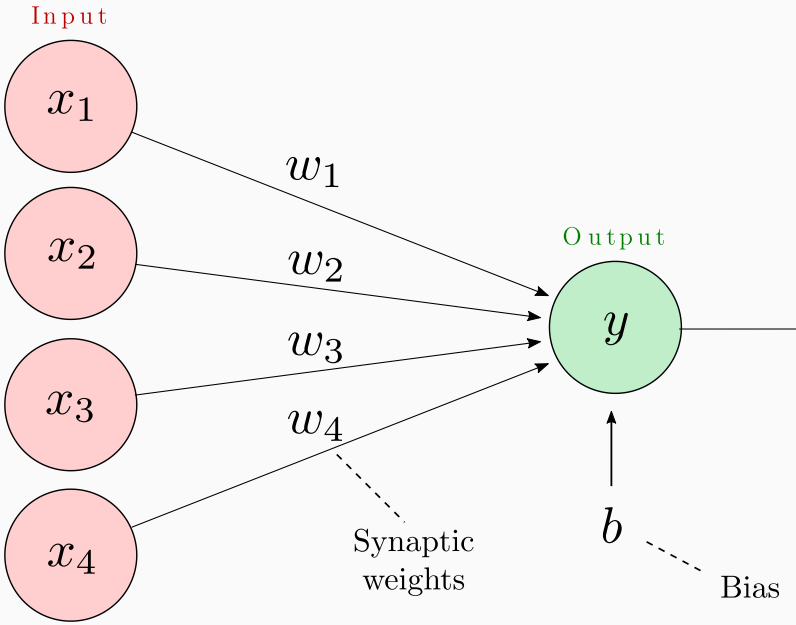


First binary classifier based on supervised learning (discrimination).

Foundation of modern artificial neural networks.

At that time: technological, scientific and philosophical challenges.

Representation of the Perceptron



$$\text{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$y = \text{sign}\left(\sum_{i=1}^{4} w_i x_i + b\right)$$

Parameters of the perceptron

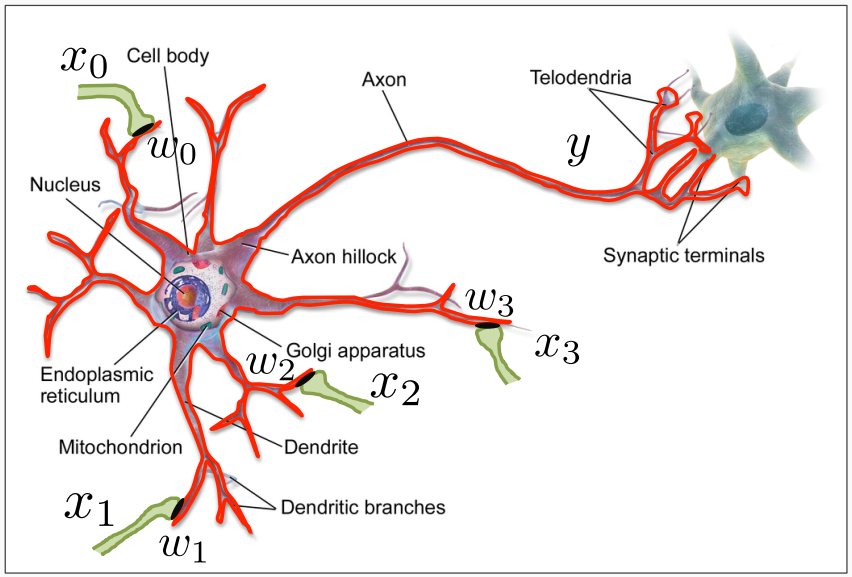
- w_k: synaptic weights
- b: bias

$\Longleftarrow$ real parameters to be estimated.

Training = adjusting the weights and biases

The origin of the Perceptron

Takes inspiration from the visual system known for its ability to learn patterns.



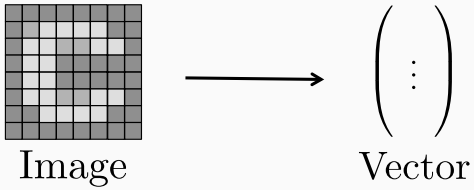
- When a neuron receives a stimulus with high enough voltage, it emits an action potential (aka, nerve impulse or spike). It is said to fire.
- The perceptron mimics this activation effect: it fires only when

$$\sum_i w_i x_i + b > 0$$

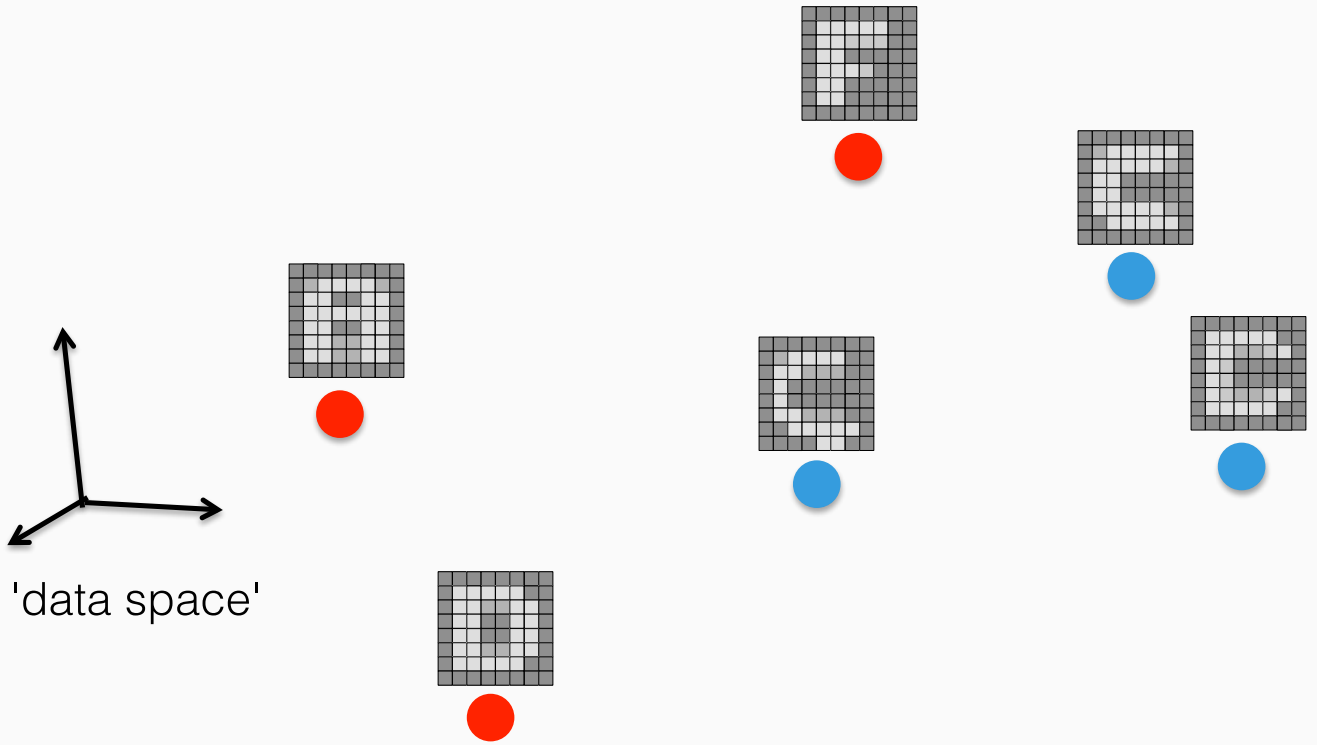
$$y = \underbrace{\operatorname{sign}(w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + b)}_{f(\boldsymbol{x};\boldsymbol{w})} = \begin{cases} +1 & \text{for the first class} \\ -1 & \text{for the second class} \end{cases}$$

Machine learning – Perceptron – Principle

1. Data are represented as vectors:



2. Collect training data with positive and negative examples:



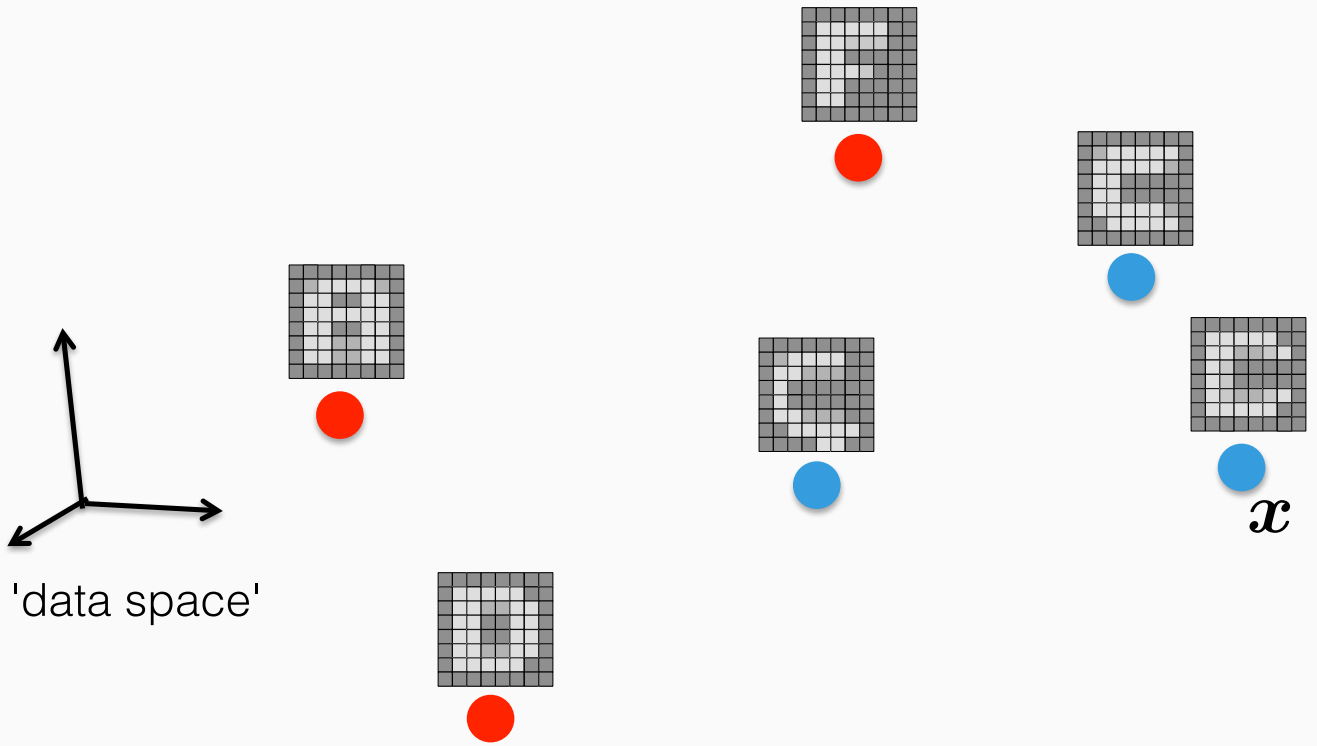
(Source: Vincent Lepetit)

❸ **Training:** find $\boldsymbol{w}$ and b so that:

- $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ is positive for positive samples $\boldsymbol{x}$,
- $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ is negative for negative samples $\boldsymbol{x}$.

