Introduction to Deep Learning

Machine Learning basics

Alexandre Allauzen

ESPCI PARIS | PSL

Dauphine | PSL
UNIVERSITÉ PARIS

MILES
Machine Intelligence and Learning Systems

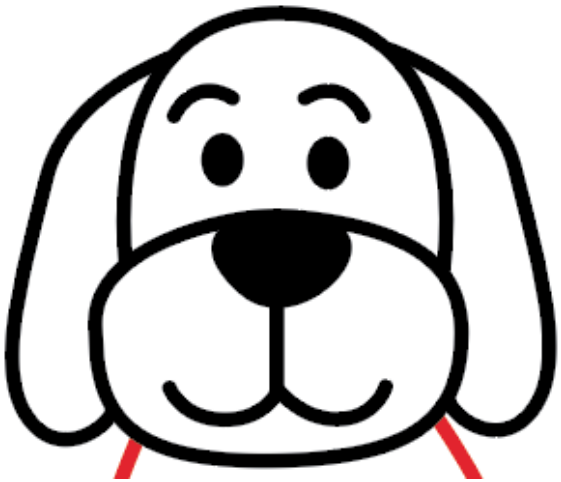
06/01/2020

Outline

Outline

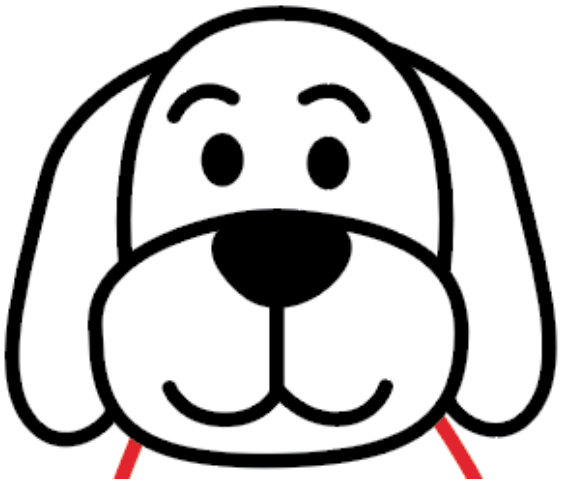
Starter: a “simple” question

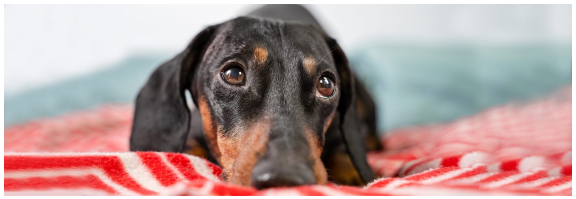
What it is ?



Starter: a “simple” question

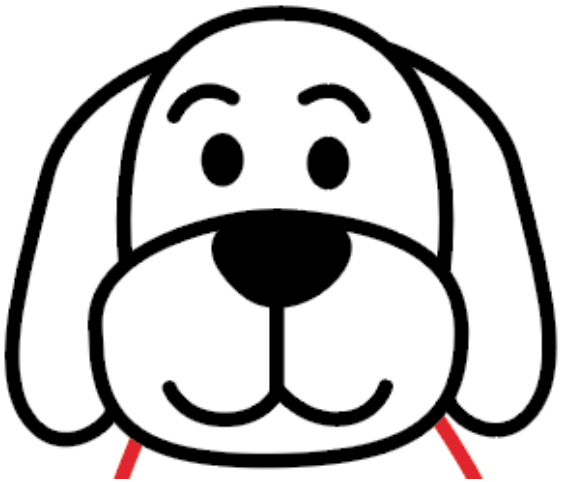
What it is ?

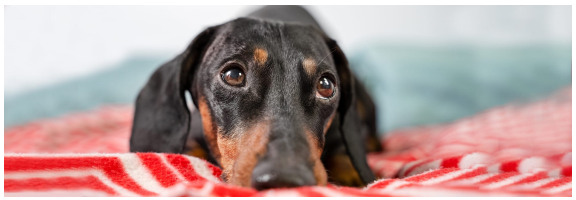




Starter: a “simple” question

What it is ?

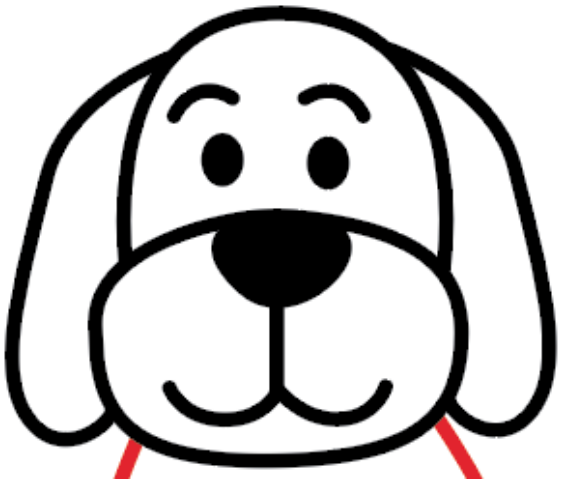


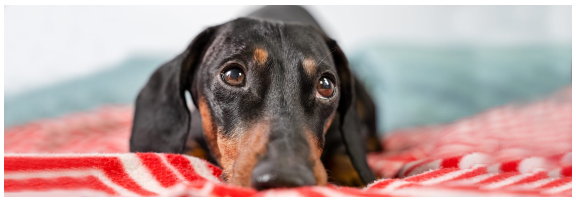


Can you write an algorithm to answer ?

Starter: a "simple" question

What it is ?





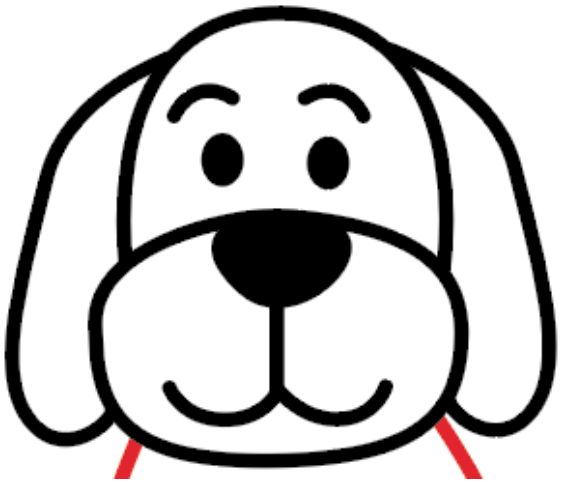
Can you write an algorithm to answer ?

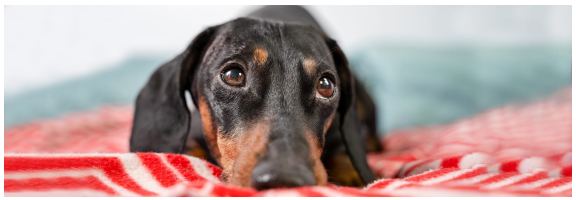
- What is the input space ?
- What do you want to predict ? human generated ? resolution ? real ? outdoor ? the type of dogs ?
- The output space ? (binary, multi-class, regression)

And then ?

Starter: a "simple" question

What it is ?





Can you write an algorithm to answer ?

- What is the input space ?
- What do you want to predict ? human generated ? resolution ? real ? outdoor ? the type of dogs ?
- The output space ? (binary, multi-class, regression)

And then ?

- What are the features ?
- The decision rule ?
- The evaluation criterion ?

Starter: a less "simple" question

Can you write an algorithm to answer ?

- What is the input space ?
- What do you want to predict ?
  a set of tagged bounding boxes
- The output space ?
  multi-class depending on the application

And then ?

Starter: a less "simple" question

Can you write an algorithm to answer ?

- What is the input space ?
- What do you want to predict ?
  a set of tagged bounding boxes
- The output space ?
  multi-class depending on the application

And then ?

- Two tasks: segmentation and classification (joint)
- What are the features ?
- The decision rule ?
- The evaluation criterion ?

Starter: a less "simple" question

Named Entity Recognition (NER)

In 1917, Einstein applied the general theory of relativity to model the large-scale structure of the universe. He was visiting the United States when Adolf Hitler came to power in 1933 and did not go back to Germany, where he had been a professor at the Berlin Academy of Sciences. He settled in the U.S., becoming an American citizen in 1940. On the eve of World War II, he endorsed a letter to President Franklin D. Roosevelt alerting him to the potential development of "extremely powerful bombs of a new type" and recommending that the U.S. begin similar research. This eventually led to what would become the Manhattan Project. Einstein supported defending the Allied forces, but largely denounced using the new discovery of nuclear fission as a weapon. Later, with the British philosopher Bertrand Russell, Einstein signed the Russell-Einstein Manifesto, which highlighted the danger of nuclear weapons. Einstein was affiliated with the Institute for Advanced Study in Princeton, New Jersey, until his death in 1955.

Tag colours:
LOCATION TIME PERSON ORGANIZATION MONEY PERCENT DATE

Can you write an algorithm ?

- The input space ?
- The prediction ?
- The output space ?

Starter: a less "simple" question

Named Entity Recognition (NER)

In 1917, Einstein applied the general theory of relativity to model the large-scale structure of the universe. He was visiting the United States when Adolf Hitler came to power in 1933 and did not go back to Germany, where he had been a professor at the Berlin Academy of Sciences. He settled in the U.S., becoming an American citizen in 1940. On the eve of World War II, he endorsed a letter to President Franklin D. Roosevelt alerting him to the potential development of "extremely powerful bombs of a new type" and recommending that the U.S. begin similar research. This eventually led to what would become the Manhattan Project. Einstein supported defending the Allied forces, but largely denounced using the new discovery of nuclear fission as a weapon. Later, with the British philosopher Bertrand Russell, Einstein signed the Russell-Einstein Manifesto, which highlighted the danger of nuclear weapons. Einstein was affiliated with the Institute for Advanced Study in Princeton, New Jersey, until his death in 1955.

Tag colours:
LOCATION TIME PERSON ORGANIZATION MONEY PERCENT DATE

Can you write an algorithm ?

- The input space ?
- The prediction ?
- The output space ?

Just build lists of Named entities ?

- language is ambiguous (e.g. Tim Cook, 1984, ...)
- language is evolving with new "people", places, domain specific terms, ...
- answer is obvious (most of the time), but hard to explain

The Machine Learning way

How to make a 'computer' do a specific task?

Traditional approach (old AI)

A program is:

- hand-coded
- specific set of instructions to complete the task
- can be explained and proved, "always" gives the correct answer

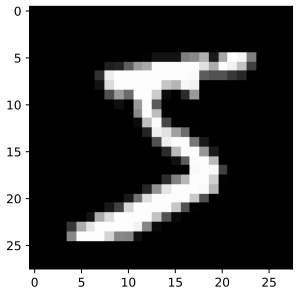
Machine Learning

A program is:

- trained or learnt
- from large amount of annotated data
- algorithm + inductive bias
- it works ... on average

Machine learning: the main tasks

Supervised classification



$$: \quad \boldsymbol{x} \in \mathbb{R}^{784} \longrightarrow c \in \{0, 1, 2, \ldots, 9\}$$

- $\mathcal{D} = (\boldsymbol{x}_{(i)}, c_{(i)})_{i=1}^{N}$
- **Supervised** learning of a **classification** task

Machine learning: the main tasks

Supervised binary classification

my wonderful friend took
me to see this movie
for our anniversary.
it was terrible.

$: \boldsymbol{x} \in \mathbb{R}^D \longrightarrow c \in \{0, 1\}$

- $\mathcal{D} = (\boldsymbol{x}_{(i)}, c_{(i)})_{i=1}^N$
- Supervised learning of a **binary** classification task

Machine learning: the main tasks

Supervised regression

my wonderful friend took me to see this movie for our anniversary. it was terrible.

