



Master DSAI

Advanced Deep Learning

2024-2025

From Artificial Intelligence to a first neuron model

Frederic Precioso

Disclaimer

If any content in this presentation is yours but is not correctly referenced or if it should be removed, please contact me and I will correct it.

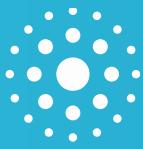


Overview

- **Context & Vocabulary**
- Math Basics
- Simple Models

Overview

- **Context & Vocabulary**
 - *What is Artificial Intelligence?*
 - *Machine Learning without Maths*
 - *Machine Learning & Statistics?*
- Math Basics
- Simple Models



CONTEXT & VOCABULARY



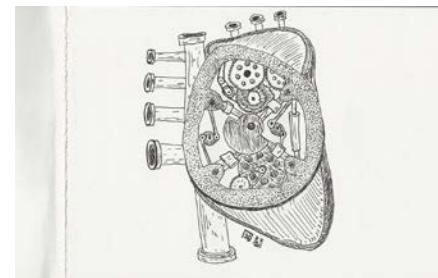
What is Artificial intelligence?



We want to believe it,
and not from “yesterday”

Artificial Intelligence

Artificial intelligence, as a field of research, is considered to have been created at the conference held on the campus of **Dartmouth College** in the summer of **1956**, even though this notion has been present since antiquity: Hephaestus built metal automatons to work for himself or protect others (Talos protecting the goddess Europa) and created Pandora, the first human woman in Greek mythology, from clay; the Golem, in Jewish folklore, is also made of clay, Yan Shi built humanoid automatons in the 10th century BC. during the Zhou Dynasty.



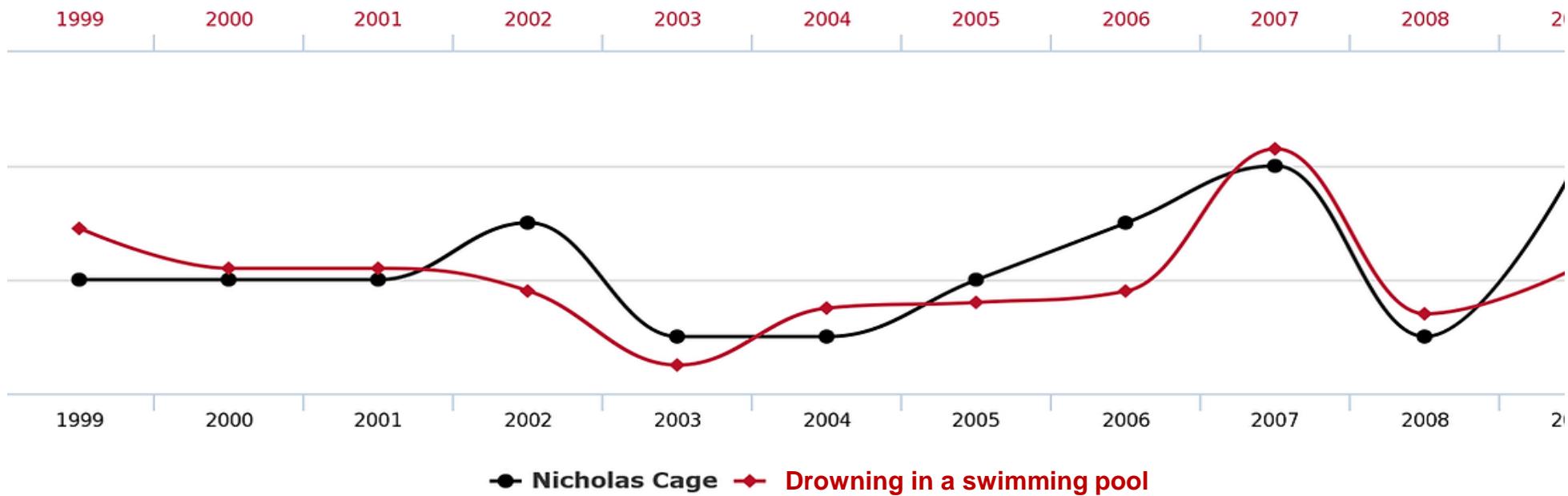
Maria dans
Metropolis 1927
De Fritz Lang

Sources :

- Wikipedia, <https://www.greeklegendsandmyths.com/automatons.html>
- http://www.alanturing.net/turing_archive/pages/Reference%20Articles/what_is_AI/What%20is%20AI02.html
- Stanford Encyclopedia of Philosophy: <https://plato.stanford.edu/entries/artificial-intelligence/>

We confuse too easily correlation and causality

Number of people who drowned by falling into a pool
correlates with
Films Nicolas Cage appeared in





How is Artificial intelligence defined?

- Closer to the Dartmouth conference but still before, the first manifesto on Artificial Intelligence, an unpublished report “***Intelligent Machinery***”, written by Alan Turing in **1948**. He already distinguished two different approaches to AI, which may be termed “***top-down***” and “***bottom-up***” (*now more commonly called knowledge-driven AI and data-driven AI respectively*).
- ***Go to read (at least from 20th Century), it is very informative on the different branches of AI:*** https://en.wikipedia.org/wiki/History_of_artificial_intelligence

(sources: Wikipedia, <https://www.greeklegendsandmyths.com/automatons.html>,

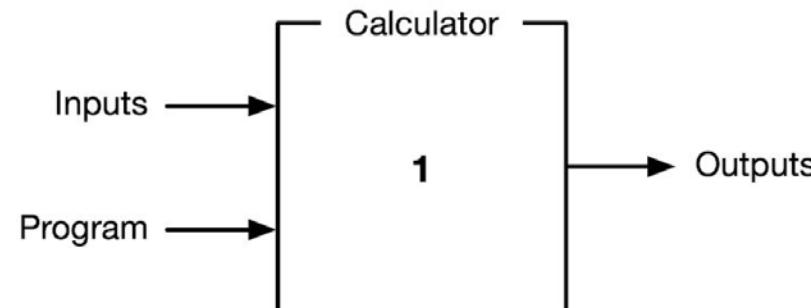
http://www.alanturing.net/turing_archive/pages/Reference%20Articles/what_is_AI/What%20is%20AI02.html

Stanford Encyclopedia of Philosophy: <https://plato.stanford.edu/entries/artificial-intelligence/>



How is Artificial intelligence defined?

- “*top-down*” or *knowledge-driven AI*
 - cognition = high-level phenomenon, independent of low-level details of implementation mechanism
 - Evolutionary Algorithms (1954, 1957, 1960), Knowledge Representation, Reasoning (1959, 1970), Expert Systems (1970), Logic, Automata, Intelligent Agent Systems (1990)...



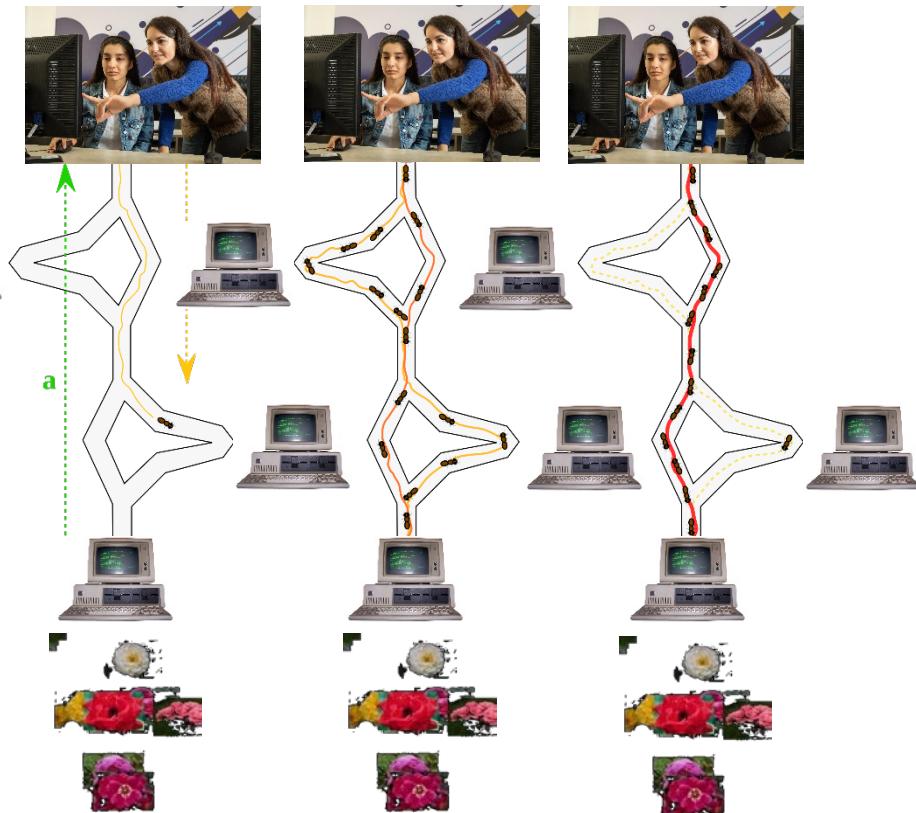
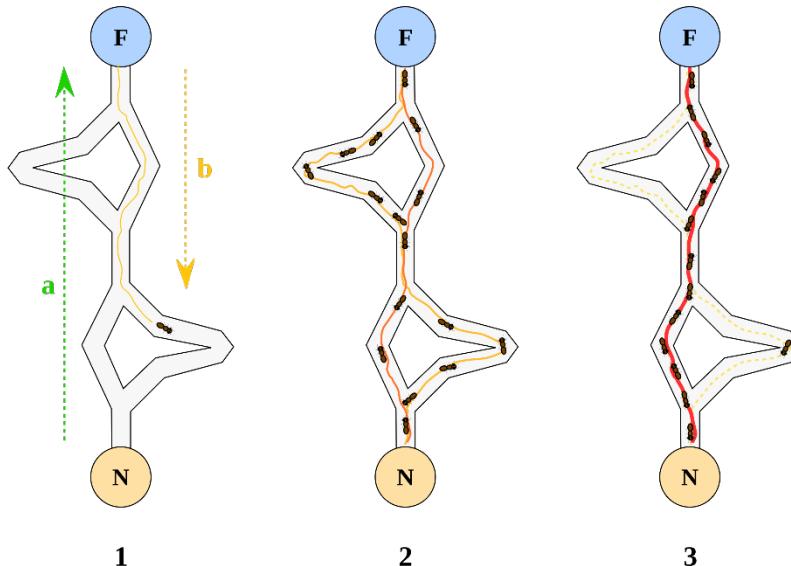
(1) Hypothetical-deductive machines

(Figure from: *Neurons spike back The invention of inductive machines and the artificial intelligence controversy*, D. Cardon, J.-P. Cointet, A. Mazières, Translated by Elizabeth Libbrecht In Réseaux Volume 211, Issue 5, 2018, pages 173 to 220)



Artificial Intelligence, Top-Down

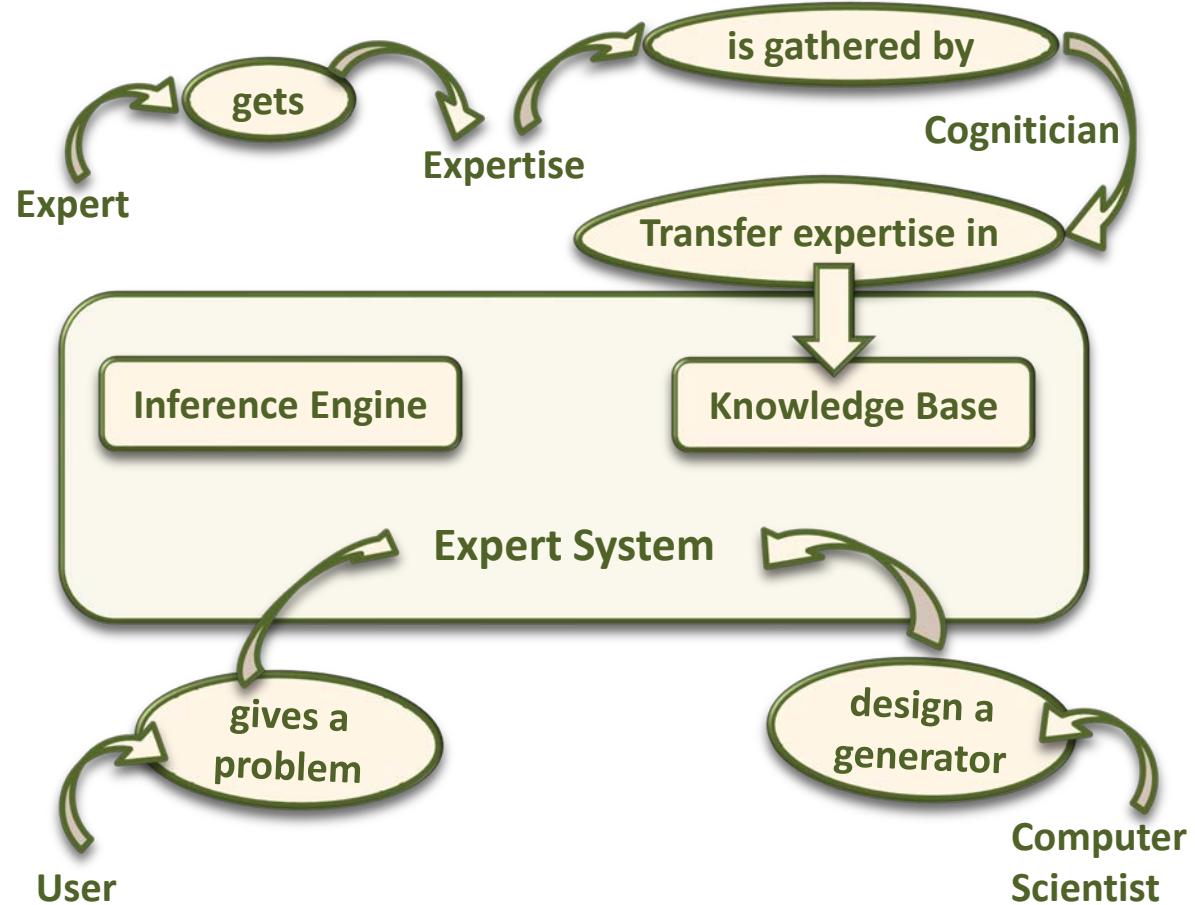
- Example of a Multi-Agent System: the Ant Colony Algorithm





Artificial Intelligence, Top-Down

- Example of an expert system:



Artificial Intelligence, Top-Down

- Expert system:

The screenshot shows a news article from RBR Insider. The header features the RBR logo (an orange 'rb' icon) and navigation links for AI, Manufacturing, Supply Chain, Robo Dev, Healthcare, CRO, Events, and All Topics. On the right, there are 'Login | Join RBR Insider' and a search icon. The main title of the article is 'A Cyclist's Encounter with an Indecisive Google Self-Driving Car'. Below the title is a subtitle: 'A bicyclist recently had a two-minute standoff with a Google self-driving car at a four-way stop in Austin, Texas. So what happened? We explain.' At the bottom left of the article area, it says '© AUGUST 26, 2015' and 'STEVE CROWE'.

The screenshot shows a news article from STAT. The title is 'STAT: IBM's Watson gave 'unsafe and incorrect' cancer treatment advice'. A red 'BRIEF' button is visible above the title. Below the title, the author is listed as 'Meg Bryant' and the date as 'PUBLISHED July 26, 2018'. The 'SHARE IT' section is partially visible at the bottom. The main content starts with a bold heading 'Dive Brief:' followed by a bulleted list of findings from a STAT review of internal IBM documents regarding Watson's cancer treatment advice.

AUTHOR
Meg Bryant

PUBLISHED
July 26, 2018

SHARE IT

Dive Brief:

- A STAT review of internal IBM documents suggests the company's Watson supercomputer wrongly advised doctors on how to treat patients' cancers.
- The documents — slides presented by then-IBM Watson Health deputy chief health officer Andrew Norden in June and July of last year — include "multiple examples of unsafe and incorrect treatment

AI Timeline

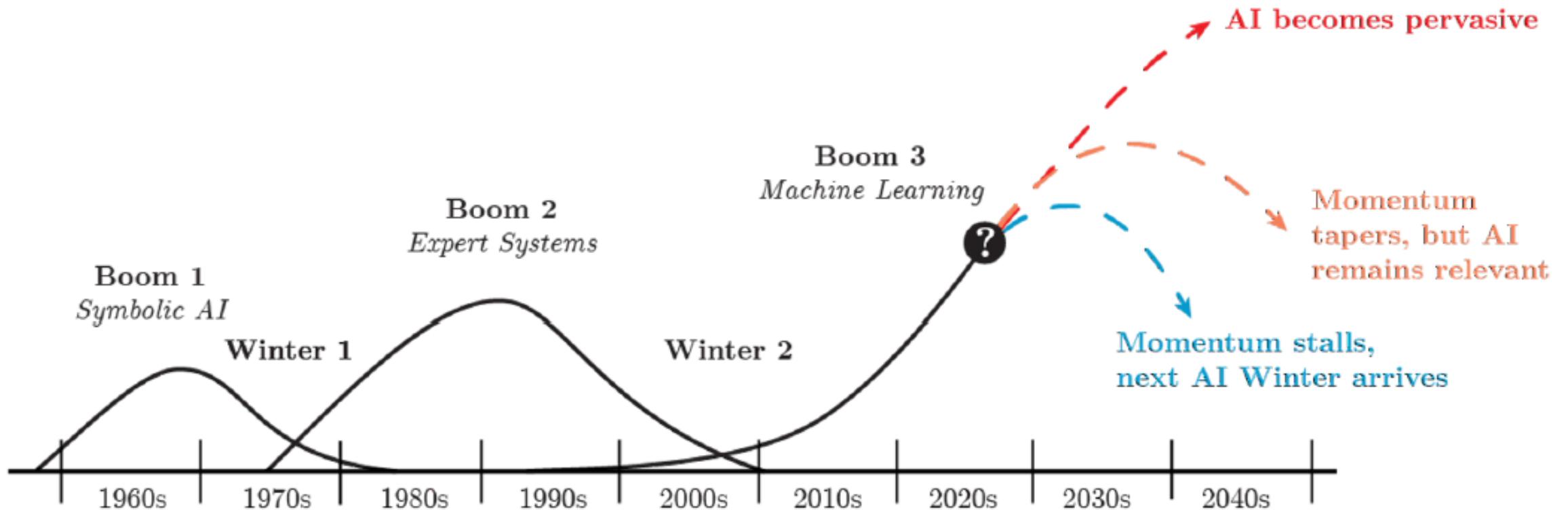


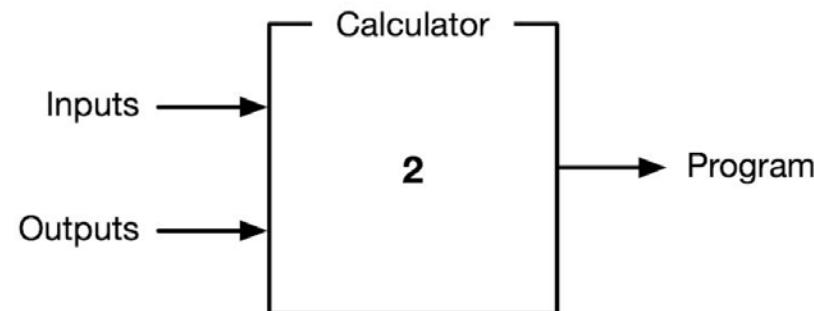
Fig. 1: Past AI Winters: What does the future hold?

Harguess, J. Is the Next Winter Coming for AI? The Elements of Making Secure and Robust AI. In Workshop on Trustworthy and Socially Responsible Machine Learning, NeurIPS 2022.



How is Artificial intelligence defined?

- **"bottom-up" or data-driven AI**
 - opposite approach, start from data to build incrementally and mathematically mechanisms taking decisions
 - First neuron (1943), first neural network machine (1950), neucognitron (1975), Decision Trees (1983), Backpropagation (1984-1986), Random Forest (1995), Support Vector Machine (1995), Boosting (1995), Deep Learning (1998/2006)...



(2) inductive machines

(Figure from: *Neurons spike back The invention of inductive machines and the artificial intelligence controversy*", D. Cardon, J.-P. Cointet, A. Mazières, Translated by Elizabeth Libbrecht In Réseaux Volume 211, Issue 5, 2018, pages 173 to 220)



How is Artificial intelligence defined?

- *AI is originally defined in 1956, by Marvin Lee Minsky:*
*“The construction of computer programs doing tasks, that are, **for the moment**, accomplished **more satisfactorily** by human beings because they require **high level mental processes** such as: learning, perceptual organization of memory and critical reasoning”.*
- There are so the “artificial” side with the usage of computers or sophisticated electronic processes and the side “intelligence” associated with its goal to imitate the (human) behavior.

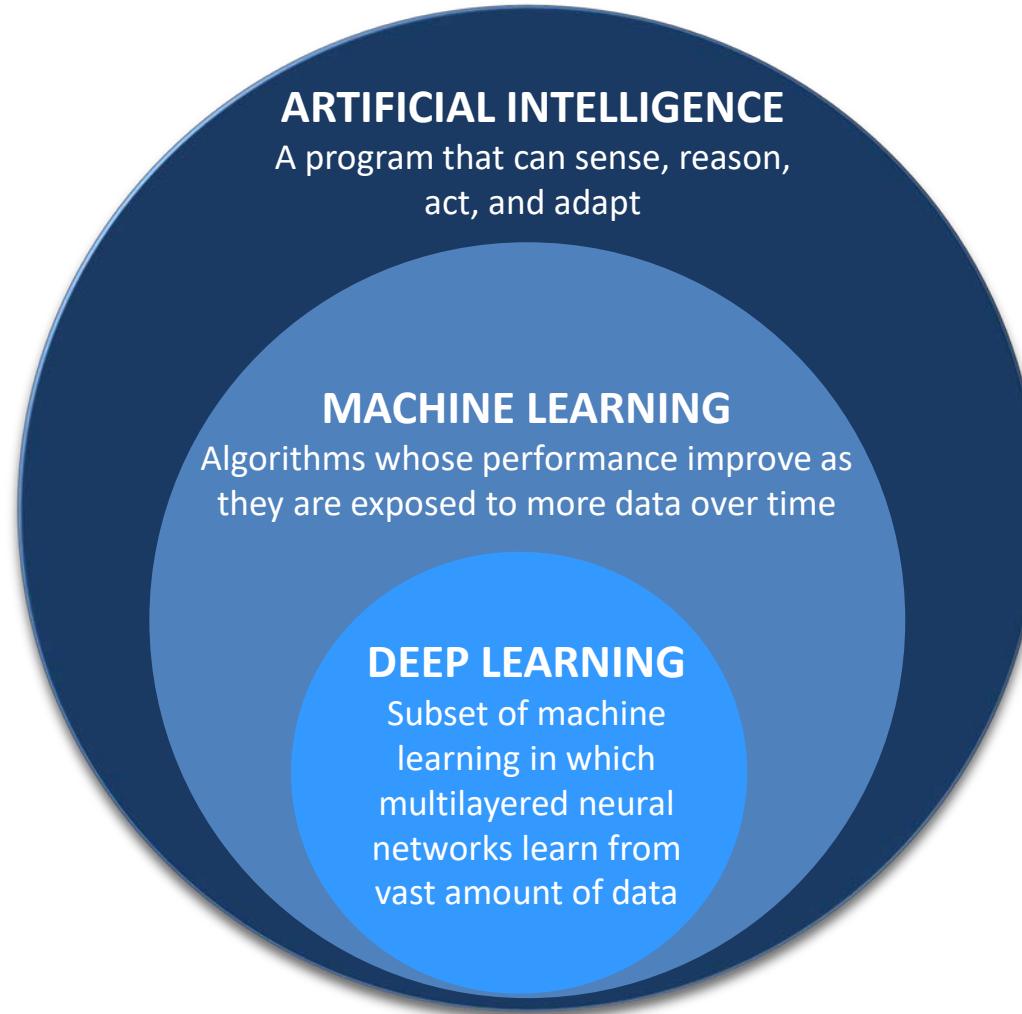


Why Artificial Intelligence is so difficult to grasp?

- Frequently, when a technique reaches **mainstream use**, it is **no longer considered as artificial intelligence**; this phenomenon is described as the *AI effect*: "AI is whatever hasn't been done yet." (*Larry Tesler's Theorem*)
- "As soon as it works, no one calls it AI any more." (John McCarthy)
- -> e.g. Path Finding (GPS), Chess electronic game, Alpha Go...
- Consequently, AI domain is continuously evolving and so very difficult to grasp.

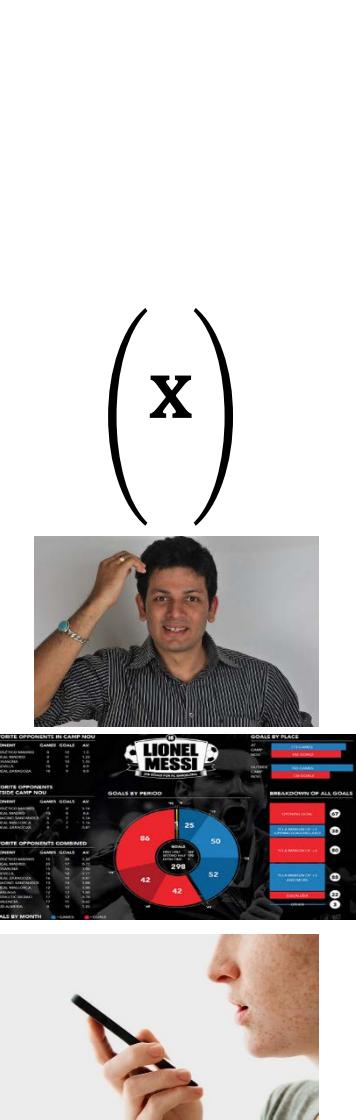


AI vs Machine Learning vs Deep Learning

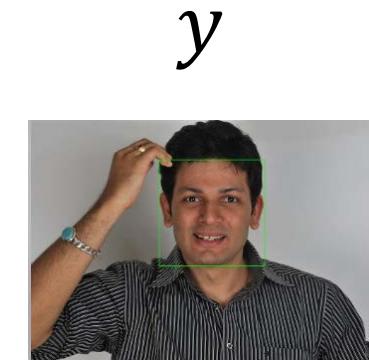




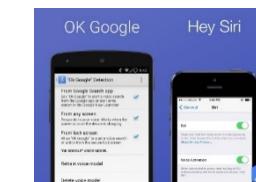
But what is Machine Learning?



Machine Learning



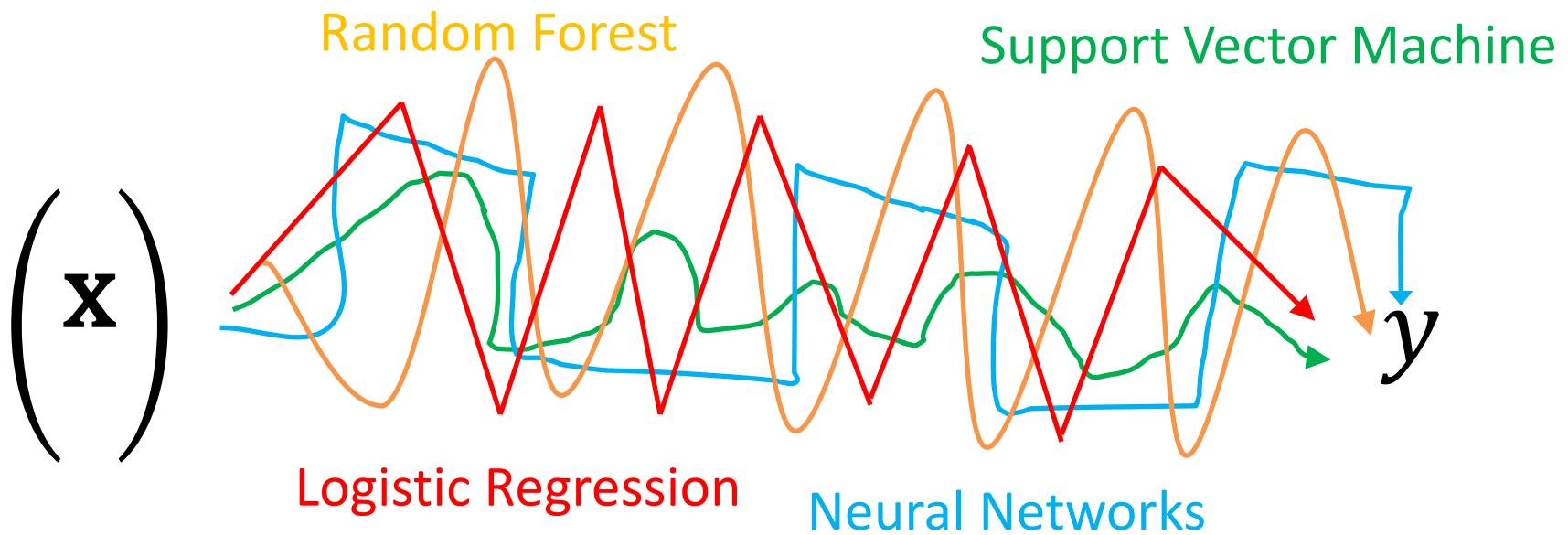
Sport bets





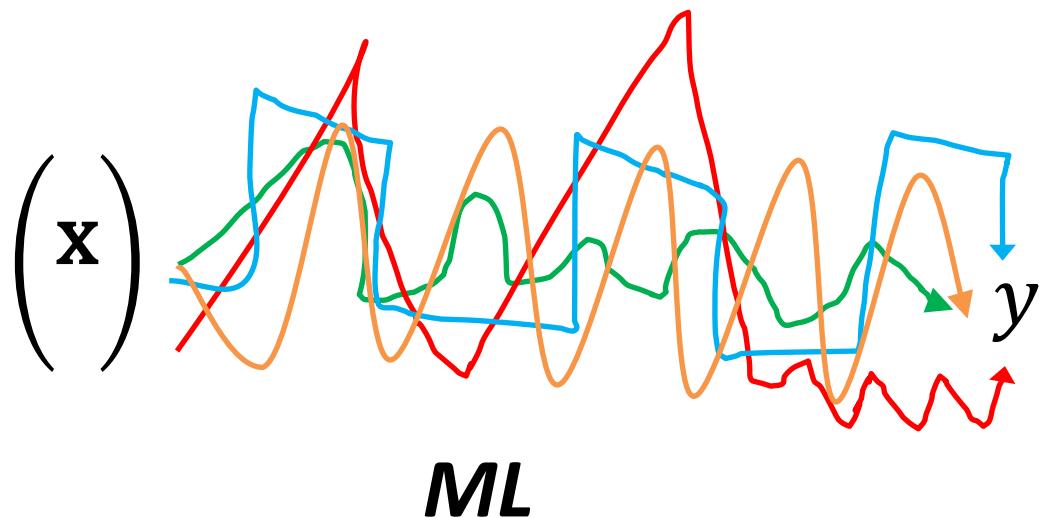
Machine Learning

$$\begin{pmatrix} \mathbf{x} \end{pmatrix} \xrightarrow{f(\mathbf{X}, \alpha) ?} y$$



Machine Learning

$$\begin{pmatrix} \mathbf{x} \end{pmatrix} \xrightarrow{f(\mathbf{X}, \alpha) ?} y$$

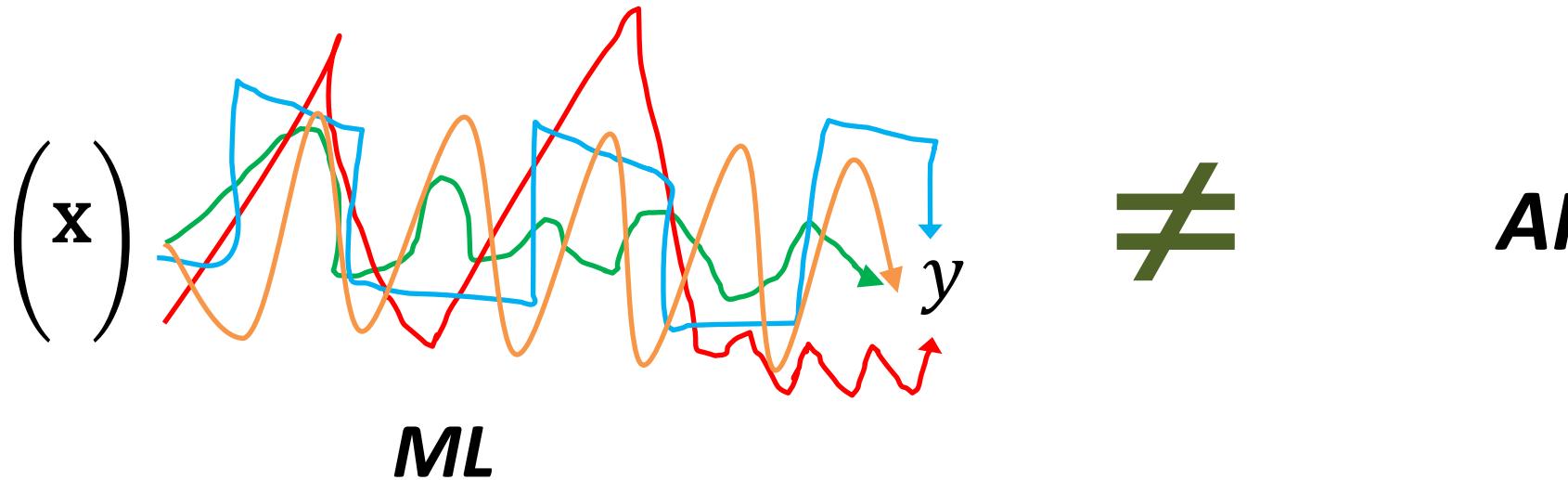


“Weather Forecasting”

Cf. Question trolley dilemma

Machine Learning

$$\begin{pmatrix} \mathbf{x} \end{pmatrix} \xrightarrow{f(\mathbf{X}, \alpha) ?} y$$



Francis Bach at *Frontier Research and Artificial Intelligence Conference*: “Machine Learning is not AI”

(https://erc.europa.eu/sites/default/files/events/docs/Francis_Bach-SEQUOIA-Robust-algorithms-for-learning-from-modern-data.pdf

<https://webcast.ec.europa.eu/erc-conference-frontier-research-and-artificial-intelligence-25#>)

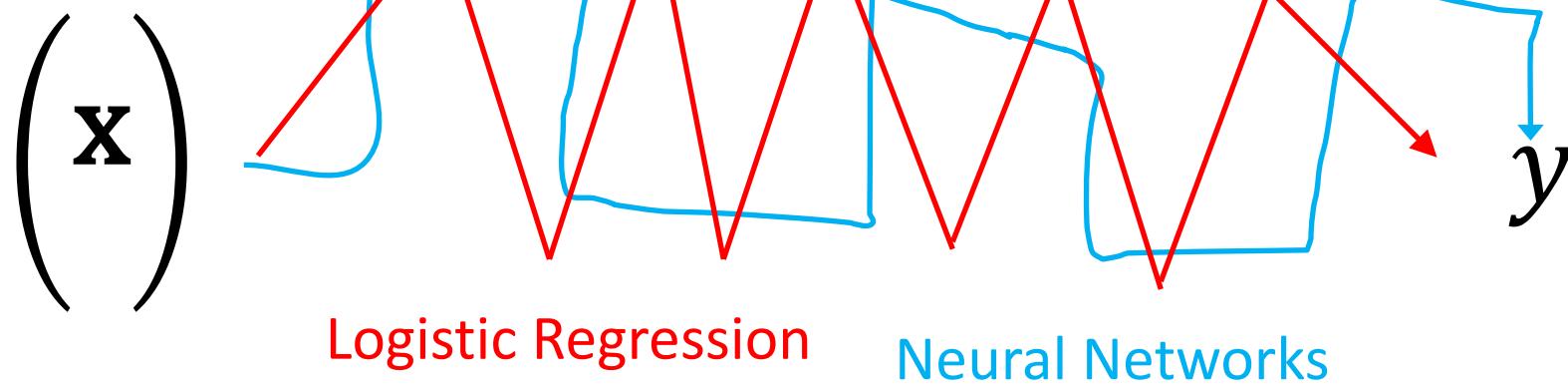
Beware of the diversion!



Trolley dilemma

Machine Learning

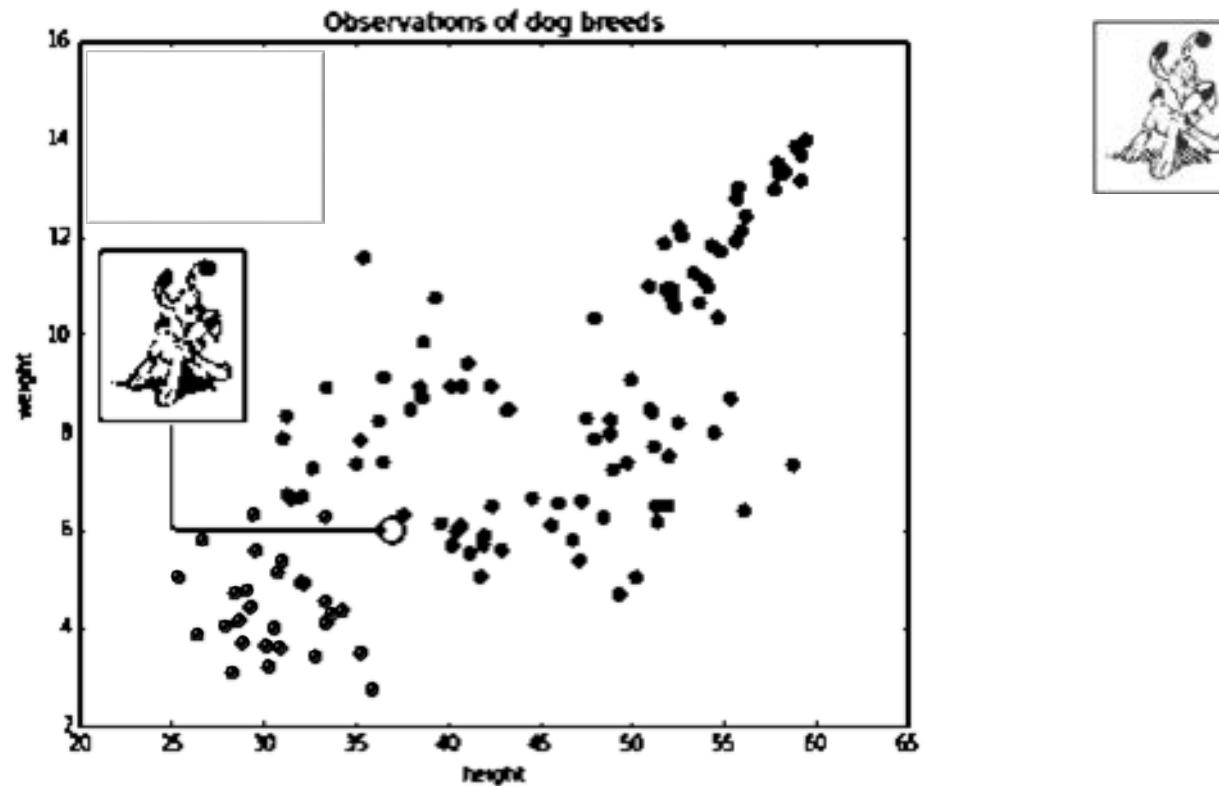
$$\begin{pmatrix} \mathbf{x} \end{pmatrix} \xrightarrow{f(\mathbf{X}, \alpha) ?} y$$





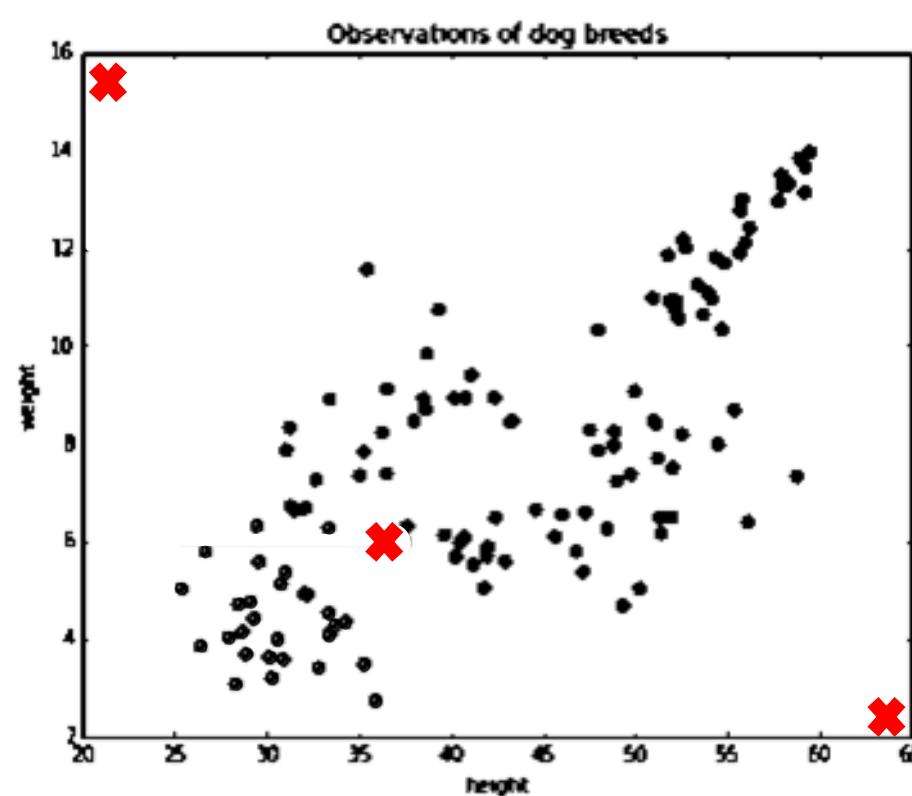
Machine Learning is Statistics?

What breed is that Dogmatix (Idéfix)?



The illustrations of the slides in this section come from the blog "Bayesian Vitalstatistix: What Breed of Dog was Dogmatix?"