Dot product:

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \sum_{i=1}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$$



(Source: Vincent Lepetit)

❸ **Training:** find $\boldsymbol{w}$ and b so that:

- $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ is positive for positive samples $\boldsymbol{x}$,
- $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ is negative for negative samples $\boldsymbol{x}$.

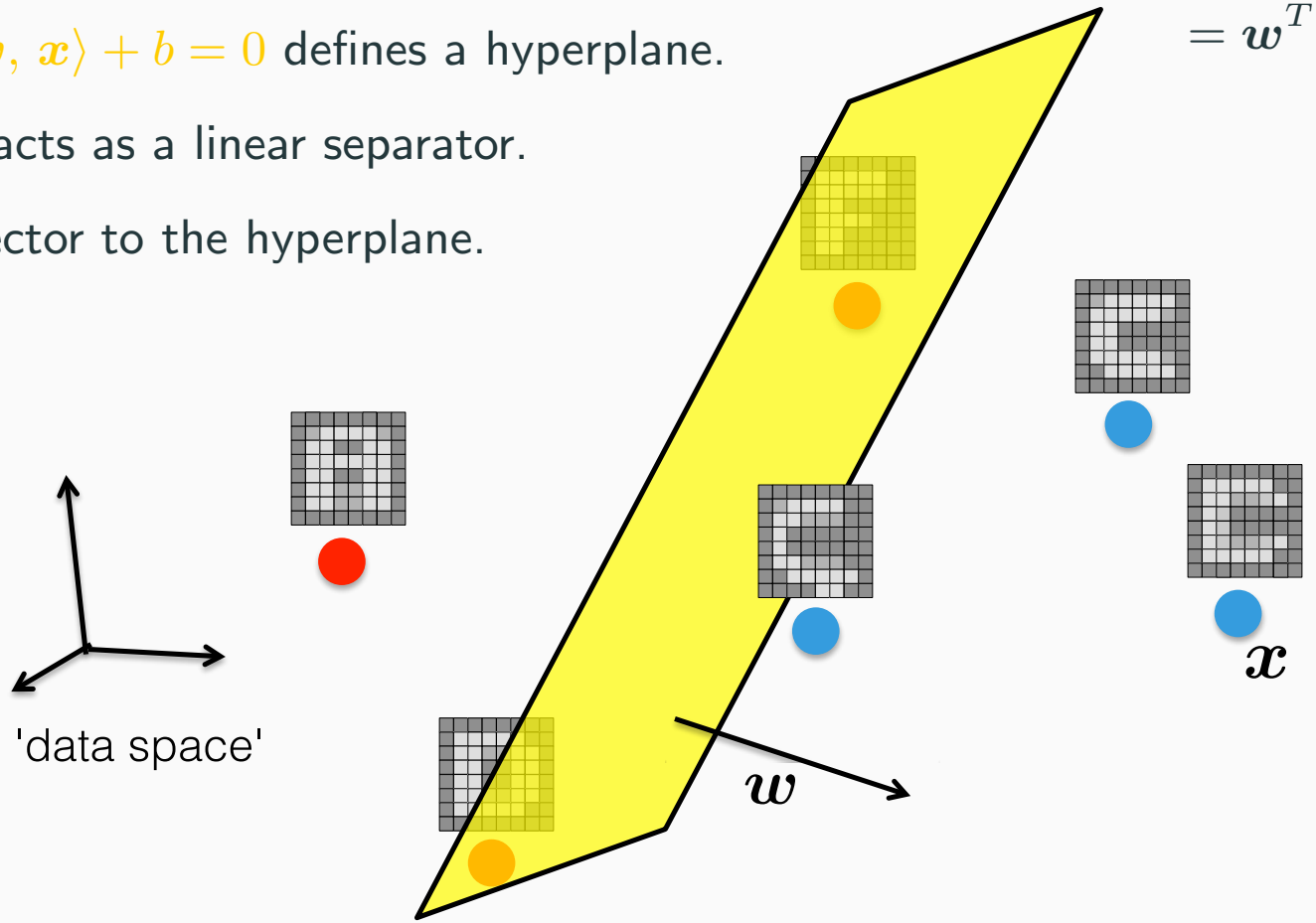
Dot product:

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \sum_{i=1}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$$

The equation $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$ defines a hyperplane.

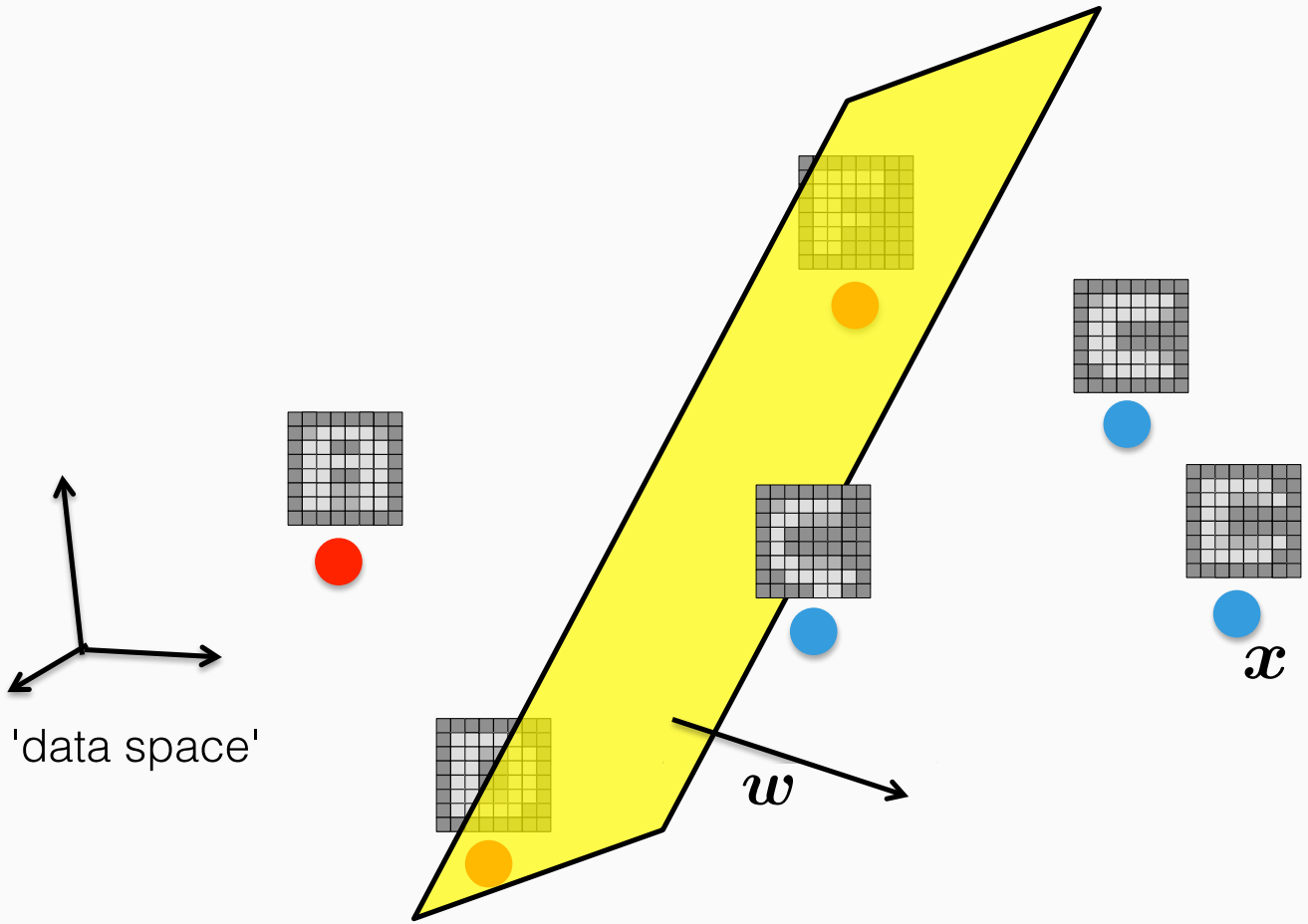
The hyperplane acts as a linear separator.

$\boldsymbol{w}$ is a normal vector to the hyperplane.



(Source: Vincent Lepetit)

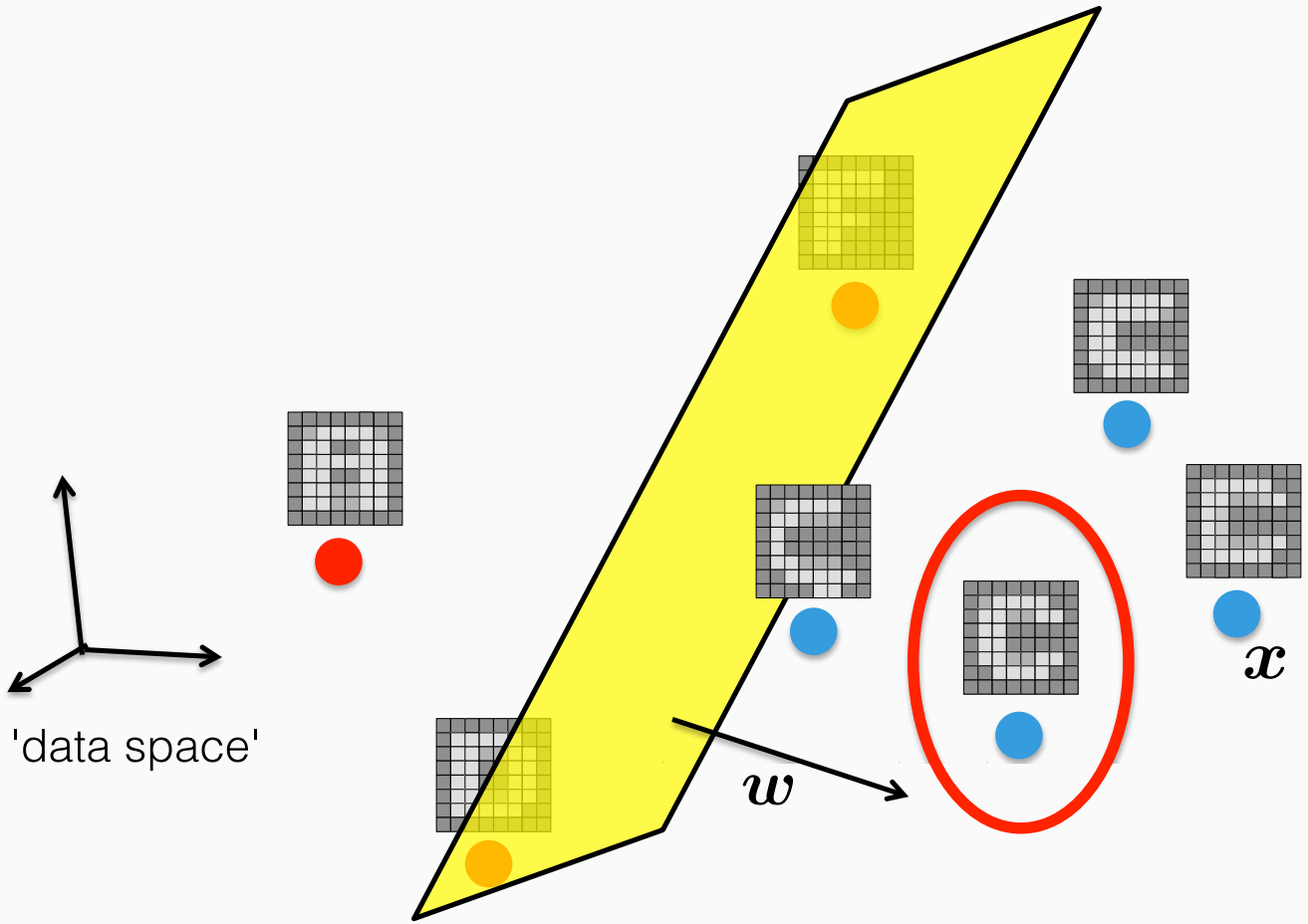
❹ **Testing:** the perceptron can now classify new examples.