: $\boldsymbol{x} \in \mathbb{R}^D \longrightarrow y \in \mathbb{R}$ $e.g$ $[0, 1]$

- $\mathcal{D} = (\boldsymbol{x}_{(i)}, \boldsymbol{y}_{(i)})_{i=1}^N$
- Supervised learning of a **regression** task

Machine learning: the main tasks

Unsupervised learning / Clustering

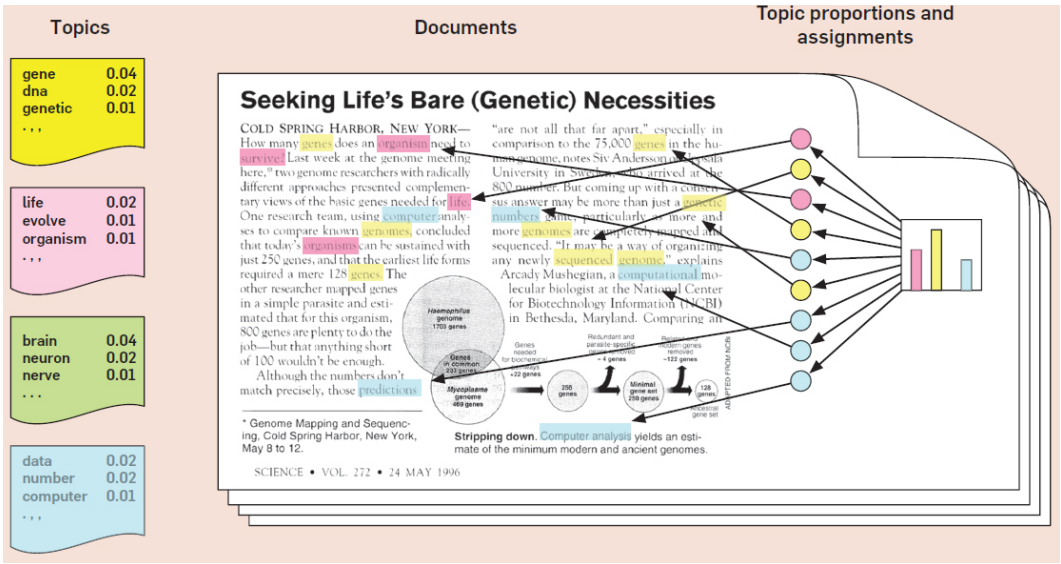
$\Longrightarrow$

$$\boldsymbol{x} \in \mathbb{R}^D \longrightarrow \boldsymbol{z} \in \mathbb{R}^C$$

- A set of observations $\mathcal{D} = (\boldsymbol{x}_{(i)})_{i=1}^N$ must be assigned to a cluster $\rightarrow z$
- The model infer a hidden/latent structure in the $\mathcal{D}$
- The *guidelines*: the structure of the model, the assumptions (distance, similarty between the $\boldsymbol{x}$, ...)
- Dimensionality reduction, data mining, ...

Machine learning: the main tasks

Unsupervised learning



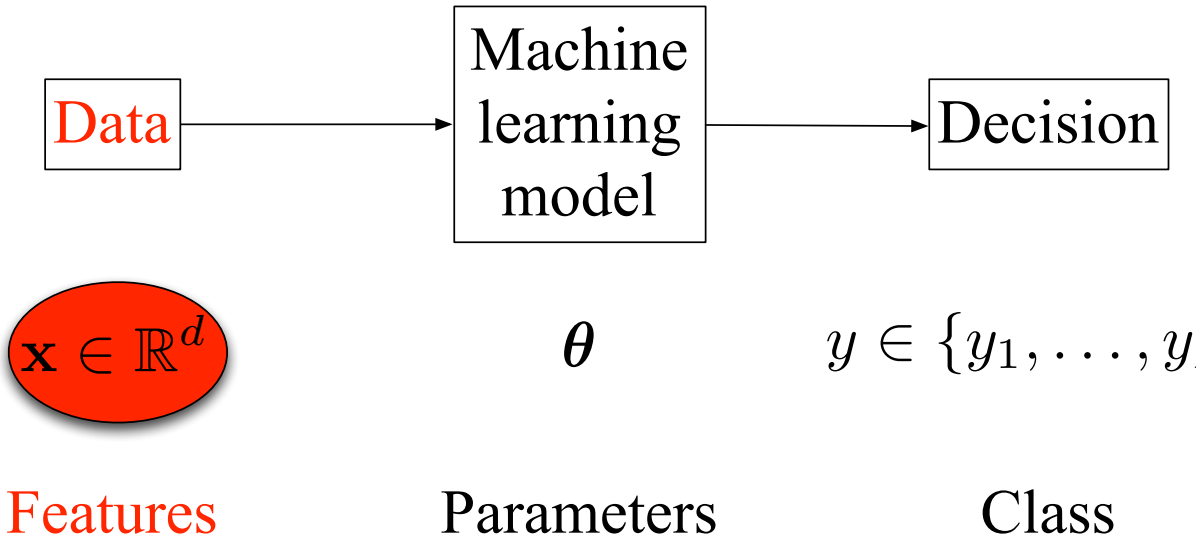
: $\boldsymbol{x}_{(i)} \in \mathbb{R}^D \longrightarrow \boldsymbol{z} \in \mathbb{R}^d$

- $\mathcal{D} = (\boldsymbol{x}_{(i)})_{i=1}^N \longrightarrow (\boldsymbol{z}_{(i)})$, the outcome of the learning algorithm
- **Unsupervised** learning

(K-means, GMM, LDA, ...)

Feature engineering

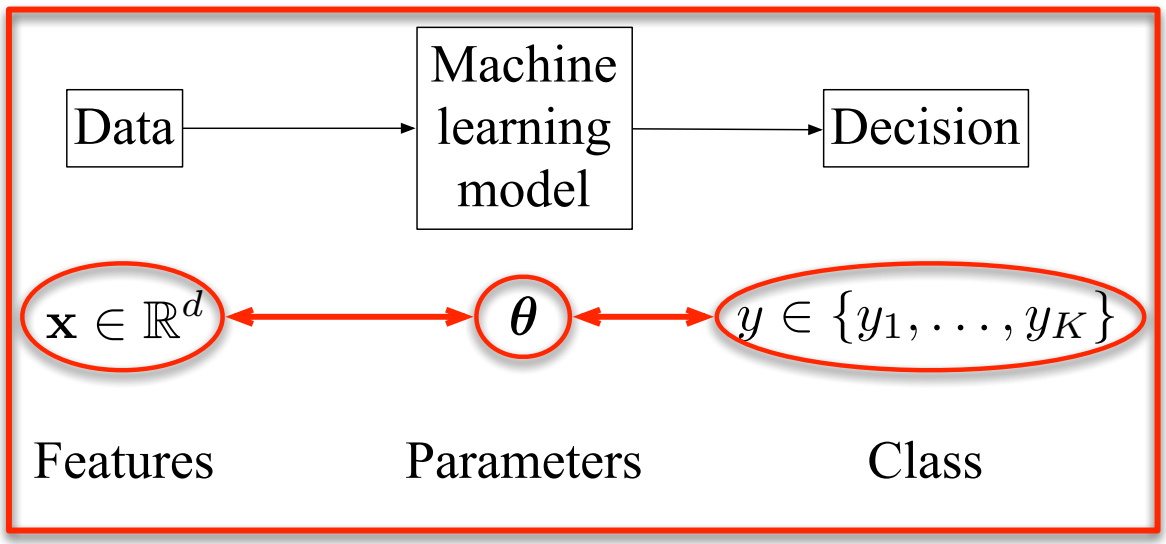
Most current machine learning works well because of human-designed representations and input features



- Time consuming and task/domain dependant
- Features are often both over-specified and incomplete
- Machine learning $\Leftrightarrow$ optimizing parameters to make the best prediction

Representation learning and Deep networks

Representation learning attempts to automatically learn useful features



- Learning a hierarchical and abstract representation
- That can be shared among tasks
- Almost all data is unlabeled ⇒ unsupervised learning

Neural Networks

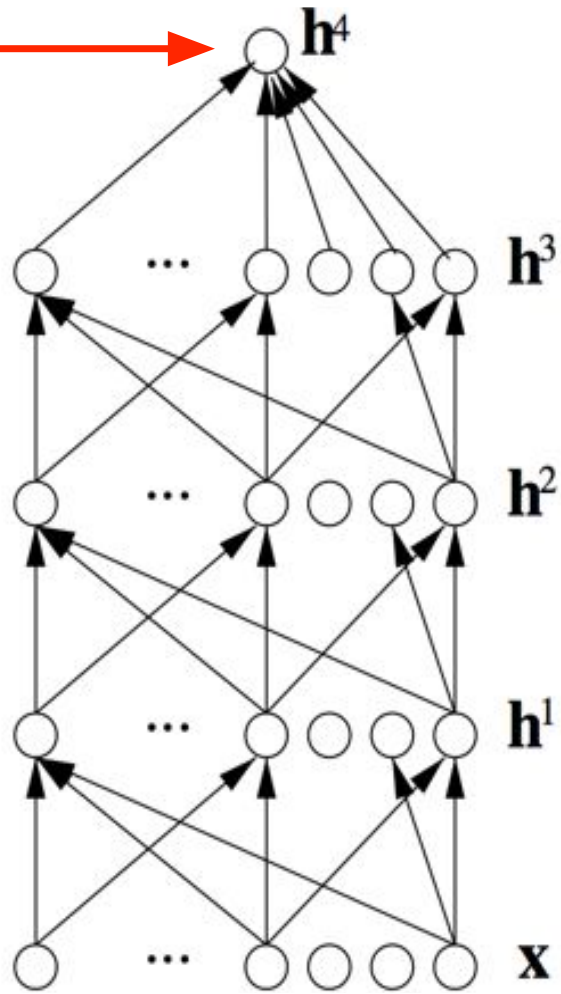
Output layer

prediction of the supervised target

Hidden Layers

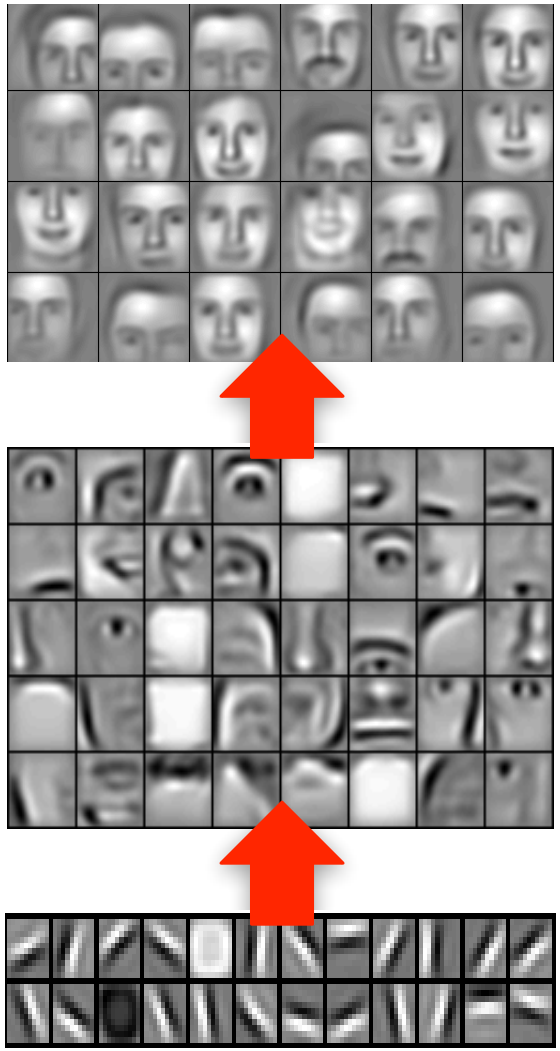
learn more abstract features as you head up

Raw inputs



(from Bengio, 2015)

Illustration

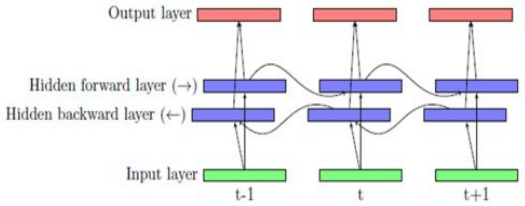


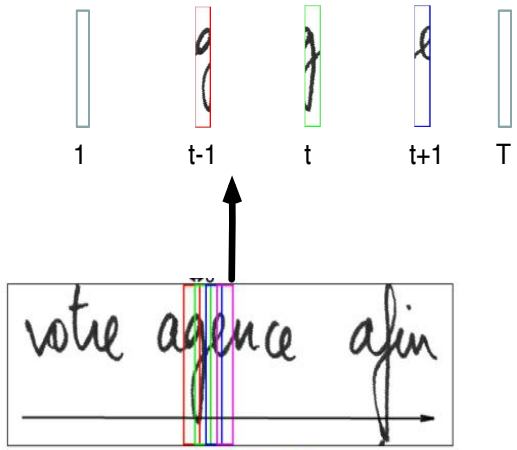
(?)