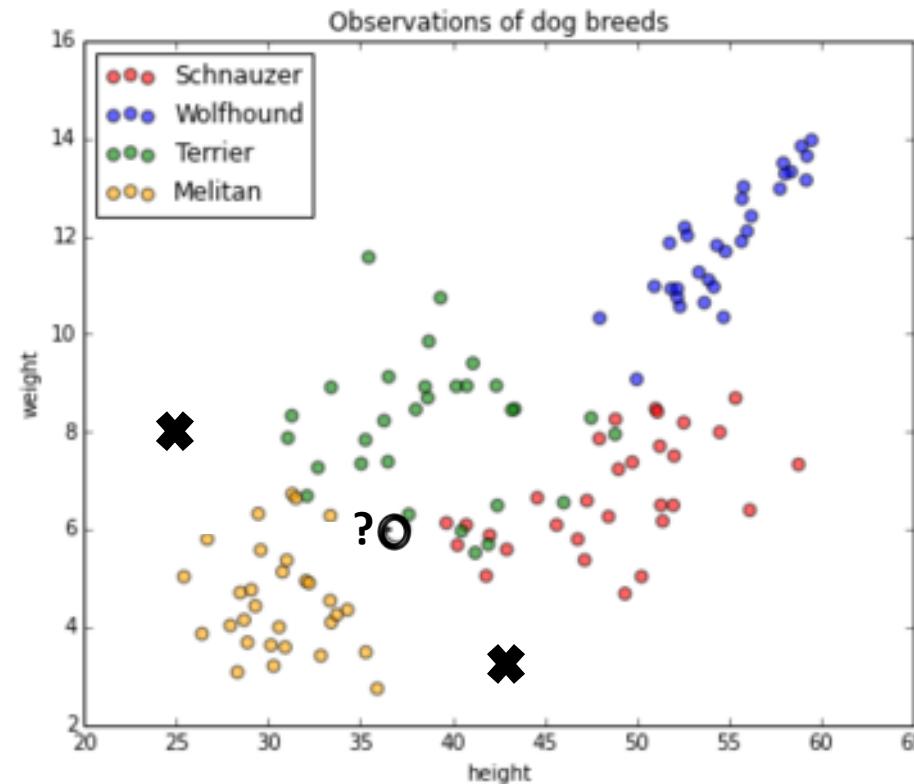
Does any real dog get this height and weight?



- Let us consider x , vectors independently generated in \mathbb{R}^d (here \mathbb{R}^2), following a probability distribution fixed but *unknown* $P(x)$.

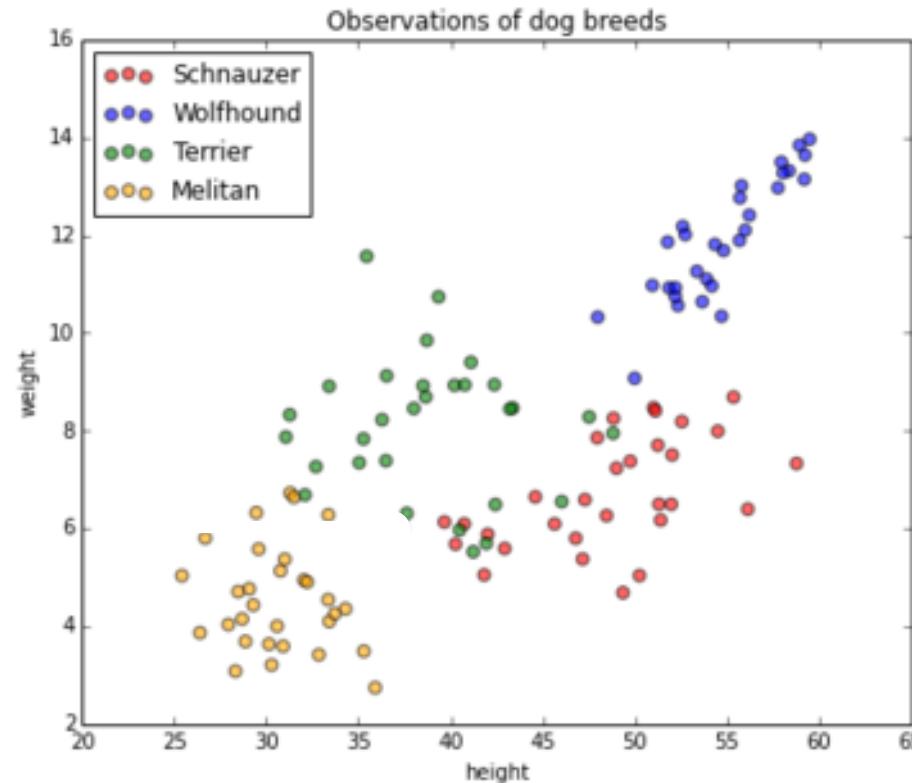
•

What should be the breed of these dogs?



- An Oracle assigns a value y to each vector x following a probability distribution $P(y/x)$ also fixed but *unknown*.

An oracle provides me with examples?

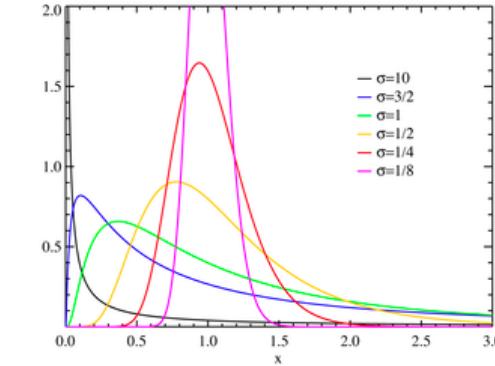
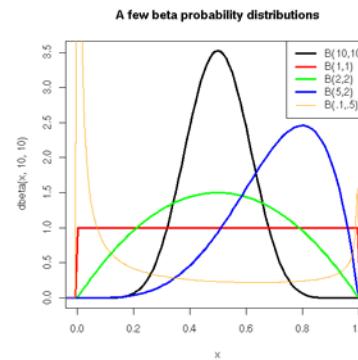
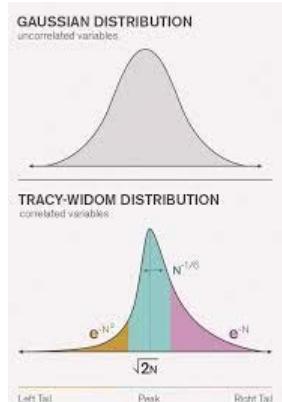


- Let S be a training set
$$S = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\},$$
with m training samples i.i.d. which follow the **joint probability**
$$P(x, y) = P(x)P(y|x).$$

Machine Learning vs Statistics

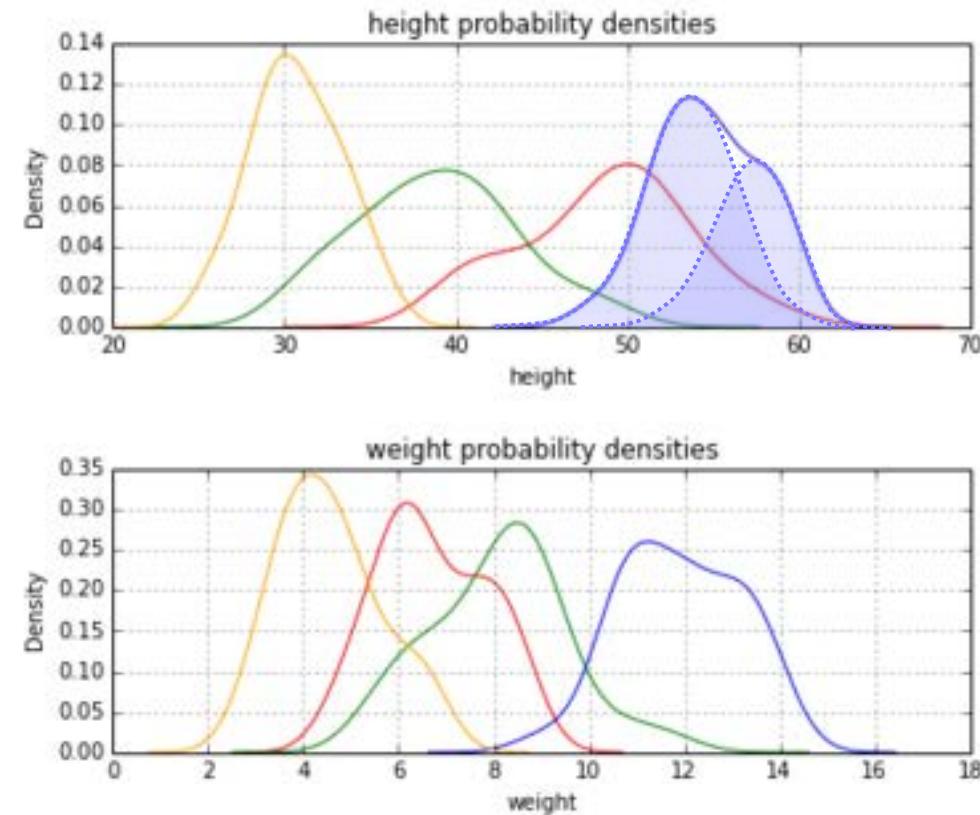
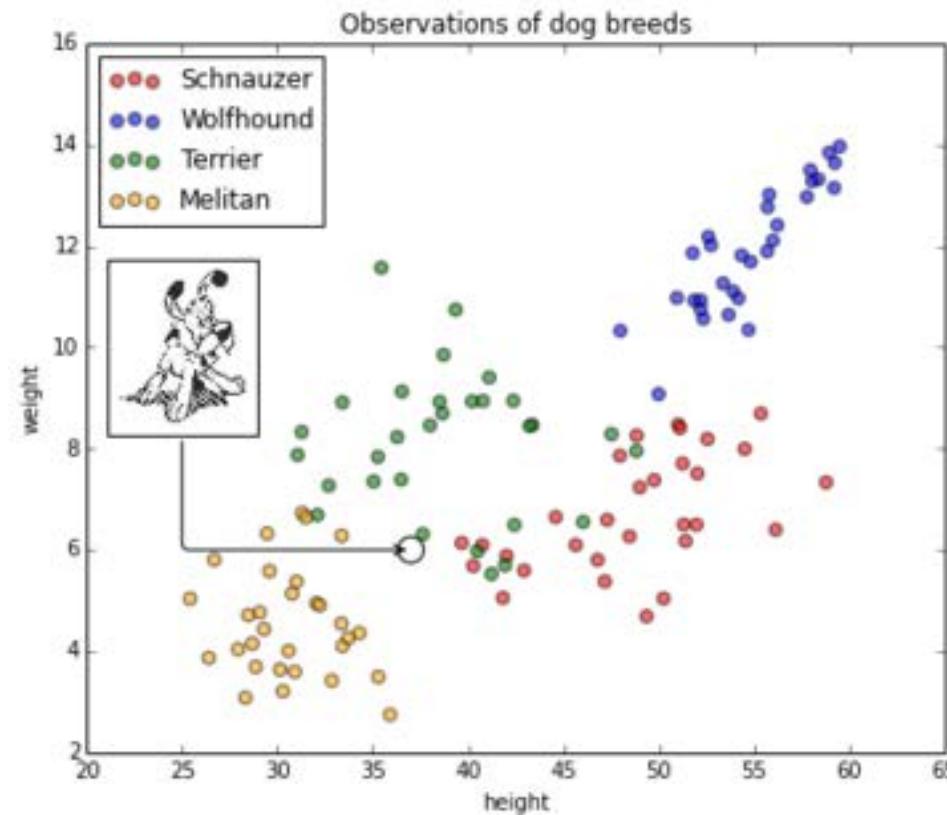
- Statistical path
- Machine learning path

Statistical solution: Models, Hypotheses...

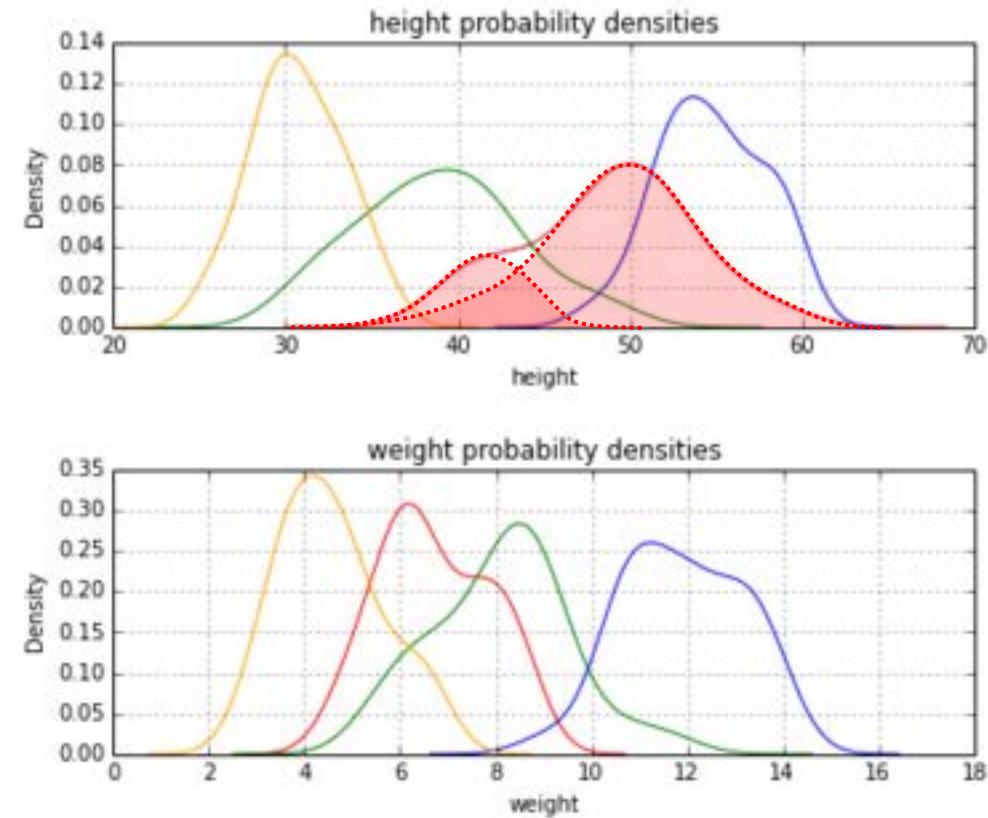
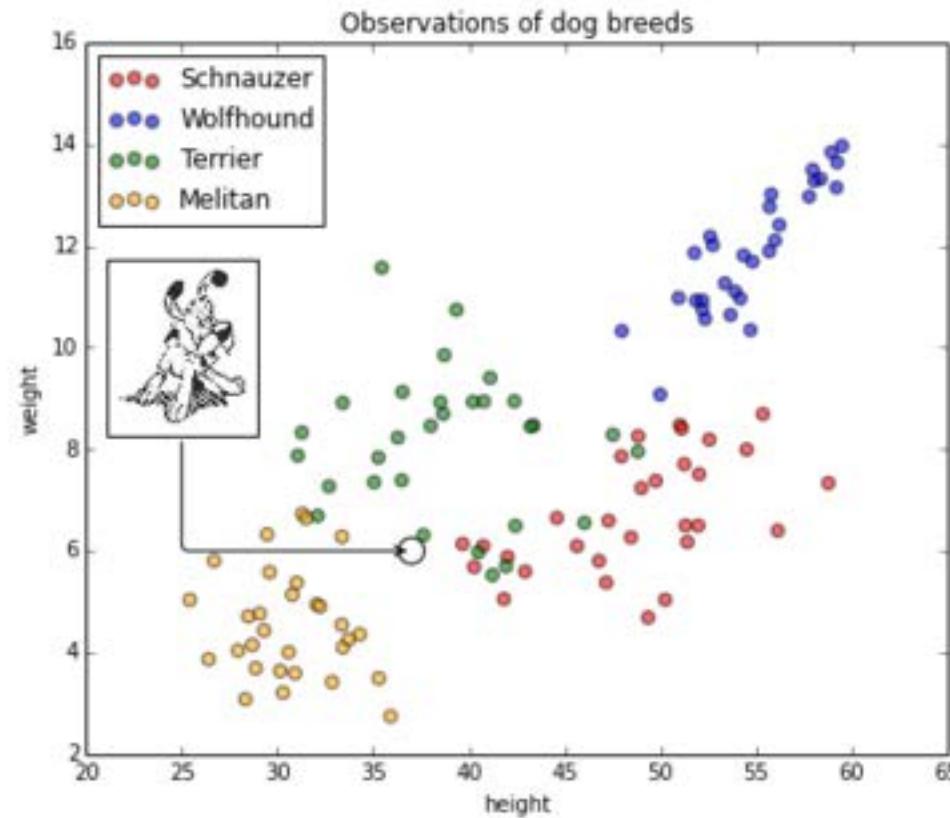


Statisticians make assumptions on the models of $P(X)$ and $P(Y)$.

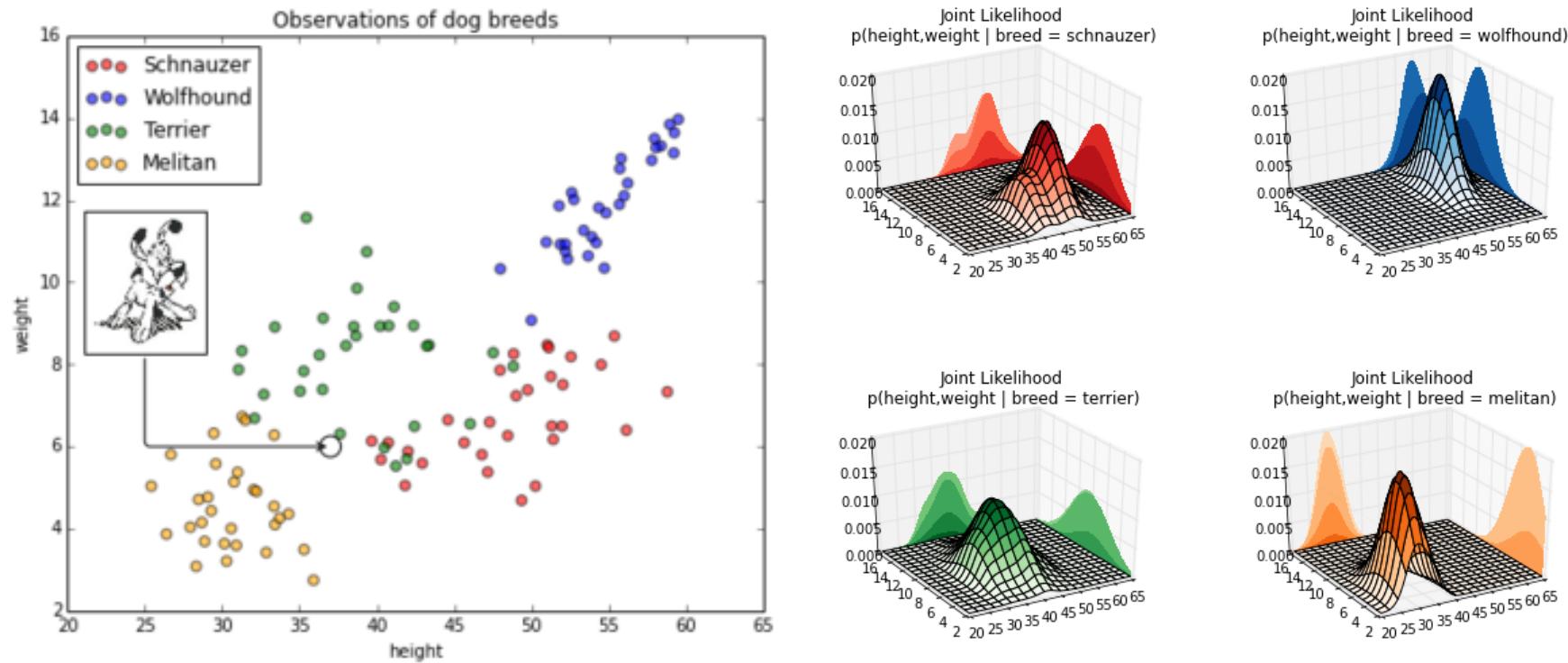
Statistical solution $P(\text{height}, \text{weight}|\text{breed})\dots$



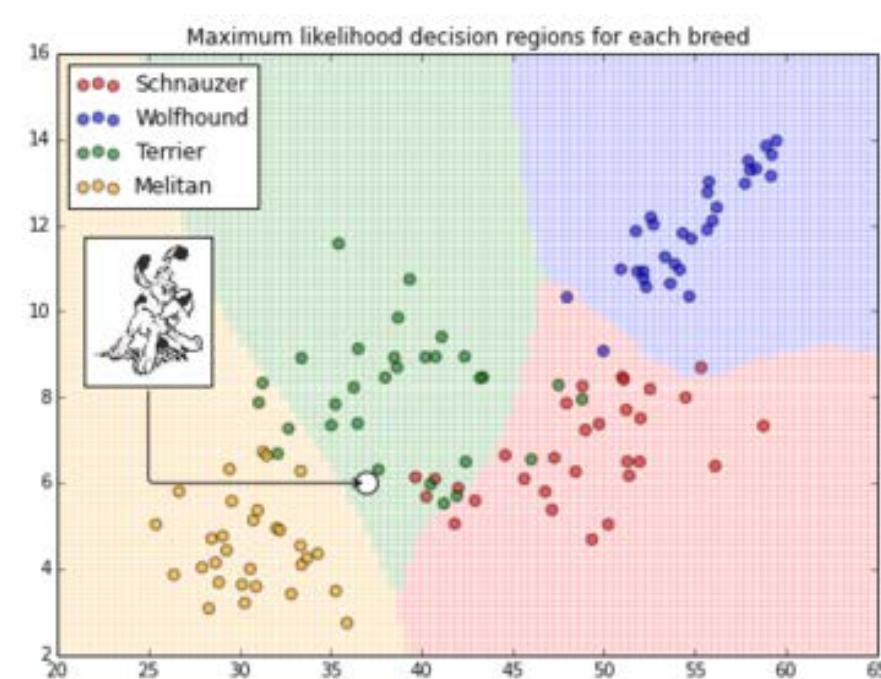
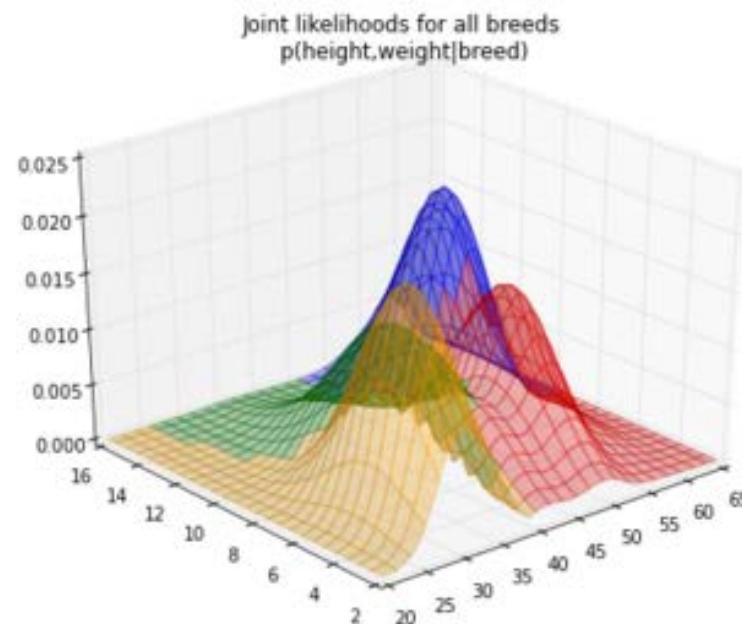
Statistical solution $P(\text{height}, \text{weight}|\text{breed})\dots$



Statistical solution $P(\text{height}, \text{weight} | \text{breed})...$



Statistical solution $P(\text{height}, \text{weight}|\text{breed})$...

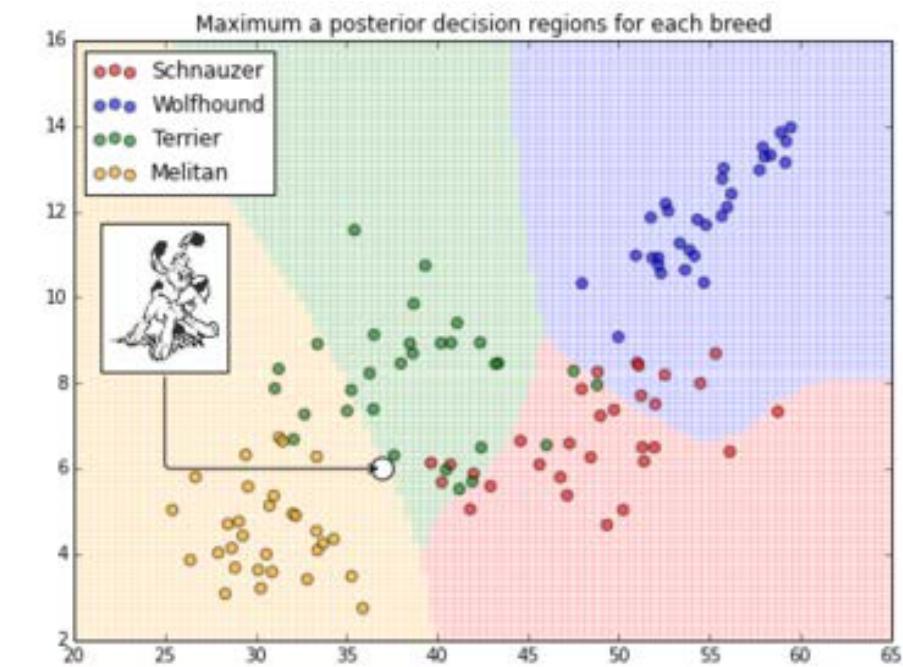
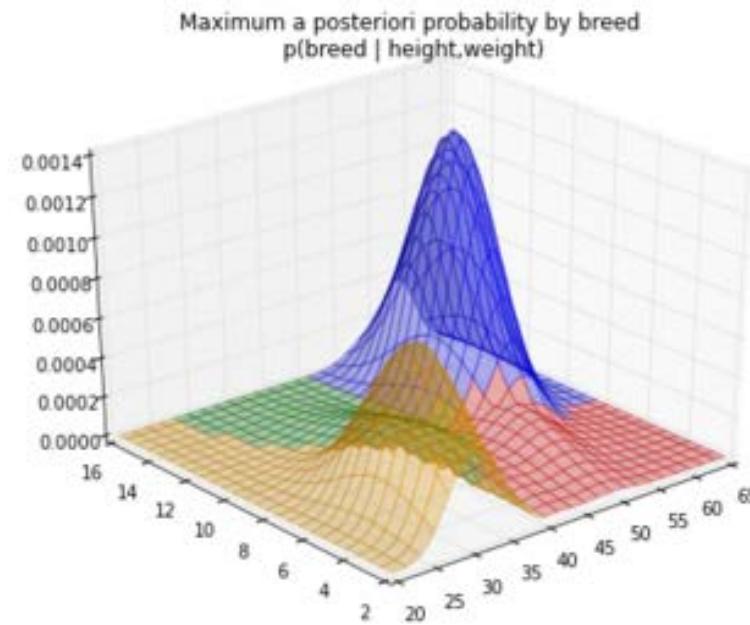




Statistical solution: Bayes, P(breed|height, weight)...

Reminder Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

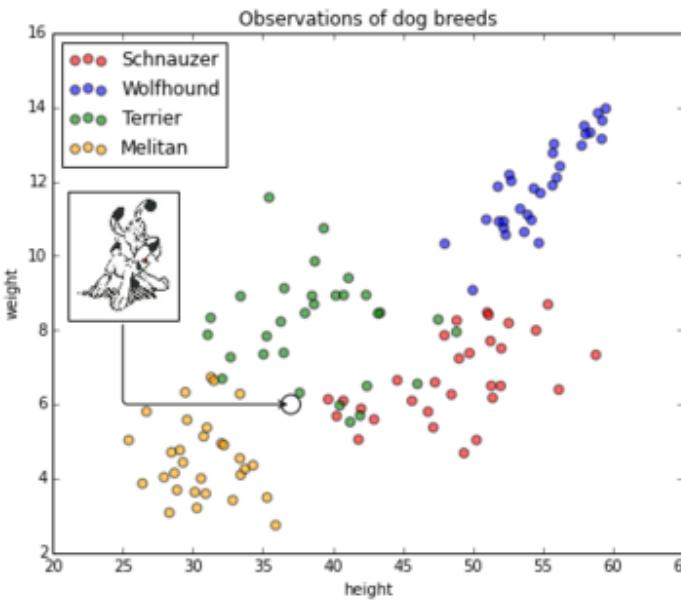


$$P(\text{breed} | \text{height}, \text{weight}) = \frac{P(\text{height}, \text{weight} | \text{breed})P(\text{breed})}{P(\text{height}, \text{weight})}$$

Machine Learning vs Statistics

- Statistical path
- Machine learning path

Machine Learning



- we have a learning machine (i.e. an algorithm) which can provide a family of functions $\{f(\mathbf{x}; \boldsymbol{\alpha})\}$, where $\boldsymbol{\alpha}$ is a set of parameters.

$$\begin{pmatrix} \mathbf{x} \end{pmatrix} \xrightarrow{f(\mathbf{X}, \boldsymbol{\alpha}) ?} y$$

The problem of (Machine) Learning

$$\begin{pmatrix} \mathbf{x} \end{pmatrix} \xrightarrow{f(\mathbf{x}, \alpha)} y$$

- The problem of learning consists in finding the function (among the $\{f(\mathbf{x}; \alpha)\}$) which provides the best approximation \hat{y} of the true label y given by the Oracle.
- “**best**” is defined in terms of minimizing a specific *error measure/cost/loss related to your problem/objectives*
 $L((\mathbf{x}, y), \alpha) \in [a; b]$.

The problem of (Machine) Learning

- Thus, the objective is to minimize the (*real*) **Risk**, i.e. the expectation of the error cost:

$$R(\alpha) = \int L((x, y), \alpha) dP(x, y)$$

where $P(x, y)$ is unknown.

- The training set $S = \{(x_i, y_i)\}_{i=1,\dots,m}$ is built through an i.i.d. sampling according to $P(x, y)$. Since we cannot compute $R(\alpha)$, we look for minimizing the **Empirical Risk** instead:

$$R_{emp}(\alpha) = \frac{1}{m} \sum_{k=1}^m L((x_i, y_i), \alpha)$$



Machine Learning fundamental Hypothesis

$S = \{(x_i, y_i)\}_{i=1, \dots, m}$ is built through an *i.i.d.* sampling according to $P(x, y)$.

Machine Learning  *Statistics*

Train through Cross-Validation

Machine Learning  *Statistics*

Training set & Test set have to be distributed according to the same law

Vapnik learning theory (1995)

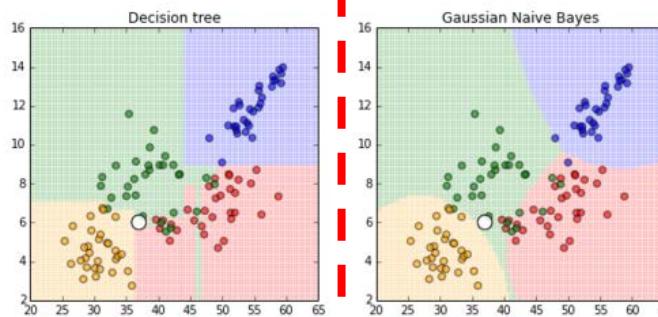
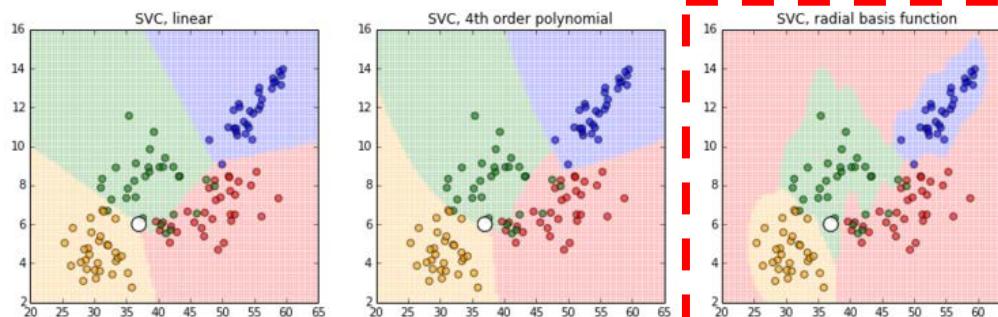
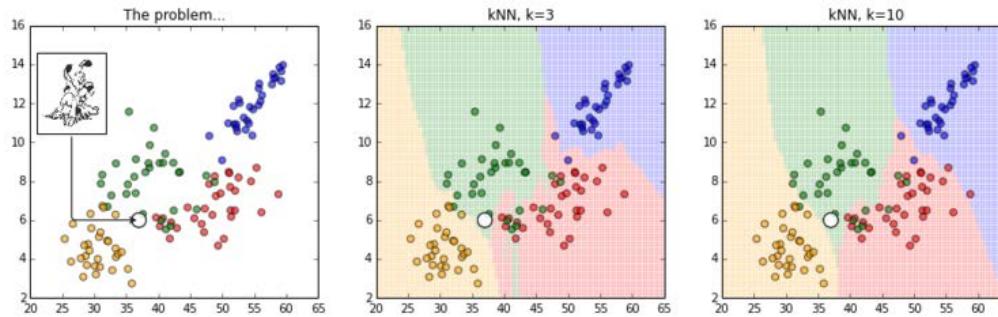
Vapnik had proven the following equation $\forall m$ with a probability at least equal to $1 - \eta$:

$$R(\alpha_m) \leq R_{emp}(\alpha_m) + (b - a) \sqrt{\frac{d_{VC}(\ln(2m/d_{VC}) + 1) - \ln(\eta/4)}{m}}$$

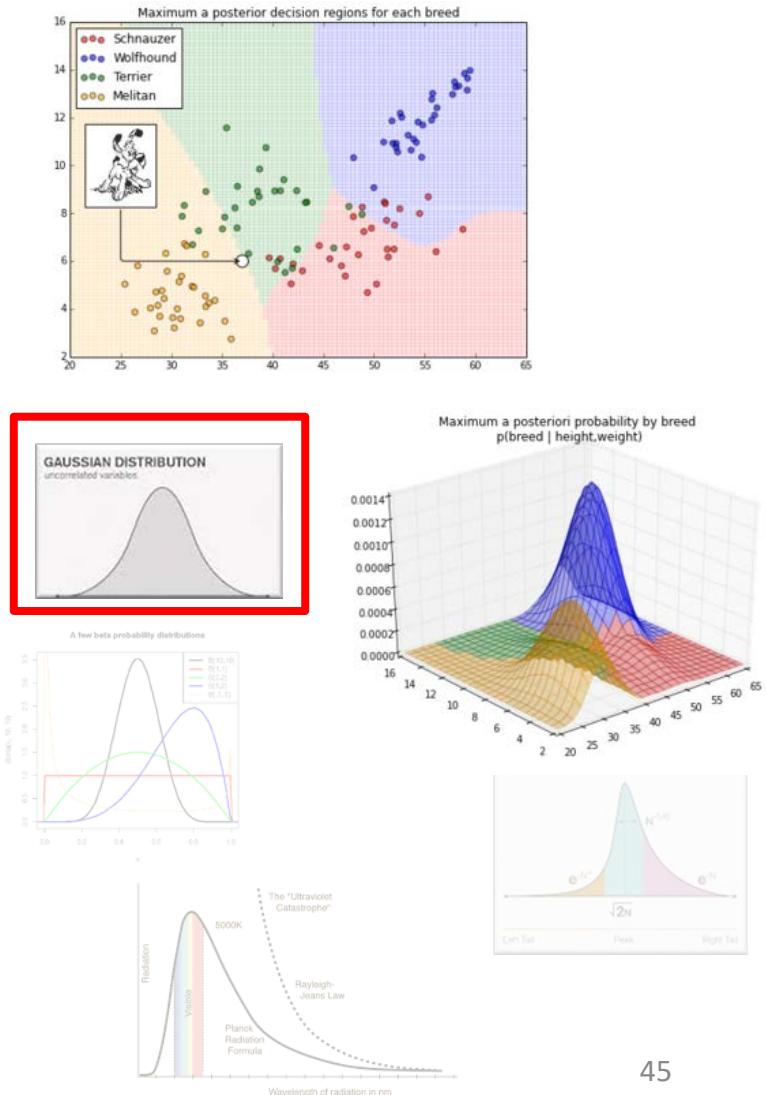
Training Error *Generalization Error*

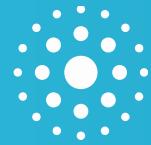
Thus minimizing the **Risk** depends on minimizing the **Empirical Risk** and the **Generalization Error** of the model which depends on m (the number of training sample), and d_{VC} (the complexity of the model family chosen, also called *Vapnik-Chervonenkis Dimension*).

Machine Learning vs Statistics



VS





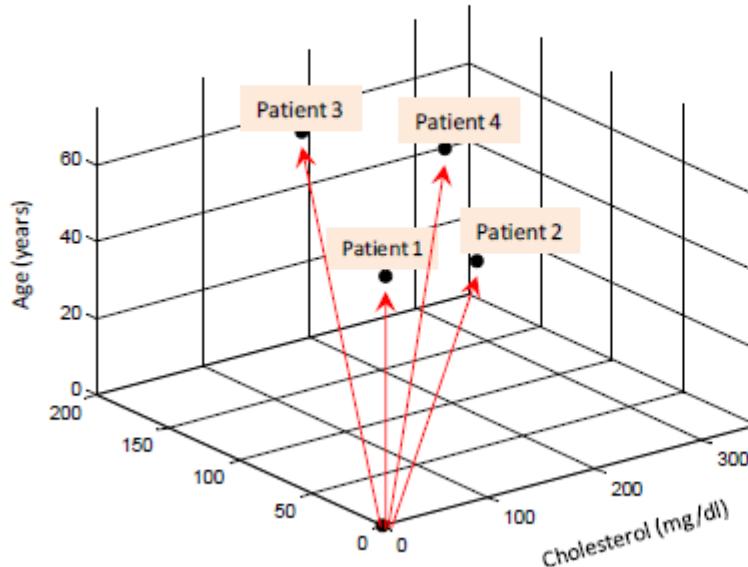
MATHEMATICAL BASICS



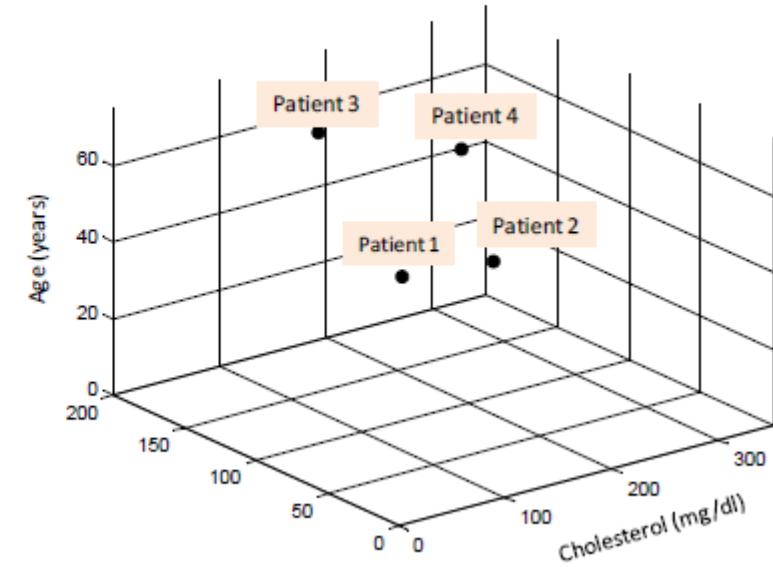
How to represent samples geometrically? Vectors & points in n-dimensional space (R^n)

Patient id	Cholesterol (mg/dl)	Systolic BP (mmHg)	Age (years)	Tail of the vector	Arrow-head of the vector
1	150	110	35	(0,0,0)	(150, 110, 35)
2	250	120	30	(0,0,0)	(250, 120, 30)
3	140	160	65	(0,0,0)	(140, 160, 65)
4	300	180	45	(0,0,0)	(300, 180, 45)

Vector representation



Point representation





Basic operation on vectors in \mathbb{R}^n

1. Multiplication by a scalar

Consider a vector $\mathbf{a} = (a_1, a_2, \dots, a_n)$ and a scalar c

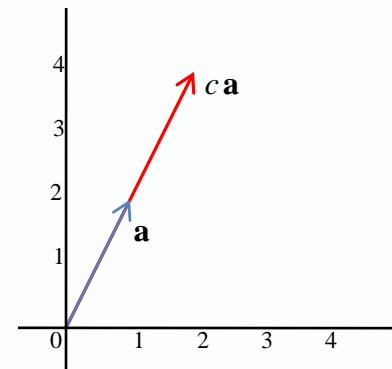
Define: $c\mathbf{a} = (ca_1, ca_2, \dots, ca_n)$

When you multiply a vector by a scalar, you “stretch” it in the same or opposite direction depending on whether the scalar is positive or negative.

$$\mathbf{a} = (1, 2)$$

$$c = 2$$

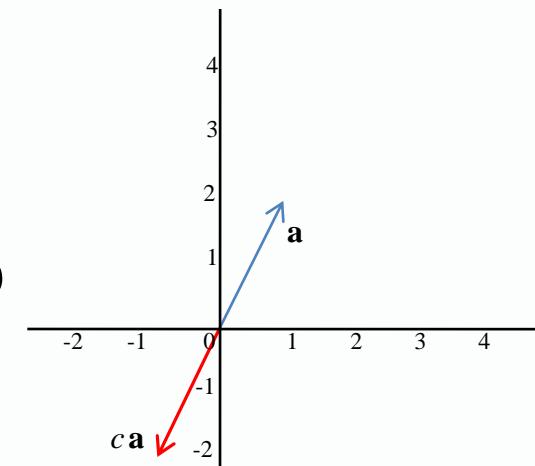
$$c\mathbf{a} = (2, 4)$$



$$\mathbf{a} = (1, 2)$$

$$c = -1$$

$$c\mathbf{a} = (-1, -2)$$



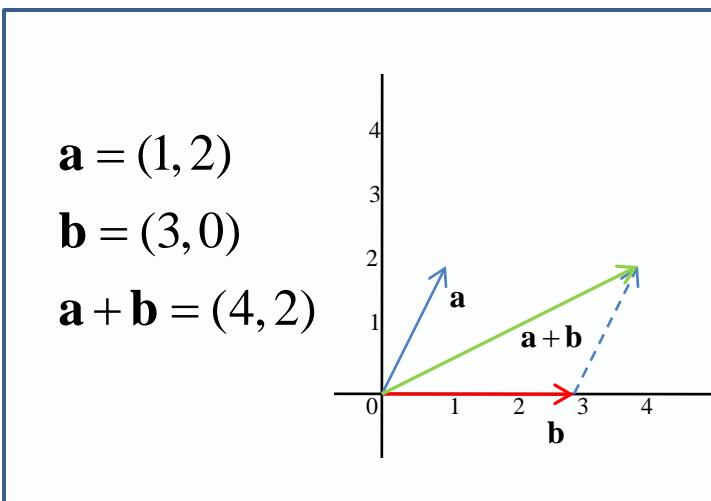


Basic operation on vectors in \mathbb{R}^n

2. Addition

Consider vectors $\mathbf{a} = (a_1, a_2, \dots, a_n)$ and $\mathbf{b} = (b_1, b_2, \dots, b_n)$

Define: $\mathbf{a} + \mathbf{b} = (a_1 + b_1, a_2 + b_2, \dots, a_n + b_n)$



Recall addition of forces in classical mechanics.