(Source: Vincent Lepetit)

❹ **Testing:** the perceptron can now classify new examples.

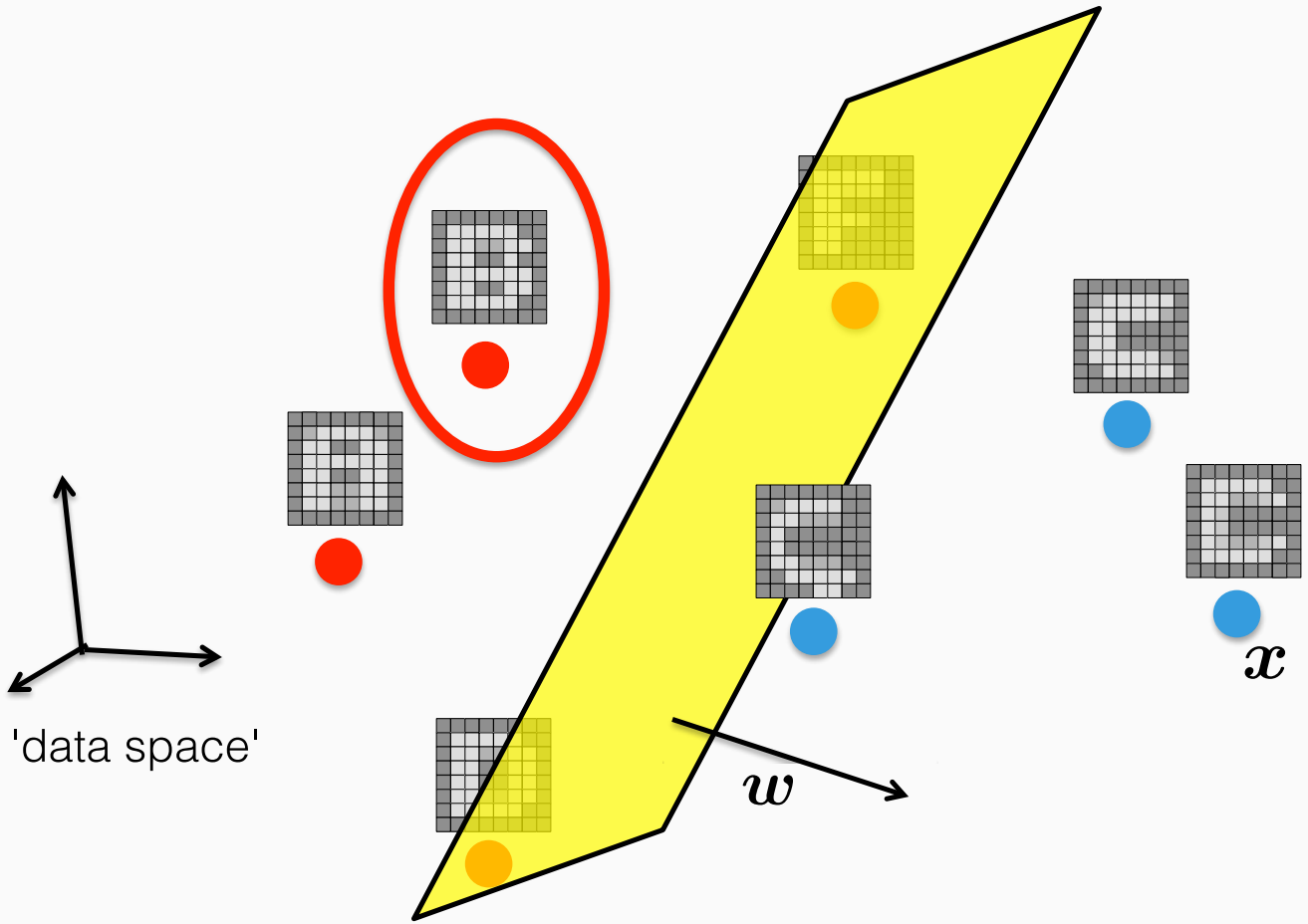
- A new example $\boldsymbol{x}$ is classified positive if $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ is positive,



(Source: Vincent Lepetit)

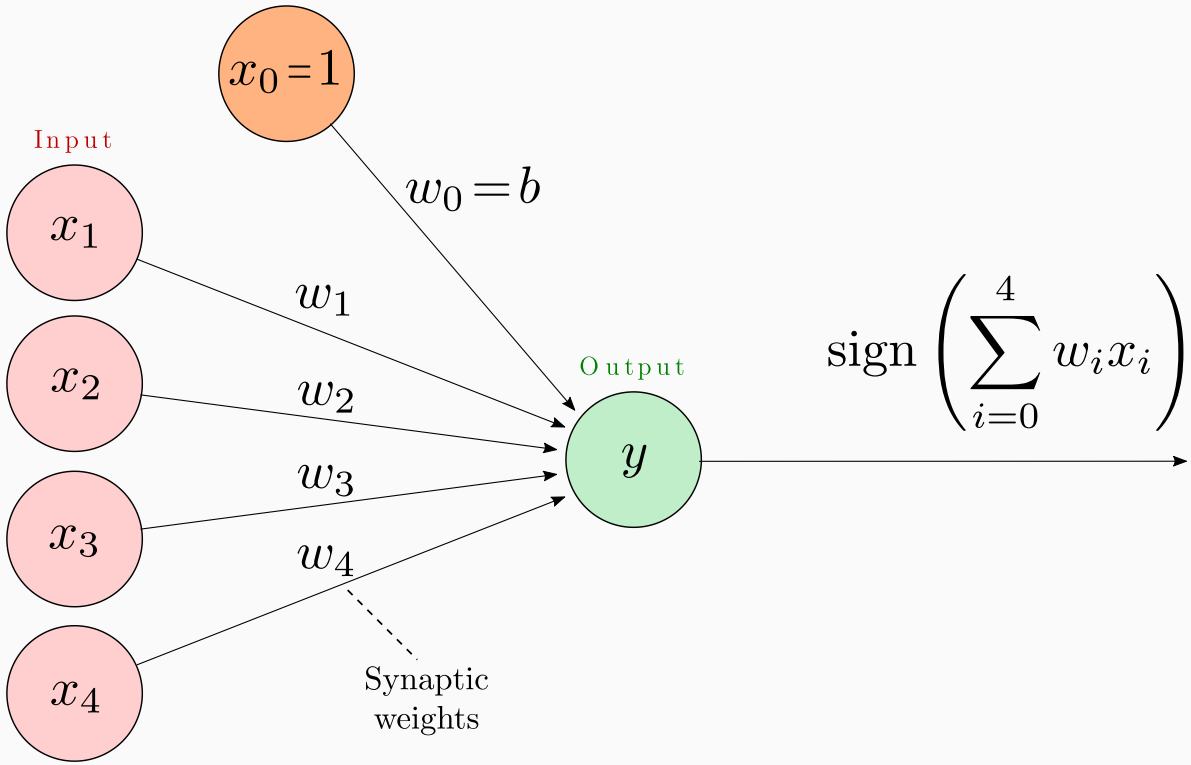
❹ **Testing:** the perceptron can now classify new examples.

- A new example $\boldsymbol{x}$ is classified positive if $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ is positive,
- and negative if $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ is negative.



(Source: Vincent Lepetit)

Alternative representation

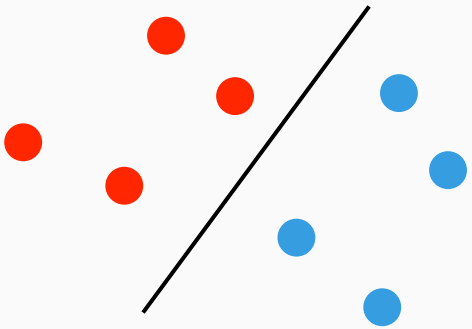


Use the zero-index to encode the bias as a synaptic weight.

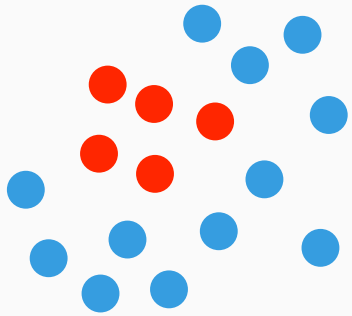
Simplifies algorithms as all parameters can now be processed in the same way.

Perceptrons book (Minsky and Papert, 1969)

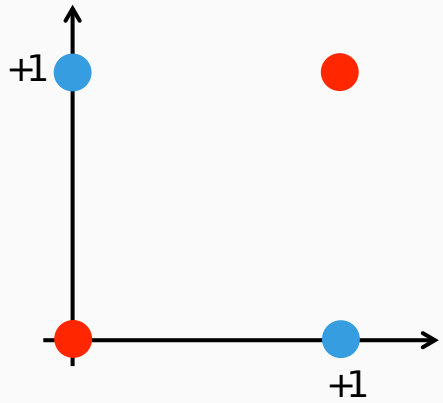
A perceptron can only classify data points that are linearly separable:



Linearly separable



Nonlinearly separable



The xor function

Seen by many as a justification to stop research on perceptrons.