Deep learning in neural networks : a success story

Since 2009, deep learning approaches won several challenges

- ImageNet since 2012 (?)
- Traffic signs recognition: superhuman performance in 2011 (?) based on (?)
- Handwritting recognition since 2009 (?) based on (?)
- Automatic Speech recognition (?)
- Machine translation (?)





Deep learning in neural networks: a long story

The breakthrough of 2006

The expression *Deep Learning* was coined around 2006 with papers on unsupervised pre-training of neural nets (**?**; **?**; **?**)

And before ? (just a few dates)

- 1958 Rosenblatt proposed the perceptron (**?**), following the work of McCulloch and Pitts in 1943 and Hebb in 1949.
- 1980 Neocognitron (**?**) or the multilayered NNets
- 1982 Hopfield network with memory and recurrence (**?**), the unsupervised SOM (**?**), Neural PCA (**?**)
- 1986 Multilayer perceptrons and backpropagation (**?**)
- 1989 Autoencoders (**?**), Convolutional network (**?**)
- 1993 Sparse coding (**?**)

What is new ?

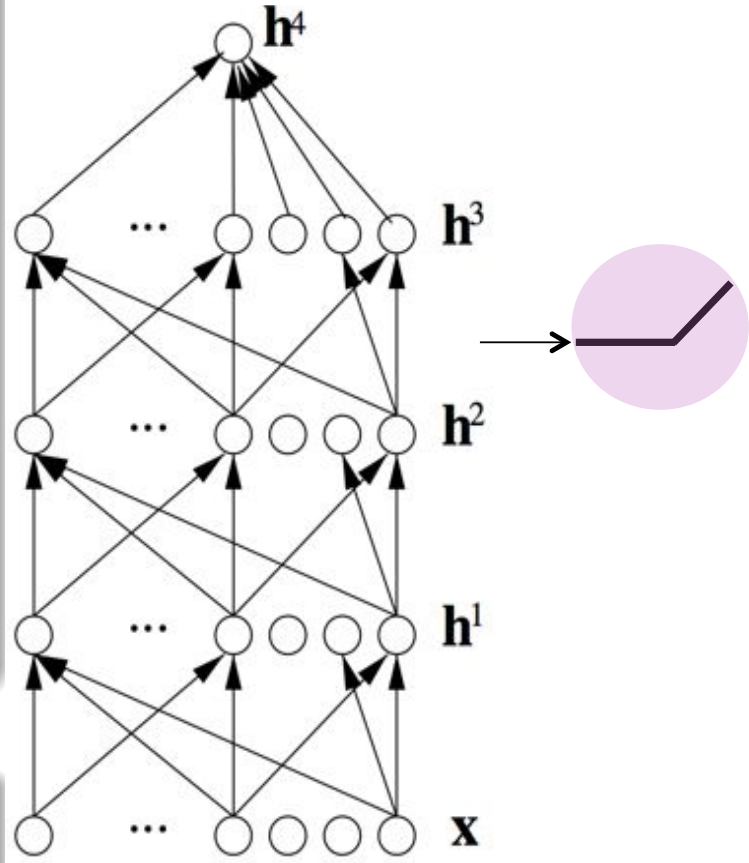
From Kyunghyun Cho's slides (2015)

Why today ?

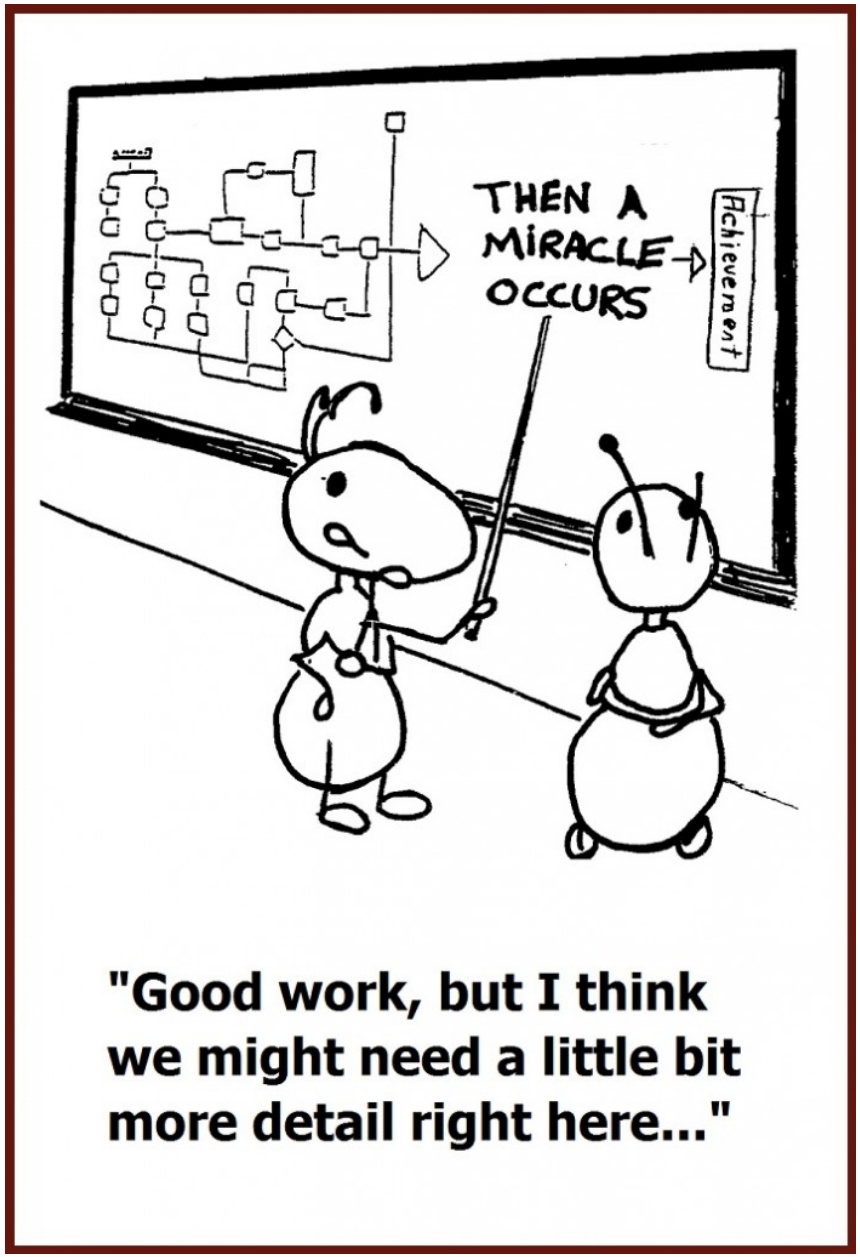
- We have connected the dots, e.g. (Probabilistic) PCA / Neural PCA / Autoencoder
- We understand learning better (regularization, architecture)
- No need to be scared of non-convex optimization (initialization)
- The huge amount of data and the growth of computational power.

What is the difference between a NNet and a Deep Network ?

An (very) intensive empirical exploration of the different issues



Y. Bengio, 2015


THEN A MIRACLE OCCURS
Achievement
"Good work, but I think we might need a little bit more detail right here..."

Outline

Expected schedules

First part

- 6/01, course: introduction to the course and basics on machine learning
- 13/01, **lab session** : machine learning and python, first steps
- 20/01, course: From linear to non-linear, the feed-forward architecture and the back-propagation algorithm
- 27/01, course: deep-learning, some important tricks and pytorch overview
- 03/02, **lab session** : deep-learning in pytorch
- 10/02, course: A first review, good practices in deep-learning, problem design, projects

The second part

- 02/03, course: Convolutional and recurrent deep-networks
- 09/03, **lab session**: Image and sequence processing
- 23/03, course: Advanced architecture - part 1
- 30/03, course: Advanced architecture - part 2

Evaluation

Homework

- A set of questions
- Following the courses and/or the lab sessions
- Submission: a short report or a notebook

Project

- Team of 2 or 3 students
- Starting from a dataset
- Submission: a report and the code

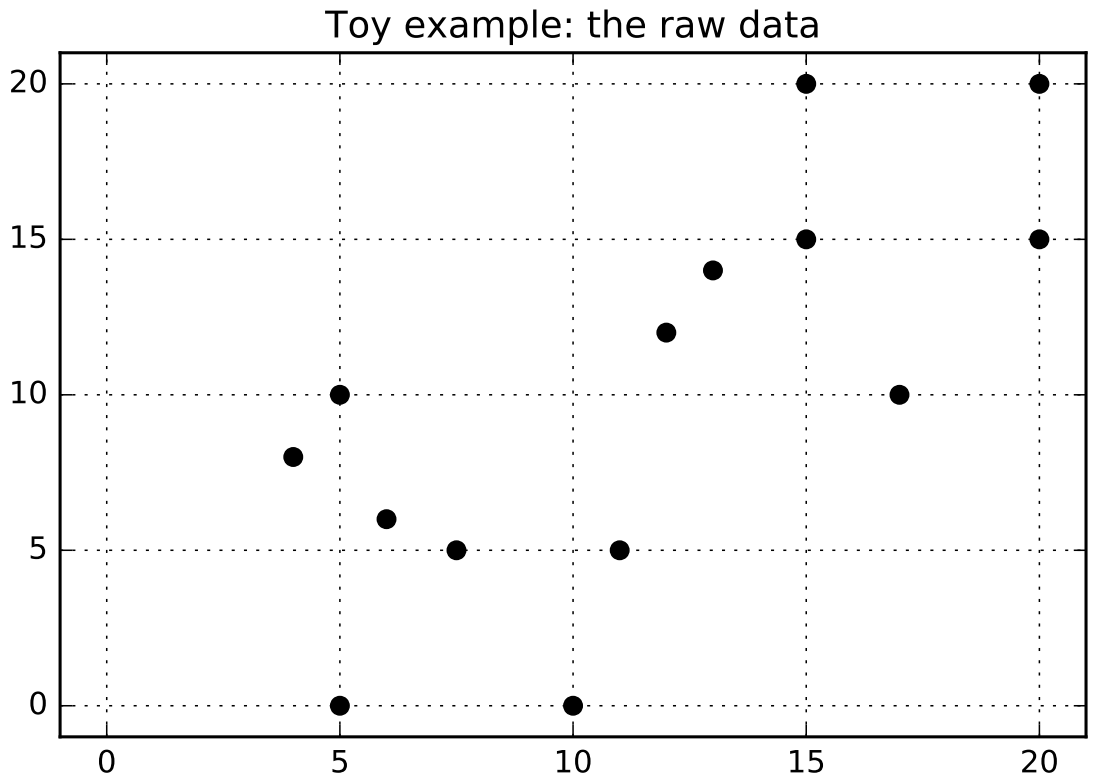
`https://allauzen.github.io/cours/IDL_ESPCI/`