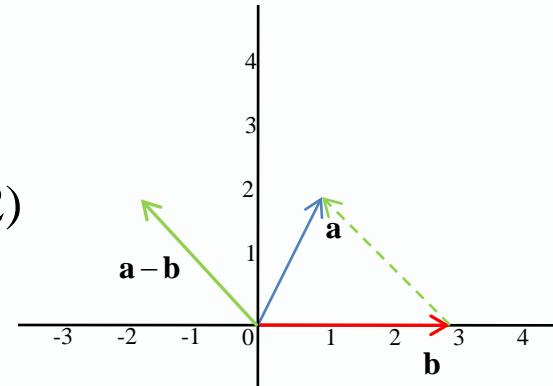
Basic operation on vectors in \mathbb{R}^n

3. Subtraction

Consider vectors $\mathbf{a} = (a_1, a_2, \dots, a_n)$ and $\mathbf{b} = (b_1, b_2, \dots, b_n)$

Define: $\mathbf{a} - \mathbf{b} = (a_1 - b_1, a_2 - b_2, \dots, a_n - b_n)$

$$\begin{aligned}\mathbf{a} &= (1, 2) \\ \mathbf{b} &= (3, 0) \\ \mathbf{a} - \mathbf{b} &= (-2, 2)\end{aligned}$$



What vector do we need to add to \vec{b} to get \vec{a} ? I.e., similar to subtraction of real numbers.



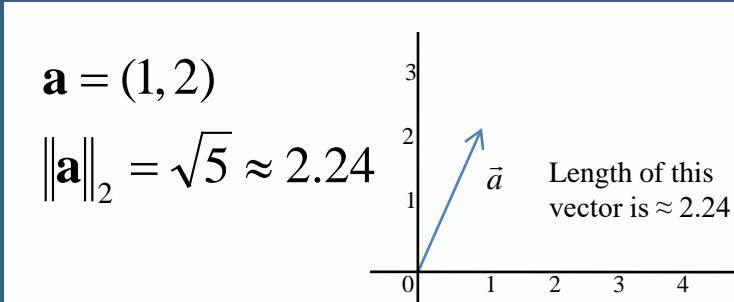
Basic operation on vectors in \mathbf{R}^n

4. Euclidian length or L2-norm

Consider a vector $\mathbf{a} = (a_1, a_2, \dots, a_n)$

Define the L2-norm: $\|\mathbf{a}\|_2 = \sqrt{a_1^2 + a_2^2 + \dots + a_n^2}$

We often denote the L2-norm without subscript, i.e. $\|\mathbf{a}\|$



L2-norm is a typical way to measure length of a vector; other methods to measure length also exist.



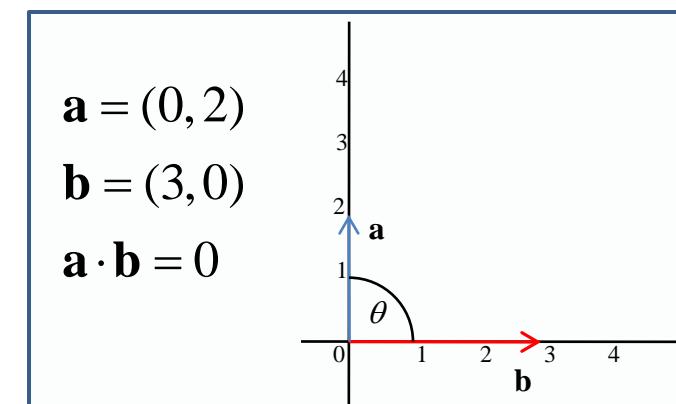
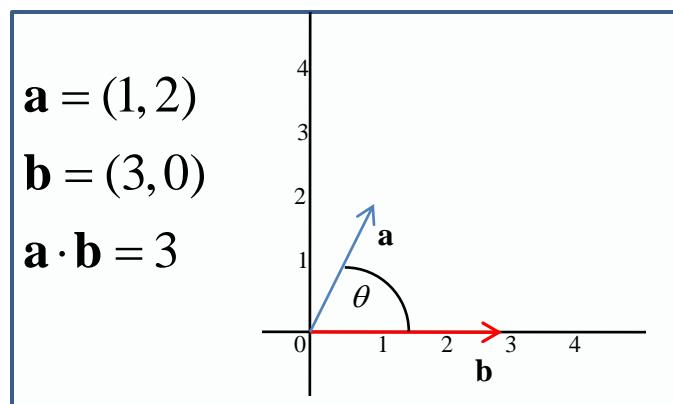
Basic operation on vectors in \mathbb{R}^n

5. Dot product

Consider vectors $\mathbf{a} = (a_1, a_2, \dots, a_n)$ and $\mathbf{b} = (b_1, b_2, \dots, b_n)$

Define dot product: $\mathbf{a} \cdot \mathbf{b} = a_1b_1 + a_2b_2 + \dots + a_nb_n = \sum_{i=1}^n a_i b_i$

The law of cosines says that $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|_2 \|\mathbf{b}\|_2 \cos \theta$ where θ is the angle between \mathbf{a} and \mathbf{b} . Therefore, when the vectors are perpendicular $\mathbf{a} \cdot \mathbf{b} = 0$.





Basic operation on vectors in \mathbf{R}^n

5. Dot product (continued)

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \dots + a_n b_n = \sum_{i=1}^n a_i b_i$$

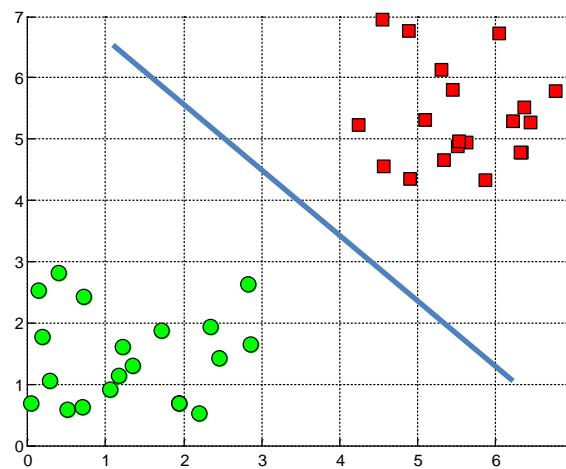
- Property: $\mathbf{a} \cdot \mathbf{a} = a_1 a_1 + a_2 a_2 + \dots + a_n a_n = \|\mathbf{a}\|_2^2$
- In the classical regression equation $y = \mathbf{w} \cdot \mathbf{x} + b$
the response variable y is just a dot product of the vector representing patient characteristics (\mathbf{X}) and the regression weights vector (\mathbf{w}) which is common across all patients plus an offset b .



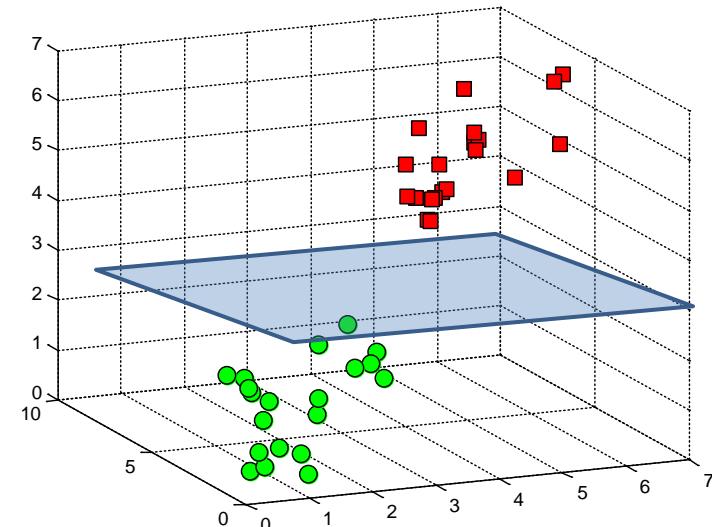
Hyperplanes as decision surfaces

- A hyperplane is a linear decision surface that splits the space into two parts;
- It is obvious that a hyperplane is a binary classifier.

A hyperplane in \mathbb{R}^2 is a line



A hyperplane in \mathbb{R}^3 is a plane



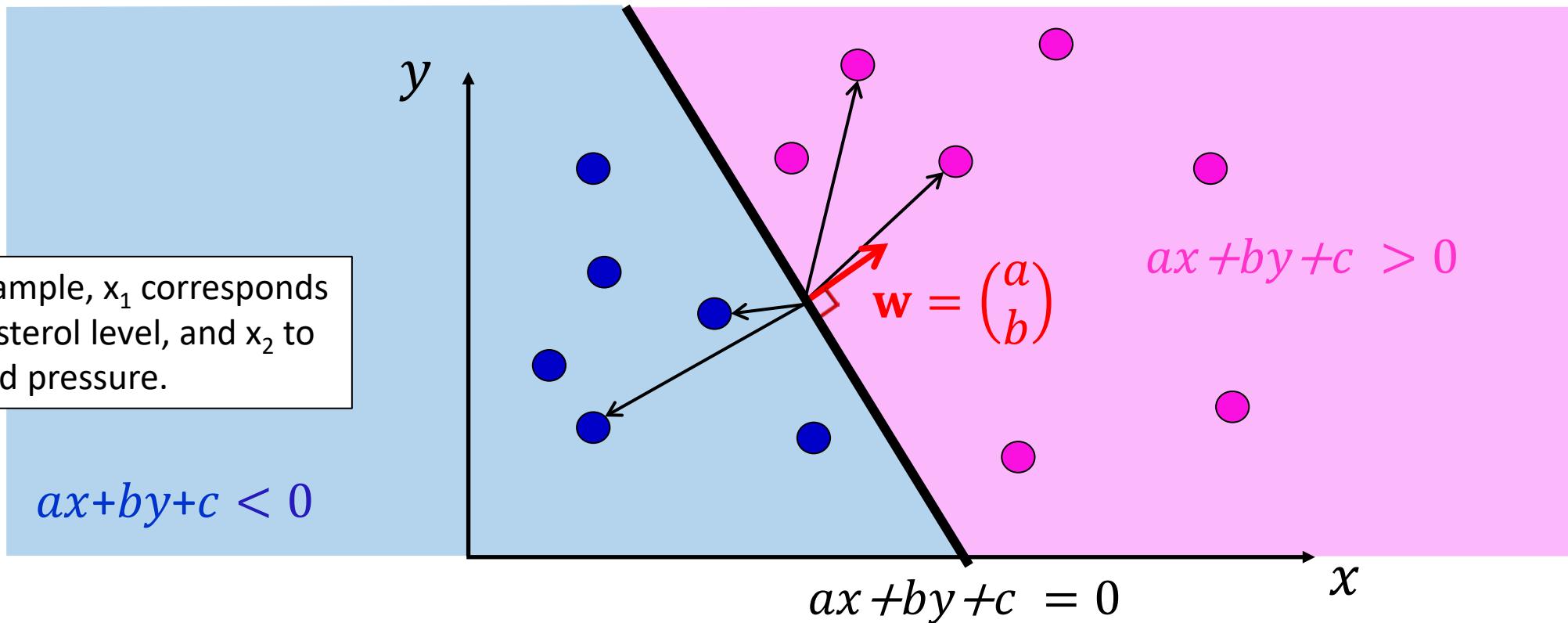
A hyperplane in \mathbb{R}^n is an $n-1$ dimensional subspace

Binary classification in 2D = defining a straight line

Reminder of the equation of a straight line in the plane (in 2D), and of the equations of the two semiplanes that it defines by cutting the space in 2.

Here, for example, x_1 corresponds to the cholesterol level, and x_2 to systolic blood pressure.

$$ax+by+c < 0$$

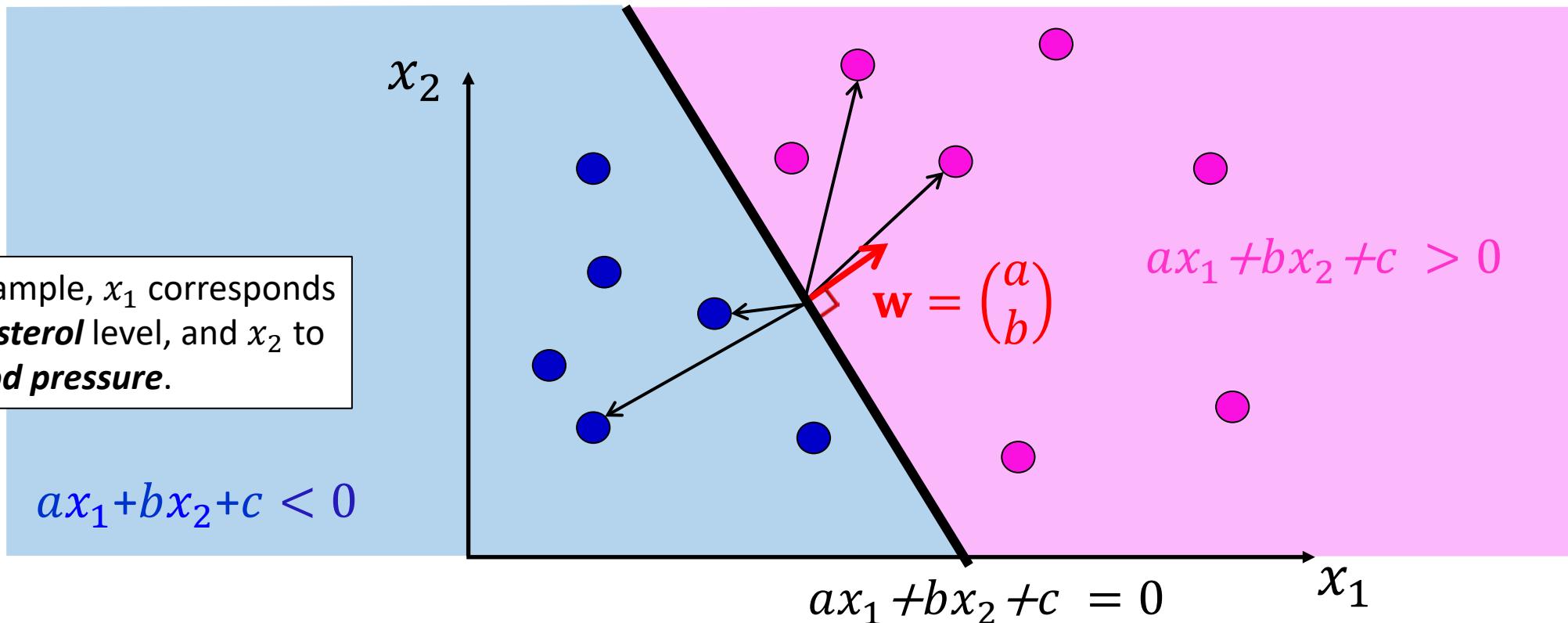


Binary classification in 2D = defining a straight line

Reminder of the equation of a straight line in the plane (in 2D), and of the equations of the two semiplanes that it defines by cutting the space in 2.

Here, for example, x_1 corresponds to the *cholesterol* level, and x_2 to *systolic blood pressure*.

$$ax_1 + bx_2 + c < 0$$

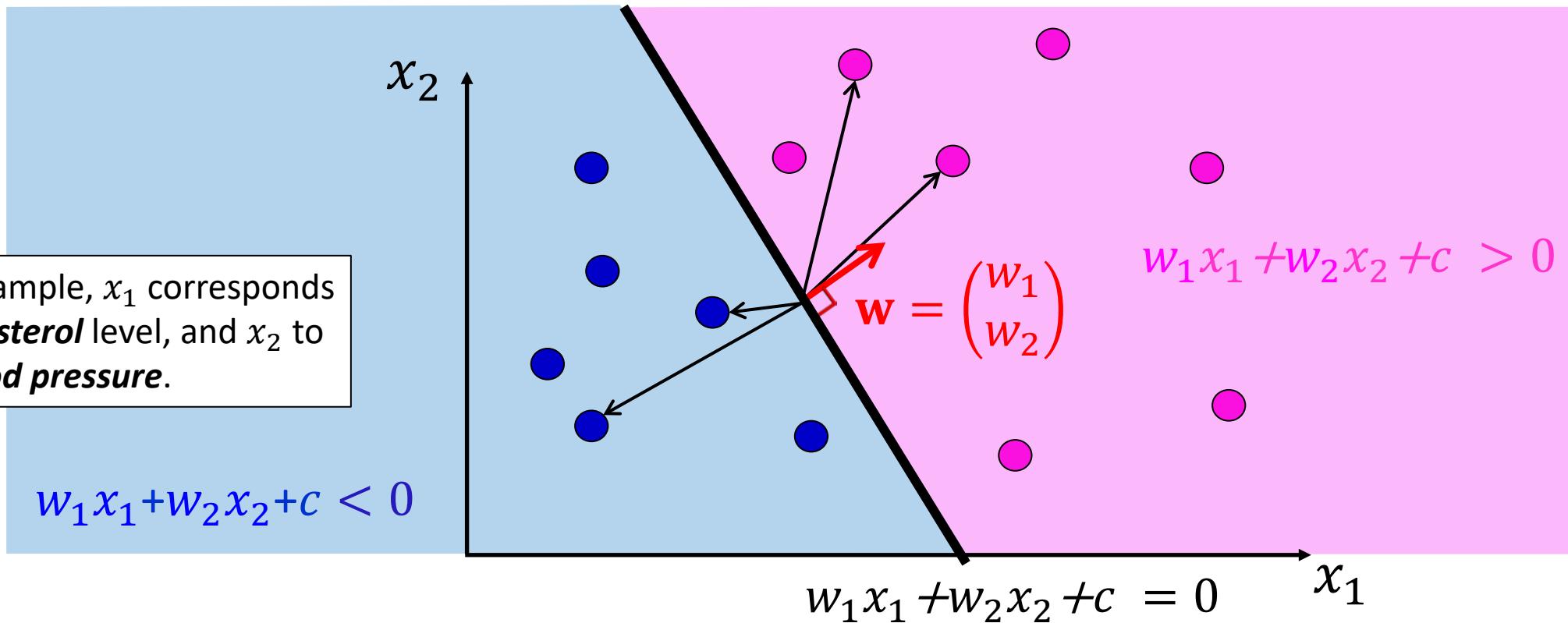


Binary classification in 2D = defining a straight line

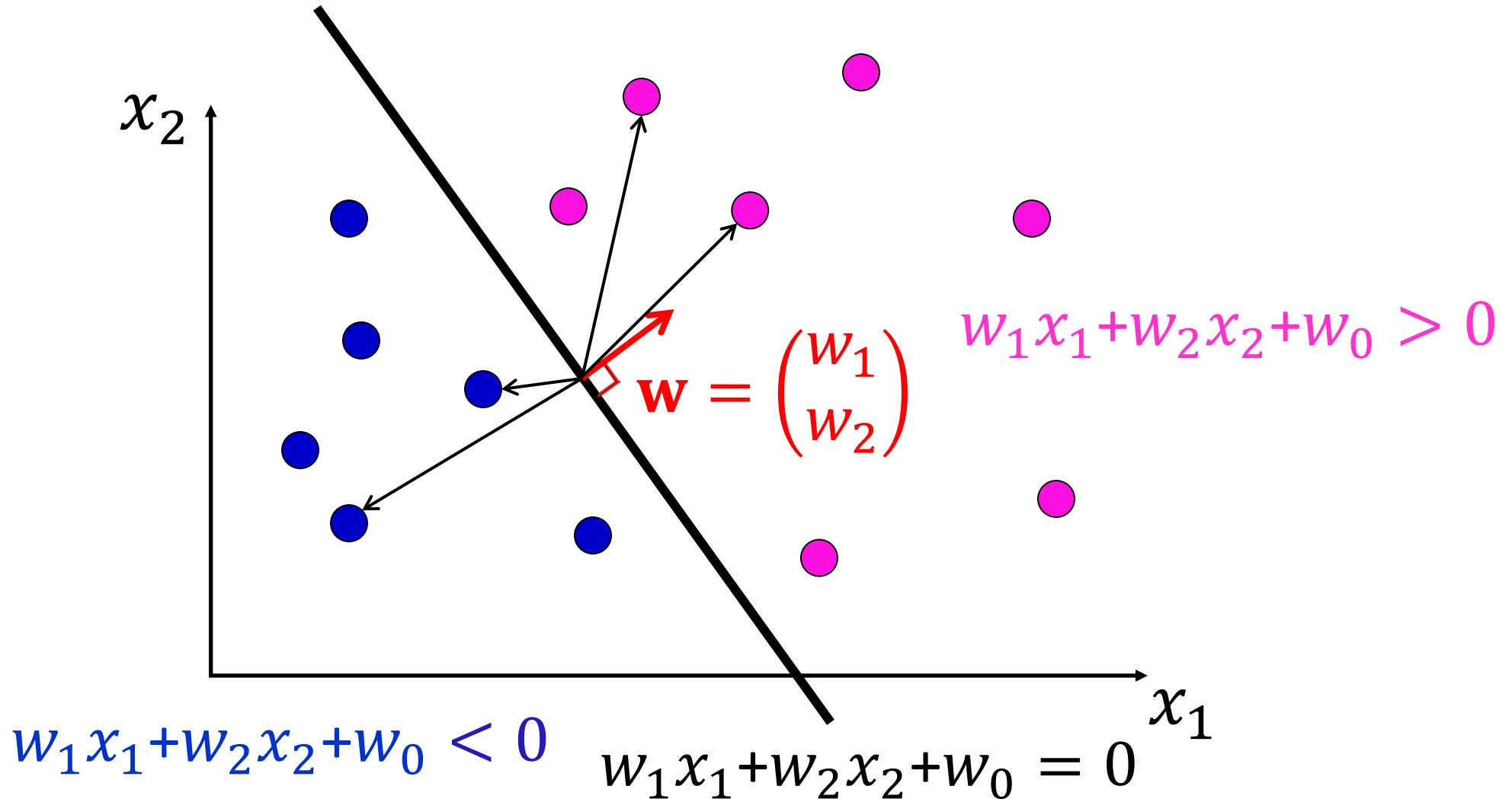
Usually weights of a neuron are called w_i .

Here, for example, x_1 corresponds to the **cholesterol** level, and x_2 to **systolic blood pressure**.

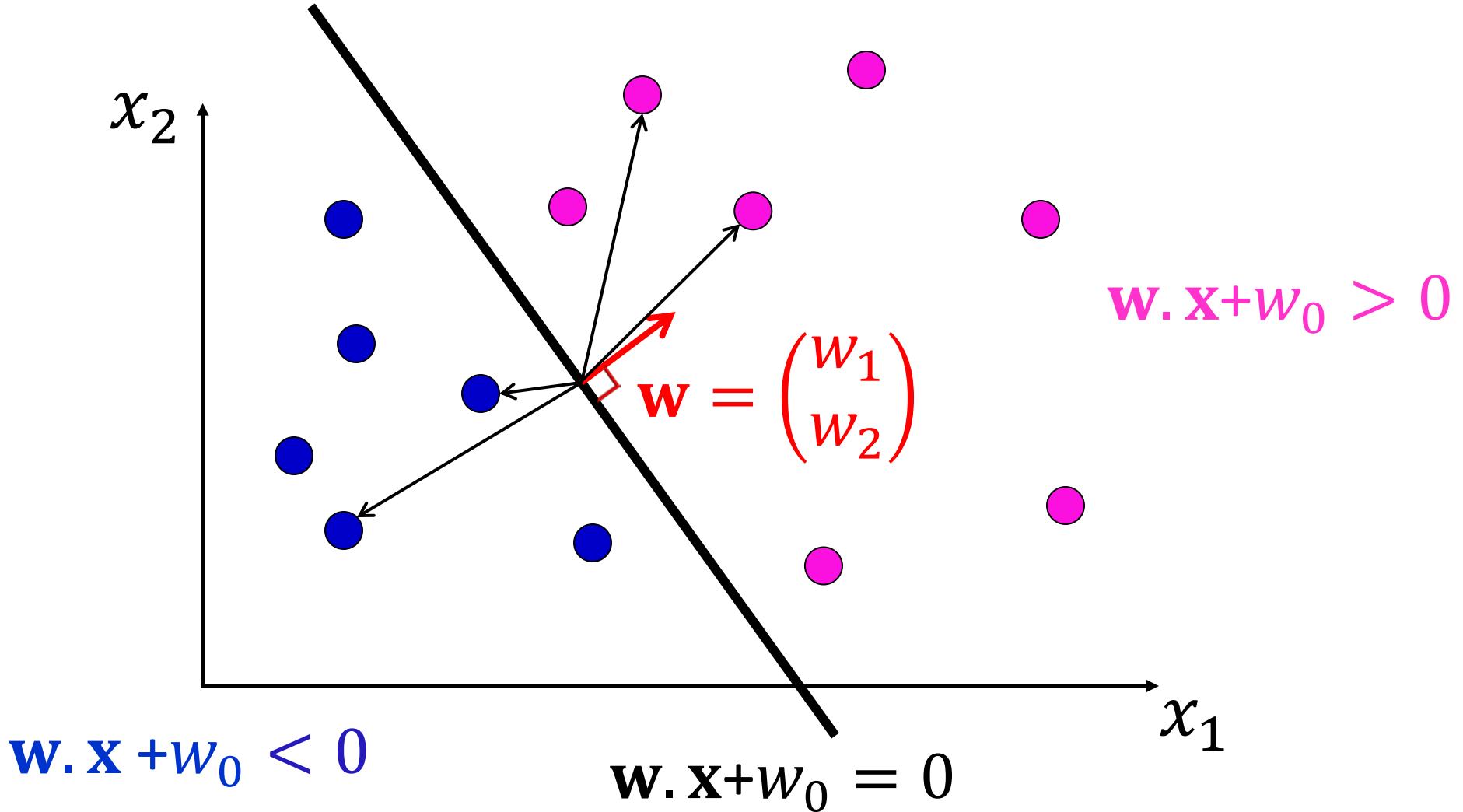
$$w_1x_1 + w_2x_2 + c < 0$$



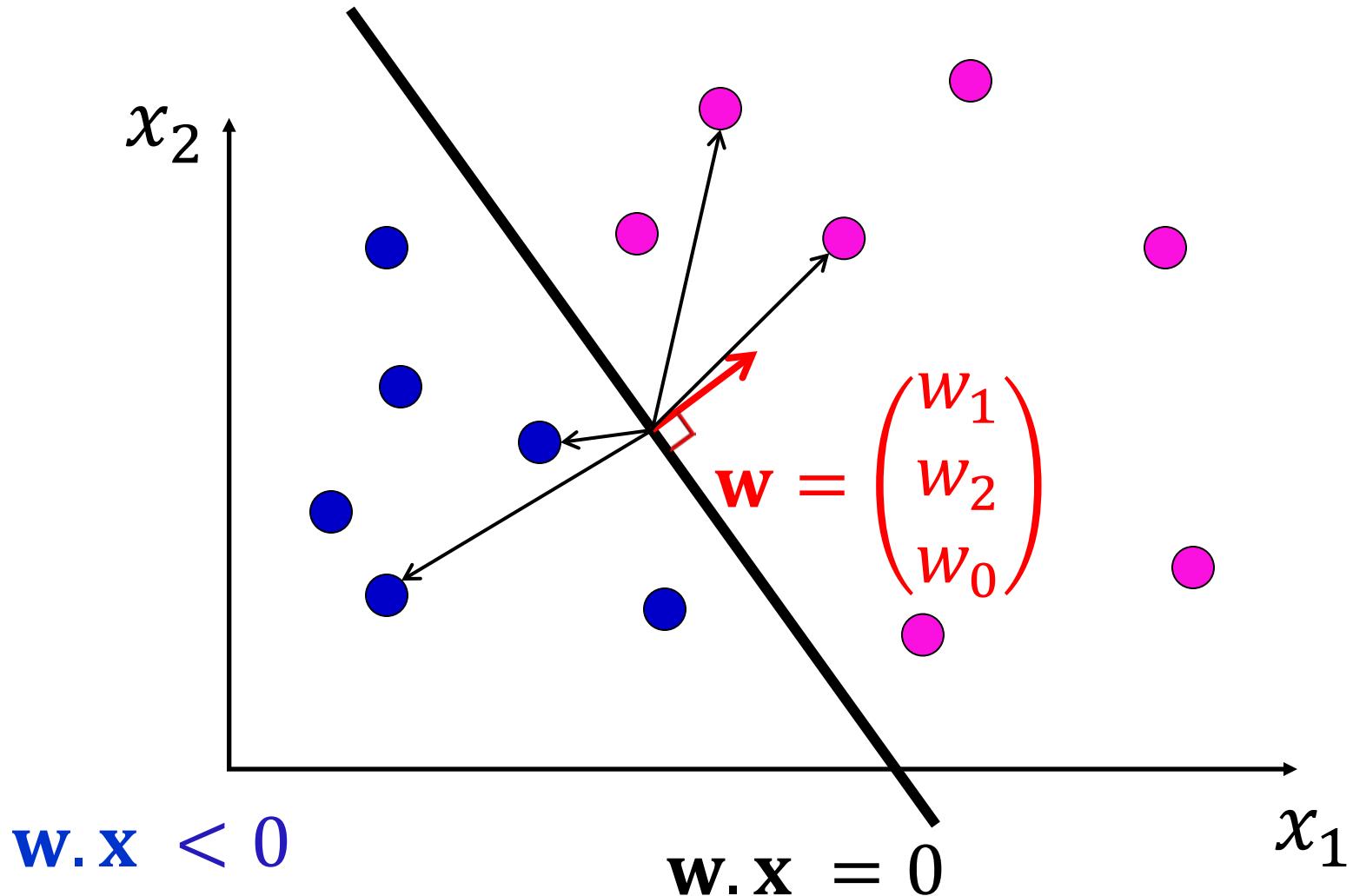
Geometry and Algebra



Geometry and Algebra

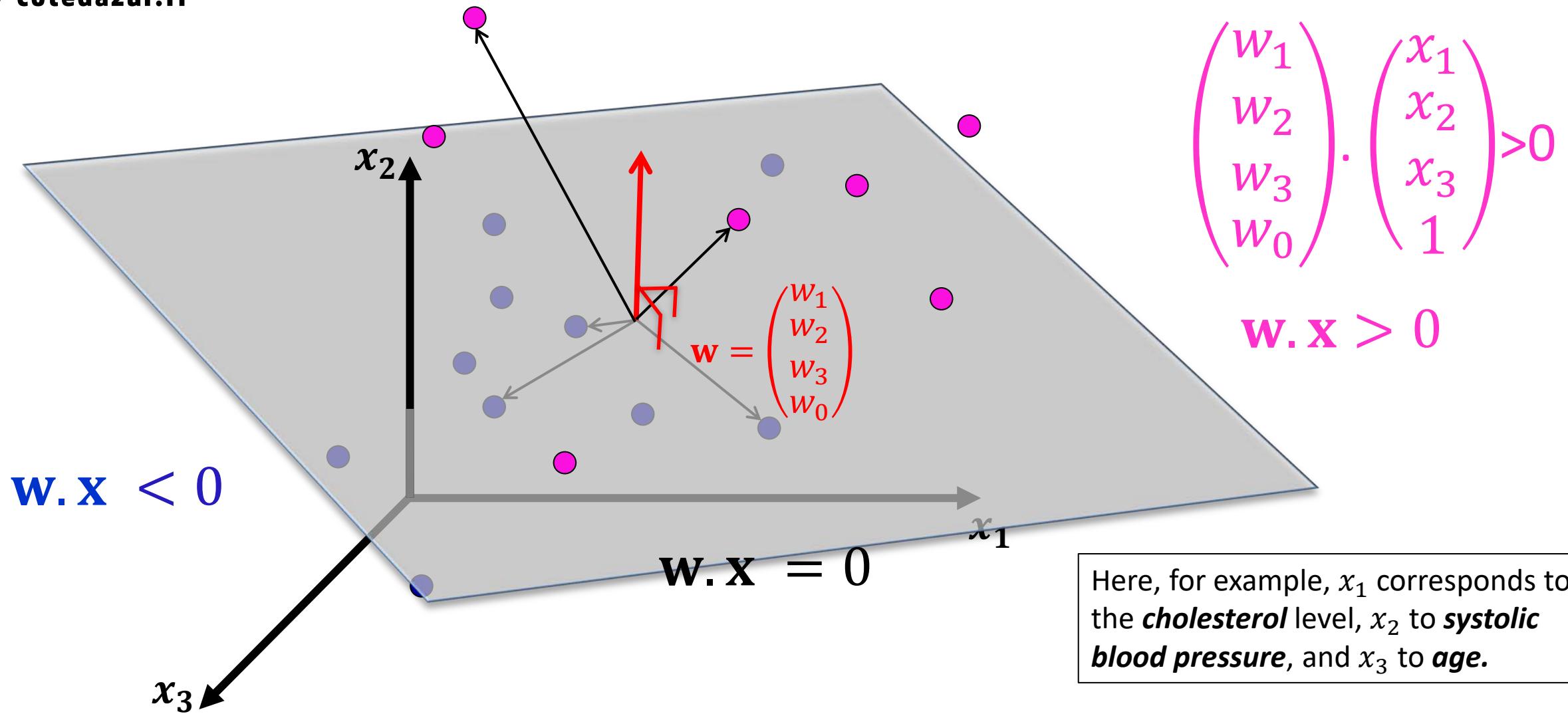


Simplification



$$\begin{pmatrix} w_1 \\ w_2 \\ w_0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} > 0$$
$$w \cdot x > 0$$

Geometry and Algebra



Derivative

- Derivative of a scalar function (of one variable)
 - $(ax)' = a$
 - $(ax + b)' = a$
 - $(g(f(x)))' = g'(f(x))f'(x)$ **Chain rule**

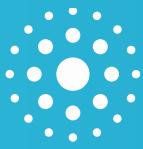
Gradient operator

- Gradient of a multivariate function (\mathbf{x} is a vector)
 - $\nabla_{\mathbf{x}}(\mathbf{a}\mathbf{x}) = \mathbf{a}$
 - $\nabla_{\mathbf{x}}(\mathbf{a}\mathbf{x} + b) = \mathbf{a}$
 - $\nabla_{\mathbf{x}}(g(f(\mathbf{x}))) = \nabla_{\mathbf{x}}g(f(\mathbf{x})) \nabla_{\mathbf{x}}f(\mathbf{x})$ **(Chain rule)**
 - $\nabla_{\mathbf{x}_i}(\mathbf{v} \cdot \mathbf{x}) = \nabla_{\mathbf{x}_i}(v_1 \cdot x_1 + v_2 \cdot x_2 + \cdots + v_n \cdot x_n) = v_i$



Overview

- Machine Learning vs Statistics
- Math Basics
- **Simple Model**
- From Simple to Complex



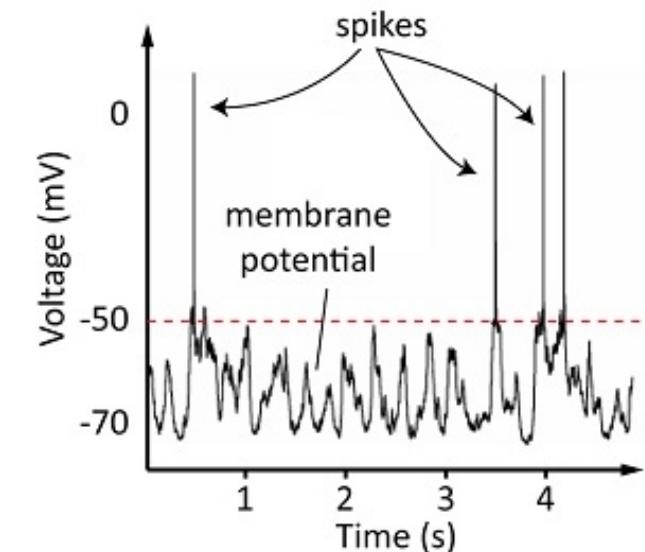
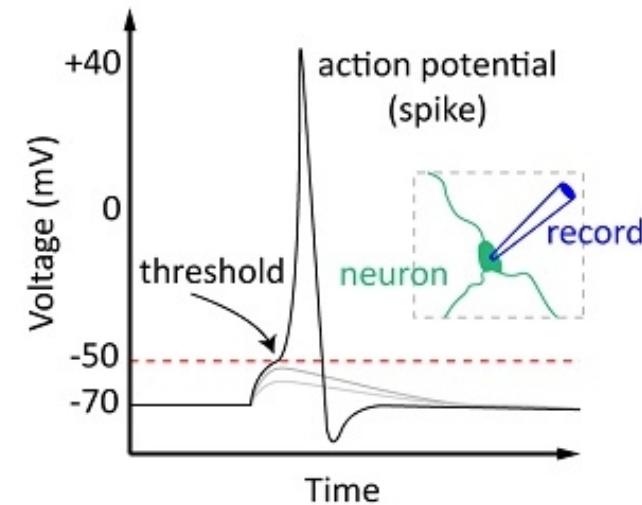
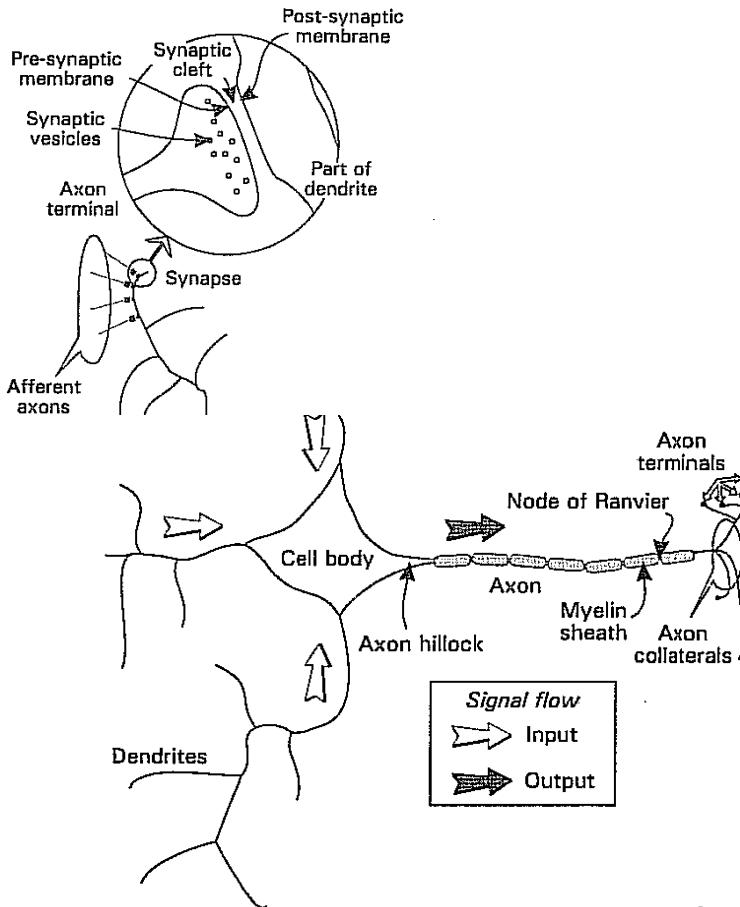
SIMPLE MODEL



Initial Model: Perceptron

Biological neuron

- Before we study artificial neurons, let's look at a biological neuron



First, biological neurons

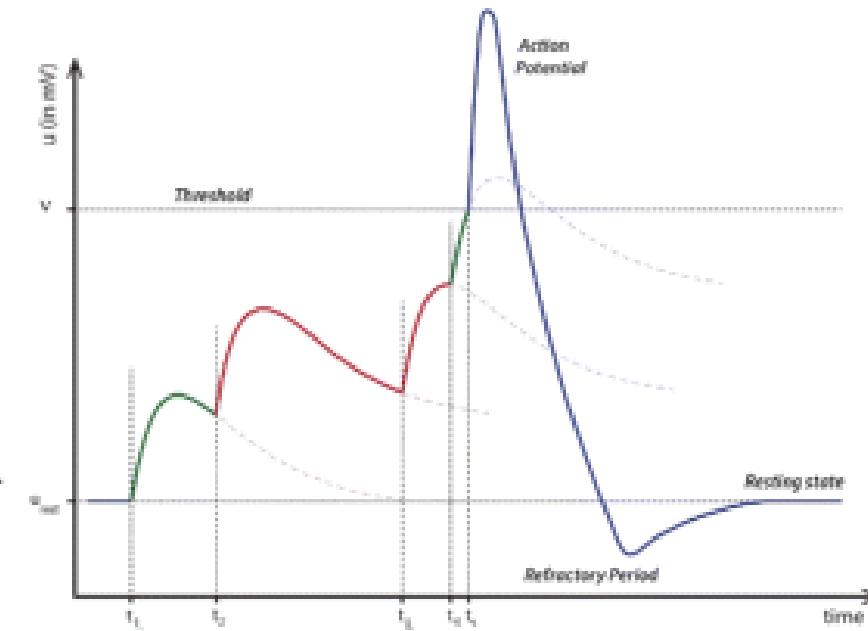
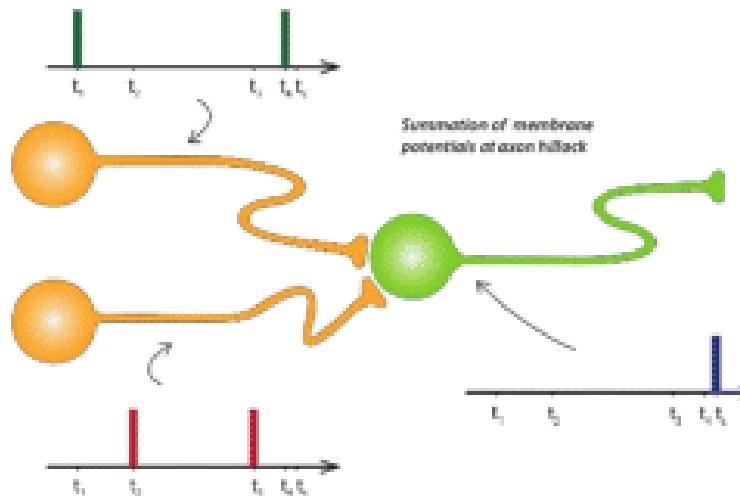
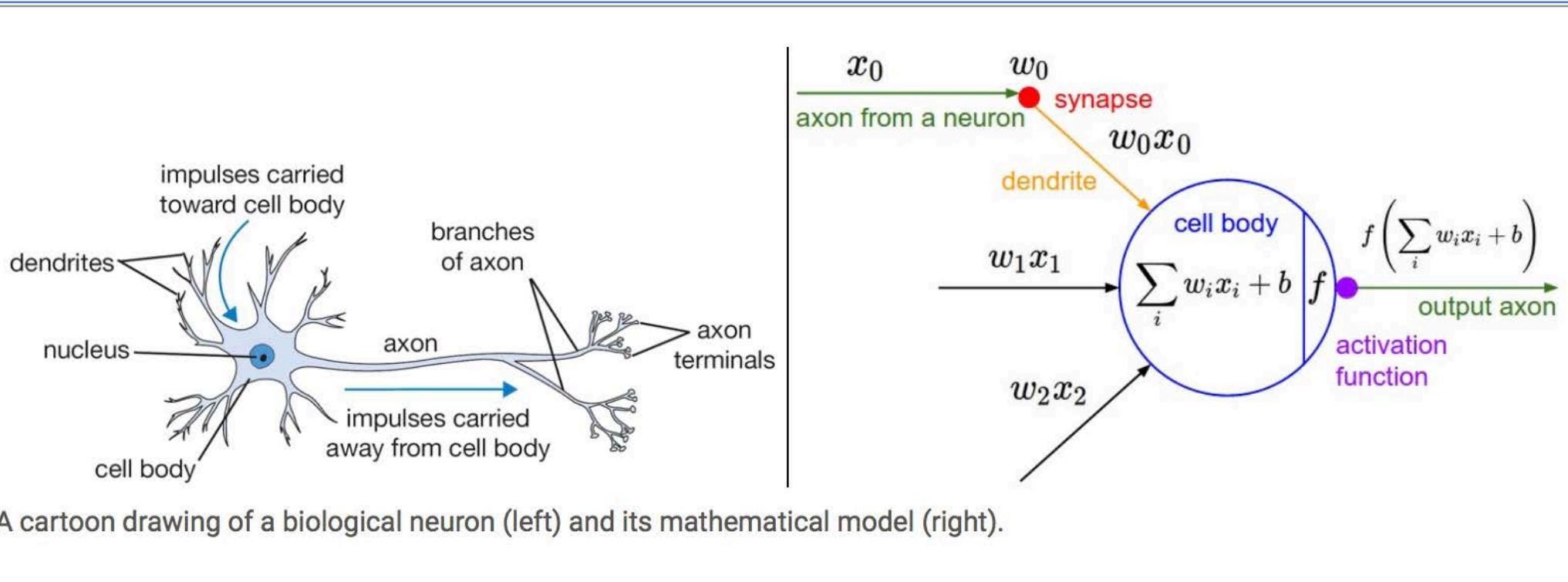


Figure from: Iakymchuk, T., et al. "Simplified spiking neural network architecture and STDP learning algorithm applied to image classification". In Journal of Image Video Proc. 2015, 4 (2015).