(Source: Vincent Lepetit)

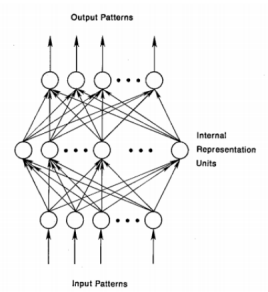
Artificial neural network

Artificial neural network

1958 Perceptron

1969
Perceptrons
book

~1980
Multilayer
network



1989
Universal
Approximation
Theorem

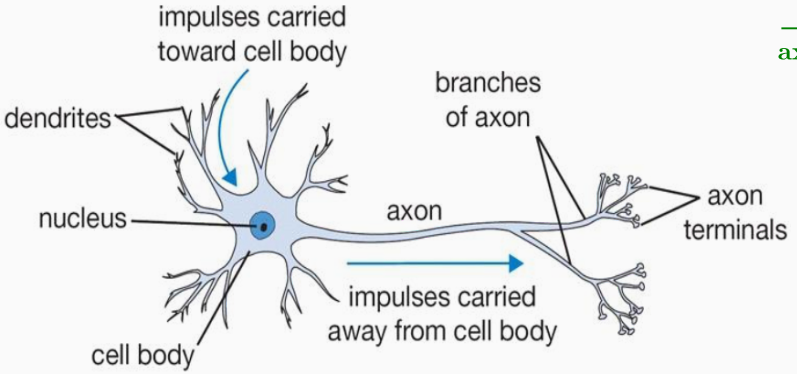
Perceptron criticized

(Source: Lucas Masuch & Vincent Lepetit)

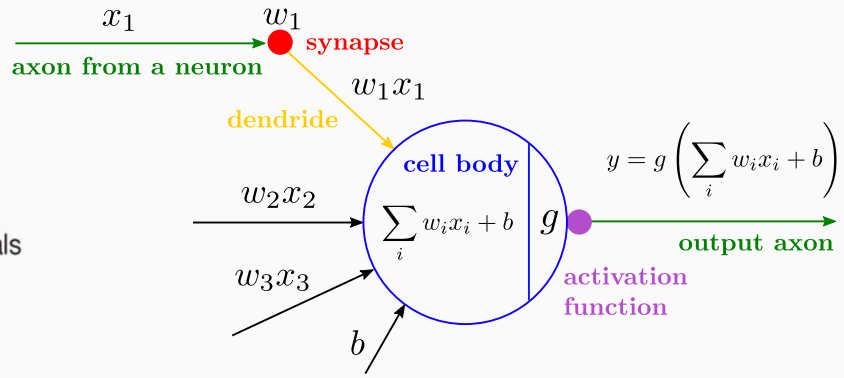
Artificial neural network

- Supervised learning method initially inspired by the behavior of the human brain.
- Consists of the inter-connection of several small units (just like in the human brain).
- Introduced in the late 50s, very popular in the 90s, reappeared in the 2010s with deep learning.
- Also referred to as Multi-Layer Perceptron (MLP).
- Historically used after feature extraction.

Artificial neuron (McCulloch & Pitts, 1943)



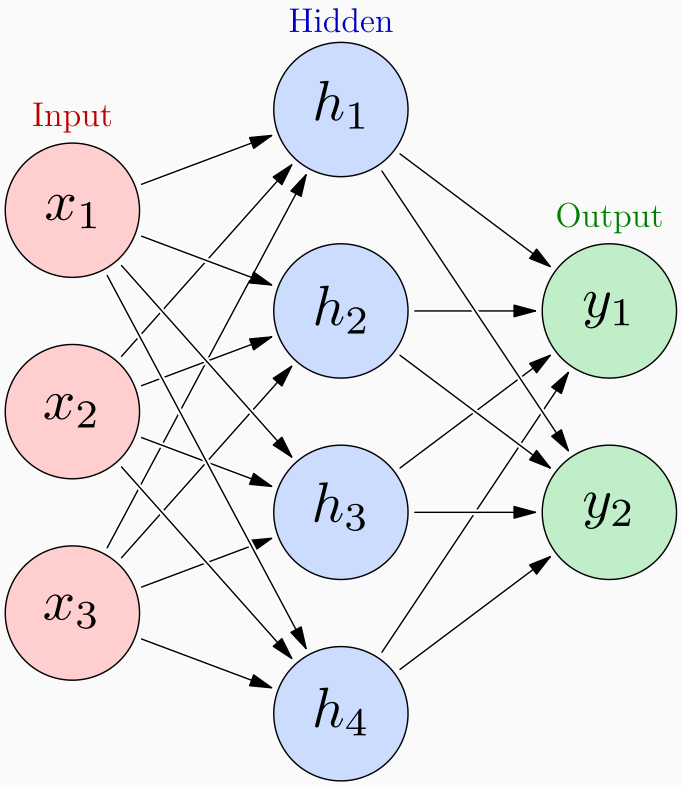
Biological neuron



Artificial neuron

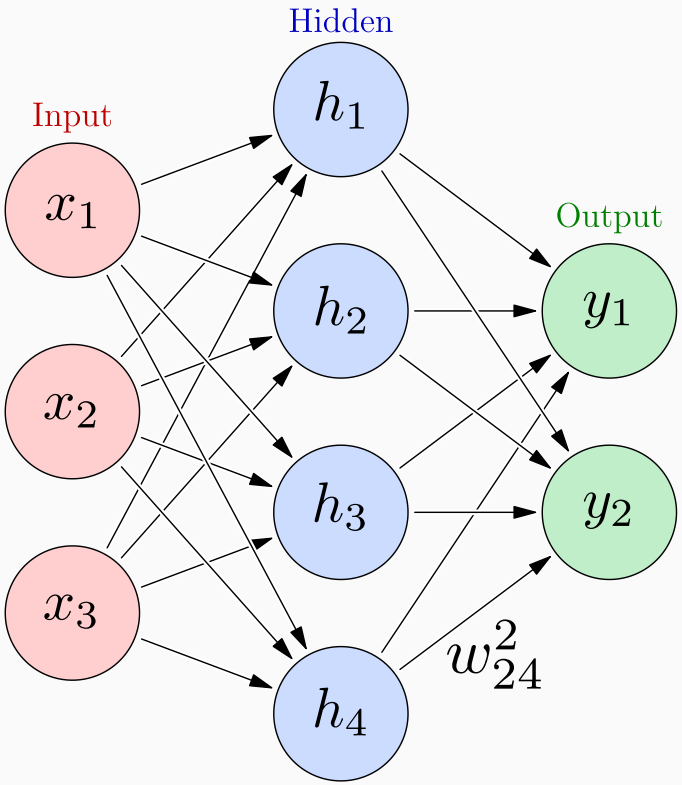
- An artificial neuron contains several incoming weighted connections, an outgoing connection and has a nonlinear activation function g.
- Neurons are trained to filter and detect specific features or patterns (e.g. edge, nose) by receiving weighted input, transforming it with the activation function and passing it to the outgoing connections.
- Unlike the perceptron, can be used for regression (with proper choice of g).

Artificial neural network / Multilayer perceptron / NeuralNet



- Inter-connection of several artificial neurons (also called nodes or units).
- Each level in the graph is called a layer:
  - Input layer,
  - Hidden layer(s),
  - Output layer.
- Each neuron in the hidden layers acts as a classifier / feature detector.
- Feedforward NN (no cycle)
  - first and simplest type of NN,
  - information moves in one direction.
- Recurrent NN (with cycle)
  - used for time sequences,
  - such as speech-recognition.

Artificial neural network / Multilayer perceptron / NeuralNet



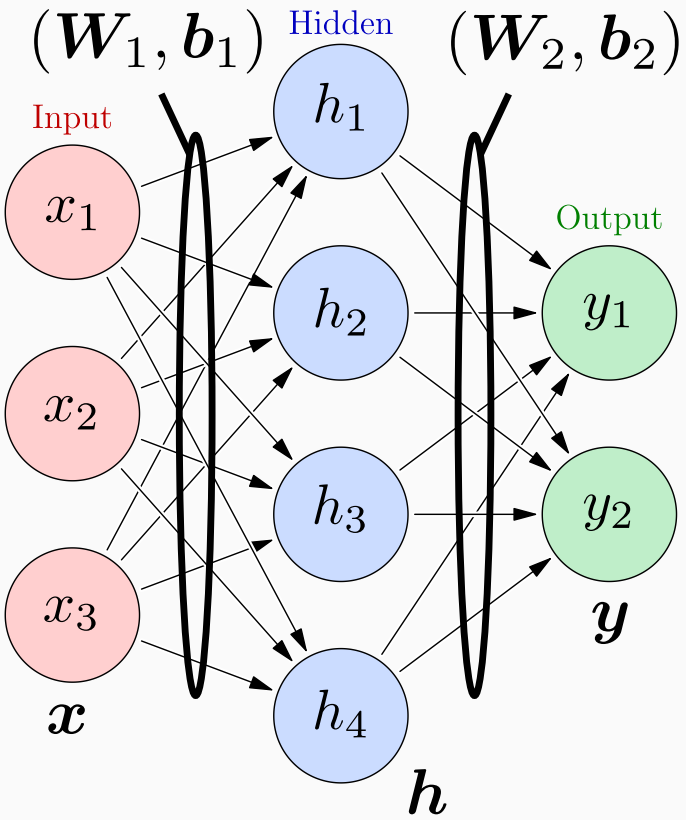
$$h_1 = g_1\left(w_{11}^1 x_1 + w_{12}^1 x_2 + w_{13}^1 x_3 + b_1^1\right)$$
$$h_2 = g_1\left(w_{21}^1 x_1 + w_{22}^1 x_2 + w_{23}^1 x_3 + b_2^1\right)$$
$$h_3 = g_1\left(w_{31}^1 x_1 + w_{32}^1 x_2 + w_{33}^1 x_3 + b_3^1\right)$$
$$h_4 = g_1\left(w_{41}^1 x_1 + w_{42}^1 x_2 + w_{43}^1 x_3 + b_4^1\right)$$

$$y_1 = g_2\left(w_{11}^2 h_1 + w_{12}^2 h_2 + w_{13}^2 h_3 + w_{14}^2 h_4 + b_1^2\right)$$
$$y_2 = g_2\left(w_{21}^2 h_1 + w_{22}^2 h_2 + w_{23}^2 h_3 + w_{24}^2 h_4 + b_2^2\right)$$

w_{ij}^k synaptic weight between previous node j and next node i at layer k.

g_k are any activation function applied to each coefficient of its input vector.