Outline

Examples of a binary classification

The dataset $\mathcal{D}$

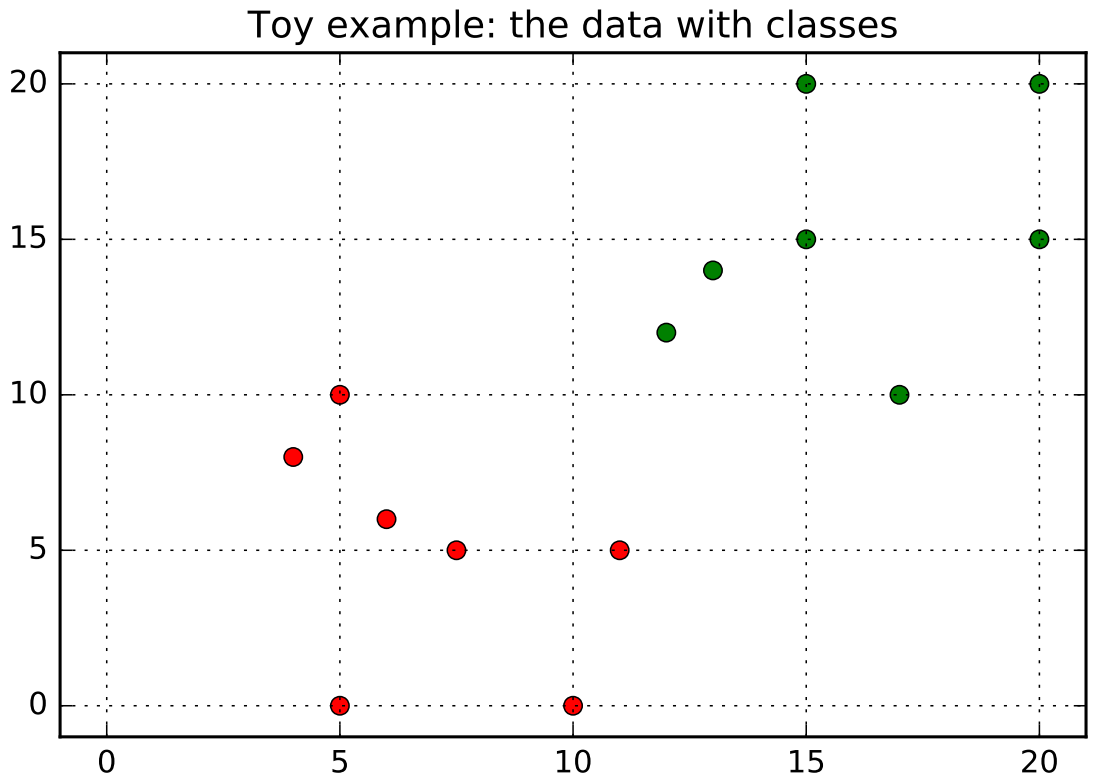
$$\mathcal{D} = \begin{pmatrix} x_1 = & 17 & 12 & 13 & 15 & 15 & 20 & 20 & 4 & 7.5 & 10 & 11 & 5 & 5 & 6 \\ x_2 = & 10 & 12 & 14 & 15 & 20 & 15 & 20 & 8 & 5 & 0 & 5 & 0 & 10 & 6 \\ c = & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



Examples of a binary classification

The dataset $\mathcal{D}$

$$\mathcal{D} = \begin{pmatrix} x_1 = & 17 & 12 & 13 & 15 & 15 & 20 & 20 & 4 & 7.5 & 10 & 11 & 5 & 5 & 6 \\ x_2 = & 10 & 12 & 14 & 15 & 20 & 15 & 20 & 8 & 5 & 0 & 5 & 0 & 10 & 6 \\ c = & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



Example of binary classification

Goal:

- Predict whether a candidate is hired ($c = 1$) or not ($c = 0$)
- knowing his results $\boldsymbol{x}$
- a candidate = 2 grades $\rightarrow \boldsymbol{x} = (x_1, x_2)$

The simplest model

$$c = 1 \text{ if } w_1 x_1 + w_2 x_2 > \alpha \text{ otherwise } c = 0$$
$$c = 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 > 0 \text{ otherwise } c = 0$$
$$c = 1 \text{ if } w_0 + \mathbf{w}^t \boldsymbol{x} > 0 \text{ otherwise } c = 0$$

Learning

- Find $\boldsymbol{\theta} = (w_0, \mathbf{w})$ given $\mathcal{D} = (\boldsymbol{x}_{(i)}, c_{(i)})_{i=1}^n$
- $c_{(i)}$ is the good answer (the supervision)

Machine Learning for classification

Given a training dataset :

$$\mathcal{D} = (\boldsymbol{x}_{(i)}, c_{(i)})_{i=1}^{n}$$

Defining a "model"

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) \to c$$

- $\boldsymbol{\theta}$: the set of (free-)parameters defining the function $f_{\boldsymbol{\theta}}$
- classification : associate a class c to $\boldsymbol{x}$, given $f_{\boldsymbol{\theta}}$ and a decision rule

Learning

- Find $\boldsymbol{\theta}$ given $\mathcal{D}$,
- Minimize the *loss function* (of $\boldsymbol{\theta}$) estimated on $\mathcal{D}$
- The loss function quantifies the difference between c and $c_{(i)}$

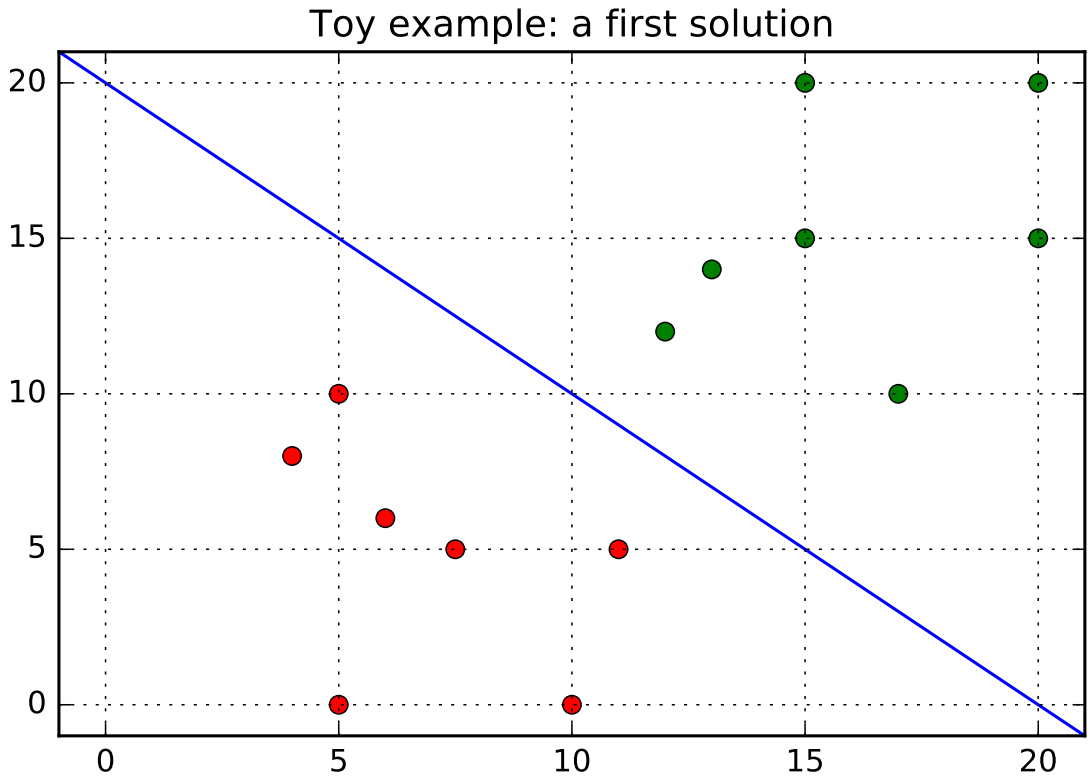
Refresher: dot or scalar product

For two vectors $\mathbf{w}$ and $\boldsymbol{x} \in \mathbb{R}^D$, the dot product results in a scalar value $\in \mathbb{R}$.

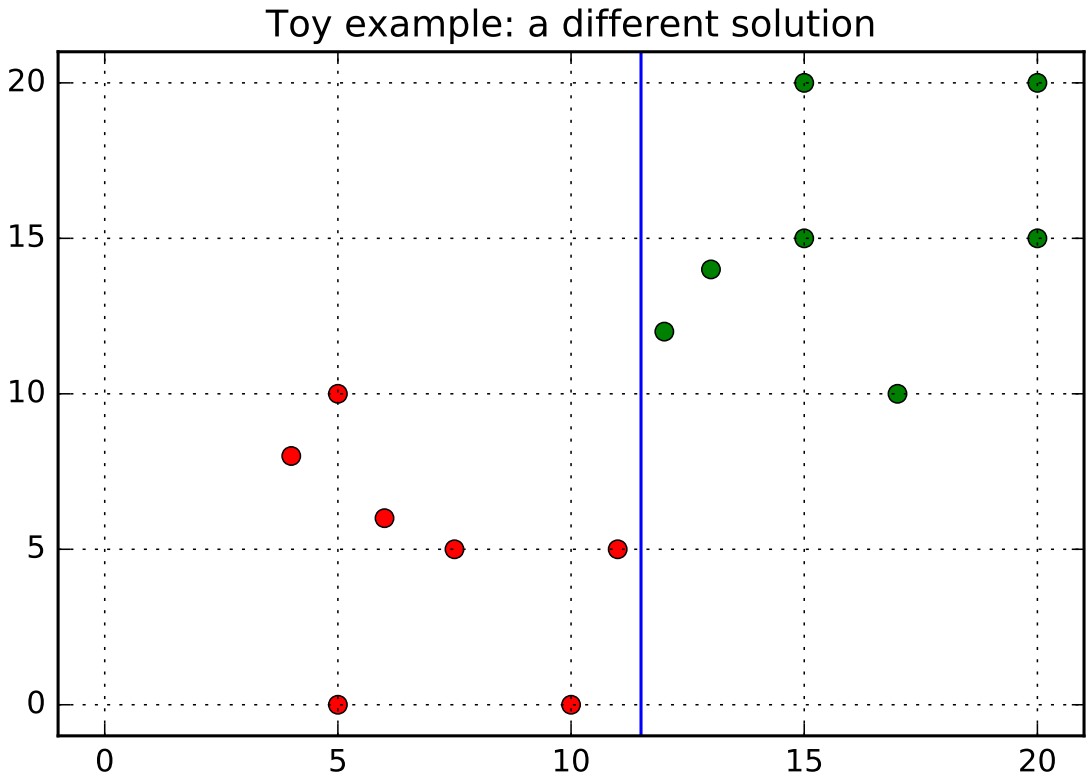
$$\mathbf{w} \cdot \boldsymbol{x} = \mathbf{w}^t \boldsymbol{x} = \langle \mathbf{w}, \boldsymbol{x} \rangle = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