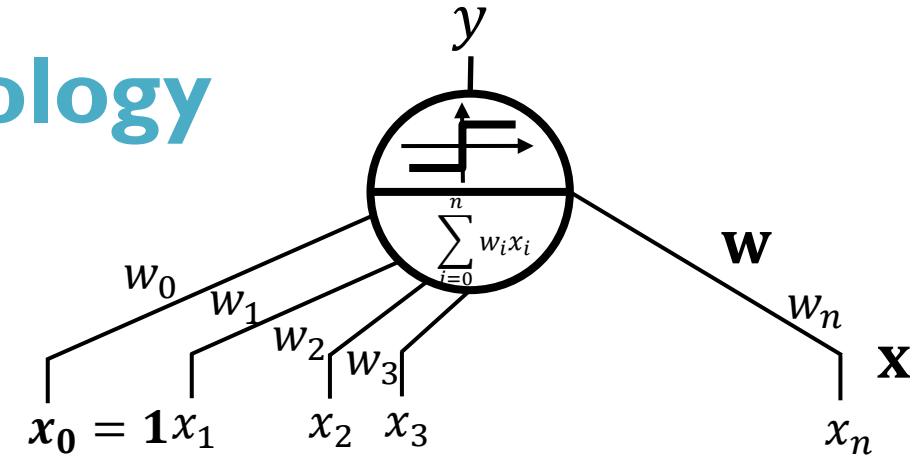
Then, artificial neurons



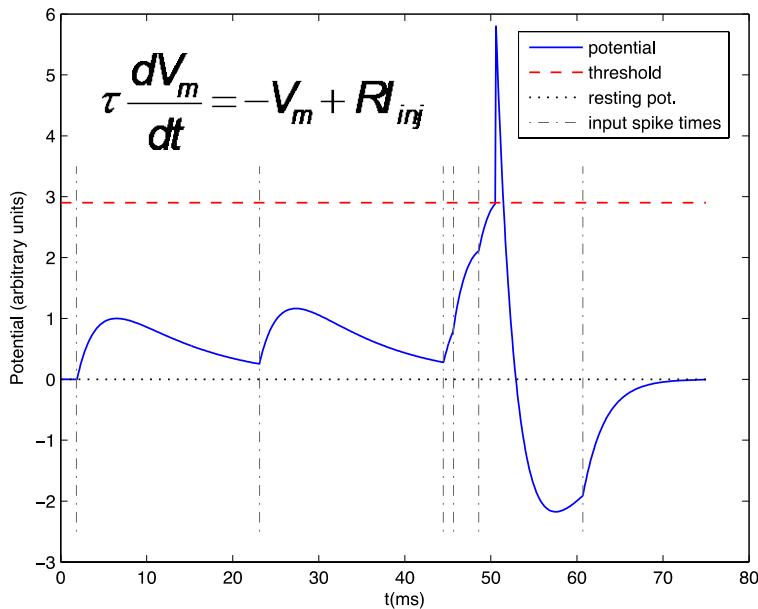
Pitts & McCulloch (1943), binary inputs & activation function f thresholding

Rosenblatt (1956), real inputs & activation function f thresholding

Artificial vs biology

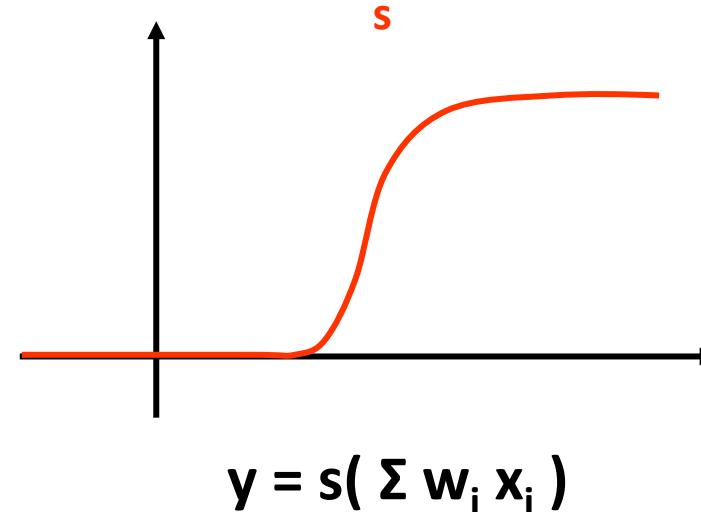


Spike-based description



Gradient descent: KO

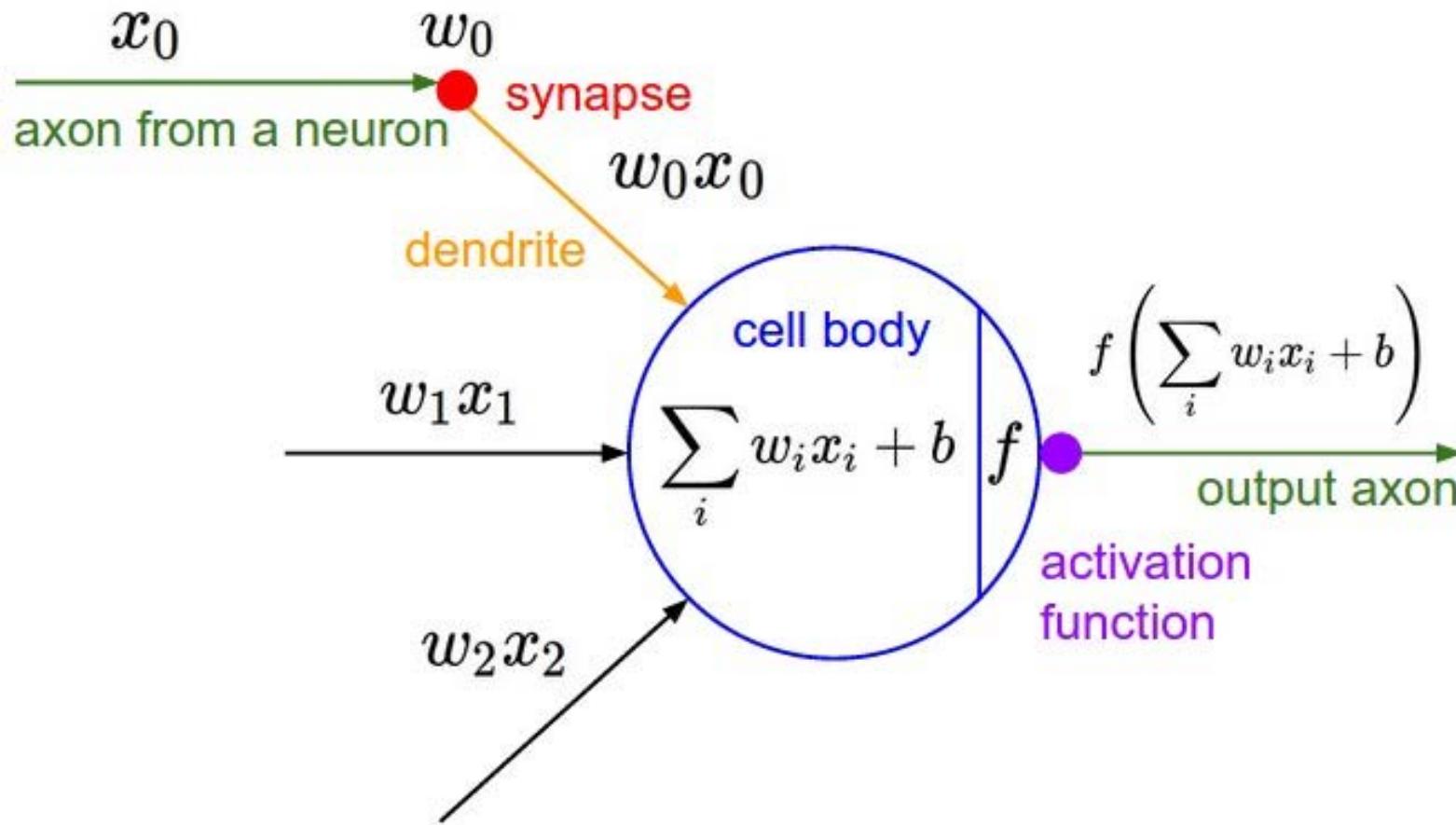
Rate-based description *Steady regime*



Gradient descent: OK

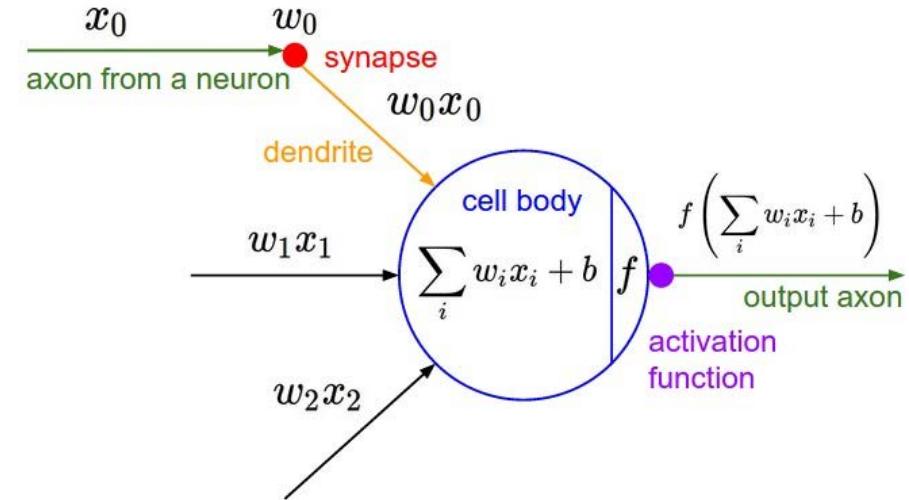
A single artificial neuron

- It is a **very simple abstract** of a biological neuron (McCulloch & Pitts, 1943)



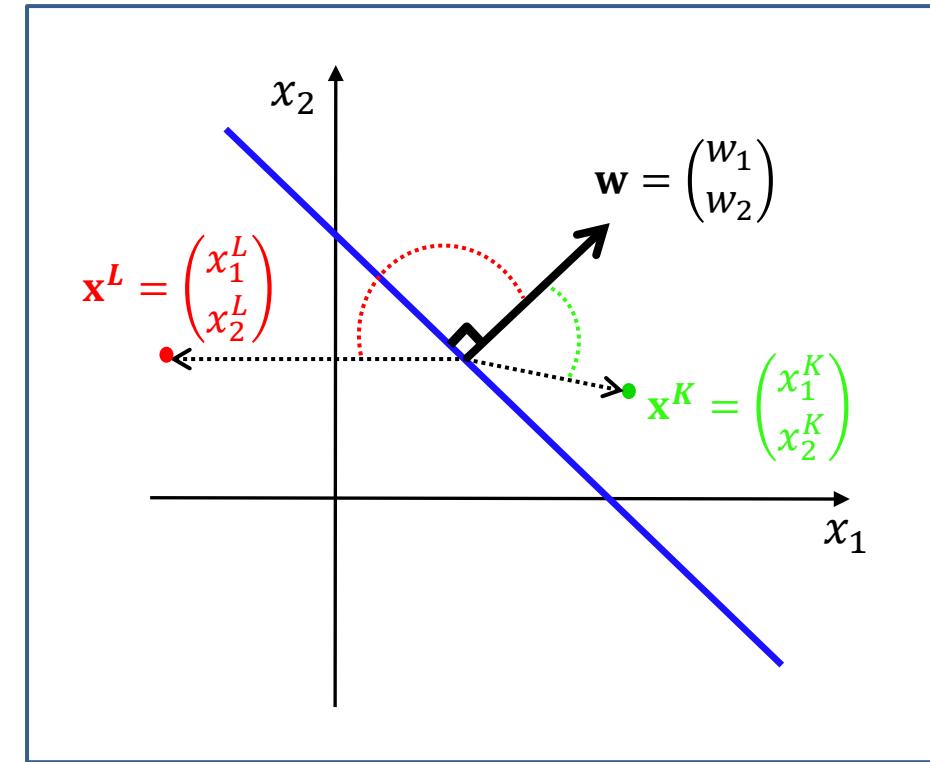
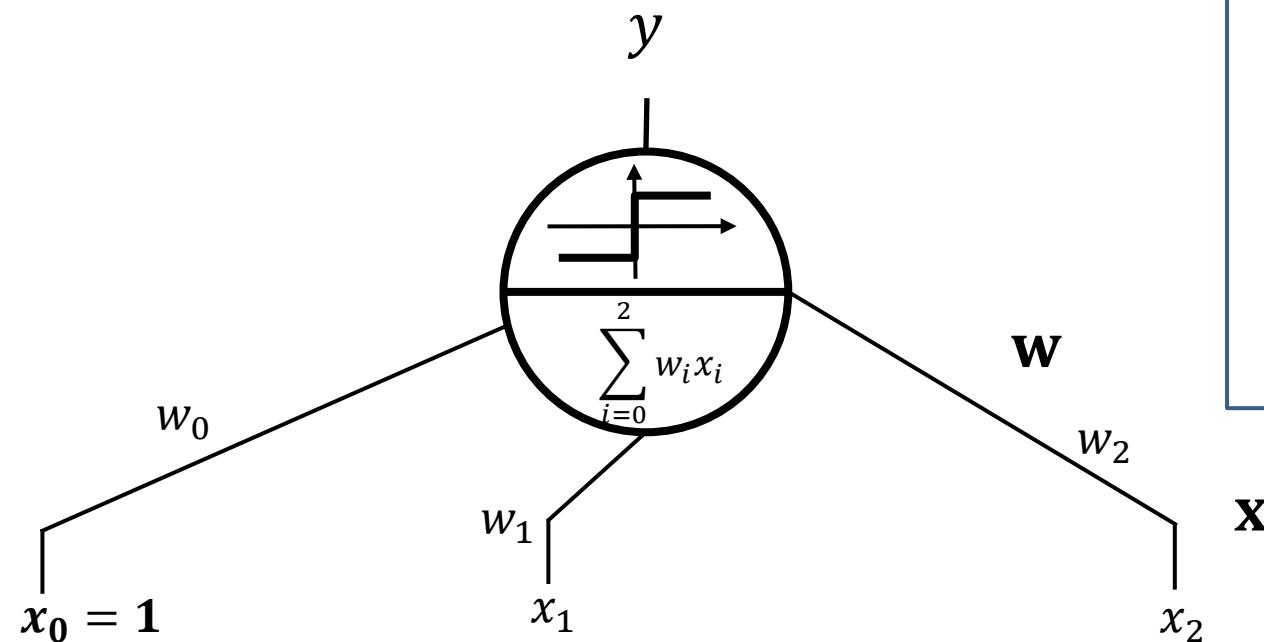
A single artificial neuron

- Each input x has an associated **weight w** which can be modified
- Inputs x corresponds to signals from other neuron axons
 - x_0 - Bias are ‘special’ inputs, with weight w_0
- Weights **W** corresponds to synaptic modulation (i.e. something like strength/amount of neurotransmitters)
- The summation corresponds to ‘cell body’
- The activation function corresponds to axon hillock - computes some function f of the weighted sum of its inputs
- So, output $y=f(z)$, corresponds to axon signal



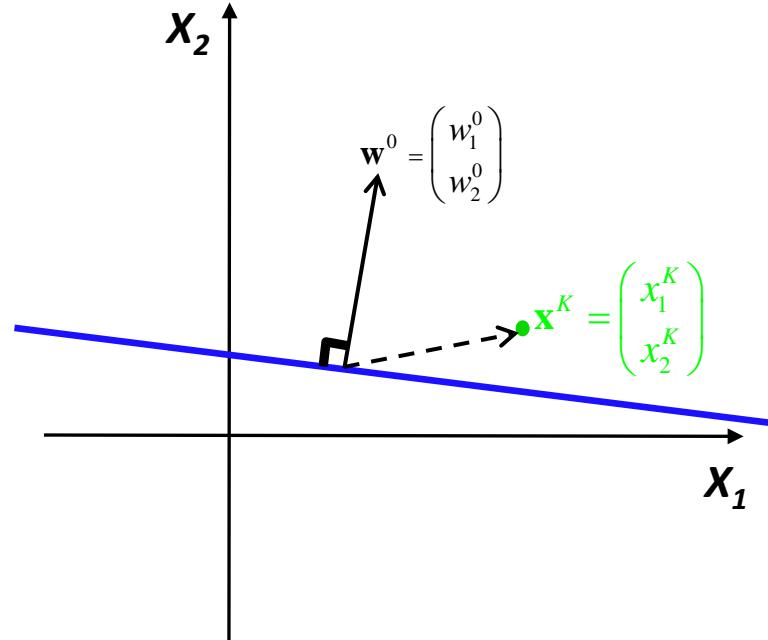
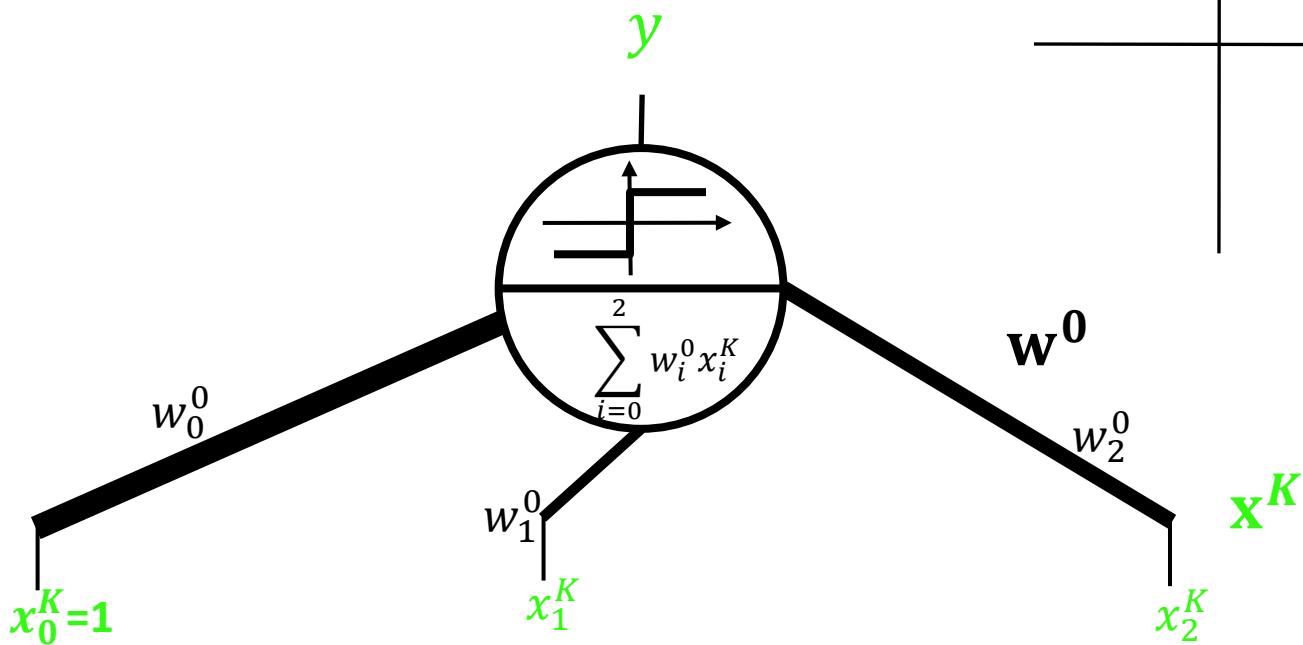


Artificial neuron = a linear classifier



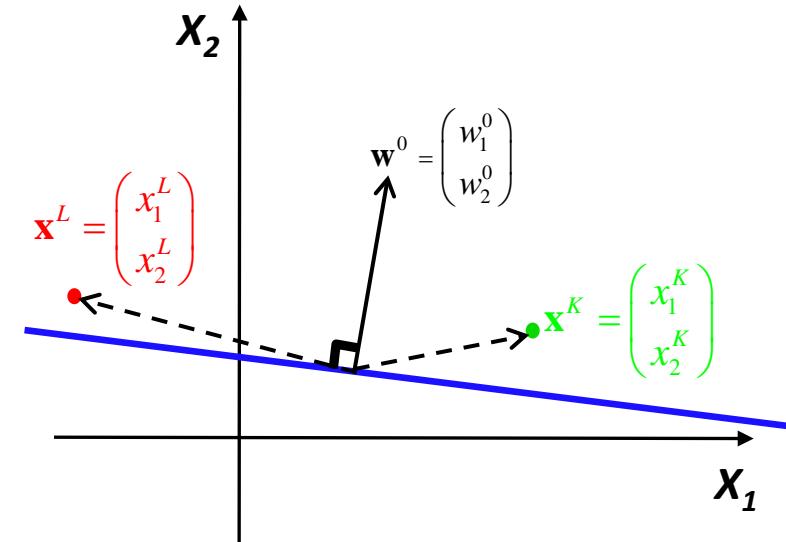
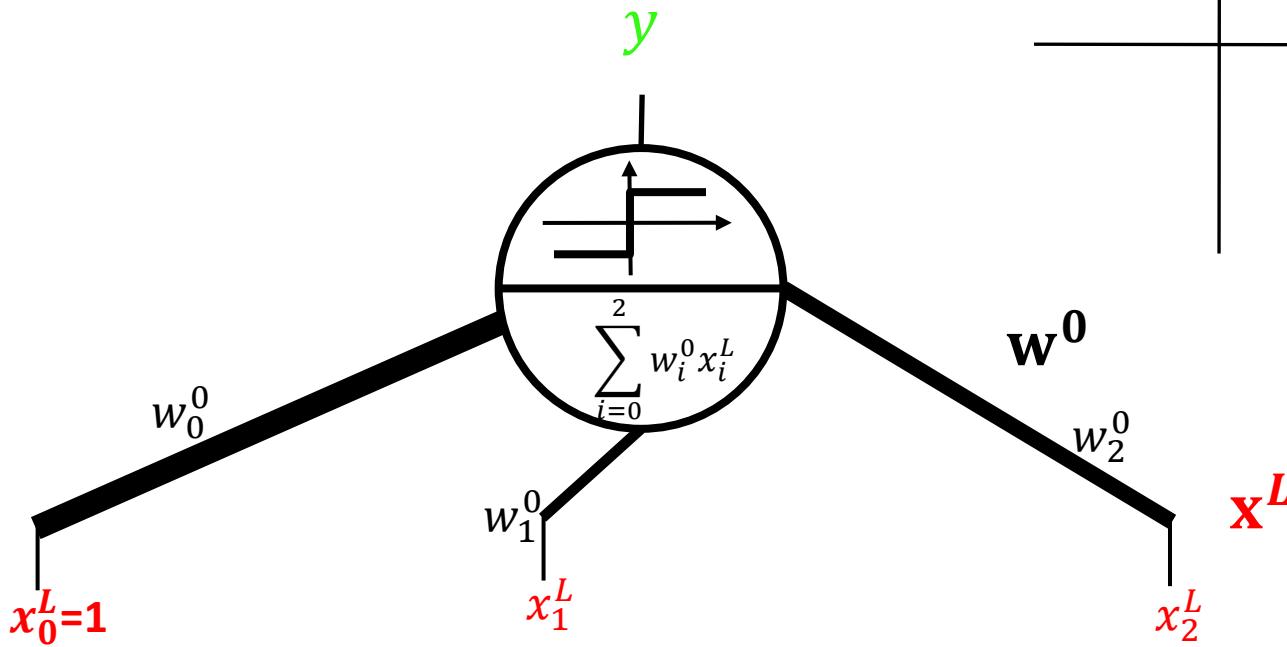
One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



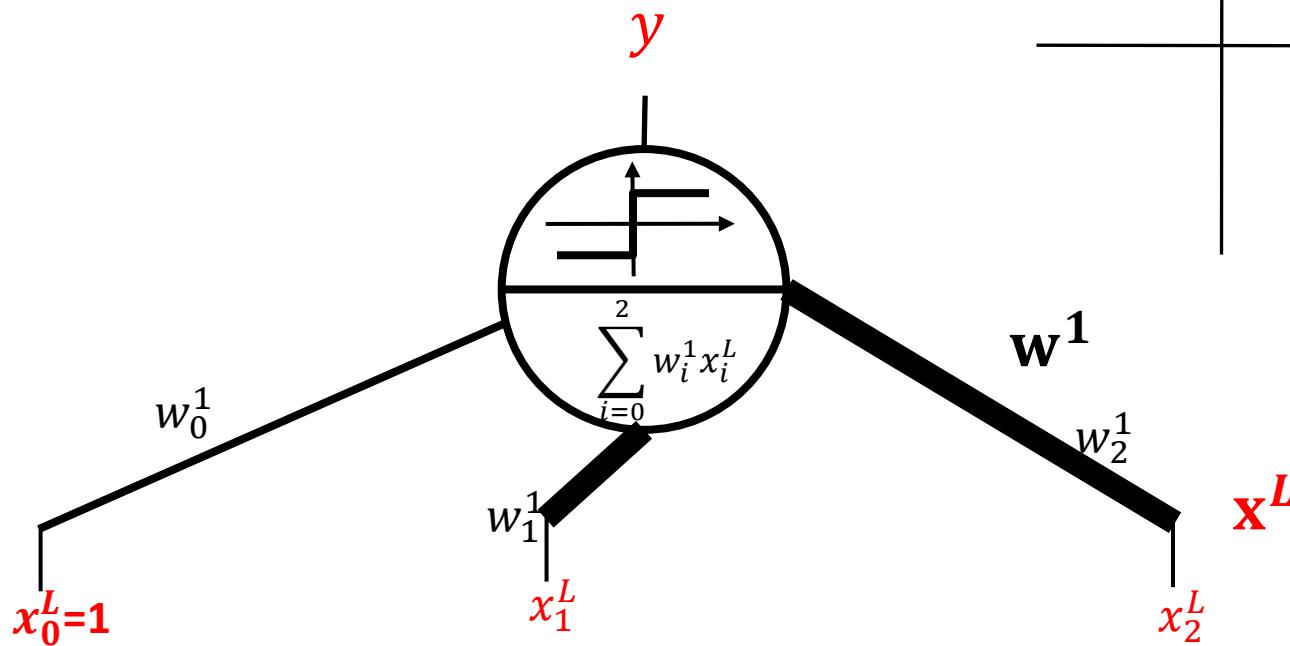
One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



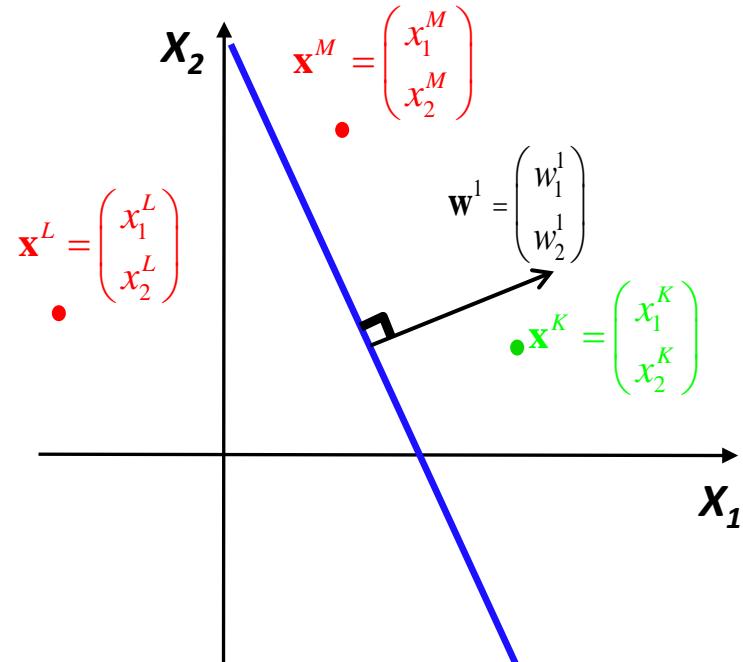
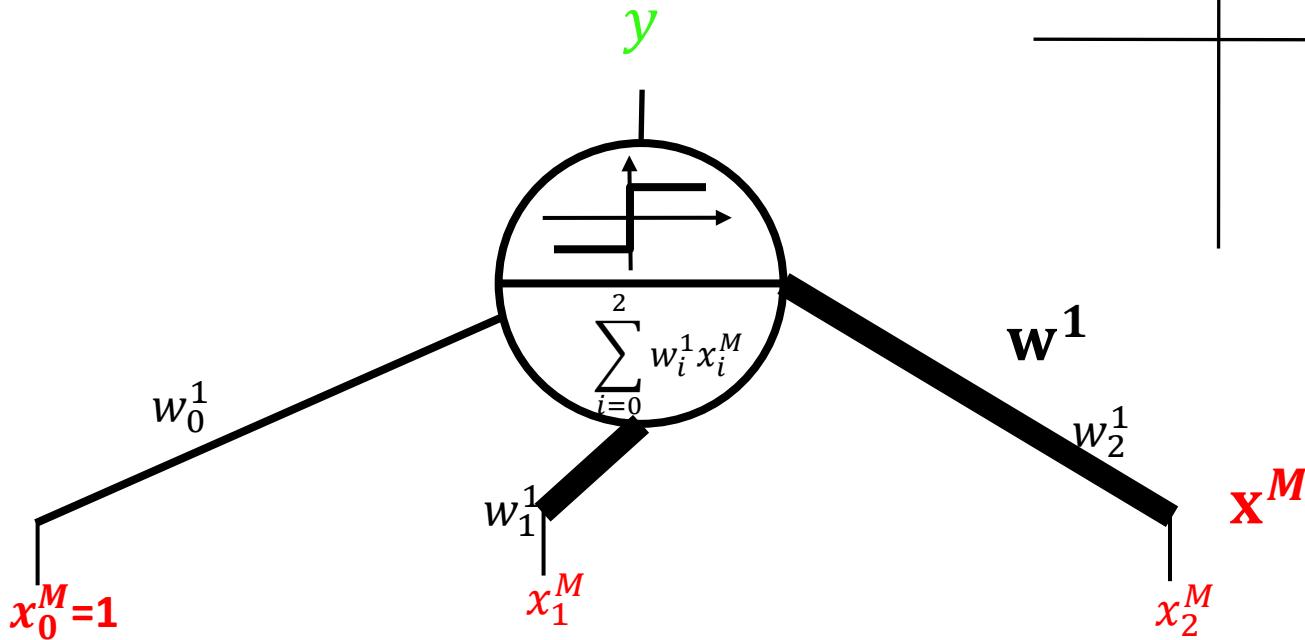
One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



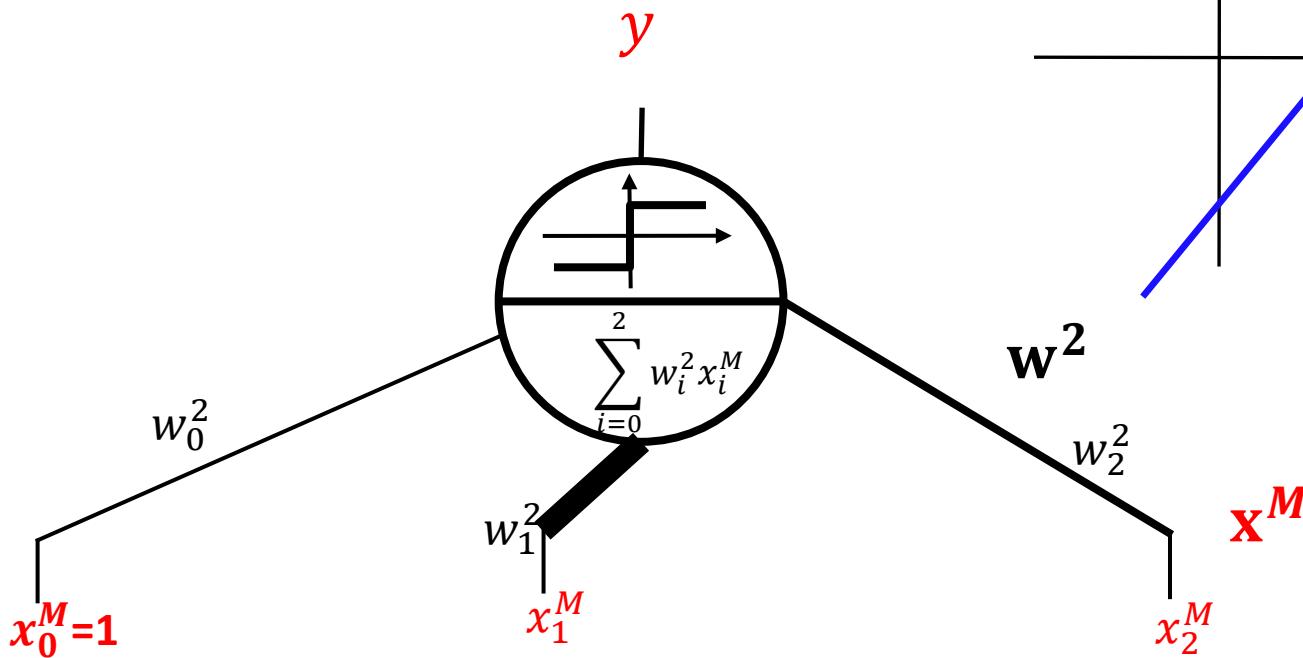
One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



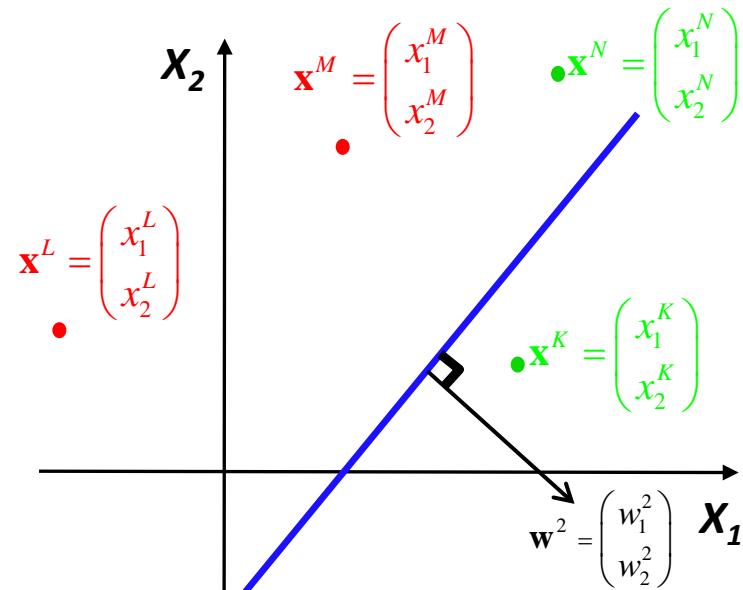
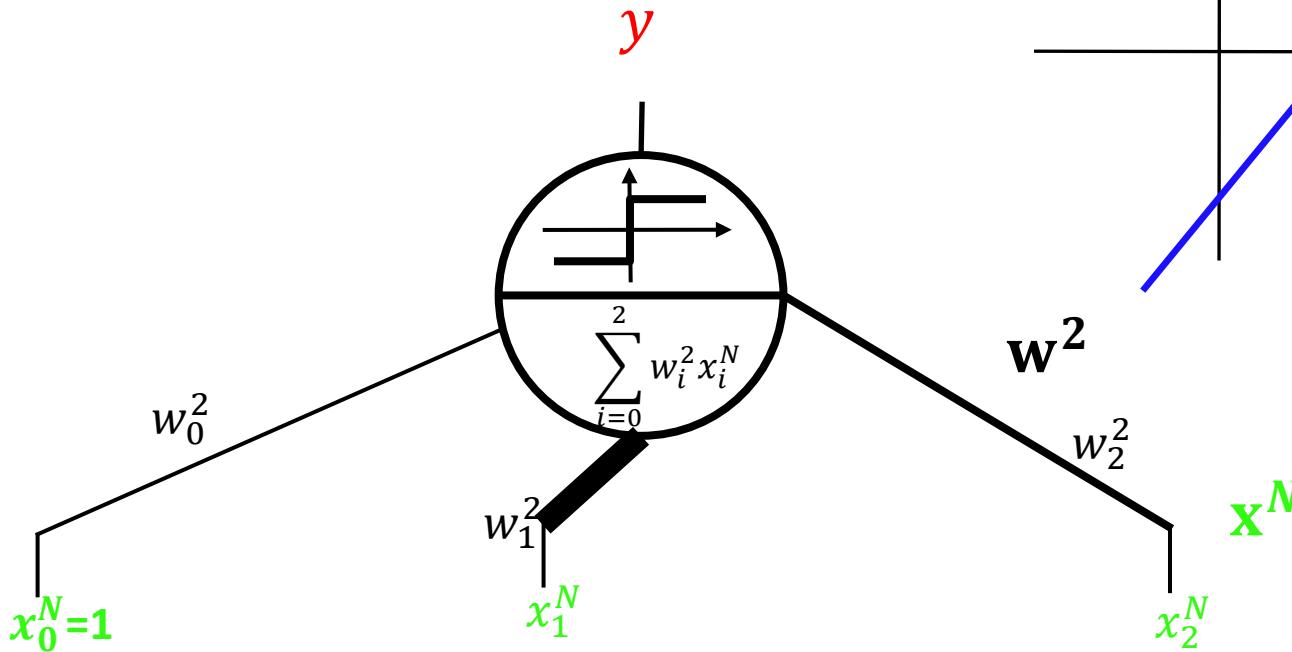
One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



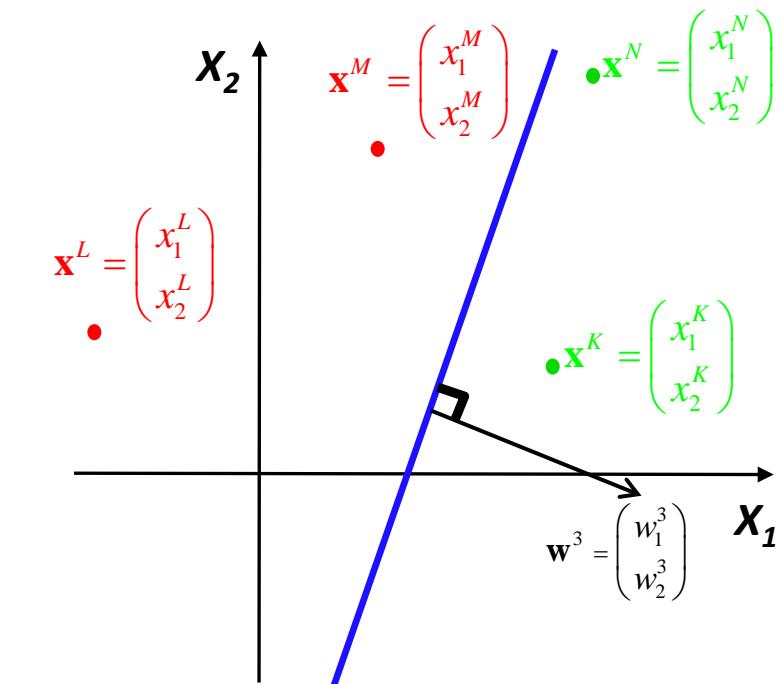
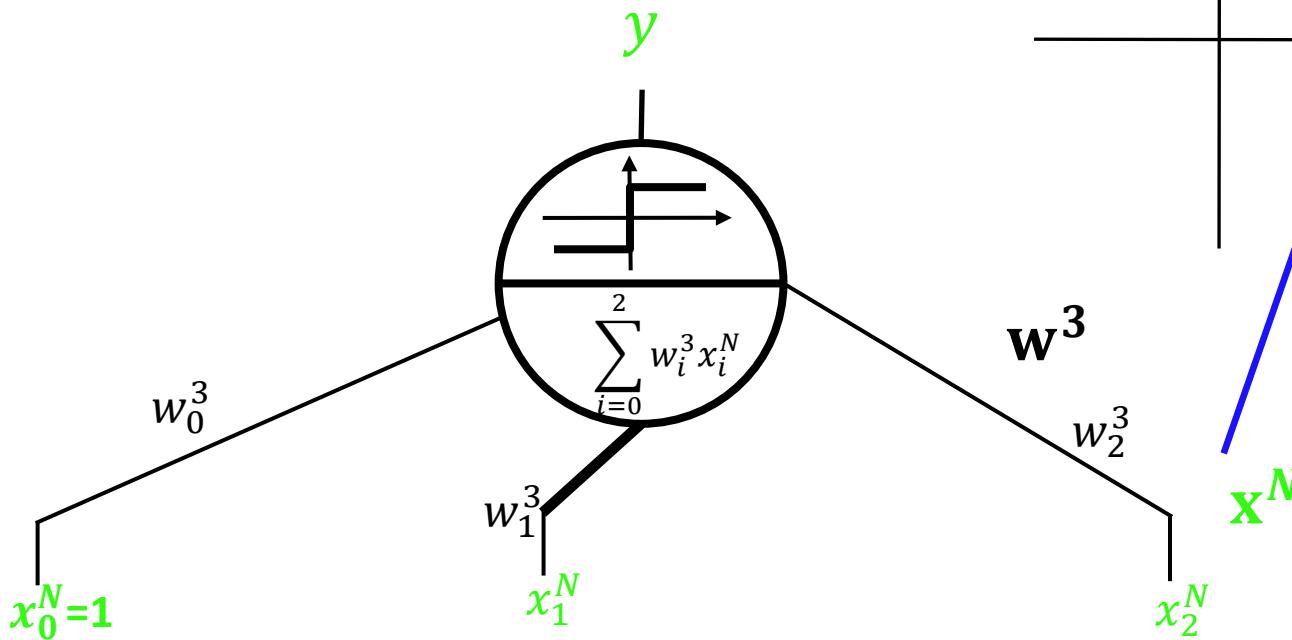
One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