Artificial neural network / Multilayer perceptron / NeuralNet



$$h_1 = g_1\left(w^1_{11}x_1 + w^1_{12}x_2 + w^1_{13}x_3 + b^1_1\right)$$
$$h_2 = g_1\left(w^1_{21}x_1 + w^1_{22}x_2 + w^1_{23}x_3 + b^1_2\right)$$
$$h_3 = g_1\left(w^1_{31}x_1 + w^1_{32}x_2 + w^1_{33}x_3 + b^1_3\right)$$
$$h_4 = g_1\left(w^1_{41}x_1 + w^1_{42}x_2 + w^1_{43}x_3 + b^1_4\right)$$
$$\boldsymbol{h} = g_1\left(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1\right)$$

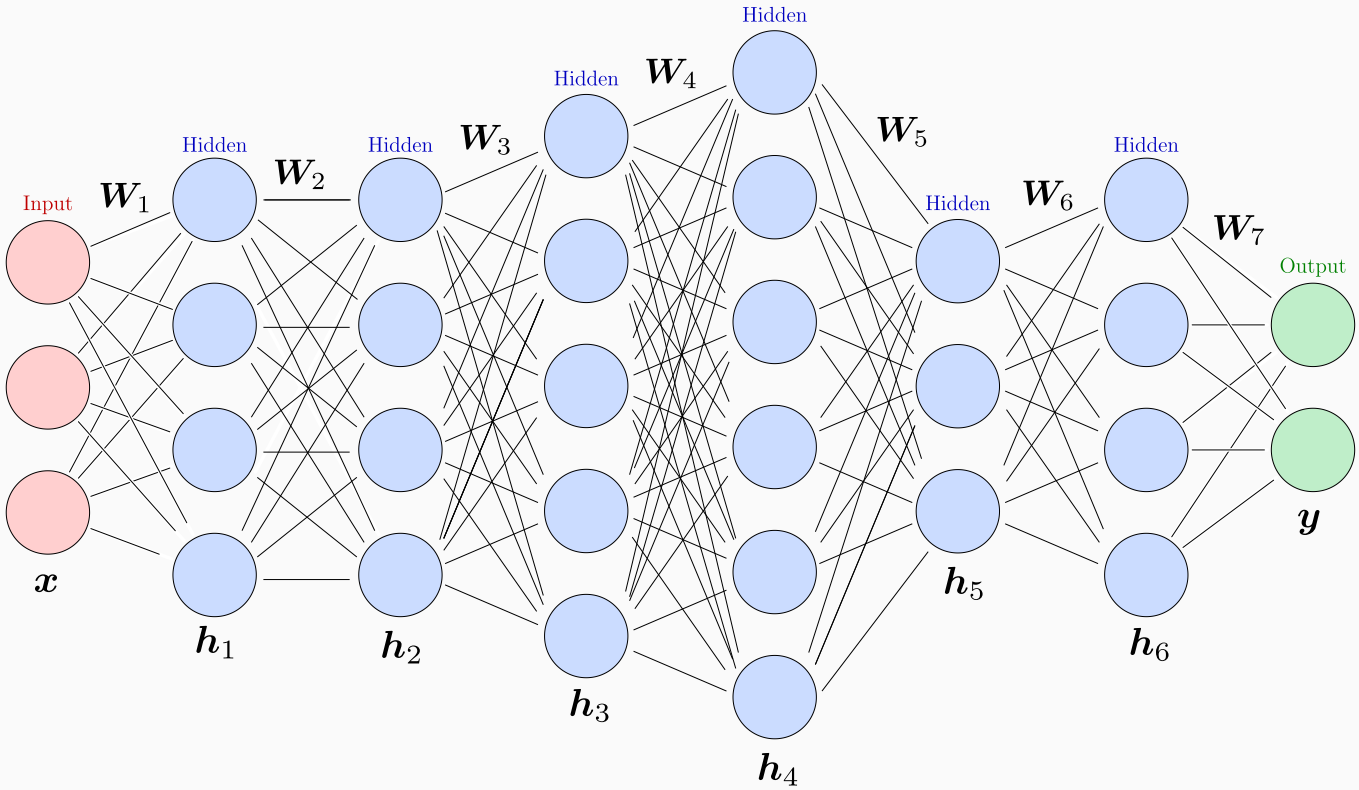
$$y_1 = g_2\left(w^2_{11}h_1 + w^2_{12}h_2 + w^2_{13}h_3 + w^2_{14}h_4 + b^2_1\right)$$
$$y_2 = g_2\left(w^2_{21}h_1 + w^2_{22}h_2 + w^2_{23}h_3 + w^2_{24}h_4 + b^2_2\right)$$
$$\boldsymbol{y} = g_2\left(\boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2\right)$$

w^k_{ij} synaptic weight between previous node j and next node i at layer k.

g_k are any activation function applied to each coefficient of its input vector.

The matrices $\boldsymbol{W_k}$ and biases $\boldsymbol{b}_k$ are learned from labeled training data.

Artificial neural network / Multilayer perceptron



It can have 1 hidden layer only (shallow network),
It can have more than 1 hidden layer (deep network),
each layer may have a different size, and
hidden and output layers often have different activation functions.

Artificial neural network / Multilayer perceptron

- As for the perceptron, the biases can be integrated into the weights:

$$\boldsymbol{W}_k \boldsymbol{h}_{k-1} + \boldsymbol{b}_k = \underbrace{\begin{pmatrix} \boldsymbol{b}_k & \boldsymbol{W}_k \end{pmatrix}}_{\tilde{\boldsymbol{W}}_k} \underbrace{\begin{pmatrix} 1 \\ \boldsymbol{h}_{k-1} \end{pmatrix}}_{\tilde{\boldsymbol{h}}_{k-1}} = \tilde{\boldsymbol{W}}_k \tilde{\boldsymbol{h}}_{k-1}$$

- A neural network with L layers is a function of $\boldsymbol{x}$ parameterized by $\tilde{\boldsymbol{W}}$:

$$\boldsymbol{y} = f(\boldsymbol{x}; \tilde{\boldsymbol{W}}) \quad \text{where} \quad \tilde{\boldsymbol{W}} = (\tilde{\boldsymbol{W}}_1, \tilde{\boldsymbol{W}}_2, \ldots, \tilde{\boldsymbol{W}}_L)^T$$

- It can be defined recursively as

$$\boldsymbol{y} = f(\boldsymbol{x}; \tilde{\boldsymbol{W}}) = \boldsymbol{h}_L, \quad \boldsymbol{h}_k = g_k \left(\tilde{\boldsymbol{W}}_k \tilde{\boldsymbol{h}}_{k-1} \right) \quad \text{and} \quad \boldsymbol{h}_0 = \boldsymbol{x}$$

- For simplicity, $\tilde{\boldsymbol{W}}$ will be denoted $\boldsymbol{W}$ (when no possible confusions).

Activation functions

Linear units: $g(a) = a$

$$\begin{aligned}
\boldsymbol{y} &= \boldsymbol{W}_L \boldsymbol{h}_{L-1} + \boldsymbol{b}_L \\
\boldsymbol{h}_{L-1} &= \boldsymbol{W}_{L-1} \boldsymbol{h}_{L-2} + \boldsymbol{b}_{L-1} \\
\hline
\boldsymbol{y} &= \boldsymbol{W}_L \boldsymbol{W}_{L-1} \boldsymbol{h}_{L-2} + \boldsymbol{W}_L \boldsymbol{b}_{L-1} + \boldsymbol{b}_L \\
\hline
\boldsymbol{y} &= \boldsymbol{W}_L \dots \boldsymbol{W}_1 \boldsymbol{x} + \sum_{k=1}^{L-1} \boldsymbol{W}_L \dots \boldsymbol{W}_{k+1} \boldsymbol{b}_k + \boldsymbol{b}_L
\end{aligned}$$

We can always find an equivalent network without hidden units, because compositions of affine functions are affine.

In general, non-linearity is needed to learn complex (non-linear) representations of data, otherwise the NN would be just a linear function. Otherwise, back to the problem of nonlinearly separable datasets.

Activation functions

Threshold units: for instance the sign function

$$g(a) = \begin{cases} -1 & \text{if} \quad a < 0 \\ +1 & \text{otherwise.} \end{cases}$$

or ReLu activation function

$$g(a) = \max(0, a) = \begin{cases} 0 & \text{if} \quad a < 0 \\ 1 & \text{otherwise.} \end{cases}$$

Discontinuities in the hidden layers
make the optimization really difficult.

We prefer functions that are continuous and differentiable.

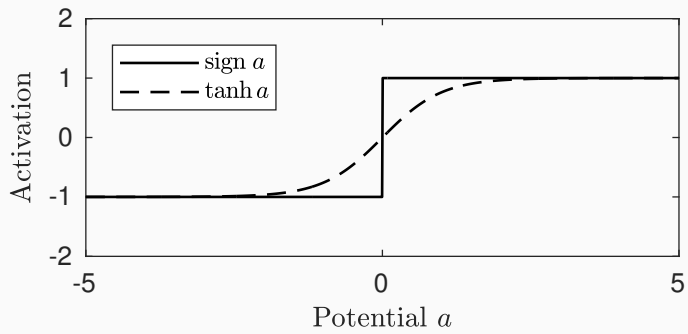
Activation functions

Sigmoidal units: for instance the hyperbolic tangent function

$$g(a) = \tanh a = \frac{e^a - e^{-a}}{e^a + e^{-a}} \in [-1, 1]$$

or the logistic sigmoid function

$$g(a) = \frac{1}{1 + e^{-a}} \in [0, 1]$$



- In fact equivalent by linear transformations :

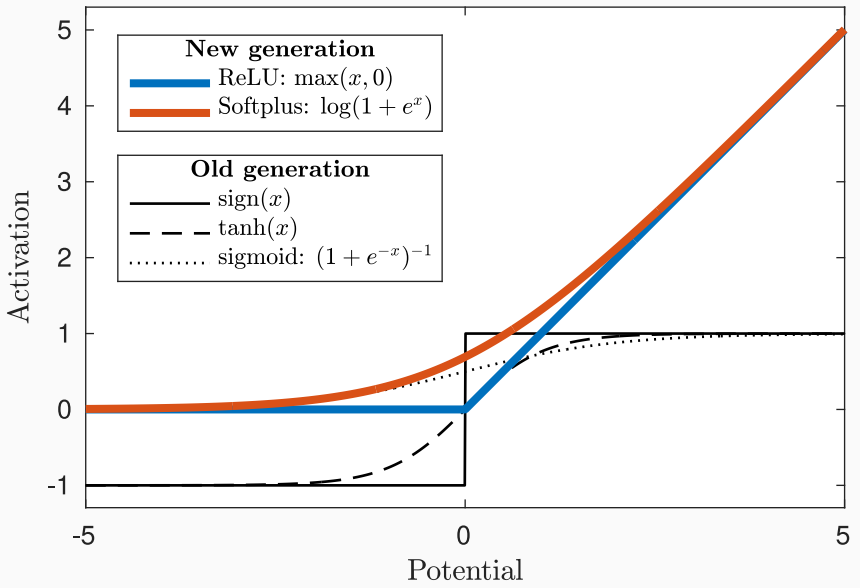
$$\tanh(a/2) = 2\text{logistic}(a) - 1$$

- Differentiable approximations of the sign and step functions, respectively.
- Act as threshold units for large values of $|a|$ and as linear for small values.

Activation functions

"Modern" units:

$$\underbrace{g(a) = \max(a, 0)}_{\text{ReLU}} \quad \text{or} \quad \underbrace{g(a) = \log(1 + e^a)}_{\text{Softplus}}$$



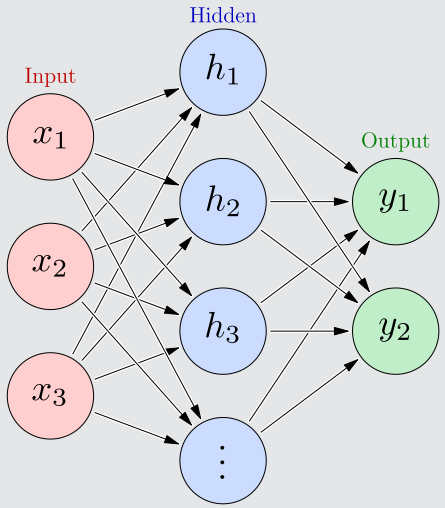
Most neural networks use ReLU (Rectifier linear unit) – $\max(a, 0)$ – nowadays for hidden layers, since it trains much faster, is more expressive than logistic function and prevents the gradient vanishing problem.