$$= w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$

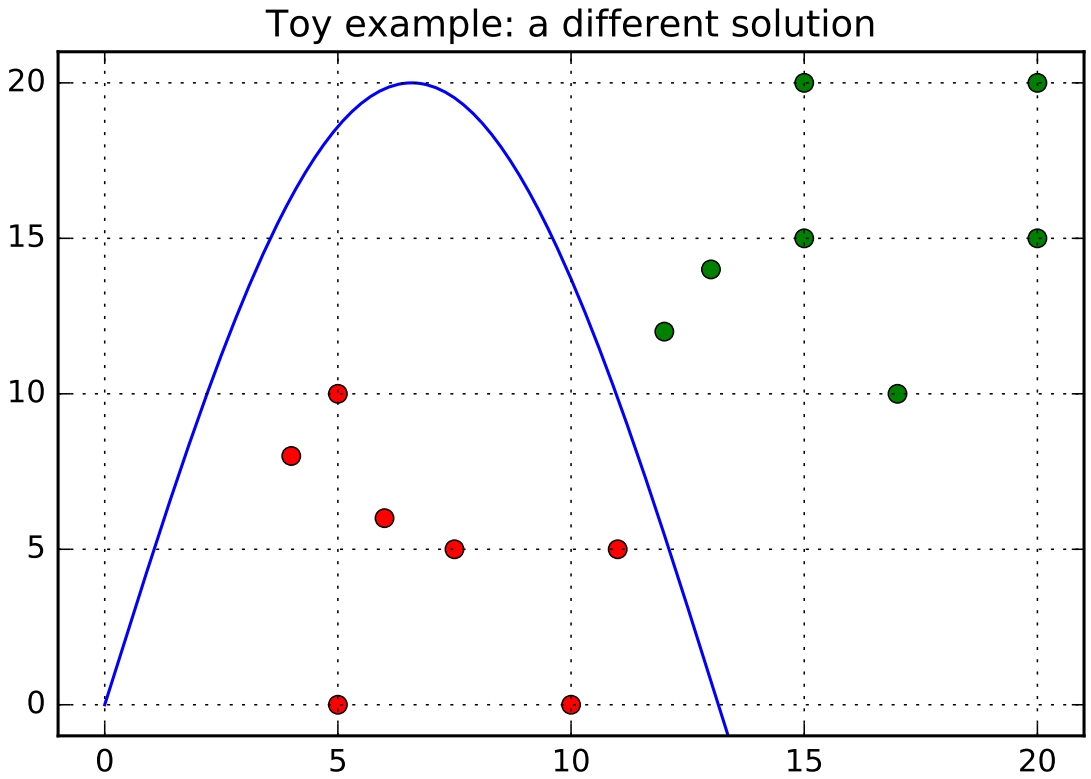
Binary classification or separation



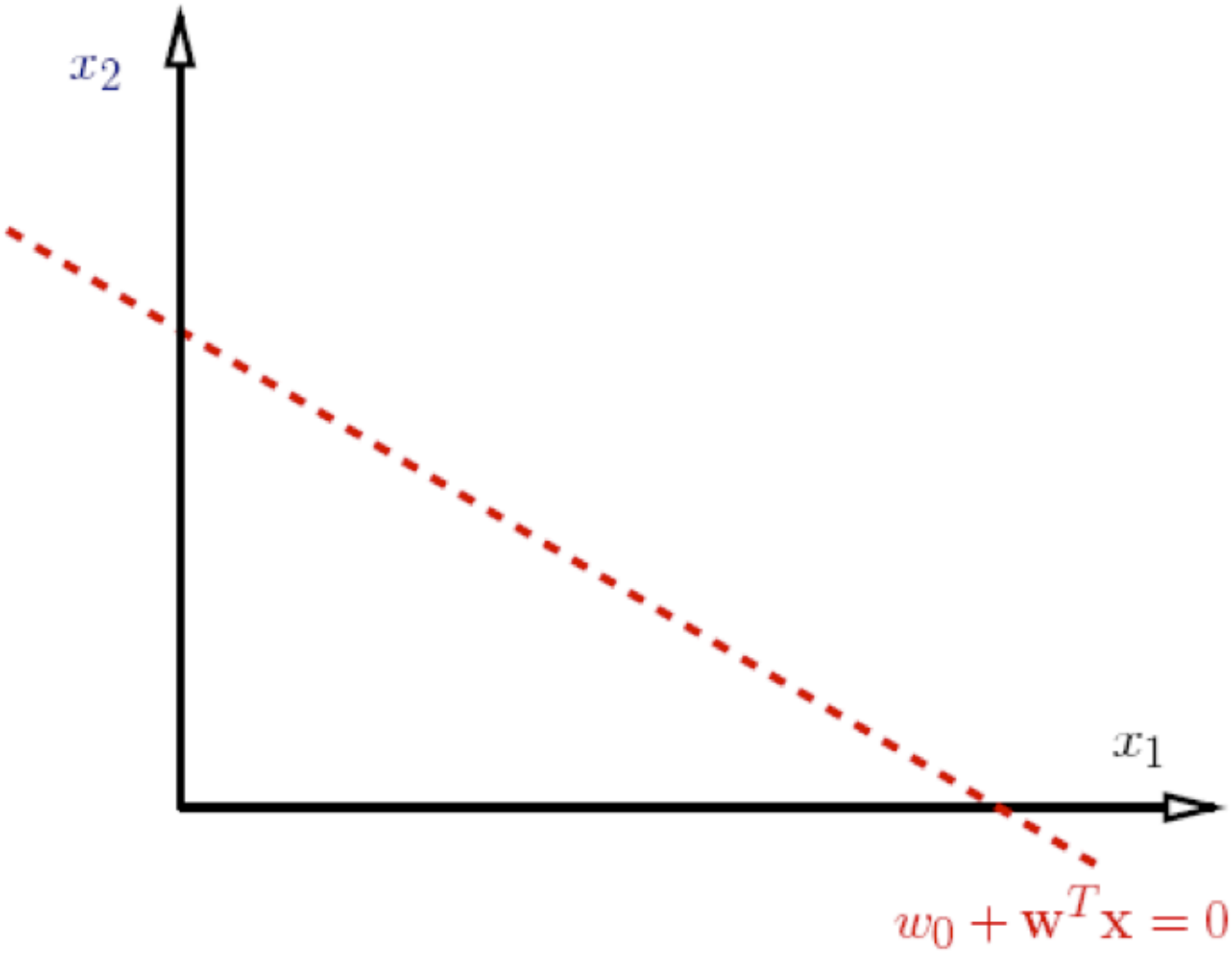
Binary classification or separation



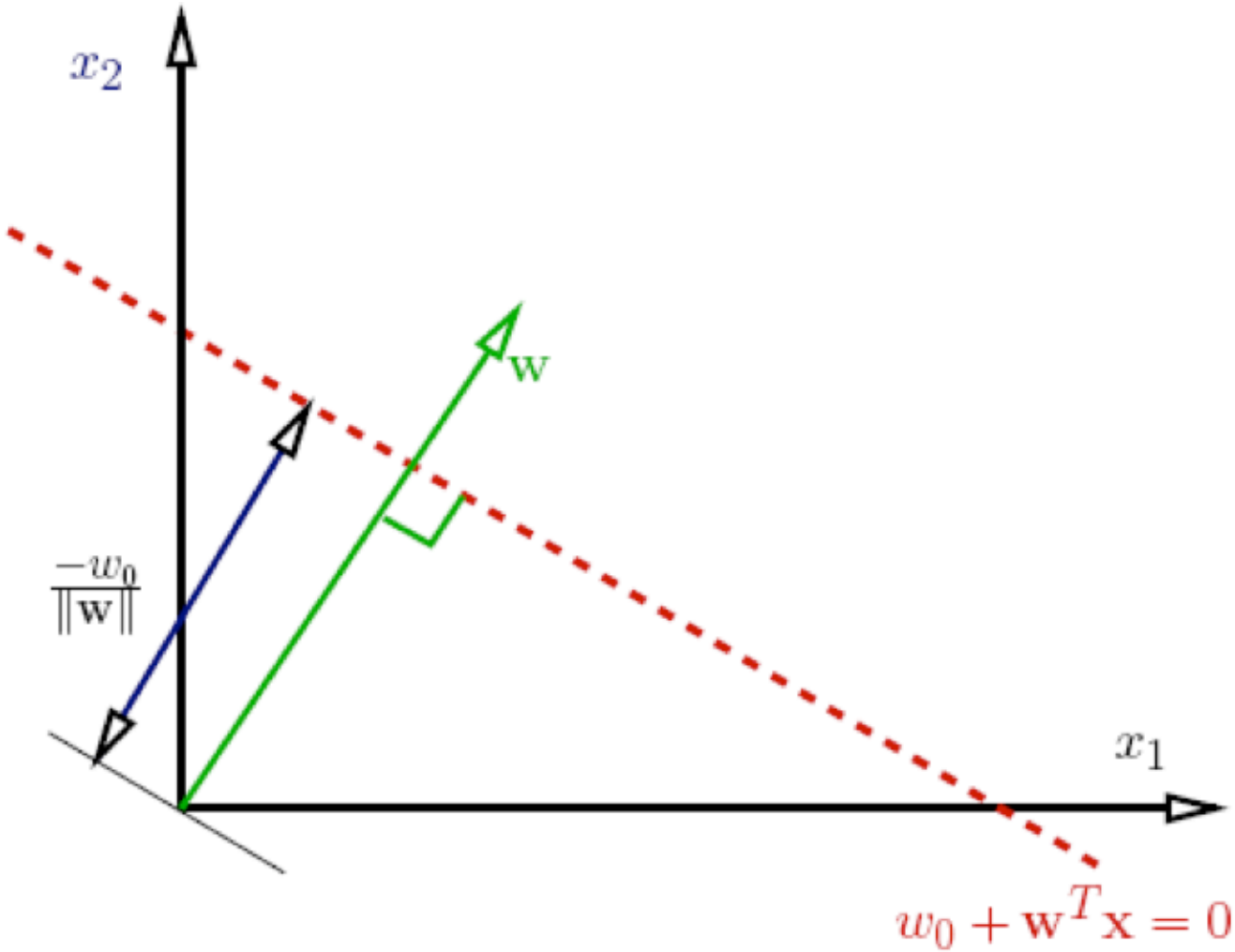
Binary classification or separation



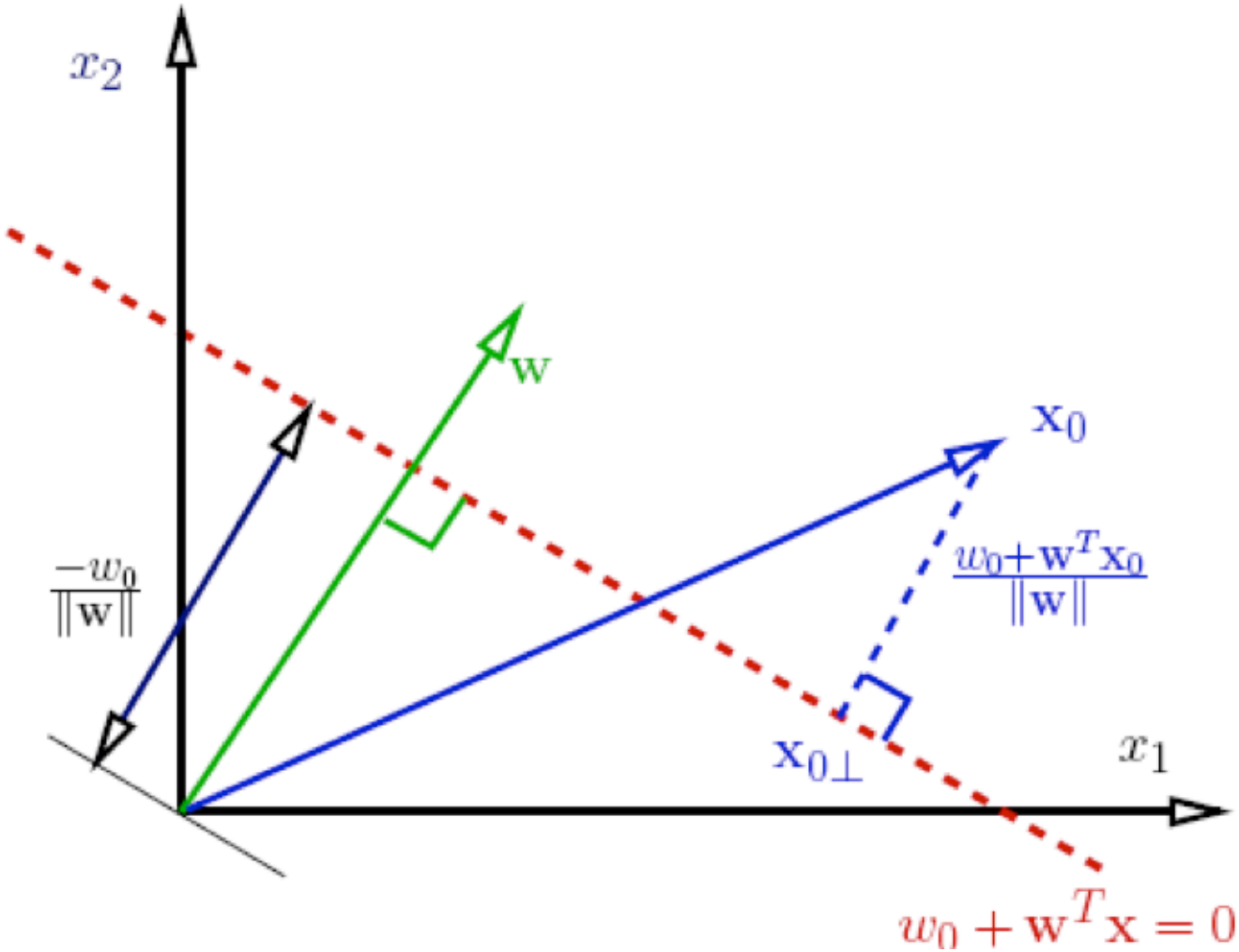
Linear separation: a bit of geometry



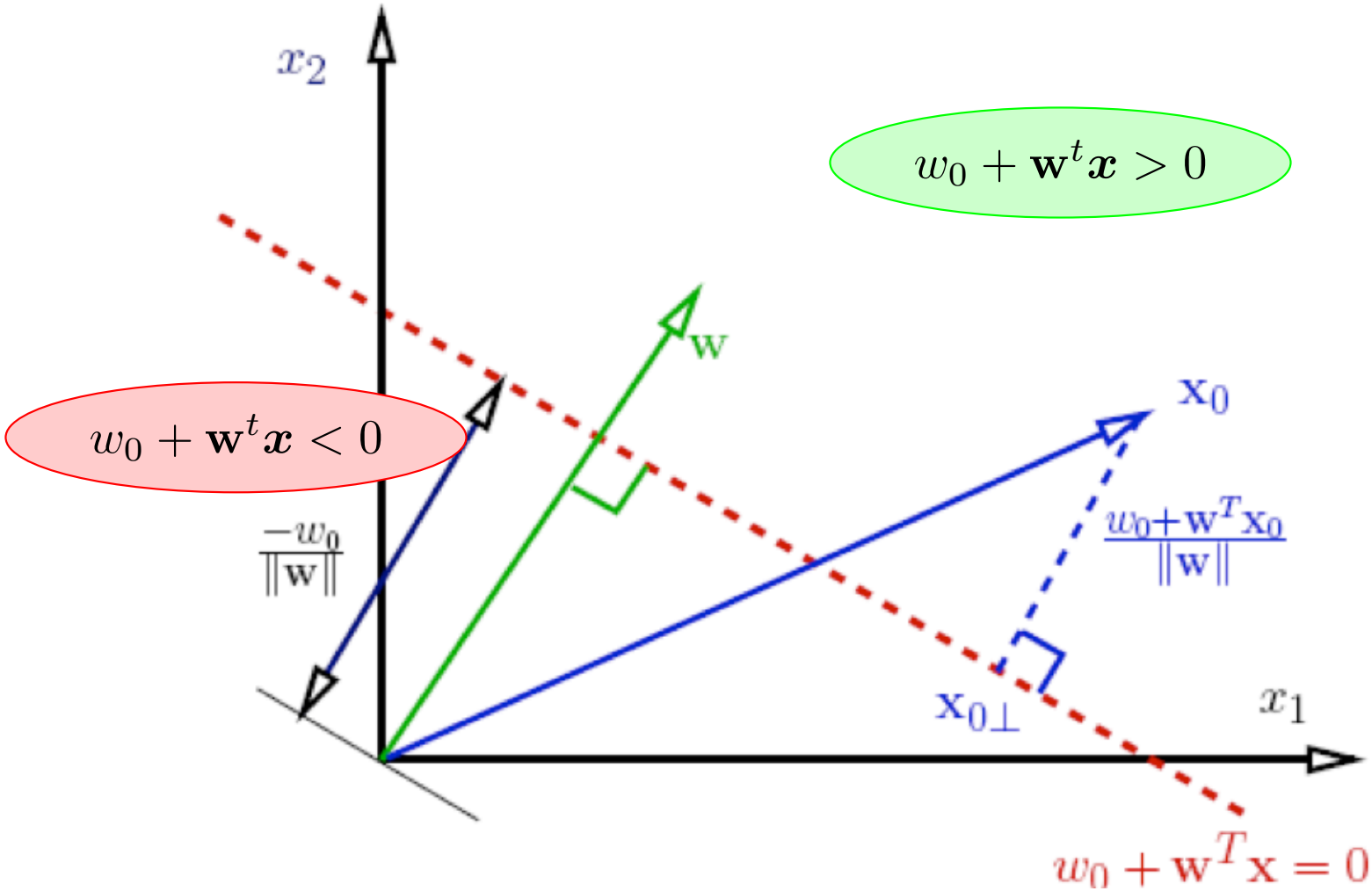
Linear separation: a bit of geometry



Linear separation: a bit of geometry



Linear separation: a bit of geometry



A model for linear separation

How to find the boundary (the loss function):

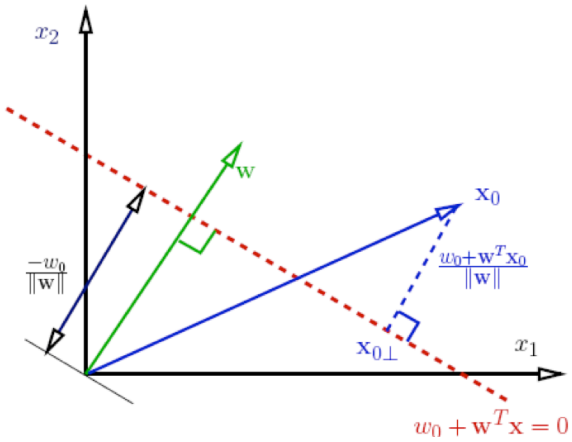
- Perceptron
- SVM
- Naive Bayes
- ...

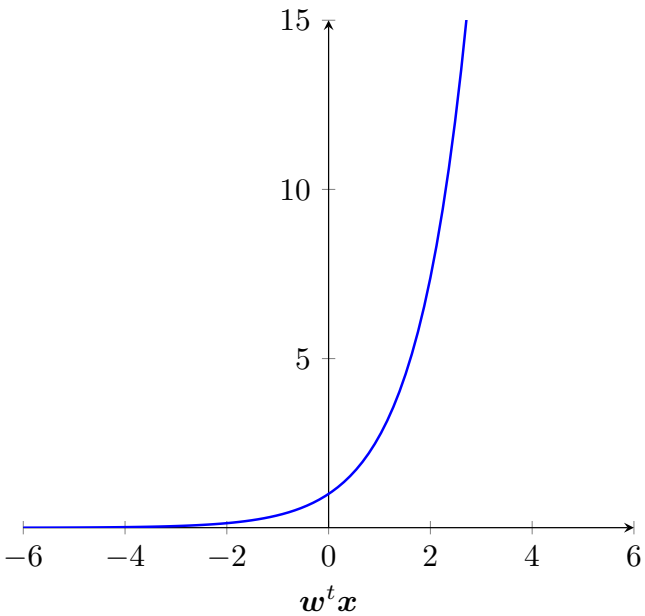
Logistic Regression

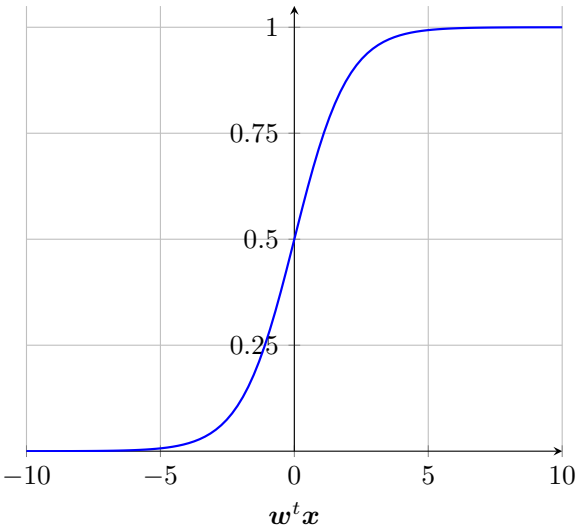
- a probabilistic criterion
- the simplest neural network : a single neuron

Probabilistic interpretation

$$-\infty < w_0 + \mathbf{w}^t \boldsymbol{x} < +\infty \qquad 0 < e^{w_0 + \mathbf{w}^t \boldsymbol{x}} < +\infty \qquad 0 < \frac{e^{w_0 + \mathbf{w}^t \boldsymbol{x}}}{1 + e^{w_0 + \mathbf{w}^t \boldsymbol{x}}} < 1$$







Logistic regression

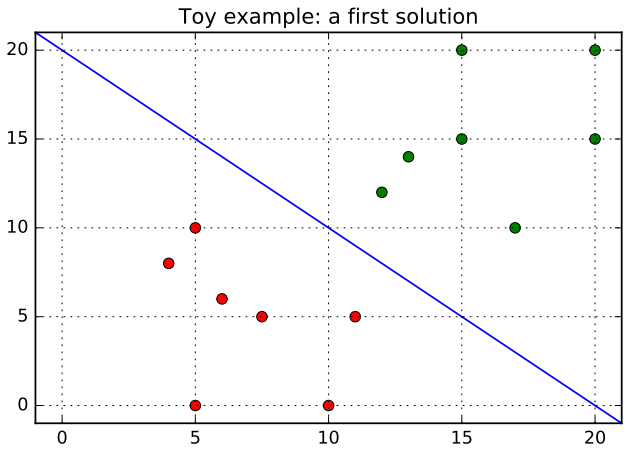
The class c is the outcome of the binary random variable C

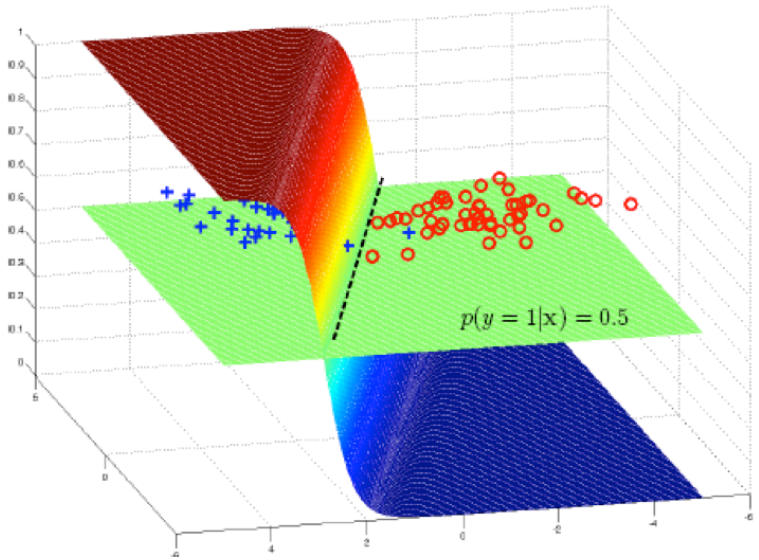
The sigmoid/logistic function

$$y = P(C = 1|\boldsymbol{x}) = \sigma(w_0 + \mathbf{w}^t\boldsymbol{x})$$

$$\sigma(a) = \frac{e^a}{1 + e^a} = \frac{1}{1 + e^{-a}}$$