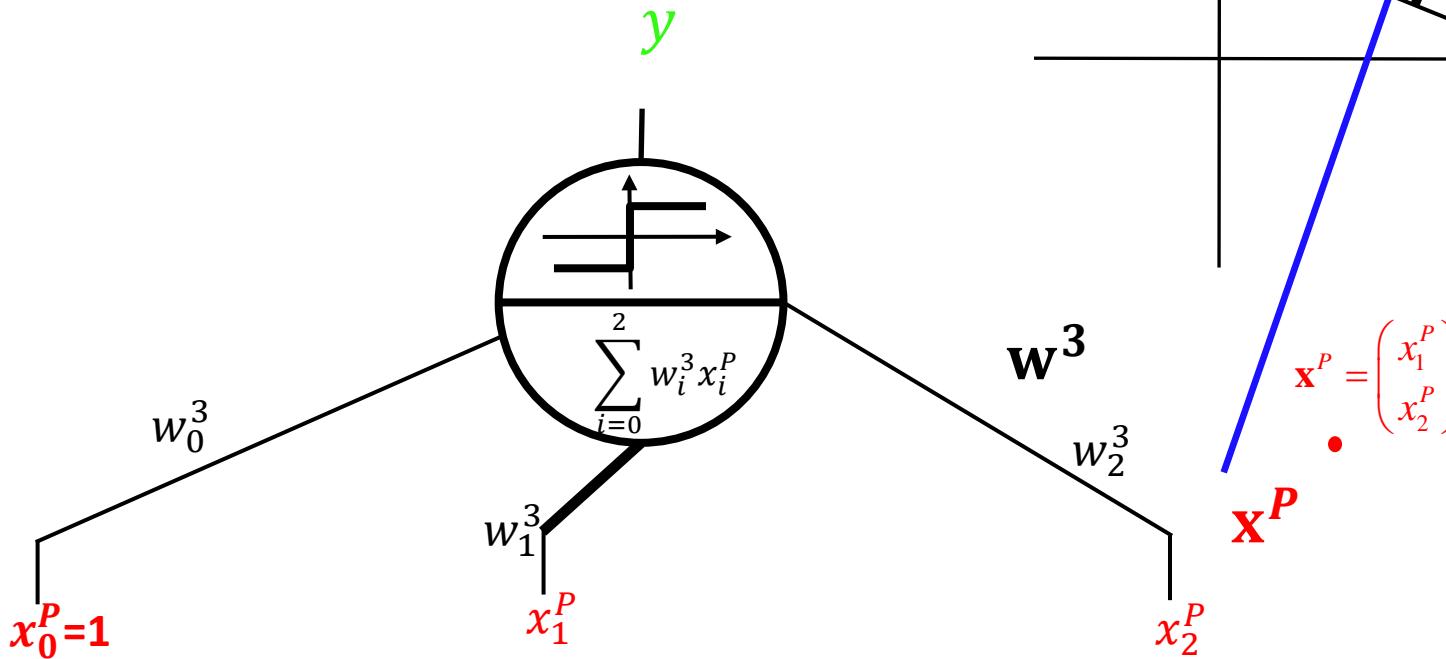
One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:

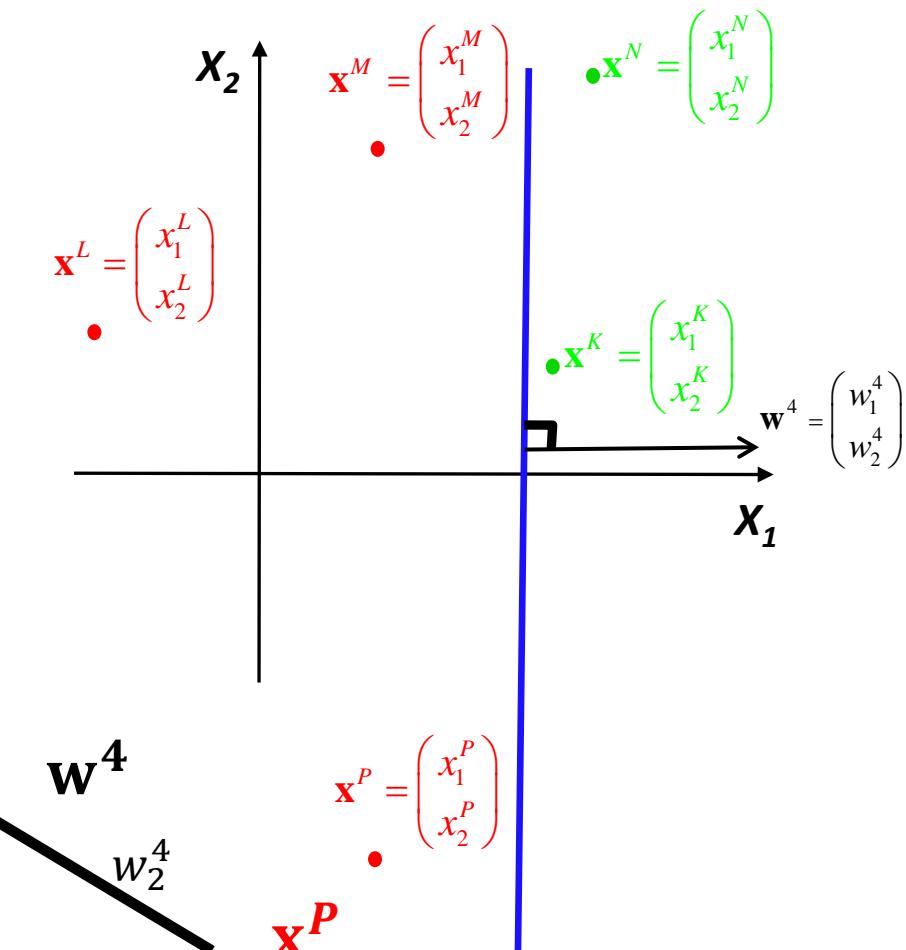
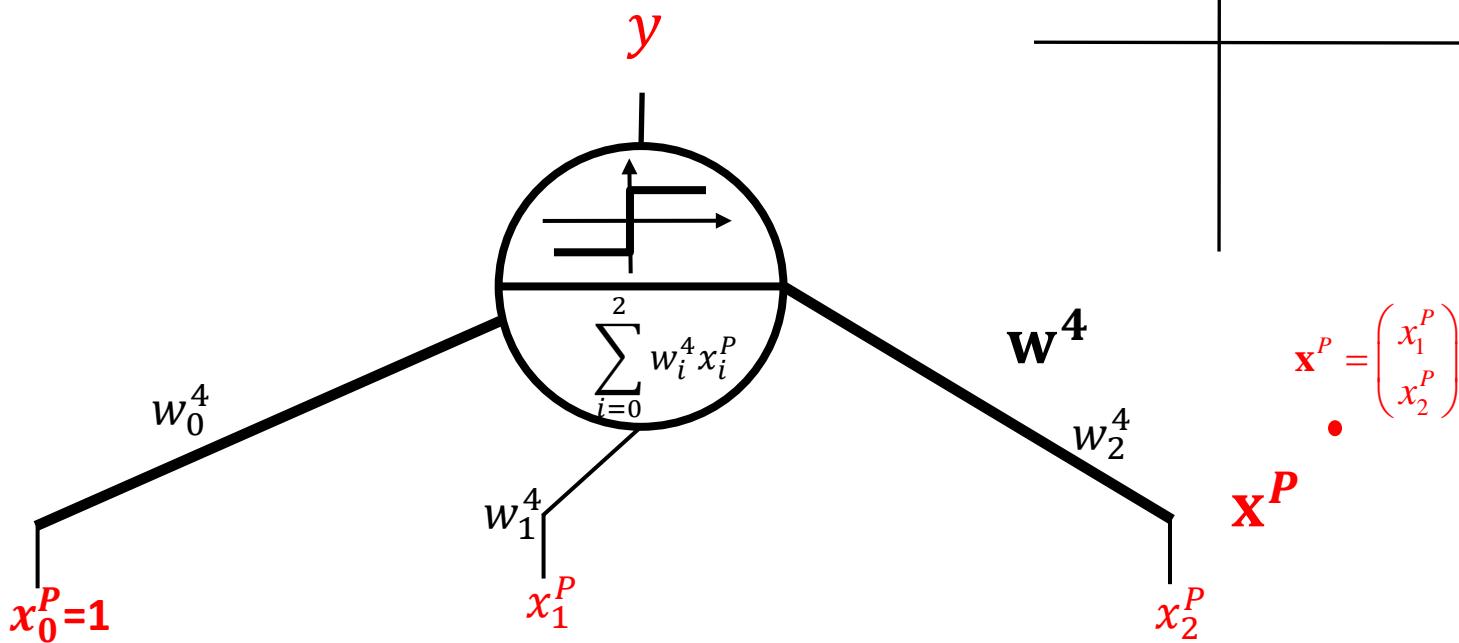


$$\mathbf{x}^M = \begin{pmatrix} x_1^M \\ x_2^M \end{pmatrix} \quad \bullet \quad \mathbf{x}^N = \begin{pmatrix} x_1^N \\ x_2^N \end{pmatrix}$$
$$\mathbf{x}^L = \begin{pmatrix} x_1^L \\ x_2^L \end{pmatrix}$$
$$\mathbf{x}^K = \begin{pmatrix} x_1^K \\ x_2^K \end{pmatrix}$$
$$\mathbf{w}^3 = \begin{pmatrix} w_1^3 \\ w_2^3 \end{pmatrix}$$

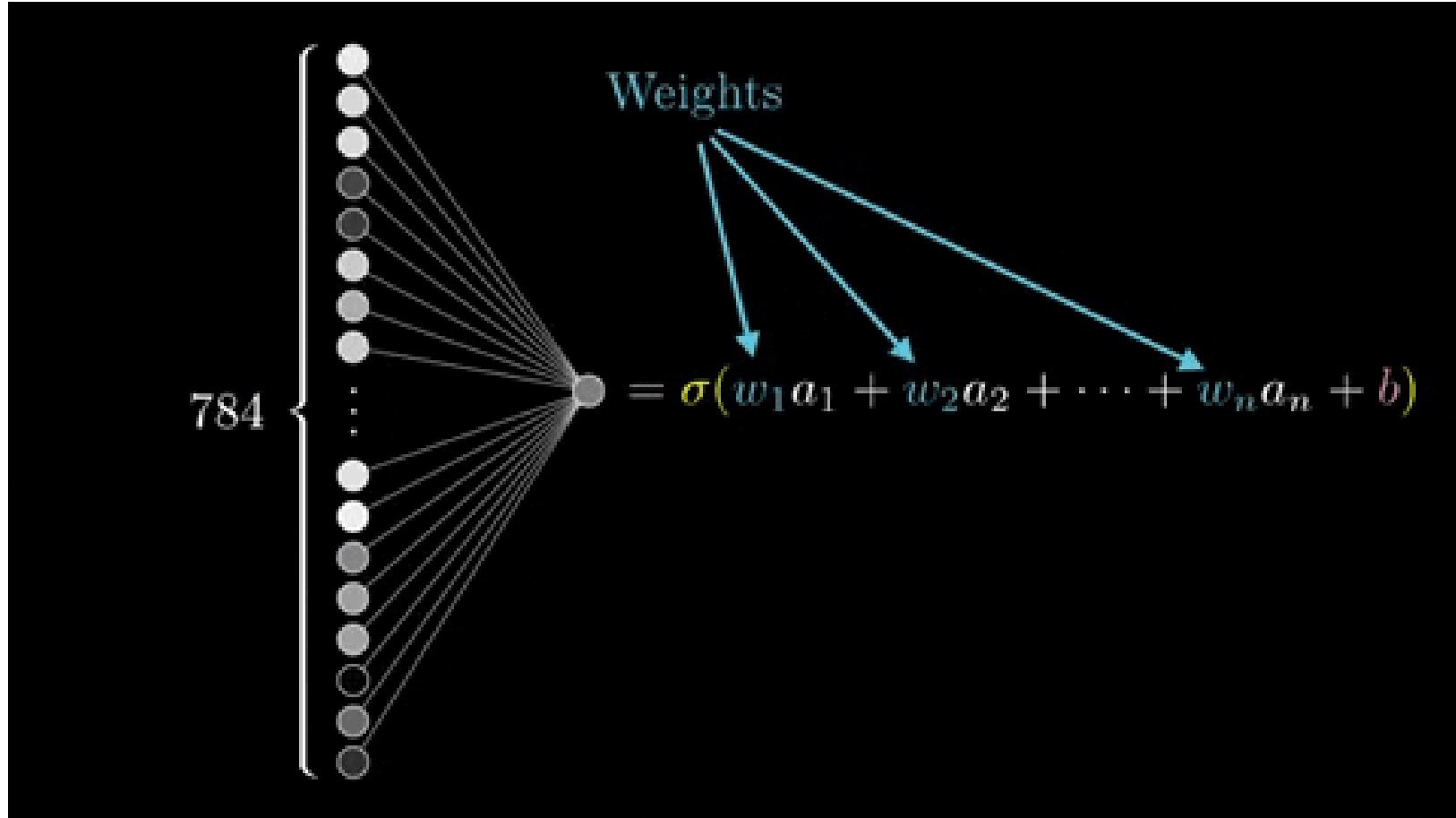
$$\mathbf{x}^P = \begin{pmatrix} x_1^P \\ x_2^P \end{pmatrix}$$

One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



One neuron = simple linear decision





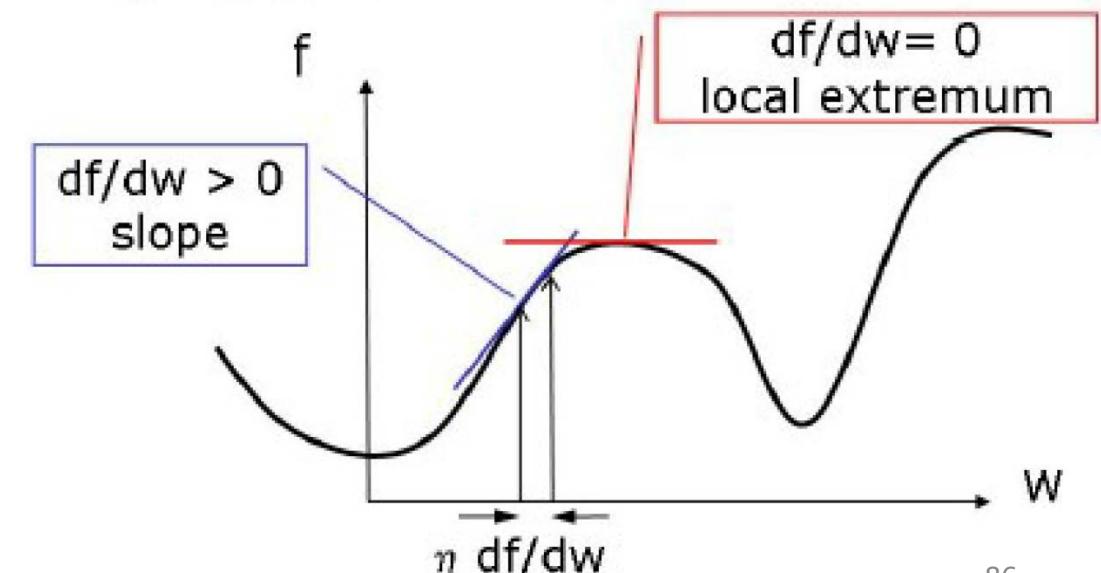
Perceptron: Rosenblatt's Algorithm (1956-1958)

Perceptron Algorithm

- Pick initial weight vector (including w_0), e.g. (0, 0,...,0)
- Repeat until all points are correctly classified
 - *Repeat for each point*
 - Calculate $y^i \mathbf{w} \mathbf{x}^i$ for point i
 - If $y^i \mathbf{w} \mathbf{x}^i > 0$, the point is correctly classified
 - Else change the weights to increase the value of $y^i \mathbf{w} \mathbf{x}^i$; change in weight proportional to $y^i \mathbf{x}^i$

Gradient Ascent

- Why pick $y^i \mathbf{x}^i$ as increment to weights?
- To maximize scalar function of one variable $f(\mathbf{w})$
 - Pick initial \mathbf{w}
 - Change \mathbf{w} to $\mathbf{w} + \eta \frac{df}{d\mathbf{w}}$ ($\eta > 0$, small)
 - Until f stops changing ($\frac{df}{d\mathbf{w}} \approx 0$)



Gradient Ascent

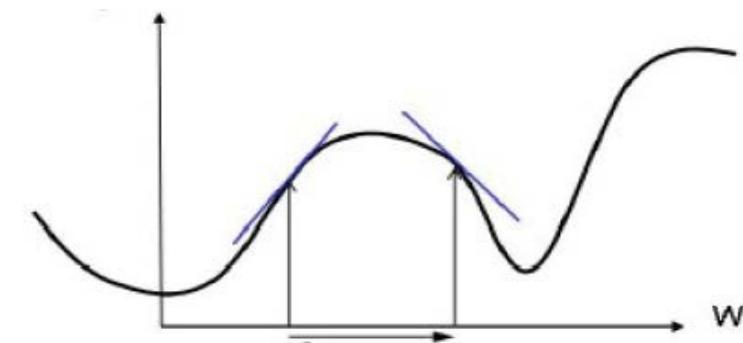
- To maximize a multivariate function $f(\mathbf{w})$
 - Pick initial \mathbf{w}
 - Change \mathbf{w} to $\mathbf{w} + \eta \nabla f_{\mathbf{w}}$ ($\eta > 0$, small)
 - Until f stops changing ($\nabla f_{\mathbf{w}} \approx 0$)
- Find local maximum, unless function is globally convex

$$\nabla f_{\mathbf{w}} = \left[\frac{\partial f}{\partial \mathbf{w}_1}, \quad \dots \quad , \frac{\partial f}{\partial \mathbf{w}_n} \right]$$

Gradient Ascent

- To maximize a multivariate function $f(\mathbf{w})$
 - Pick initial \mathbf{w}
 - Change \mathbf{w} to $\mathbf{w} + \eta \nabla f_{\mathbf{w}}$ ($\eta > 0$, small)
 - Until f stops changing ($\nabla f_{\mathbf{w}} \approx 0$)
- Find local maximum, unless function is globally convex
- If f is non-linear, the learning rate η has to be chosen very carefully
 - Too small \Rightarrow slow convergence
 - Too big \Rightarrow oscillations

$$\nabla f_{\mathbf{w}} = \left[\frac{\partial f}{\partial \mathbf{w}_1}, \dots, \frac{\partial f}{\partial \mathbf{w}_n} \right]$$



Gradient Ascent

- Maximize margin of misclassified points

$$f(\mathbf{w}) = \sum_{\text{on } \mathbf{i} \text{ misclassified points}} y^i \mathbf{w} \mathbf{x}^i$$

$$\nabla_{\mathbf{w}} f(\mathbf{w}) = \sum_{\text{on } \mathbf{i} \text{ misclassified points}} y^i \mathbf{x}^i$$

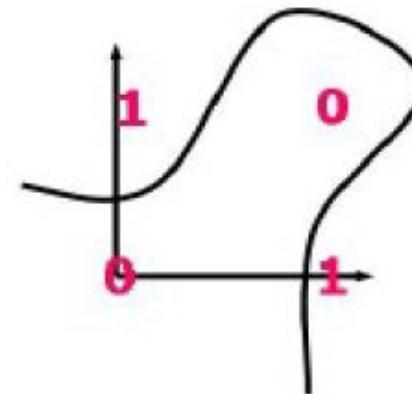
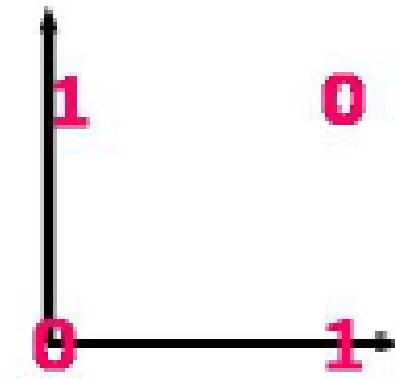
- Off-line training: Compute, at each iteration, the gradient as sum over all training points
- On-line training: Approximate gradient by one of the terms in the sum: $y^i \mathbf{x}^i$
(principle of the *Stochastic Gradient Descent*, SGD)

Perceptron Algorithm

- Each change of w decreases the error on a specific point. However, changes for several points are correlated, that is different points could change the weights in opposite directions. Thus, this iterative algorithm requires several loops to converge.
- Guarantee to find a separating hyperplane if one exists – if data is linearly separable.
- If data are not linearly separable, then this algorithm loops indefinitely.

Beyond Linear Separability

- Values of the XOR boolean function cannot be separated by a single perceptron unit [Minsky and Papert, 1969].

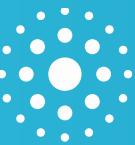


Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press.



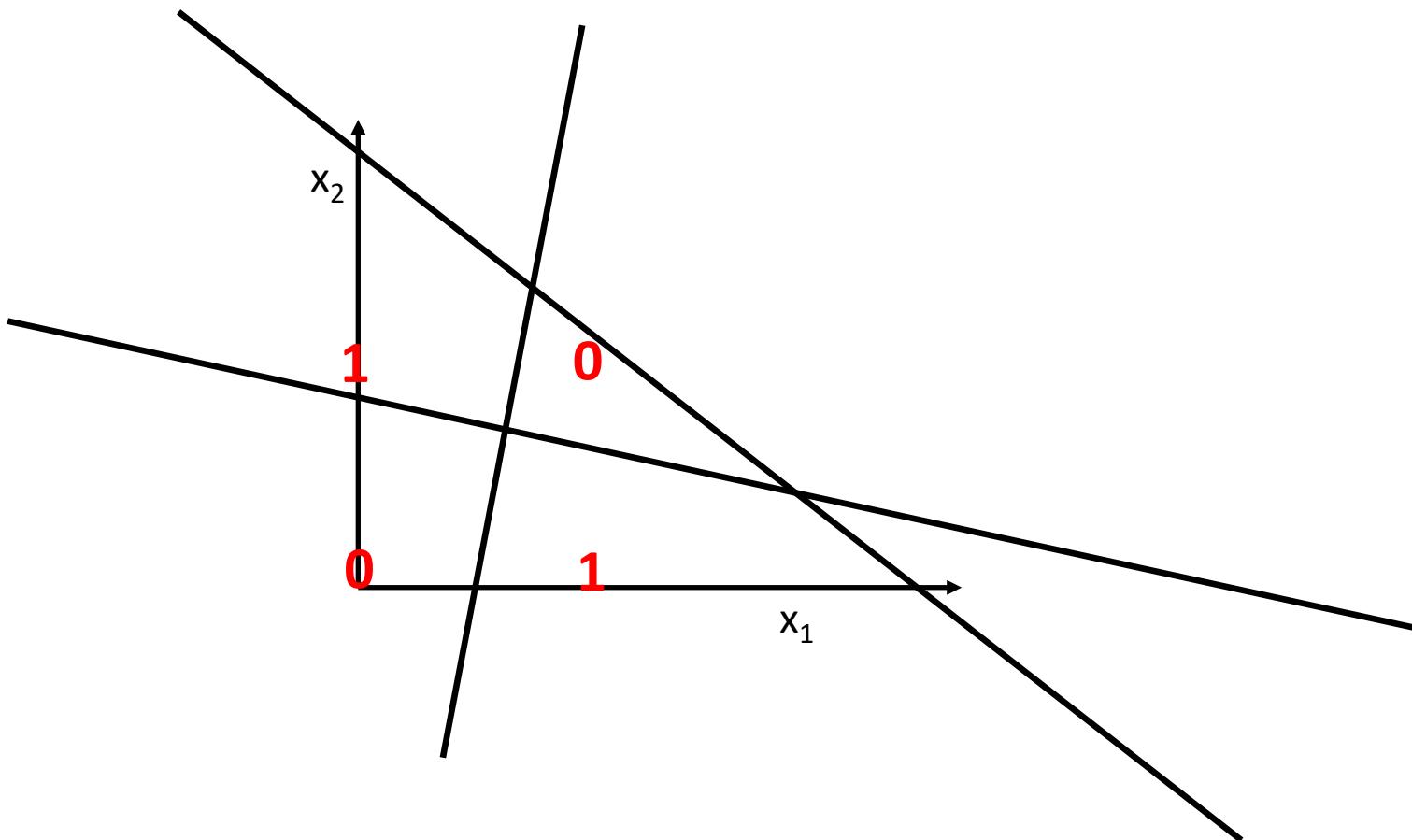
Overview

- Machine Learning vs Statistics
- Math Basics
- Simple Model
- **From Simple to Complex**

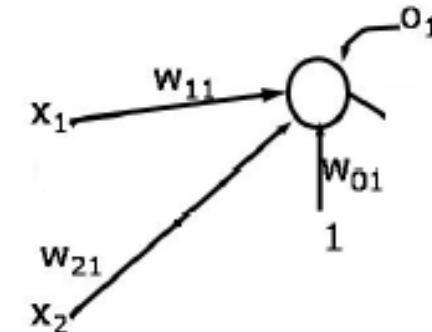
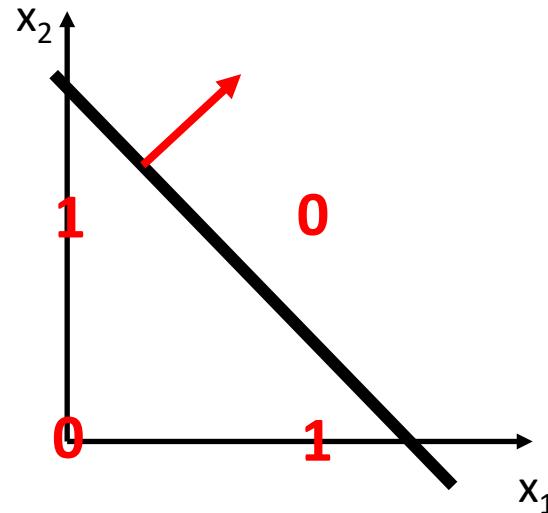


FROM SIMPLE TO COMPLEX

Problem which cannot be solved with a unique straight line

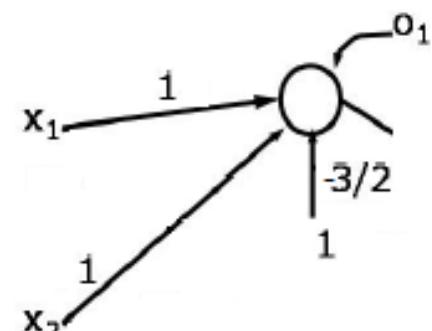


Problem which cannot be solved with a unique straight line



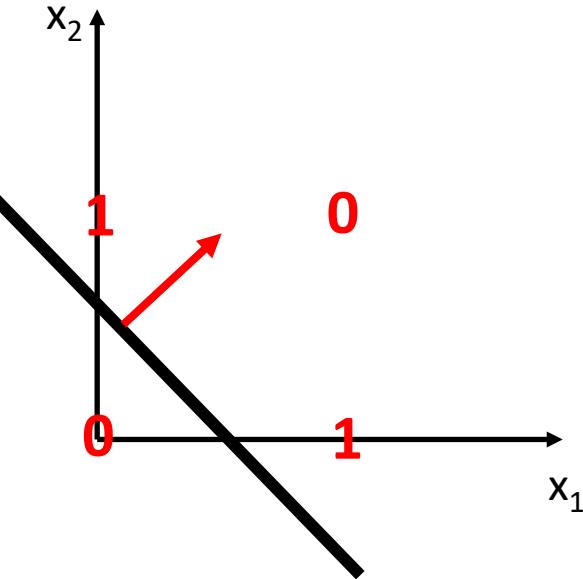
$$w_{01} = -\frac{3}{2} \quad w_{11} = w_{21} = 1$$

x_1	x_2	o_1
0	0	0
0	1	0
1	0	0
1	1	1

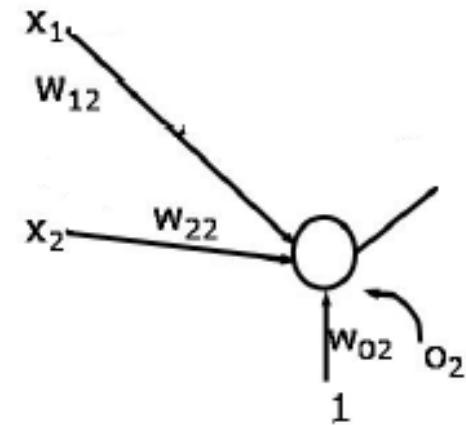




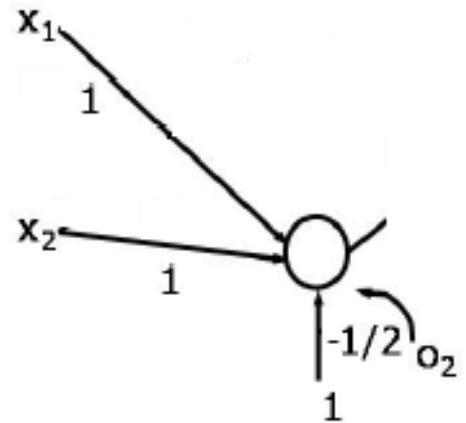
Problem which cannot be solved with a unique straight line



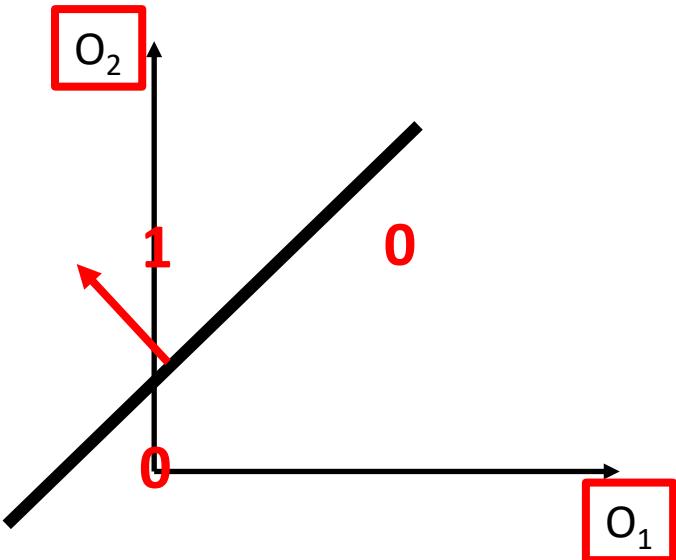
x_1	x_2	o_1	o_2
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1



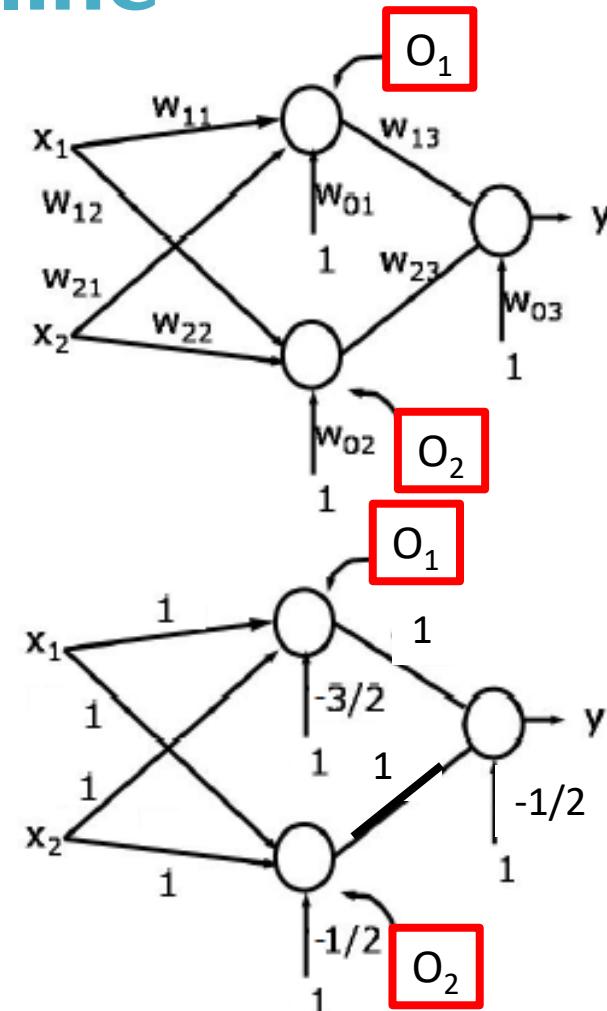
$$w_{02} = -1/2 \quad w_{12} = w_{22} = 1$$



Problem which cannot be solved with a unique straight line

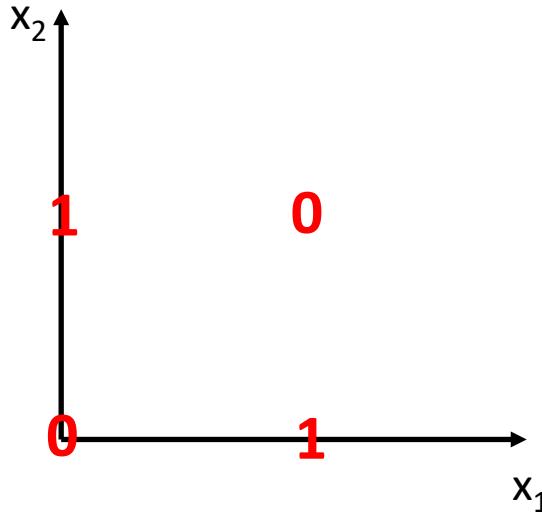


o_1	o_2	y
0	0	0
0	1	1
0	1	1
1	1	0



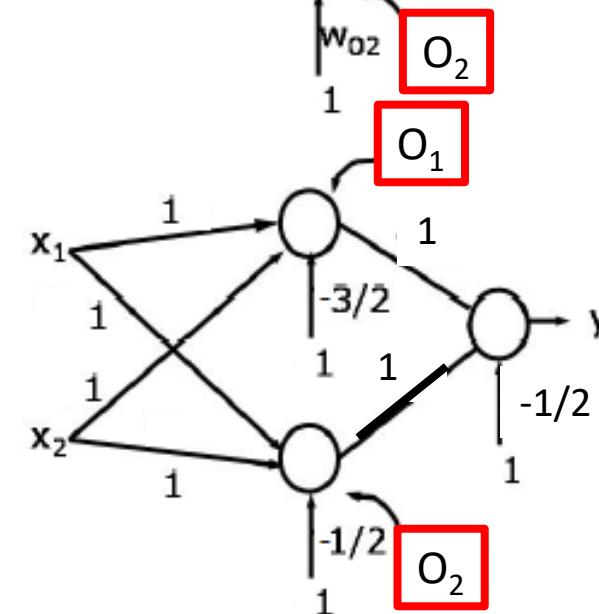
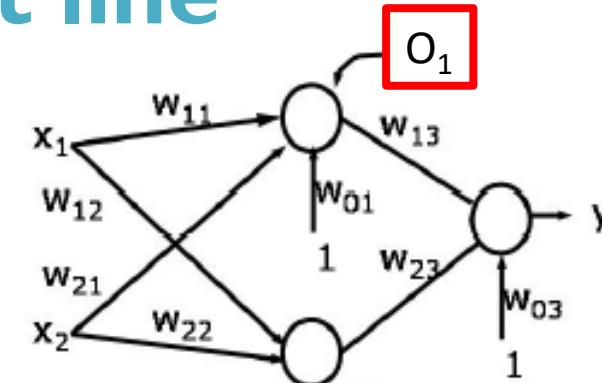
$$w_{23}O_2 + w_{13}O_1 + w_{03} = 0$$

$$w_{03} = -1/2, w_{13} = -1, w_{23} = 1$$



x_1	x_2	o_1	o_2	y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

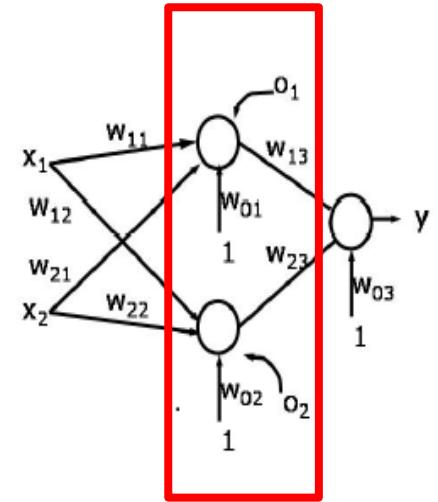
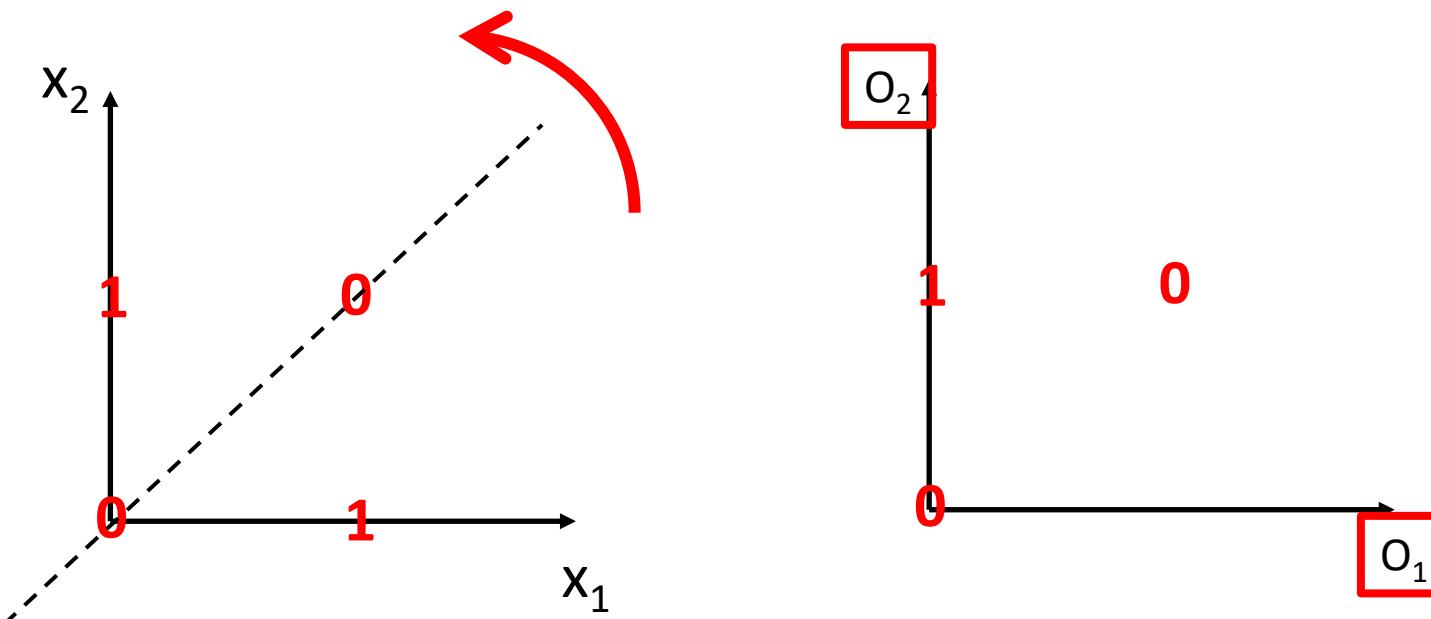
Problem which cannot be solved with a unique straight line



$$w_{23}o_2 + w_{13}o_1 + w_{03} = 0$$

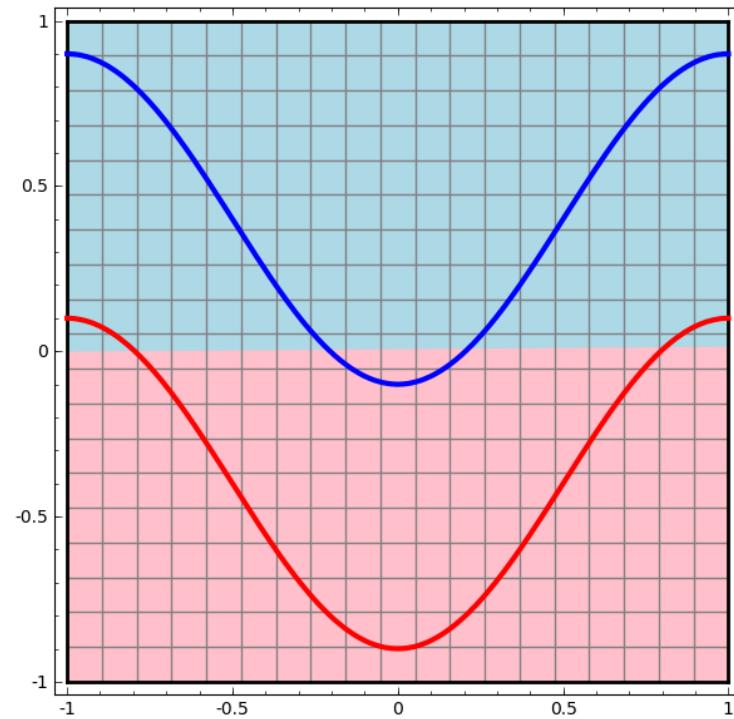
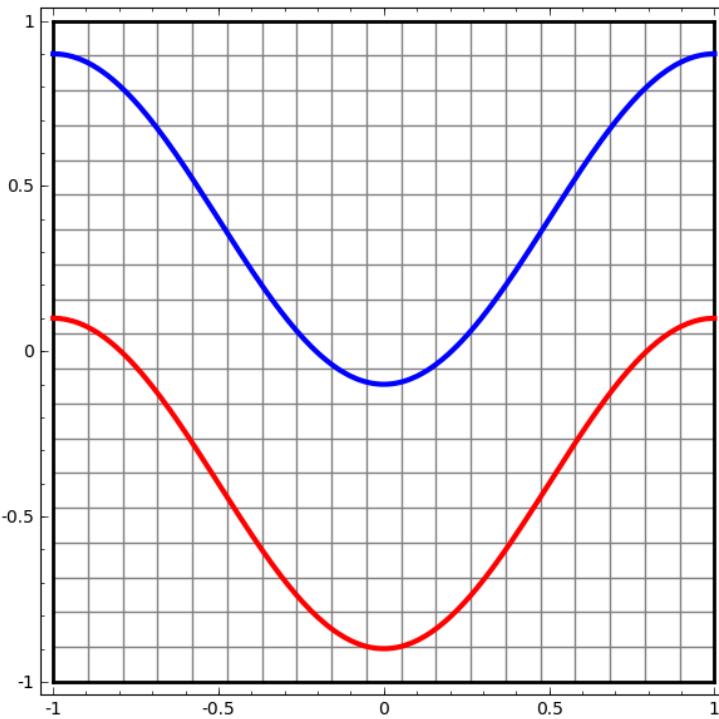
$$w_{03} = -1/2, w_{13} = -1, w_{23} = 1$$