(Source: Lucas Masuch)

Universal Approximation Theorem

(Hornik et al, 1989; Cybenko, 1989)

Any continuous function $f : K \subset \mathbb{R}^N \to \mathbb{R}^K$ can be uniformly approximated by a feedforward shallow network (*i.e.*, with 1-hidden layer only) with a sufficient number of neurons in the hidden layer.

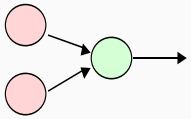


$$\boldsymbol{h} = g(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1)$$
$$\boldsymbol{y} = \boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2$$

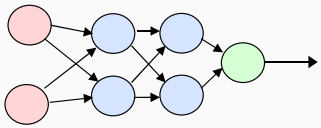
- Works if and only if g is not polynomial (and thus non linear).
- The theorem does not say how large the network needs to be.
- No guarantee that the training algorithm will be able to train the network.

Machine learning – ANN

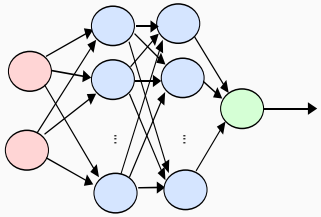
1 neuron

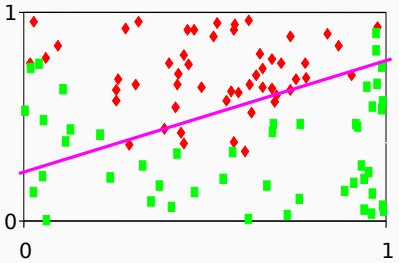


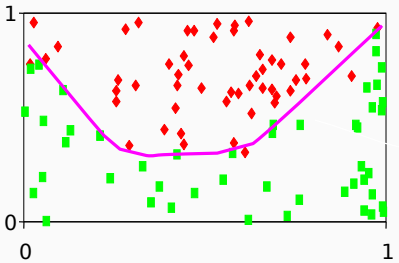
2+2+1 neurons

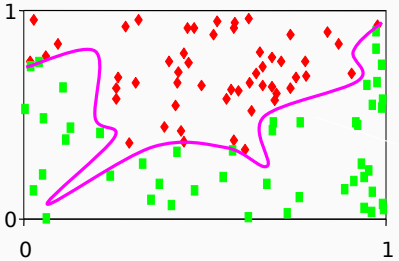


10+10+1 neurons









Complexity/capacity of the network

$\Rightarrow$

Trade-off between generalization and overfitting.

Learning from examples

3 main ingredients

1. Training set / examples:

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$$

2. Machine or model:

$$\boldsymbol{x} \rightarrow \underbrace{f(\boldsymbol{x}; \theta)}_{\text{function / algorithm}} \rightarrow \underbrace{\boldsymbol{y}}_{\text{prediction}}$$

θ: parameters of the model

3. Loss, cost, objective function / energy:

$$\underset{\theta}{\operatorname{argmin}}\ E(\theta; \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$$

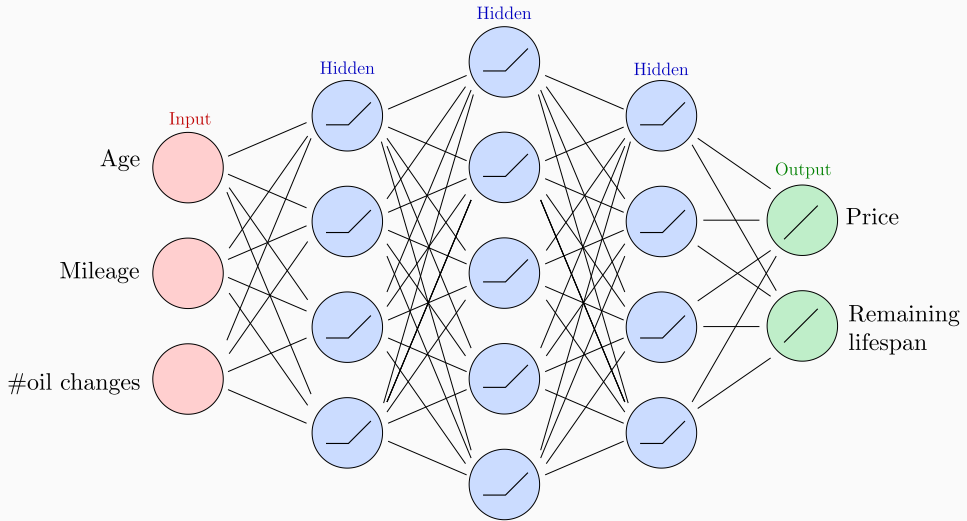
Approximation – Least square regression

- **Goal:** Predict a real multivariate function.
- **How:** estimate the coefficients $\boldsymbol{\theta} = (\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2, \ldots)$ of $\boldsymbol{d}_i = f(\boldsymbol{x}_i; \boldsymbol{W}), 1 \leqslant i \leqslant d$?
  from labeled training examples where labels are real vectors:

$$\mathcal{T} = \{(\boldsymbol{x}^i, \boldsymbol{d}^i)\}_{i=1..N}$$

i-th training example — desired output for sample i — number of training samples

- **Typical architecture:**



Approximation – Least square regression

- **Loss:** As for the polynomial curve fitting, it is standard to consider the sum of square errors (assumption of Gaussian distributed errors)

$$E(\boldsymbol{W}) = \sum_{i=1}^{N} \|\boldsymbol{y}^i - \boldsymbol{d}^i\|_2^2 = \sum_{i=1}^{N} \|f(\boldsymbol{x}^i; \boldsymbol{W}) - \boldsymbol{d}^i\|_2^2$$

  and look for $\boldsymbol{W}^*$ such that $\nabla E(\boldsymbol{W}^*) = 0$.

Approximation – Least square regression

- **Loss:** As for the polynomial curve fitting, it is standard to consider the sum of square errors (assumption of Gaussian distributed errors)

$$E(\boldsymbol{W}) = \sum_{i=1}^{N} \|\boldsymbol{y}^i - \boldsymbol{d}^i\|_2^2 = \sum_{i=1}^{N} \|f(\boldsymbol{x}^i; \boldsymbol{W}) - \boldsymbol{d}^i\|_2^2$$

  and look for $\boldsymbol{W}^*$ such that $\nabla E(\boldsymbol{W}^*) = 0$.

- **Solution:** Provided the network has enough flexibility and the size of the training set grows to infinity

$$\boldsymbol{y}^\star = f(\boldsymbol{x}; \boldsymbol{W}^\star) = \underbrace{\mathbb{E}[\boldsymbol{d}|\boldsymbol{x}] = \int \boldsymbol{d} p(\boldsymbol{d}|\boldsymbol{x})\, d\boldsymbol{d}}_{\text{posterior mean}}$$

Approximation – Least square regression

- **Loss:** As for the polynomial curve fitting, it is standard to consider the sum of square errors (assumption of Gaussian distributed errors)

$$E(\boldsymbol{W}) = \sum_{i=1}^{N} \|\boldsymbol{y}^i - \boldsymbol{d}^i\|_2^2 = \sum_{i=1}^{N} \|f(\boldsymbol{x}^i; \boldsymbol{W}) - \boldsymbol{d}^i\|_2^2$$

  and look for $\boldsymbol{W}^*$ such that $\nabla E(\boldsymbol{W}^*) = 0$.

- **Solution:** Provided the network has enough flexibility and the size of the training set grows to infinity

$$\boldsymbol{y}^\star = f(\boldsymbol{x}; \boldsymbol{W}^\star) = \underbrace{\mathbb{E}[\boldsymbol{d}|\boldsymbol{x}] = \int \boldsymbol{d} p(\boldsymbol{d}|\boldsymbol{x})\, d\boldsymbol{d}}_{\text{posterior mean}}$$

- **Problem:** Inefficient for image classification. (Cf TD for e new cost function).

Training

We want to have $f(x, \boldsymbol{\theta})$ such that for all $(\boldsymbol{x}, \boldsymbol{d}) \in \mathcal{T}$,

$$f(x, \boldsymbol{\theta}) \approx d$$

by minimizing a loss function $E(\boldsymbol{\theta})$.

Algorithm: (Stochastic) Gradient Descent (SGD)

- Initialize $\boldsymbol{\theta} = (\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2, \ldots)$ randomly
- Repeat until convergence
  - For all $(\boldsymbol{x}, d) \in \mathcal{T}$
    - Compute: $y = f(x, \boldsymbol{\theta})$
    - Update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \gamma \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta})$

An iterative algorithm trying to find a minimum of a real function.

Gradient descent

- Let F be a real function, coercive, and twice-differentiable such that:

$$\| \underbrace{\nabla^2 F(x)}_{\text{Hessian matrix of } F} \|_2 \leqslant L, \quad \text{for some } L > 0.$$

- Then, whatever the initialization x^0, if $0 < \gamma < 2/L$, the sequence

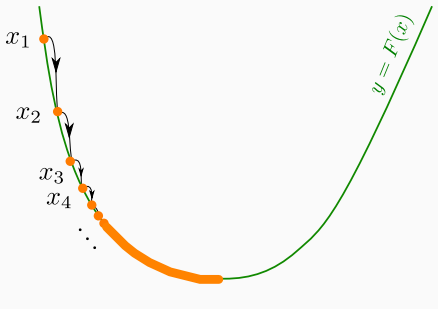
$$x^{(n+1)} = x^{(n)} \underbrace{- \gamma \nabla F(x^{(n)})}_{\text{direction of greatest descent}},$$

converges to a stationary point $x^\star$ (*i.e.*, it cancels the gradient)

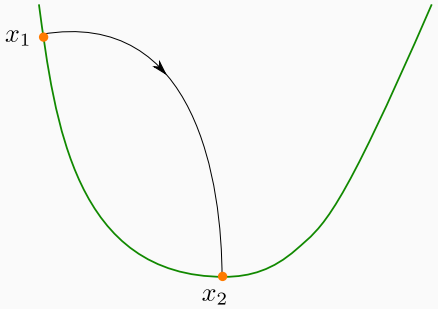
$$\nabla F(x^\star) = 0 \ .$$

- The parameter γ is called the step size (or **learning rate** in ML field).
- A too small step size γ leads to slow convergence.

