

Introduction to Artificial Intelligence

Course EIB11AI001

International Bachelor of Science in AI (IBSAI), Year 1 (B1)

Angelo Saadeh
angelo.saadeh@ens.fr
45 rue d'Ulm, Office A321, 75005 Paris

Fall 2026

Objectives

- Overview of all fundamentals : History, applications, ethical implications.
- Explore some specific AI methods and techniques.
- Understand what is possible with AI and its limitations.

Topics

1. What is Artificial Intelligence ?
2. Optimization
3. Logic, Reasoning, Knowledge Representation
4. Machine Learning, supervised and unsupervised methods
5. Model Evaluation & Metrics
6. Deep Learning
7. Reinforcement Learning & Active Learning
8. Classical NLP & Embeddings
9. Attention & Transformers
10. Generative AI
11. Agentic AI & Multi-Agent Systems
12. Ethics, Legal, Geopolitics, Environment

Evaluation

- Weekly evaluations through 5 to 10 minute quizzes : 20% of the final grade.
- One final evaluation : 80% of the final grade.
- Same attendance rules apply to this course.

References I

[1] Ronald J. Brachman and Hector J. Levesque. *Knowledge Representation and Reasoning*. Elsevier, 2004.

[2] Steven L. Brunton. *Optimization: A Bootcamp for Machine Learning, Inverse Problems, and Control*. Cambridge University Press, 2026.

[3] Liang-Chieh Chen et al. *Rethinking Atrous Convolution for Semantic Image Segmentation*. arXiv:1706.05587. 2017.

[4] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* (1989).

[5] DARPA. *DARPA Neural Network Study*. AFCEA International Press, 1988.

[6] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009.

[7] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[8] Aidan Hogan et al. *Knowledge Graphs*. Springer, 2021.

References II

[9] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* (1991).

[10] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations (ICLR 2015)*. 2015.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25 (NIPS 2012)*. 2012.

[12] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* (1998).

[13] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. 2014.

[14] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*. 2008.

[15] Andrew Ng. *Convolutional Neural Networks*. 2017.

References III

[16] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[17] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems 28 (NIPS 2015)*. 2015.

[18] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* (2015).

[19] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

[20] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* (2014).

[21] Wikipedia contributors. *Machine learning*. Wikipedia, The Free Encyclopedia. 2026.

Chapter 1: Definitions and Foundations

Definitions

Foundations of Artificial Intelligence

History of Artificial Intelligence

Current Expectations of Artificial Intelligence

Intelligence

Intelligence

Webster's Definition

1. The ability to learn or understand things or to deal with new or difficult situations : Reason
2. Mental acuteness : Cleverness
3. ...

French Larousse

1. The quality of someone who, in a given field, demonstrates a desire to understand, reflect, and acquire knowledge, and who readily adapts their behavior to these objectives.
2. ...

Artificial Intelligence

Artificial Intelligence

Artificial intelligence is a set of techniques aimed at designing computational systems that imitate, reproduce or surpass human intelligence :

Artificial Intelligence

Artificial intelligence is a set of techniques aimed at designing computational systems that imitate, reproduce or surpass human intelligence :

- Learning
- Reasoning
- Problem-solving
- Perception
- Decision-making
- Adaptability
- Generalization
- Language / Communication
- Planning
- Memory
- ...

Artificial Intelligence

1. Should Artificial Intelligence be more faithful to Human performance or to Rationality ?
2. Should Intelligence be a property of internal thought processes and reasoning or behavior ?

Artificial Intelligence

1. Should Artificial Intelligence be more faithful to Human performance or to Rationality ?
2. Should Intelligence be a property of internal thought processes and reasoning or behavior ?

	Humanly	**Rationally**
Thinking	Thinking Humanly	Thinking Rationally
Acting	Acting Humanly	Acting Rationally

Can Machines Think ?

Edsger Dijkstra (1984) : "The question of whether machines can think is about as relevant as the question of whether submarines can swim."

A linguistic issue :
Swim : Use limbs (submarines don't swim)
Fly : Move in the sky (planes fly)
Word usage in English VS design of planes or submarines
плавать (Plavat) : Swim, float, sail
Think : does it require "a brain" ? "brain-like parts ?"

Alan Turing : "We never have any direct evidence about the internal mental states of other humans, it is usual to have the polite convention that everyone thinks."

John Searle (1980) : rejects the polite convention, with the Chinese Room argument.

Acting Humanly : The Turing Test Approach

Alan Turing (1912-1954) : Mathematician, Computer scientist, and Cryptanalyst know for designing one of the first computers (among other things).

Turing Test (1950) : A machine is intelligent if it is indistinguishable from a human being who can ask it arbitrary questions.
The machine would require :

- Natural language processing to communicate
- Knowledge representation to store what it knows or hears
- Automated reasoning to answer questions and draw conclusions
- ...

Acting Humanly : The Turing Test Approach

- The idea is to measure intelligence by performance on some kind of open-ended behavioral task.
- We can't 100% agree whether any program has passed the test :
  - ELIZA (60s), MGONZ (1989), NATACHATA (2009), CYBERLOVER(2007)
  - In 2014, a chatbot called Eugene Goostman fooled 33% of the untrained amateur judges (impersonating a Ukrainian boy to justify grammar mistakes)
  - Nowadays' ChatGPT and other AI chatbots are harder to distinguish from humans during conversations
- Passing the Turing test really means a system is Intelligent ?
- Is it easier to mimic humans than it is to reason ?
- Was the Turing test a test for human gullibility ? :-)

Acting Rationally : The Rational Agent Approach

Agent : From the Latin *Agere* – to do – it is something that acts.

- **Human vs. Rational** : The standard of rationality is mathematically well defined and completely general.
- **Acting vs. Thinking** : Acting is more general than thinking, because correct inference is just one of several possible mechanisms for achieving rationality.

Weak AI vs. Strong AI

- **Weak AI** : a system that is intelligent for a specific task.
- **Strong AI** : a system capable of handling all tasks requiring intelligence, on par with human beings.

Chapter 1: Definitions and Foundations

Definitions

Foundations of Artificial Intelligence

History of Artificial Intelligence

Current Expectations of Artificial Intelligence

Philosophy

- Can formal rules be used to draw valid conclusions ?
- How does the mind arise from a physical brain ?
- ...

Dualism : there is a part of the human mind that is outside of nature, exempt from physical laws.

Empiricism : nothing is in the understanding, which was not first in the senses.

Induction : general rules are acquired by exposure to repeated associations between their elements.

Utilitarianism : rational decision making based on maximizing utility should apply to all spheres of human activity, including public policy decisions made on behalf of many individuals.

Mathematics

- Formal Logic : What are the formal rules to draw valid conclusions ?
- Probability Theory : How to deal with uncertain information ?
- Statistics : How to analyze data ?
- Algorithms : How to perform computations and solve problems ?

Other Disciplines

Economics : How to make decisions when people disagree or payoffs are in the future ?

- Game theory
- Decision theory

Neuroscience : How do brains process information ?

- Neurons (in neural networks) receive and conduct signals
- Mapping areas of the brain to part of the body it controls

Psychology : How do humans and animals think and act ?

Linguistics : How does language relate to thought ?

Computer Engineering : How to build an efficient computer ?

Chapter 1: Definitions and Foundations

Definitions

Foundations of Artificial Intelligence

History of Artificial Intelligence

Current Expectations of Artificial Intelligence

Artificial Neural Networks

Warren McCulloch and Walter Pitts (1943) : Neural network with ON and OFF neurons with thresholds :

- A neuron is ON when stimulated by neighboring neurons
- All the logical connectives (AND, OR, NOT, etc.) could be implemented by simple network structures
- Any computable function could be computed by some network of connected neurons

Neural Network Computer : Stochastic Neural Analog Reinforcement Calculator (SNARC)

Marvin Minsky and Dean Edmonds (1951) built the first neural network computer called SNARC, which used :

- 300 vacuum tubes
- A surplus automatic pilot mechanism from a B-24 bomber to simulate a network of 40 neurons.

Other inceptions

- Checker playing programs (1952)
- Dartmouth Workshop (1956) – first use of the term Artificial intelligence

Expectations : What can AI do ?

A machine could never do :

Expectations : What can AI do ?

A machine could never do :

List by Alan Turing

- Be kind, resourceful, beautiful, friendly
- Have initiative
- Have a sense of humor
- Tell right from wrong
- Make mistakes
- Fall in love
- Enjoy strawberries and cream
- Learn from experience
- Use words properly
- Do something really new
- ...

Early Successes

General Problem Solver (GPS) (1957) :

- Imitate human problem-solving protocols (applied mainly on puzzles)
- First program to embody the "thinking humanly" approach
- Given a current state, what to do to reach the goal state

Geometry Theorem Prover (1958) was able to prove theorems that many students of mathematics would find quite tricky.

Arthur Samuel's checkers (1959) reinforcement learning like method which proves that computers can learn and not only follow a list of instructions.

Early Successes and First AI Winter

- **Perceptron** (1957) simulation
- **LISP** (1958) AI programming language
- **SAINT** (1963) solved closed-form calculus integration problems
- **ANALOGY** (1968) solved geometric analogy problems
- **STUDENT** (1964) solved algebra story problems

Reasons for Failure

- Programs didn't really understand the task
- Intractability
- Representational limits of the tools being used

Expert Systems (1970s)

Domain-specific AI programs that mimics decision-making of human experts by using a knowledge base of facts and logical if-then rules.

- **DENDRAL** (1965) chemical analysis
- **MYCIN** (1972) system for diagnosing blood infections

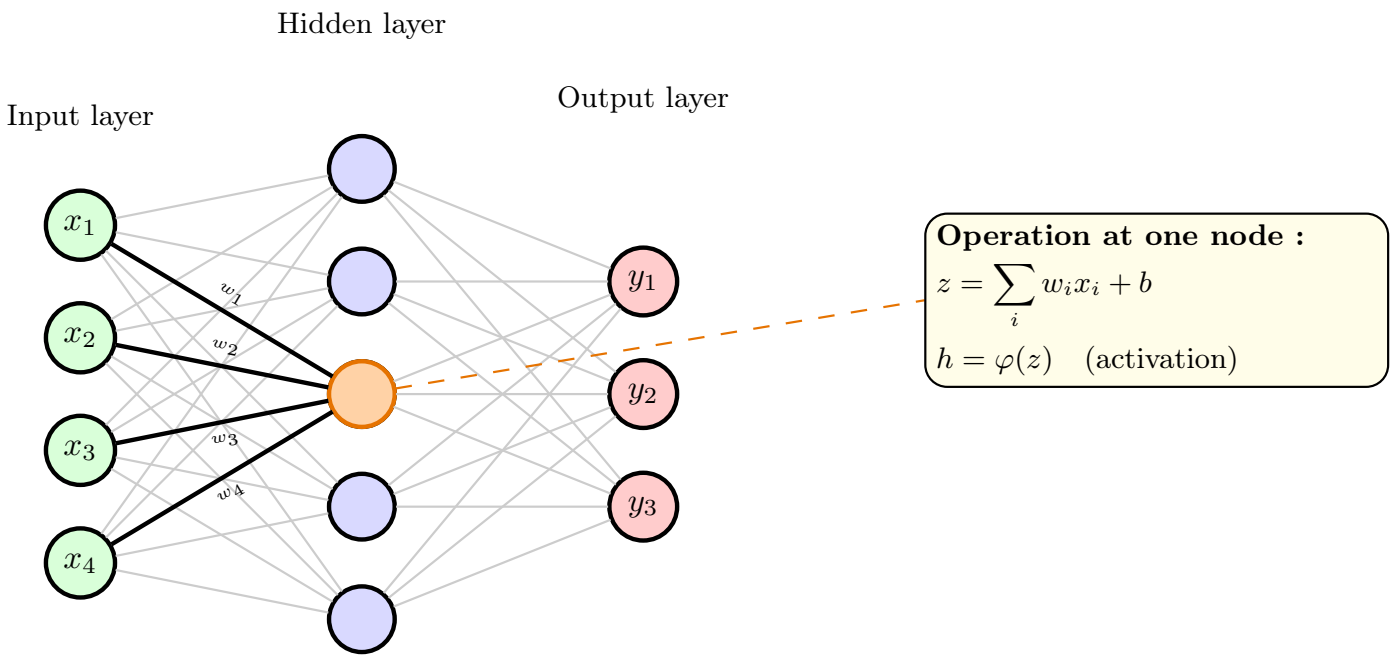
Advantages : Non-black-box, we can trace back the reasoning

Disadvantages :

- Limited to narrow, well-defined problem areas
- To solve a hard problem, we have to almost know the answer already
- It is difficult to build and maintain expert systems for complex domains

Companies failed to deliver on extravagant promises which led to the second AI winter.

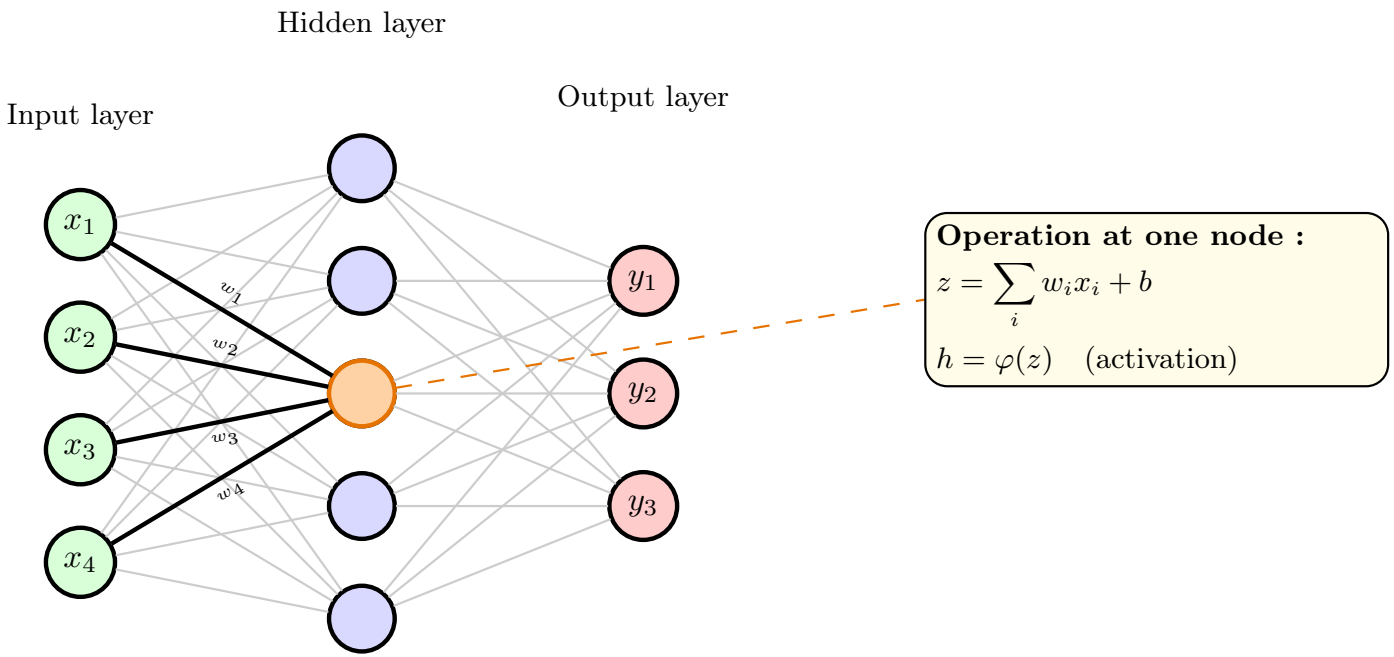
Return of Neural Networks



Neural Networks are a system composed of many simple processing elements operating in parallel, whose function is determined by the network structure, connection strengths, and the processing performed at the computing elements or nodes.

Definition from the *DARPA Neural Network Study* (1988) [5]

Return of Neural Networks



What Changed from the 1960s is the backpropagation of error : a learning procedure capable to adjust the weights. Networks with more than one layer can now be trained.

Machine Learning (since late 1980s)

Machine Learning is the field of study and development of statistical algorithms that can learn from data and generalize to unseen data.

- No more hard-coded rules, but instead a system that can learn from examples
- No more boolean logic, but instead probabilistic reasoning

Example of machine learning tasks and algorithms :

Machine Learning (since late 1980s)

Machine Learning is the field of study and development of statistical algorithms that can learn from data and generalize to unseen data.

- No more hard-coded rules, but instead a system that can learn from examples
- No more boolean logic, but instead probabilistic reasoning

Example of machine learning tasks and algorithms :

- Classification : Decision Trees, Logistic Regression, Neural Networks, ...
- Regression : Linear Regression, Neural Networks, ...
- Dimensionality Reduction : Autoencoders, Principal Component Analysis, ...
- Clustering : K-means, ...
- ...

Definition from Wikipedia, "Machine learning" [21]

Big Data

- Advances in computing and the Web enabled creation of massive "big data" sets (text, images, speech/video, etc.)
- Most examples in these data sets are unlabeled, yet learning algorithms can still extract high accuracy from them
- Sometimes increasing data set size by orders of magnitude improves performance more than algorithmic improvements do
- Big data (combined with the shift to machine learning) helped revive AI's commercial appeal

Problems with Big Data but also data in general :

- Sometimes data quality and algorithm design matter more than volume
- Large unlabeled data sets can encode systematic biases
- Many tasks still need labeled data, human annotation is costly
- Privacy concerns and regulations (like GDPR) limit data collection and use

Deep Learning (since early 2010s)

Learning with artificial neural networks, but :

- Potentially a large number of layers
- A layer is not necessarily fully connected to the next layer : in convolutional neural networks, a small neighborhood of each layer is connected to the next
- Not necessarily feedforward : in recurrent neural networks, there can be backward links ; in residual neural networks, connections that skip layers
- Potentially complex architecture (layer organization)
- Fast training made possible by major advances in computing power on specialized chips (GPU, then TPU)

Large Language Models and AI Chatbots

Large language models (LLMs), models often trained on the task of predicting the most probable next word.

Chatbots are :

- Conversational agents interacted with via a Web interface or application
- Their underlying foundation are fine-tuned LLMs training for text generation given a prompt (hidden pre-prompt + conversation history + user's latest message)
- They are often connected to other models allowing image, audio file, etc. inputs
- Potentially also connected on output to other tools : code execution, image generation, etc.

Chapter 1: Definitions and Foundations

Definitions

Foundations of Artificial Intelligence

History of Artificial Intelligence

Current Expectations of Artificial Intelligence

Intelligent System Expectations

Intelligent System Expectations

- Know the world (common sense, general knowledge, ability to acquire specific knowledge) and structure this knowledge
- Reason, apply rules, perform deductions, possibly under uncertainty
- Solve complex problems
- Plan a complex strategy
- Generalize from examples, detect patterns, classify objects, predict
- Perceive the world (understanding of text, structured documents, speech, images, video...)
- Generate content (text, images, speech...)
- Collaborate with other intelligent agents

All of them in a potentially integrated way

The Tasks are Very Different

- We often do not use the same models, tools, and techniques to represent the world, generate content, predict a behavior, collaborate, etc.
- The computer scientists, mathematicians, and physicists behind the advances in these techniques tend to position their work within precise scientific fields rather than claim to be working "in AI" :
  - knowledge representation
  - computer vision
  - natural language processing
  - reasoning, automated theorem proving
  - machine learning, classification
  - planning
  - constraint solving
  - multi-agent systems
  - ...

More Expectations of Intelligent Systems

- We also expect an intelligent system to be capable of :
  - performing calculations
  - efficiently sorting data
  - handling large volumes of data
  - applying simple, fully specified algorithms
  - ...
- but none of this has anything specific to do with imitating or reproducing intelligence – it is the basis of computer science : we do not include this among the tasks of AI.

Risks of Artificial Intelligence

Benefits : accelerating scientific research, freeing humans from repetitive work, increasing production of goods and services...

Risks :

- Lethal autonomous weapons
- Mass surveillance and behavioral manipulation
- Biased decision making (parole, loans, hiring...)
- Impact on employment and inequality
- Safety-critical failures (self-driving cars, infrastructure)
- Cybersecurity (AI-powered attacks, phishing, malware)

Chapter 2: Optimization

Motivation and Introduction to Optimization

Convexity

Linear Programming

Gradient Descent

More Constrained Optimization Solutions

What is an Optimization Problem ?

What is an Optimization Problem ?

In its simplest form, an optimization problem involves minimizing or maximizing an objective function $f(x)$ with respect to a set of decision variables $x \in \mathbb{R}^n$:

$$\min_{x \in \mathbb{R}^n} f(x)$$

What is an Optimization Problem ?

In its simplest form, an optimization problem involves minimizing or maximizing an objective function $f(x)$ with respect to a set of decision variables $x \in \mathbb{R}^n$:

$$\min_{x \in \mathbb{R}^n} f(x)$$

We say that f reaches a global minimum in a point x^* of $\mathbb{R}^n$ if, for any $x \in \mathbb{R}^n$, we have $f(x^*) \leq f(x)$.

Simple Examples :

What is an Optimization Problem ?

In its simplest form, an optimization problem involves minimizing or maximizing an objective function $f(x)$ with respect to a set of decision variables $x \in \mathbb{R}^n$:

$$\min_{x \in \mathbb{R}^n} f(x)$$

We say that f reaches a global minimum in a point x^* of $\mathbb{R}^n$ if, for any $x \in \mathbb{R}^n$, we have $f(x^*) \leq f(x)$.

Simple Examples :

1. Optimal field crossing : minimize the time it takes to cross a field by choosing the best path.
   Solve : 2×1 *km field, speed* 5 *km/h on grass,* 10 *km/h on road*
2. Data fitting : find the best line that fits a set of data points.
   Solve : $y = ax + b$ *for given data points (0,0),(1,1),(2,3)*

Sometimes the answer is not straightforward : We use numerical approximations

Examples of Algorithms for One-dimensional Optimization :

- Newton's Algorithm
- Dichotomy
- Quadratic interpolation

Optimization under Constraints

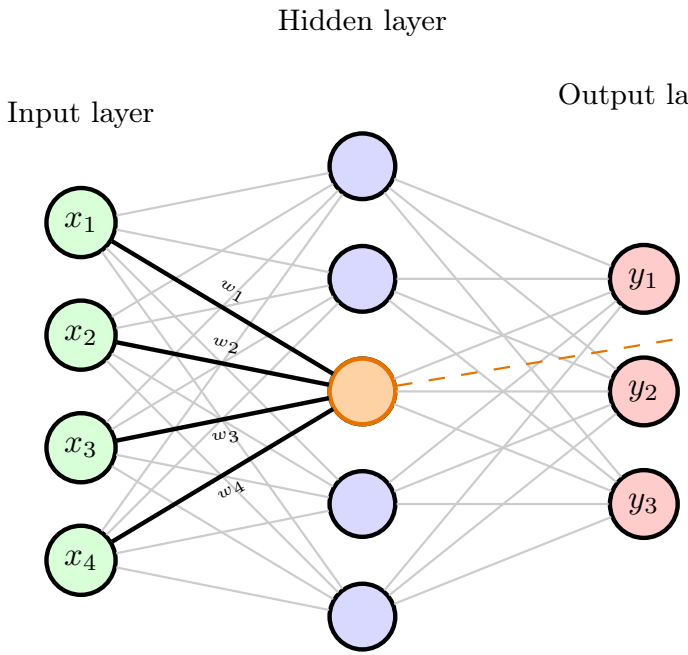
In general, a constrained optimization problem over a variable x may be written in the following standard form :

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ \text{subject to } &g_i(x) \leq 0, \ i = 1, ..., m, \\ \text{and } &h_j(x) = 0, \ j = 1, ..., p. \end{aligned} \tag{1}$$

The inequality and equality constraints determine a feasible set $\mathcal{S}$ of admissible values of x.

Example : for $x \in \mathbb{R}$, $f(x) = x^2$ and $g(x) = 1 - x$.

Optimization in Artificial Intelligence

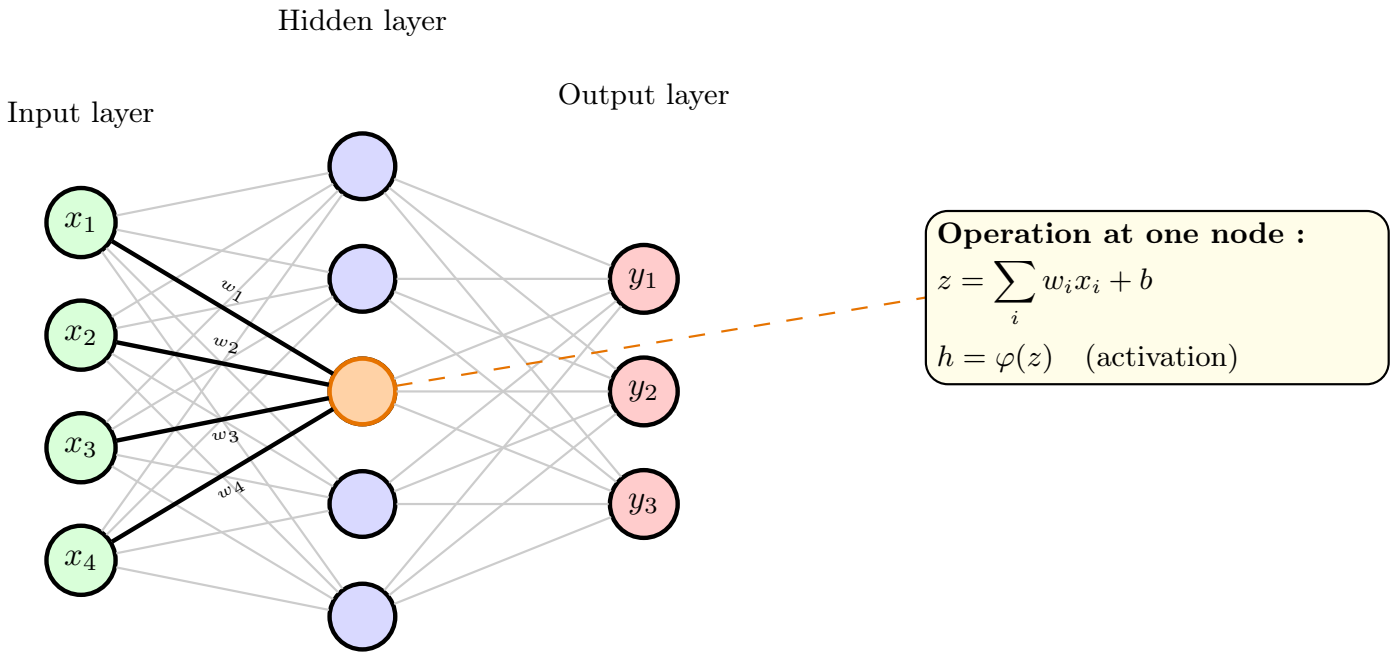


Operation at one node :

$$z = \sum_i w_i x_i + b$$

$$h = \varphi(z) \quad \text{(activation)}$$

Optimization in Artificial Intelligence



- The network's prediction is compared to the true answer, giving an **error** function : $\mathcal{L}(w)$.
- Training a neural network : solving $\min_w \mathcal{L}(w)$
- How do we actually find the w that minimizes the error ?

Chapter 2: Optimization

Motivation and Introduction to Optimization

Convexity

Linear Programming

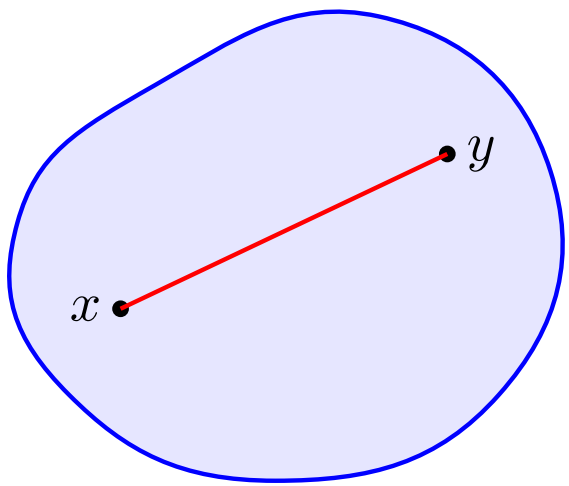
Gradient Descent

More Constrained Optimization Solutions

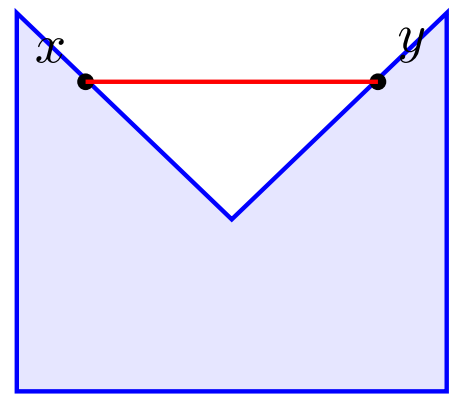
Convex Sets

A set $C \subseteq \mathbb{R}^n$ is **convex** if for all $x, y \in C$ and $\lambda \in [0, 1]$:

$$\lambda x + (1 - \lambda) y \ \in \ C$$



Convex

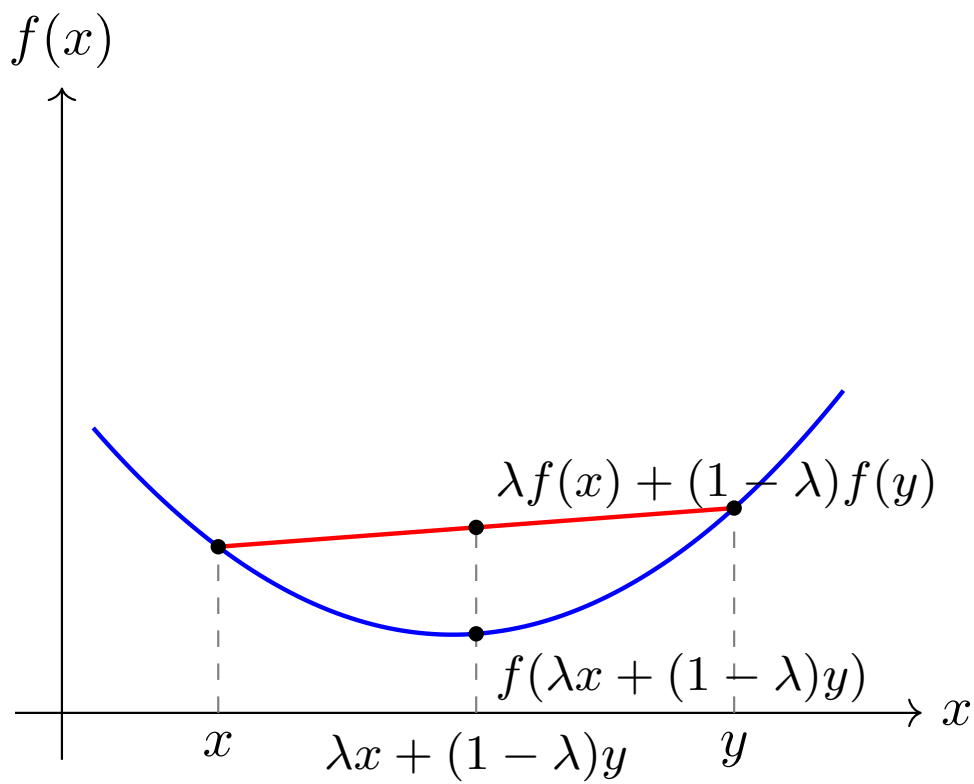


Non-convex

Convex Functions

A function $f : C \subset \mathbb{R}^n \to \mathbb{R}$ is **convex** if its domain is a convex set and for all $x, y \in \text{dom}(f)$ and $\lambda \in [0, 1]$:

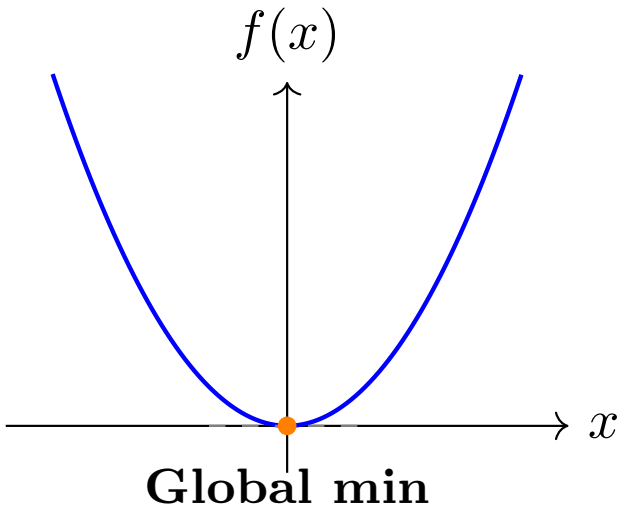
$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

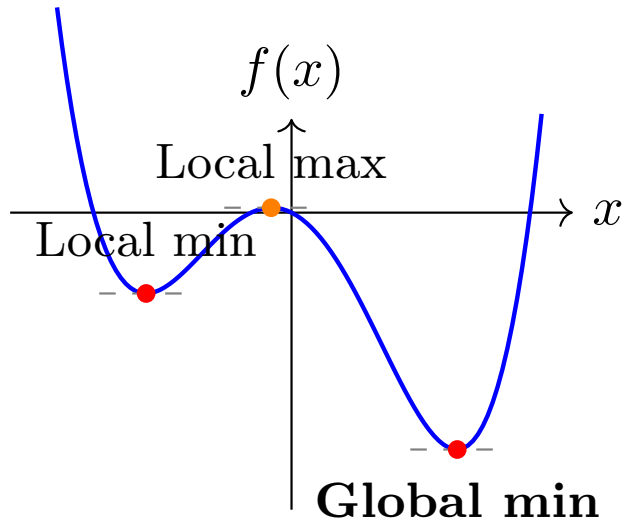


Why Convexity Matters

Suppose we want to find the minimum of these functions

Construct an algorithm to find the minimum of f with only having access to the images of a given x





- When the objective function f and the feasible set $\mathcal{S}$ are both convex, then we have a convex optimization problem
- Convex optimization problems come with strong theoretical guarantees and efficient computational algorithms

Why Convexity Matters

- Non-convex problems may have multiple local minima, saddle points, etc.
- A local minimum of a convex optimization problem is also a global minimum (proof by contradiction)
- If the optimization problem is strictly convex, then the minimum is unique
- Same guarantees for maximization problems when f is concave and the feasible set is convex.

Example : Linear programming is a convex optimization problem

About Convex Functions

- The sum of convex functions is a convex function (useful when we add terms to the objective functions)
- Norms are convex
- If a function is convex then its opposite is concave

Most Machine Learning Problems Are Nonconvex

Machine learning models are trained using optimization algorithms to find the parameters that best fit the training data.

- Most machine Learning optimization problems are high-dimensional and nonconvex
- They have many local minima
- There are few generic guarantees for convergence or global optimality in nonconvex optimization

Chapter 2: Optimization

Motivation and Introduction to Optimization

Convexity

Linear Programming

Gradient Descent

More Constrained Optimization Solutions

What Makes a Problem Linear ?

A linear optimization problem could be rewritten in a matrix form :

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} c^T x \\ &\text{subject to } A \cdot x \leq b \end{aligned} \tag{2}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. So we have m inequality constraints. It is also a common constraint that the variables are non-negative i.e. $x_i \geq 0$

Applications

What is linear programming used for :

- Resource allocation and workforce scheduling
- Factory layout, space usage, production scheduling, and inventory management
- Portfolio optimization, risk management, and pricing
- Airline flight, crew, and maintenance scheduling
- ...

Simple Example

A factory produces two kinds of products, p_1 and p_2, using two machines m_1 and m_2.
A product p_1 goes through m_1 for 30 minutes and m_2 for 40 minutes.
A product p_2 goes through m_1 for 20 minutes and m_2 for 10 minutes.
The machine m_1 is available 6000 minutes per month.
The machine m_2 is available 4000 minutes per month.
The profit realized on a unit of the product p_1 is 400 euros.
The profit realized on a unit of the product p_2 is 200 euros.

Simple Example

A factory produces two kinds of products, p_1 and p_2, using two machines m_1 and m_2.
A product p_1 goes through m_1 for 30 minutes and m_2 for 40 minutes.
A product p_2 goes through m_1 for 20 minutes and m_2 for 10 minutes.
The machine m_1 is available 6000 minutes per month.
The machine m_2 is available 4000 minutes per month.
The profit realized on a unit of the product p_1 is 400 euros.
The profit realized on a unit of the product p_2 is 200 euros.

$$\begin{aligned} \text{Maximize} \quad & z = 400x_1 + 200x_2 \\ \text{subject to :} \quad & 30x_1 + 20x_2 \leq 6000 \\ & 40x_1 + 10x_2 \leq 4000 \\ & x_1 \geq 0, \quad x_2 \geq 0 \end{aligned} \tag{3}$$

Feasible Region as a Polytope : Why the Optimum Sits at a Corner

- The set of points in $\mathbb{R}^n$ of coordinates $x_1, ..., x_n$ fulfilling the $m + n$ constraints is a polyhedron called constraints polyhedron
- This polyhedron is convex
- The problem admits a maximum reached in at least one vertex of the constraints polyhedron

Solutions

Identifying vertices is challenging in higher dimensions

- **Simplex Method :** search from vertex to vertex in directions that increase the objective function
- **Interior Point Methods (IPMs) :** traverse the interior of the feasible region rather than explore the boundary

Chapter 2: Optimization

Motivation and Introduction to Optimization

Convexity

Linear Programming

Gradient Descent

More Constrained Optimization Solutions

The Gradient

The gradient of a function $f(\mathbf{x})$ where $x \in \mathbb{R}^n$ is defined for continuously differentiable functions as :

$$\nabla f = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}$$

$\dfrac{\partial f}{\partial x_i}$ measures how much f changes when the component x_i is varied.

Finding the Minimum

If x^* is an unconstrained minimum and f is differentiable at x^*, then $\nabla f(x^*) = 0$.

Some examples to be careful of : $\min_x \sqrt{x}$, $\min_x x^3$, $\min_x x$.

Example : $E(a, b) = 5a^2 + 6ab + 3b^2 - 14a - 8b + 10$

Finding the Minimum

If x^* is an unconstrained minimum and f is differentiable at x^*, then $\nabla f(x^*) = 0$.

Some examples to be careful of : $\min_x \sqrt{x}$, $\min_x x^3$, $\min_x x$.

Example : $E(a,b) = 5a^2 + 6ab + 3b^2 - 14a - 8b + 10$ Gradient :

$$\nabla E(a,b) = \begin{pmatrix} \frac{\partial E}{\partial a} \\ \frac{\partial E}{\partial b} \end{pmatrix} = \begin{pmatrix} 10a + 6b - 14 \\ 6a + 6b - 8 \end{pmatrix}$$

Solve $\nabla E(a,b) = 0$:

$$10a + 6b = 14 \quad (1)$$

$$6a + 6b = 8 \quad (2)$$

So we get $a = \frac{3}{2}$ and $b = -\frac{1}{6}$.

Candidate : $(a^*, b^*) = \left(\frac{3}{2}, -\frac{1}{6}\right)$, with $E(a^*, b^*) = \frac{1}{6}$.

We can also do it Numerically

Finding points x^* where $\nabla f(x^*) = 0$ could be challenging.

We can use iterative methods to obtain a sequence $(x_k)_{k\in\mathbb{N}}$ that converges to the local minimum x^* : We seek an algorithm $\mathcal{A}$:

$$x_{k+1} = \mathcal{A}(x_k)$$

such that starting with an initial guess x_0 give us :

$$f(x_0) \geq f(x_1) \geq f(x_2) \geq \cdots \geq f(x_k) \geq f(x^*)$$

Gradient Descent

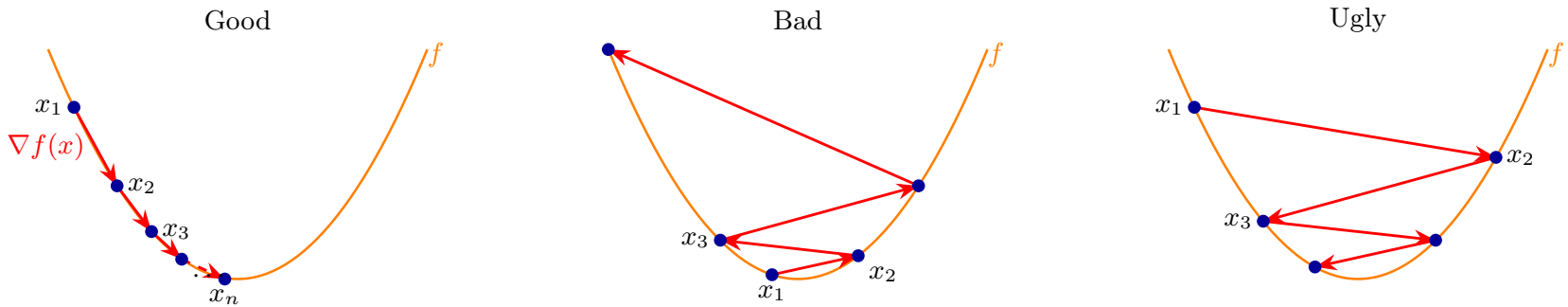
- The gradient $\nabla f(x)$ points uphill : steepest increase
- $-\nabla f(x)$ points downhill : steepest decrease
- So we can build the gradient descent algorithm :

$$x_{k+1} = x_k - \alpha \cdot \nabla f(x_k)$$

We call α the step size, it is also referred to as the **learning rate** in machine learning.

How to choose a step size

Step size should not be too small or too big :



- **Fixed** : stays the same during the whole algorithm
- **Decaying** : decreasing the step size with each iteration
- **Adaptive methods** : good in machine learning
- ...