Problem which cannot be solved with a unique straight line



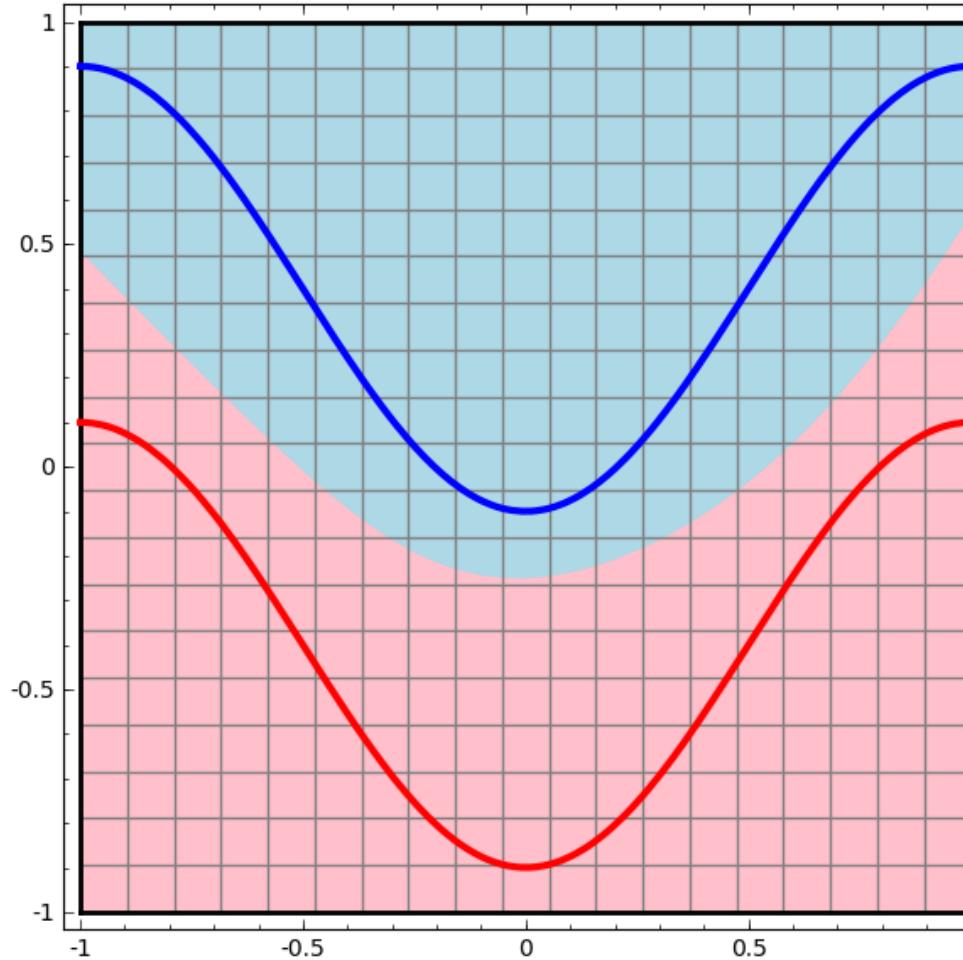
Adding a hidden layer of neurons has fold the input space

One perceptron



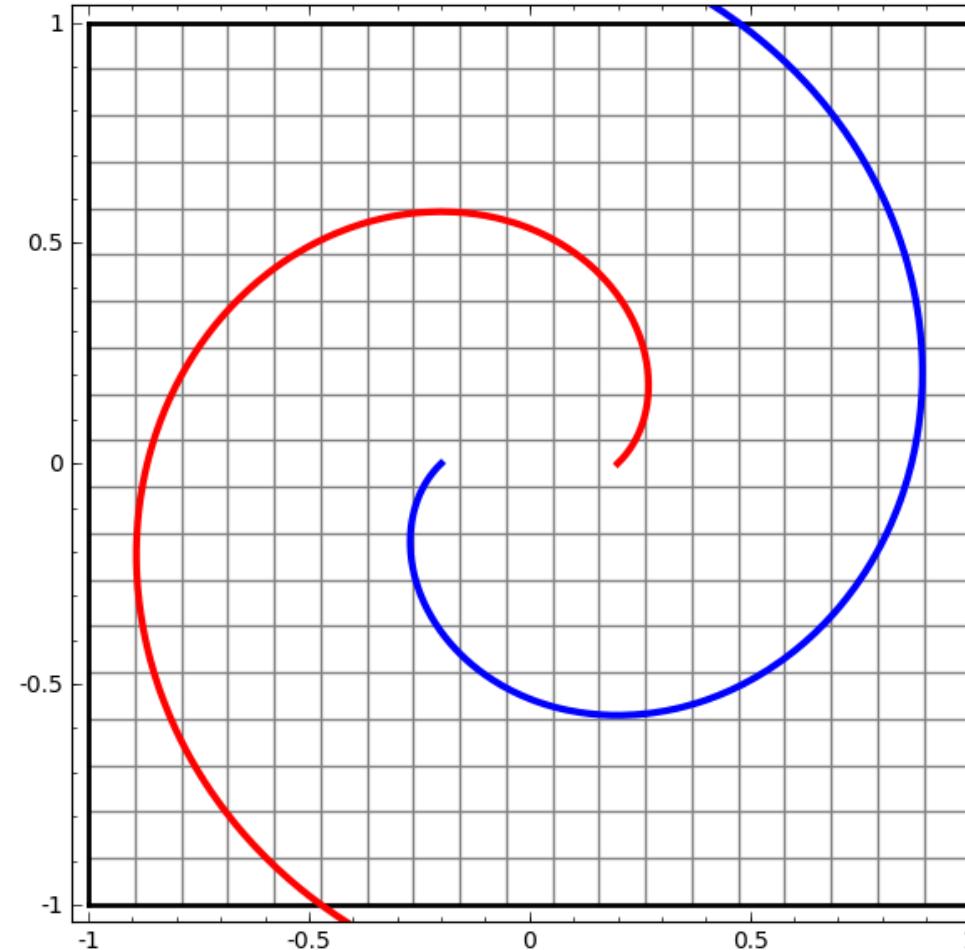
Illustrations from: <http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/>

Multi-Layer Perceptron, manifold disentanglement



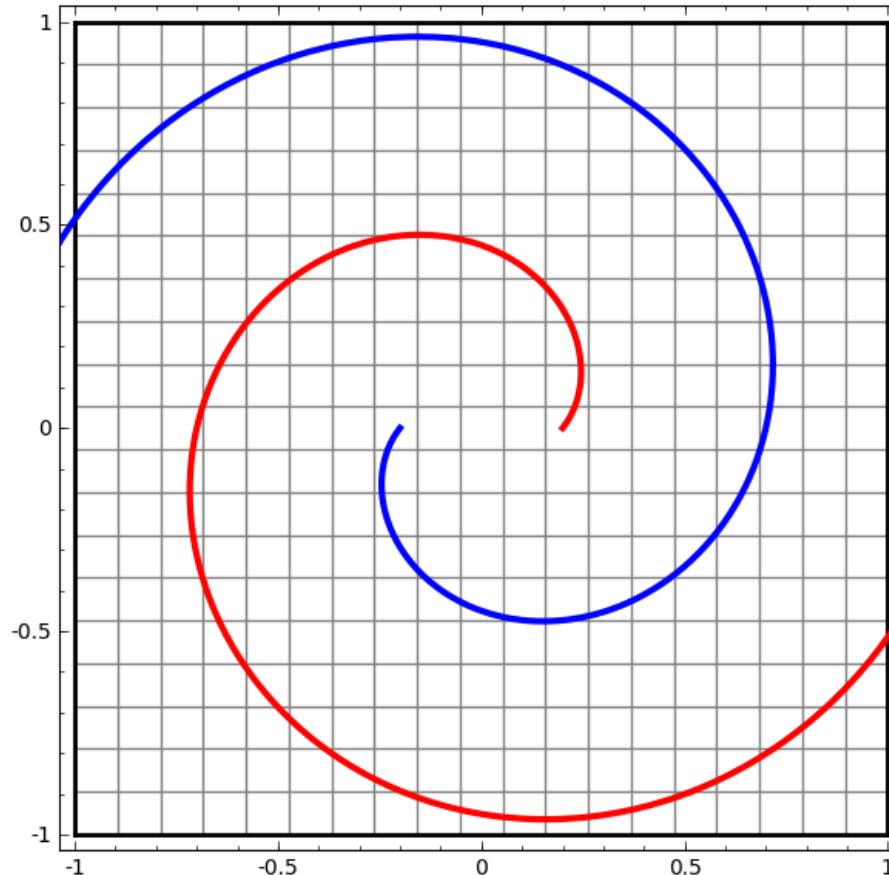
Illustrations from: <http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/>

Multi-Layer Perceptron, manifold disentanglement

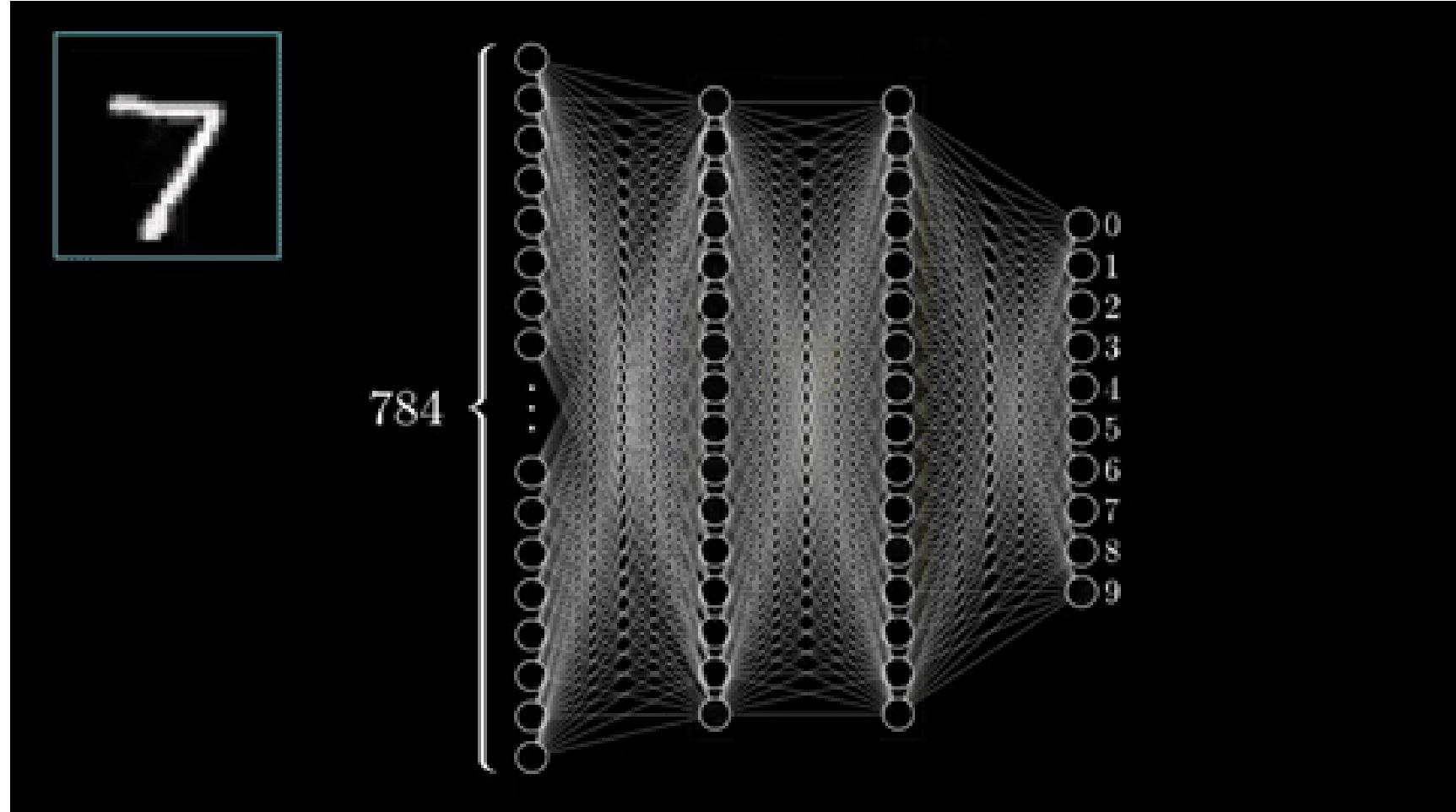




Multi-Layer Perceptron, manifold disentanglement

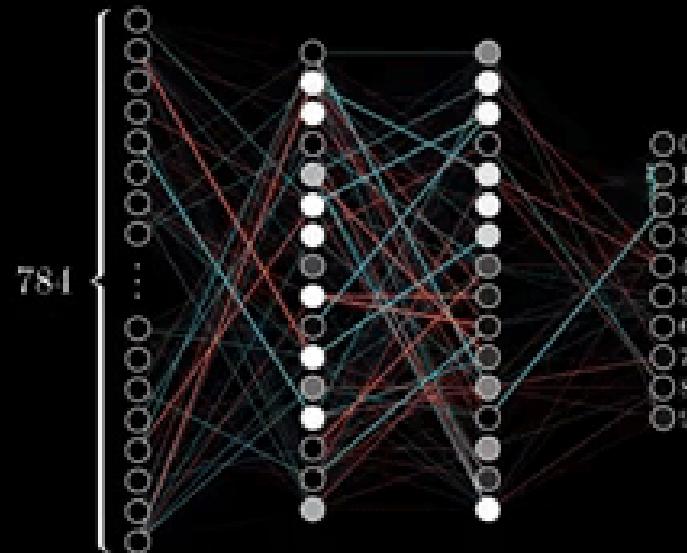


Solution: Neural Network

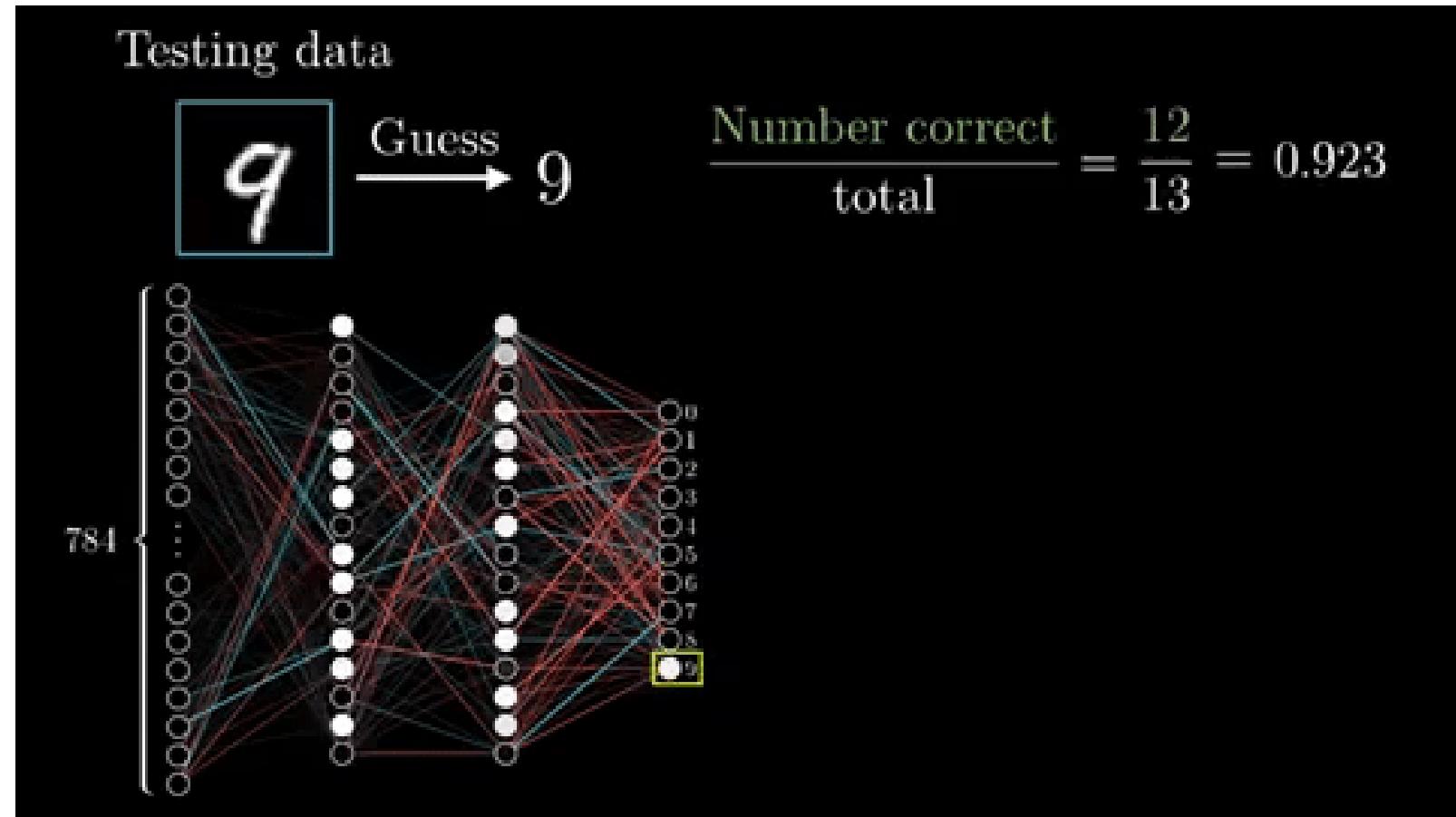


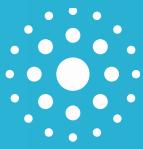
Training

Training in
progress. . .



Inference/Test





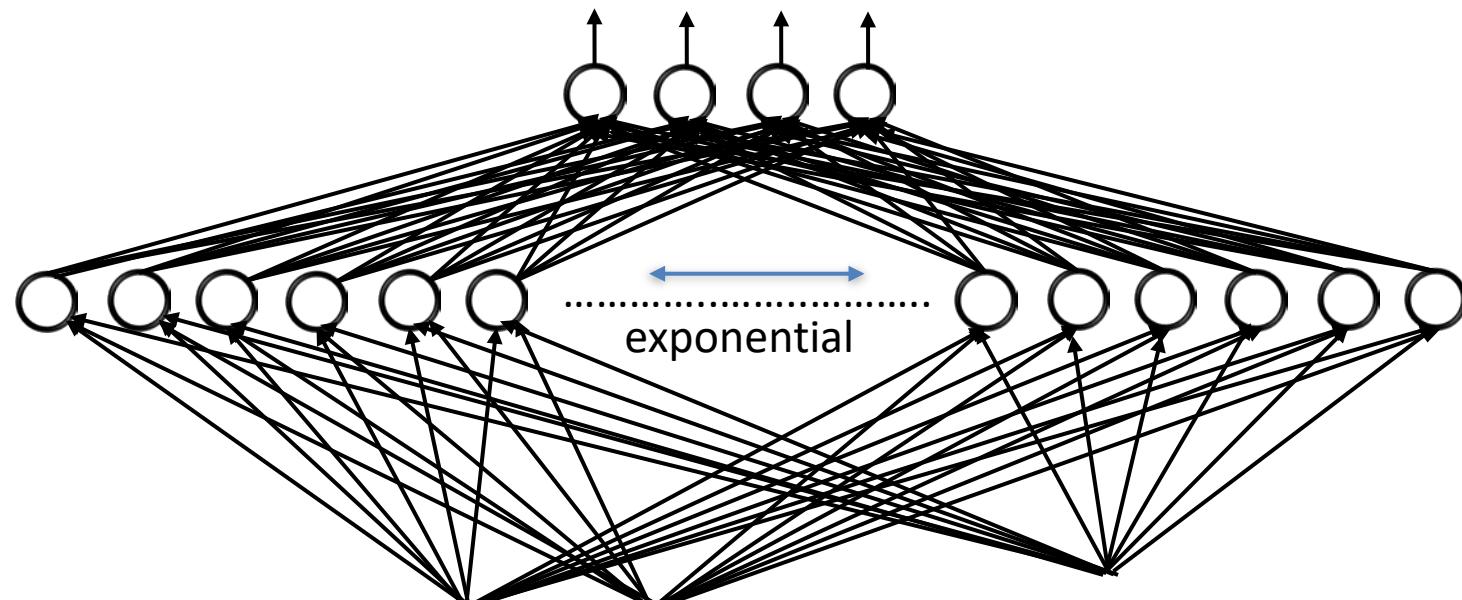
DEEP LEARNING PRINCIPLES



Deep representation origins

- **Theorems (Cybenko (1989), Hornik & Stinchcombe & White (1989)), Allan Pinkus (1999)):**
*A neural network with one single hidden layer is a **universal approximator**, i.e. it can represent any continuous function on compact subsets of \mathbb{R}^n with error 0!!*

 \Rightarrow 1 layers is enough...but hidden layer size may be **exponential** to get error prediction lower than ε , or even **infinite** for a prediction error 0



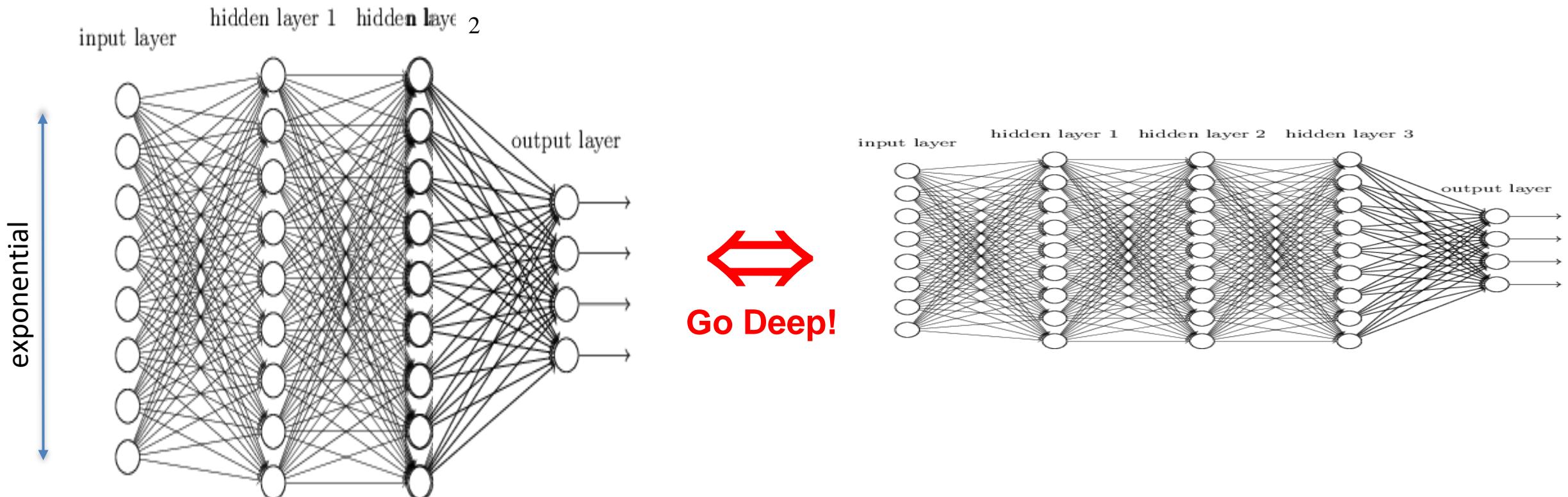
Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4), 303-314.
Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5), 359-366.
Pinkus, Allan (1999). "Approximation theory of the MLP model in neural networks". *Acta Numerica*. 8: 143–195.



Deep representation origins

univ-cotedazur.fr

- **Theorem Hastad (1986), Bengio et al. (2007)** Functions representable compactly with k layers would require $k-1$ layers with exponential size



Hastad, J. (1986). Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th annual ACM Symposium on Theory of Computing*, pp. 6–20 Berkeley, California. ACM Press.
Johan T. Hastad. *Computational Limitations for Small Depth Circuits*. MIT Press, Cambridge, MA, 1987.

Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*. MIT Press, 2007.

The Blessing of dimensionality: Thomas Cover's Theorem (1965)

Cover's theorem states: A complex pattern-classification problem cast in a high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space.

Through repeated sequence of Bernoulli trials:

The number of groupings that can be formed by $(l-1)$ -dimensional hyperplanes to separate N points in two classes is:

$$O(N, l) = 2 \sum_{i=0}^l \frac{(N-1)!}{(N-1-i)!i!}$$

Notice: The total number of possible groupings is 2^N



Deep Neural Network Approximation Theory (2021)

“...deep networks provide exponential approximation accuracy—i.e., the approximation error decays exponentially in the number of nonzero weights in the network—of the multiplication operation, polynomials, sinusoidal functions, and certain smooth functions.

Moreover, this holds true even for one-dimensional oscillatory textures and the Weierstrass function—a fractal function, neither of which has previously known methods achieving exponential approximation accuracy.

We also show that in the approximation of sufficiently smooth functions finite-width deep networks require strictly smaller connectivity than finite-depth wide networks.”

Elbrächter, D., Perekrestenko, D., Grohs, P., & Bölcskei, H. (2021). Deep neural network approximation theory. IEEE Transactions on Information Theory, 67(5), 2581-2623.

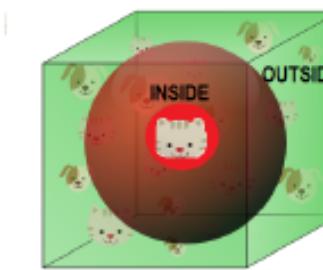


The curse of dimensionality [Bellman, 1956]

univ-cotedazur.fr

- Euclidian distance is not relevant in high dimension: $d \geq 10$
 - ➊ look at the examples at distance at most r
 - ➋ the hypersphere volume is too small: practically empty of examples

$$\frac{\text{volume of the sphere of radial } r}{\text{hypersphere of } 2r \text{ width}} \xrightarrow{d \rightarrow \infty} 0$$



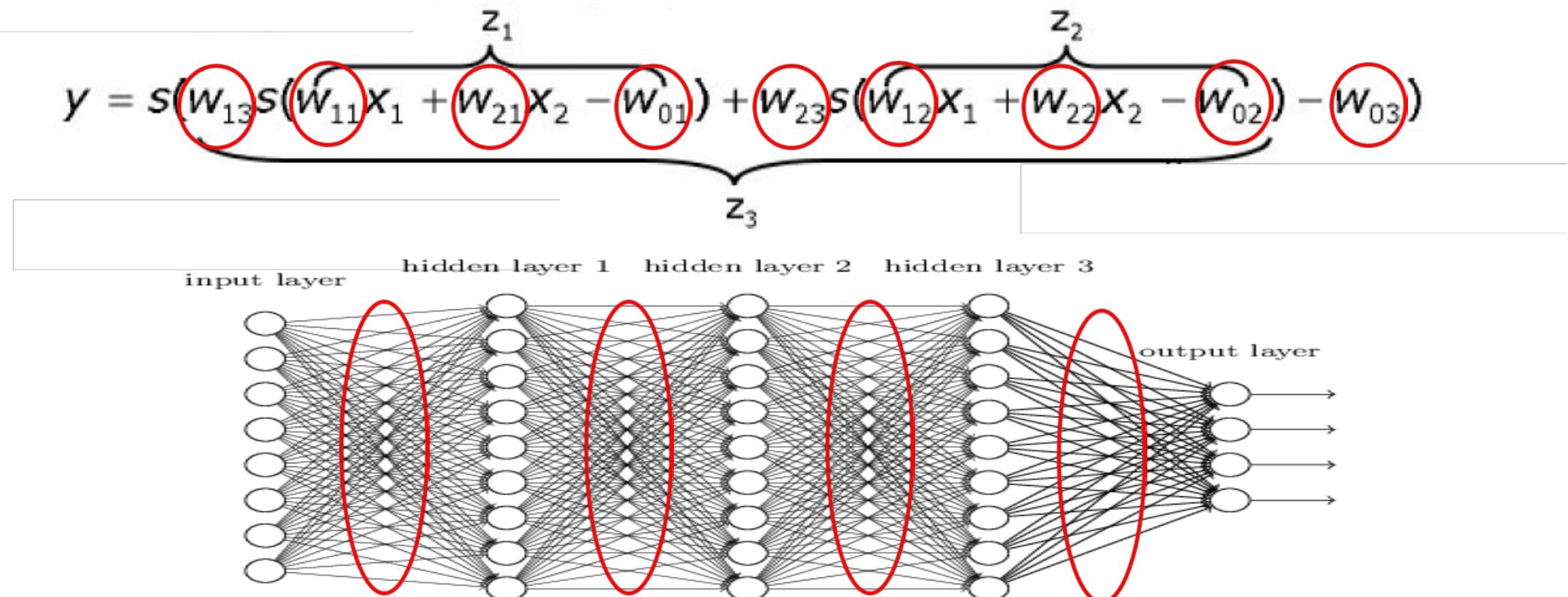
- ➌ need a number of examples exponential in d

Remark

Specific care for data representation

Structure the network?

- Can we put any structure reducing the space of exploration and providing useful properties (invariance, robustness, sequentiality...)?





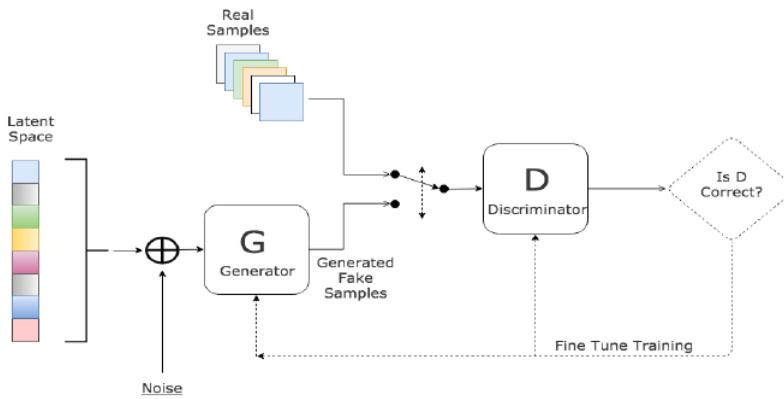
Enabling factors

- Why do it now ? Before 2006, training deep networks was unsuccessful because of practical aspects
 - faster CPU's
 - parallel CPU architectures
 - advent of GPU computing
 - Advances in ML/Optim (1995 -> 2005)
- Hinton, Osindero & Teh « A Fast Learning Algorithm for Deep Belief Nets », Neural Computation, 2006
- Bengio, Lamblin, Popovici, Larochelle « Greedy Layer-Wise Training of Deep Networks », NIPS'2006
- Ranzato, Poultney, Chopra, LeCun « Efficient Learning of Sparse Representations with an Energy-Based Model », NIPS'2006
- Results...
 - 2009, sound, interspeech +~24%
 - 2011, text, +~15% without linguistic at all
 - 2012, images, ImageNet +~20%
 - 2020, molecules/graphs, AlphaFold, +~24%
(sorry in French, <https://www.youtube.com/watch?v=OGewxRMME8o>)

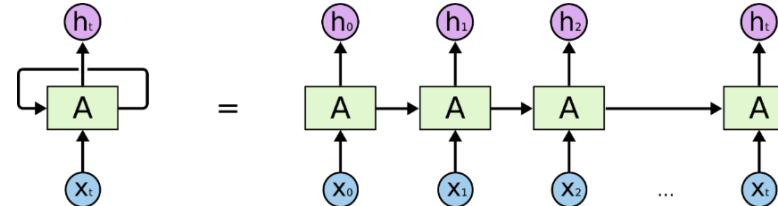
Network architectures



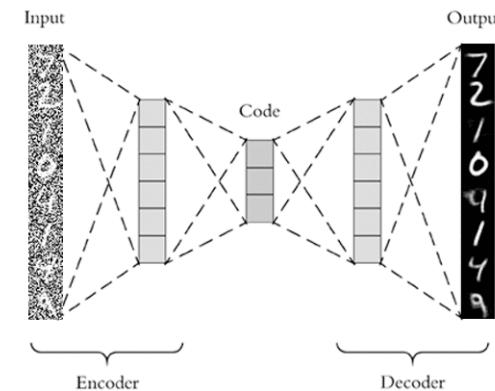
Spatial Correlation (images, sounds...)



Generative Adversarial Network (Data Generator)



Sequential Correlation
(machine translation, speech 2 text, time
series...)



Dimension reduction, compression,
denoising/data cleansing, data generator

Deep learning = Learning representations/features

- The traditional model of pattern recognition (since the late 50's)
 - Fixed/engineered features (or fixed kernel) + trainable classifier



- End-to-end learning / Feature learning / Deep learning
 - Trainable features (or kernel) + trainable classifier

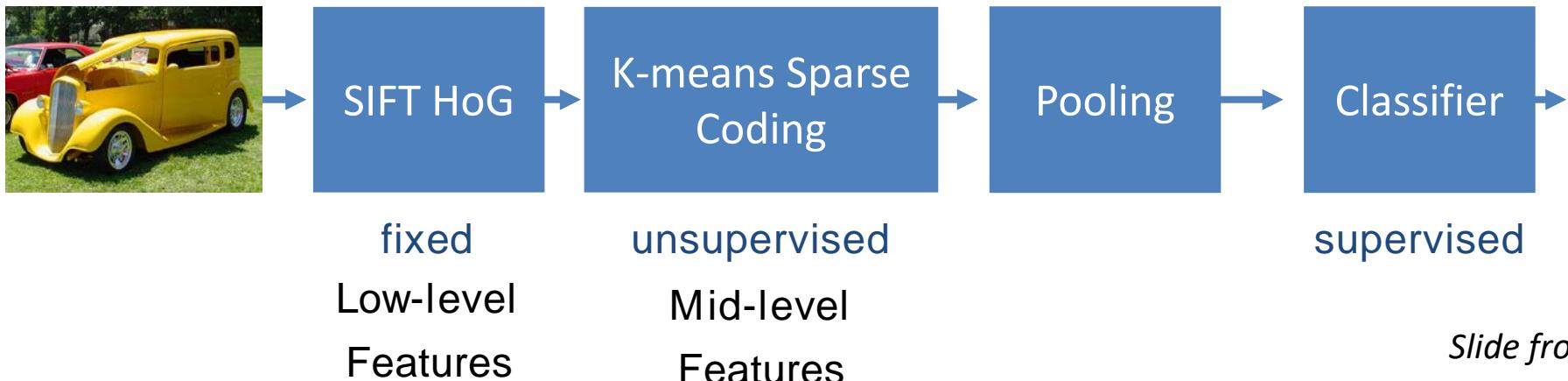


Architecture of “mainstream” pattern recognition systems

- Modern architecture for pattern recognition
 - Speech recognition: early 90's – 2011



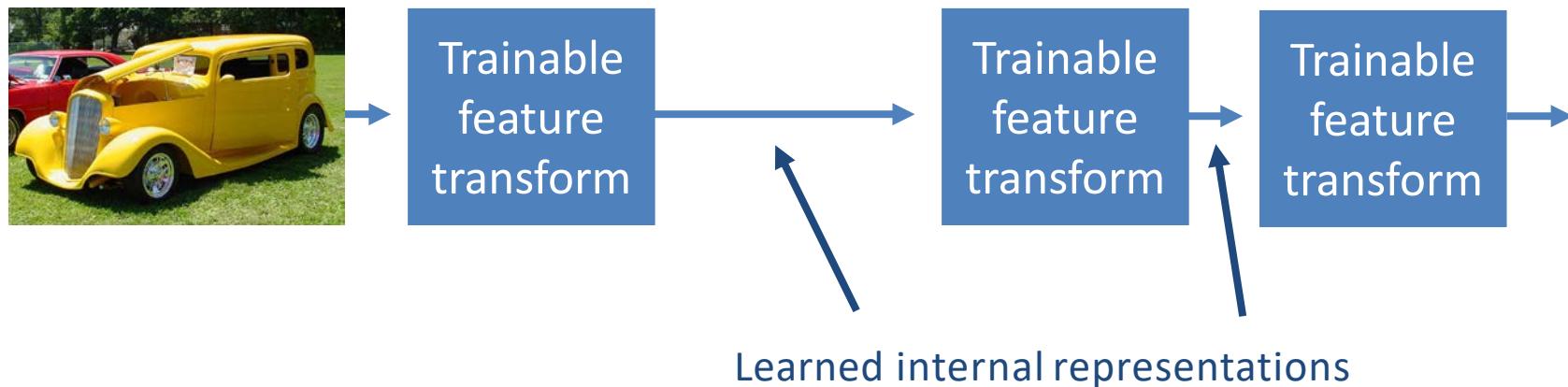
- Object Recognition: 2006 - 2012



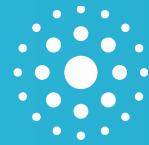


Trainable feature hierarchies: end-to-end learning

- A hierarchy of trainable feature transforms
 - Each module transforms its input representation into a higher-level one.
 - High-level features are more global and more invariant
 - Low-level features are shared among categories



- How can we make all the modules trainable and get them to learn appropriate representations?

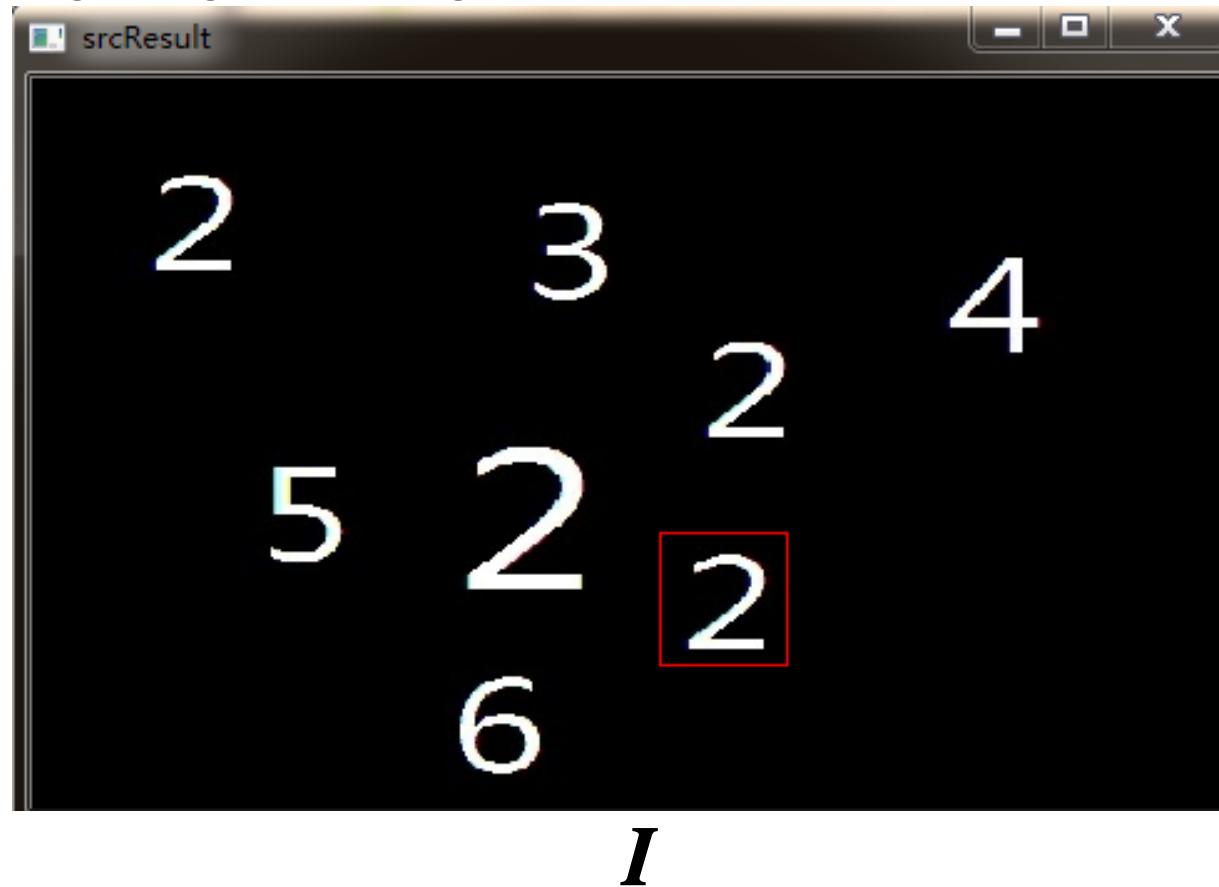


Spatial correlations

**CONVOLUTIONAL NEURAL NETWORKS
(AKA CNN, ConvNET)**

Do you know template matching?

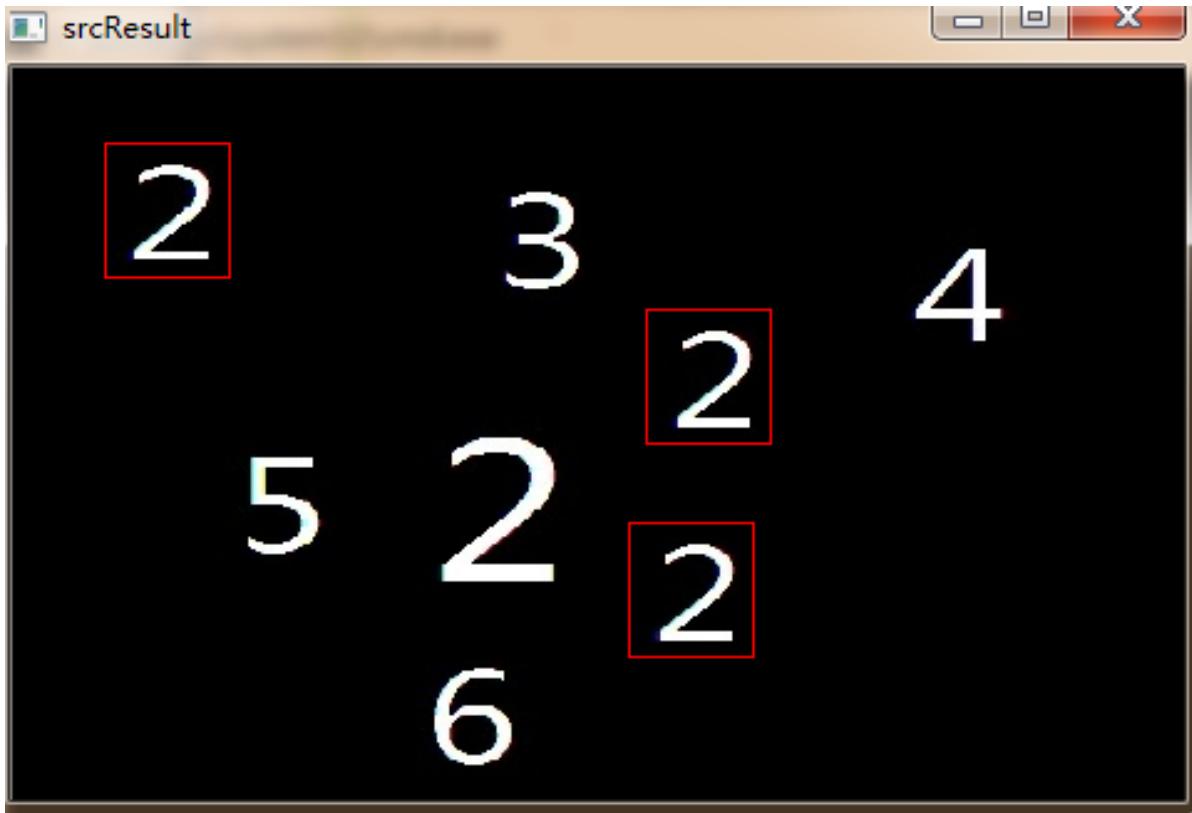
Single target matching results:



(picture from <https://programmer.group/5ca5086fed10b.html>)

Do you know template matching?