$$a = w_0 + \mathbf{w}^t\boldsymbol{x} \in \mathbb{R}$$





Logistic regression model for binary classification

- The model is defined by
$$\boldsymbol{\theta} = (w_0, \mathbf{w})$$
- Its output is $y = P(C = 1|\boldsymbol{x}) = \sigma(w_0 + \mathbf{w}^t \boldsymbol{x})$
- The decision rule (c inference from the model output y):

$$c = 1 \text{ if } y = P(C = 1|\boldsymbol{x}) > P(c = 0|\boldsymbol{x}), \quad \text{and } c = 0 \text{ otherwise}$$
$$c = 1 \text{ if } y = P(C = 1|\boldsymbol{x}) > 0.5, \quad \text{and } c = 0 \text{ otherwise}$$

Now let us define a criterion to *learn* $\boldsymbol{\theta}$

Outline

Refresher: Bernoulli distribution

Let c be the outcome of a binary random variable C:

A Bernouilli distribution of parameter π

$$P(C = 1|\pi) = \pi$$
$$P(C = 0|\pi) = 1 - \pi$$
$$P(C = c|\pi) = \pi^c(1 - \pi)^{1-c}$$

For n observations i.i.d (independent, identically distributed) : $(c_{(1)}, ..., c_{(n)})$

$$P((c_{(1)}, ..., c_{(n)})|\pi) = \prod_{i=1}^{n} P(C = c_{(i)}|\pi) = \prod_{i=1}^{n} \pi^{c_{(i)}}(1 - \pi)^{1-c_{(i)}}$$

Refresher: Bernoulli, exercices

Given the following observations:

$$\mathcal{D} = (0, 0, 1, 0, 1, 1, 0, 0, 1, 0)$$

- Assume we know π, compute $P(\mathcal{D}|\pi)$
- And what if $\mathcal{D} = (1, 0, 0, 0, 1, 1, 0, 0, 0, 1)$?
- What are the sufficient statistics under our assumptions ?
- If you should estimate π (learn) from $\mathcal{D}$, what is your guess ?
- What is the underlying criterion ?