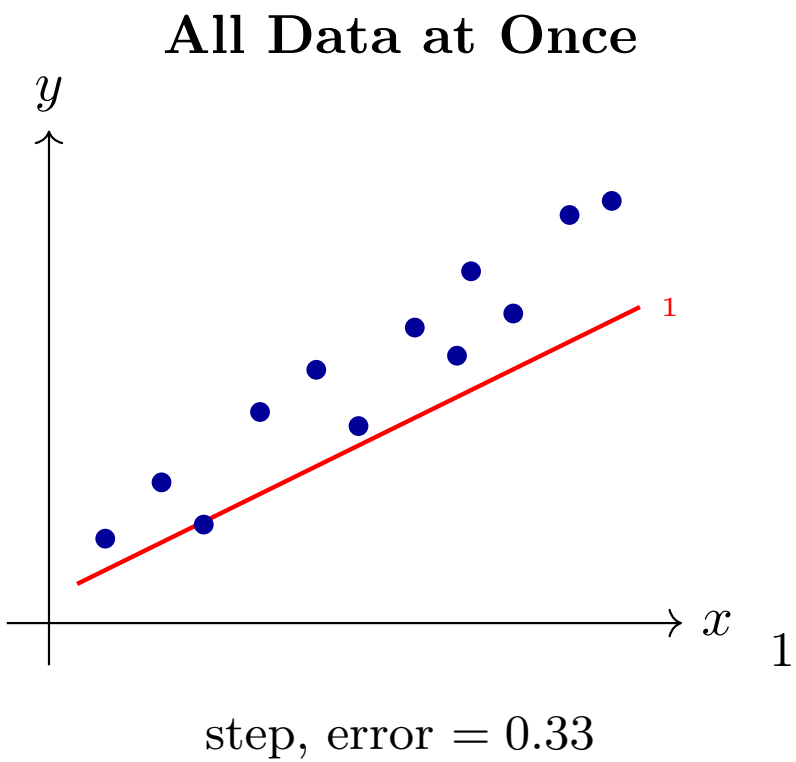
Scaling Problem with Gradient Descent

In machine learning, an objective function minimizing error on all data points has computational burdens :

- **Memory constraints** : the entire dataset may not fit in memory
- **Streaming/online data** : new data may arrive continuously, so the whole dataset may not be fixed or fully available

Stochastic Gradient Descent (SGD) : Approximating the gradient at each step given a mini-batch of data

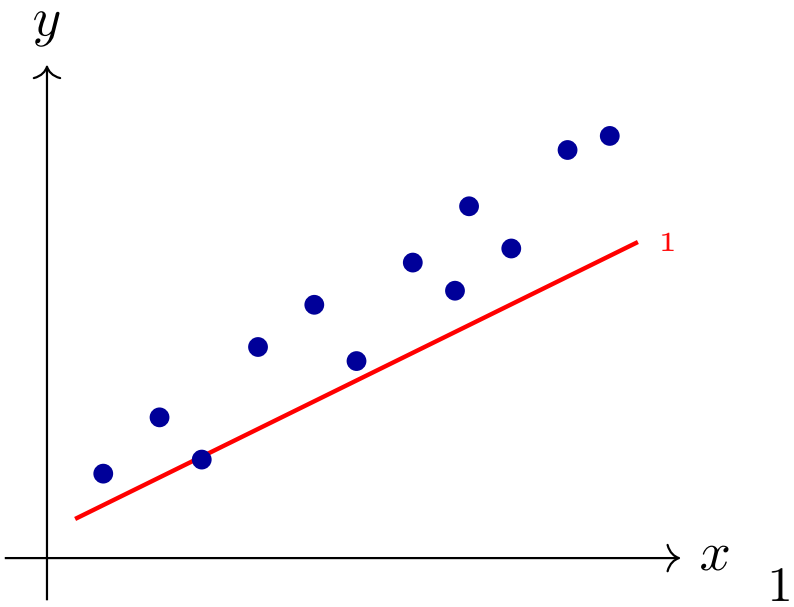
Example : Full-Batch vs. Mini-Batch Data Fitting



step, error = 0.33

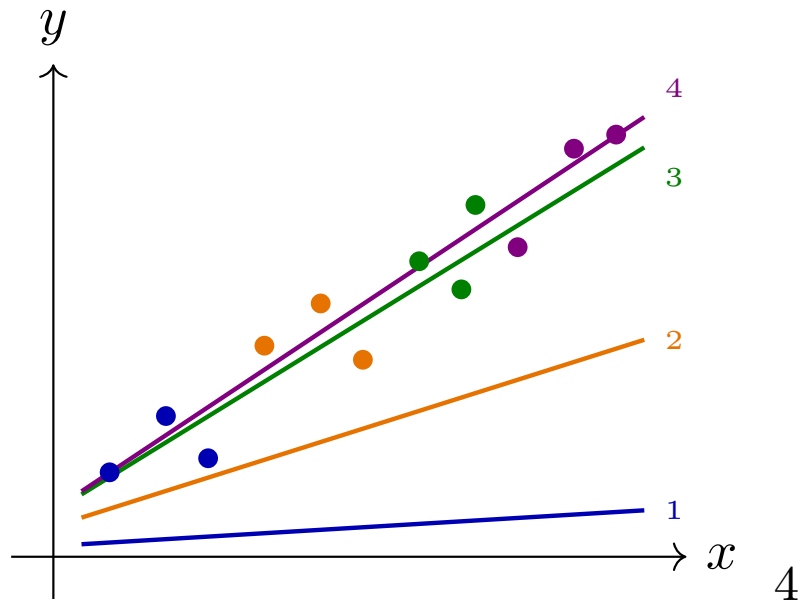
Example : Full-Batch vs. Mini-Batch Data Fitting

All Data at Once



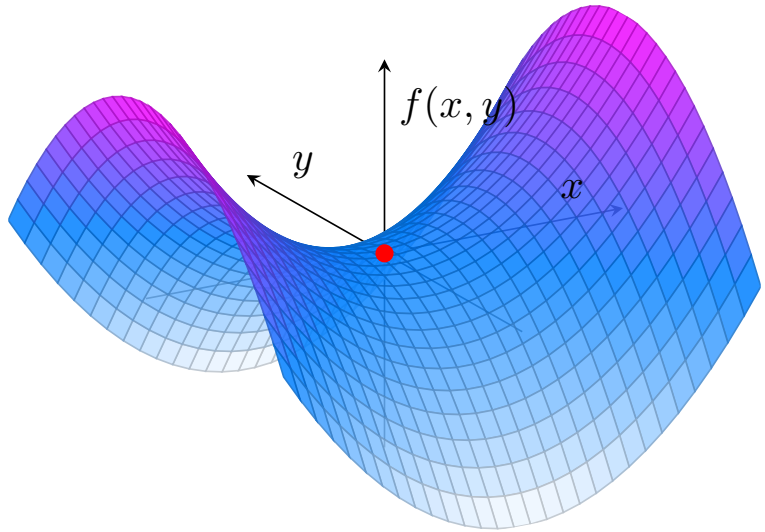
step, error = 0.33

Split into Batches



mini-steps, error = 0.055

Converging to a Saddle Point



- Problem : Mini-batch fitting (SGD) can cause the convergence to become zig-zaggy
- Problem : The gradient near saddle points is small
- Solution : We need to give the algorithm inertia to carry through these flat saddle regions of small gradient

Heavy Ball Momentum :

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1})$$

ADAM : Adaptive Moment Estimation

- ADAM uses a momentum term as in heavy ball momentum
- ADAM scales the step size (learning rate) with a moving average of past squared gradients

Adam is the default starting point for most deep learning today.

Composition and the Chain Rule

Take a function f built by composing several simpler functions :
In one dimension :

$$(f(g(x)))' = f'(g(x)) \cdot g'(x)$$

In multiple dimensions :

$$x \xrightarrow{f_1} z_1 \xrightarrow{f_2} z_2 \xrightarrow{f_3} \cdots \xrightarrow{f_L} z_L = f(x)$$

so that $z_i = f_i(z_{i-1})$, with $z_0 = x$. The derivative can be computed as :

$$\frac{df}{dx} = \frac{dz_L}{dz_{L-1}} \cdot \frac{dz_{L-1}}{dz_{L-2}} \cdots \frac{dz_2}{dz_1} \cdot \frac{dz_1}{dx}$$

Chapter 2: Optimization

Motivation and Introduction to Optimization

Convexity

Linear Programming

Gradient Descent

More Constrained Optimization Solutions

We saw that :

- We can optimize a linear function with linear constraints
- Gradient descent can help us converge to a minimum of a function through iterations
- We can adjust the algorithm to have a smooth descent that converges better

Problem : How to optimize a non-linear objective function with constraints ?

Lagrange Multipliers

Problem : minimize $f(x)$ subject to $h(x) = 0$

Solve :

$$\begin{cases} \nabla f(x) + \lambda \nabla h(x) = 0 \\ h(x) = 0 \end{cases}$$

then compare f at the solutions.

Example : minimize $f(x, y) = xy$ on $x^2 + y^2 = 4$

Solution

System :

$$\begin{cases} y + 2\lambda x = 0 \\ x + 2\lambda y = 0 \\ x^2 + y^2 = 4 \end{cases}$$

Solve : From the first two equations, $y = -2\lambda x$ and $x = -2\lambda y$, so

$$x = 4\lambda^2 x.$$

If $x = 0$, then $y = 0$, which violates $x^2 + y^2 = 4$. Hence $\lambda^2 = \frac{1}{4}$, so $\lambda = \pm\frac{1}{2}$.

- $\lambda = \frac{1}{2}$: $y = -x$, so $2x^2 = 4$, giving $(\sqrt{2}, -\sqrt{2})$ and $(-\sqrt{2}, \sqrt{2})$, with $f = -2$
- $\lambda = -\frac{1}{2}$: $y = x$, so $2x^2 = 4$, giving $(\sqrt{2}, \sqrt{2})$ and $(-\sqrt{2}, -\sqrt{2})$, with $f = 2$

Minimum : $f = -2$ at $(\sqrt{2}, -\sqrt{2})$ and $(-\sqrt{2}, \sqrt{2})$.

Other Methods

These methods also assume certain conditions on the objective and constraints.

- **Karush–Kuhn–Tucker (KKT)** :
  Extends Lagrange multipliers to inequality constraints
- **Descent Method** :
  Iteratively choose a feasible point with $f(x_{k+1}) < f(x_k)$
- **Linearization** :
  Approximate f at x_k by $L_{x_k}(x) = f(x_k) + \nabla f(x_k)^\top (x - x_k)$ and minimize it over the feasible domain to get x_{k+1}

Chapter 3: Logic, Knowledge Representation and Reasoning

Definitions

Logic

Motivation

“How can a system know what to do if it faces a situation it wasn’t explicitly programmed for ?”

Motivation

"How can a system know what to do if it faces a situation it wasn't explicitly programmed for ?"

We want to build systems that reason intelligently, so we need a way to encode real-world facts.

- How to represent these facts about the real world ?
- How to reason about these facts ?
- Which representations are appropriate for dealing with the real world ?

Knowledge

Knowledge : Information and facts about the real world.
Examples :
Water boils at 100 degrees Celsius
The sky is blue

Types of knowledge :

- Declarative knowledge : facts about who, what, when, . . .*knowing that . . .*
- Procedural knowledge : facts about how *knowing how . . .*
- Affective knowledge : to know what grief feels like, . . .
- . . .

In this chapter, we focus on **declarative knowledge**.

Representation

Symbols are used to represent things in the real world.

- Burger for fast food restaurants
- $3 \leq 5$
- $human \rightarrow mortal$
- ...

Representation : Use of one thing (like a symbol, image, or word) to stand for something else, making it easier to understand, use, or communicate.

Representation

Symbols are used to represent things in the real world.

- Burger for fast food restaurants
- $3 \leq 5$
- $human \rightarrow mortal$
- ...

Representation : Use of one thing (like a symbol, image, or word) to stand for something else, making it easier to understand, use, or communicate.

Knowledge representation :

- Symbolic encoding of propositions (facts) believed by some system.
- Structuring and organizing that encoding in a way that computers can store, process, and use (*Logical statements, graphs, etc.*)

Why do we need Reasoning ?

Is knowing facts alone enough ?

Why do we need Reasoning ?

Is knowing facts alone enough ?
No, reasoning computes new beliefs by inferring from represented facts.

Why do we need Reasoning ?

Is knowing facts alone enough ?
No, reasoning computes new beliefs by inferring from represented facts.

- Don't want : "do action A if P is in the list of instructions"
- Want : "do action A if our beliefs about the world satisfy P"

Example :

- "Patient x is allergic to medication m."
- "Anyone allergic to m is also allergic to m'."

  Is it safe to prescribe m' to x ?

$\Rightarrow$ "Do NOT prescribe m' to x."

Reasoning

Reasoning : Manipulation of symbols encoding propositions to produce representations of new propositions.

Examples :

- Symbols $\mathbb{N} \subseteq \mathbb{Z}$ and $\mathbb{Z} \subseteq \mathbb{R}$ then $\mathbb{N} \subseteq \mathbb{R}$
- Symbols "All humans are mortal" and "Alan is human" then "Alan is mortal"

Reasoning could be seen as a form of calculation, but instead of numbers, we manipulate symbols representing propositions.

For reasoning to be trustworthy, the new symbol must represent a true fact whenever the symbols we started from were true.

Forms of Reasoning

- **Inductive reasoning**
  Build general knowledge from particular observations
  Every bird I've seen has feathers, and wings. All birds have feathers and wings.

- **Deductive reasoning**
  Use general knowledge on particular cases to infer new facts
  All humans are mortal and Alan is human, so Alan is mortal.

- **Abductive reasoning** Use general knowledge and incomplete observations to get the most plausible conclusion
  The floor is wet and there's a dripping umbrella by the door. It must have rained and someone brought the umbrella in.

In Knowledge Representation we focus mostly on deductive reasoning

Knowledge Representation and Reasoning

For a system to use knowledge intelligently, its knowledge representation must :

1. encode information about the world or the task
2. be usable by the system to reason, make decisions, or act.

Example : ColorOf() vs ColorOf() and MadeOf()

Knowledge Representation and Reasoning

For a system to use knowledge intelligently, its knowledge representation must :

1. encode information about the world or the task
2. be usable by the system to reason, make decisions, or act.

Example : ColorOf() vs ColorOf() and MadeOf()

Knowledge base (KB) : the stored facts and rules.
Knowledge-based system : a KB plus a **reasoner** that uses it to decide or act.

Good Knowledge Representations in Practice

A good representation should :

- Be fast/tractable, not just possible
- Have unambiguous notation, with a precise meaning
- Be easy for humans to read, write, and understand

Good Knowledge Representations in Practice

A good representation should :

- Be fast/tractable, not just possible
- Have unambiguous notation, with a precise meaning
- Be easy for humans to read, write, and understand

But : most modern AI (deep learning, LLMs) is optimization-based not knowledge based.

Applications of Knowledge-based systems

An expert system is one kind of knowledge-based system : a knowledge base plus a reasoner for one narrow domain.

- **Planning** : robots and logistics describe states and actions logically to find a plan
- **Verification** : logic solvers prove math proofs are correct
- ...

Why do we use Knowledge-Based Systems ?

Explanation and traceability

- "Why is grass green ?" → "Because grass is a form of vegetation, and vegetation is green."
- LLMs could explain but they are not guaranteed to be correct.

Easy to correct/debug

- "No, the sky isn't yellow, it's blue." → just edit in the KB.

Can add new knowledge

- Add a new fact : "Canaries are yellow."
- In deep learning, you would need to retrain the model with new data.

New applications

- Reuse the same KB for a different task, e.g. "list all yellow things."

Chapter 3: Logic, Knowledge Representation and Reasoning

Definitions

Logic

Why Do We Use Logic ?

- What We Need :
  A way to **represent** knowledge and **reason** over it.
- What Logic Offers :
  - **Syntax** : the rules for what counts as a valid formula
  - **Semantics** : the meaning of a formula (true/false)
  - **Inference rules** : rules for deriving new formulas from existing ones (*from* P *and* $P \Rightarrow Q$, *derive* Q)

Multiple Logics exist : propositional, first-order, description, higher-order, etc.

Propositional Logic

What Is It ?

The simplest logic : it deals with whole statements (**propositions**) that are either **true** or **false**, combined using connectives.

Atomic Propositions

A basic statement with no internal structure, it can't be broken down further.

e.g. P : "It is raining" Q : "The grass is wet"

Connectives

$P \wedge Q$	**and**	true when both are true
$P \vee Q$	**or**	true when at least one is true
$\neg P$	**not**	true when P is false
$P \Rightarrow Q$	**if, then**	true except when P is true and Q is false
$P \Leftrightarrow Q$	**iff**	true when both have the same value

Examples of Propositional Logic

Translate to Propositional Logic :

- "Alan can only enter the club if he is over 18 and has a ticket."
- "If it rains, Alan won't walk to school, unless he has an umbrella."

Build a Truth Table :

Build truth tables for each pair of formulas below. Are they equivalent ?

$$
\begin{array}{rcl}
p \Rightarrow q & \text{and} & \neg p \vee q \\
p \wedge (q \vee r) & \text{and} & (p \wedge q) \vee (p \wedge r) \\
p \vee (q \wedge r) & \text{and} & (p \vee q) \wedge (p \vee r) \\
\neg(p \wedge q) & \text{and} & \neg p \vee \neg q \\
p \Rightarrow q & \text{and} & \neg q \Rightarrow \neg p
\end{array}
$$

What about the truth table of : $[(p \Rightarrow q) \wedge p] \Rightarrow q$

Logic exercise I

Solve this problem by responding by true or false to each statement.

p = "Alan is eating pizza"
q = "The sun is shining"
r = "Tomorrow is Monday"

Given that "today is Monday" and that "$(p \wedge q) \Rightarrow r$".

Statement 1 : Alan isn't eating pizza or the sun isn't shining
Statement 2 : If the sun is shining then Alan isn't eating pizza
Statement 3 : If the sun is shining and Alan is eating pizza then a volcano is erupting

Logic exercise II

1. Given the definition of the XOR ($\oplus$) operator below, fill in the truth table and find a way to express $p \oplus q$ in terms of $p, q, \wedge, \vee,$ and $\neg$:

p	q	$p \oplus q$	$p \vee q$	$p \wedge q$
T	T	F		
T	F	T		
F	T	T		
F	F	F		

2. Show that $p \oplus q$ is equivalent to $\neg(p \Leftrightarrow q)$ without using a truth table.
3. How do you interpret in English the result from (2) : $p \oplus q$ is equivalent to $\neg(p \Leftrightarrow q)$.

First-Order Logic

Limitations of Propositional Logic :
Write in propositional logic : "All humans are mortal, Alan is a human, therefore Alan is mortal".

First-Order Logic (FOL) : extends propositional logic with :

- **Objects** : a specific thing in the world : $Alan$, $Paris$
- **Variables** : an unspecified object : x, y
- **Predicates** : a property or relation applied to objects, true or false : $Student(x)$, $Loves(x, y)$
- **Functions** : maps object(s) to another object : $father_Of(x)$, $age_Of(x)$
- **Connectives** : same as propositional logic : $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
- **Universal quantifier** ($\forall$) : "for all."
- **Existential quantifier** ($\exists$) : "there exists."

Examples

- "All students are humans"

$$\forall x\, Student(x) \Rightarrow Human(x)$$

- "Some students are not humans"

$$\exists x\, Student(x) \wedge \neg Human(x)$$

- "All humans are mortal" and "Alan is human" so "Alan is mortal."

$$\big((\forall x\, Human(x) \Rightarrow Mortal(x)) \wedge Human(Alan)\big) \Rightarrow Mortal(Alan)$$

- "To enter the club, you must be over 18 and have a ticket."

$$\forall x\, Enter(x) \Rightarrow (Over18(x) \wedge HasTicket(x))$$

- "If it rains, nobody walks to school unless they have an umbrella."

$$\forall x\, (Rain \wedge \neg HasUmbrella(x)) \Rightarrow \neg Walks(x)$$

Logic exercise III

Translate the following sentences into first-order Logic :

- "Horses run faster than dogs"
- "There is no largest prime number"
- "There exists a largest prime number"
- "No number is both even and odd"

Limitations of First-Order Logic

- **Can't quantify over predicates :**
  FOL only quantifies over objects, never over properties
  "Whatever is true for all cats is also true for all dogs"
  this requires higher-order logic.
- **Reasoning can be very costly**
- **Only handles deduction**

Chapter 4: Machine Learning

Motivation

Data

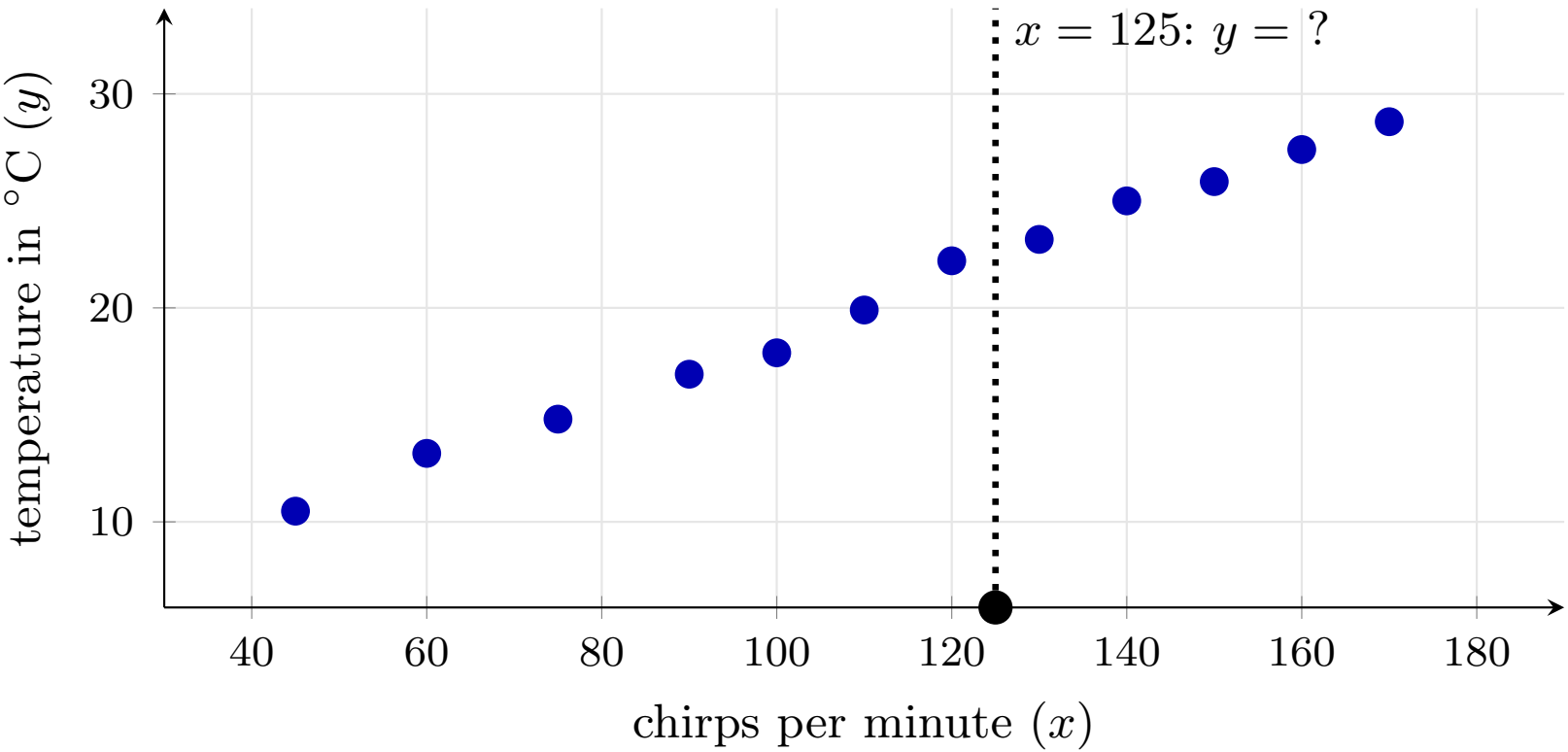
Definitions

Supervised learning

Unsupervised learning

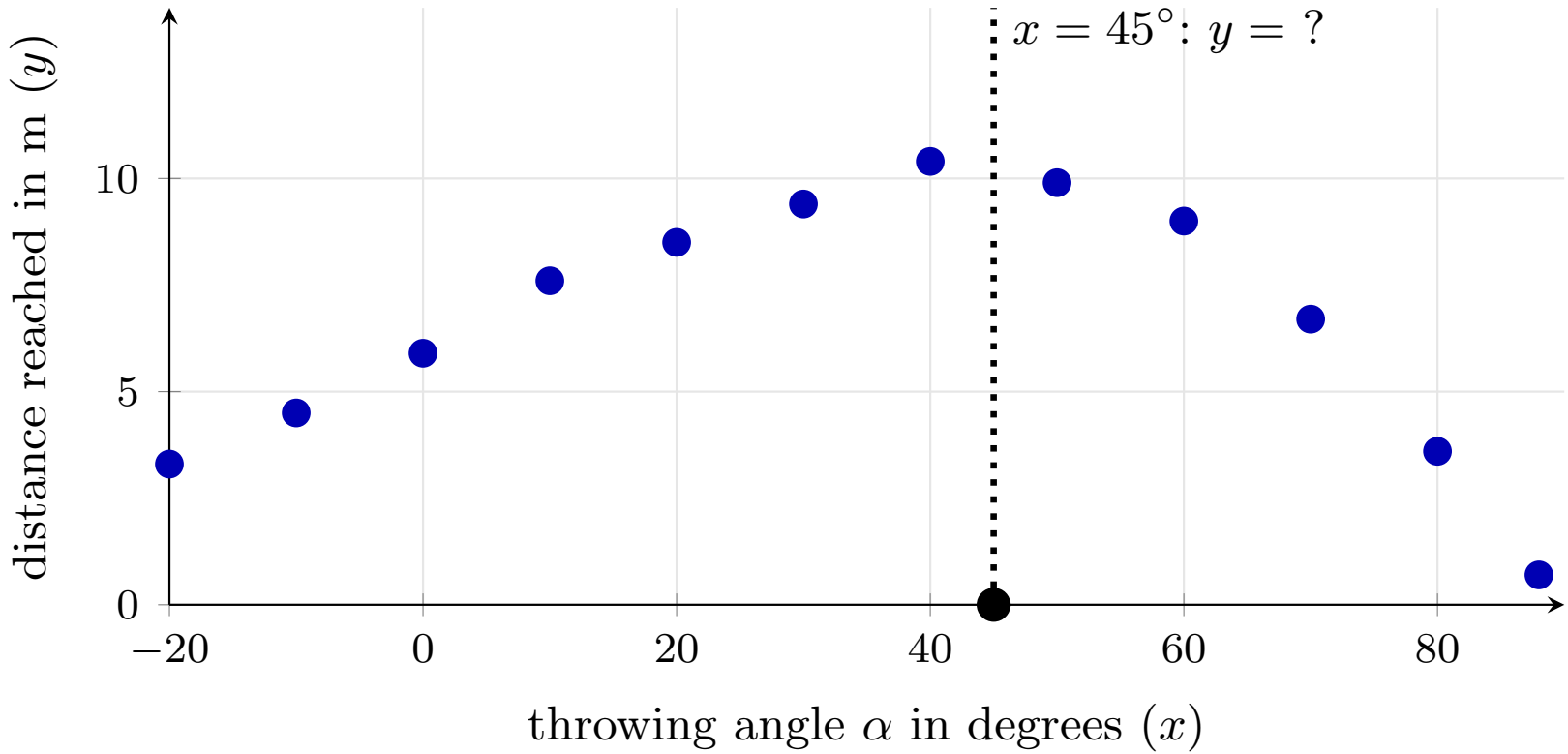
Performance metrics

Learning from examples : crickets



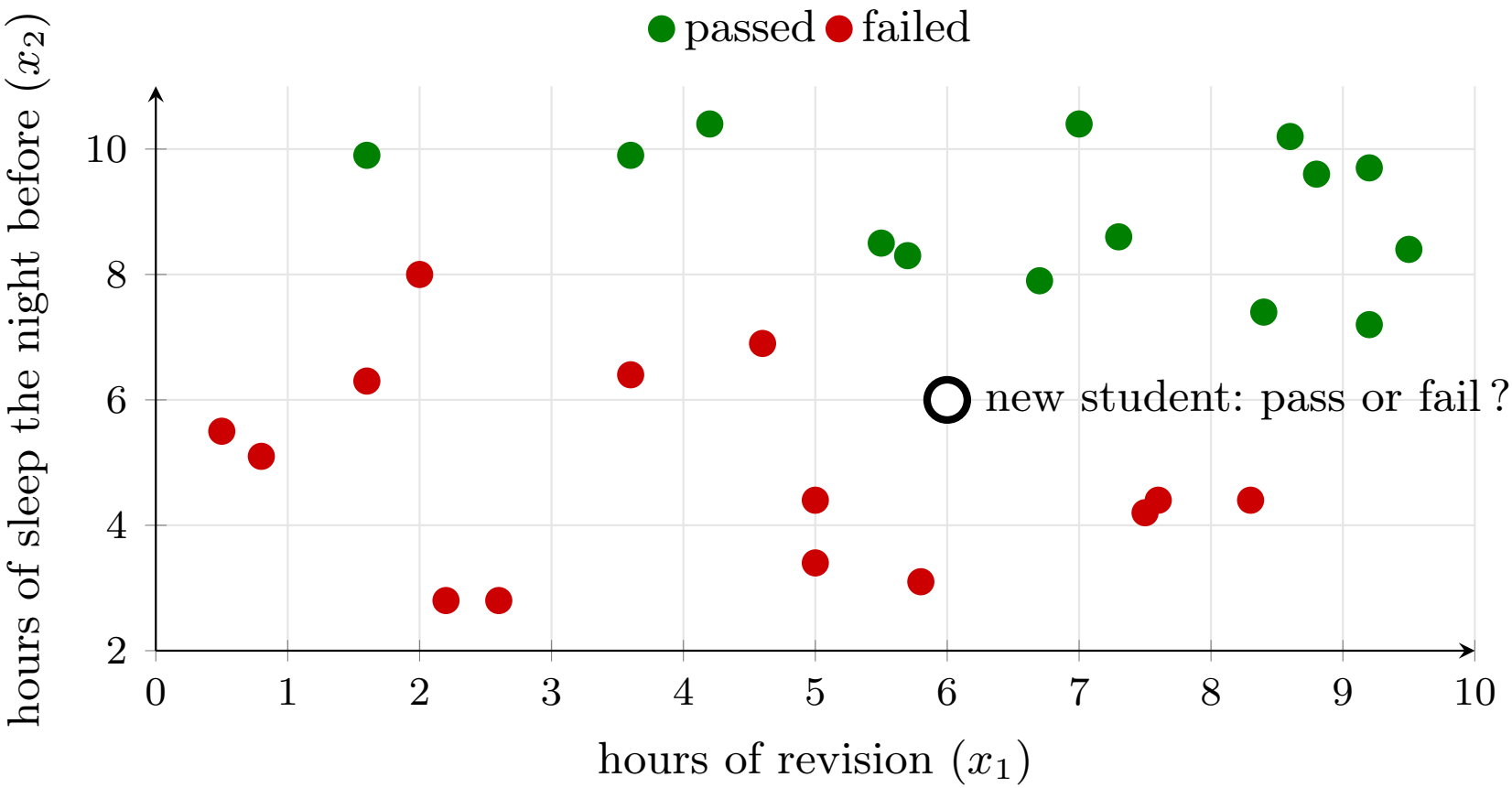
Tonight we hear 125 chirps per minute : how warm is it ?

Learning from examples : throwing a ball



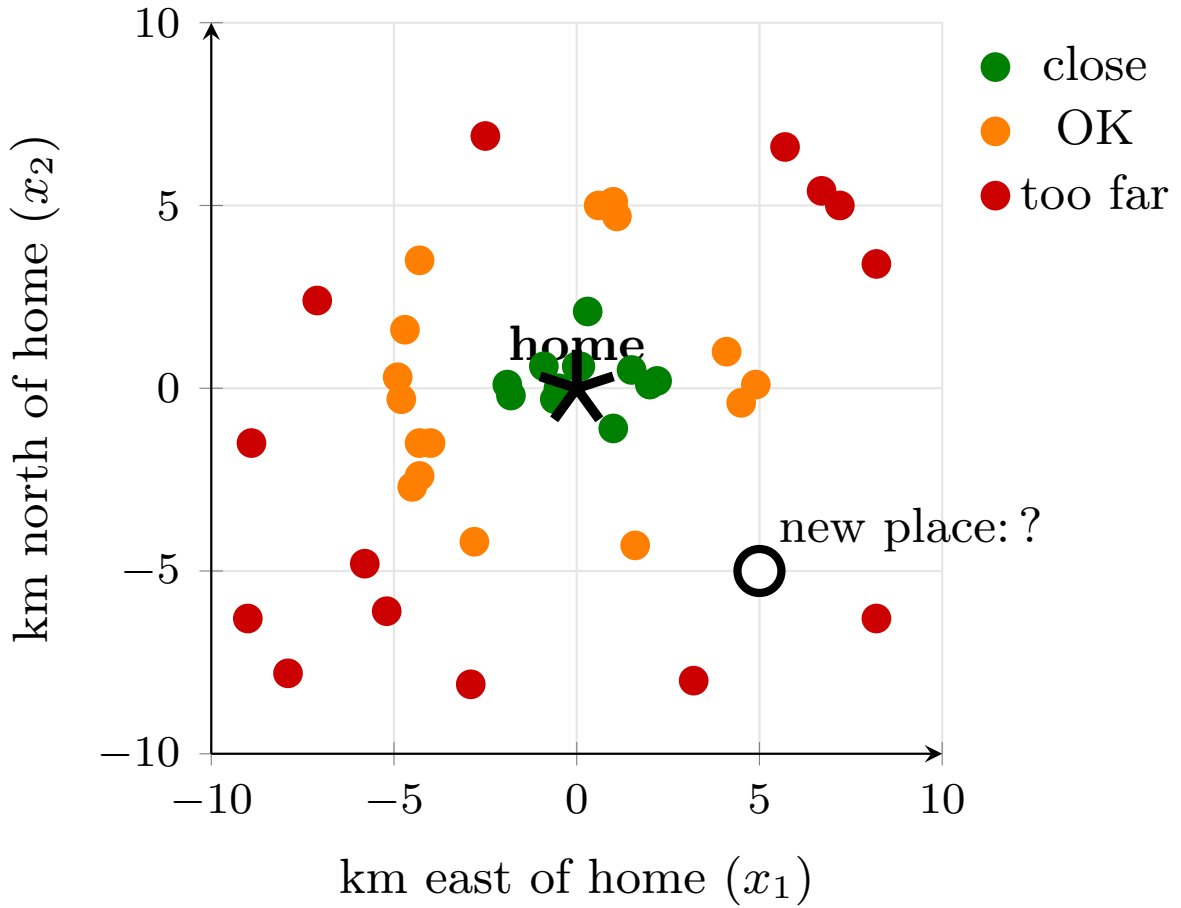
If we throw at $45°$, how far does the ball go ?

Learning from examples : pass or fail ?



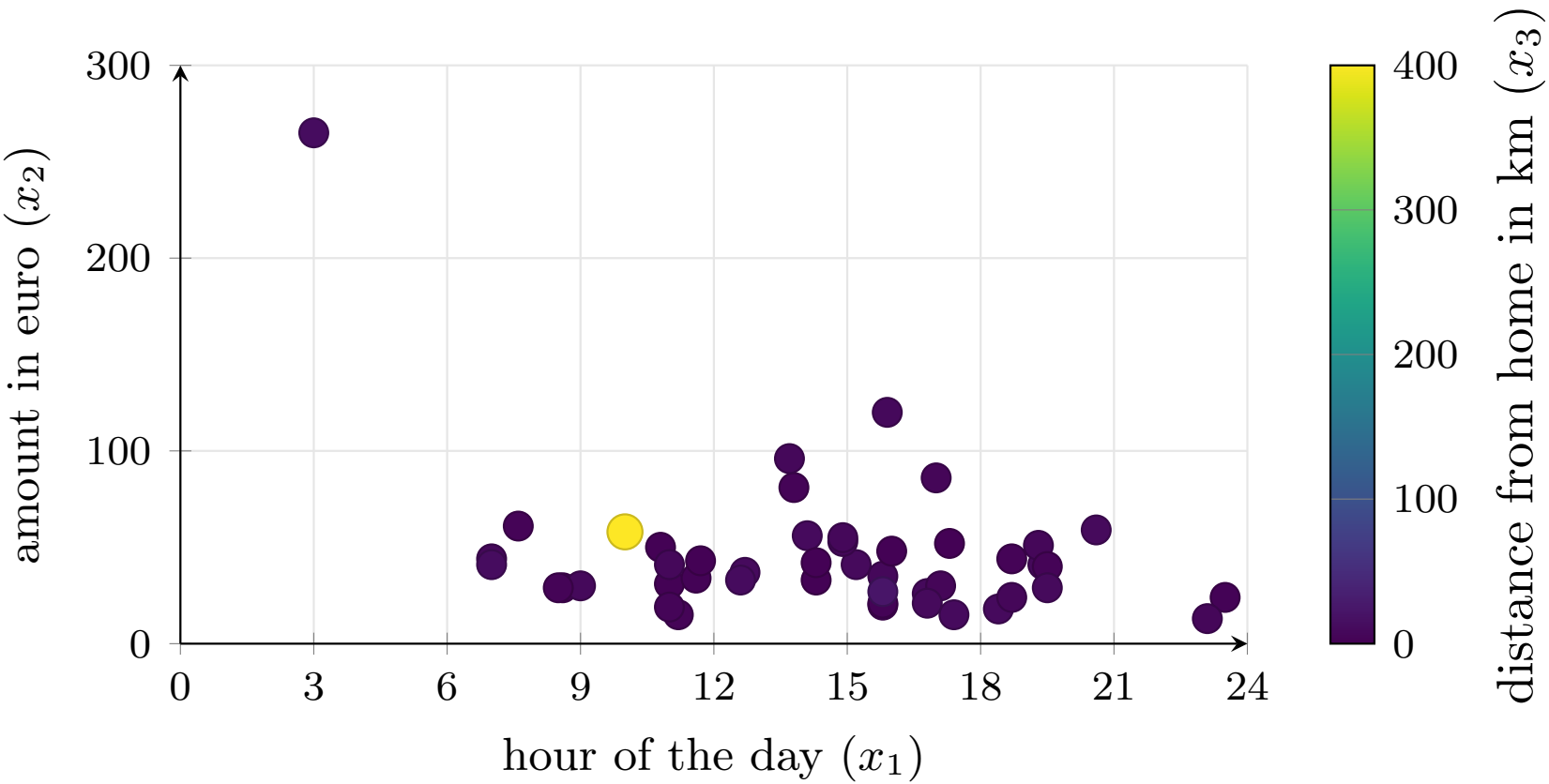
Which color for the new student ?

Learning from examples : is it too far ?



How would we rate the new place ?

Learning from examples : no labels



No labels : which transactions look suspicious ?

Chapter 4: Machine Learning

Motivation

Data

Definitions

Supervised learning

Unsupervised learning

Performance metrics

The data

- **Example** or sample, instance, observation :
  One item the model sees, like one point on the previous plots.
- **Features** or inputs, attributes, x :
  A measurable property of an example, usually a number.
- **Label** or target, output, y :
  The answer we want to predict.
- **Labeled / unlabeled example** :
  With or without its label.
- **Dataset** :
  A collection of examples.

What does data look like? A table

A dataset with n examples and d features is a matrix X with n rows and d columns, plus a vector y of n labels.

Crickets ($d = 1$)

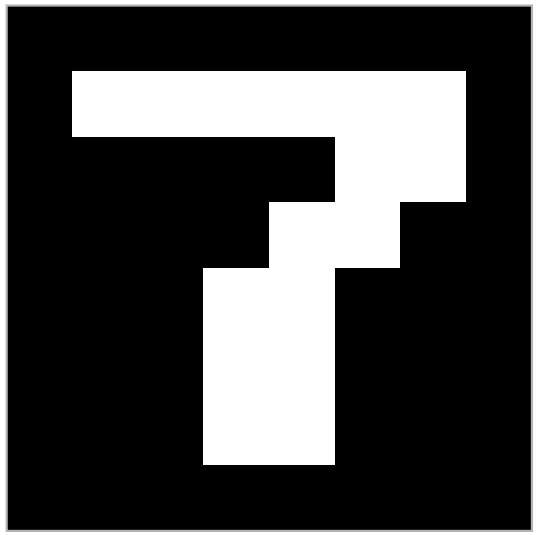
	x (chirps/min)	y (°C)
1	45	10.5
2	60	13.2
3	75	14.8
⋮	⋮	⋮
n	170	28.7

Transactions ($d = 3$, no label)

	hour	amount	km
1	14.3	33	1
2	17.4	15	10
3	3.0	265	12
⋮	⋮	⋮	⋮

- A row is one example (one point) :
  its **feature vector** is $x^{(i)} = (x_1^{(i)}, \ldots, x_d^{(i)})$
  its **label** is $y^{(i)}$.
- A column is one feature, measured on all examples.

What does data look like ? An image



what we see

0	0	0	0	0	0	0	0
0	255	255	255	255	255	255	0
0	0	0	0	0	255	255	0
0	0	0	0	255	255	0	0
0	0	0	255	255	0	0	0
0	0	0	255	255	0	0	0
0	0	0	255	255	0	0	0
0	0	0	0	0	0	0	0

what the computer sees

An 8×8 grayscale image gives a vector with $d = 64$ features :

$x = (0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 255, 255, 255, 255, 255, 0,$
$0, 0, 0, 0, 0, 255, 255, 0, 0, 0, 0, 0, 255, 255, 0, 0,$
$0, 0, 0, 255, 255, 0, 0, 0, 0, 0, 0, 255, 255, 0, 0, 0,$
$0, 0, 0, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

A 24 MP (5712×4284 px) RGB image gives a vector with $d \sim 73.4$ million features.

What does data look like ? Text

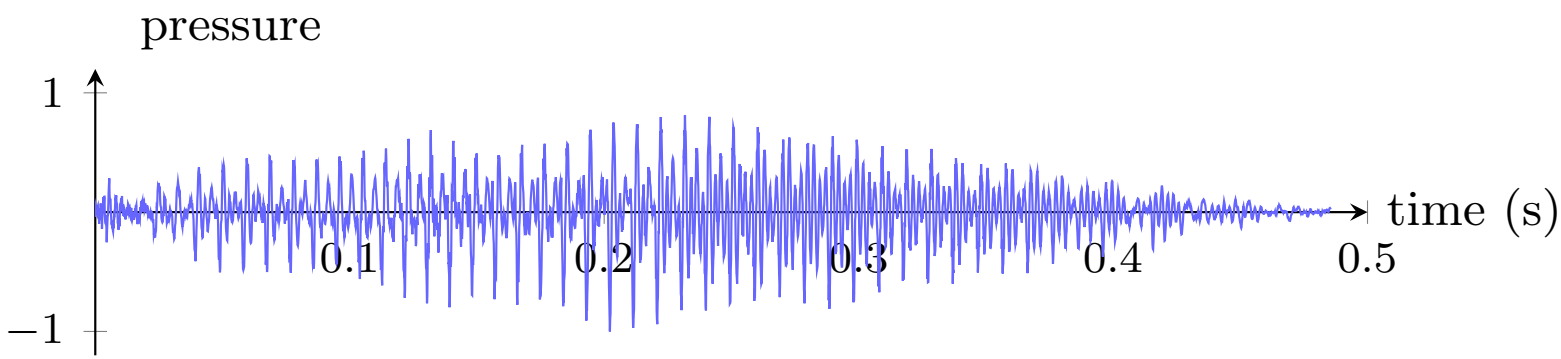
Step 1 : list all the words that appear : the **vocabulary**.

Step 2 : for each text, count how many times each word appears.

		math	is	great	you	are	amazing
1	"math is great"	1	1	1	0	0	0
2	"you are amazing"	0	0	0	1	1	1
3	"math is amazing"	1	1	0	0	0	1

- Each row is a vector : $x^{(1)} = (1, 1, 1, 0, 0, 0)$, with $d = 6$.
- Real vocabularies have tens of thousands of words.
- The order of the words is lost.
- There are other techniques to represent text.

What does data look like ? Sound



- A microphone measures the air pressure at regular instants : each measurement is one number.
- This real recording of someone saying “three” has 3 886 measurements, in order : a feature vector with $d = 3\,886$.

  $x = (-0.04, -0.03, 0.04, 0.03, 0.05, 0.02, \dots)$
- Here 8 000 measurements per second ; a phone voice recorder takes 44 100 per second : one minute is a vector of about 2.6 million features.

What does data look like ? Everything else

Data	**Raw form**	**As numbers**	**Typical size**
Image	grid of pixels	1 grid (gray) or 3 grids, red/green/blue, of values 0–255	28×28 digit : 784 numbers ; 12 MP photo : 36 million
Video	sequence of images	frames $\times$ height $\times$ width $\times$ 3	1 second of HD : 187 million numbers
Text	sequence of words	word counts, or one number (ID) per word	50 000 words in the vocabulary
Sound	pressure wave	amplitudes sampled per second	1 minute : 2.6 million numbers
Network (friends, roads)	nodes and links	adjacency matrix : 1 if linked, else 0	1 000 users : 1 million entries

Chapter 4: Machine Learning

Motivation

Data

Definitions

Supervised learning

Unsupervised learning

Performance metrics

The tasks

- **Supervised learning** : learn from labeled examples to predict the label of new ones.
- **Regression** : the label is a number.
  Example : Temperature from chirps, distance from the throwing angle.
- **Classification** : the label is a **class**, one of a finite set of categories.
  Example : pass/fail (binary), close/OK/too far (multiclass).
- **Unsupervised learning** : only features, no labels ; find structure in the data.
  Example : Anomaly detection finds the suspicious transactions.
- **Prediction** ($\hat{y}$, inference) : applying what was learned to a new input.

The model and training

- **Model** (hypothesis) : a function from features to a prediction, picked from a family.
  Example : A line $\hat{y} = ax + b$ in linear regressions.
- **Parameters** (weights) : the numbers inside the model that are learned from the data.
  Example : The slope a and the intercept b.
- **Loss function** (cost, error, objective) : how wrong a prediction is compared to the label.
- **Training** (fitting) : finding the parameters that minimize the loss on the data.
  Solve for the best line, gradient descent, . . .
- **Hyperparameters** : settings we choose before training ;
  Learning rate, batch size, model architecture, . . .