Multi-objective matching result:



I



F

(picture from <https://programmer.group/5ca5086fed10b.html>)

Do you know template matching?

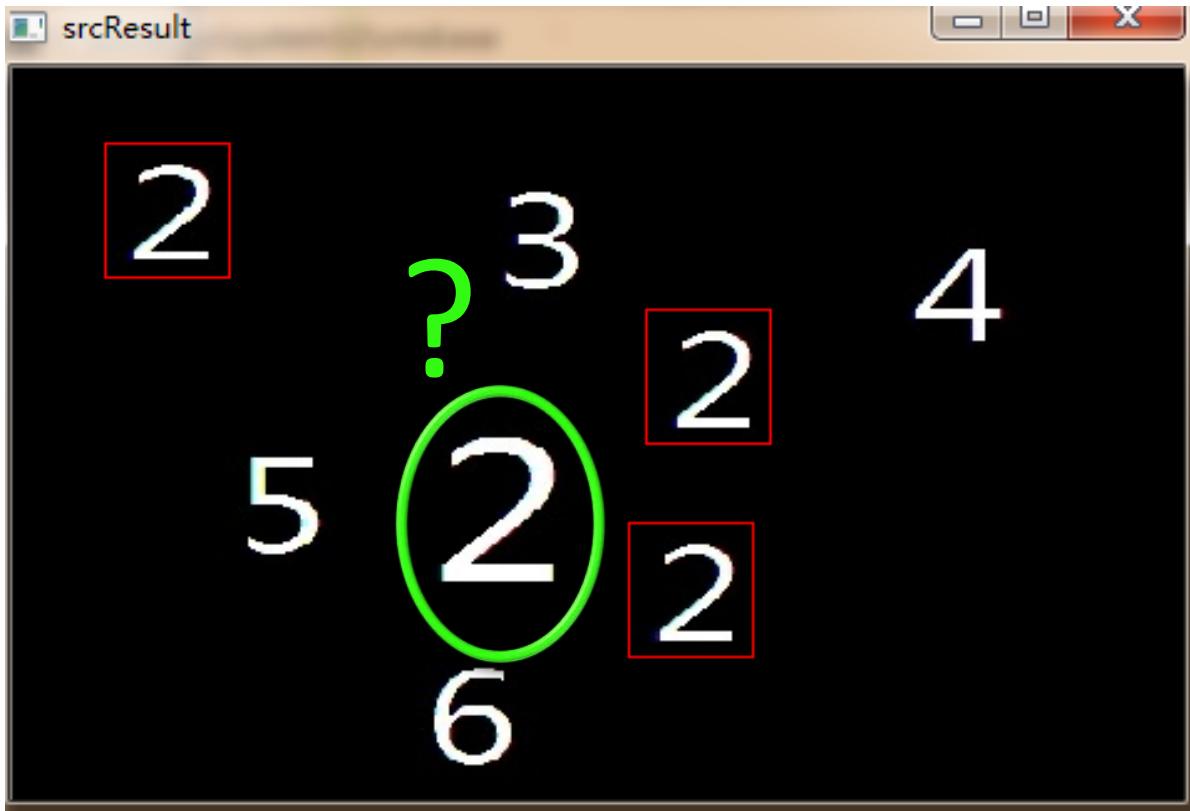
How to compute Template matching? Cross-correlation!

$$F \circ I(x, y) = \sum_{j=-N}^N \sum_{i=-N}^N F(i, j)I(x+i, y+j)$$

When this is maximal, the Filter is perfectly positioned with respect to the pattern to detect.

Do you know template matching?

Multi-objective matching result:



I

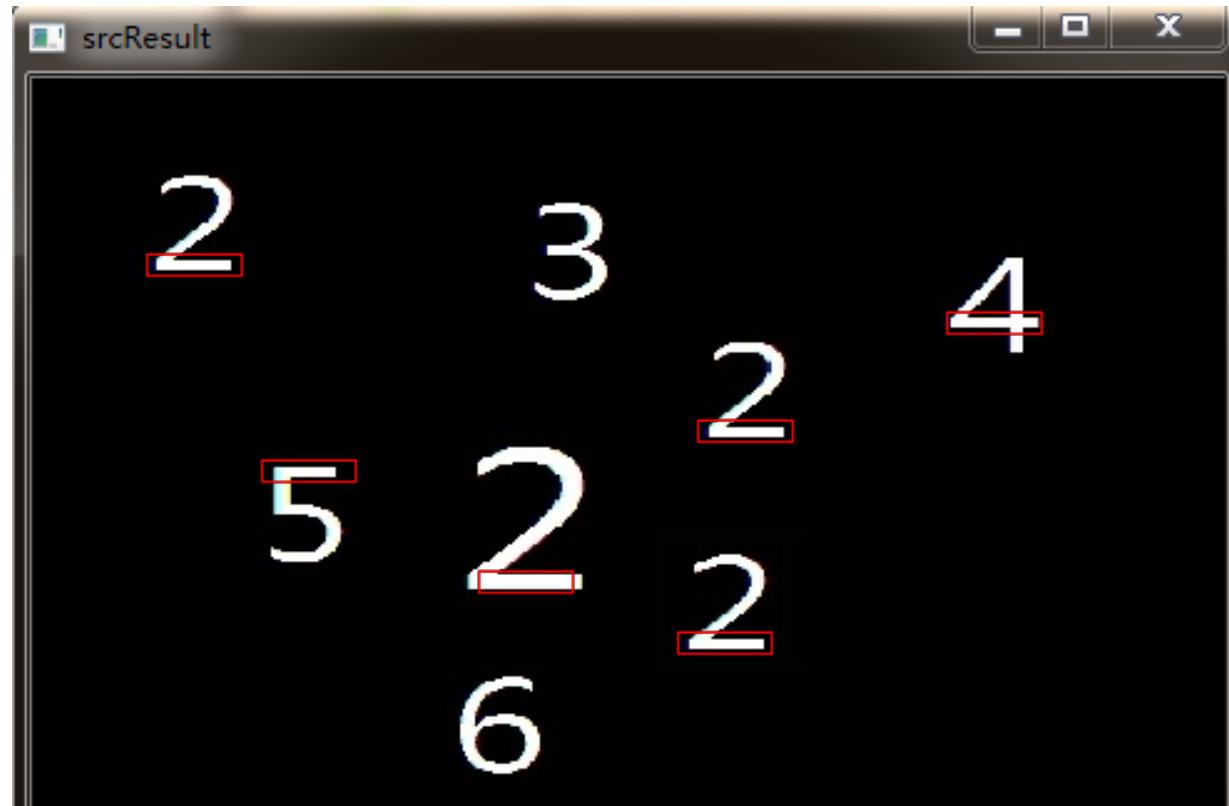


F

(picture from <https://programmer.group/5ca5086fed10b.html>)

Are there more universal templates?

Multi-objective matching result:



I

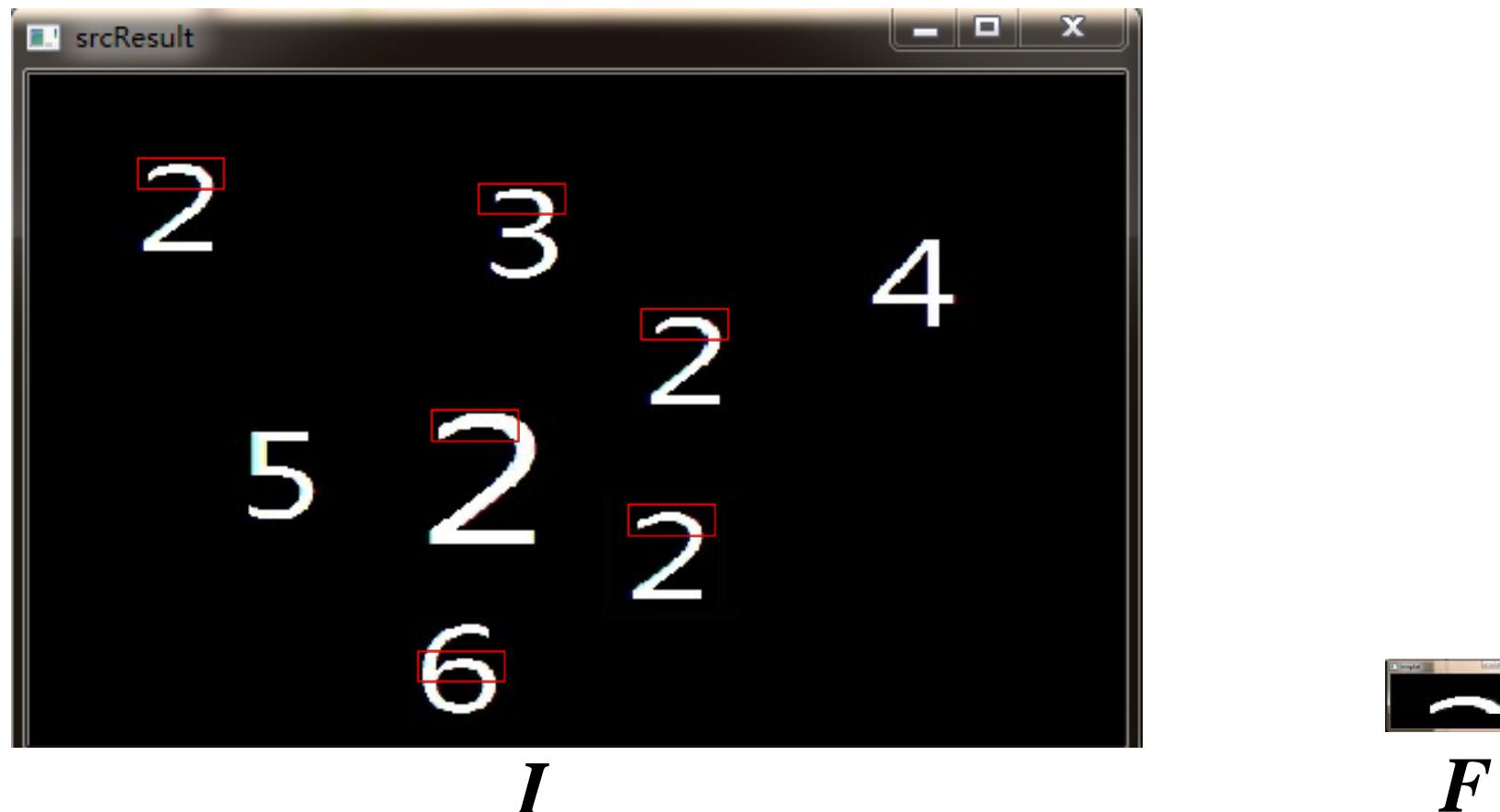
F



(picture from <https://programmer.group/5ca5086fed10b.html>)

Are there more universal templates?

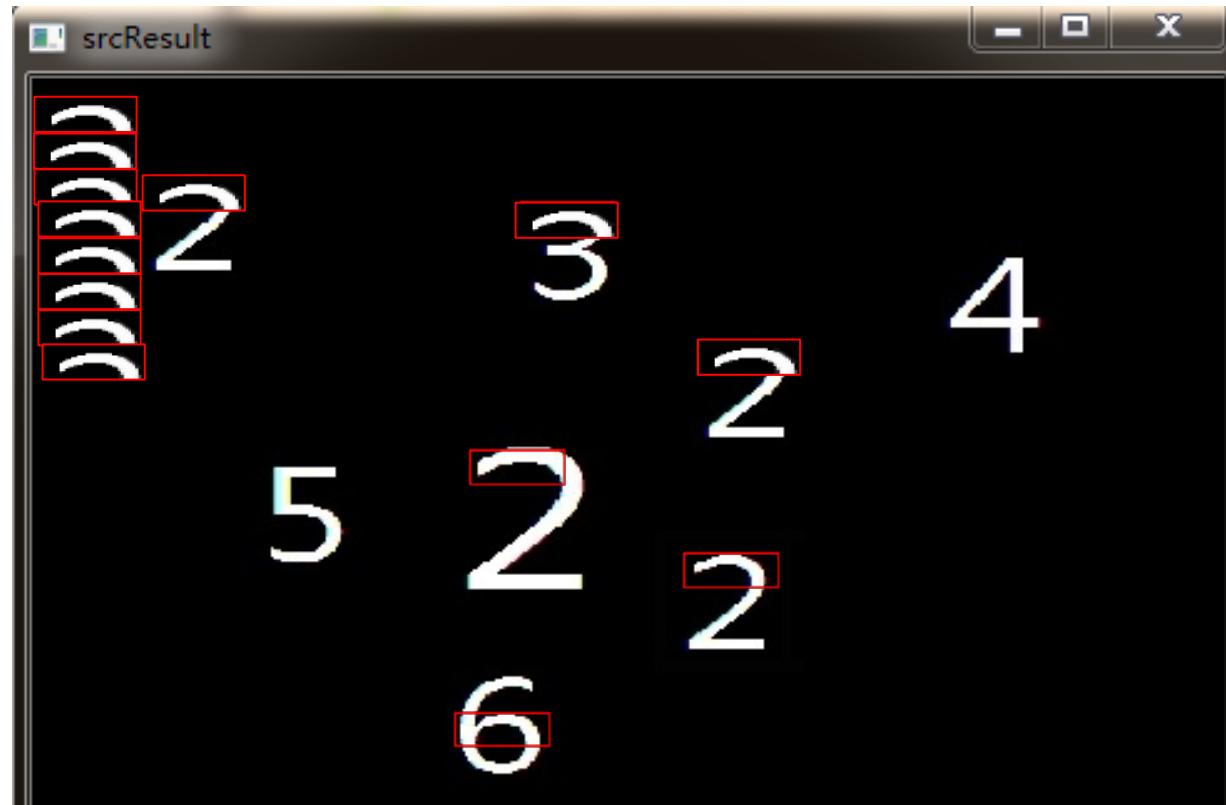
Multi-objective matching result:



(picture from <https://programmer.group/5ca5086fed10b.html>)

How to locate these templates?

By scanning I , we acquire **translation invariance**



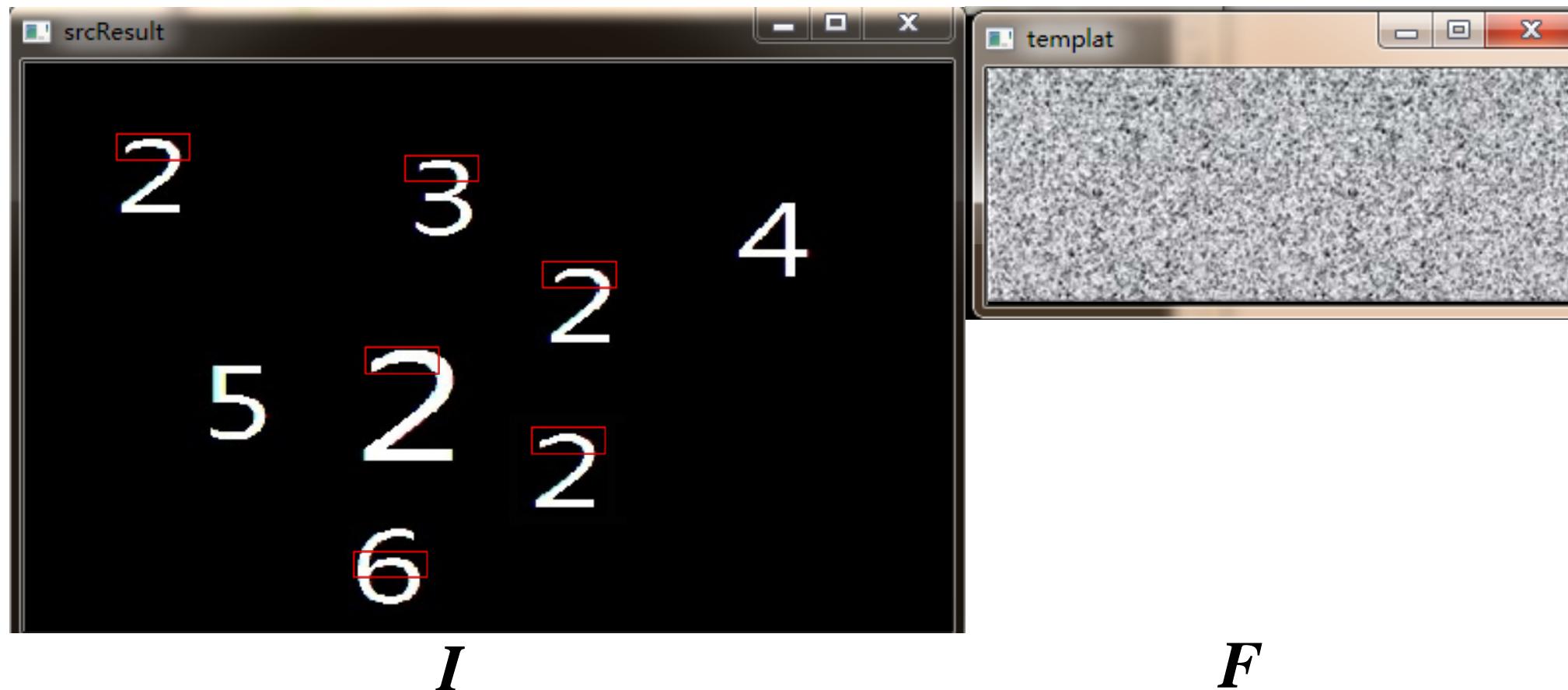
I



F

How to define these templates?

By learning what filter maximizes **local cross-correlations**



Cross-correlation or Convolution?

- How to define these templates?

By learning what filter maximizes **local cross-correlations**

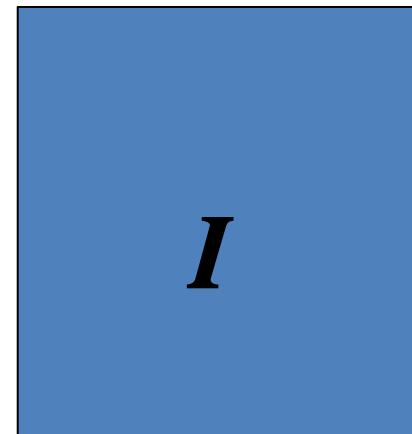
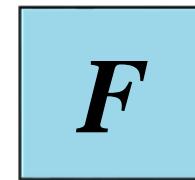
$$F \circ I(x, y) = \sum_{j=-N}^N \sum_{i=-N}^N F(i, j)I(x+i, y+j)$$

Or **local convolutions**

$$F * I(x, y) = \sum_{j=-N}^N \sum_{i=-N}^N F(i, j)I(x-i, y-j)$$

Cross-correlation or Convolution?

- Convolution:
 - Flip the filter in both dimensions (bottom to top, right to left)
 - Then apply cross-correlation

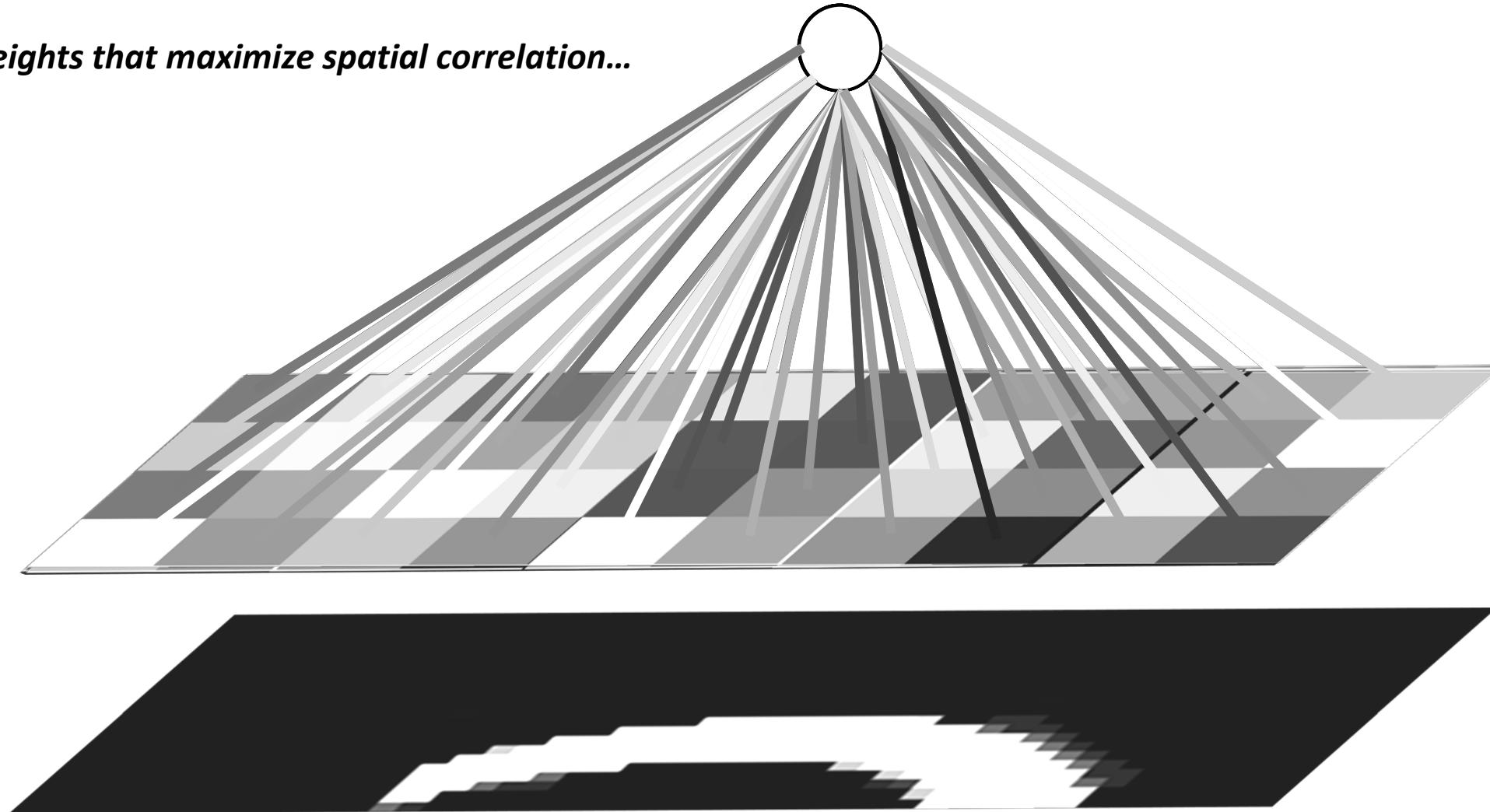


Cross-correlation or Convolution?

- Convolution:
 - Flip the filter in both dimensions (bottom to top, right to left)
 - Then apply cross-correlation
- Actually, The Convolutional operation widely used in Convolution Neural Network (CNN) is commonly and largely considered as a **misnomer**
 - The operation that is used is strictly speaking a correlation.
 - Both the operators have **translational invariance** and **locality**.

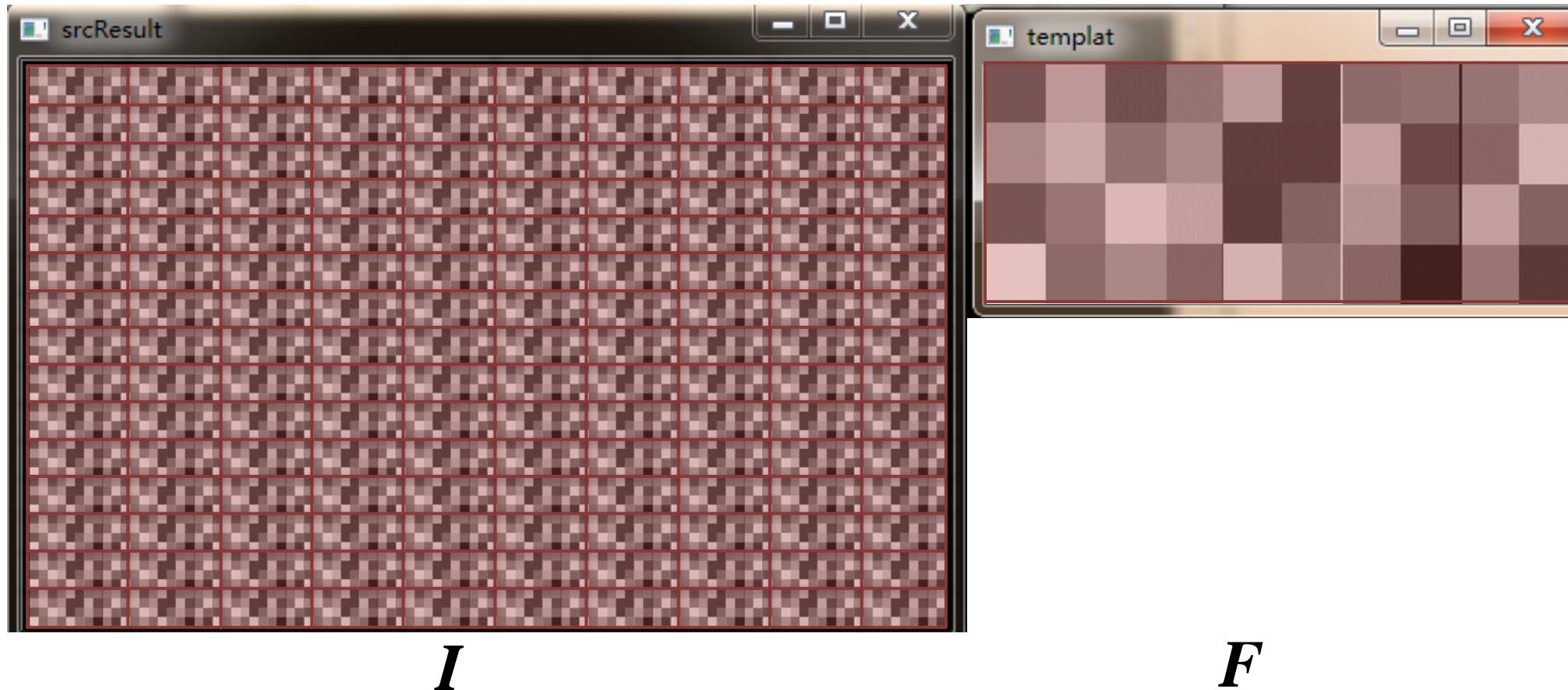
One single neuron is a correlator

Set the weights that maximize spatial correlation...



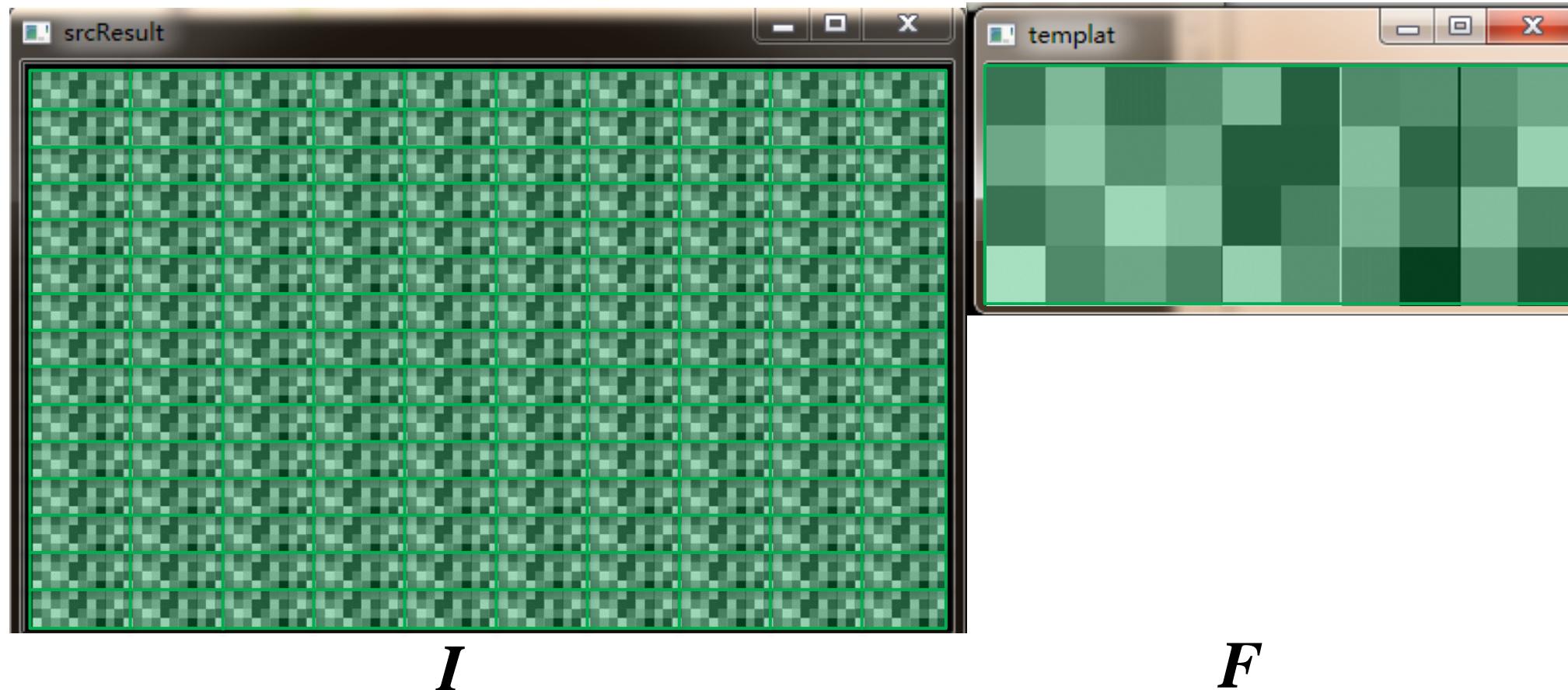
How to define these templates?

And instead of scanning I , let us just **duplicate** the template **into a grid**.



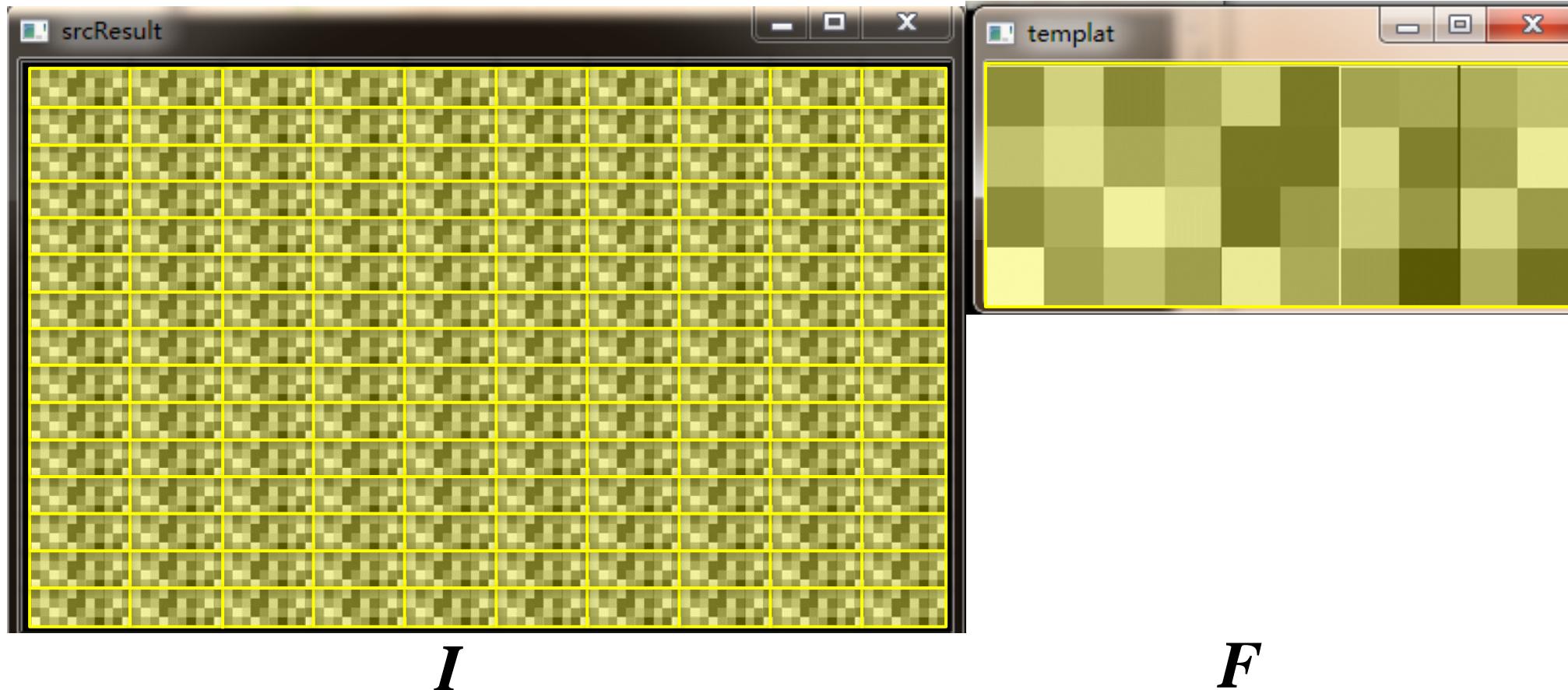
How to define these templates?

And instead of scanning I , let us just **duplicate** the template **into a grid**.



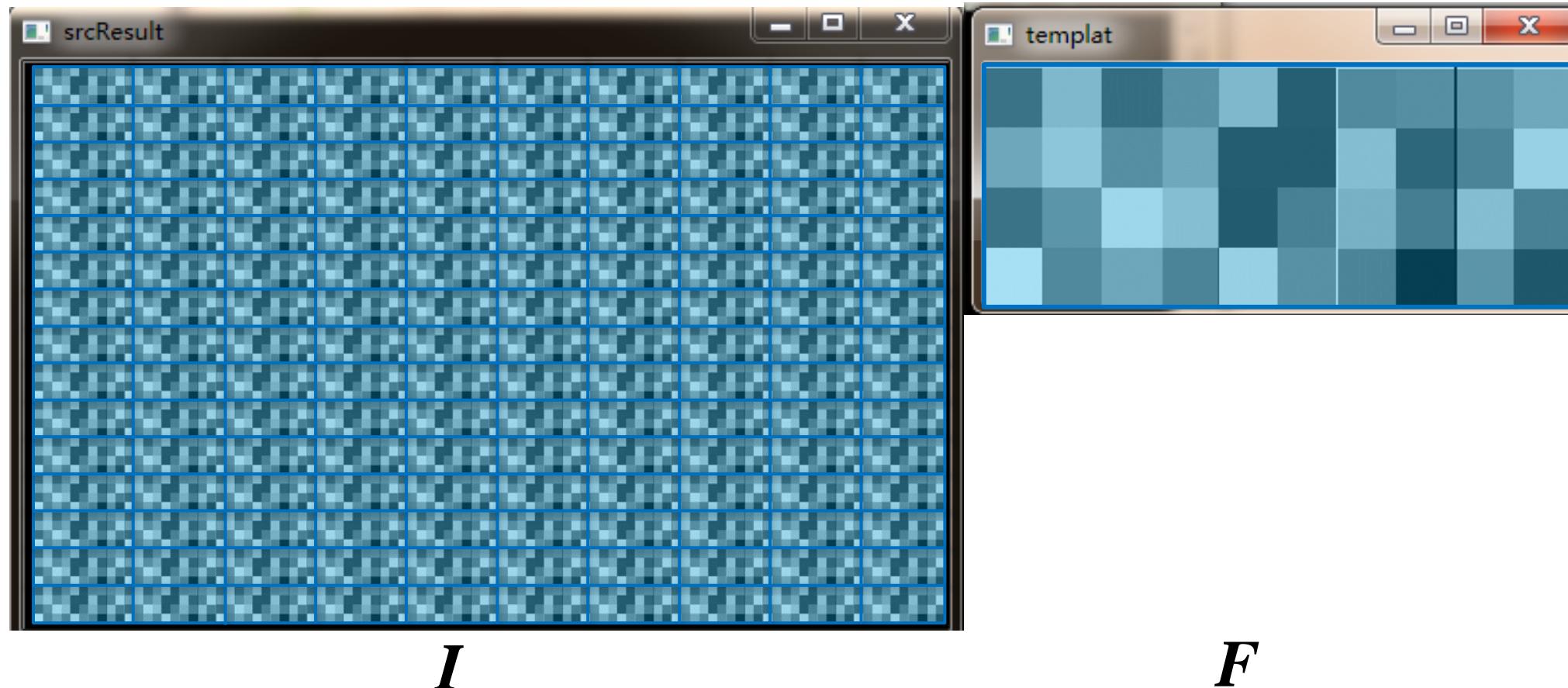
How to define these templates?

And instead of scanning I , let us just **duplicate** the template **into a grid**.



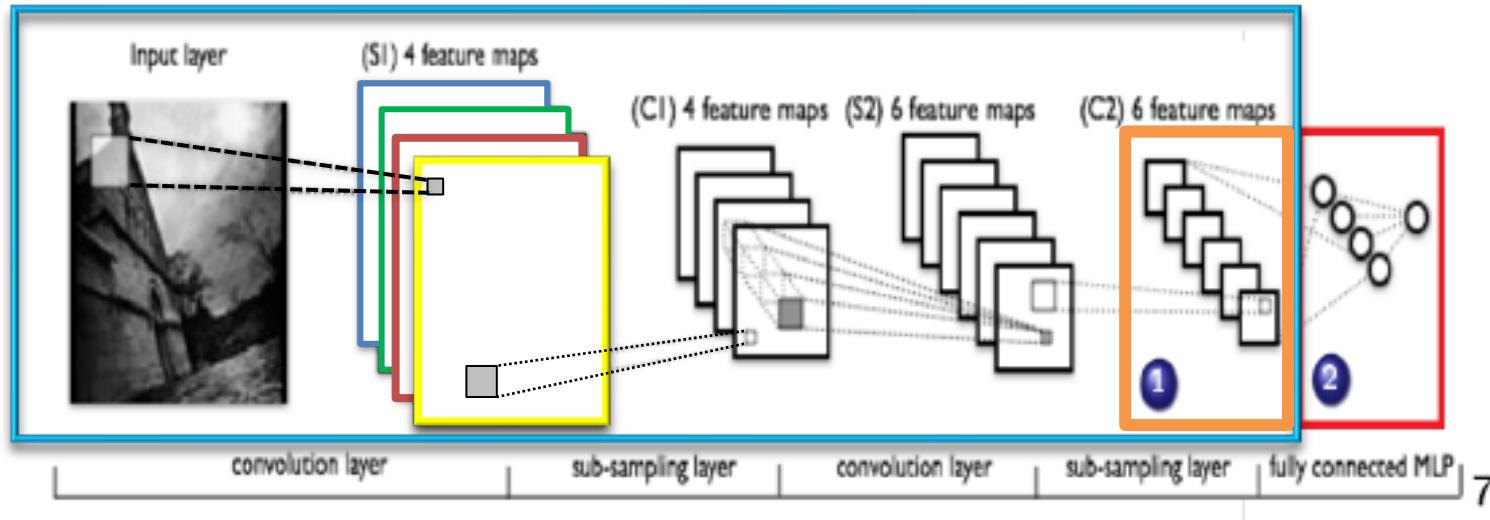
How to define these templates?

And instead of scanning I , let us just **duplicate** the template **into a grid**.