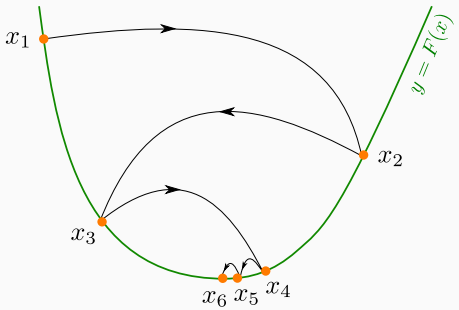
One dimension



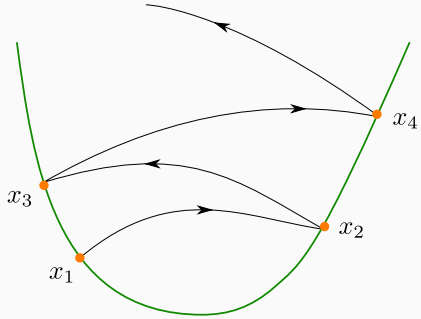
Small step size
Slow convergence



Good step size
Fast convergence

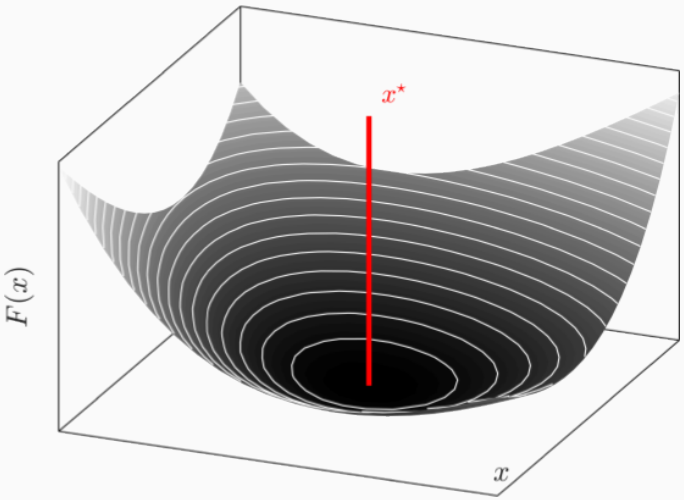


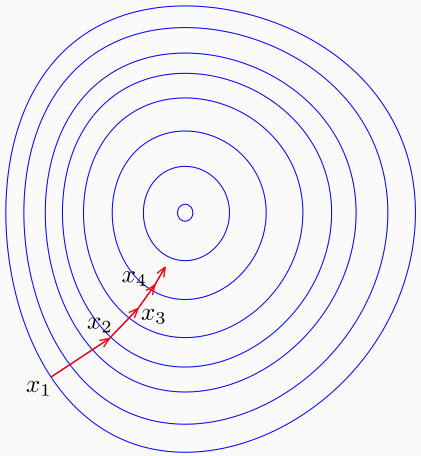
Large step size
Slow convergence

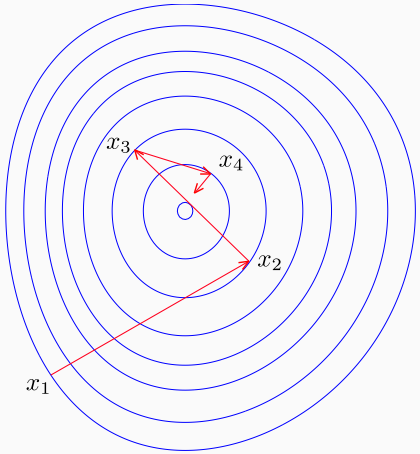


Too large step size
Divergence

Two dimensions

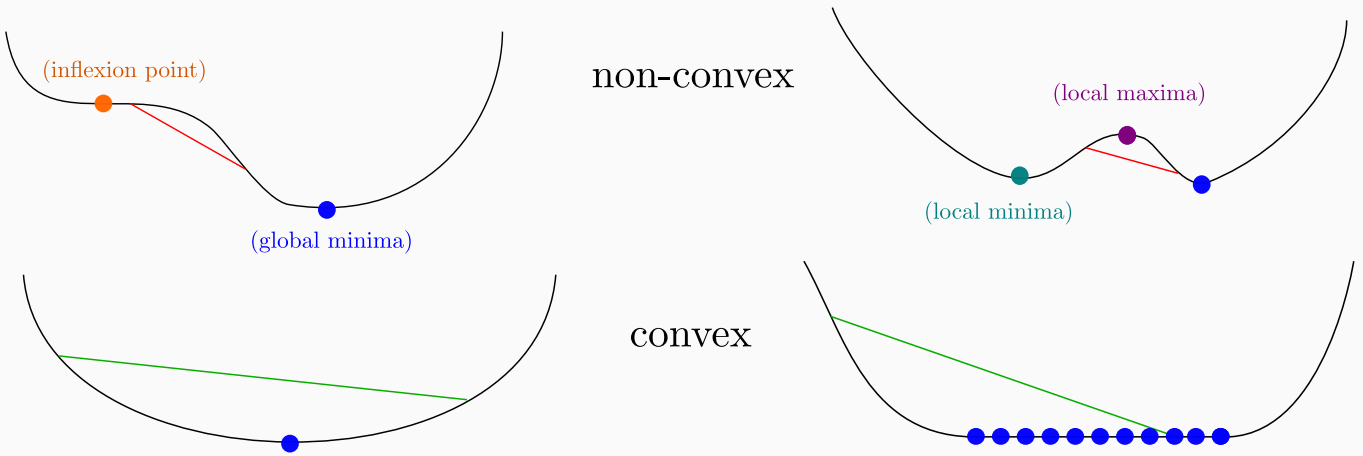






Non convexity in machine learning

But for neural network the cost is **not convex**...



Learning from examples

1. Training set / examples:

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$$

2. Machine or model:

$$\boldsymbol{x} \rightarrow \underbrace{f(\boldsymbol{x}; \theta)}_{\text{function / algorithm}} \rightarrow \underbrace{\boldsymbol{y}}_{\text{prediction}}$$

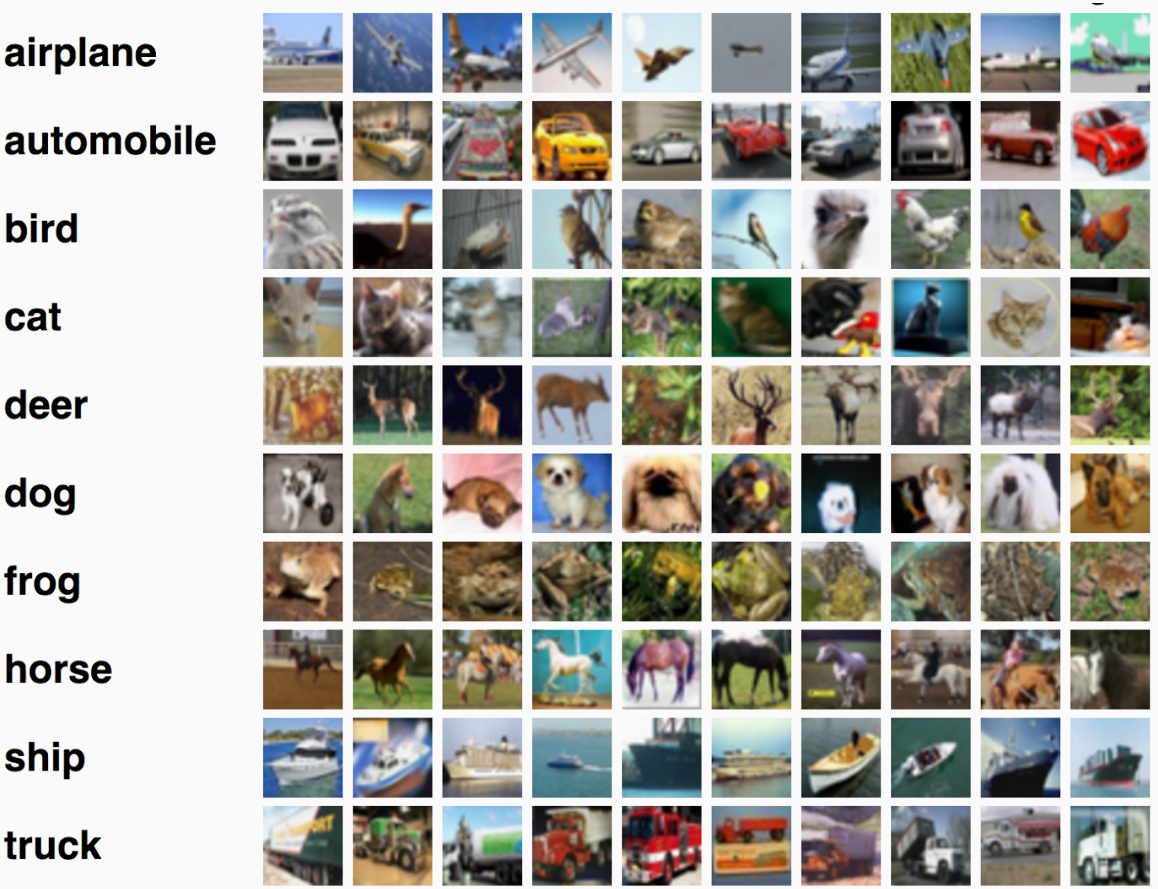
θ: parameters of the model

3. Loss, cost, objective function / energy:

$$\underset{\theta}{\operatorname{argmin}}\ E(\theta; \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$$

4. Minimization of E via SGD.
5. Control of $f(\cdot, \theta)$ with a test dataset.

Classification example



Questions?

Next class: Backpropagation

Slides from Charles Deledalle

Sources, images courtesy and acknowledgment

K. Chatfield
P. Gallinari,
C. Hazırbaş
A. Horodniceanu
Y. LeCun
V. Lepetit
L. Masuch
A. Ng
M. Ranzato

Focus on classification: the multivariate logistic regression

Notations

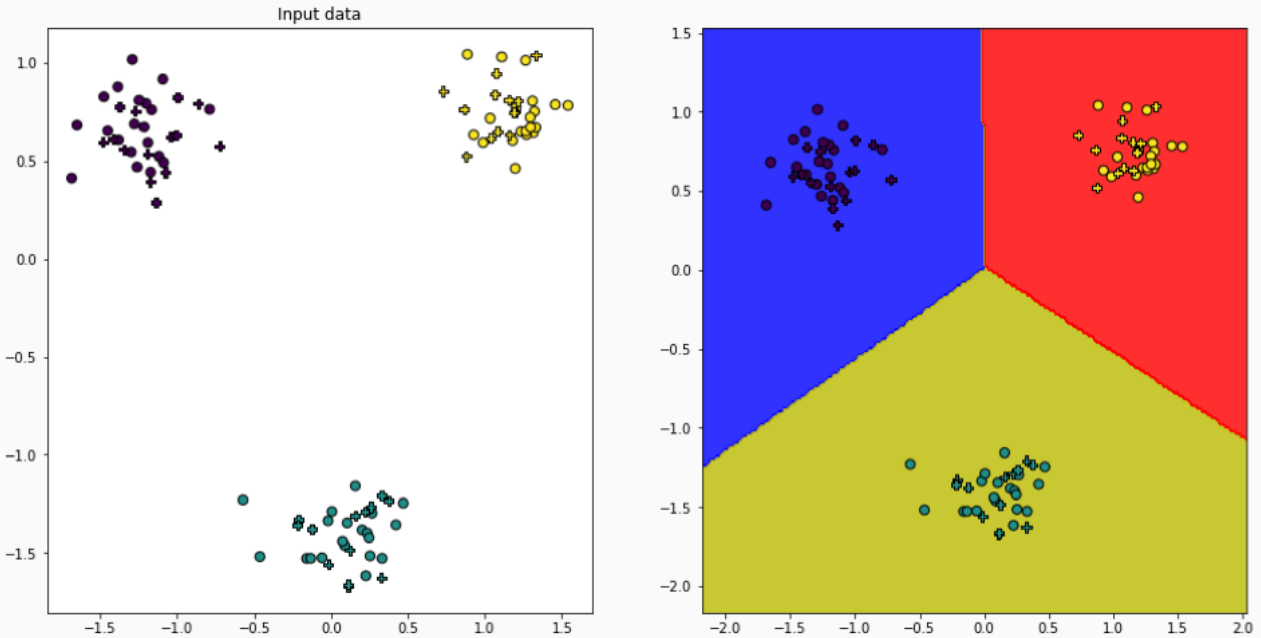
- **Goal:** Classify an object $\boldsymbol{x}$ into one among K classes $C_1, \ldots, C_K$.
- Training set: $\mathcal{T} = \{(\boldsymbol{x}_n, t_n),\ n = 1, \ldots, N\}$, $t_n \in \{1, \ldots, K\}$ encodes the class of $\boldsymbol{x}_n$.
-
  - Class 1: $t_n = 1$ if $\boldsymbol{x}_n \in C_1$,
  - Class 2: $t_n = 2$ if $\boldsymbol{x}_n \in C_2$,
  - $\ldots$
  - Class K: $t_n = K$ if $\boldsymbol{x}_n \in C_K$,,

Feature transform

- We apply a feature transform $\varphi : \mathbb{R}^p \to \mathbb{R}^D$ to each $\boldsymbol{x_n}$:

$$\varphi_n = \varphi(\boldsymbol{x_n}), \quad n = 1, \ldots, N.$$

- Depending on context it allows to increase ($D > p$) or decrease ($D < p$) the dimension in a way to favor class discrimination (e.g. PCA...).
- This is a non linear map that should make the classes linearly separable.