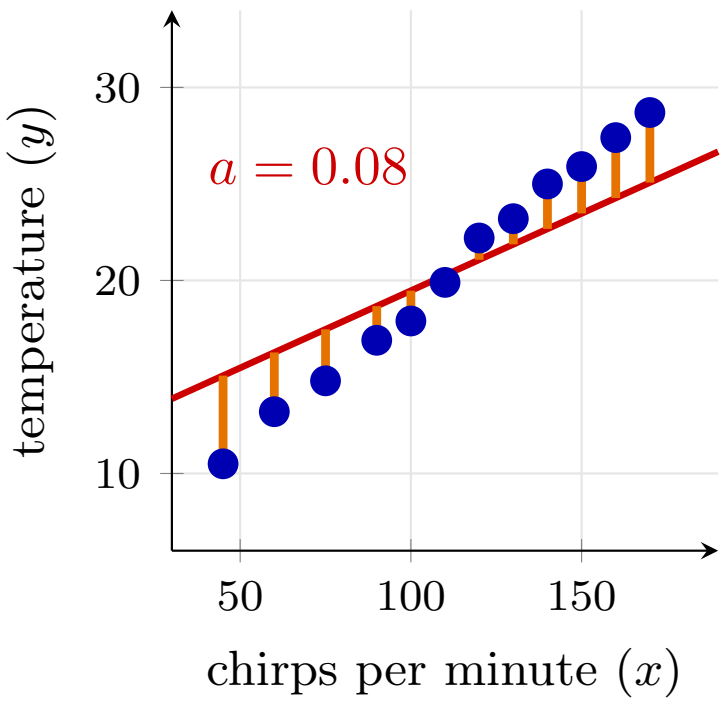
Other definitions

- **Training set** : the examples used to fit the parameters.
- **Validation set** : examples to compare hyperparameters.
- **Test set** : examples to measure how the model does on unseen data.
- **Generalization** : predicting correctly on new, unseen examples.
- **Overfitting** : the model matches the training data too closely, noise included, and fails on new data.
  Example : A degree-10 polynomial through the 12 projectile points.
- **Underfitting** : the model is too simple to capture the pattern.
  Example : A straight line through the projectile data.

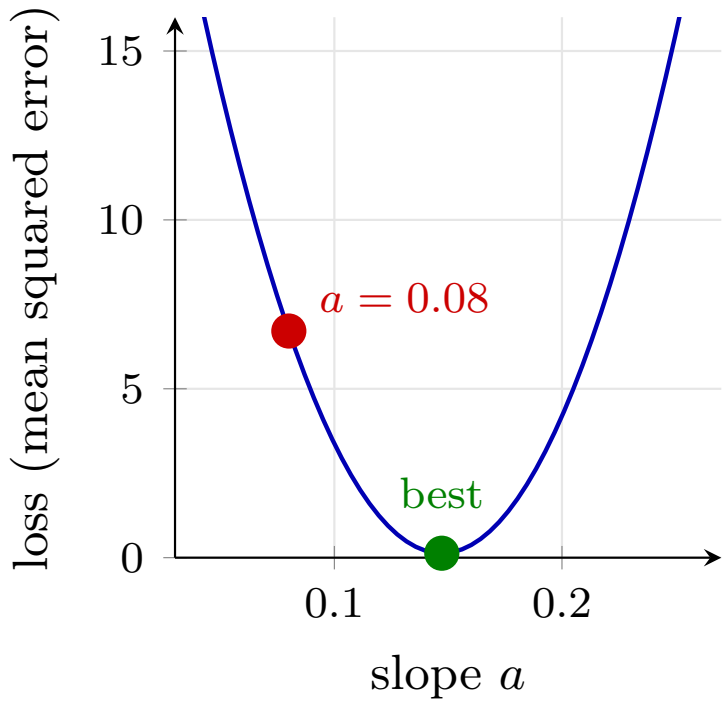
Recap of all the vocabulary and the synonyms

example	sample, instance, observation, data point, record, row
feature	attribute, variable, input, x, X
label	target, output, response, ground truth, y
model	predictor
parameters	weights, coefficients, w
loss function	cost function, error function, objective function
training	fitting, learning
prediction	estimate, output, inference, $\hat{y}$

Fitting the data

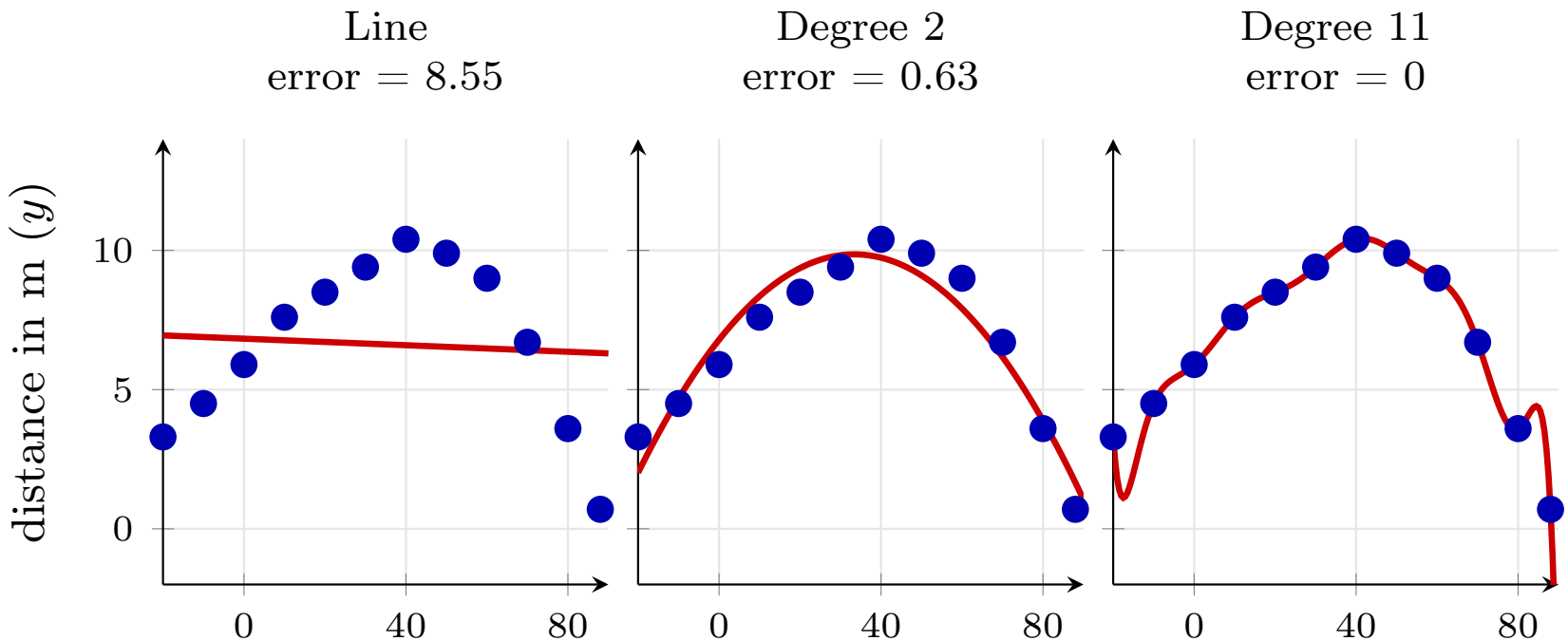


The model



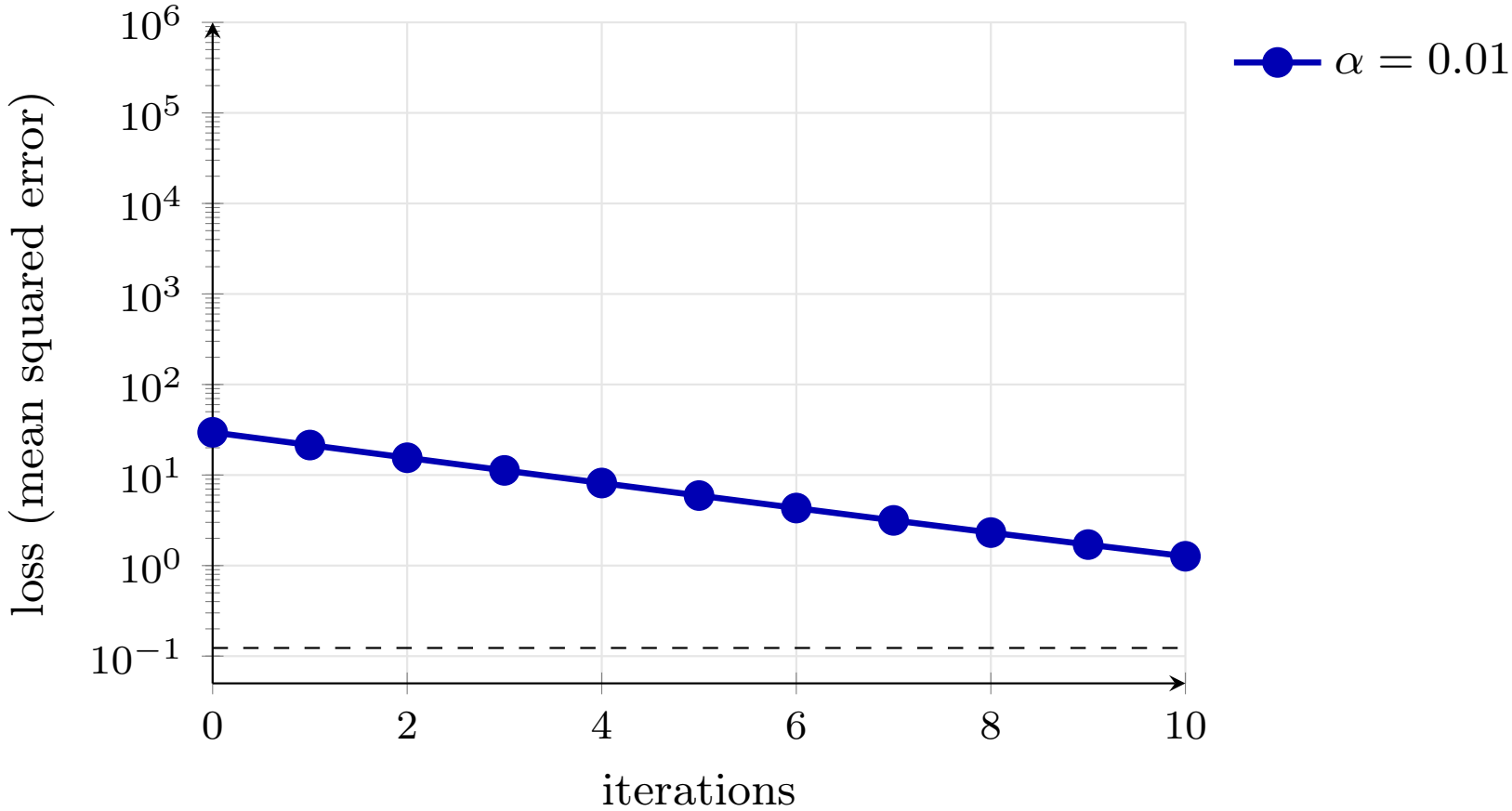
The loss function

Overfitting and underfitting

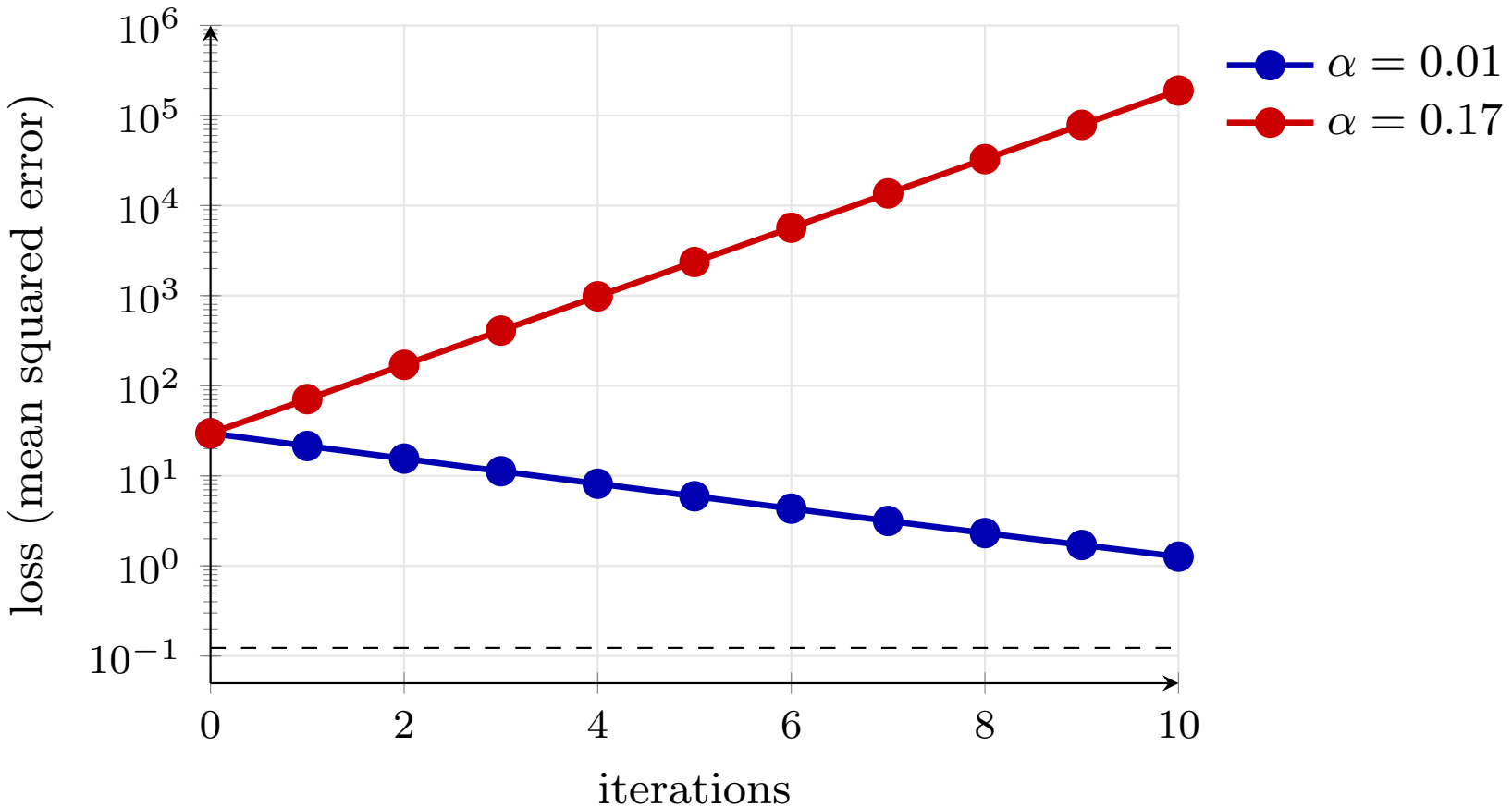


x = throwing angle.

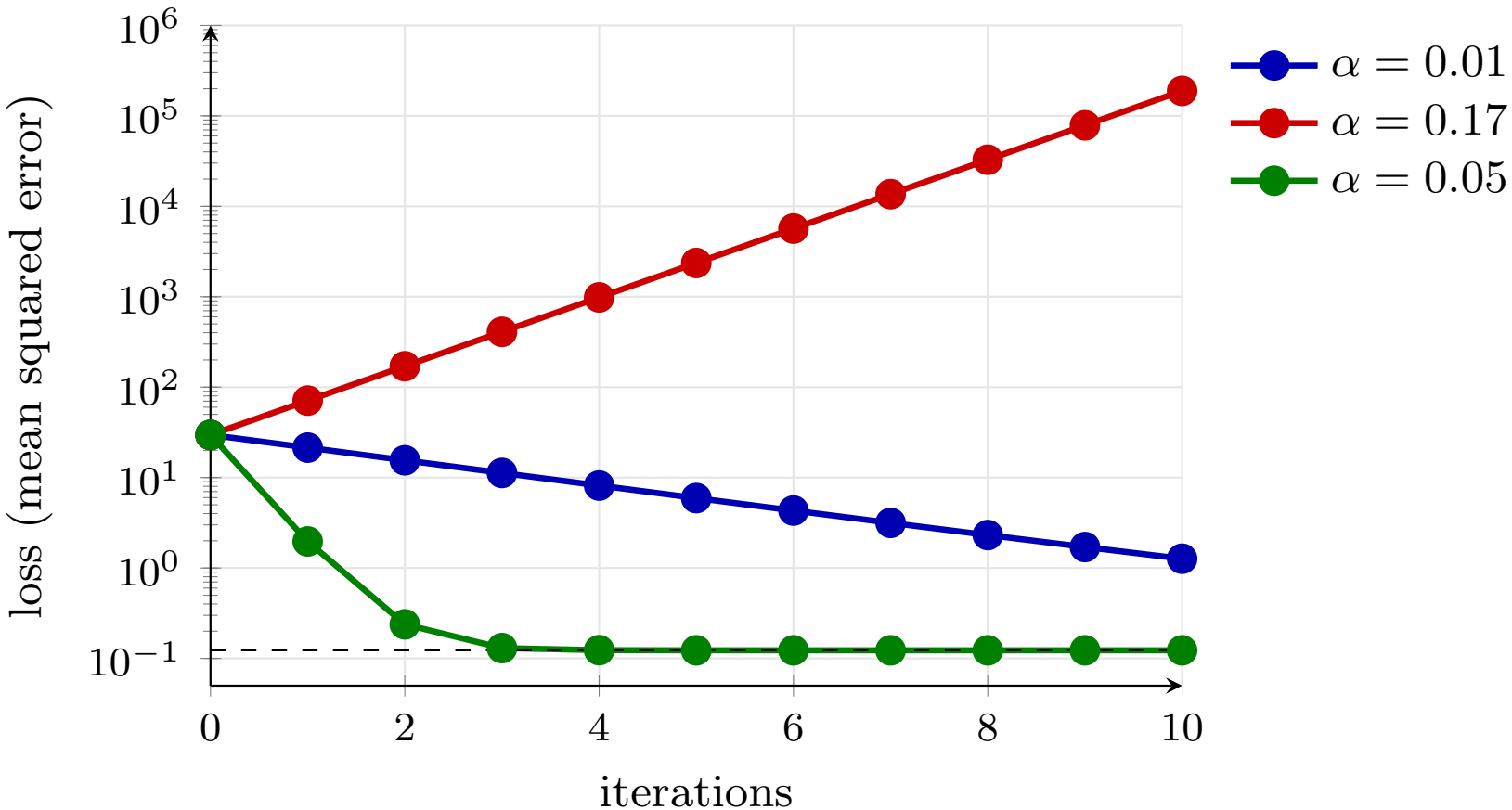
Hyperparameters : the learning rate



Hyperparameters : the learning rate



Hyperparameters : the learning rate



Chapter 4: Machine Learning

Motivation

Data

Definitions

Supervised learning

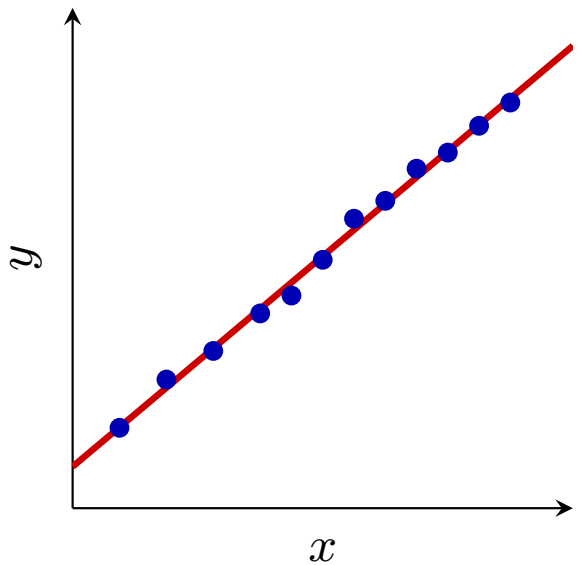
Unsupervised learning

Performance metrics

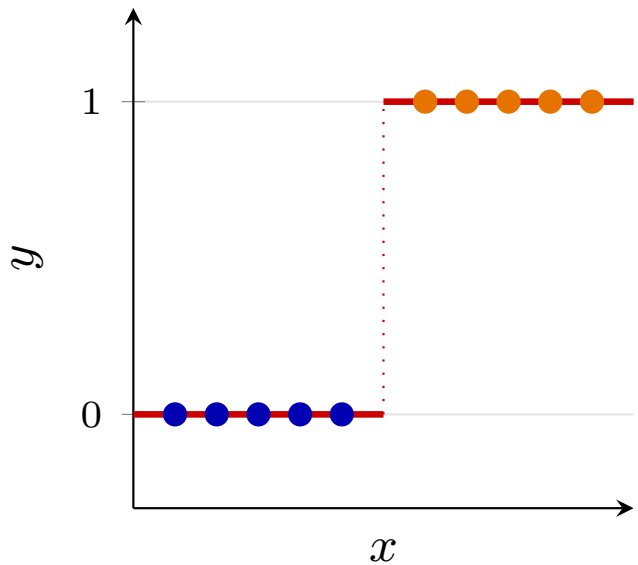
Supervised learning : setup

Data : $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$.
Goal : a function f with $f(x) \approx y$.

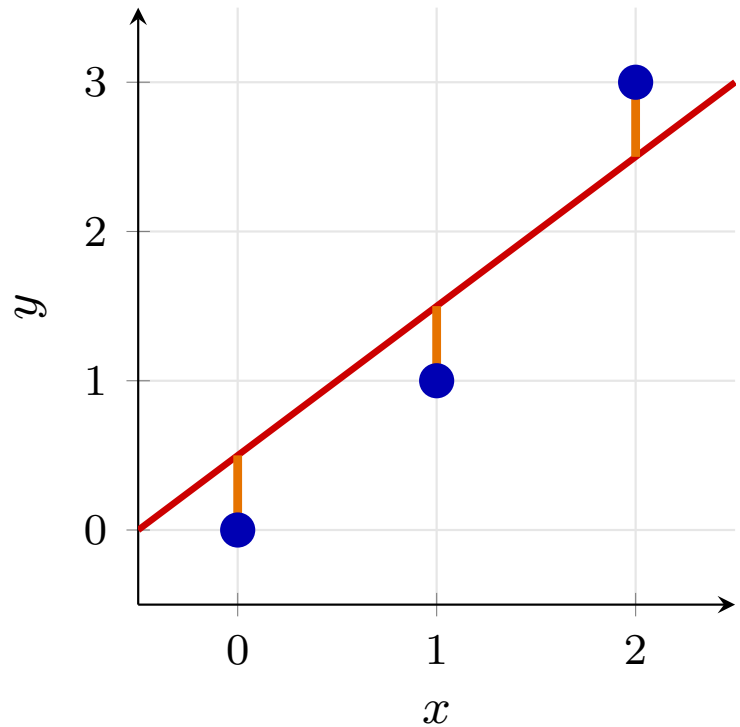
Regression
$y \in \mathbb{R}$



Classification
$y \in \{0, 1\}$



Linear regression : fitting a line



$$\text{Model} : \hat{y} = ax + b \qquad \text{Loss} : L(a, b) = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

Solving linear regression : matrix inversion

We want $\underbrace{\begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{w} = \underbrace{\begin{pmatrix} 0 \\ 1 \\ 3 \end{pmatrix}}_{y}$

Multiply both sides by $X^\top$:

$$X^\top X\, w = X^\top y \qquad \Rightarrow\ w = (X^\top X)^{-1} X^\top y$$

$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{pmatrix} w = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 3 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 3 \\ 3 & 3 \end{pmatrix} w = \begin{pmatrix} 7 \\ 4 \end{pmatrix} \qquad \Rightarrow\ w = \begin{pmatrix} 3/2 \\ -1/6 \end{pmatrix}$$

Solving linear regression : gradient = 0

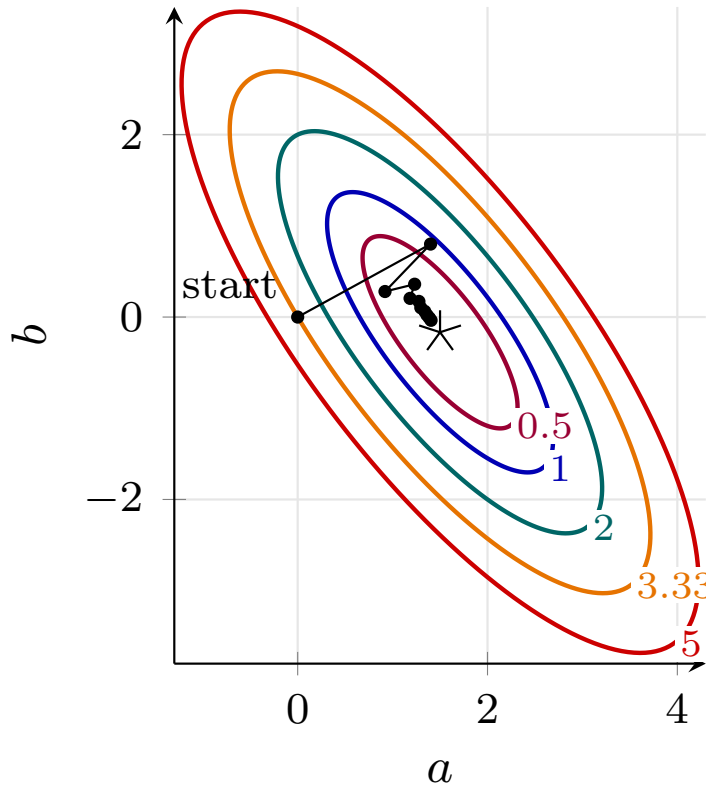
$$L(a,b) = \tfrac{1}{3}\Big[\, b^2 + (a+b-1)^2 + (2a+b-3)^2 \Big]$$

$$\frac{\partial L}{\partial a} = \tfrac{2}{3}\big(5a+3b-7\big) = 0 \qquad \frac{\partial L}{\partial b} = \tfrac{2}{3}\big(3a+3b-4\big) = 0$$

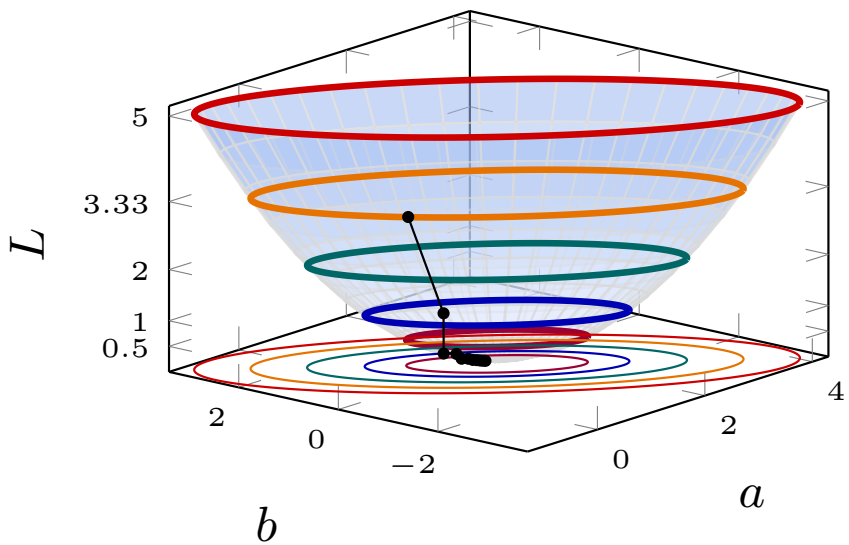
$$\Rightarrow \begin{pmatrix} 5 & 3 \\ 3 & 3 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \end{pmatrix} \Rightarrow\ a = \tfrac{3}{2},\ b = -\tfrac{1}{6}$$

Solving linear regression : gradient descent

2D view of the loss function



3D view of the loss function



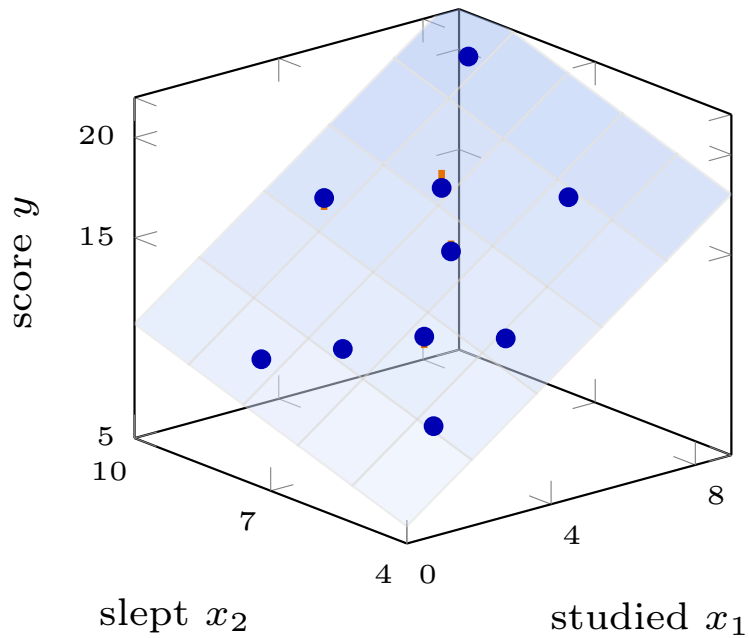
$$\begin{pmatrix} a \\ b \end{pmatrix} \leftarrow \begin{pmatrix} a \\ b \end{pmatrix} - \alpha \nabla L(a,b)\,, \qquad \nabla L(a,b) = \begin{pmatrix} \partial L/\partial a \\ \partial L/\partial b \end{pmatrix} = \frac{2}{3} \begin{pmatrix} 5a+3b-7 \\ 3a+3b-4 \end{pmatrix}$$

Linear regression : several features

Each example has 2 features :
$\left(x^{(i)}, y^{(i)}\right) = \left(x_1^{(i)}, x_2^{(i)}, y^{(i)}\right)$

$$\hat{y}^{(i)} = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)}$$

$$X = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} \end{pmatrix}$$



Same solutions :

- Solve : $w = (X^\top X)^{-1} X^\top y$
- Gradient descent : $w \leftarrow w - \alpha \nabla L(w)$

Linear regression : pros and cons

Use it when the output depends linearly on the features.

Pros

- Simple, fast, a single solution
- Easy to interpret : one weight per feature
- A closed-form solution exists (matrix inversion)

Cons

- Only straight lines (planes)
- Squared error : outliers pull the line

Guarantees for the solution :

- $X^\top X$ is invertible if the columns of X are linearly independent
- In gradient descent the learning rate α should be small enough

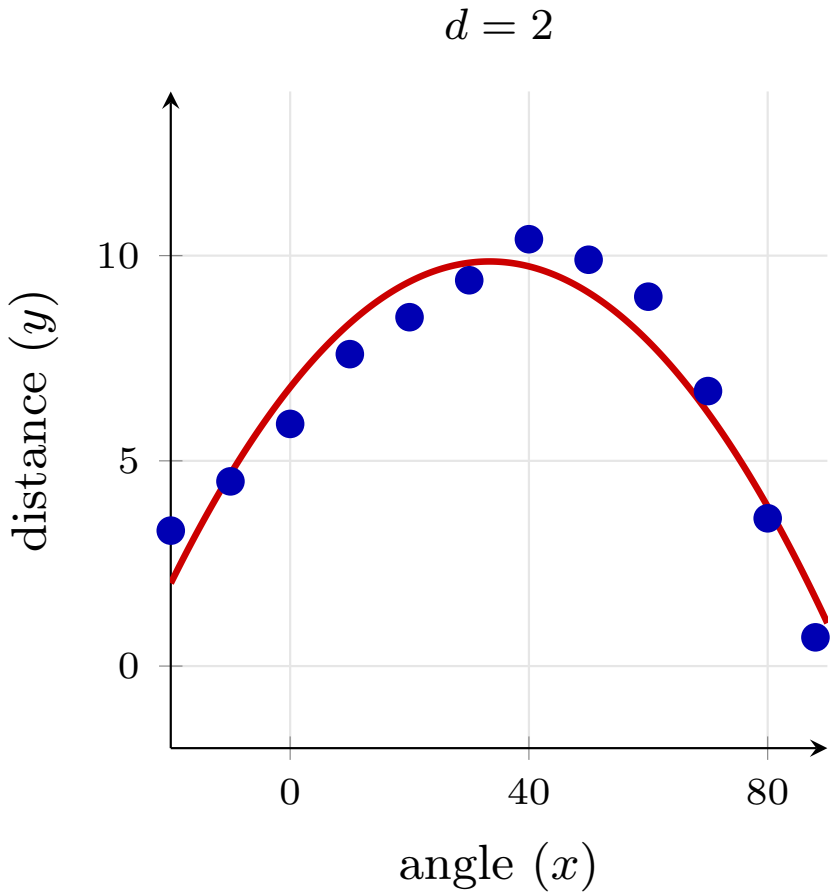
Hyperparameters : none for the closed form ; the learning rate α for gradient descent, ...

Higher degrees : polynomial regression

Polynomial of degree d :

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + \cdots + w_d x^d$$

$$X = \begin{pmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \dots & (x^{(1)})^d \\ \vdots & & & & \vdots \\ 1 & x^{(n)} & (x^{(n)})^2 & \dots & (x^{(n)})^d \end{pmatrix}$$



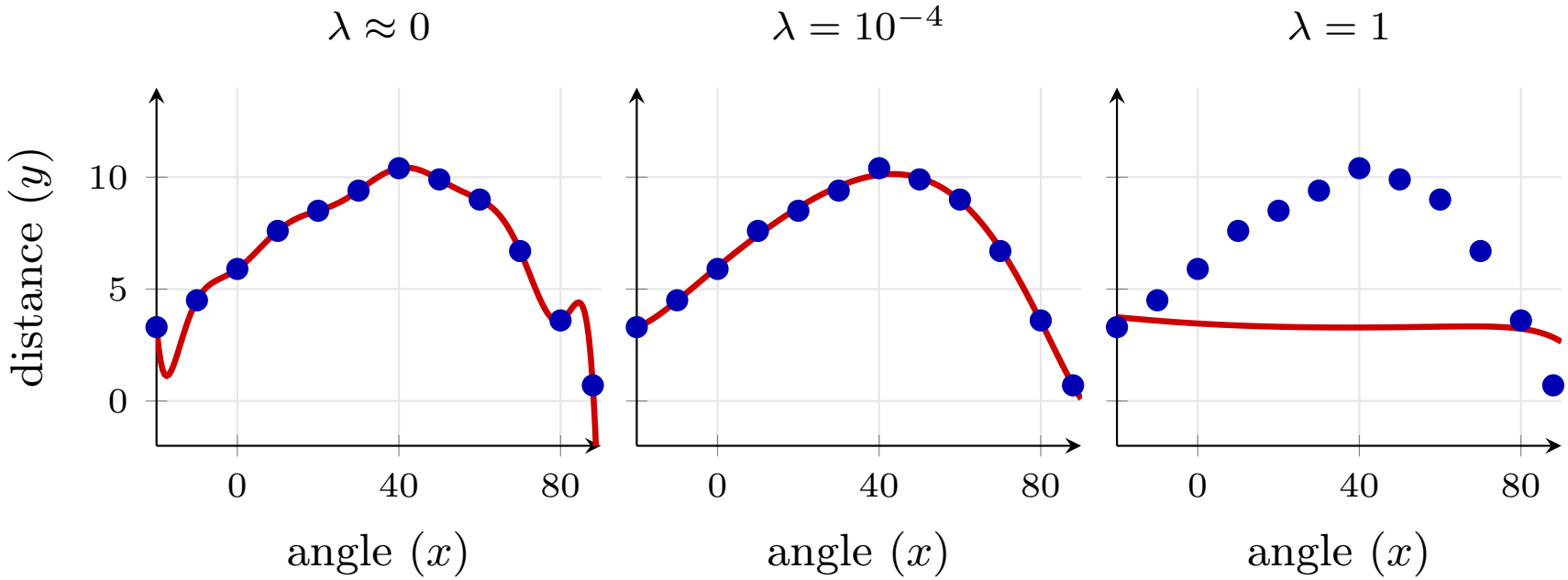
Linear in the parameters : same $w = (X^\top X)^{-1} X^\top y$.
The weights can also be found by gradient descent.

Regularization : preventing overfitting

- Stop the training early
- Simplify the model (like reduce the degree of the polynomial if the model is a polynomial)
- Add regularization terms to the loss function
- Remove outliers from the dataset
- ...

Regularization : a penalty on the weights

Degree-11 polynomial P fitted to the 12 points



$$L(w) = \underbrace{\sum_{i=1}^{12} \left(P(x^{(i)}) - y^{(i)}\right)^2}_{\text{error on the data}} + \underbrace{\lambda \sum_{k=0}^{11} w_k^2}_{\text{penalty on large weights}}$$

where $P(x) = w_0 + w_1 x + \cdots + w_{11} x^{11}$

Perceptron : a linear classifier

Model for example i, with $x^{(i)}, w \in \mathbb{R}^d$

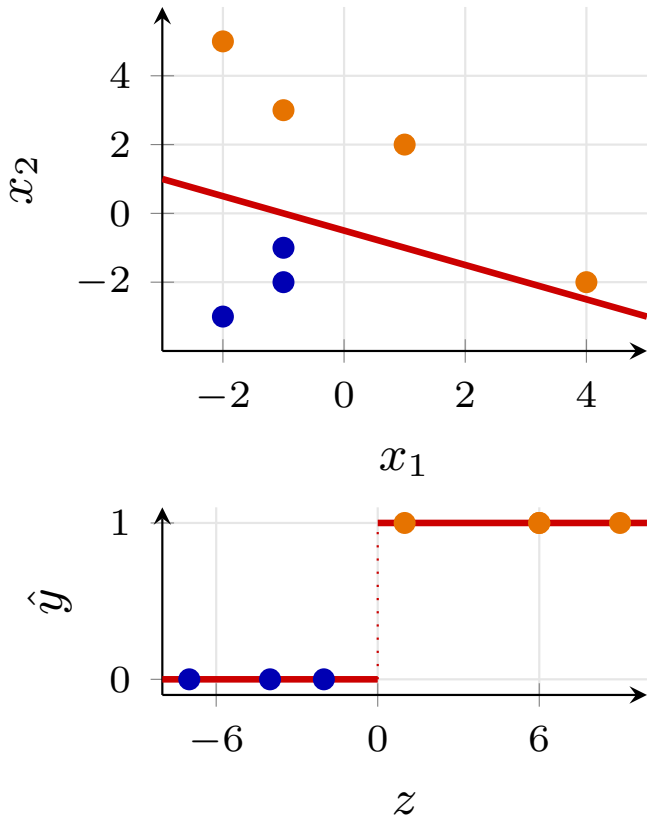
$$z^{(i)} = w^\top x^{(i)} + b \qquad \hat{y}^{(i)} = \begin{cases} 1 & \text{if } z^{(i)} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Algorithm

Start with $w = 0$ and $b = 0$ and repeat over the examples until there is no mistake. For each misclassified example i :

$$w \leftarrow w + \left(y^{(i)} - \hat{y}^{(i)}\right) x^{(i)}$$

$$b \leftarrow b + \left(y^{(i)} - \hat{y}^{(i)}\right)$$



iteration	1	2	3	4	5	6	7	8–14
point x	$(-1,-1)$	$(-1,-2)$	$(1,2)$	$(-1,3)$	$(4,-2)$	$(-2,-3)$	$(-2,5)$	same
label y	0	0	1	1	1	0	1	
prediction $\hat{y}$	0	0	0	1	1	0	1	no mistake
w, b after	$(0,0), 0$	$(0,0), 0$	$(1,2), 1$	$(1,2), 1$	$(1,2), 1$	$(1,2), 1$	$(1,2), 1$	$(1,2), 1$

Perceptron : guarantees, pros and cons

Use it when the classification problem is simple.

Guaranteed to work if the classes are linearly separable.

Pros

- Very simple
- Learns one example at a time

Cons

- Linear separation only : it cannot handle classes that no line or plane separates
- If the classes are not separable, the algorithm never stops
- The line depends on the order of the examples

Hyperparameters : the step size is 1.

Logistic regression : a probabilistic classifier

Model for example i, with $x^{(i)}, w \in \mathbb{R}^d$

$$z^{(i)} = w^\top x^{(i)} + b$$

$$\hat{y}^{(i)} = \sigma\left(z^{(i)}\right) = \frac{1}{1 + e^{-z^{(i)}}}$$

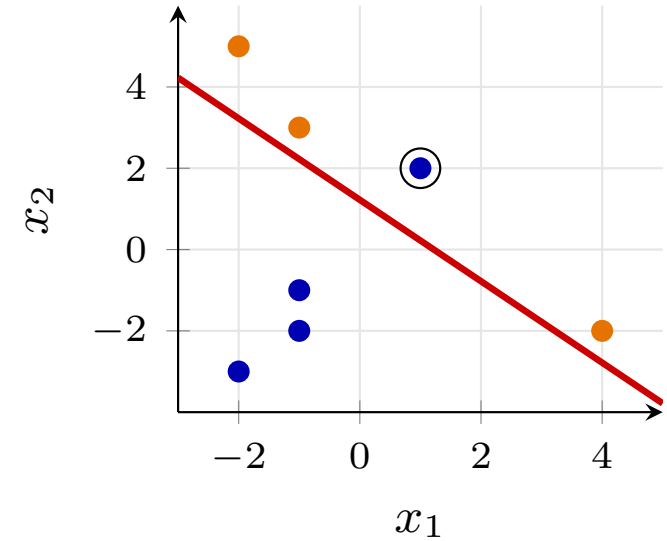
Algorithm

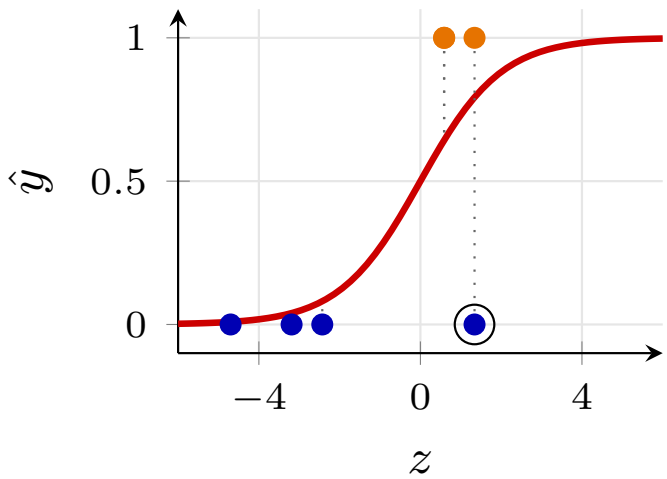
Start with $w = 0$ and $b = 0$ and repeat over the examples. For each example i :

$$w \leftarrow w + \alpha \left(y^{(i)} - \hat{y}^{(i)}\right) x^{(i)}$$

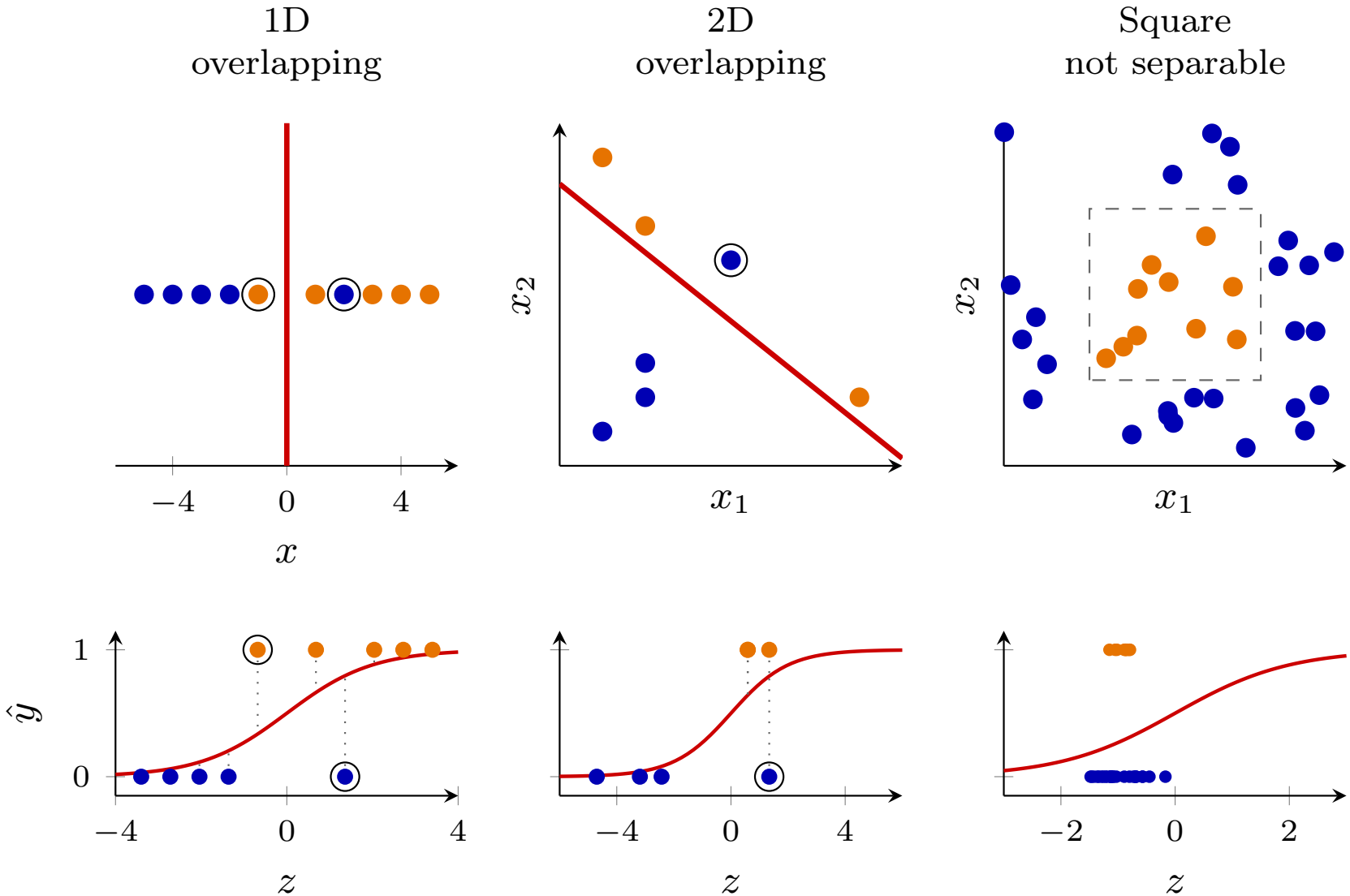
$$b \leftarrow b + \alpha \left(y^{(i)} - \hat{y}^{(i)}\right)$$

with α the learning rate.