The conditionnal "Likelihood" measured on $\mathcal{D}$

Given $\mathcal{D} = (\boldsymbol{x}_{(i)}, c_{(i)})_{i=1}^n = (\mathcal{X}, \tilde{\mathcal{C}})$ and a logistic regression model, $\boldsymbol{\theta} = (w_0, \mathbf{w})$.

$$P(\tilde{\mathcal{C}}|\mathcal{X}, \boldsymbol{\theta}) = \prod_{i=1}^{n} P(C = c_{(i)}|\boldsymbol{x}_{(i)}, \boldsymbol{\theta})$$

$$P(\tilde{\mathcal{C}}|\mathcal{X}, \boldsymbol{\theta}) = \prod_{i=1}^{n} \pi_{(i)}^{c_{(i)}} (1 - \pi_{(i)})^{1 - c_{(i)}}$$

$$\pi_{(i)} = y_{(i)} = \sigma(w_0 + \mathbf{w}^t \boldsymbol{x}_{(i)}) = \frac{1}{1 + e^{-(w_0 + \mathbf{w}^t \boldsymbol{x}_{(i)})}}$$

The loss function (to minimize)

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = -log(P(\tilde{\mathcal{C}}|\mathcal{X}, \boldsymbol{\theta})) = \sum_{i=1}^{n} -\big(c_{(i)} log(y_{(i)}) + (1 - c_{(i)}) log(1 - y_{(i)})\big)$$

$$= \sum_{i=1}^{n} l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}), \text{ with } l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}) \text{ the pointwise loss.}$$

Outline

Optimisation Program

Let $\mathcal{D} = (\boldsymbol{x}_{(i)}, c_{(i)})_{i=1}^{n} = (\mathcal{X}, \tilde{\mathcal{C}})$, and a logistic regression model, $\boldsymbol{\theta} = (w_0, \mathbf{w})$.

The loss function

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = -log(P(\tilde{\mathcal{C}}|\mathcal{X}, \boldsymbol{\theta})) = -\big(\sum_{i=1}^{n} c_{(i)} log(\pi_{(i)}) + (1 - c_{(i)}) log(1 - \pi_{(i)})\big)$$

with:

$$\pi_{(i)} = \sigma(w_0 + \mathbf{w}^t \boldsymbol{x}_{(i)}) = \frac{1}{1 + e^{-(w_0 + \mathbf{w}^t \boldsymbol{x}_{(i)})}}$$

Learning = optimisation

Find $\boldsymbol{\theta}$ in ordre to minimize $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$?

Function minimization

Bad news

No closed form solution in general !

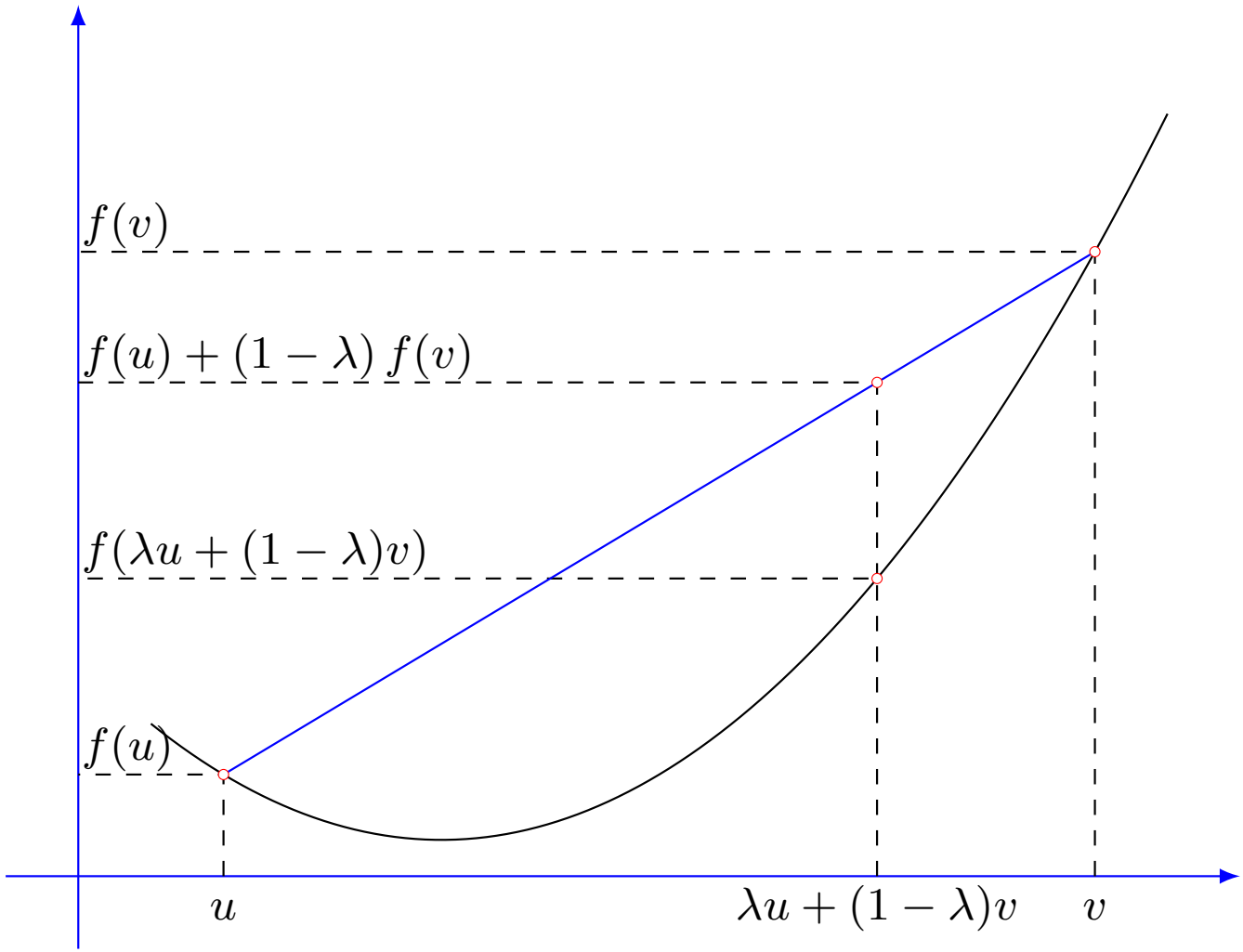
Good news

- The function is convex *w.r.t* $\boldsymbol{\theta}$
- Convex functions are "easy" to minimize

$\rightarrow$ An efficient, easy and generic algorirthm exists:

The (stochastic) gradient descent with many variants

A convex function



$$f\left(\lambda u + (1-\lambda)v\right) \leq \lambda\, f(u) + (1-\lambda)\, f(v).$$

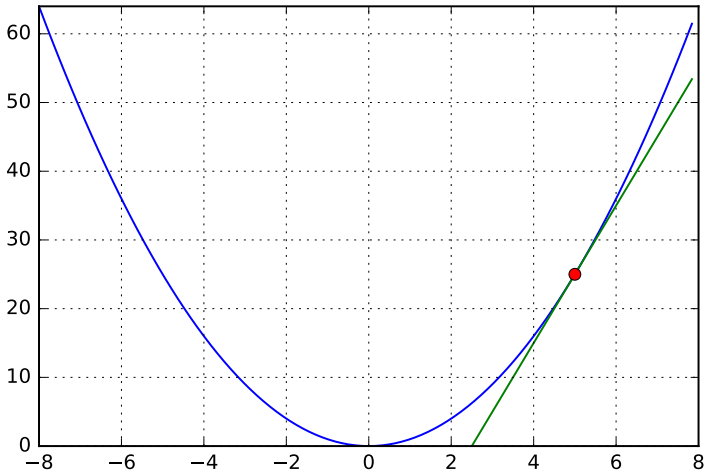
Gradient Descent - intuition

How to reach the minimum ? How to recognize it ?

Caution :

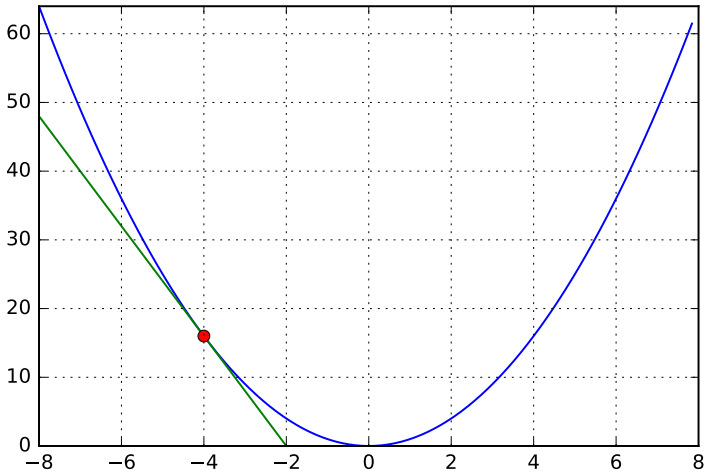
- The whole landscape is unknown
- Only pointwise knowledge: the function value and its (partial) derivatives

Interpretation of partial derivative



$$w_k = w_k + \eta \frac{\partial l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)})}{\partial w_k}$$

$$l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}) \nearrow$$



$$w_k = w_k - \eta \frac{\partial l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)})}{\partial w_k}$$

$$l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}) \searrow$$

$\eta > 0$ **and small**

Minimization of l $w.r.t$ w_k

Sketch

0. Start from a random point,
1. evaluate the slope,
2. follow it a bit, and
   loop back to the previous step

Algorithm

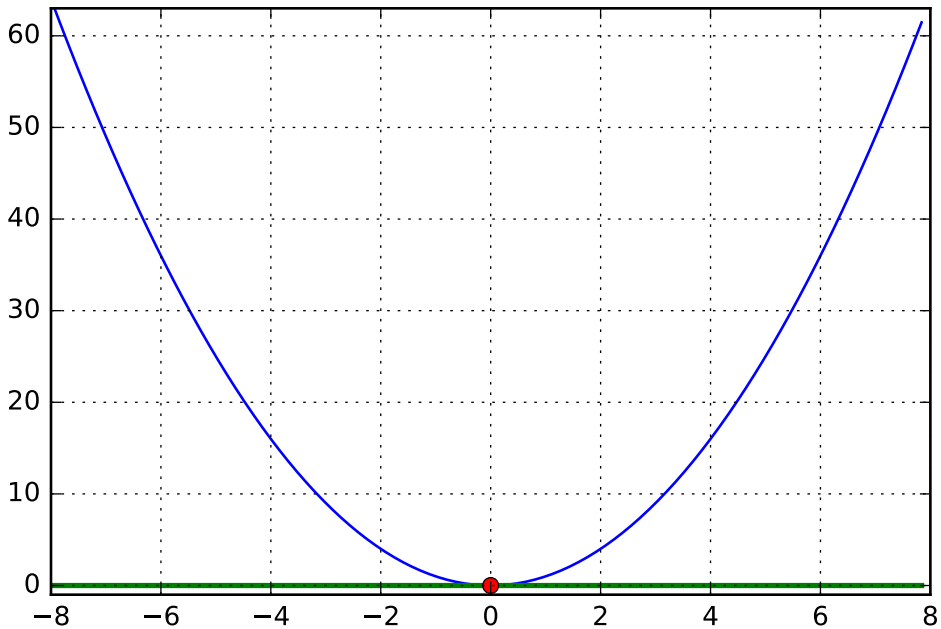
- Pick $\eta > 0$ and the initial value of w_k.
- while *true* :
  - compute $l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)})$
  - compute the derivative and update w_k

$$w_k = w_k - \eta \frac{\partial l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)})}{\partial w_k}$$