Multiclass classification – Multivariate logistic regression

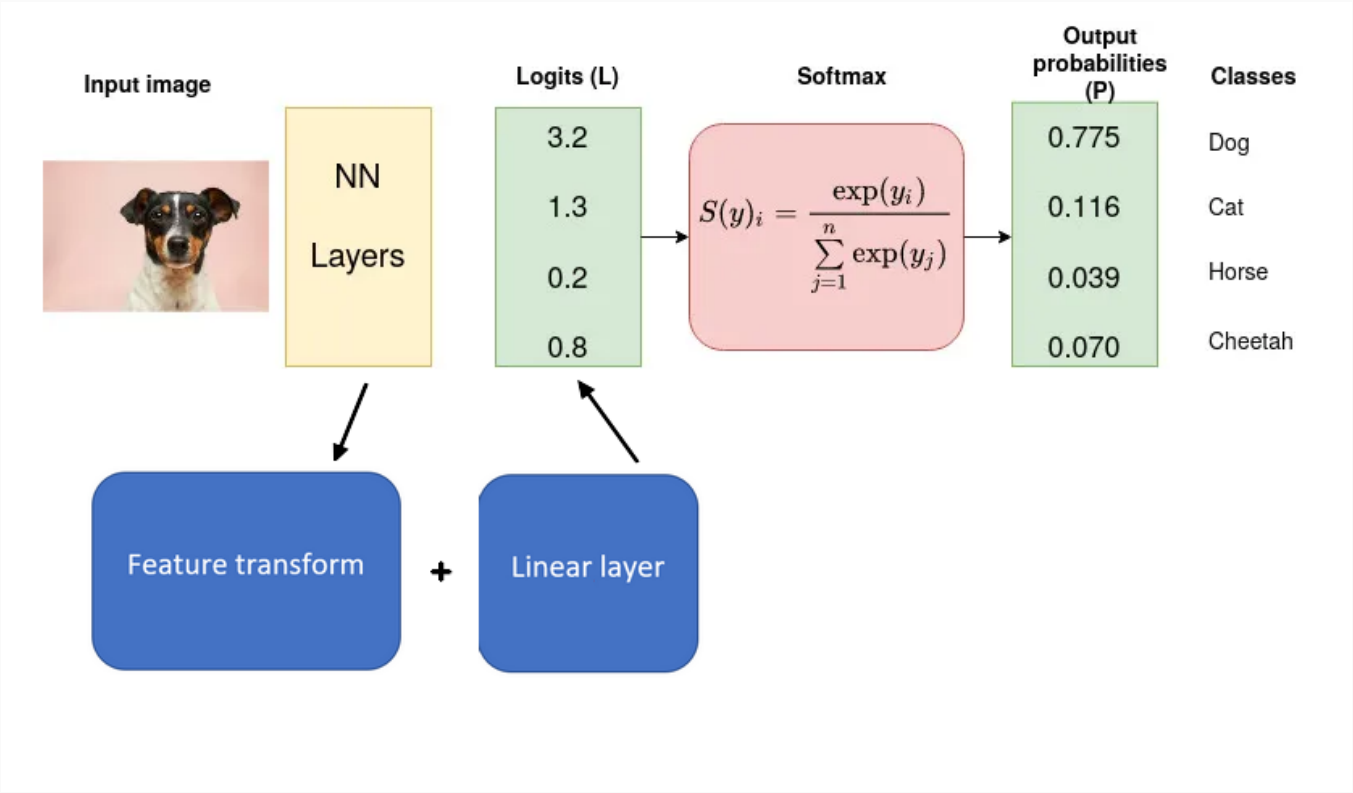
(aka, multinomial classification)

- **Goal:** Classify an object $\boldsymbol{x}$ into one among K classes $C_1, \ldots, C_K$.
- **How:** Estimate the coefficients $\boldsymbol{W}$ of a multivariate function

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{W}) \in [0,1]^K \quad \text{s.t.} \quad \sum_{k=1}^{K} y_k = 1.$$

  from training examples $\mathcal{T} = \{(\boldsymbol{x}^i, \boldsymbol{d}^i)\}$ where $\boldsymbol{d}^i$ is the class of $\boldsymbol{x}^i$
  - Class 1: $\boldsymbol{d}^i = 1$ if $\boldsymbol{x}^i \in C_1$
  - Class 2: $\boldsymbol{d}^i = 2$ if $\boldsymbol{x}^i \in C_2$
  - ...
  - Class K: $\boldsymbol{d}^i = K$ if $\boldsymbol{x}^i \in C_K$
- $\boldsymbol{y}_k = f(\boldsymbol{x}; \boldsymbol{W})$ is understood as the probability of $\boldsymbol{x} \in C_k$.

In real-life

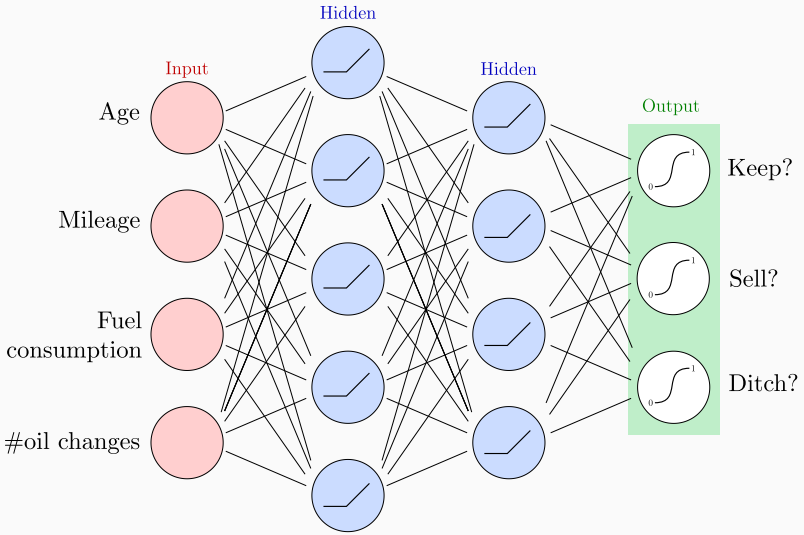


In real-life, all the network have to be trained. The feature transform is trained with its linear separation.

For this session, we will just train the last layer. We suppose that we have a good feature transform to have linearly separable data.

Multiclass classification – Multivariate logistic regression

- **Typical architecture:**



- Hidden layer:

$$\mathrm{ReLU}(a) = \max(a, 0)$$

- Output layer:

$$\mathrm{softmax}(\boldsymbol{a})_k = \frac{\exp(a_k)}{\sum_{\ell=1}^{K} \exp(a_\ell)}$$

- Softmax maps $\mathbb{R}^K$ to the set of probability vectors $\{\boldsymbol{y} \in (0,1)^K,\ \sum_{k=1}^K \boldsymbol{y}_k = 1\}$.
- Smooth version of winner-takes-all activation model (maxout).
- The final decision function is winner-takes-all

$$\operatorname{argmax}_k \mathrm{softmax}(\boldsymbol{a}) = \operatorname{argmax}_k \boldsymbol{a}$$

Multiclass classification – Multivariate logistic regression

- **Loss:** it is standard to consider the **cross-entropy** for K classes (assumption of multinomial distributed data)

$$E(\boldsymbol{W}) = -\sum_{i=1}^{N}\sum_{k=1}^{K} d_k^i \log y_k^i \quad \text{with} \quad \boldsymbol{y}^i = f(\boldsymbol{x}^i; \boldsymbol{W}) = \text{softmax}(\boldsymbol{a}) \in (0,1)^K.$$

$$= -\sum_{i=1}^{N}\left[a_{d^i} - \log\left(\sum_{k=1}^{K}\exp(a_k)\right)\right] \quad \text{with } d^i \text{ the class of } \boldsymbol{x}^i.$$

and look for $\boldsymbol{W}^*$ such that $\nabla E(\boldsymbol{W}^*) = 0$.

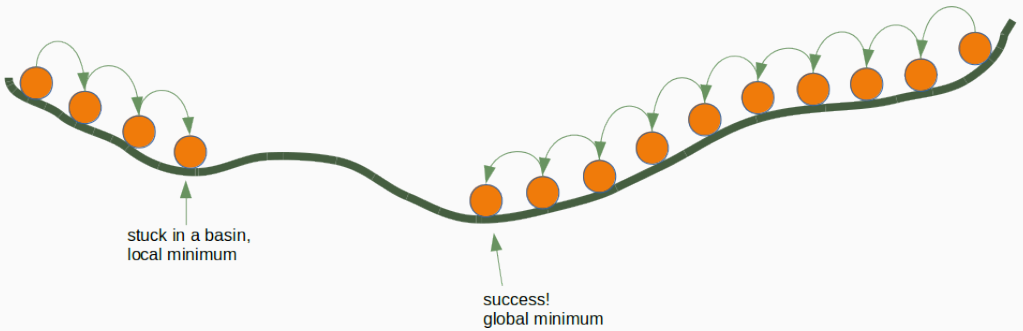
- **Solution:** Provided the network has enough flexibility and the size of the training set grows to infinity

$$y_k^\star = f_k(\boldsymbol{x}; \boldsymbol{W}^\star) = \underbrace{\mathbb{P}(C_k|\boldsymbol{x})}_{\text{posterior probability}}$$

The end

Questions ?

Next course: Backpropagation



Questions?

Next class: Backpropagation

Slides from Charles Deledalle

Sources, images courtesy and acknowledgment

K. Chatfield
P. Gallinari,
C. Hazırbaş
A. Horodniceanu
Y. LeCun
V. Lepetit
L. Masuch
A. Ng
M. Ranzato