Logistic regression : three datasets



Logistic regression : pros and cons

Use it for classification, when lines or planes roughly separate the classes and you want probabilities.

Pros

- Outputs a probability
- One weight per feature : interpretable

Cons

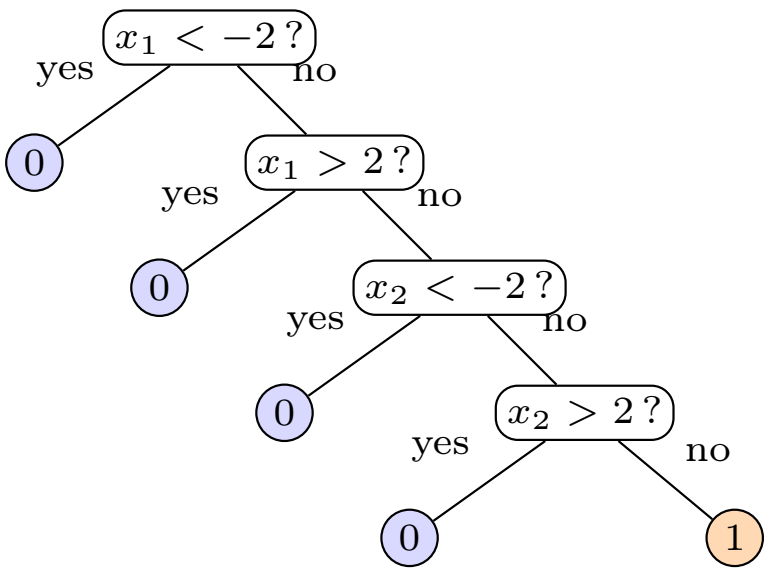
- Linear separation only (the square : fails)
- No closed form : needs gradient descent

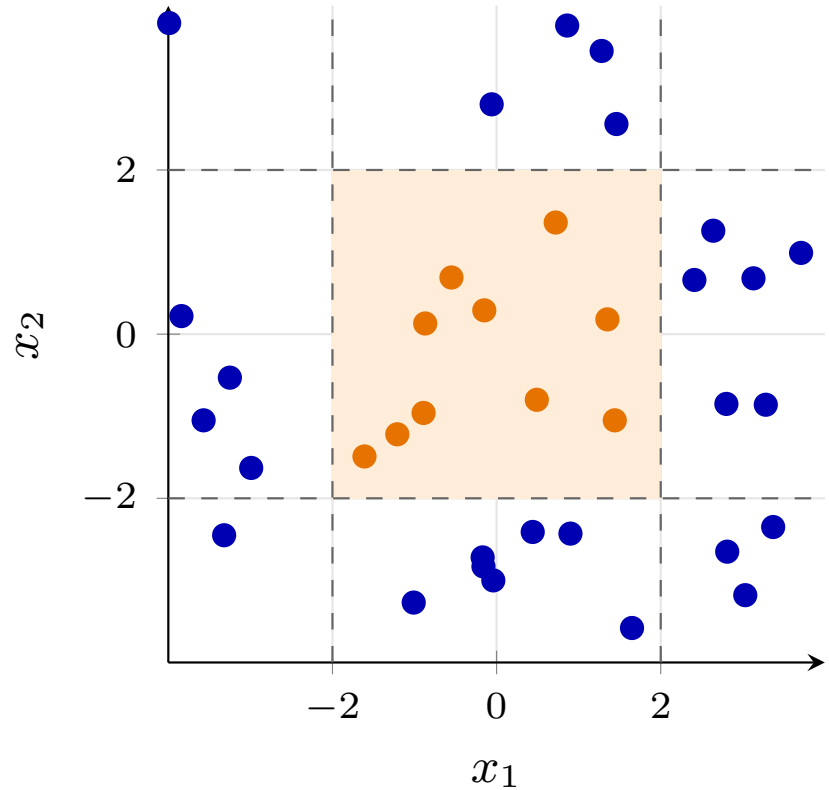
Guaranteed to work if

- the classes are linearly (almost) separable by a line or plane
- α is small enough : gradient descent then reaches the best weights

Hyperparameters : the learning rate α, the number of steps, the regularization strength (optional).

Decision trees : the idea





Algorithm : pick the cut on one feature that best separates the classes, then repeat on each side until the groups are pure or the tree is deep enough.

Decision trees : guarantees, pros and cons

Use it when you want readable rules, or the features have different units and mixed types.

Guaranteed to work if

- the tree is grown deep enough : it fits any training set in which no two identical inputs have different labels ;
- no guarantee on new data : a deep tree overfits.

Pros

- Readable rules
- No feature scaling
- Non-linear boundaries

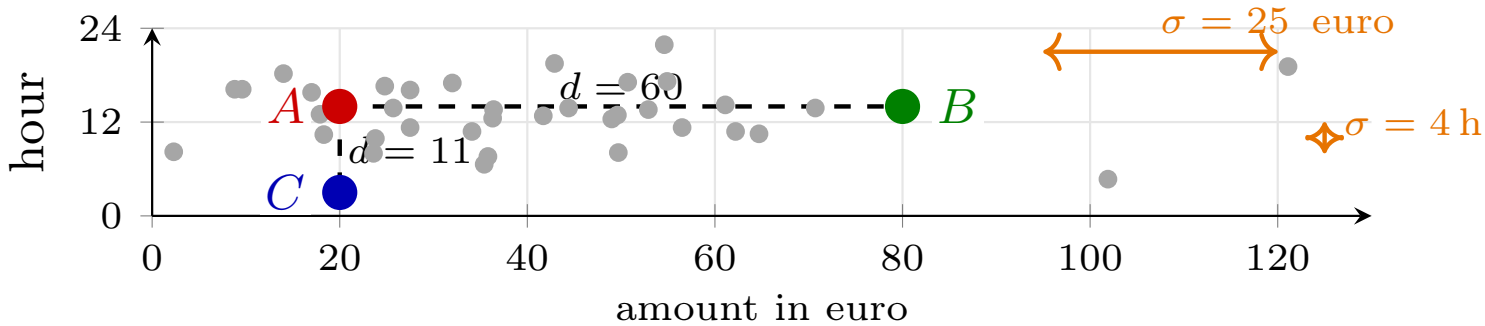
Cons

- Unstable : a small change in the data changes the tree
- Axis-aligned cuts only
- Overfits when deep

Hyperparameters : maximum depth, minimum examples per leaf.

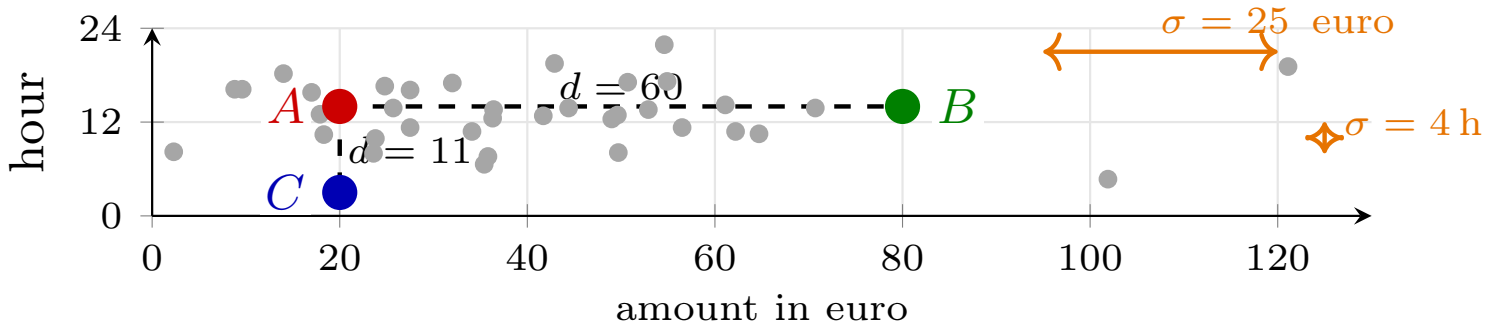
Feature scaling : why

Transactions : $A = (20, 14)$, $B = (80, 14)$, $C = (20, 3)$

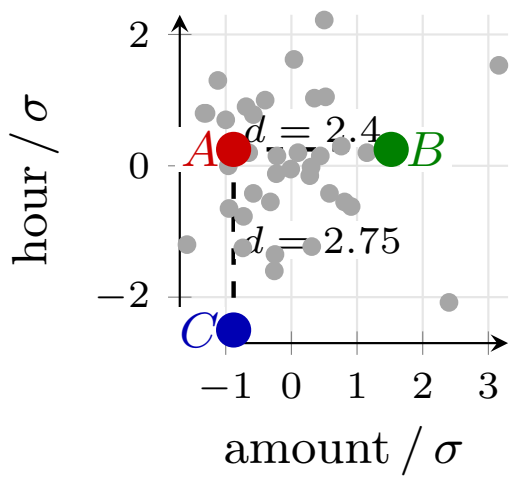


Feature scaling : why

Transactions : $A = (20, 14)$, $B = (80, 14)$, $C = (20, 3)$



If we divide each feature by its standard deviation :



Feature scaling : how

Rescale each feature to a comparable scale :

- **Standardization** : $x' = \dfrac{x - \mu}{\sigma}$ (mean 0, standard deviation 1)
- **Min–max** : $x' = \dfrac{x - x_{\min}}{x_{\max} - x_{\min}} \in [0, 1]$, sensitive to outliers

Needed for kNN, k-means and PCA. Speeds up gradient descent. Not for trees.

Compute μ, σ on the *training* set, then apply the *same* μ, σ to validation, test and new data.

Feature scaling and gradient descent

Points $(0,0),(1,1),(2,3)$: $L(a,b) = \frac{1}{3}\sum_{i=1}^{3}\left(a\,x^{(i)} + b - y^{(i)}\right)^2$, and $\alpha = 0.3$.

Raw x

$$\frac{\partial L}{\partial a} = \tfrac{2}{3}(5a + 3b - 7)$$
$$\frac{\partial L}{\partial b} = \tfrac{2}{3}(3a + 3b - 4)$$

step	a	b	L
0	0	0	3.333
1	1.400	0.800	0.813
2	0.920	0.280	0.298
3	1.232	0.360	0.170
4	1.184	0.205	0.125
5	1.277	0.172	0.102

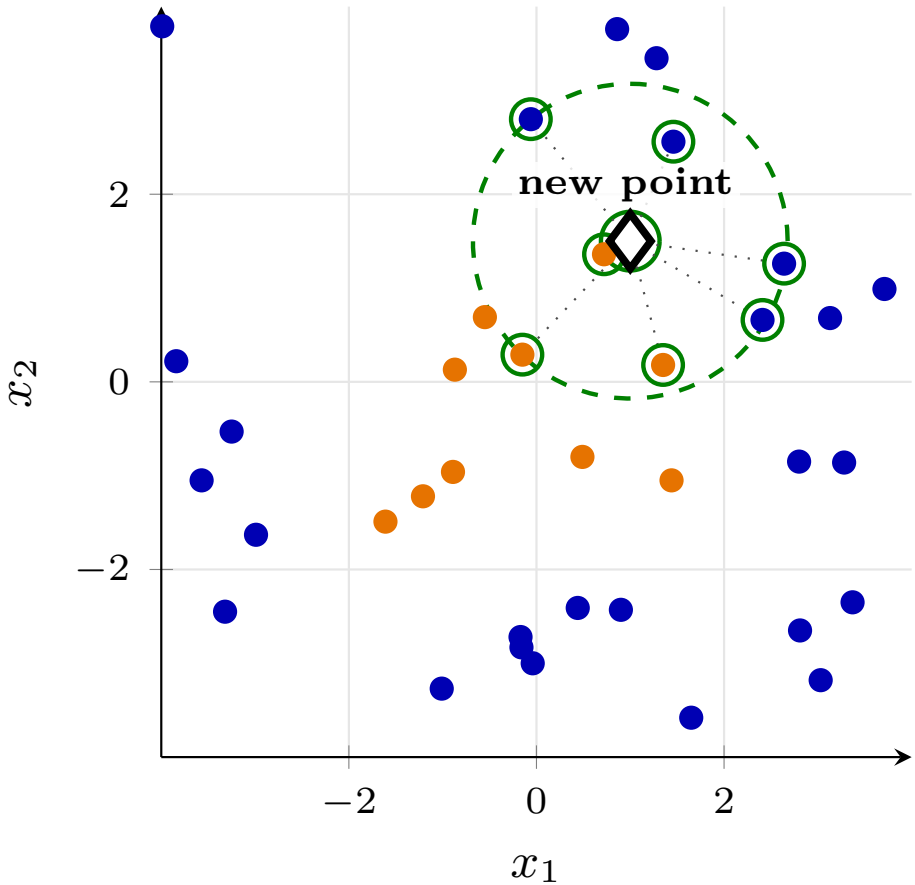
Standardized $x' = \dfrac{x - 1}{\sqrt{2/3}}$

$$\frac{\partial L}{\partial a'} = \tfrac{2}{3}(3a' - 3.67)$$
$$\frac{\partial L}{\partial b'} = \tfrac{2}{3}(3b' - 4)$$

step	a'	b'	L
0	0	0	3.333
1	0.735	0.800	0.580
2	1.029	1.120	0.140
3	1.146	1.248	0.069
4	1.193	1.299	0.058
5	1.212	1.320	0.056

Loss within 0.001 of the optimum (0.056) : 16 steps raw, 5 steps standardized.

k-nearest neighbors : the idea



Model : the training data itself, no parameters.

Algorithm : to classify a new point, find its k closest training points and take the majority class.

k-nearest neighbors : guarantees, pros and cons

Use it when the data is small, has few features, and the boundary is irregular.

Pros

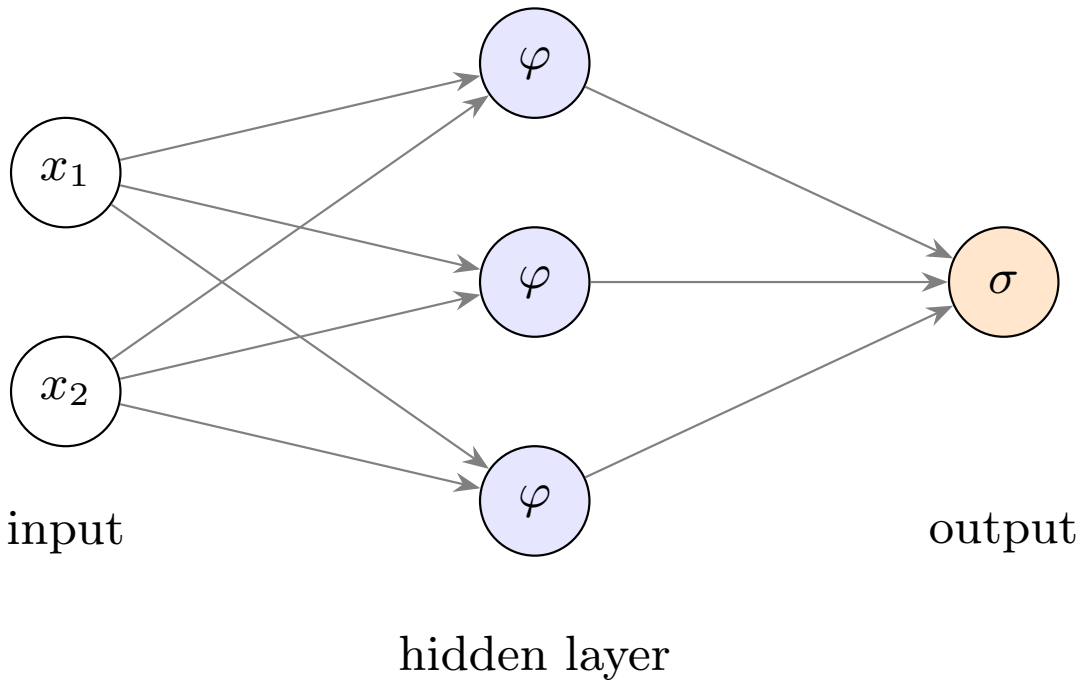
- No training
- Non-linear boundaries
- Very easy to explain

Cons

- Slow predictions (compare with all the data)
- Needs feature scaling
- Struggles with many features

Hyperparameters : k (small : noisy ; large : too smooth), the distance (L_1, L_2, L_p, cosine, ...), neighbor weighting, ...

Non-linear models : neural networks



Each node : $h = \varphi(w^\top x + b)$, an **activation function**.

Chapter 4: Machine Learning

Motivation

Data

Definitions

Supervised learning

Unsupervised learning

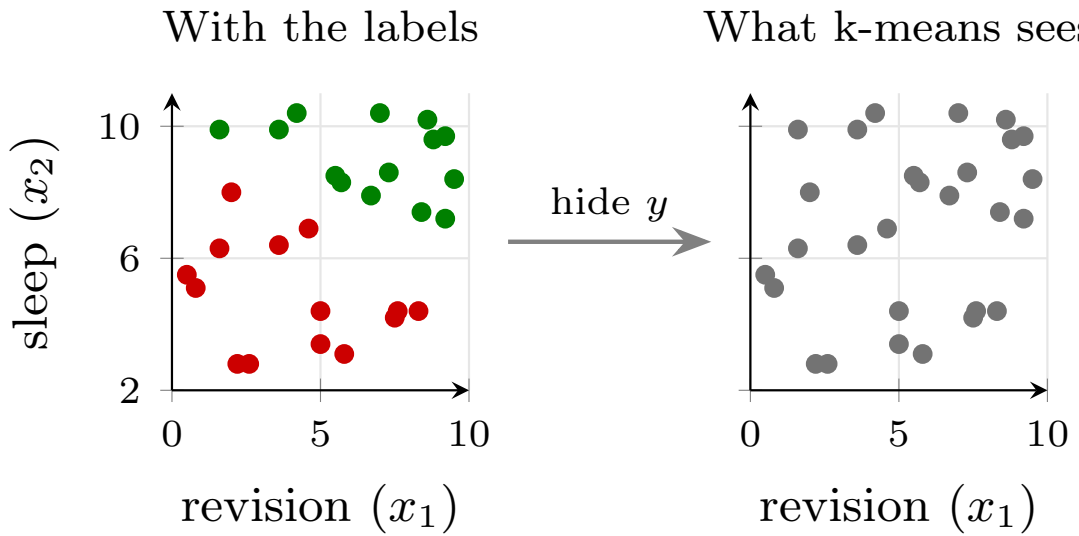
Performance metrics

Unsupervised learning : setup

Data : $x^{(1)}, \ldots, x^{(n)}$, no labels.
Goal : find structure in the data.

Task	We look for	Method
Clustering	groups of similar examples	k-means, hierarch.
Dimensionality reduction	fewer features, same information	PCA
Anomaly detection	rare, very different examples	isolation forest

k-means : the idea



Suppose we do not have the labels : can we still find the groups ?

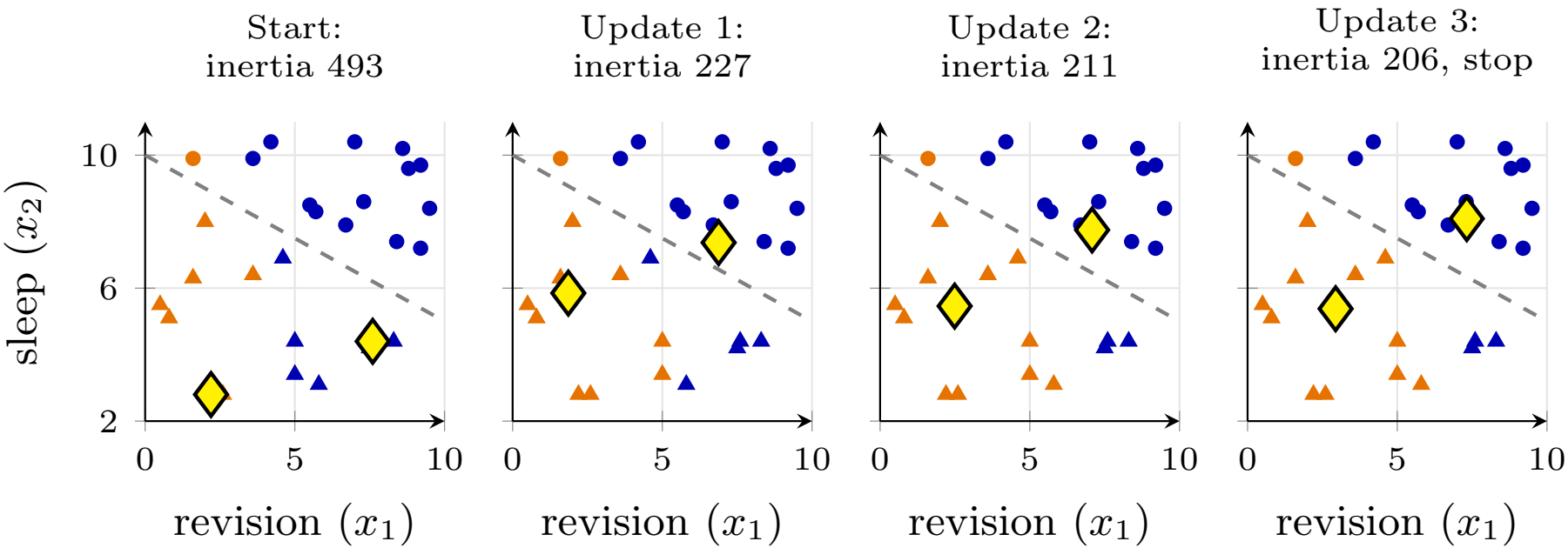
Model : k centroids ; each point belongs to the nearest one.

Loss (inertia) : sum of squared distances to the nearest centroid.

Algorithm :

1. Start with k centroids (e.g. k of the points).
2. Assign each point to its nearest centroid.
3. Move each centroid to the mean of its points.
4. Repeat 2–3 until nothing changes.

k-means on pass/fail : $k = 2$

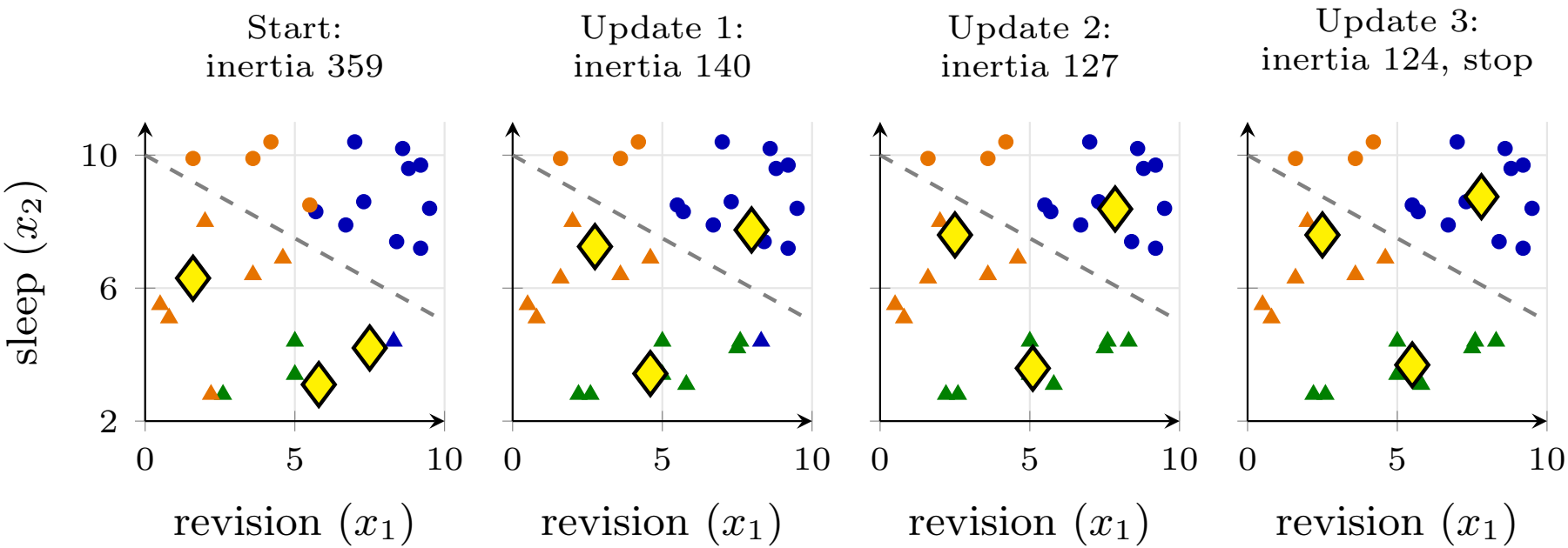


Color : cluster. Dot : passed, triangle : failed (k-means does not see this). Dashed : the true pass/fail boundary.

Start : two failed students as centroids. **After 3 updates** nothing changes : the inertia only went down.

The two clusters roughly follow pass/fail, but a few students near the boundary land on the wrong side.

k-means on pass/fail : $k = 3$



Start : three failed students as centroids. **After 3 updates** nothing changes.

Lower inertia than with $k = 2$ (124 vs 206) : more centroids, so points are closer to one. Inertia alone cannot choose k.

k-means groups by **distance** : one cluster sits on the boundary and mixes passed and failed students. Clusters are not labels.

k-means : guarantees, pros and cons

Use it when the groups are compact and round.

Guarantees

- always stops : each step lowers the inertia or keeps it ;
- the end result depends on the initial centroids : with $k = 3$ on our students, random starts end anywhere between inertia 124 and 177.

Pros

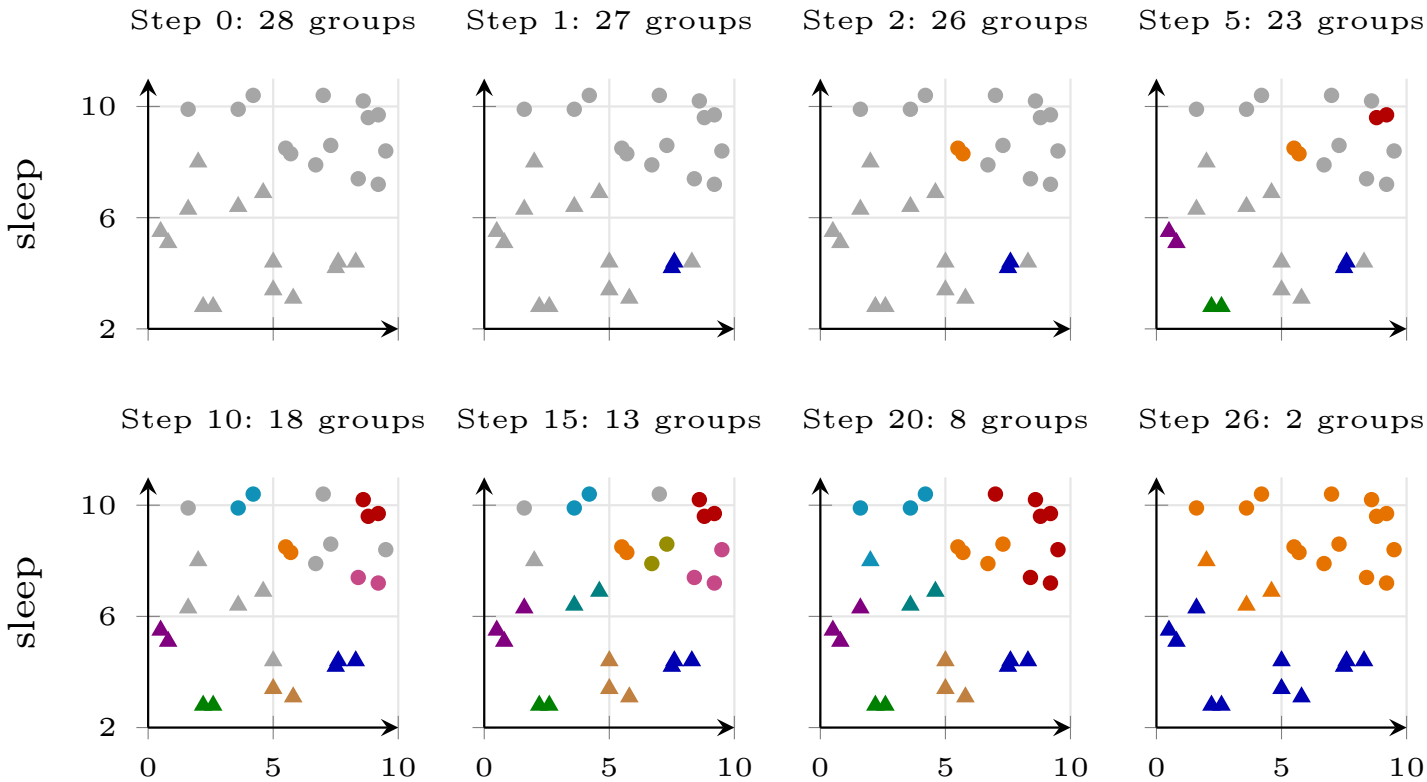
- Very simple and fast
- Works on large datasets

Cons

- k must be chosen
- Needs feature scaling

Hyperparameters : k, the initialization, the number of restarts.

Hierarchical clustering : the idea

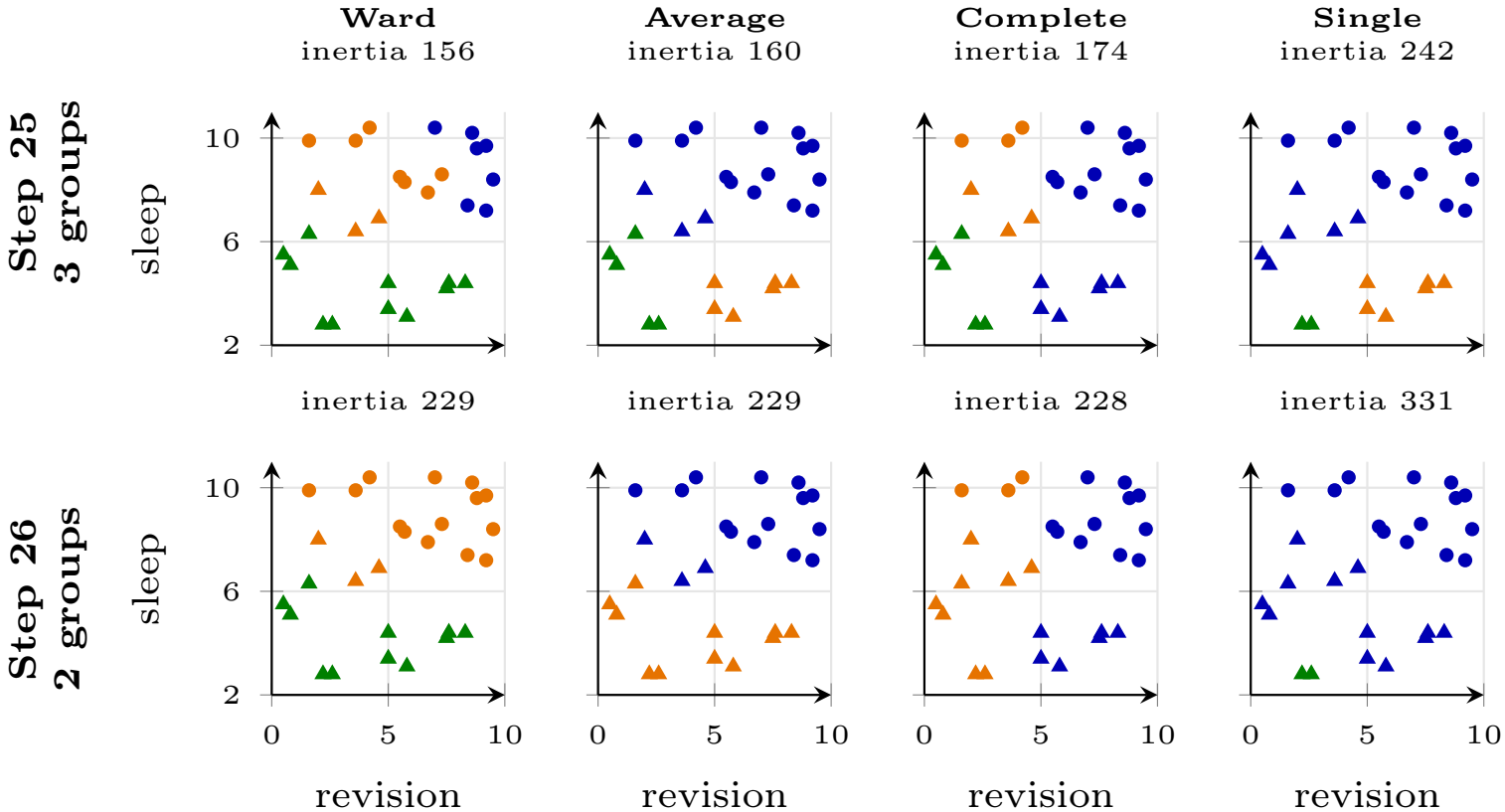


Algorithm : start with every student as its own group. Each step merges the **two closest groups**.

"Closest" can mean different things (next slide).

With the final 2 groups : inertia 229, worse than k-means (206), because a merge is never undone.

Hierarchical clustering : the linkage



linkage	how two groups are compared	sizes, step 25	sizes, step 26
Ward	the merge that adds the least inertia	11, 10, 7	17, 11
average	mean distance between the two groups	17, 6, 5	17, 11
complete	the farthest pair of points	17, 6, 5	17, 11
single	the closest pair of points	20, 6, 2	26, 2

Hierarchical clustering : pros and cons

Use it when the dataset is small and you do not know k.

Pros

- No k to choose in advance
- Same result every time

Cons

- Slow and memory hungry for big data : all pairs of points
- Depends on the distance and on how groups are compared
- Needs feature scaling

Hyperparameters : the distance between points, the linkage, the cut.

Principal Component Analysis (PCA) : the idea

Goal : replace the d columns by $k \leq d$ **linear combinations** $z = w_1 x_1 + \cdots + w_d x_d$ of them, losing as little as possible.

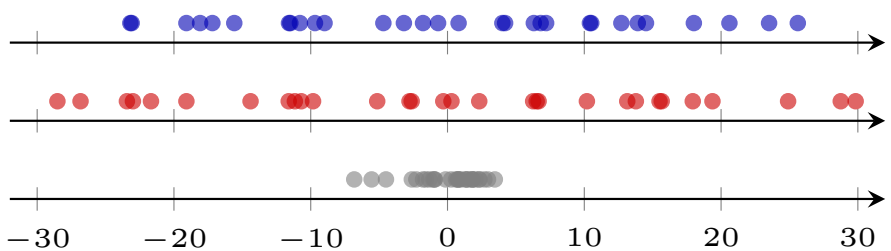
	km from the average village	
village	east x_1	north x_2
1	-23.2	-16.6
2	-10.8	-3.9
3	7.2	7.6
4	25.6	15.4
⋮	⋮	⋮

$$z_1 = 0.83\, x_1 + 0.55\, x_2$$
$$z_2 = -0.55\, x_1 + 0.83\, x_2$$

keep z_1, drop z_2

village	PC1 z_1	PC2 z_2
1	-28.5	-1.0
2	-11.2	2.7
3	10.2	2.3
4	29.8	-1.3
⋮	⋮	⋮

How : choose the weights $w_1, \ldots, w_d$ so the new column **varies the most** :



east alone: variance 200

PC1: variance **286** (max)

PC2: variance 6 → drop

PCA : the algorithm

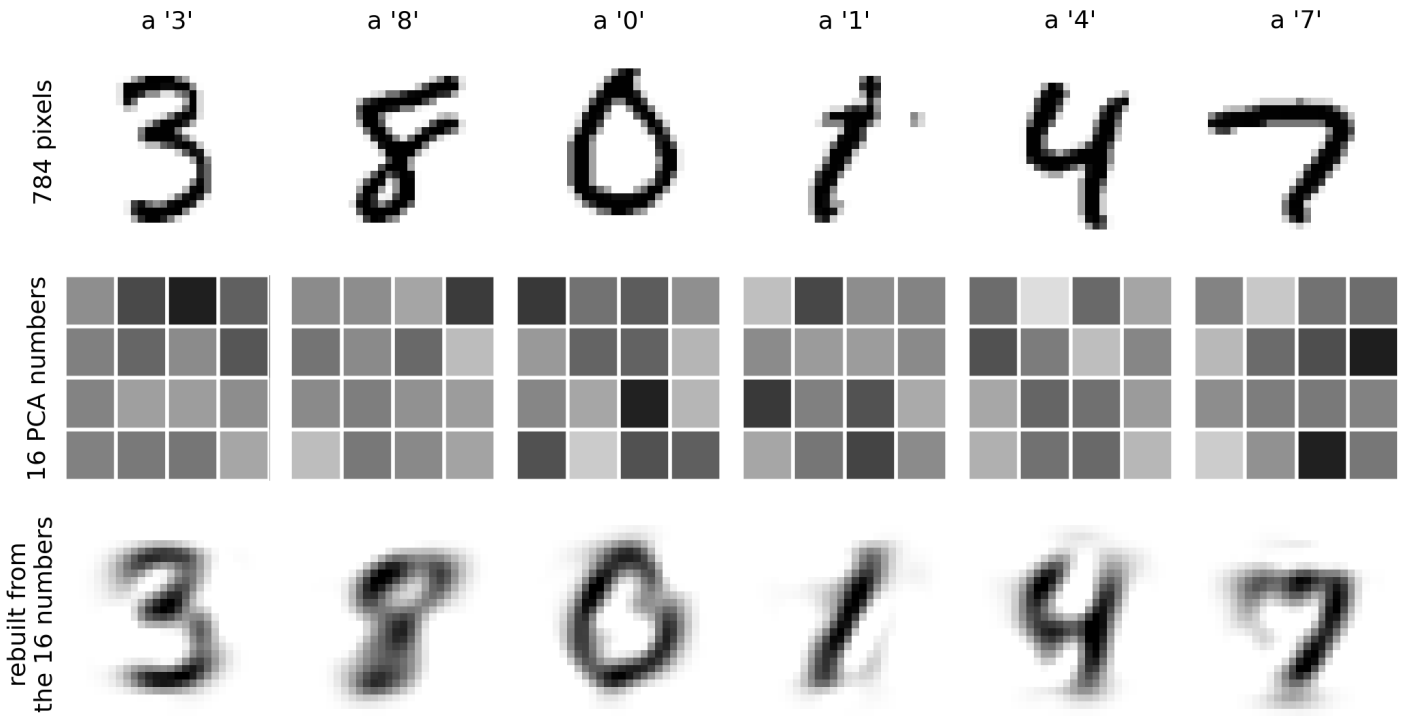
Data : n rows, d features (X is $n \times d$).

1. **Center the data** : subtract each column's mean to get X_c ($n \times d$). If the columns have different units, **standardize** instead : $x' = (x - \mu)/\sigma$.
2. **Compute the covariance matrix** : $C = X_c^\top X_c/(n-1)$, a $d \times d$ matrix.
3. **Eigen-decompose** C ; sort eigenvalues from largest to smallest.
4. **Keep** the eigenvectors of the top k eigenvalues as W_k ($d \times k$).
5. **Project** : $Z = X_c W_k$ ($n \times k$), which gives the new k columns.
6. **Reconstruct** : $\hat{X}_c = Z\, W_k^\top$ ($n \times d$), an approximation of X_c from only k columns.

No need to memorize these steps and equations.

PCA on images : 784 features become 16

Data : PCA fitted on the 60 000 training images of **all ten digits**.



PCA before a classification

Logistic regression on the 60 000 training digits, tested on 10 000 new ones.

	784 pixels	**16 PCA numbers**
Iterations	52	52
Time per iteration	21 ms	4 ms
Training time	1.1 s	0.2 s
Train accuracy	92.9%	85.3%
Test accuracy	**92.2%**	**85.8%**

PCA : pros and cons

Use for : compression, 2D plots, preprocessing, anomaly detection

Pros

- Fast, no iterations
- Uncorrelated new features

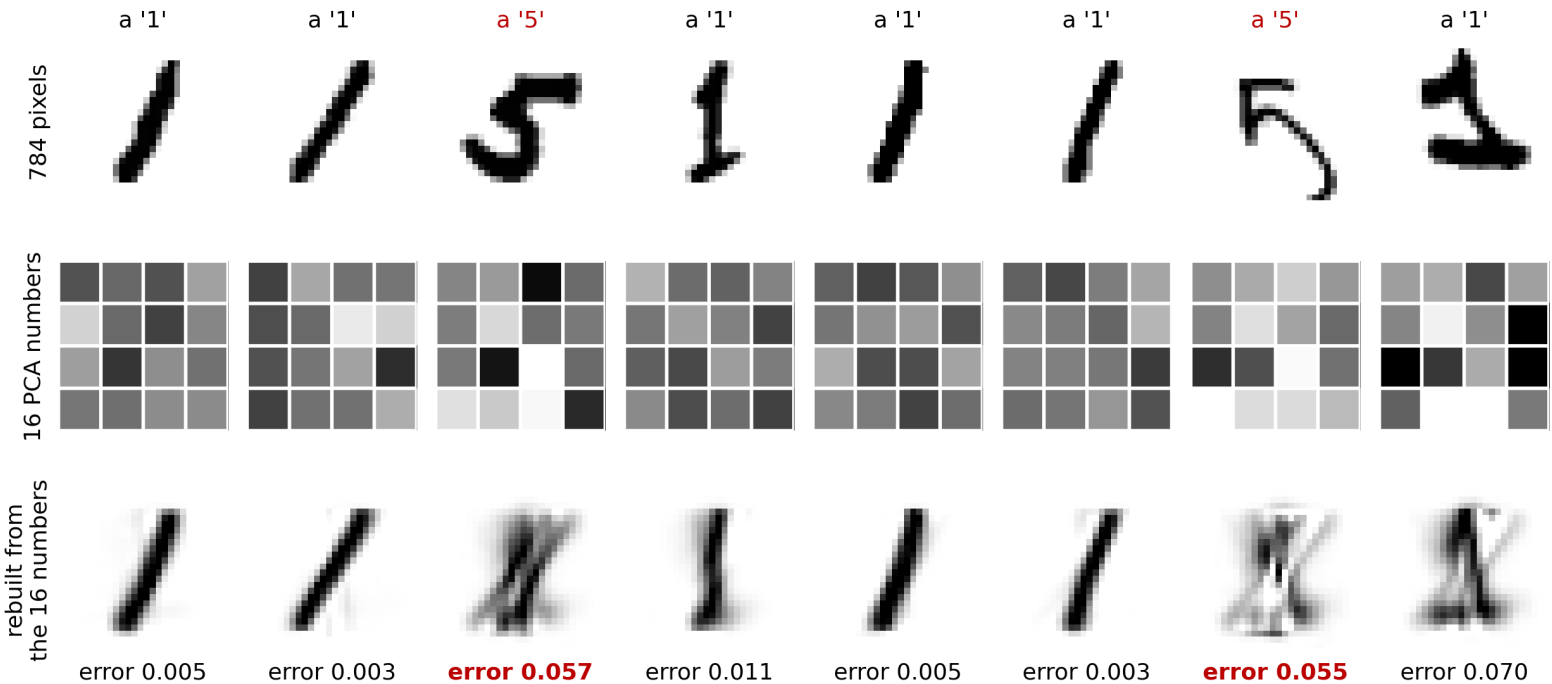
Cons

- Linear structure only
- Ignores labels

Hyperparameters : the number of components k.