"Convergence"

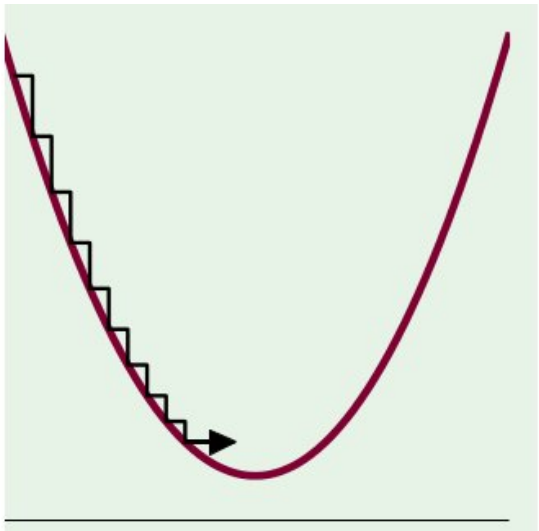
At the minimum:

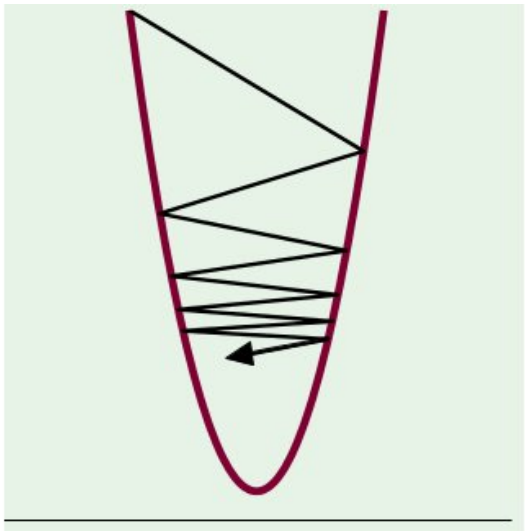
$$\frac{\partial l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)})}{\partial w_k} = 0$$

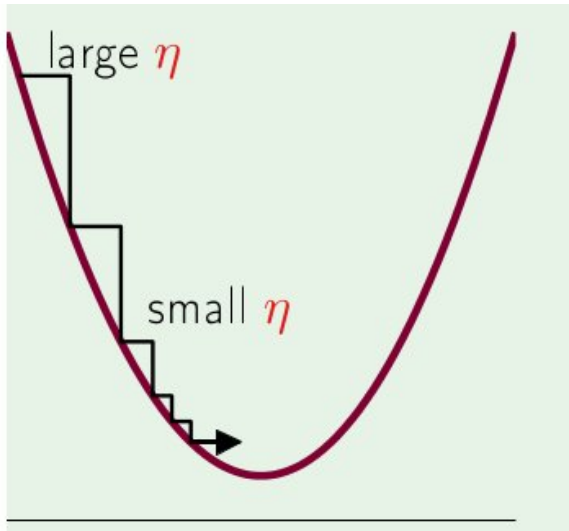


The learning rate : η

- small steps: a valid approximation but boring !
- large steps: oscillation and divergence







- Constant or adaptative η
- This is the role of the *optimizer* !

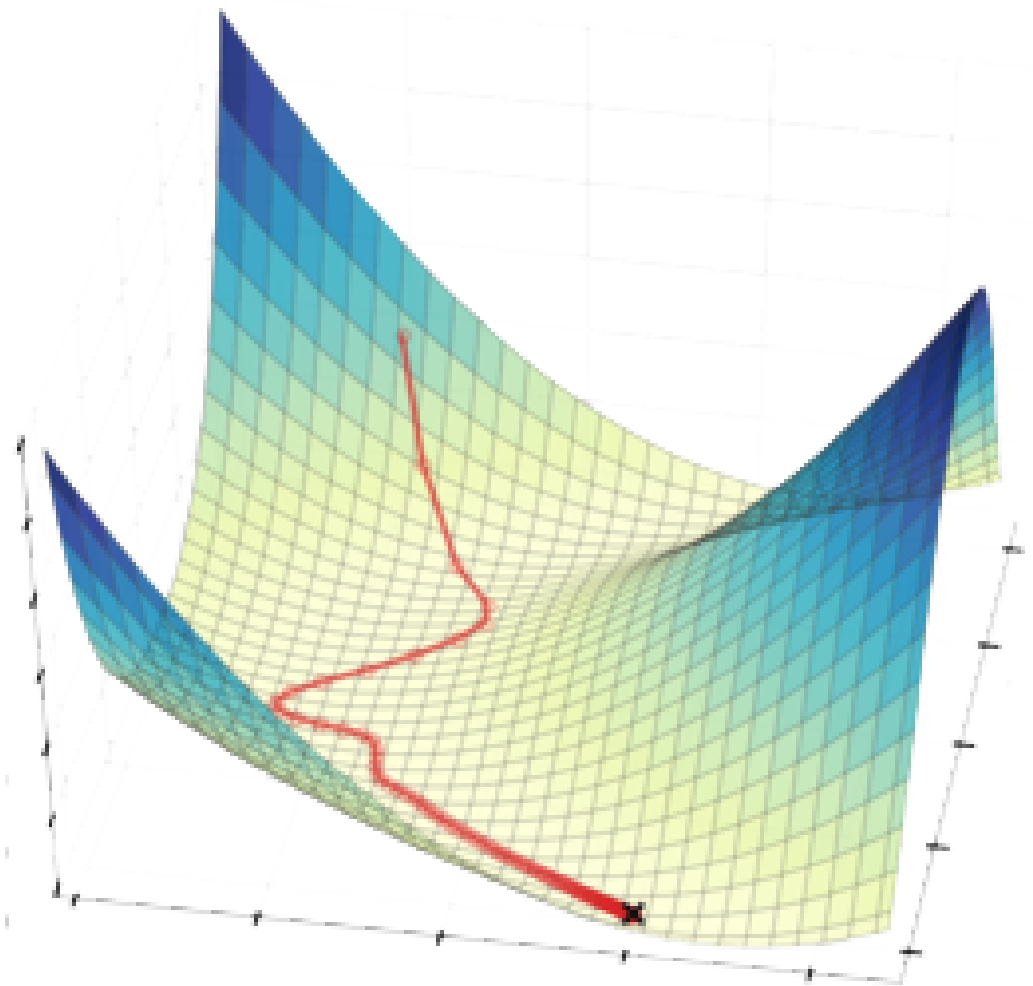
Generalization : the gradient

- All the parameters are in one vector : $\boldsymbol{\theta} = (w_0, \mathbf{w}) = (w_0, w_1, w_2, ..., w_D)$
- Compute all the partial derivatives and store them in one vector:

$$\begin{pmatrix} \partial l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)})/\partial w_0 \\ \partial l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)})/\partial w_1 \\ \partial l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)})/\partial w_2 \\ ... \\ \partial l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)})/\partial w_D \end{pmatrix} = \nabla_{\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}} l$$

$\nabla_{\boldsymbol{\theta}} l$ is the gradient (vector) of l w.r.t $\boldsymbol{\theta}$.

Gradient descent in 2D



Gradient descent: the algorithm

Batch / Online

Batch minimization

while:

$$\left.\begin{array}{l} \text{Pour } i = 1...n : \\ \qquad \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) += l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}) \\ \qquad \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) += \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}) \\ \boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) \end{array}\right\} \text{one epoch}$$

Stochastic Gradient Descent

while:

$$\left.\begin{array}{l} \text{shuffle } \mathcal{D} \\ \text{for } i = 1...n : \\ \qquad \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) += l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}) \\ \qquad \boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}) \end{array}\right\} \text{one epoch}$$

A trade-off: the mini-batch

A terminololy point

Gradient Descent

A simple yet efficient algorithm to minimize a convex function

Batch *vs* Online

- Batch: consider the whole $\mathcal{D}$ to estimate the gradient and the update
- Online: consider the training examples one by one
- Mini-batch: the trade-off

Stochastic

For online and mini-batch training: randomize the order in $\mathcal{D}$

Outline

Summary

Machine Learning overview

- A model defines a **function**: $\boldsymbol{x} \rightarrow \boldsymbol{y}$
- The output is then used by a decision rule.
- This function is defined by the parameters $\boldsymbol{\theta}$.
- **Training/Learning** finds the good values for $\boldsymbol{\theta}$
- How ? by minimizing a **loss function** $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$.
- How ? by **Stochastic Gradient Descent**
- The **learning rate** is an **hyper-parameter**.

From logistic regression to artificial neurone

- An artificial neurone is a **linear model** followed by a non-linear **activation**
- With the sigmoid activation, it is similar to the **logistic regression**


P. Baldi and K. Hornik.

1989.

Neural networks and principal component analysis: Learning from examples without local minima.

Neural Networks, 2(1):53–58, January.


Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle.

2007.

Greedy layer-wise training of deep networks.

In B. Schölkopf, J.C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press.


Dan C. Ciresan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber.

2012.

Multi-column deep neural network for traffic sign classification.

Neural Networks, 32:333–338.


Dj Field, 1993.

Scale-invariance and self-similar 'wavelet' transforms: an analysis of natural scenes and mammalian visual systems, pages 151–193.

Oxford University press.


Kunihiko Fukushima.

1980.

Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.

Biological Cybernetics, 36:193–202.

Alex Graves and Juergen Schmidhuber.

2009.

Offline handwriting recognition with multidimensional recurrent neural networks.

In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. Curran Associates, Inc.

G. E. Hinton and R. R. Salakhutdinov.

2006.

Reducing the dimensionality of data with neural networks.

Science, 313(5786):504–507, JUL 28.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh.

2006.

A fast learning algorithm for deep belief nets.

Neural Computation, 18(7):1527–1554, JUL.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury.

2012.

Deep neural networks for acoustic modeling in speech recognition.

Signal Processing Magazine.

Sepp Hochreiter and Jürgen Schmidhuber.
1997.
Long short-term memory.
Neural Comput., 9(8):1735–1780, November.

J. J. Hopfield.
1982.
Neural networks and physical systems with emergent collective computational abilities.
Proceedings of the National Academy of Sciences of the United States of America, 79(8):2554–2558.

Teuvo Kohonen.
1982.
Self-organized formation of topologically correct feature maps.
Biological Cybernetics, 43(1):59–69.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton.
2012.
Imagenet classification with deep convolutional neural networks.
In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel.

1989.

Backpropagation applied to handwritten zip code recognition.

Neural Comput., 1(4):541–551, December.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng.

2009.

Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.

In *Proceedings of the International Conference of Machine Learning (ICML)*, pages 609–616.

Erkki Oja.

1982.

Simplified neuron model as a principal component analyzer.

Journal of Mathematical Biology, 15(3):267–273.

F. Rosenblatt.

1958.

The perceptron: A probabilistic model for information storage and organization in the brain.

Psychological Review, 65(6):386–408.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams.

1986.

Learning representations by back-propagating errors.

Nature, 323(6088):533–536, 10.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin.

2017.

Attention is all you need.

In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.