PCA : can the reconstruction error find outliers ?

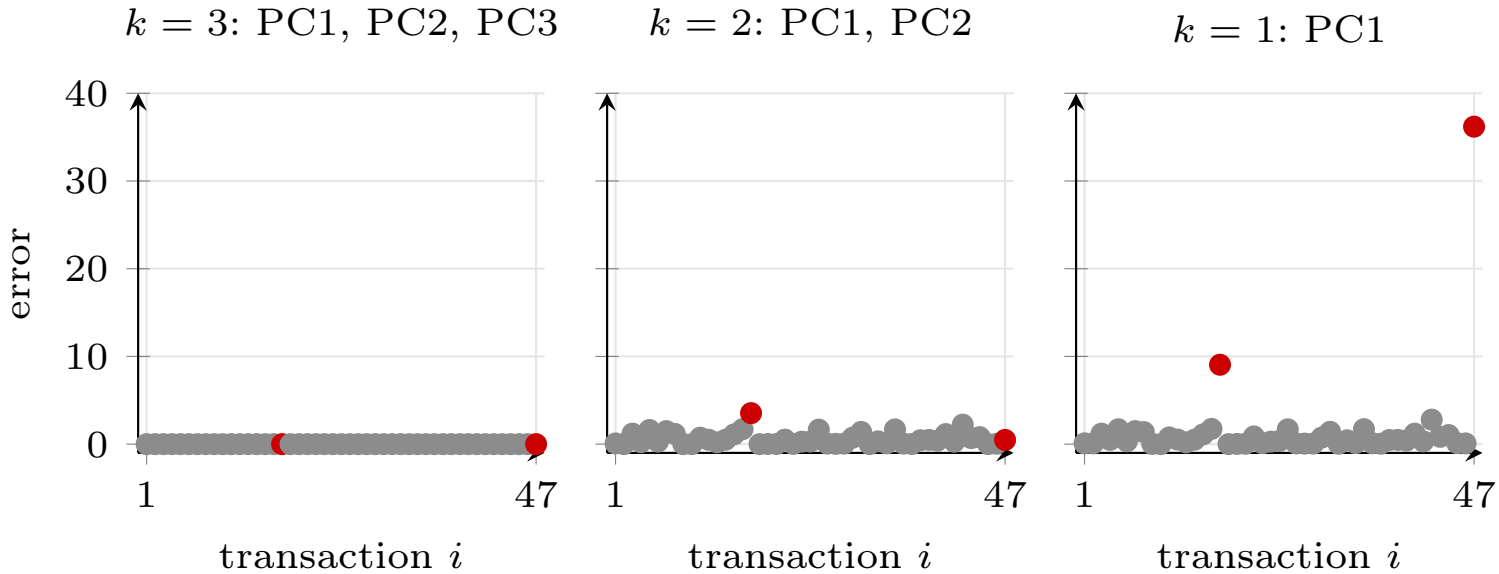
Data : **not** the 60 000 of the previous slide, only the 6742 images of a 1 plus two 5s (6744). PCA with 16 numbers fitted on them.



The 5s rank 6th and 7th of 6744 ; only odd 1s (like the last) score higher.

PCA : can the reconstruction error find outliers ?

Idea : Odd points are rebuilt badly :
Data : 47 transactions (hour, euro, km), standardized.
Rebuild equation : $\hat{X}_c = Z\,W_k^\top$.



- $k = 3 = d$: exact rebuild, error 0 : nothing to detect.
- $k = 2$: the 265 euro one stands out, the 400 km one does not.
- $k = 1$: both odd ones have the largest errors.

Isolation forest : the idea

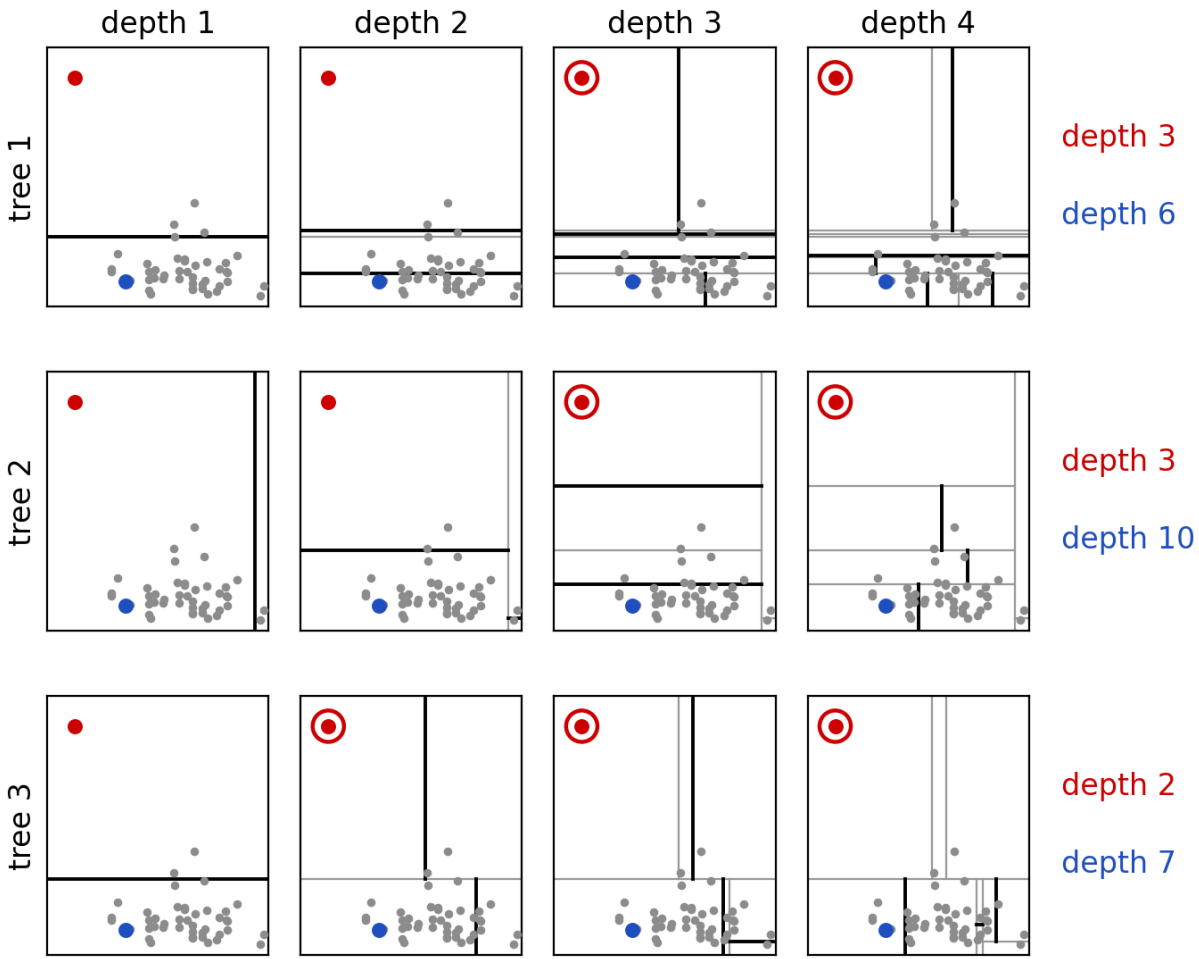
Idea : anomalies are few and different, so random cuts isolate them quickly.

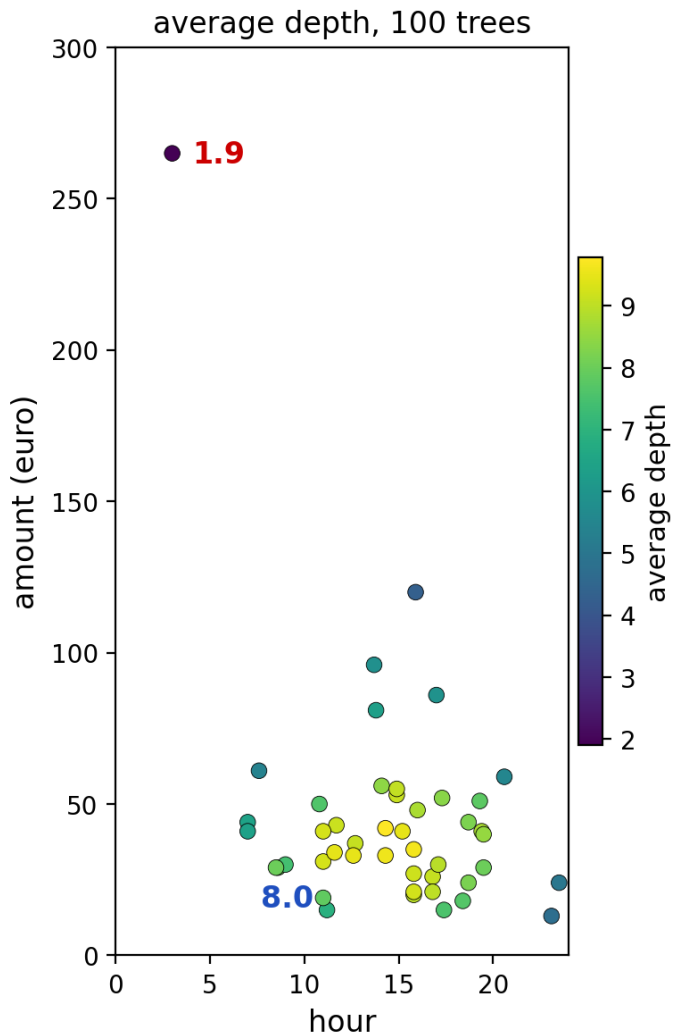
Steps

1. Build a **random tree** : divide (cut) the points at a random feature and a random value (between the min and max of the group), then do the same in each group, until every point is alone.
2. Record the **depth** of each point : the number of cuts needed to isolate it.
3. **Repeat** with many new random trees (about 100).
4. **Average** the depth of each point over all the trees.
5. The **lowest averages** are the anomalies.

Liu, Ting & Zhou, "Isolation Forest" (2008) [14]

Isolation forest : random trees





Isolation forest : guarantees, pros and cons

Use it when there are no labeled anomalies, many examples and many features.

Pros

- Fast, even with many features
- No distances : no feature scaling
- No labels needed

Cons

- A score, not an answer : the threshold must be chosen
- Similar anomalies grouped together look normal

Hyperparameters : the number of trees, the threshold, ...

Chapter 4: Machine Learning

Motivation

Data

Definitions

Supervised learning

Unsupervised learning

Performance metrics

How do we know if a model is good ?

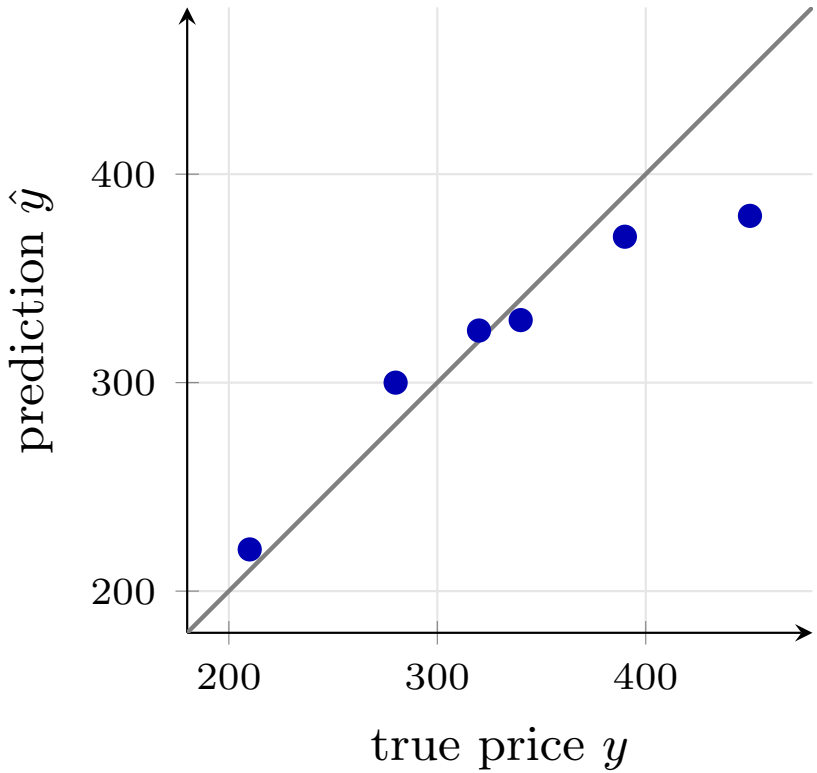
- Compare the predictions with the truth, on **unseen data**.
- The right metric depends on the task and on what a mistake costs.

Task	**Question**
Regression	How far are the predictions ?
Classification	What kind of mistakes ?
Clustering	Compact and well separated ?
Dimensionality reduction	Representative of the data ?
Anomaly detection	Are the flagged cases real ?

Regression : mean absolute error (MAE)

y	$\hat{y}$	$\hat{y} - y$	$\lvert\hat{y} - y\rvert$
210	220	+10	10
390	370	−20	20
340	330	−10	10
280	300	+20	20
450	380	−70	70
320	325	+5	5

price in k euro (6 test apartments)



$$\text{MAE} = \frac{1}{n}\sum_i |\hat{y}^{(i)} - y^{(i)}| = \frac{135}{6} = 22.5 \text{ k euro}.$$

This means, on average, the prediction is off by 22.5 k euro.

Regression : root mean squared error (RMSE)

Same apartments, errors $+10$, -20, -10, $+20$, -70, $+5$.

$\text{MSE} = \frac{1}{n}\sum_i (\hat{y}^{(i)} - y^{(i)})^2 = \frac{5925}{6} = 987.5 \text{ k euro}^2$,

$\text{RMSE} = \sqrt{\text{MSE}} = 31.4$ k euro In RMSE the large errors count more than in MAE.

Regression : the coefficient of determination R^2

Compare the model with the simplest one : the mean price (331.7 k euro).

$$R^2 = 1 - \frac{\sum_i (\hat{y}^{(i)} - y^{(i)})^2}{\sum_i (\bar{y} - y^{(i)})^2} = 1 - \frac{5925}{35083} = 0.83$$

- $R^2 = 1$: good model.
- $R^2 = 0$: bad model, because predicting the average output is as accurate.

Classification : the confusion matrix

Running example : 1000 card transactions, 10 are frauds. The model flags 20 of them.

	flagged	**not flagged**
fraud (10)	8 TP	2 FN
not fraud (990)	12 FP	978 TN

TP true positive : fraud, flagged.
FN false negative : fraud, missed.
FP false positive : false alarm.
TN true negative : normal, not flagged.

Classification : accuracy

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{n} = \frac{8 + 978}{1000} = 98.6\% \text{ : the share of correct answers.}$$

A trap : a model that never flags anything has accuracy $\frac{990}{1000} = 99\%$ and catches 0 frauds.

- Accuracy is misleading when the classes are very unbalanced (frauds, diseases, defects).
- It treats all mistakes alike : a missed fraud and a false alarm have the same cost.

Use it when the classes are roughly balanced and the mistakes have the same cost.

Classification : precision and recall

Precision $= \dfrac{\text{TP}}{\text{TP} + \text{FP}} = \dfrac{8}{20} = 40\%$

Of the flagged transactions, how many are real frauds ?

Important when a false alarm is costly.

Example : blocking the card of an honest customer.

Recall $= \dfrac{\text{TP}}{\text{TP} + \text{FN}} = \dfrac{8}{10} = 80\%$

Of the real frauds, how many did we catch ?

Important when a miss is costly.

Example : an undetected fraud, a missed disease.

Classification : the F1 score

One number for both : the harmonic mean of precision P and recall R.

$$F_1 = \frac{2PR}{P+R} = \frac{2 \times 0.4 \times 0.8}{0.4 + 0.8} = 0.53$$

Good for comparing models on unbalanced data with a single number.

Unsupervised learning : performance metrics

No labels, so no "true answer" to compare with : we measure how well the structure fits the data.

Clustering : Inertia (squared distances to the centroids)

$$\sum_i \|x^{(i)} - \mu_{c(i)}\|^2$$

Dimensionality reduction : Reconstruction error :

$$\frac{1}{n}\sum_i \|x^{(i)} - \hat{x}^{(i)}\|^2, \qquad \hat{X} = Z\,W_k^\top + \mu$$

There are many more evaluation metrics, depending on the task.

Chapter 5: Deep Learning

Motivation

Neural networks

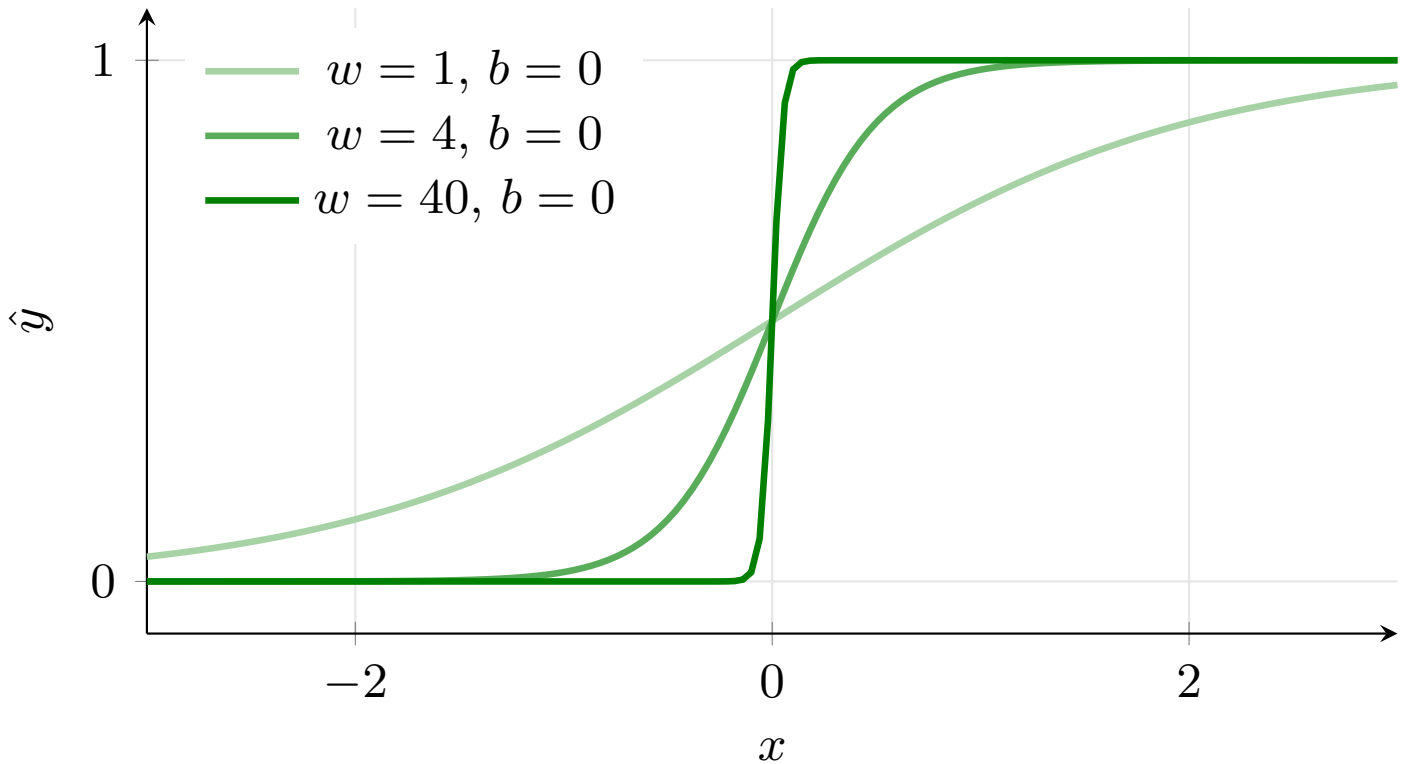
Training

Convolutional neural networks and computer vision

Summary

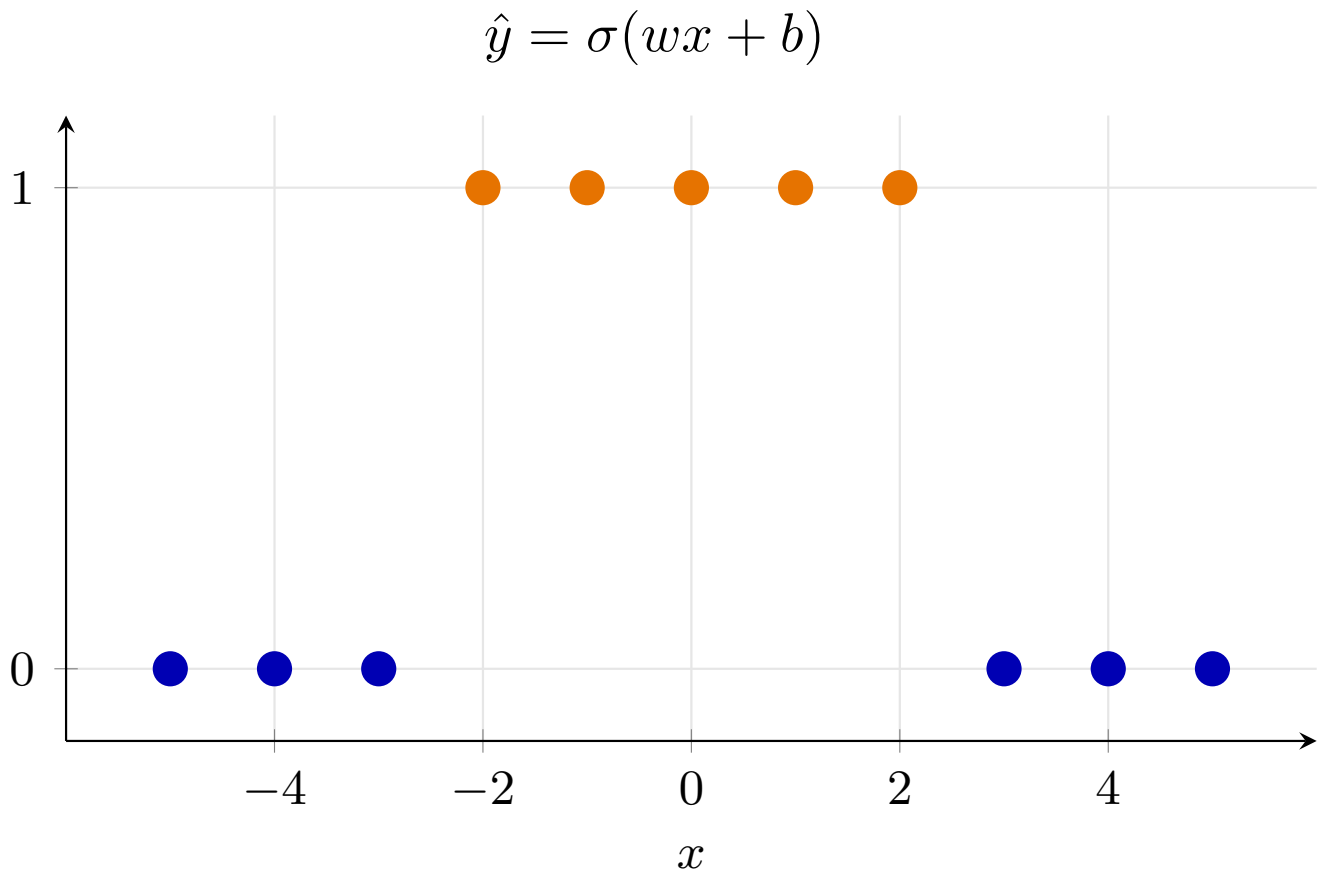
Logistic Regression

$$\hat{y} = \sigma(wx + b) \qquad \text{with} \qquad \sigma(z) = \frac{1}{1 + e^{-z}}$$



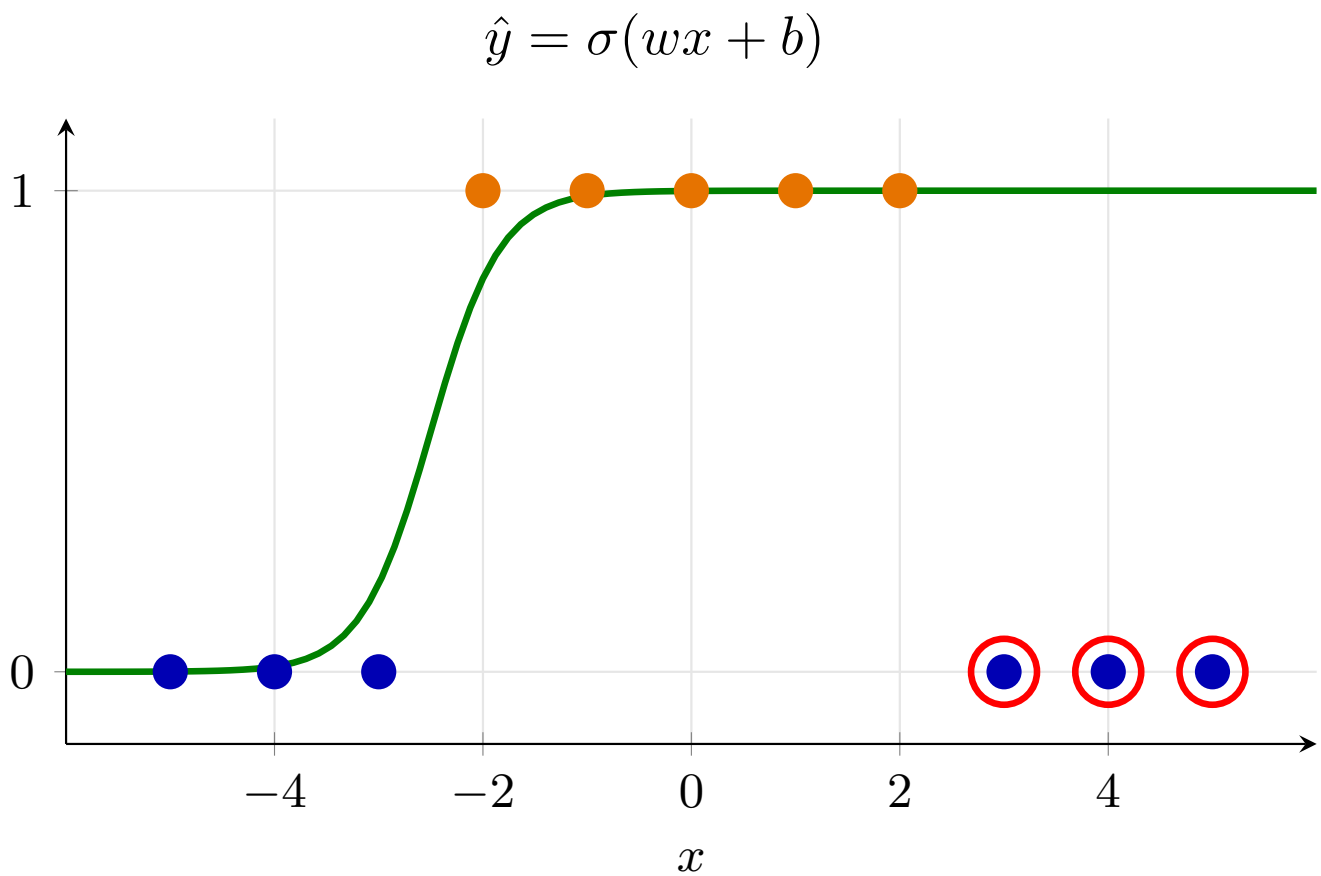
The larger w, the closer to a **step**.

Classification



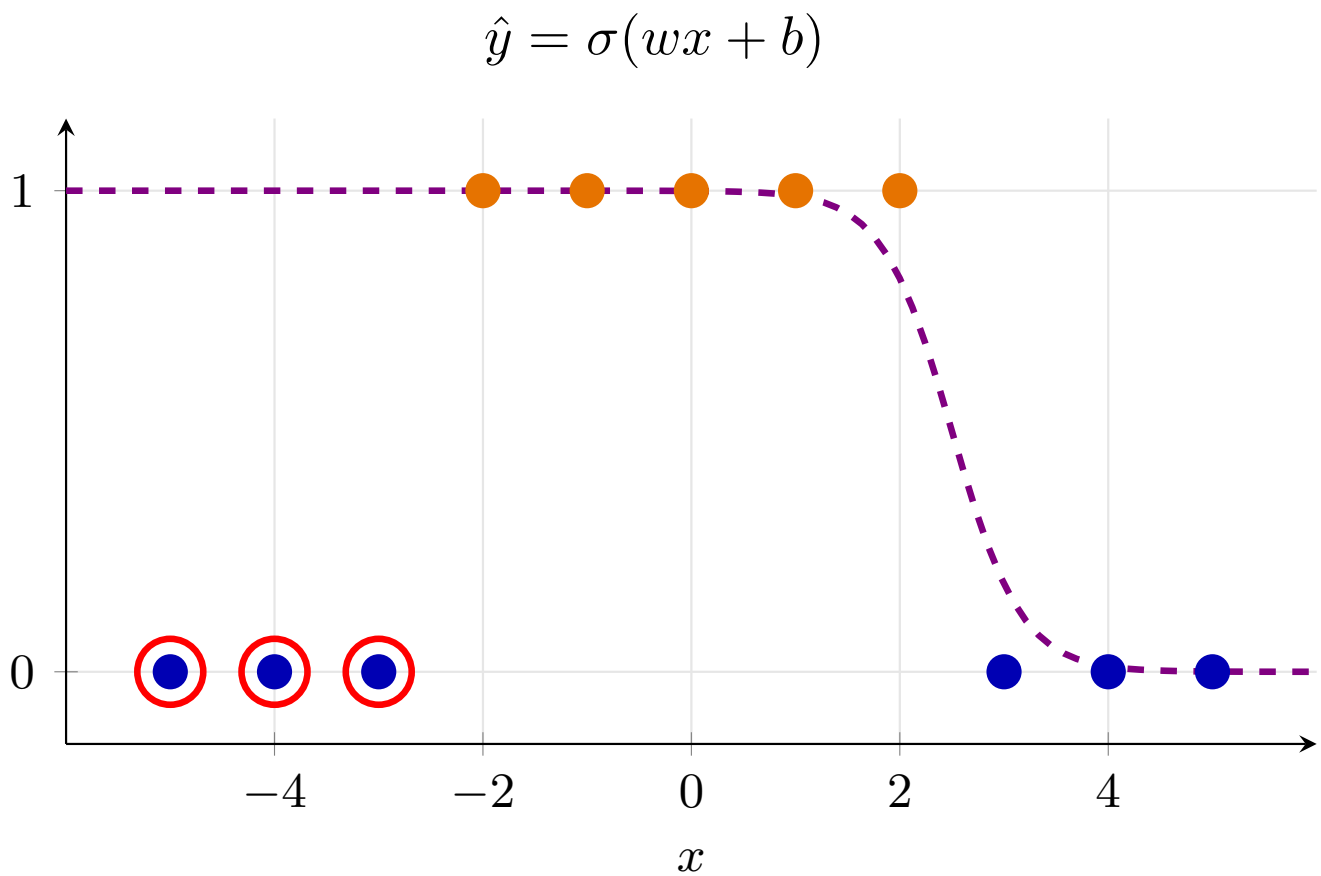
Orange in the middle, blue on both sides.

Classification



$w = 3$, $b = 7.5$: it goes **up**... 3 mistakes.

Classification

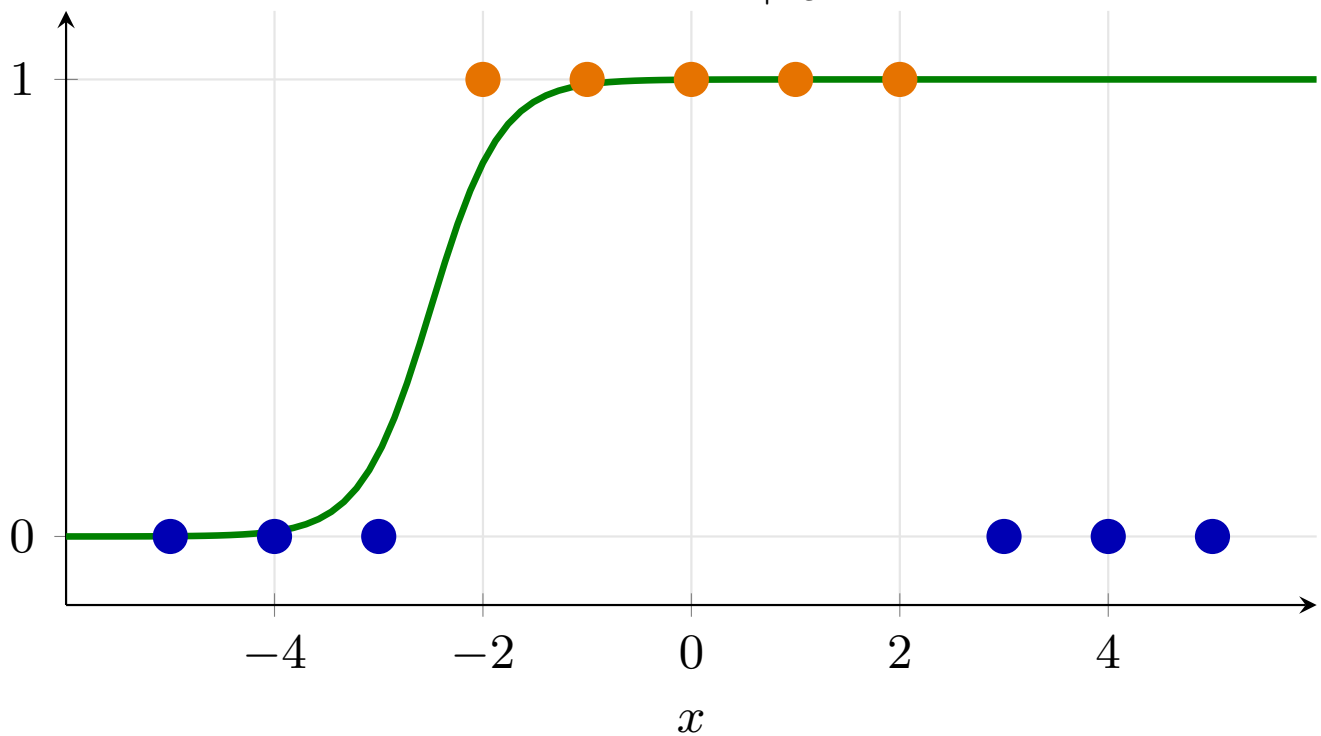


$w = -3$, $b = 7.5$: . . . or **down**, never both. 3 mistakes.

Adding two sigmoids

$$\hat{y} = \sigma(3x + 7.5) + \sigma(-3x + 7.5) - 1$$

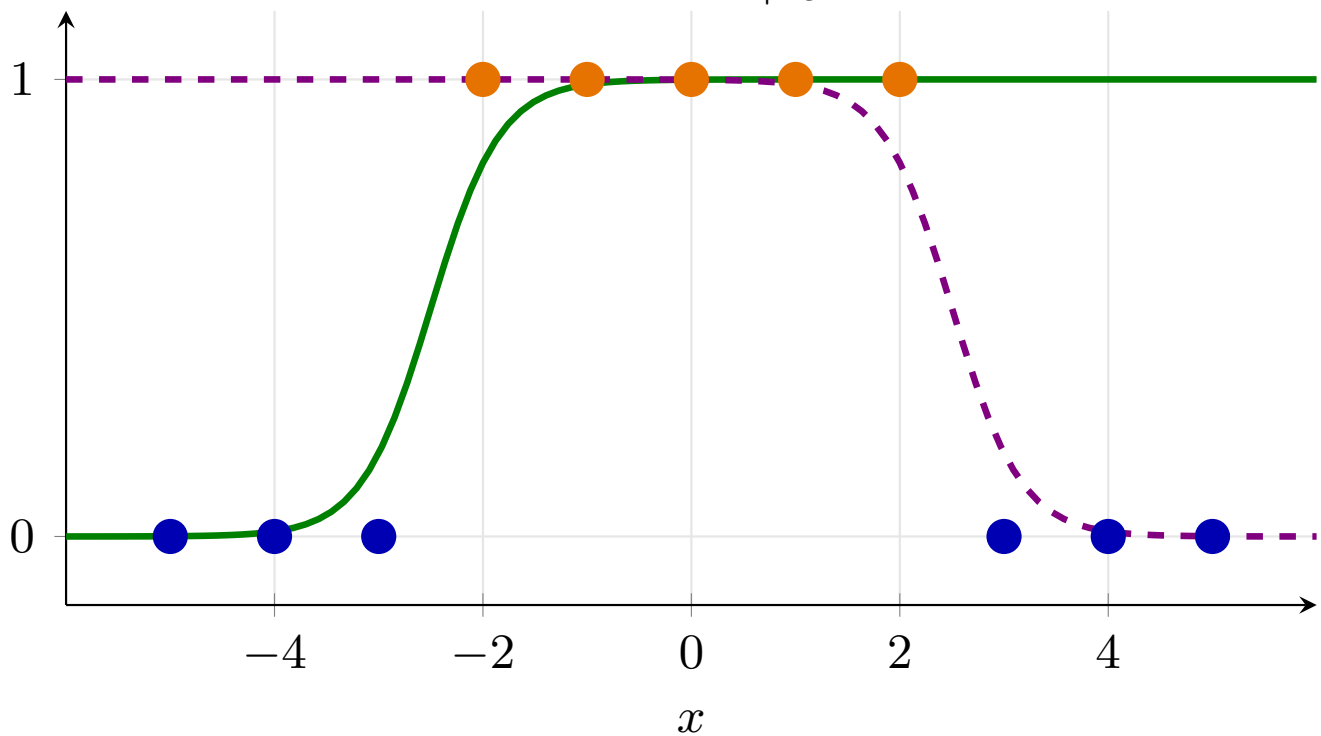
$$\text{with } \sigma(z) = \frac{1}{1 + e^{-z}}$$



Adding two sigmoids

$$\hat{y} = \sigma(3x + 7.5) + \sigma(-3x + 7.5) - 1$$

$$\text{with } \sigma(z) = \frac{1}{1 + e^{-z}}$$

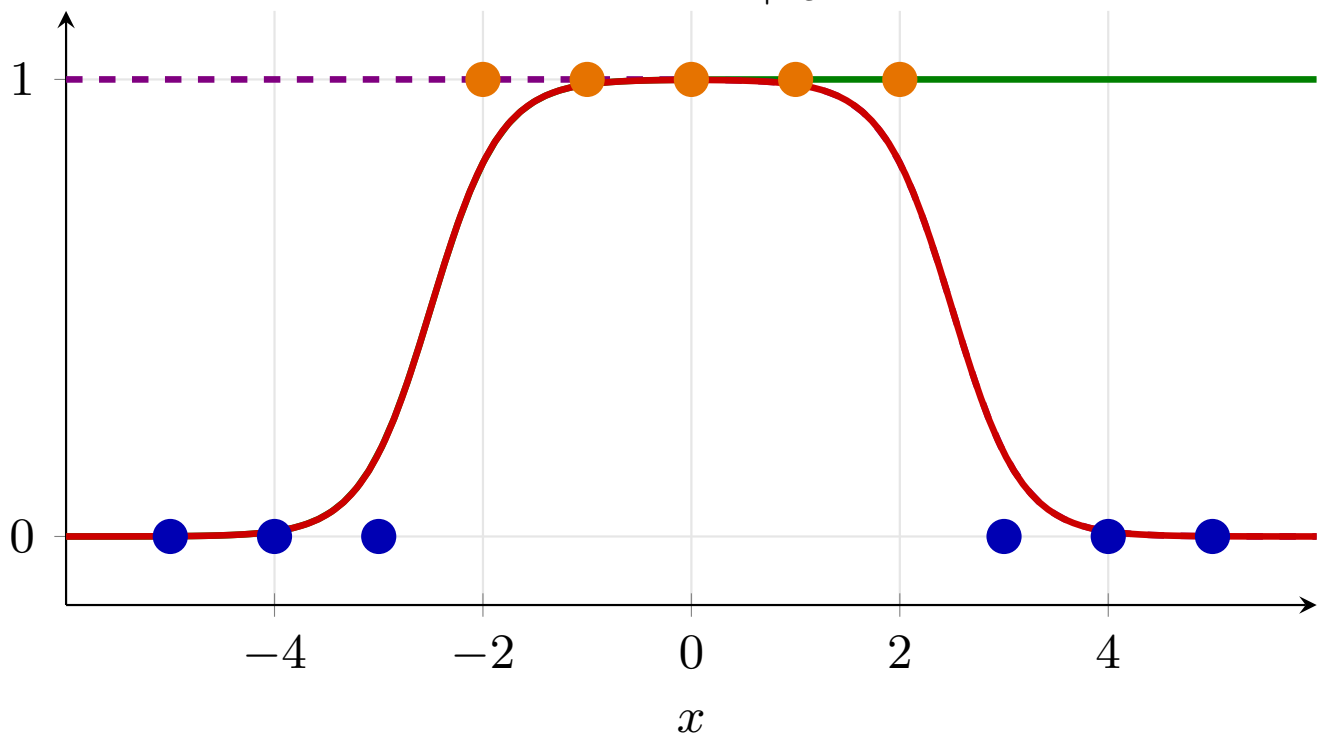


After M. Nielsen, *Neural Networks and Deep Learning*, ch. 4 [16]

Adding two sigmoids

$$\hat{y} = \sigma(3x + 7.5) + \sigma(-3x + 7.5) - 1$$

$$\text{with } \sigma(z) = \frac{1}{1 + e^{-z}}$$



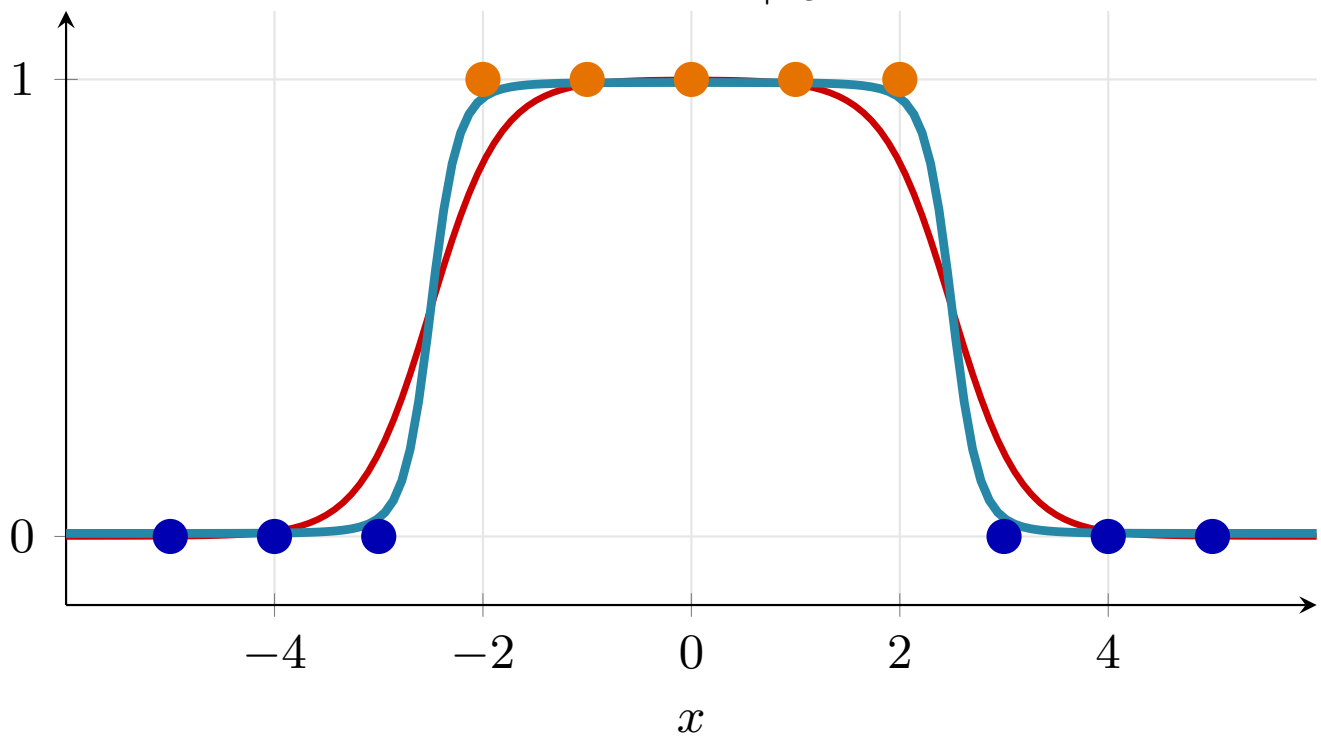
After M. Nielsen, *Neural Networks and Deep Learning*, ch. 4 [16]

Adding two sigmoids

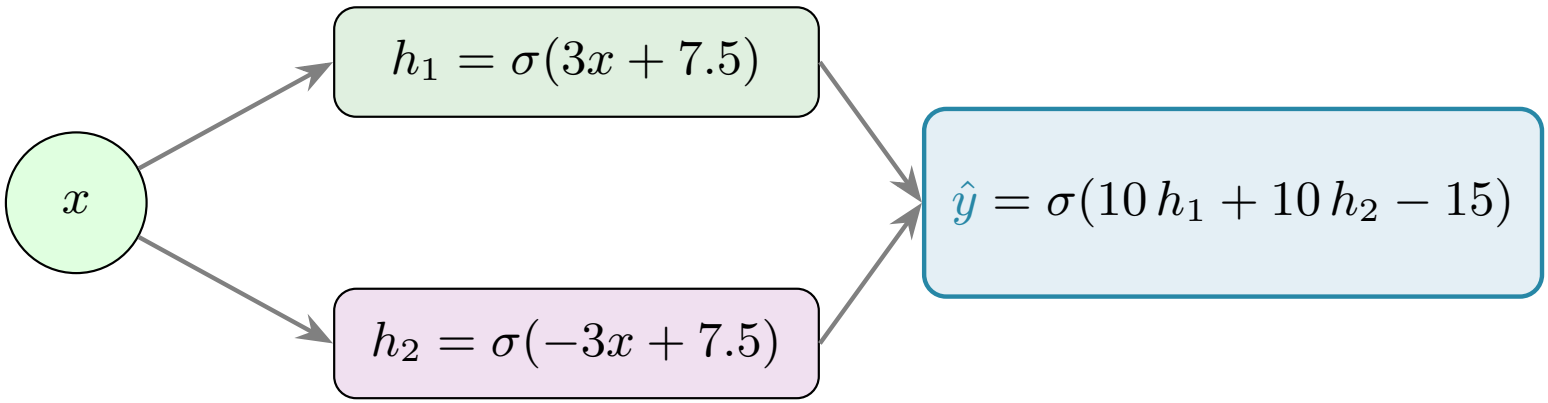
$$\hat{y} = \sigma(3x + 7.5) + \sigma(-3x + 7.5) - 1$$

$$\hat{y} = \sigma\big(10\,\sigma(3x + 7.5) + 10\,\sigma(-3x + 7.5) - 15\big)$$

$$\text{with } \sigma(z) = \frac{1}{1 + e^{-z}}$$



The steps of the computation

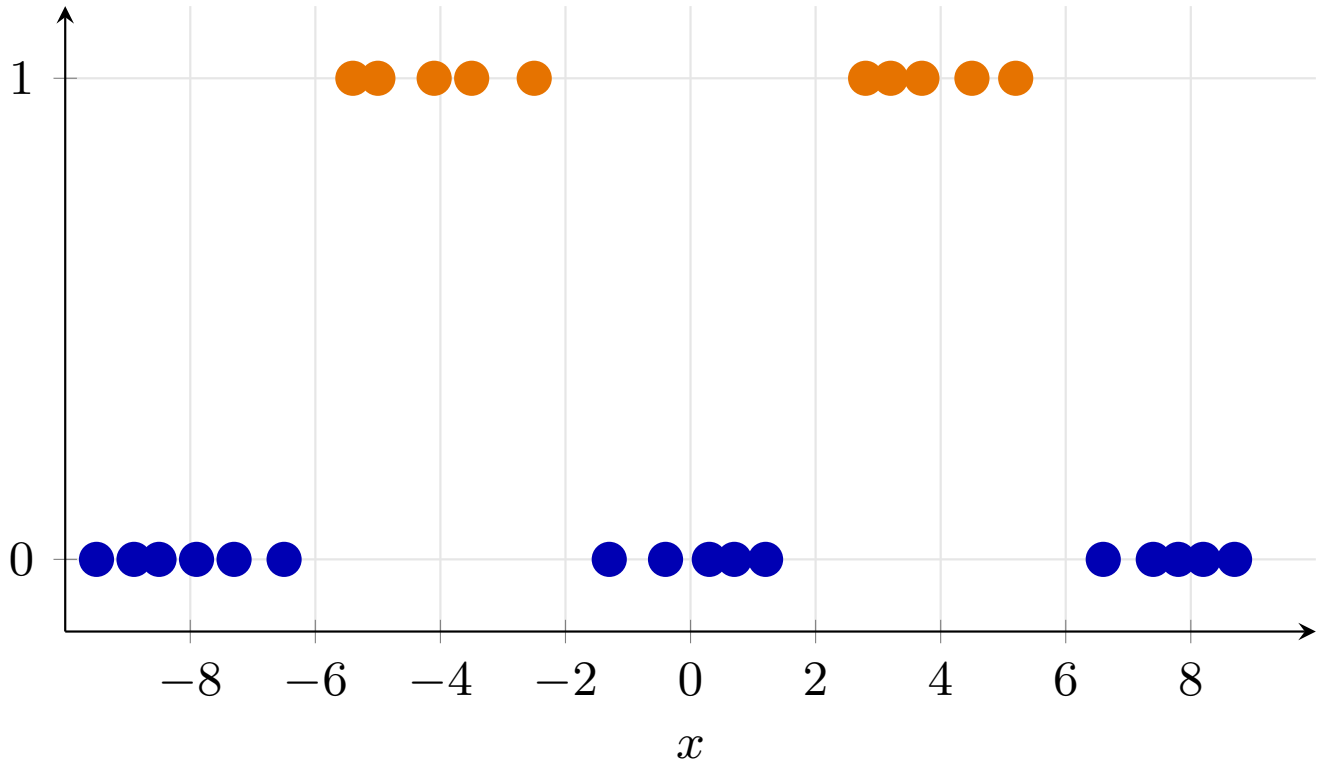


- To guarantee a probability in classification tasks we add one more **sigmoid** operation at the end :

$$\hat{y} = \sigma(10\,h_1 + 10\,h_2 - 15).$$

- Does this look like a **neural network** ?

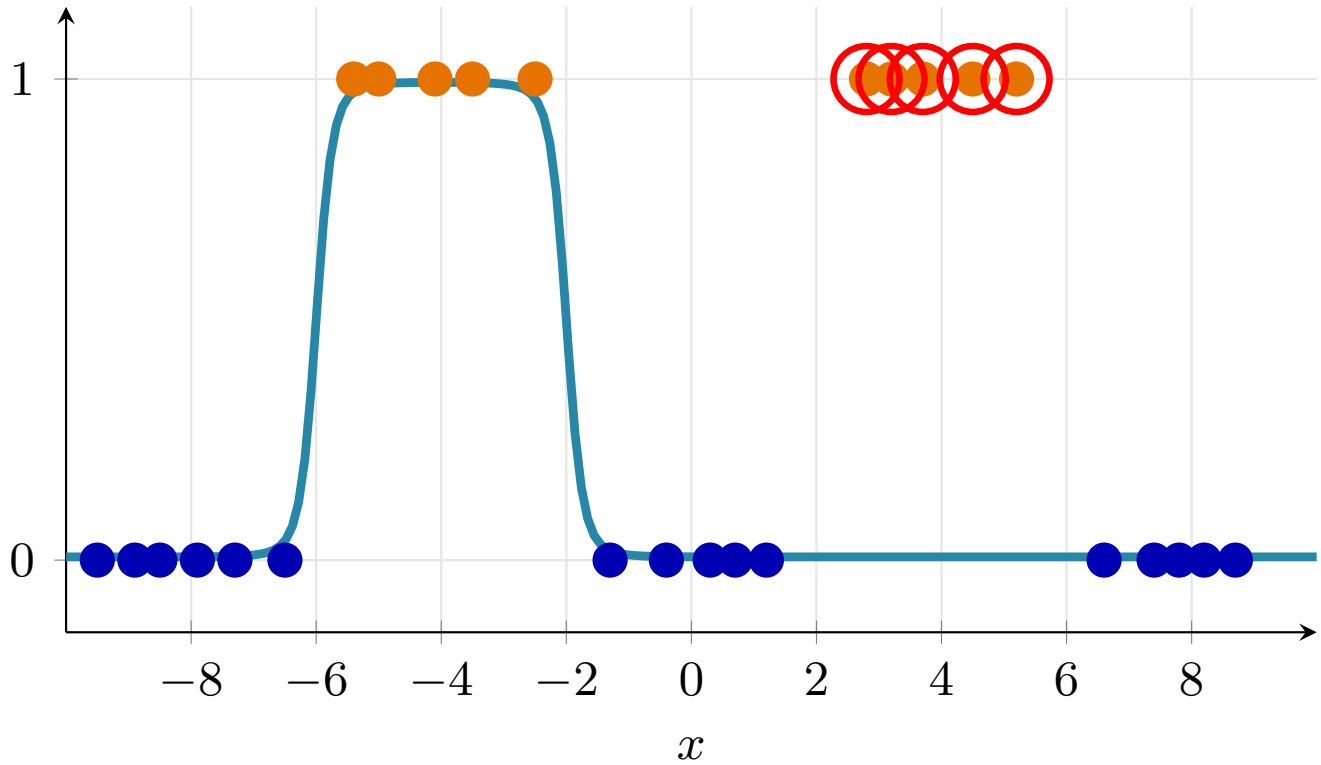
Adding more sigmoids



Two orange groups.

Adding more sigmoids

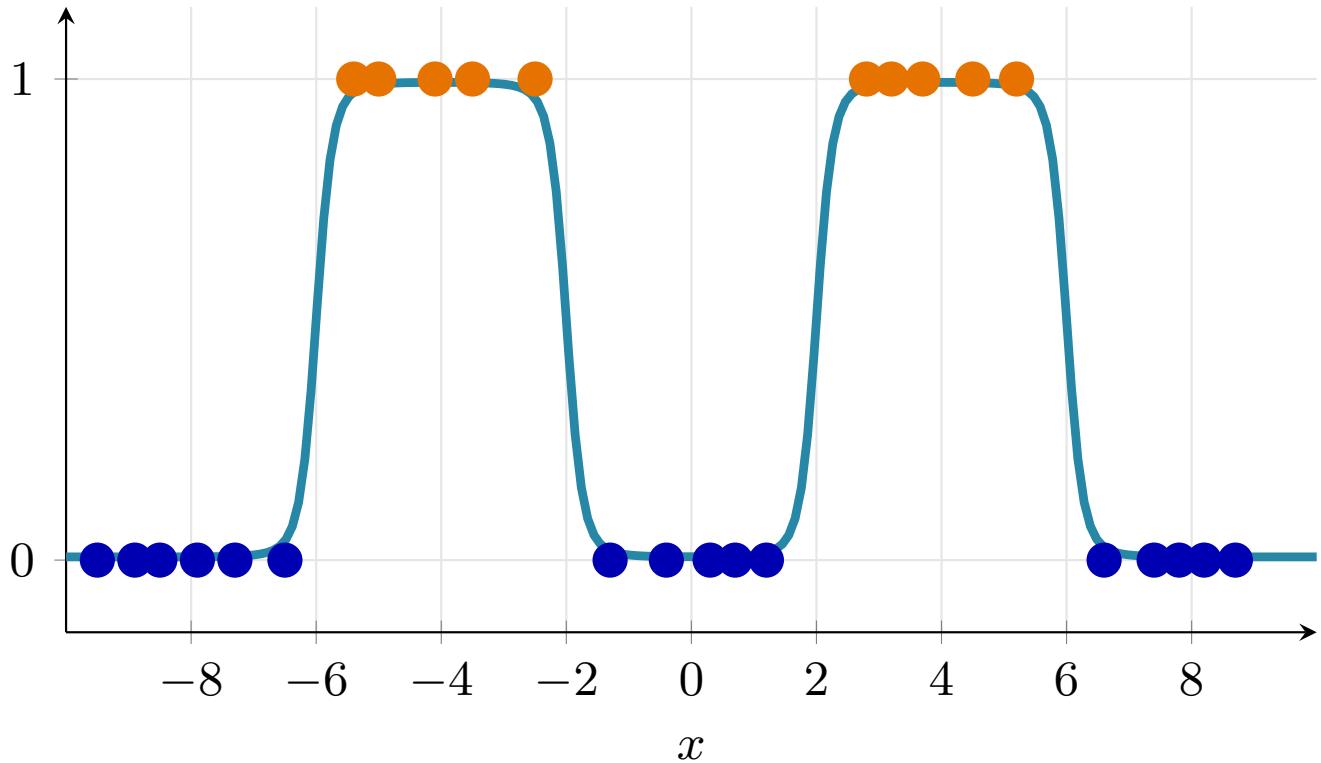
$$\hat{y} = \sigma\big(10\,\sigma(3x + 18) + 10\,\sigma(-3x - 6) - 15\big)$$



2 sigmoids : one bump. 5 mistakes.

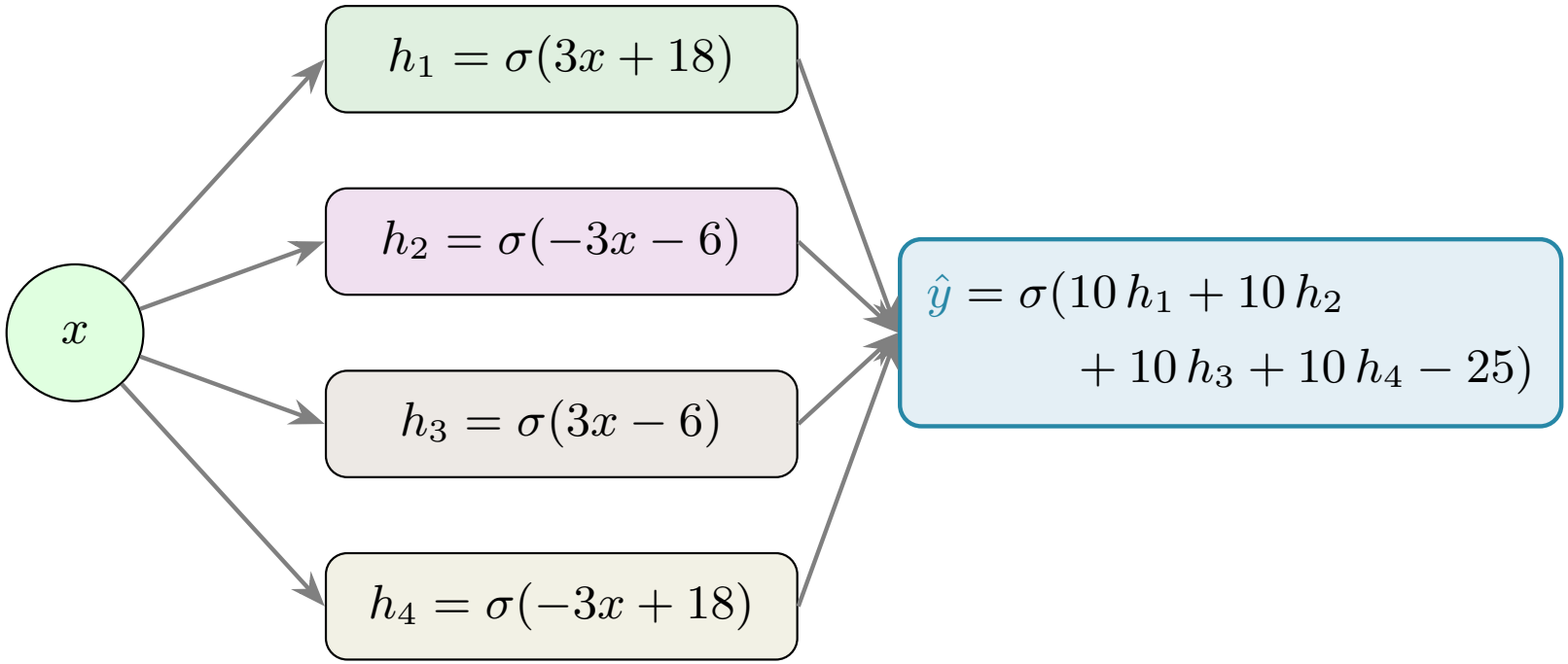
Adding more sigmoids

$$\hat{y} = \sigma\big(10\,\sigma(3x+18) + 10\,\sigma(-3x-6) + 10\,\sigma(3x-6) + 10\,\sigma(-3x+18) - 25\big)$$

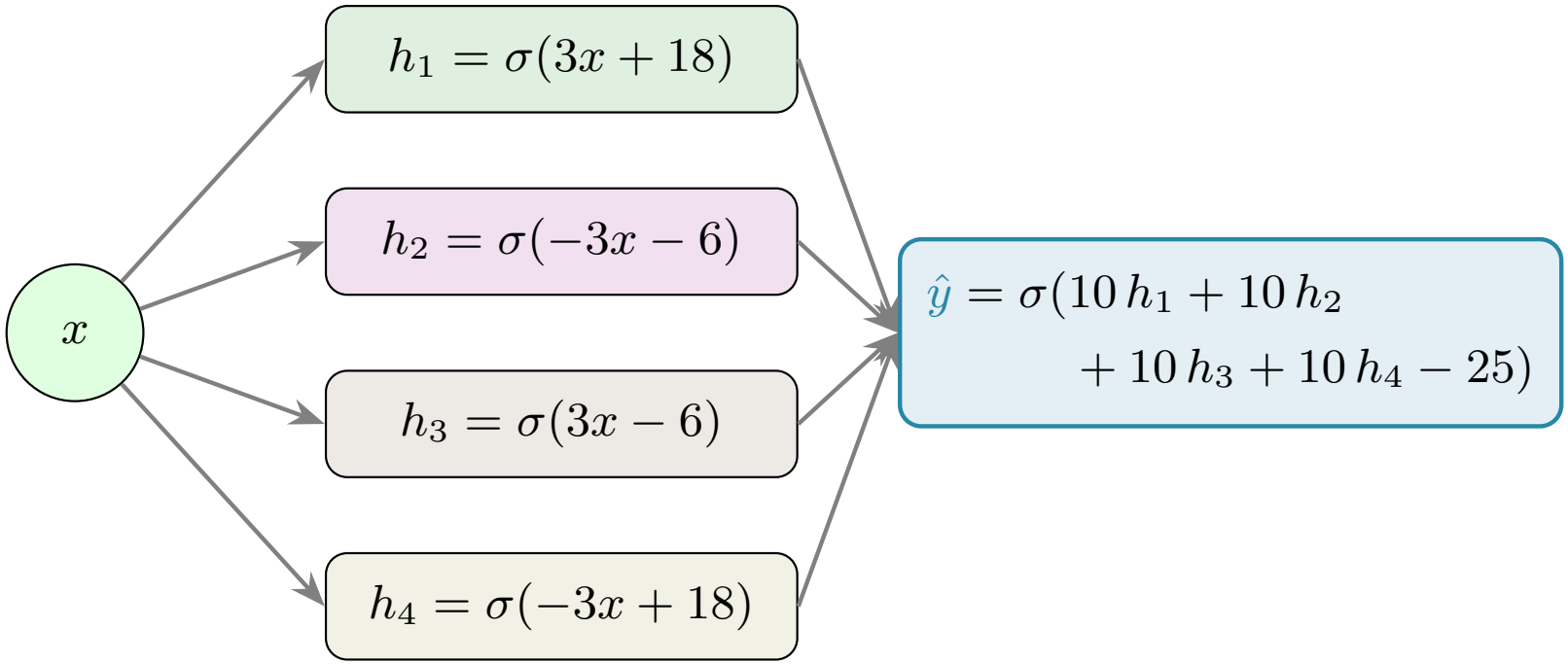


4 sigmoids : two bumps. **0 mistakes**.

Neural network with 4 neurons in 1 layer

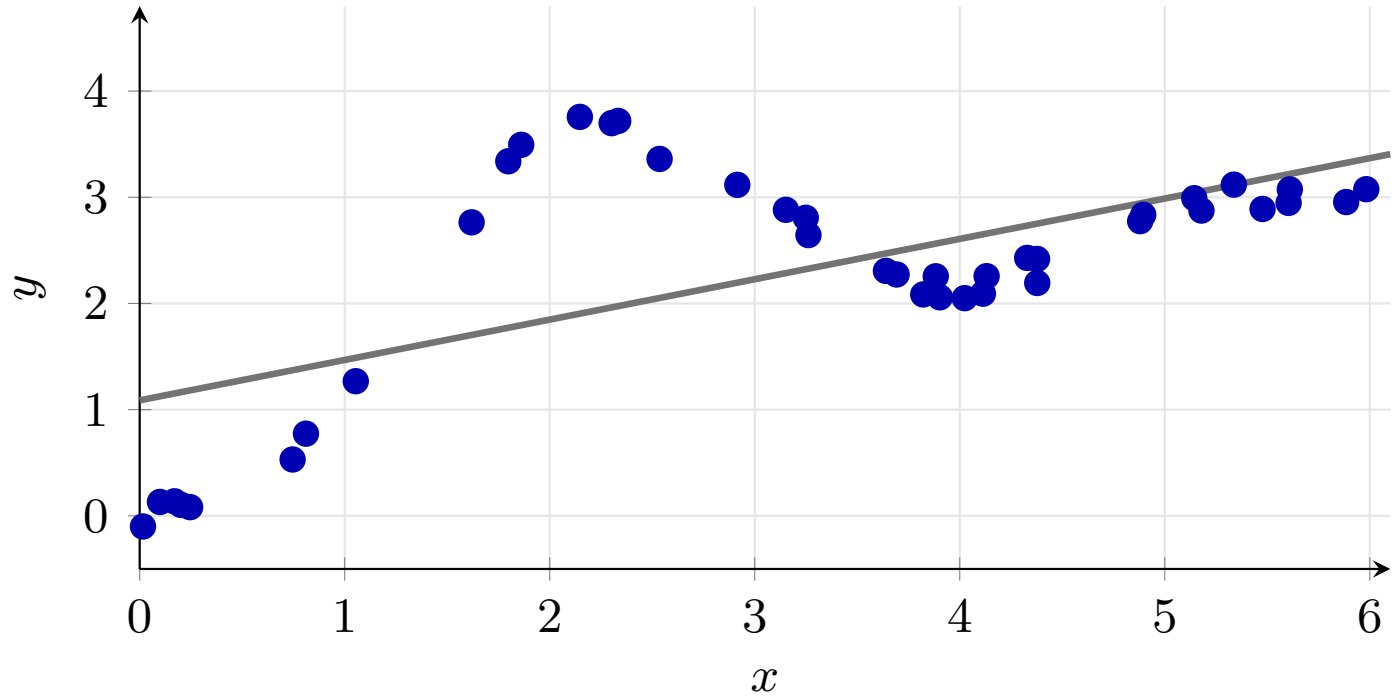


Neural network with 4 neurons in 1 layer



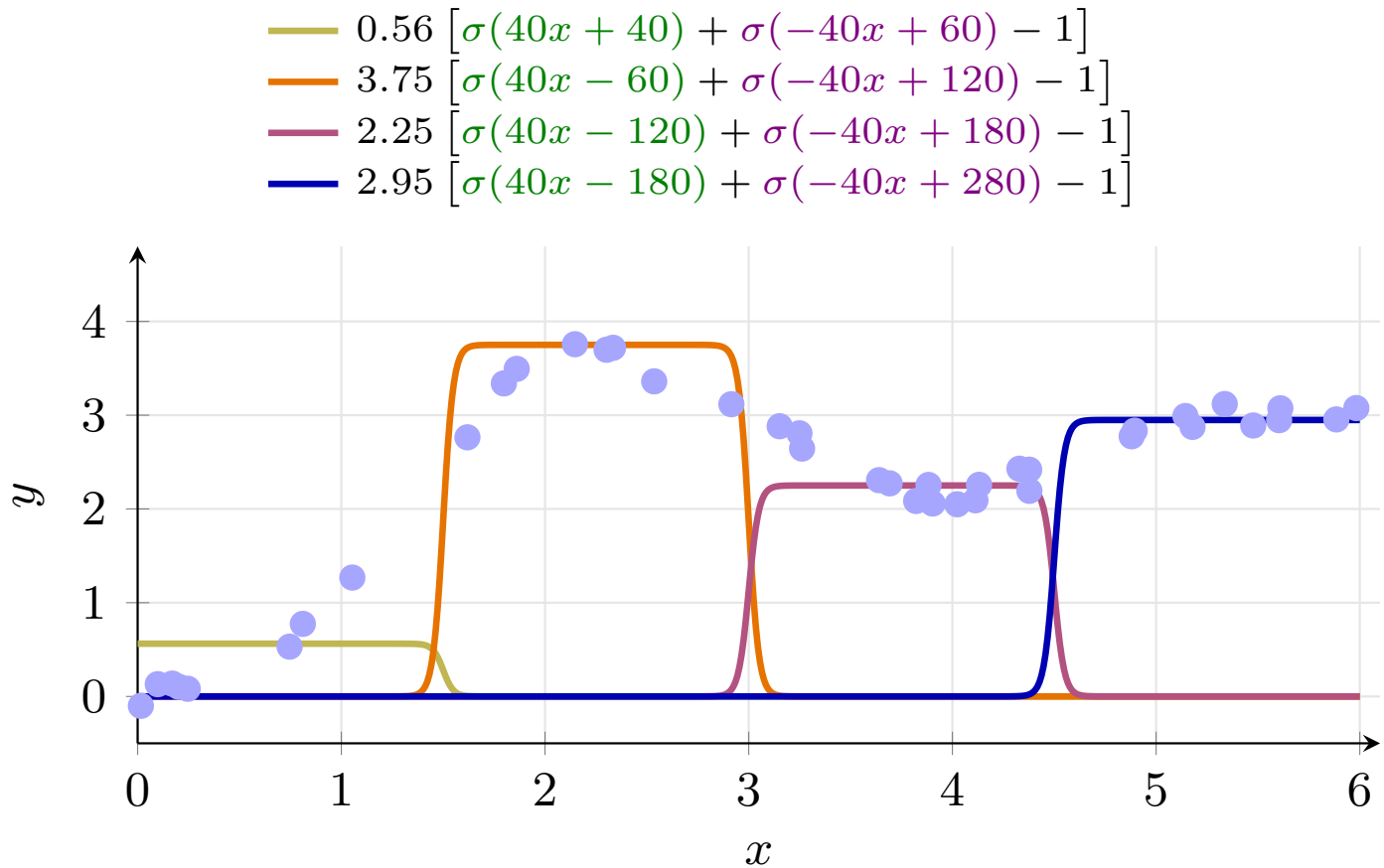
$$h = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} = \sigma\left(\begin{pmatrix} 3 \\ -3 \\ 3 \\ -3 \end{pmatrix} x + \begin{pmatrix} 18 \\ -6 \\ -6 \\ 18 \end{pmatrix}\right) \qquad \hat{y} = \sigma\left(\begin{pmatrix} 10 & 10 & 10 & 10 \end{pmatrix} h - 25\right)$$

A line cannot follow the curve



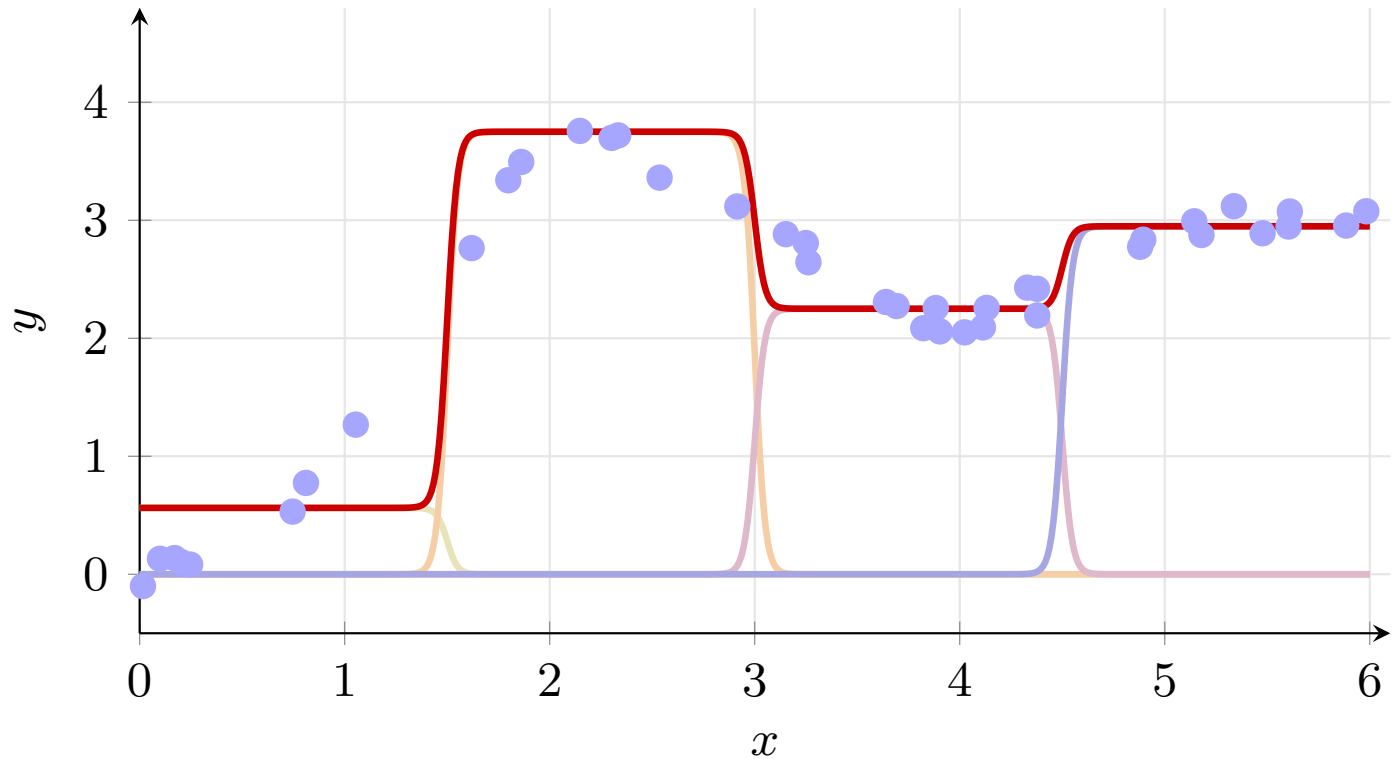
- The best **line** fits poorly : far from most points.
- A **polynomial** ? With d features, degree m needs $O(d^m)$ new features.
  For an image, $d = 784$, $m = 3$: **80 931 145** features.

4 pairs of sigmoids



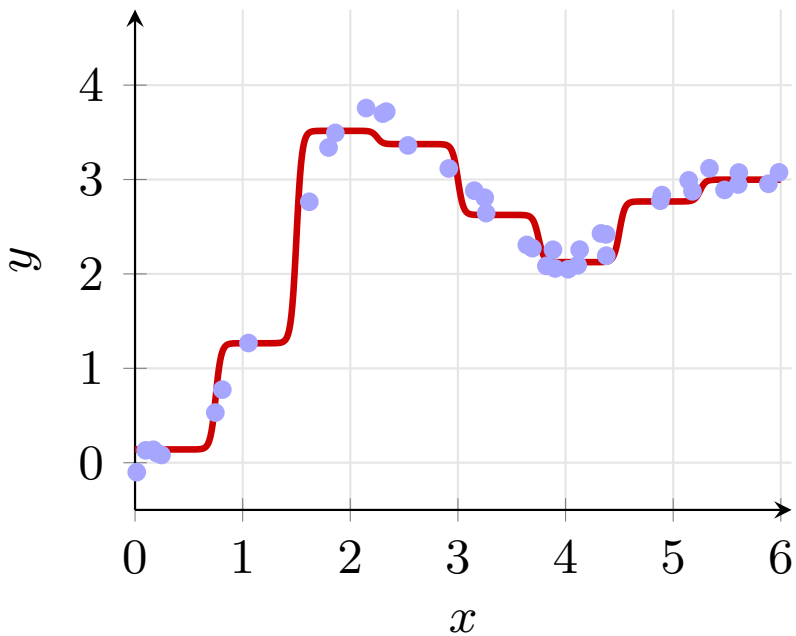
After M. Nielsen, *Neural Networks and Deep Learning*, ch. 4 [16]

Sum of the 4 pairs

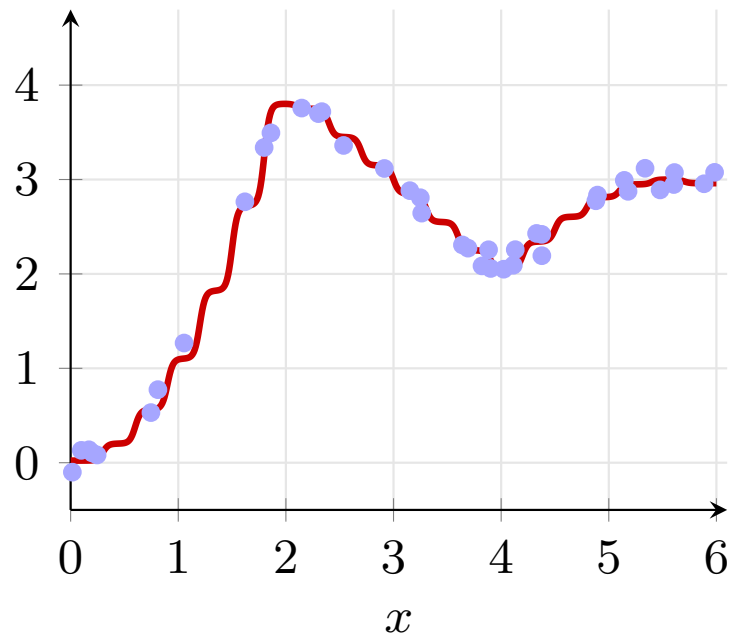


More pairs

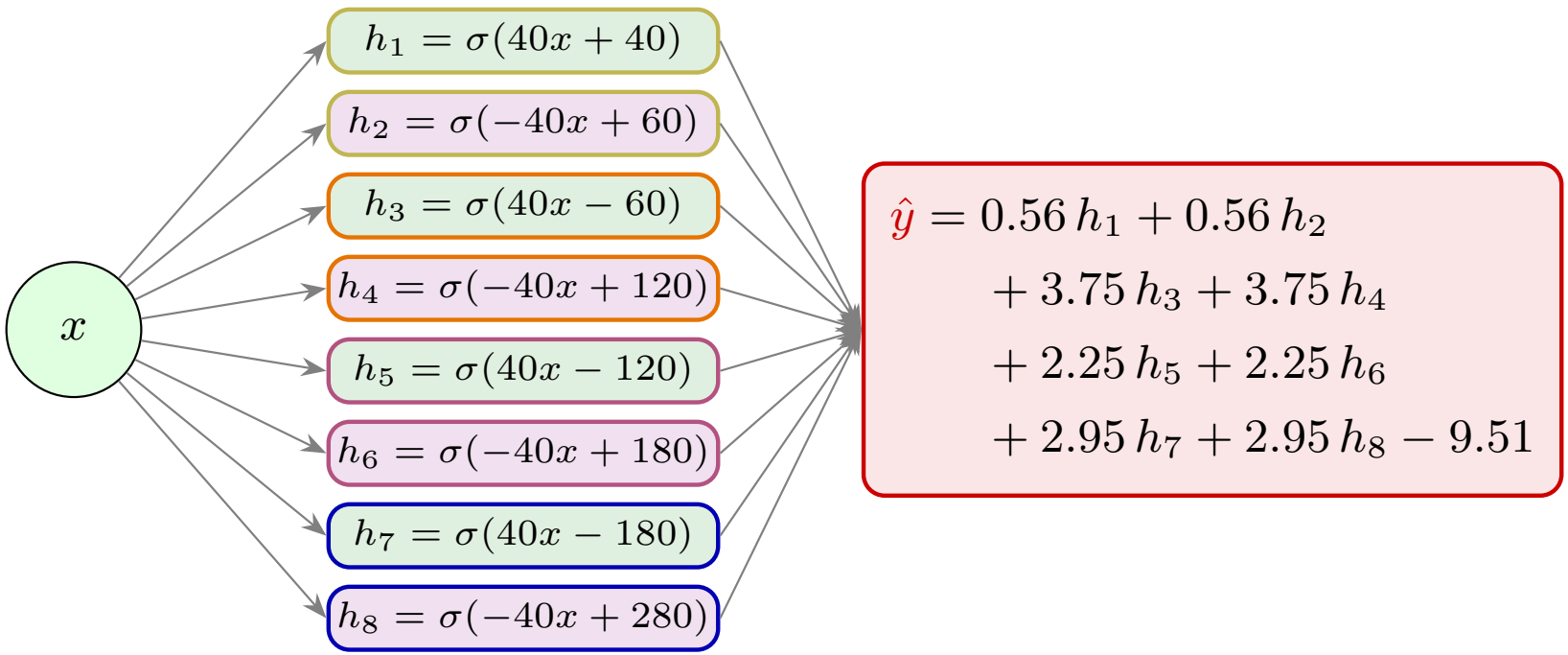
8 pairs (16 sigmoids)



20 pairs (40 sigmoids)

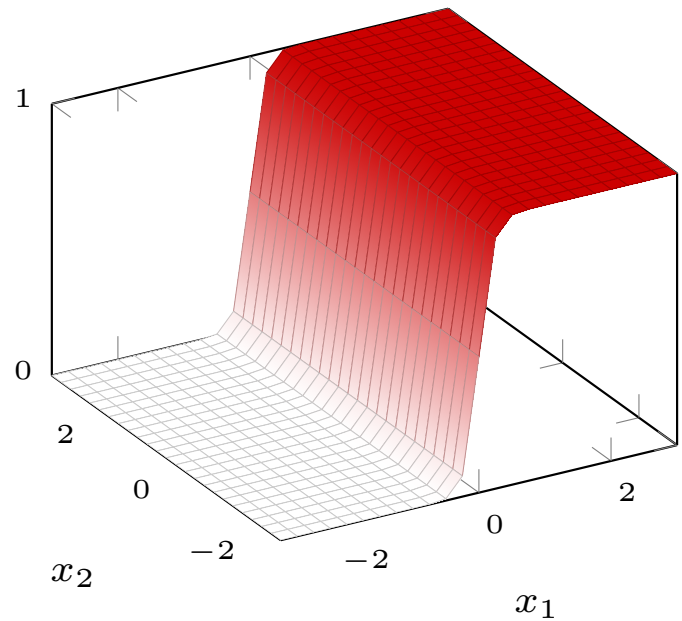


Neural network with 8 neurons in 1 layer

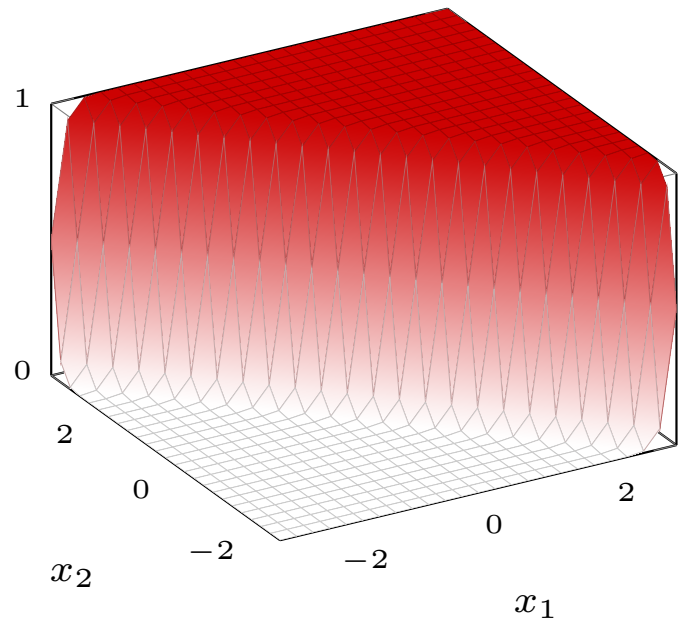


Two features : one sigmoid

$\sigma(10\,x_1)$



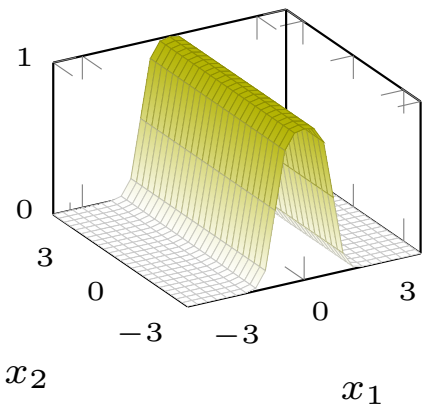
$\sigma(10\,x_1 + 10\,x_2)$



With 2 features, a sigmoid is a **step** in one direction, chosen by w_1 and w_2.

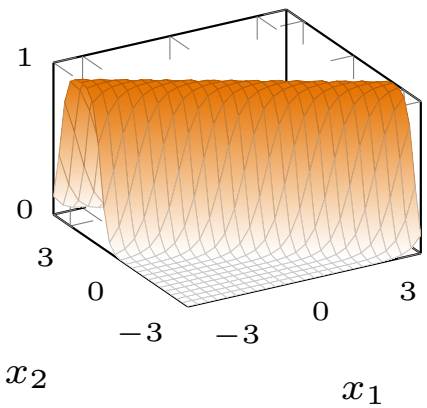
Two features : a sum of sigmoids

$p_1 = \sigma(6x_1 + 6)$
$+\sigma(-6x_1 + 6) - 1$



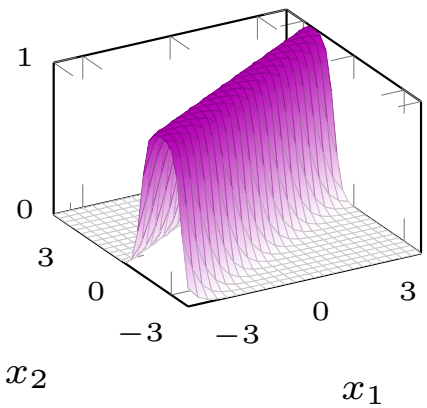
+

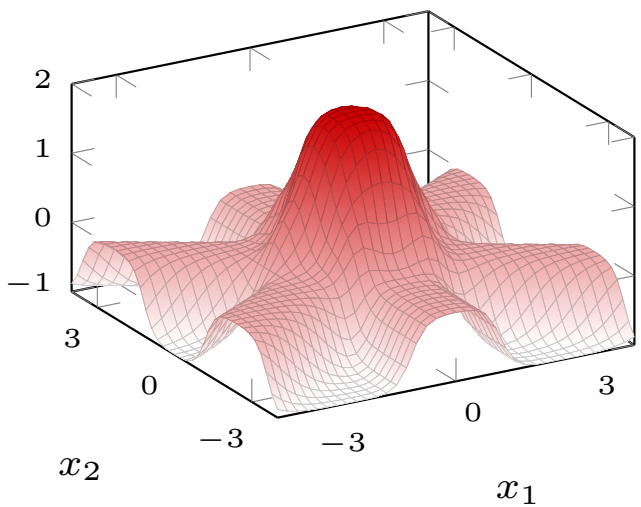
$p_2 = \sigma(3x_1 + 5x_2 + 6)$
$+\sigma(-3x_1 - 5x_2 + 6) - 1$



+

$p_3 = \sigma(-3x_1 + 5x_2 + 6)$
$+\sigma(3x_1 - 5x_2 + 6) - 1$

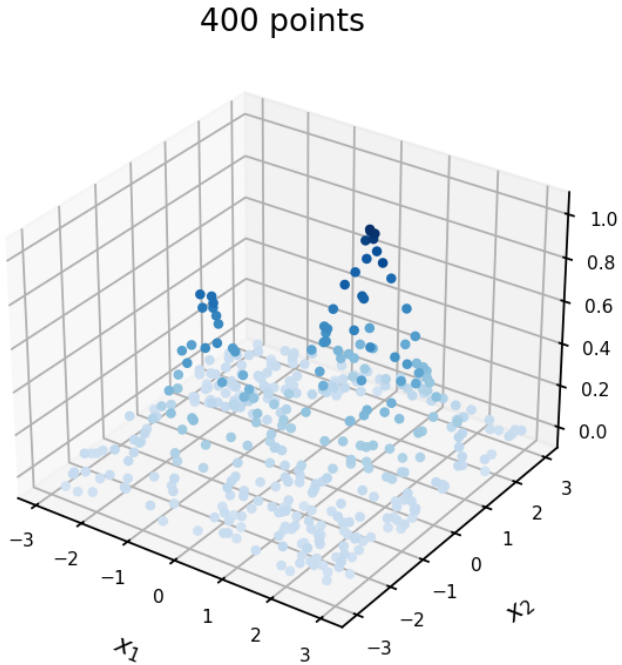


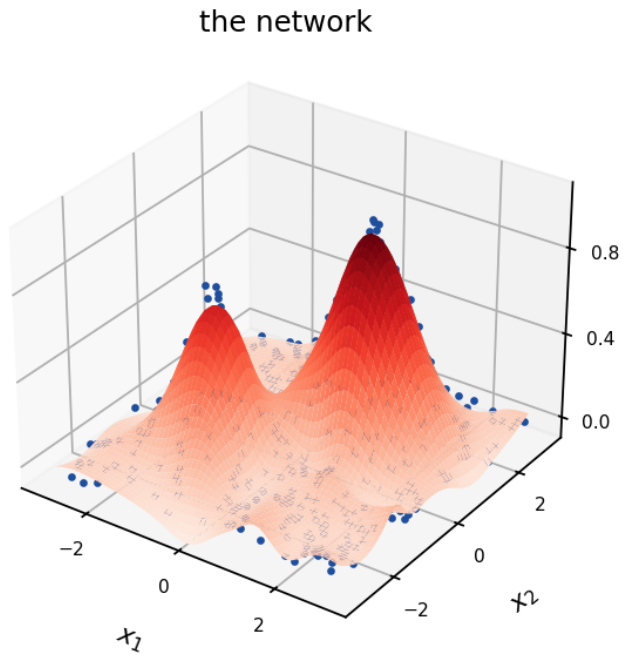


$$\hat{y} = p_1 + p_2 + p_3 - 1$$

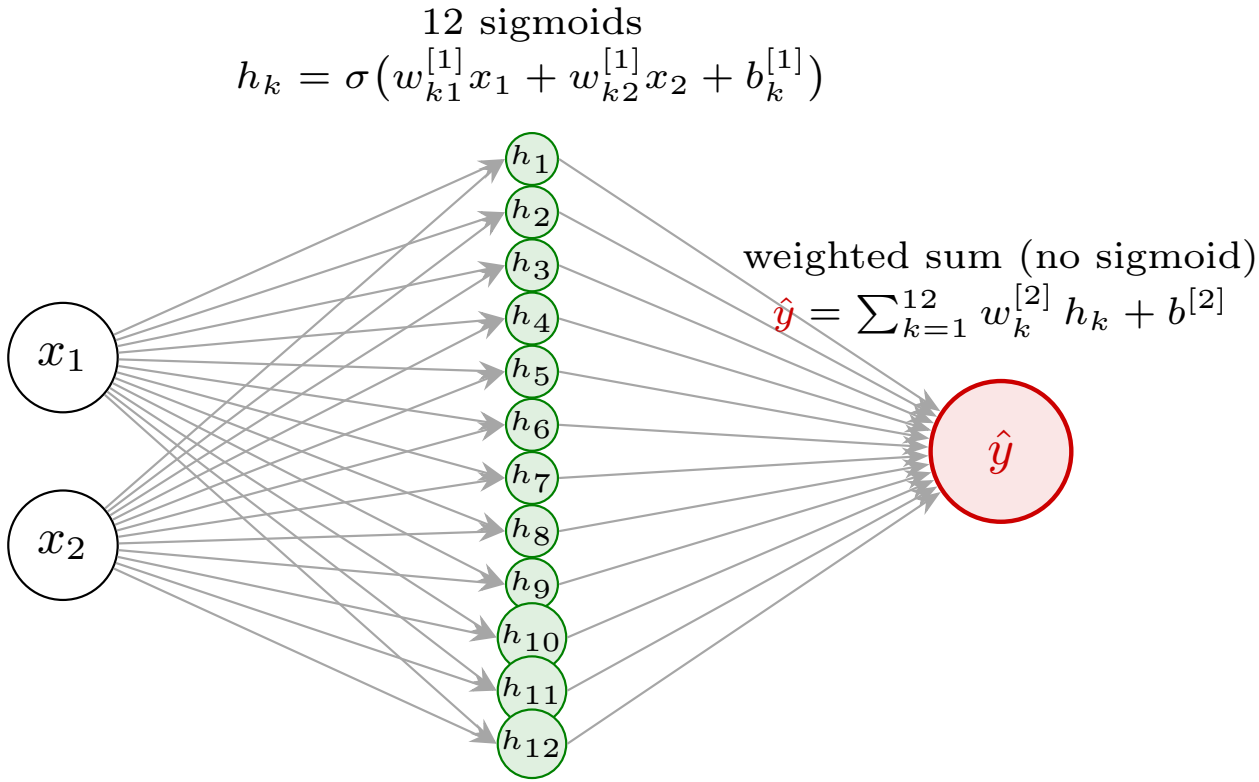
After M. Nielsen, *Neural Networks and Deep Learning*, ch. 4 [16]

Fitting points in 3D



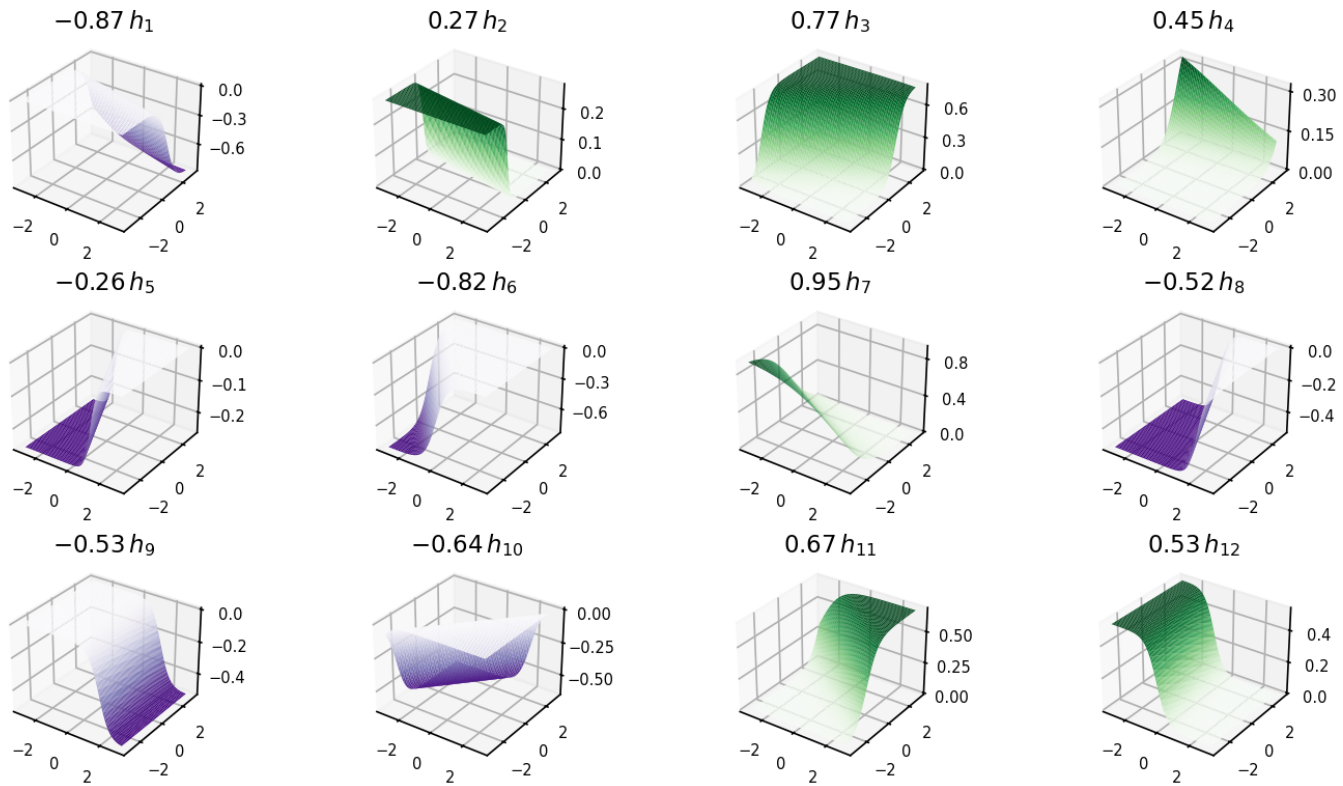


The network we trained



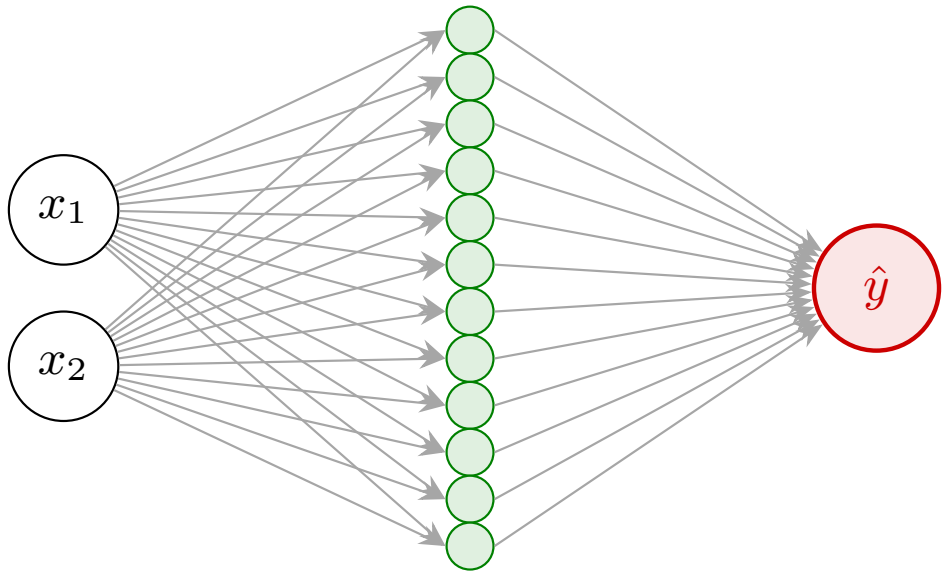
- Trained for **8 000 iterations**, each one on all 400 points.
- Each iteration changes the weights a little to reduce the **mean squared error**.

Inside the network



$\hat{y}$ = the sum of these **12 steps** -0.11.

A neural network is just a function

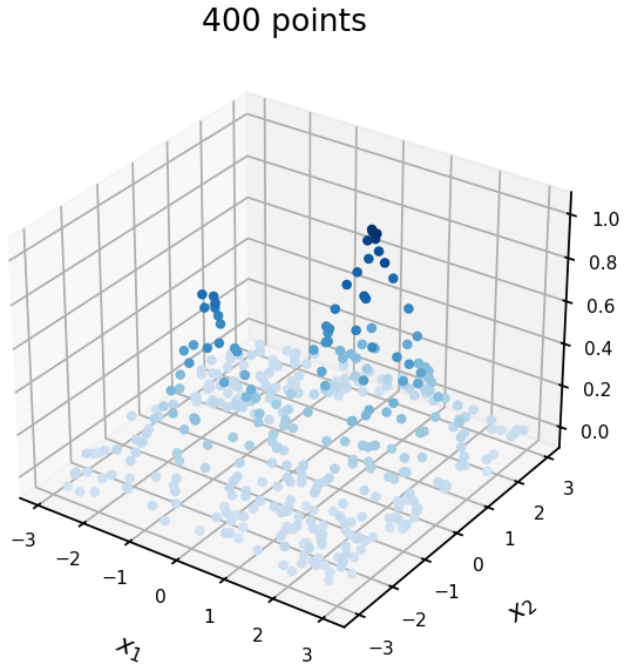


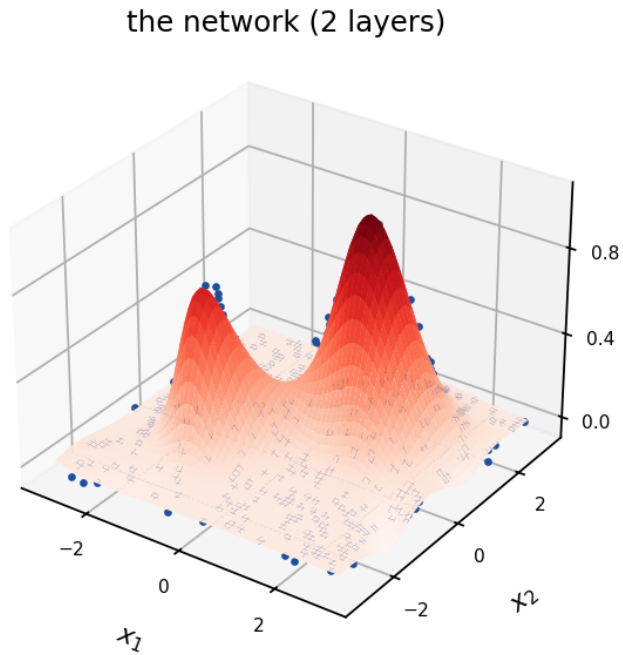
$$\underbrace{\hat{y}}_{1\times1} = \underbrace{W^{[2]}}_{1\times12}\,\sigma\Big(\underbrace{W^{[1]}}_{12\times2}\,\underbrace{x}_{2\times1} + \underbrace{b^{[1]}}_{12\times1}\Big) + \underbrace{b^{[2]}}_{1\times1}$$

What if we used a polynomial to fit the 400 points ?

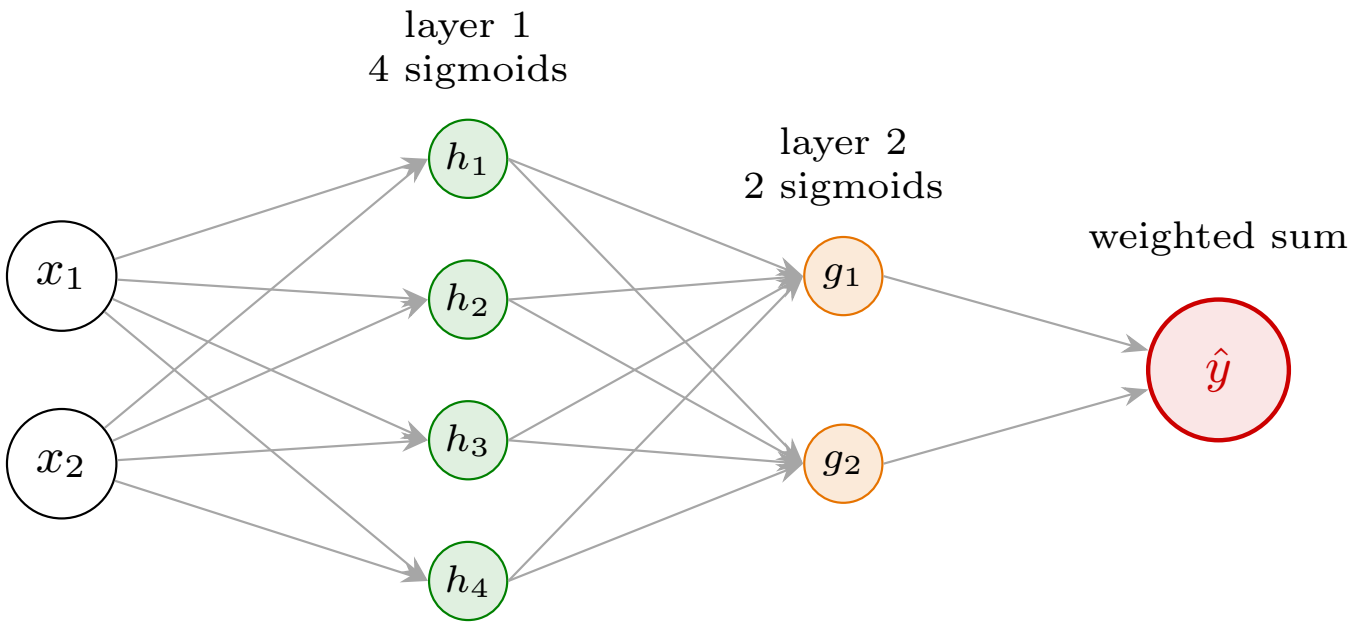
- A network with **1 layer** of 12 sigmoids : **49 weights**, error (RMSE) 0.037.
- Same error with a **polynomial** : degree 10, **66 features**.

Fitting the same points with 2 layers





The network with 2 layers



	neurons	weights	error (RMSE)
1 layer	6	25	0.076
1 layer	12	49	0.037
2 layers	**4 + 2**	**25**	**0.035**
Polynomial	degree 10	66	0.036

Chapter 5: Deep Learning

Motivation

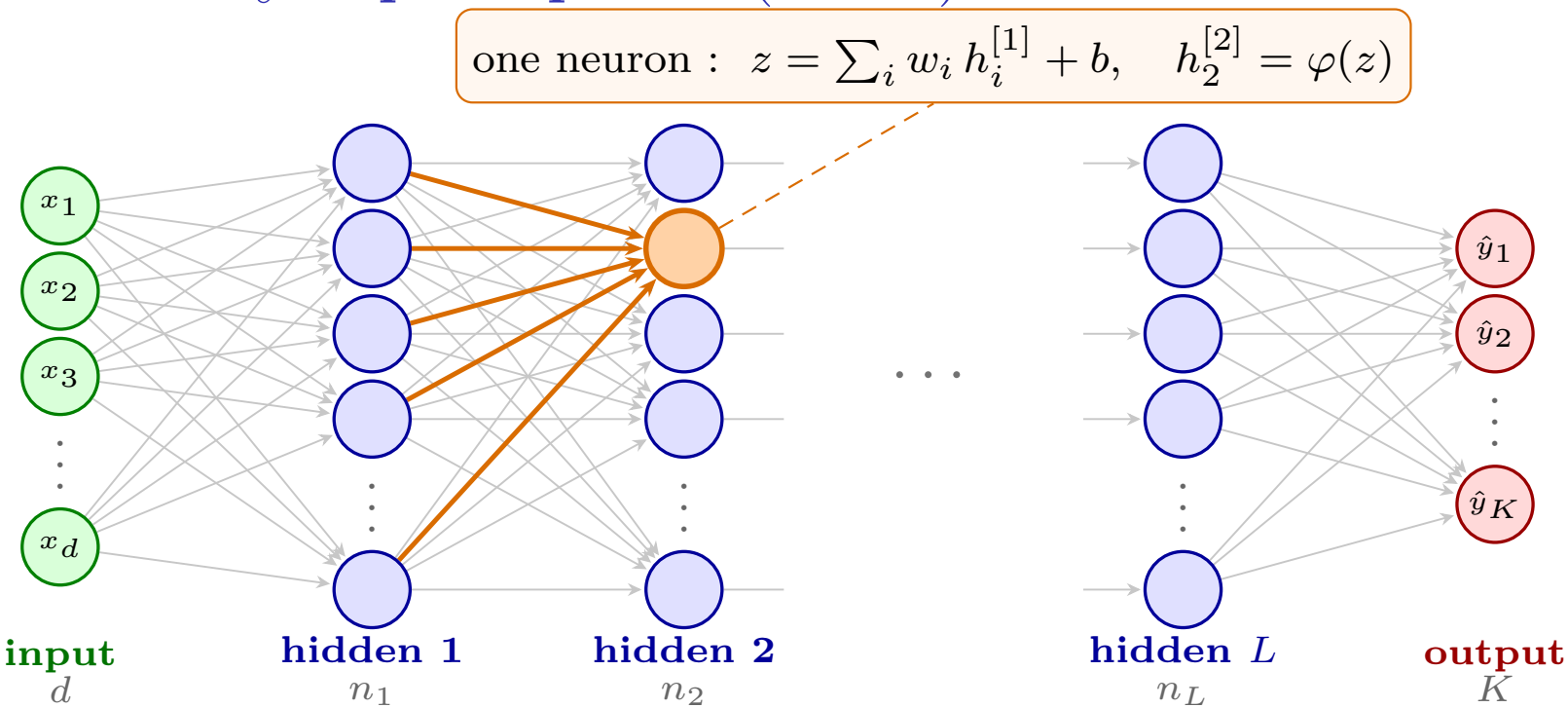
Neural networks

Training

Convolutional neural networks and computer vision

Summary

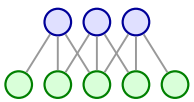
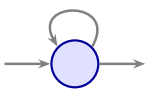
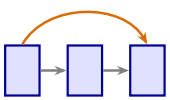
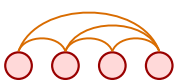
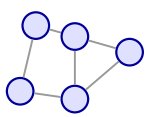
The multi-layer perceptron (MLP)



$$h^{[0]} = x, \quad h^{[\ell]} = \varphi\big(W^{[\ell]} h^{[\ell-1]} + b^{[\ell]}\big) \text{ for } \ell = 1, \ldots, L,$$
$$\hat{y} = g\big(W^{[L+1]} h^{[L]} + b^{[L+1]}\big)$$

- $W^{[\ell]}$ is $n_\ell \times n_{\ell-1}$.
- φ : the **activation function**, so far σ.
- g : identity (regression), sigmoid or softmax (classification).

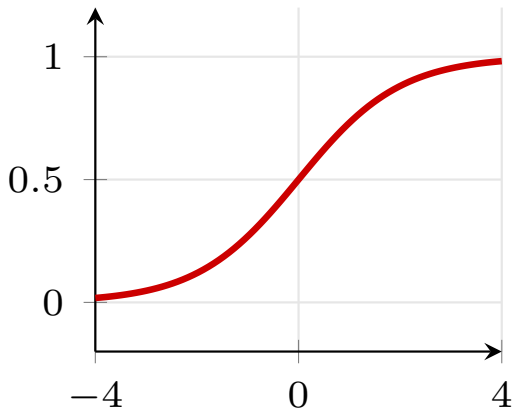
Other neural network architectures

	Architecture	What changes	Applications
	**CNN** convolutional neural network	Small **windows**, shared weights.	Images : faces, medical scans, self-driving cars.
	**RNN** recurrent neural network	A **loop** : the output is fed back in.	Sequences : speech, time series, text (before 2017).
	**ResNet** residual network	**Skip** connections : $h^{[\ell+1]} = h^{[\ell]} + f(h^{[\ell]})$.	Very deep networks (100+ layers) for images.
	**Transformer** attention	Each word **chooses** which words to look at.	Chatbots (ChatGPT, Claude), translation.
	**GNN** graph neural network	Connections follow a **graph**.	Molecules (new drugs), social networks, travel times on maps.

All of them are still built from neurons $\varphi(w^\top x + b)$ and trained the same way.

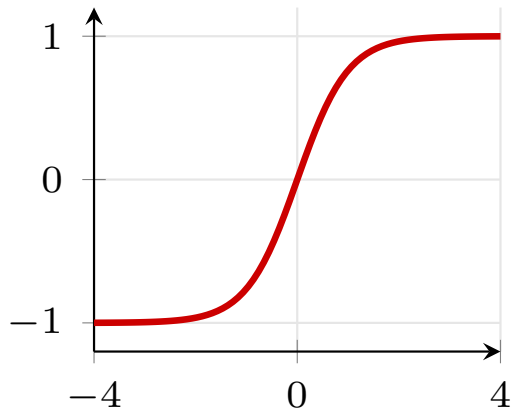
Activation functions

sigmoid



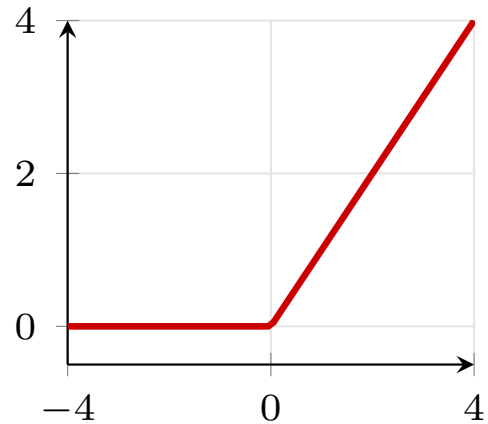
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

tanh



$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

ReLU



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

The choice depends on :

- **the output we need** : a probability → sigmoid.
- **the depth** : many layers → ReLU (not flat for $z > 0$).
- **loops** : a value repeated many times → tanh (stays in $(-1, 1)$).

Softmax, and many others

Softmax : K scores $\to$ K probabilities that sum to 1.

$$z = W^{[L+1]} h^{[L]} + b^{[L+1]} \qquad \text{softmax}(z)_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

Example with $K = 3$ scores $z = (2,\ 1,\ 0.1)$:

	class 1	class 2	class 3	sum
1. scores z_k	2	1	0.1	
2. exponentials e^{z_k}	7.39	2.72	1.11	11.21
3. divide by the sum	$\frac{7.39}{11.21} = \mathbf{0.66}$	$\frac{2.72}{11.21} = \mathbf{0.24}$	$\frac{1.11}{11.21} = \mathbf{0.10}$	1

Many others : Leaky ReLU, ELU, GELU, SiLU (Swish), softplus, ...

Why do we need the activation function ?

With φ :

$$\hat{y} = W^{[2]}\,\varphi\big(W^{[1]}x + b^{[1]}\big) + b^{[2]}$$

Without φ, the two layers collapse into one :

$$\hat{y} = W^{[2]}\big(W^{[1]}x + b^{[1]}\big) + b^{[2]} \;=\; \underbrace{W^{[2]}W^{[1]}}_{W}\,x + \underbrace{W^{[2]}b^{[1]} + b^{[2]}}_{b} \;=\; Wx + b$$

- A stack of linear layers is still **one linear model**, however deep.
- The non-linear φ between layers is what lets the network **bend**.

Universal approximation theorem

Theorem (Cybenko 1989, Hornik 1991)
A network with *one* hidden layer and enough neurons can approximate any continuous function, **on a bounded region**, as closely as we want.

- More pairs of sigmoids → closer to the curve.
- It says the network *exists*, not that training will find it, nor how many neurons are needed.
- In practice, **deep** (many layers) works better than **wide** (one huge layer) :
  on our 400 points, 2 layers reach the same error with **about half** the weights.

The output layer and the loss

Task	**Output layer**	**Loss**
Regression	identity	mean squared error $\frac{1}{n}\sum_{i=1}^{n}\left\|\hat{y}^{(i)}-y^{(i)}\right\|^2$
Binary classification	1 neuron, sigmoid	small when the true class gets a **high probability**
Multiclass (K classes)	K neurons, softmax	

In every case, during training we **minimize the loss** : we look for the weights with the **smallest loss** (error).

Chapter 5: Deep Learning

Motivation

Neural networks

Training

Convolutional neural networks and computer vision

Summary

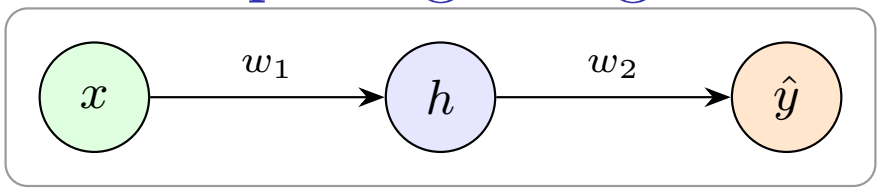
Training a network : gradient descent

$$\min_{\theta} \mathcal{L}(\theta), \qquad \theta = \left(W^{[1]}, b^{[1]}, \ldots, W^{[L+1]}, b^{[L+1]}\right)$$

0. **Initialization** : $b^{[\ell]} = 0$, and the weights $W^{[\ell]}$ drawn at random from a normal distribution.
1. **Forward pass** : compute $\hat{y}$ and the loss on a mini-batch.
2. **Backward pass** : compute $\nabla_\theta \mathcal{L}$ (backpropagation).
3. **Update** : $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$.
4. Repeat from step 1 until convergence.

- $\mathcal{L}$ is **not convex** : the local minimum found depends on the initialization.

Backpropagation : computing the gradient of the loss



the network f

The loss on one example (x, y) :

$$\begin{aligned}
\mathcal{L}(w_1, w_2) &= (\hat{y} - y)^2 \\
&= \big(f(x) - y\big)^2 && \text{since } \hat{y} = f(x) \\
&= \big(w_2\,\mathrm{ReLU}(w_1 x) - y\big)^2 && \text{since } f(x) = w_2\,\mathrm{ReLU}(w_1 x)
\end{aligned}$$

- x and y are **data** : fixed numbers. The **variables** are the weights w_1, w_2.
- Gradient descent needs $\partial\mathcal{L}/\partial w_1$ and $\partial\mathcal{L}/\partial w_2$: the **chain rule**, from the loss back to the input.

The chain rule

Two functions

$$x \xrightarrow{g} u \xrightarrow{f} y$$

$$(f \circ g)'(x) = f'\big(g(x)\big) \cdot g'(x)$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Three functions

$$x \xrightarrow{k} v \xrightarrow{g} u \xrightarrow{f} y$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dx}$$

The chain rule

Two functions

$$x \xrightarrow{g} u \xrightarrow{f} y$$

$$(f \circ g)'(x) = f'\big(g(x)\big) \cdot g'(x)$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Three functions

$$x \xrightarrow{k} v \xrightarrow{g} u \xrightarrow{f} y$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dx}$$

The loss

$$\mathcal{L}(w_1, w_2) = \big(w_2 \,\mathrm{ReLU}(w_1 x) - y\big)^2$$

$$w_1 \xrightarrow{\times x} z \xrightarrow{\mathrm{ReLU}} h \xrightarrow{\times w_2} \hat{y} \xrightarrow{(\hat{y}-y)^2} \mathcal{L}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h} \cdot \frac{\partial h}{\partial z} \cdot \frac{\partial z}{\partial w_1}$$

$$= 2(\hat{y} - y) \cdot w_2 \cdot \mathrm{ReLU}'(z) \cdot x$$

$$w_2 \xrightarrow{\times\, \mathrm{ReLU}()} \hat{y} \xrightarrow{(\hat{y}-y)^2} \mathcal{L}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2} = 2(\hat{y} - y) \cdot \mathrm{ReLU}(w_1 x)$$

Backpropagation in an MLP (for the fearless)

Forward, for $\ell = 1, \dots, L+1$ (with $h^{[0]} = x$, $\hat{y} = z^{[L+1]}$, loss $\mathcal{L} = \|\hat{y} - y\|^2$) :

$$z_j^{[\ell]} = \sum_k W_{jk}^{[\ell]}\, h_k^{[\ell-1]} + b_j^{[\ell]}, \qquad h_j^{[\ell]} = \varphi\big(z_j^{[\ell]}\big) \ (\ell \le L)$$

Backward, for $\ell = L+1, \dots, 1$: $\delta_j^{[\ell]} = \dfrac{\partial \mathcal{L}}{\partial z_j^{[\ell]}}$

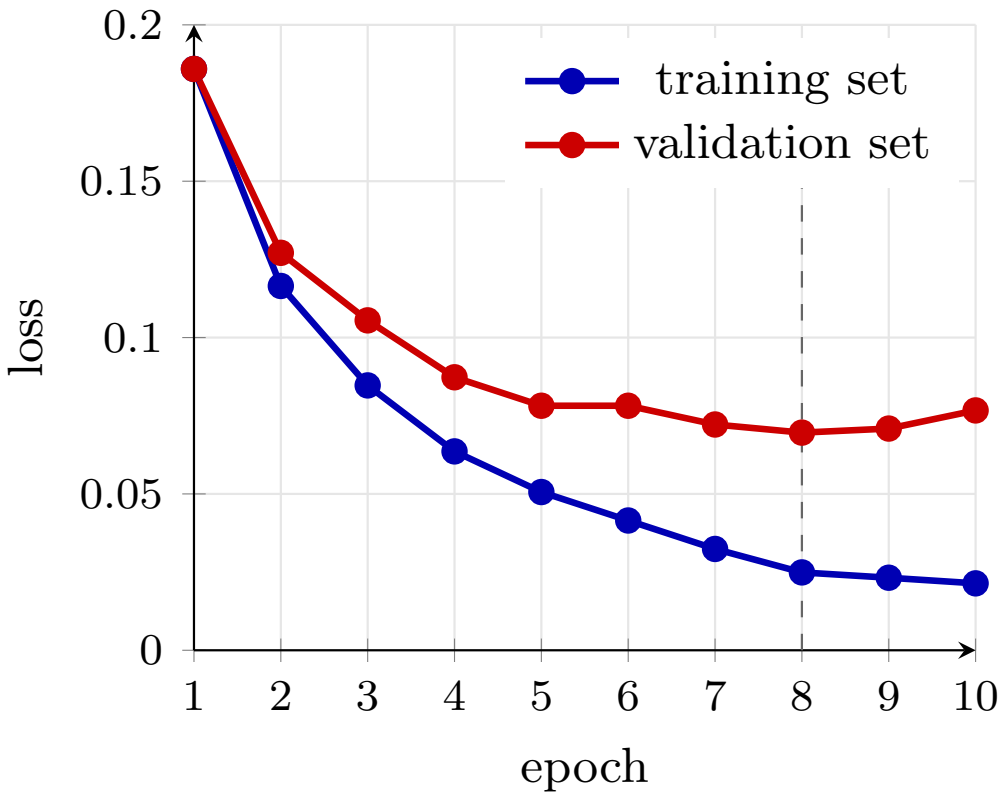
$$\delta_j^{[L+1]} = 2\big(\hat{y}_j - y_j\big), \qquad \delta_j^{[\ell]} = \varphi'\big(z_j^{[\ell]}\big) \sum_i W_{ij}^{[\ell+1]}\, \delta_i^{[\ell+1]}$$

Update of layer ℓ :

$$W_{jk}^{[\ell]} \leftarrow W_{jk}^{[\ell]} - \alpha \underbrace{\delta_j^{[\ell]}\, h_k^{[\ell-1]}}_{\partial \mathcal{L}/\partial W_{jk}^{[\ell]}} \qquad b_j^{[\ell]} \leftarrow b_j^{[\ell]} - \alpha \underbrace{\delta_j^{[\ell]}}_{\partial \mathcal{L}/\partial b_j^{[\ell]}}$$

In practice (PyTorch) : `loss.backward()` computes every gradient by backpropagation, then `optimizer.step()` updates every weight.

Overfitting in neural networks



Experiment : MLP 784 → 128 → 10 on MNIST.
Epoch : one pass over the whole training set.

Solutions to overfitting

- **Early stopping**
- **Penalty on the weights**
- **Dropout** : set some random neurons to 0 during training
- **Add more data**

Hyperparameters of a neural network

- **Architecture** : number of layers (depth), neurons per layer (width), activation functions.
- **Optimizer** : Plain SGD or Adam, learning rate α.
- **Batch size** : number of examples per gradient step.
- **Number of epochs**, or early stopping.
- **Regularization** : penalty on the weights, dropout rate.

Chosen on the **validation set**, often by trial and error.

Chapter 5: Deep Learning

Motivation

Neural networks

Training

Convolutional neural networks and computer vision

Summary

Why not an MLP on images ?

- A 28×28 MNIST image : 784 inputs. With 128 hidden neurons : 100 352 weights in the first layer.
- A 1000×1000 color photo : 3 million inputs. With 1000 hidden neurons : **3 billion** weights in the first layer.
- The MLP **flattens** the image : it does not know which pixels are neighbors.

The convolution

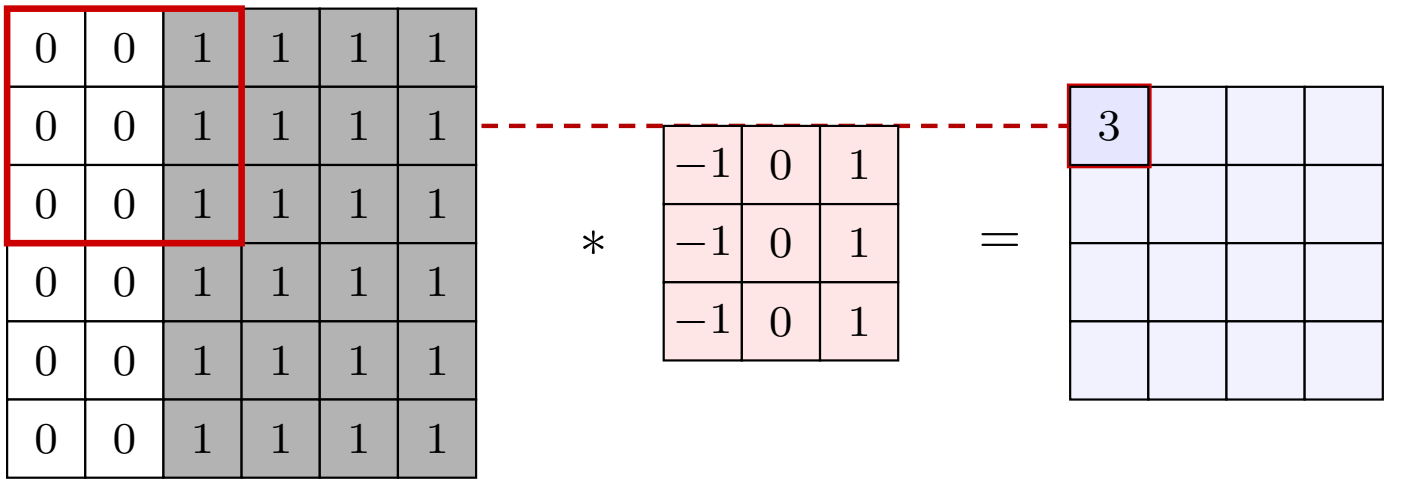
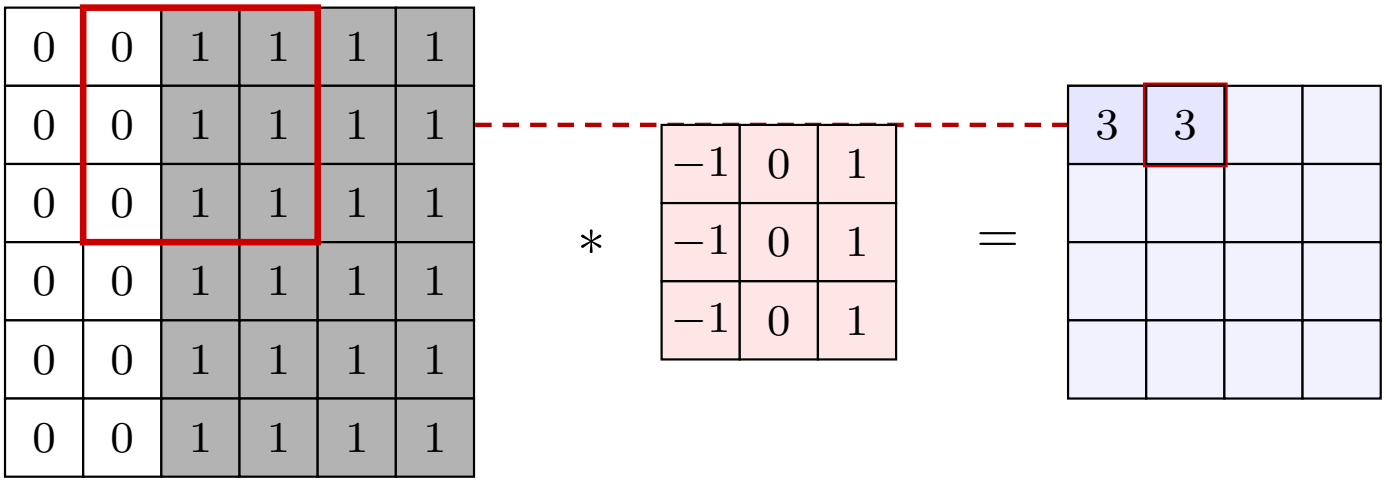


image 6×6 filter 3×3 feature map 4×4

- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution



- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution

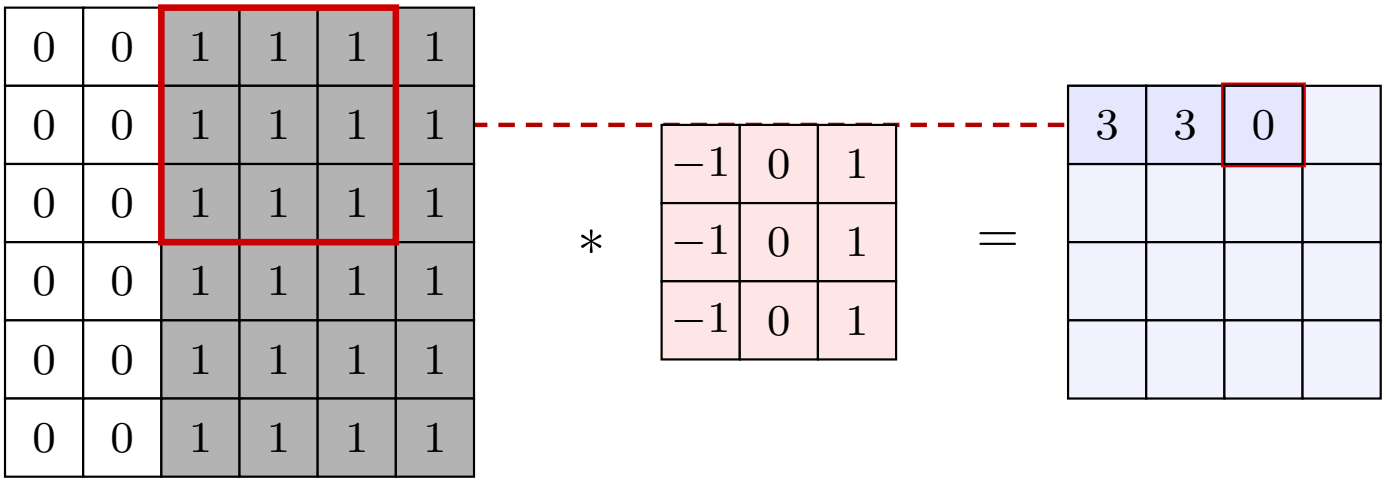


image 6×6 filter 3×3 feature map 4×4

- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution

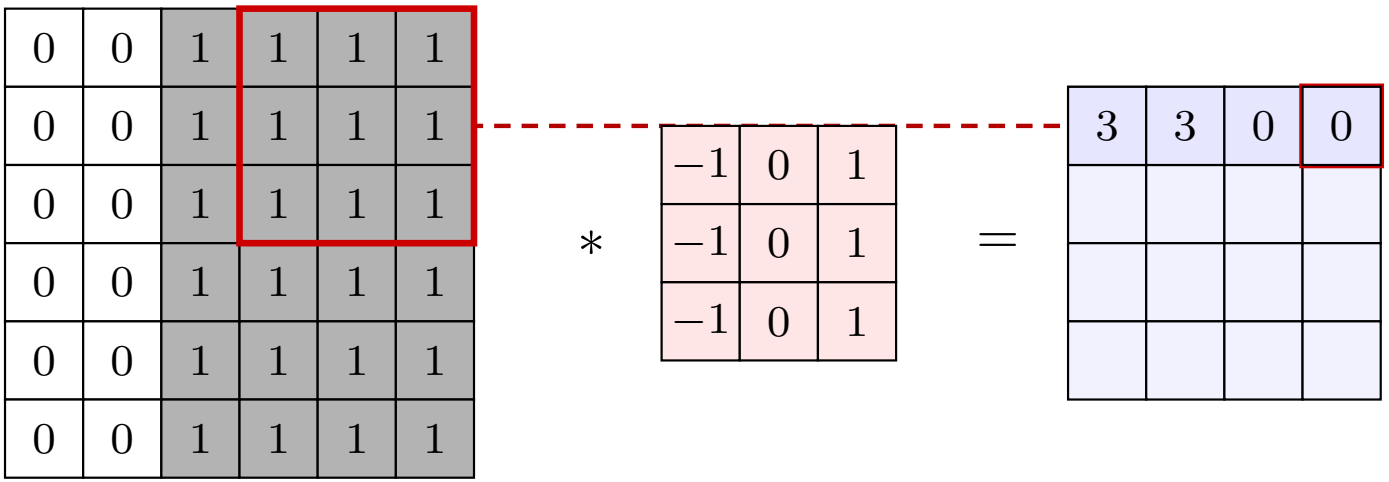
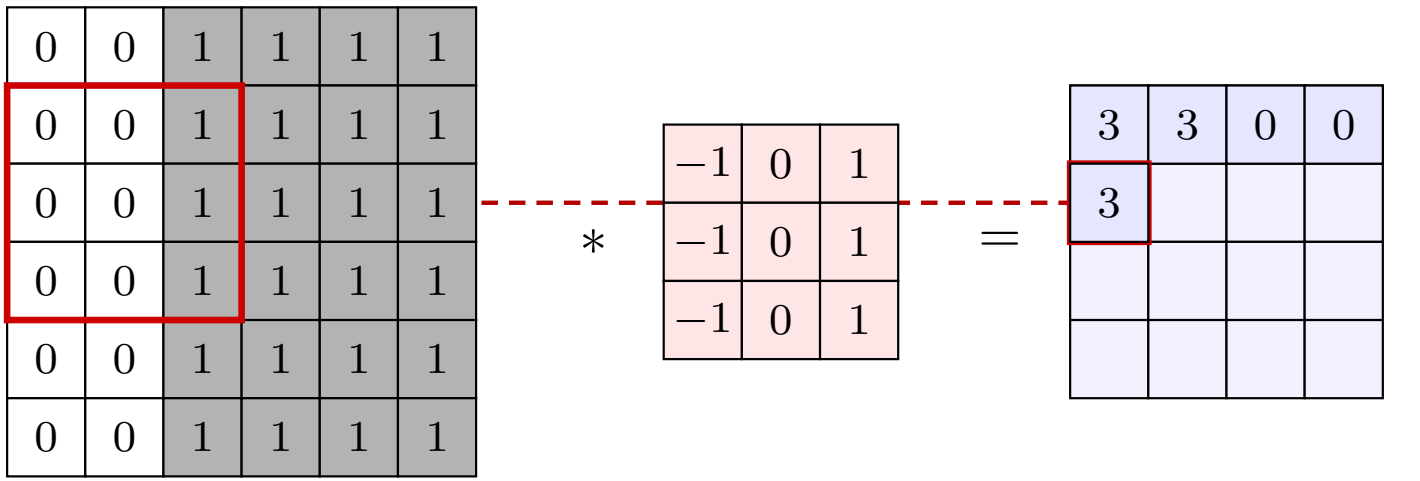


image 6×6 filter 3×3 feature map 4×4

- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

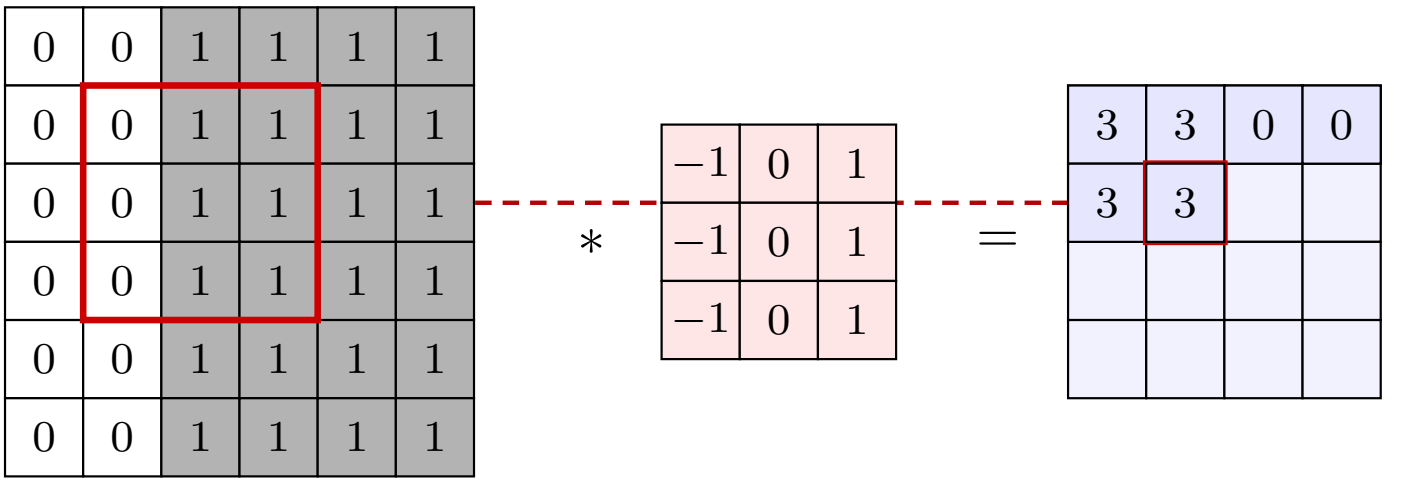
The convolution



- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

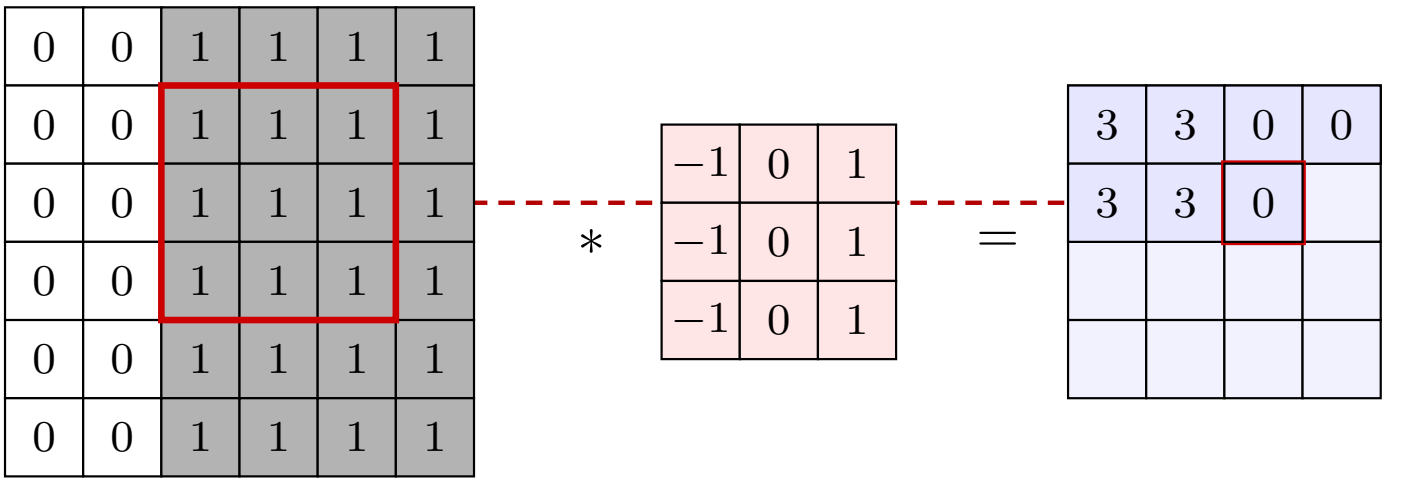
The convolution



- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution



- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution

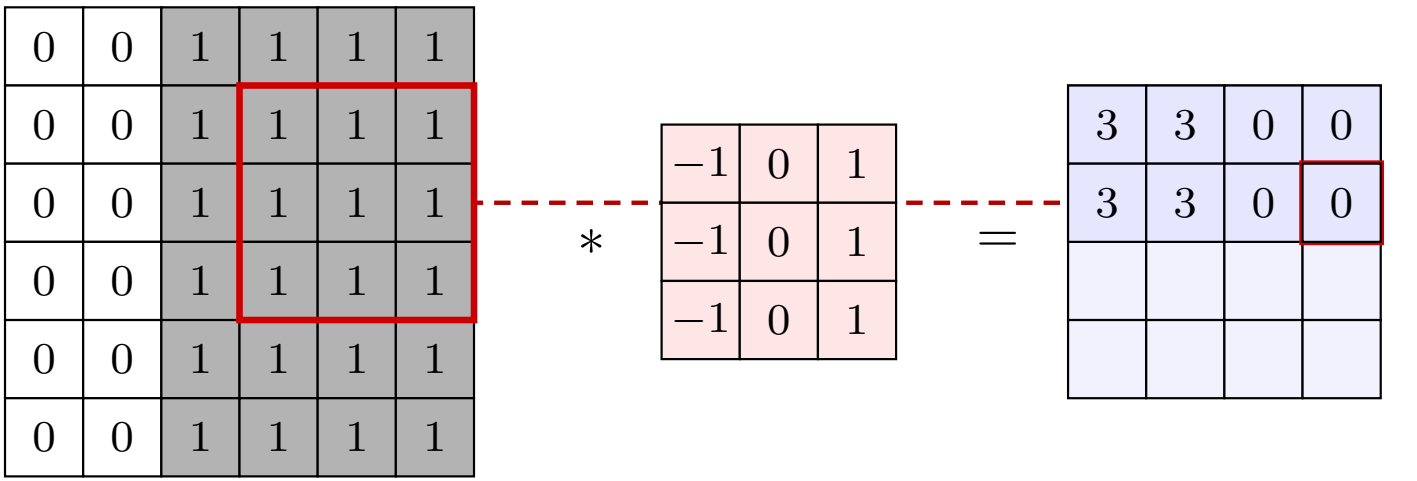


image 6×6 filter 3×3 feature map 4×4

- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution

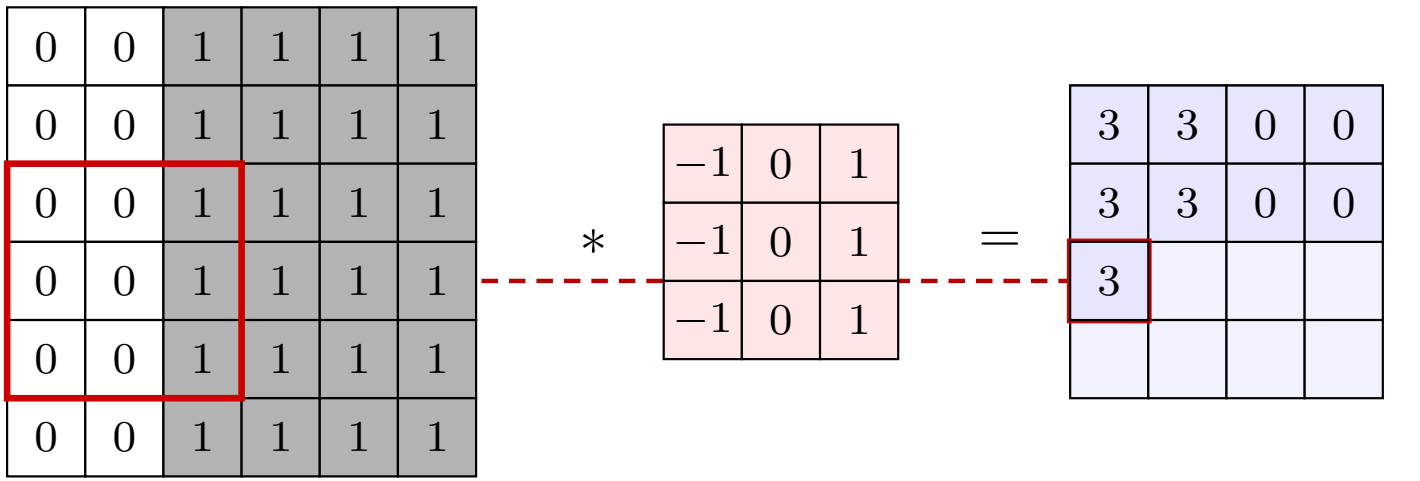
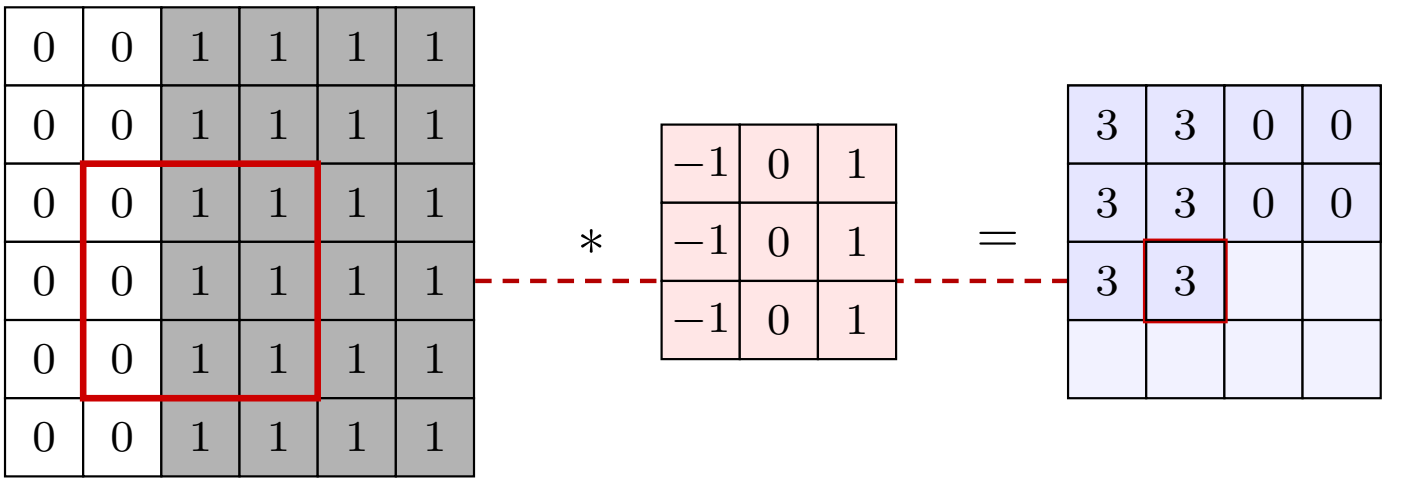


image 6×6 filter 3×3 feature map 4×4

- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

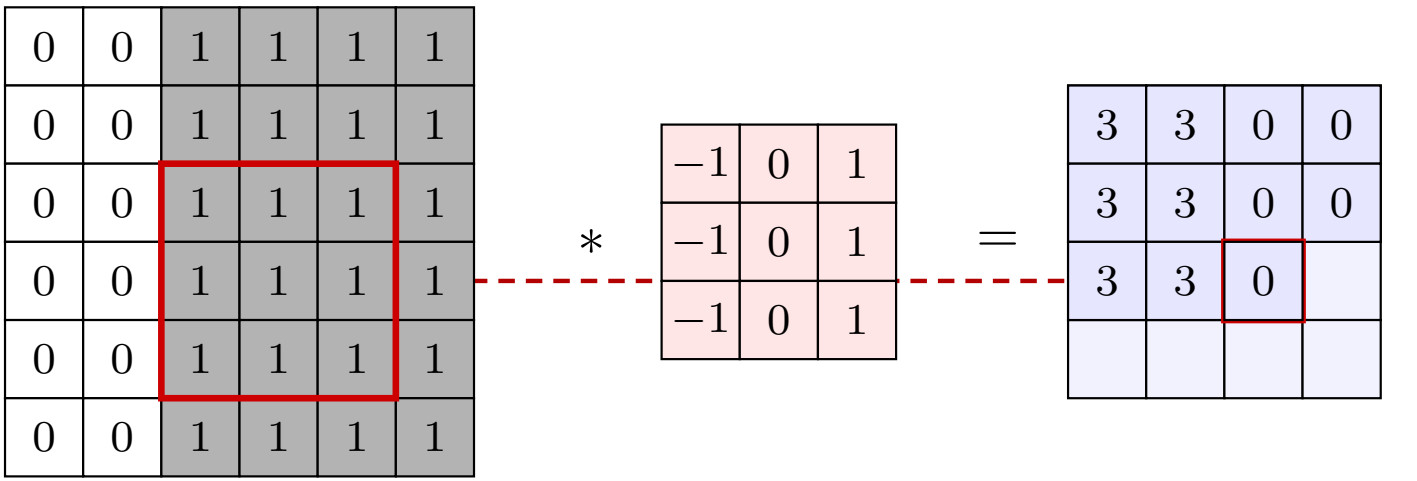
The convolution



- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution



- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution

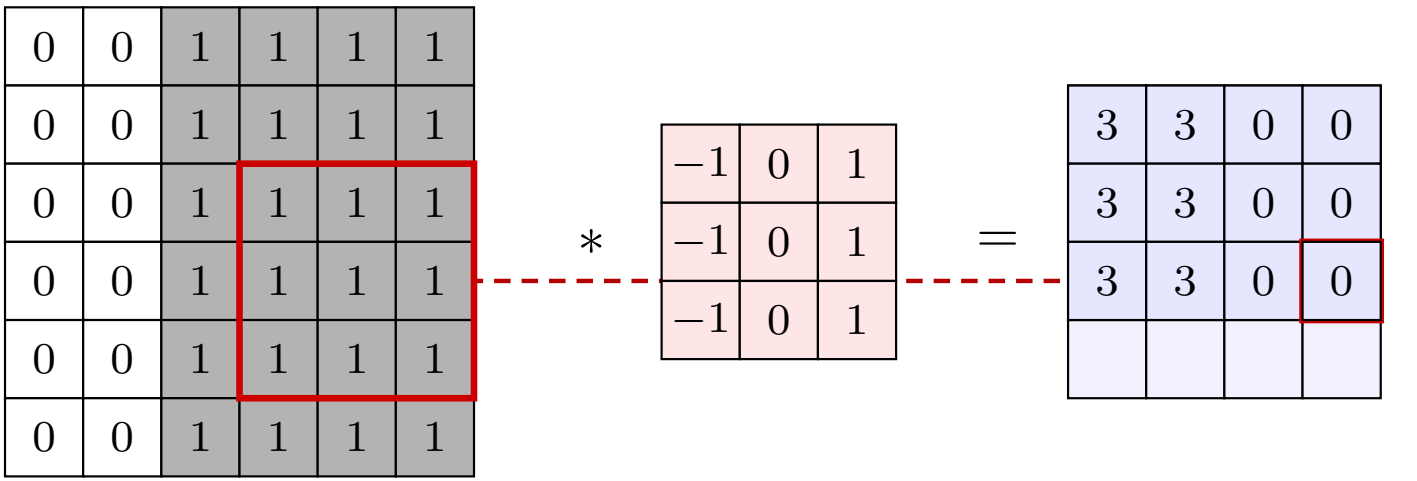


image 6×6 filter 3×3 feature map 4×4

- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution

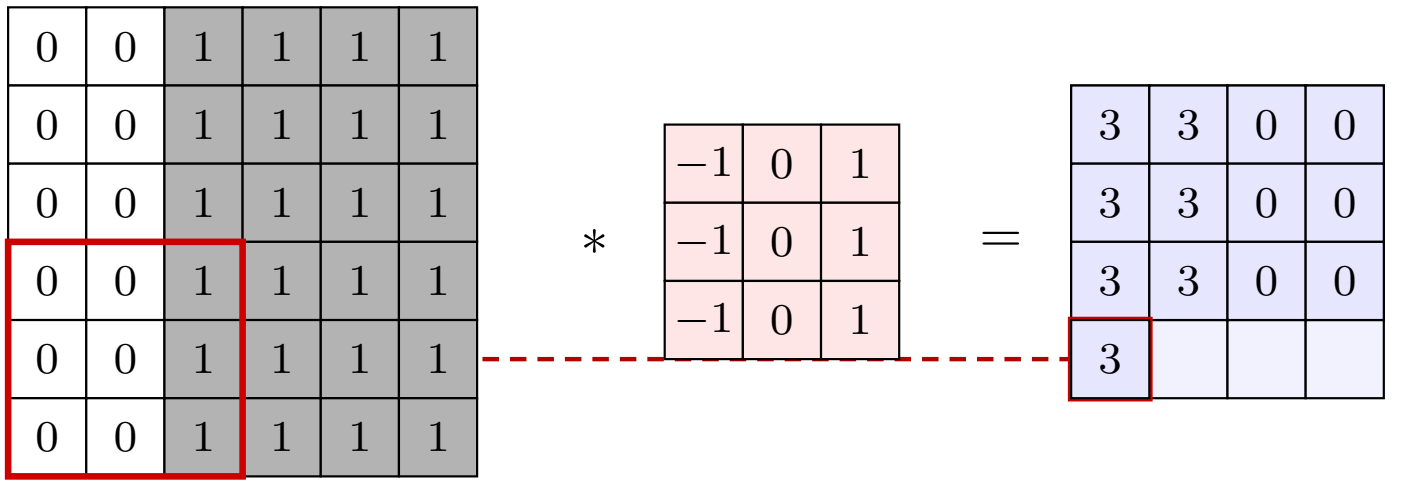
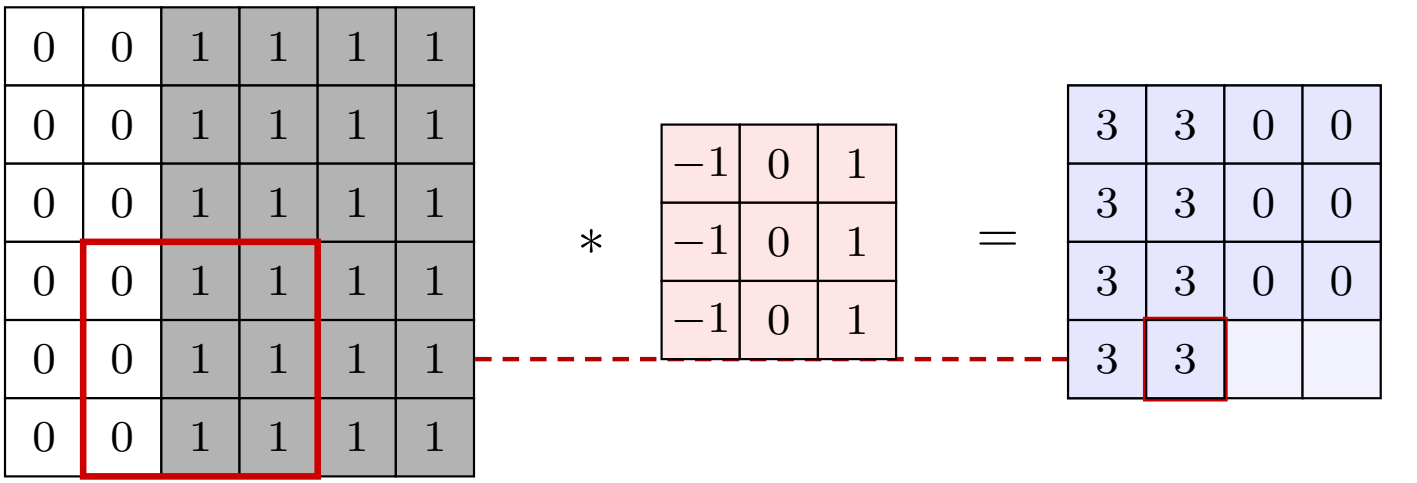


image 6×6 filter 3×3 feature map 4×4

- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

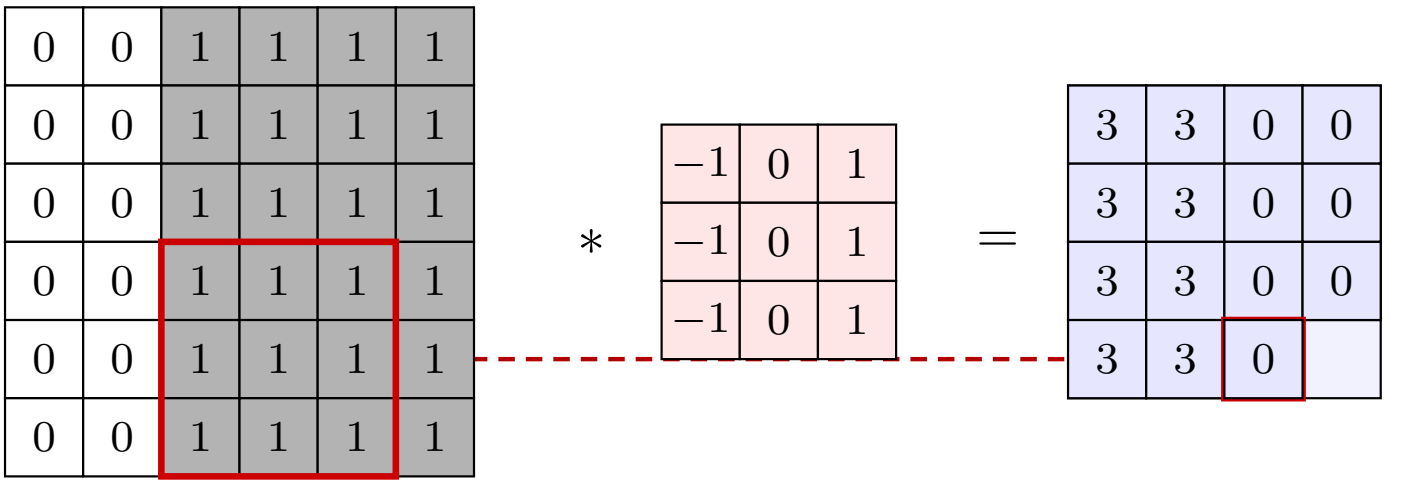
The convolution



- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution



- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

The convolution

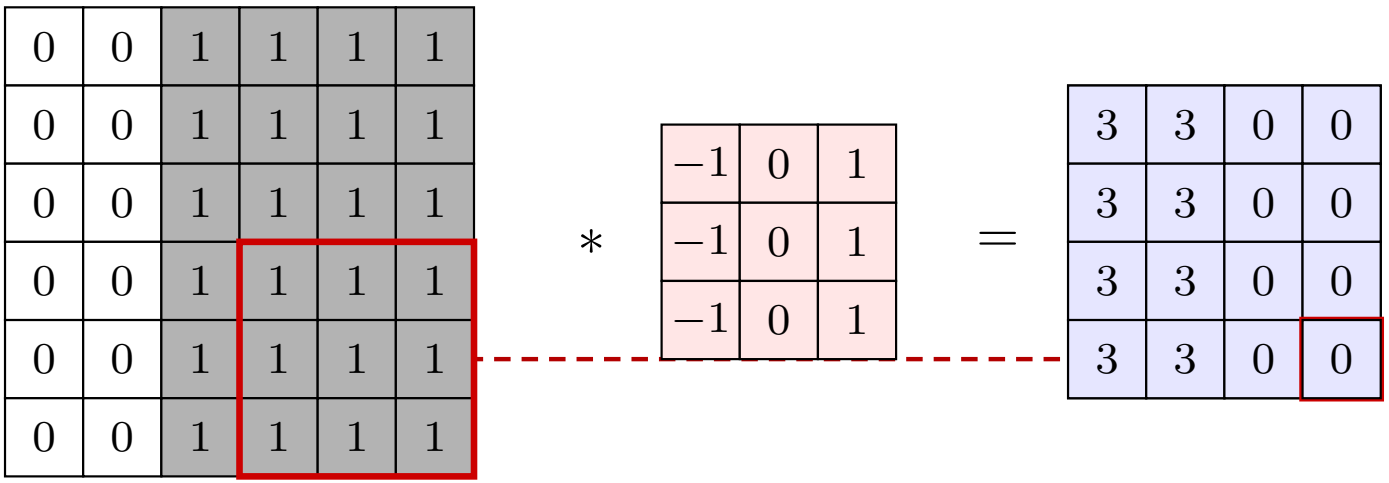
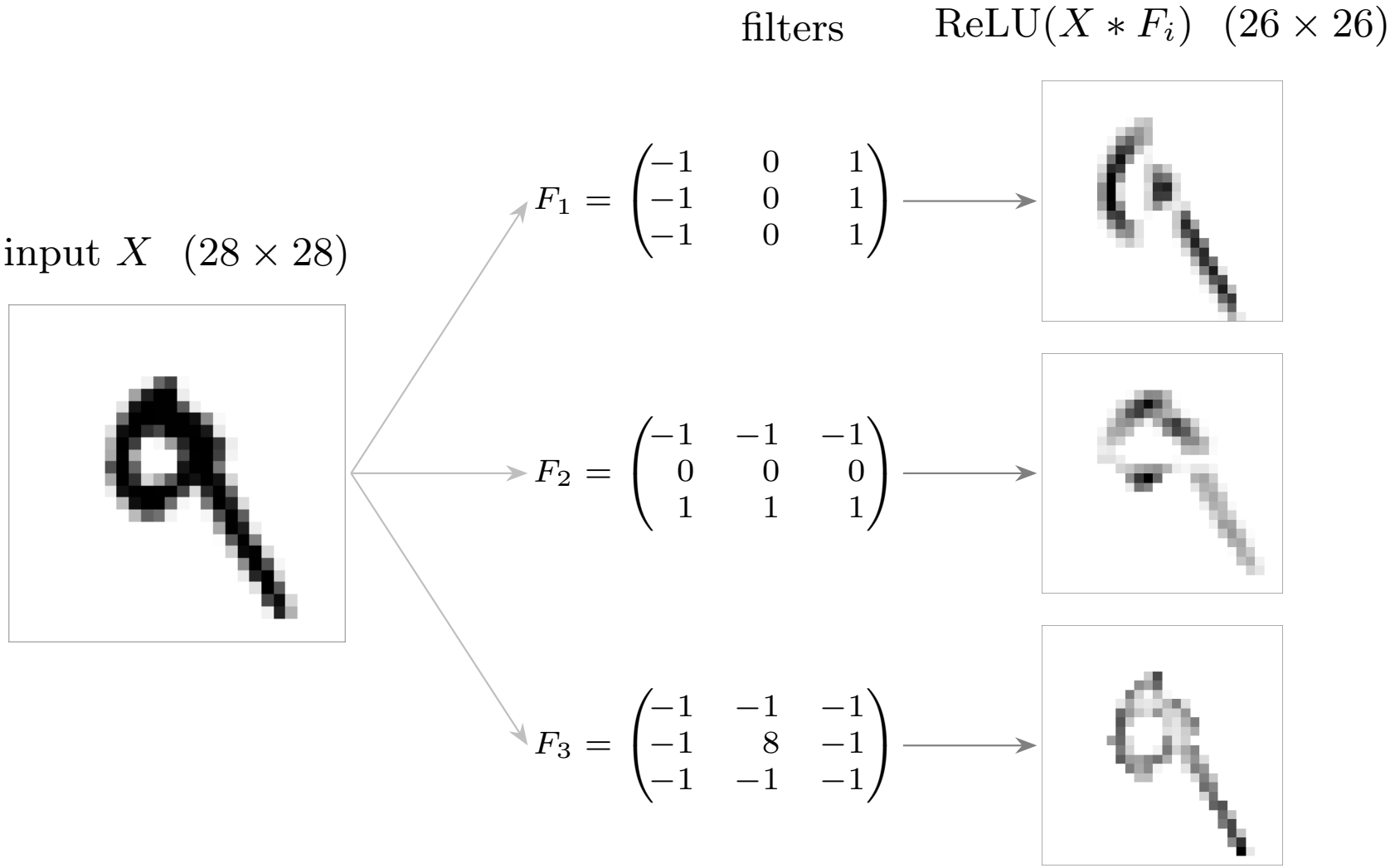


image 6×6 filter 3×3 feature map 4×4

- This filter asks "how much more ink on the right than on the left ?"
- The 9 numbers of the filter are **weights** that are **learned**.
- After computing the feature map, we apply an activation function (ReLU).

Example after A. Ng, *Deep Learning Specialization*, "Edge detection example" [15]

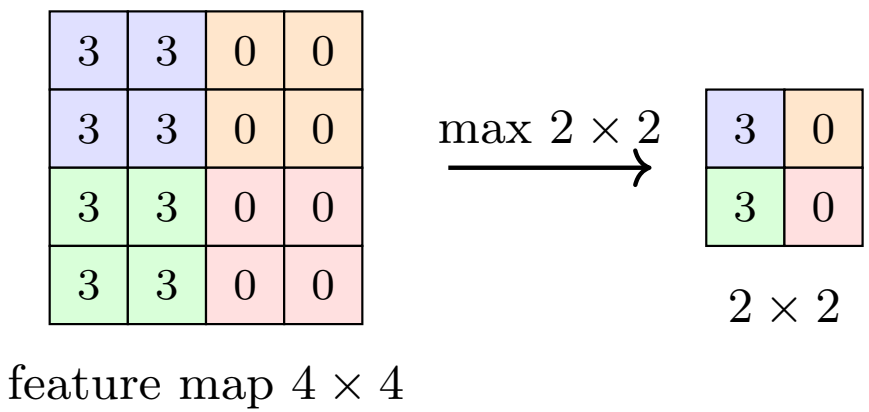
Filters on a real digit



Pooling

Pooling : slide a window over the feature map and **summarize** each window by one number (its max, its mean, ...). No weights to learn.

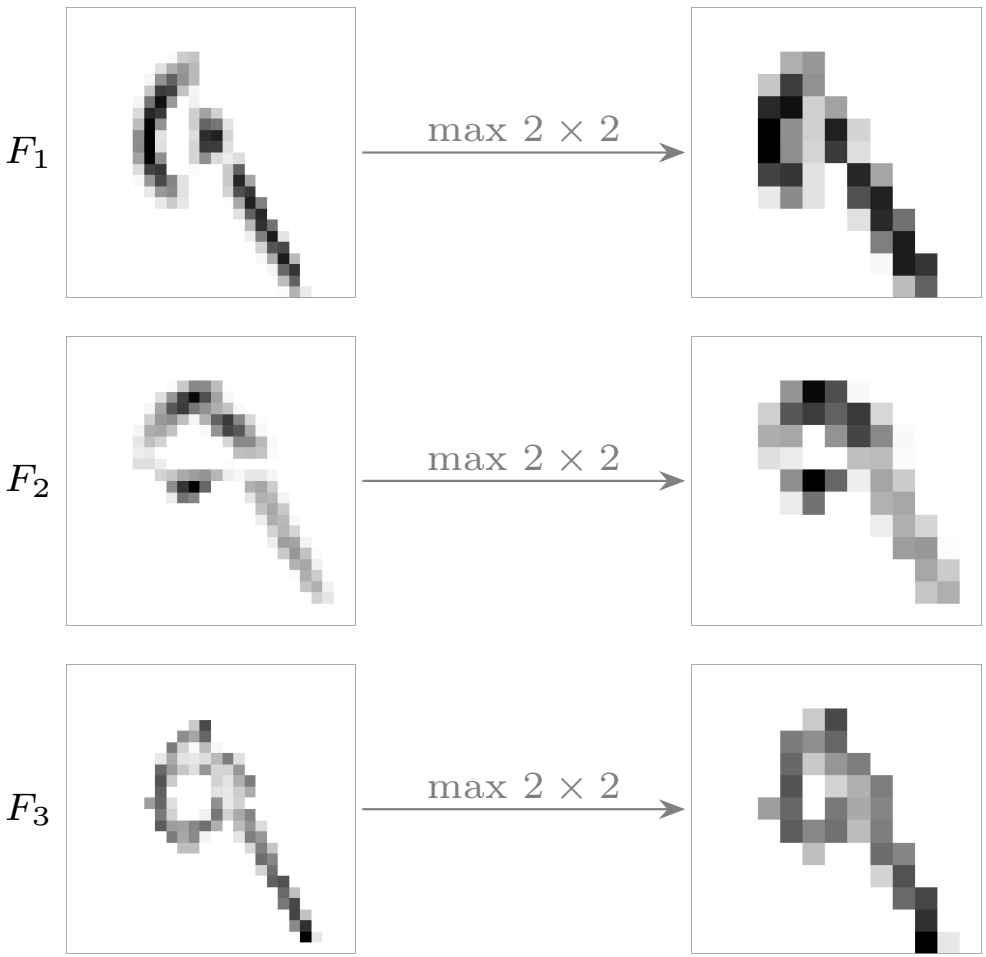
Example : max pooling, 2×2 windows that do not overlap.



feature map 4×4

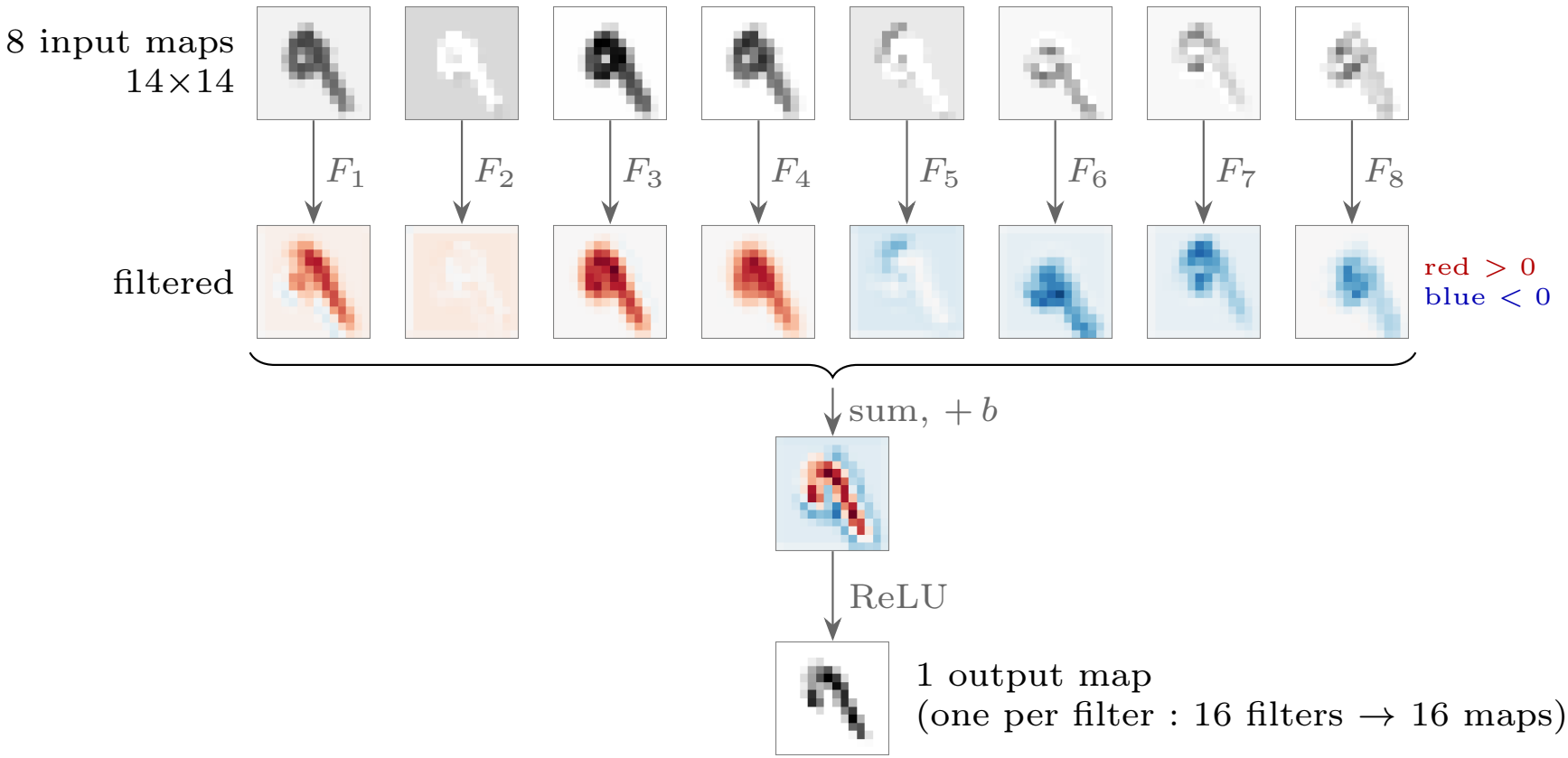
Pooling on the real digit

$\mathrm{ReLU}(X * F_i)$ (26×26) after pooling (13×13)

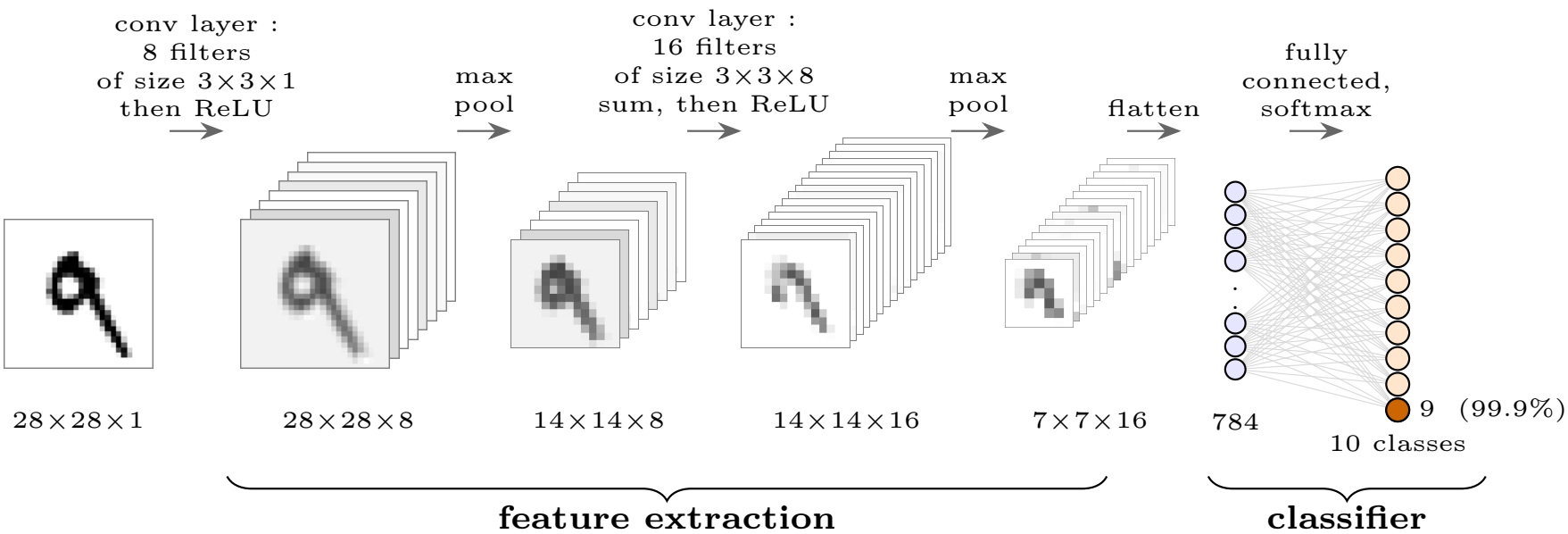


Second convolution layer : one filter sees all 8 maps

The input now has 8 maps, so one filter F has size $\mathbf{3 \times 3 \times 8}$.
It has 8 slices $F_1, \ldots, F_8$ of size 3×3 : slice F_i filters map i.



A convolutional neural network (CNN)



MNIST : three models

Trained on the 60 000 training digits, tested on 10 000 new ones.
Inference time : predicting the 10 000 new digits.

	Logistic regression	**MLP**	**CNN**
Hidden layers	0	1 (128 neurons)	2 conv + 2 pool
Parameters	7 850	101 770	9 098
Size (32-bit numbers)	31 KB	407 KB	36 KB
Training time, CPU	5 s	4 s	39 s
Training time, GPU	–	9 s	15 s
Inference time, CPU	13 ms	2 ms	126 ms
Inference time, GPU	–	2 ms	56 ms
Train accuracy	93.9%	99.4%	99.3%
Test accuracy	**92.6%**	**97.7%**	**98.9%**

Computer vision task : image classification



One label per image : **Egyptian cat** (34%), tabby (6%), ...

ResNet-50 trained on ImageNet. Image : scikit-image data (Chelsea the cat).

Computer vision task : object detection



A **box** and a label for every object.

Faster R-CNN trained on COCO. Image : COCO dataset.

Computer vision task : segmentation



One label **per pixel** : person, bicycle.

DeepLabV3 trained on COCO. Image : COCO dataset.

ImageNet and the deep learning revolution

- **ImageNet** : 1.2 million labeled training images, 1000 classes. A yearly competition from 2010 to 2017.
- **2012, AlexNet** : a CNN with 8 layers and 60 million parameters, trained on 2 GPUs. The true class is among its 5 best guesses for **84.7%** of the images, against 73.8% for the best competitor.
- Since then, every winner has been a neural network.

This progress required three conditions :

1. **Data** : large labeled datasets.
2. **Compute** : GPUs.
3. **Algorithms** : ReLU, dropout, better initialization.

Transfer learning

- Training a large CNN from scratch needs millions of images and many GPU hours.
- The early layers (edges, textures, shapes) are useful for almost any image.

In practice

1. A network **pretrained** on ImageNet is taken.
2. Its last layer is replaced by a new one for the new classes.
3. Only the new layer is trained (or all layers are **fine-tuned** with a small learning rate) on a few hundred images.

The same idea, at a much larger scale, gives the foundation models of Generative AI.

Chapter 5: Deep Learning

Motivation

Neural networks

Training

Convolutional neural networks and computer vision

Summary

Deep learning : guarantees, pros and cons

Use it for large datasets where the features are hard to design by hand :

- **Images** (classification, face recognition, image generation) : CNN, ResNet, Transformer, ...
- **Sound** : RNN, CNN, Transformer, ...
- **Text** : RNN, Transformer, ...

Pros

- Learns the features itself
- Any non-linear function
- Architecture can encode prior knowledge (CNN for images)
- Pretrained models can be reused

Cons

- Needs a lot of data and compute
- Many hyperparameters
- Non-convex : no guarantee to find the best weights
- A black box : hard to explain a prediction

Guaranteed to work if : nothing is guaranteed in general. The universal approximation theorem only says a good network exists.

Hyperparameters : depth, width, activation, optimizer, learning rate, batch size, epochs, regularization.