Deep representation by CNN

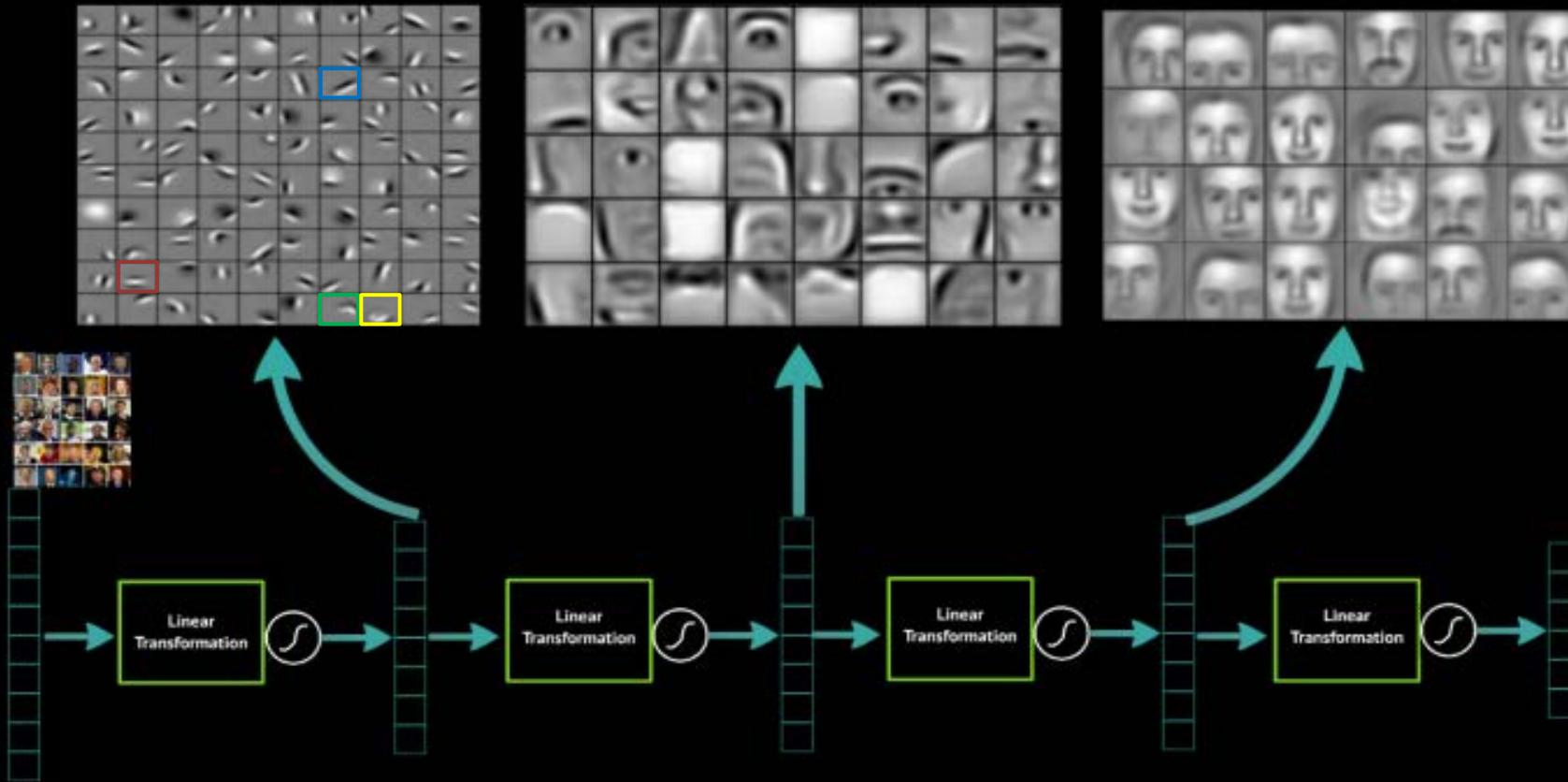
- feature map = result of the convolution
- convolution with a filter extract characteristics (*edge detectors*)
- extract parallelised characteristics at each layer



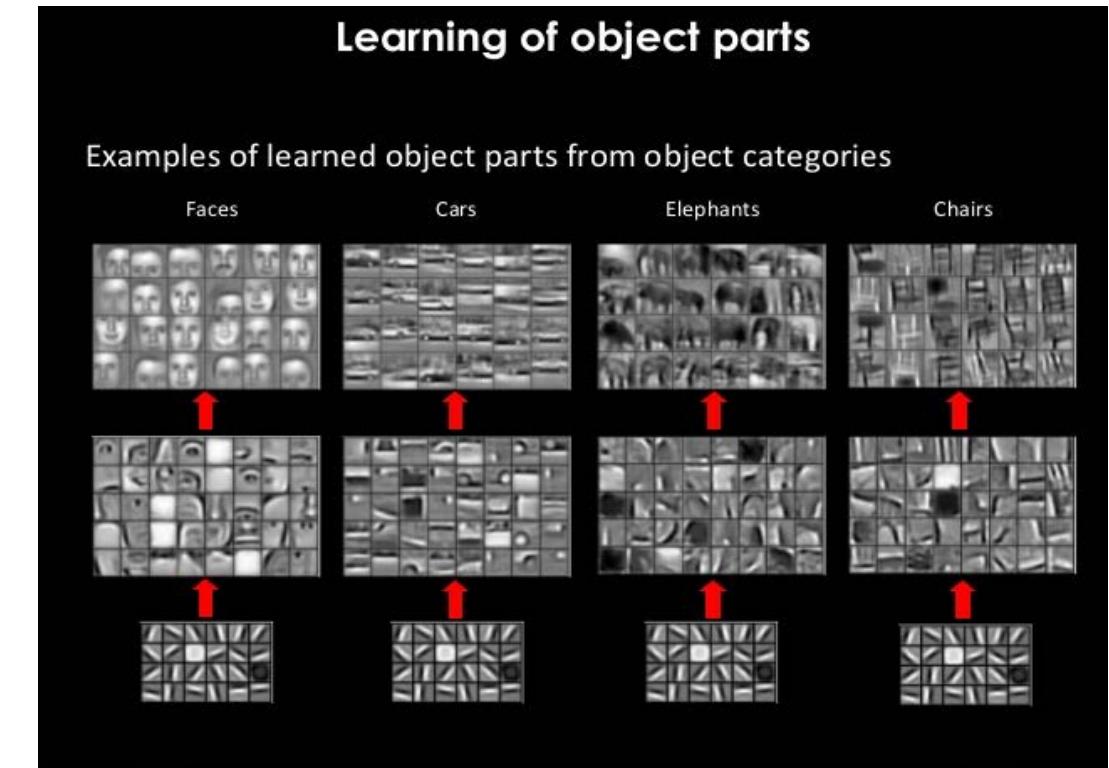
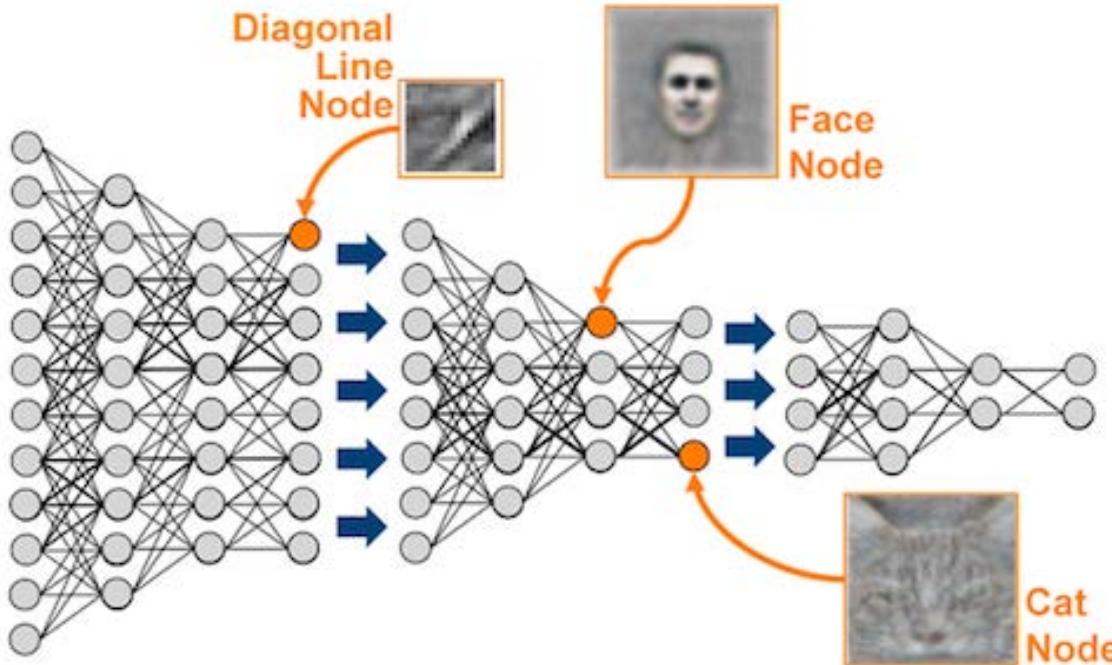
- ➊ final representation of our data
- ➋ classifier (MLP)

Deep representation by CNN

Deep Learning learns layers of features

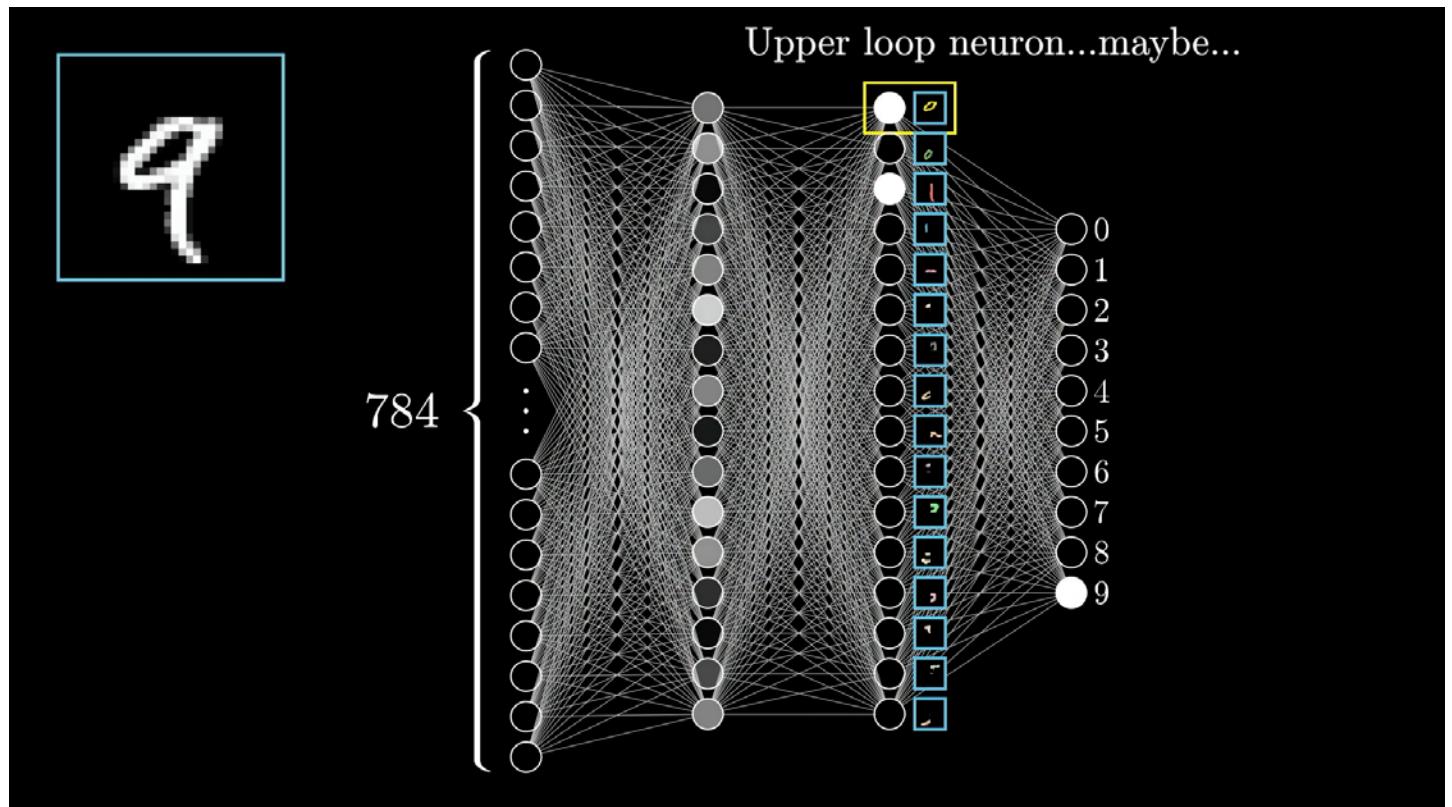
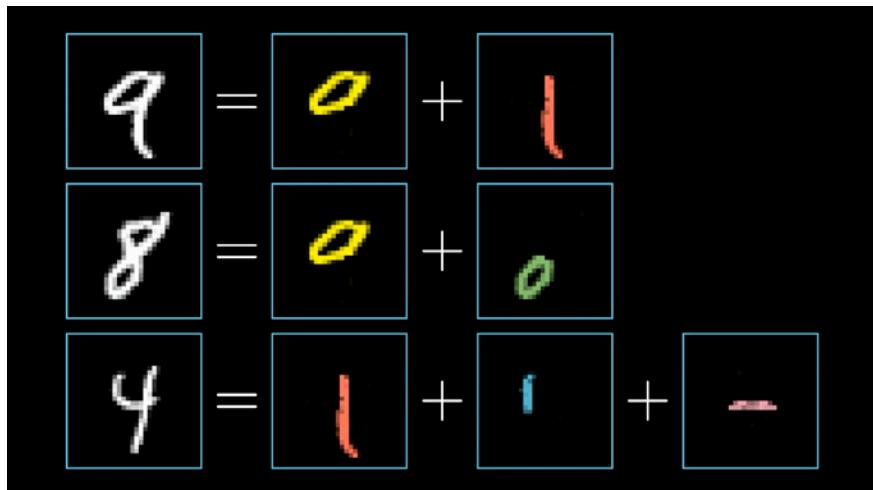


Deep representation by CNN



Deep representation by CNN

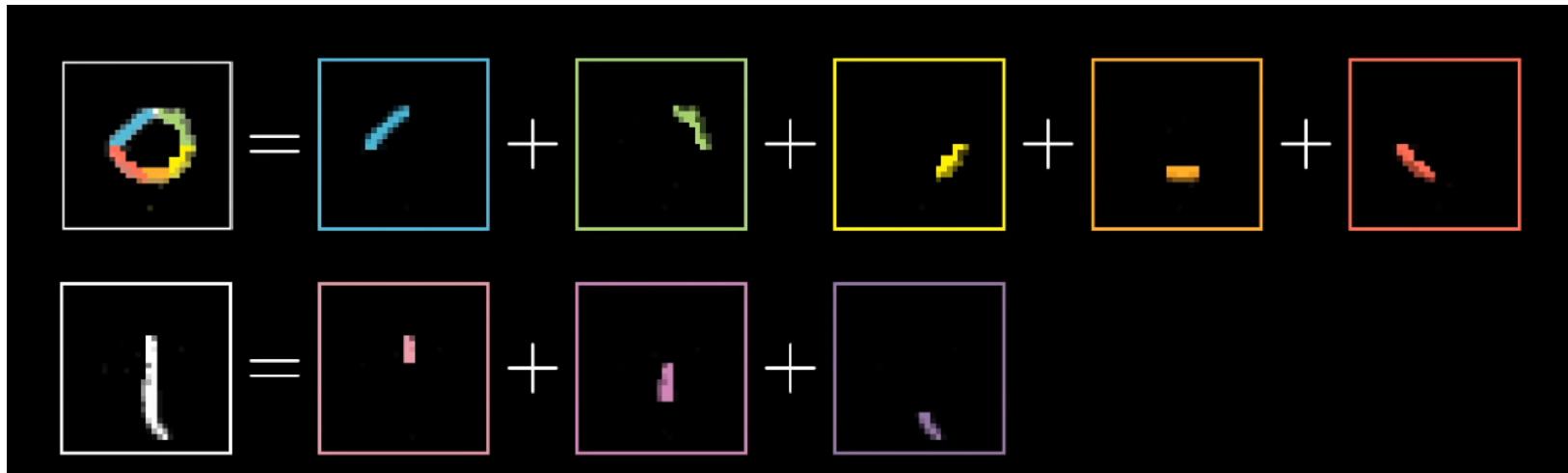
- We hope that the inner layers will detect basic patterns, "as we" break down our visual reasoning:



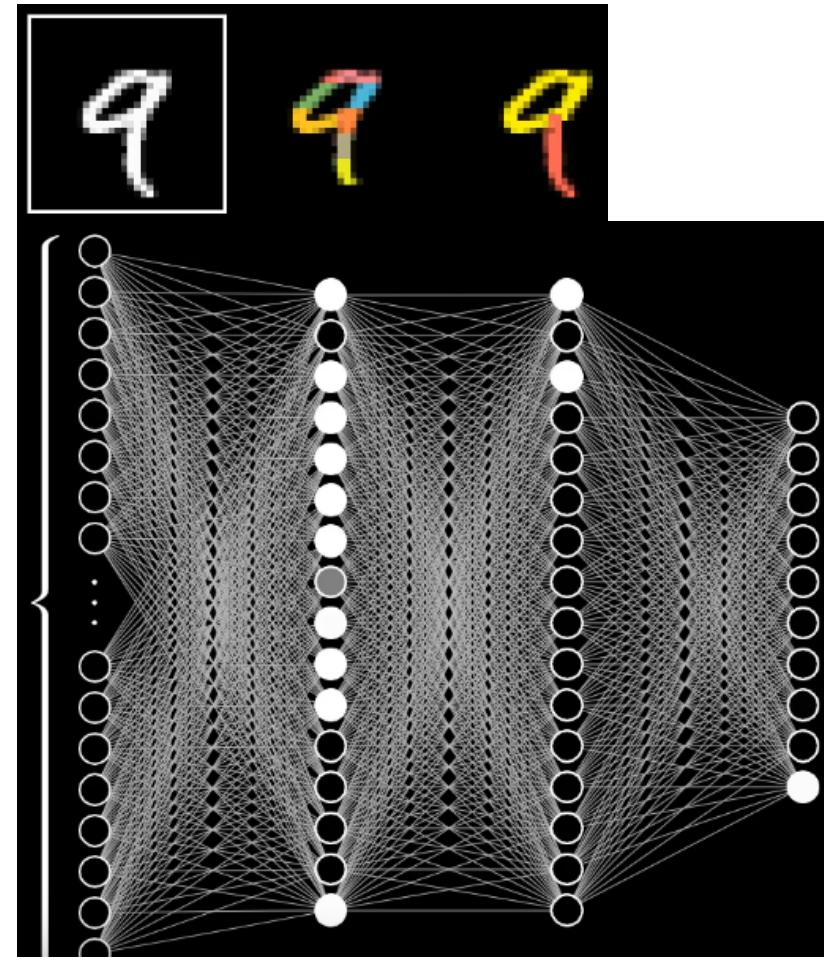
© 3Blue1Brown

Deep representation by CNN

- These patterns (loops, bars, etc.) are in turn broken down into sub-patterns:



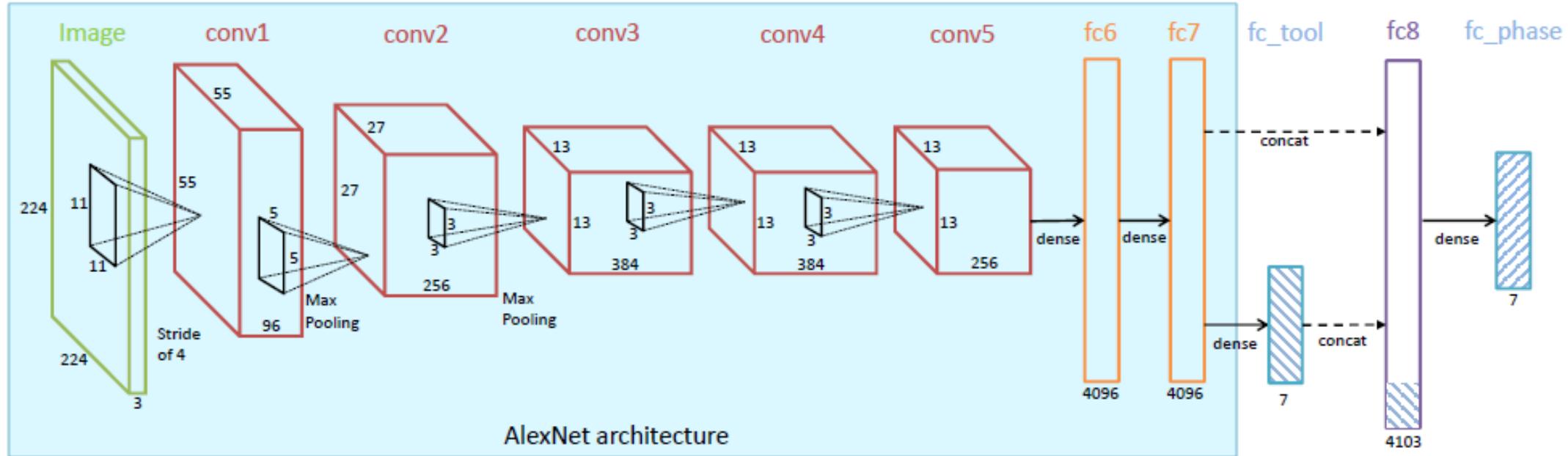
Deep representation by CNN





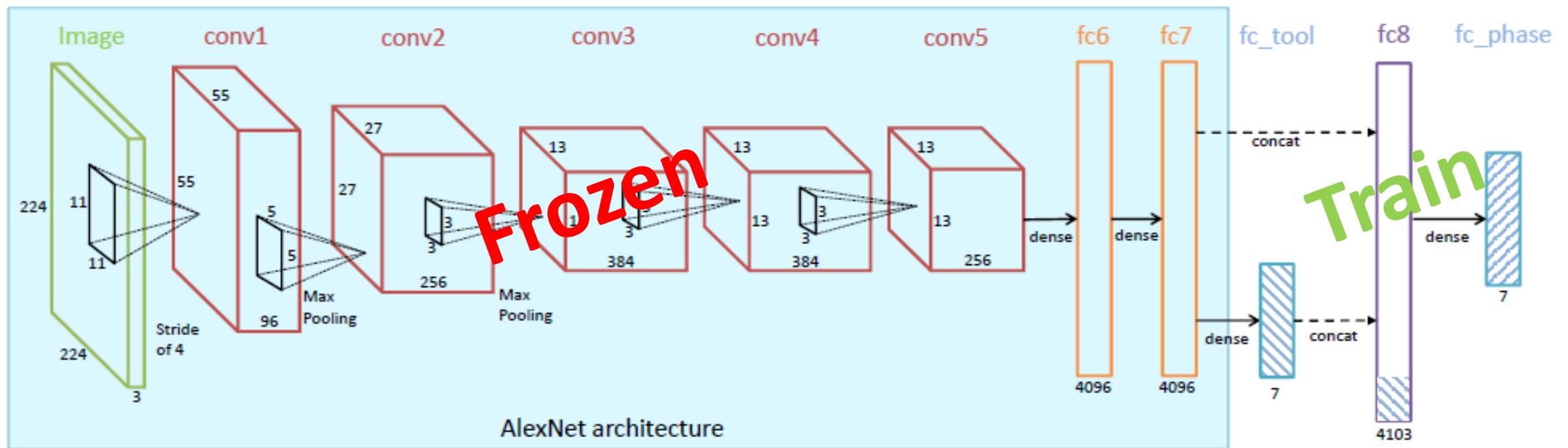
Transfer Learning

Transfer Learning



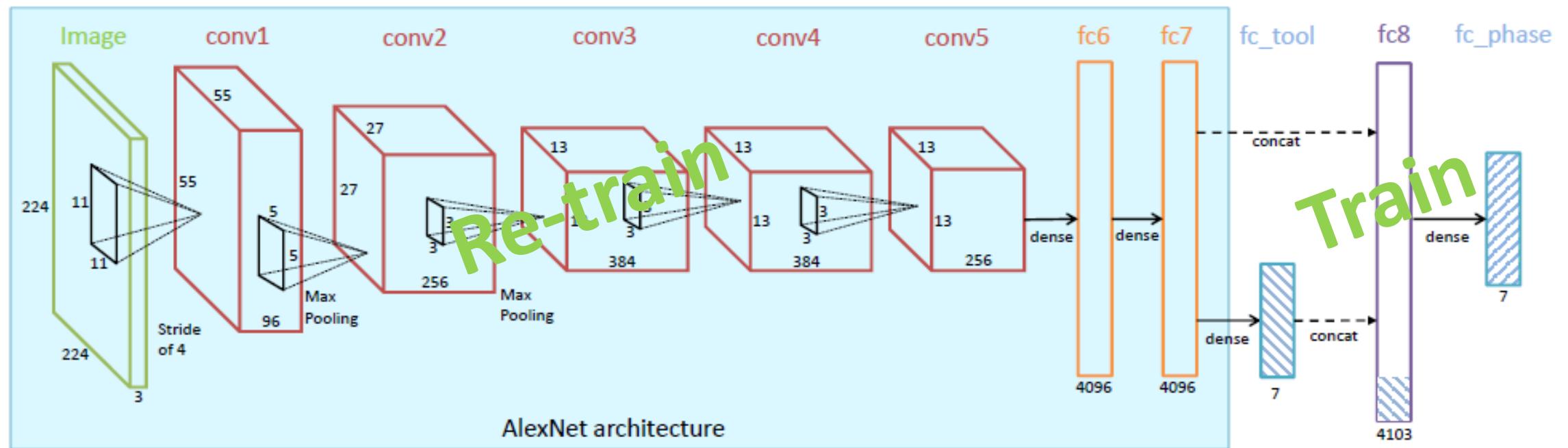
Transfer Learning!!

Very small training set



Transfer Learning!!

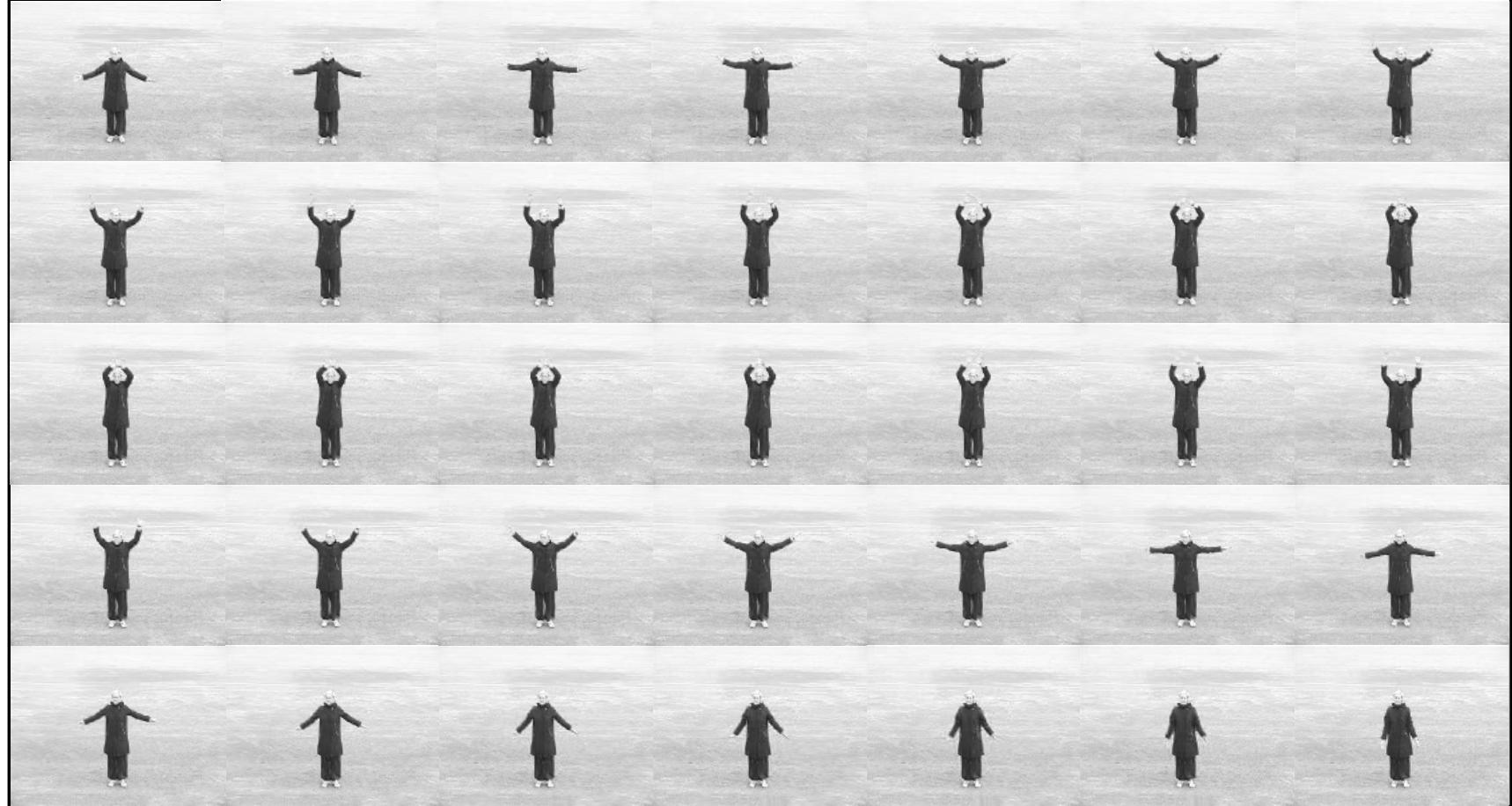
Small learning kit



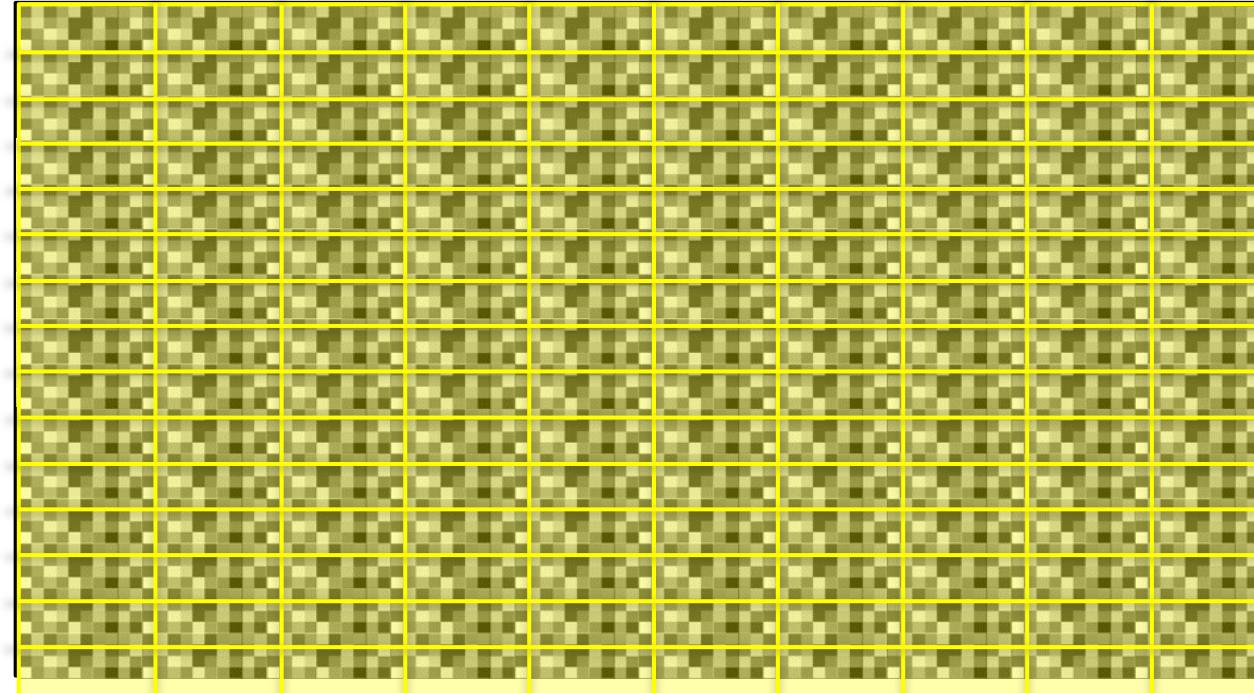
CNN for temporal data?



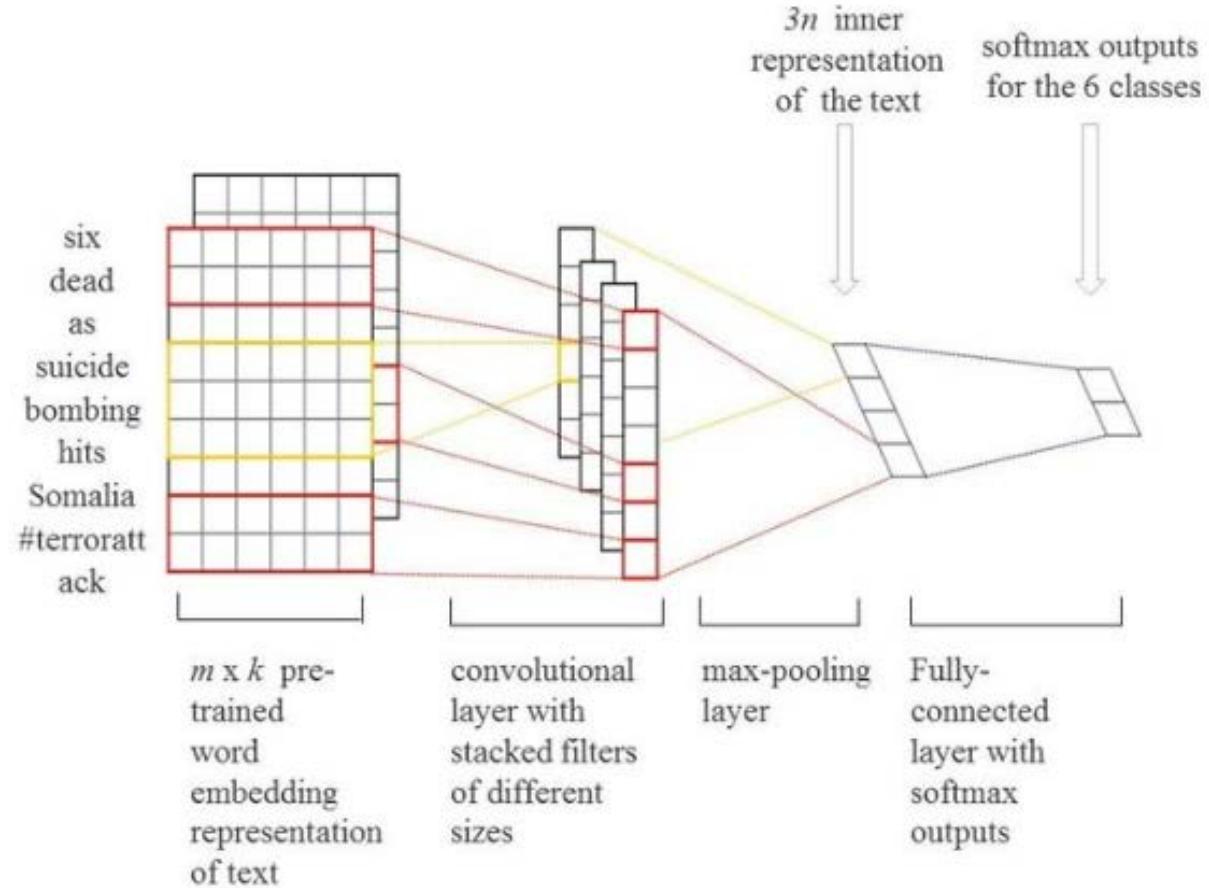
CNN for temporal data?

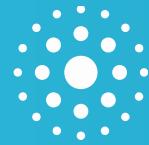


CNN for temporal data?



CNNs for Text





Sequential correlations

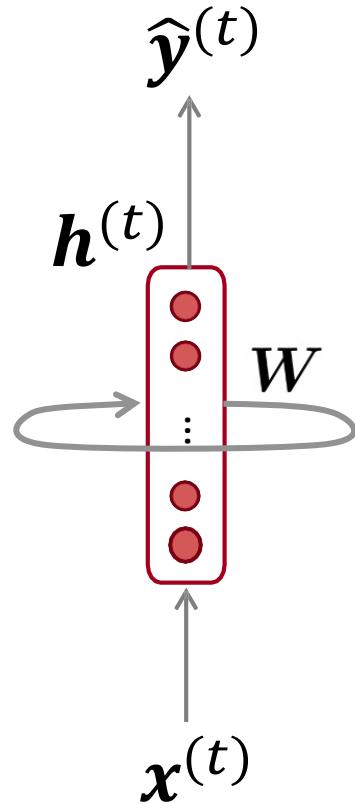
**RECURRENT NEURAL NETWORKS
(AKA RNN, LSTM / GRU)**



Recurrent Neural Networks (RNN)

A family of neural architectures

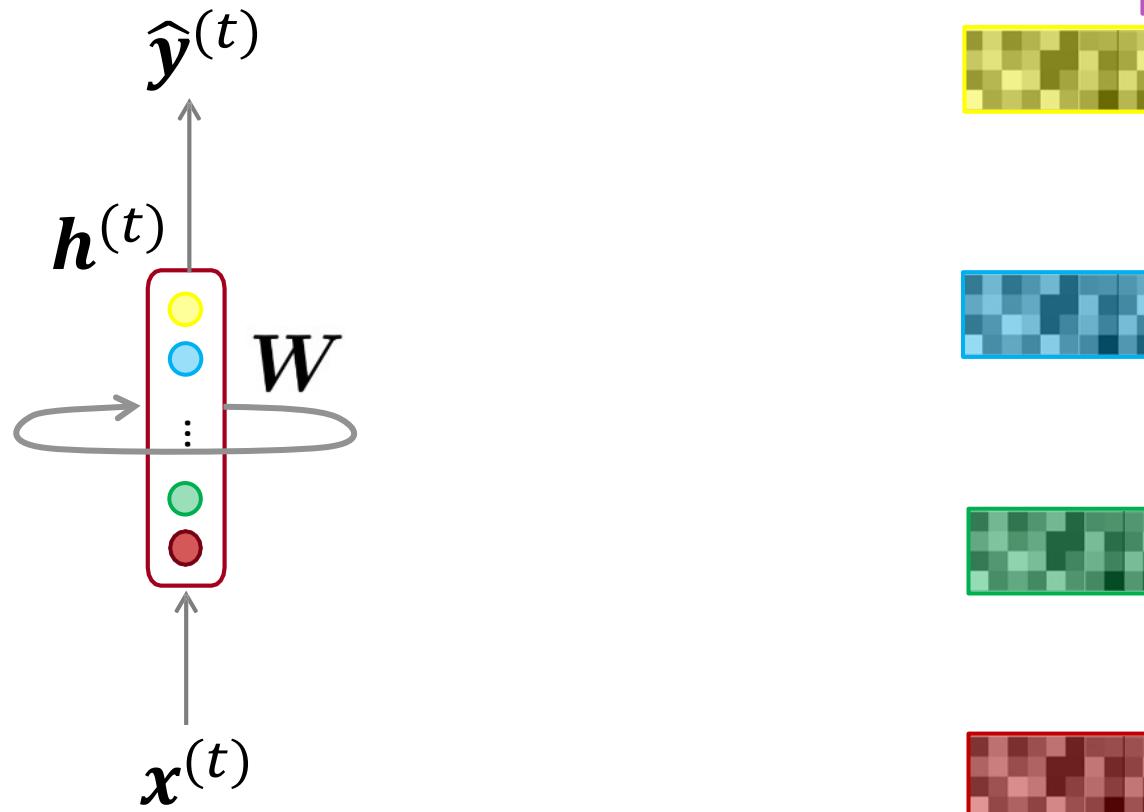
Core idea: Apply the same weights W repeatedly



Recurrent Neural Networks (RNN)

A family of neural architectures

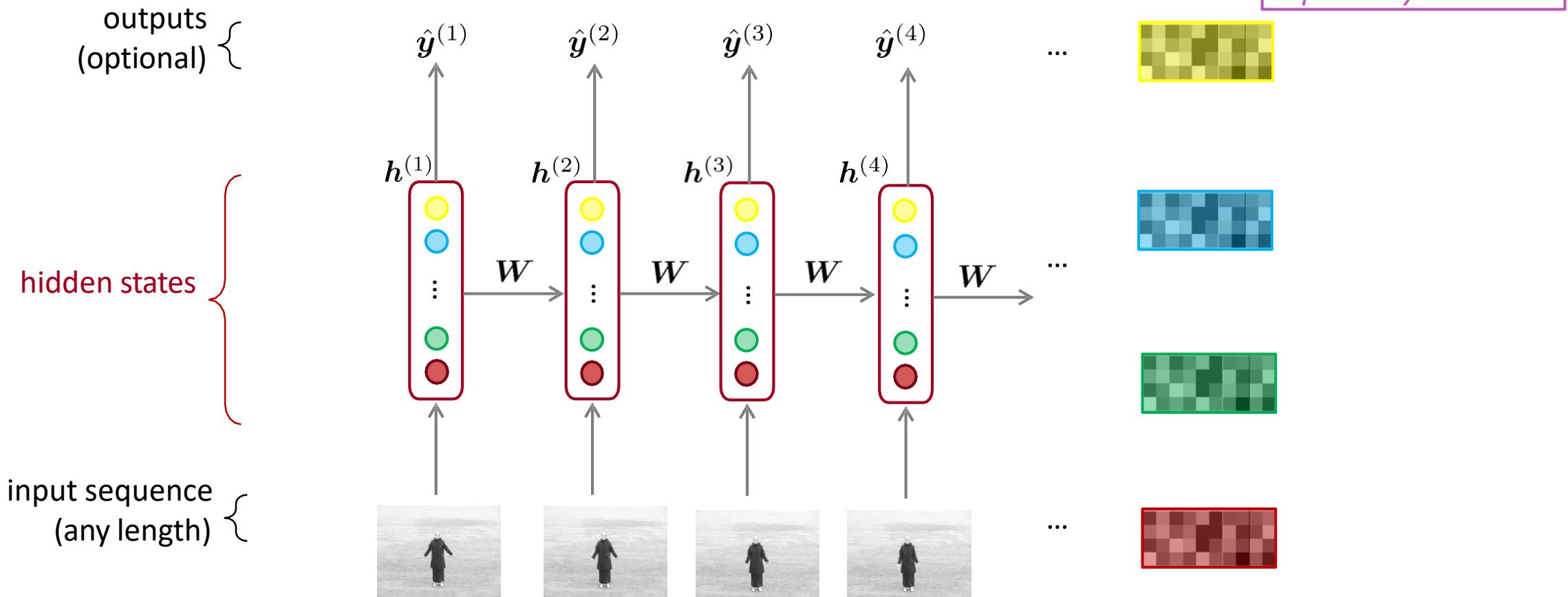
Core idea: Apply the same weights W repeatedly





Recurrent Neural Networks (RNN)

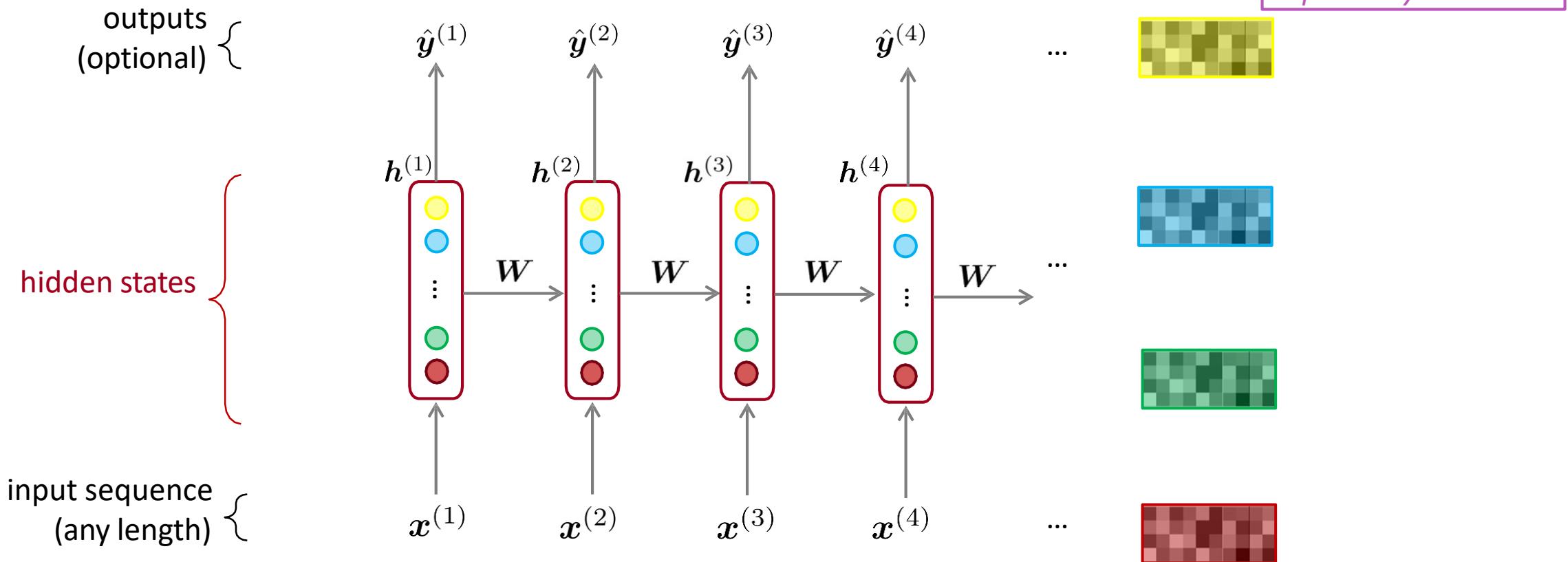
A family of neural architectures





Recurrent Neural Networks (RNN)

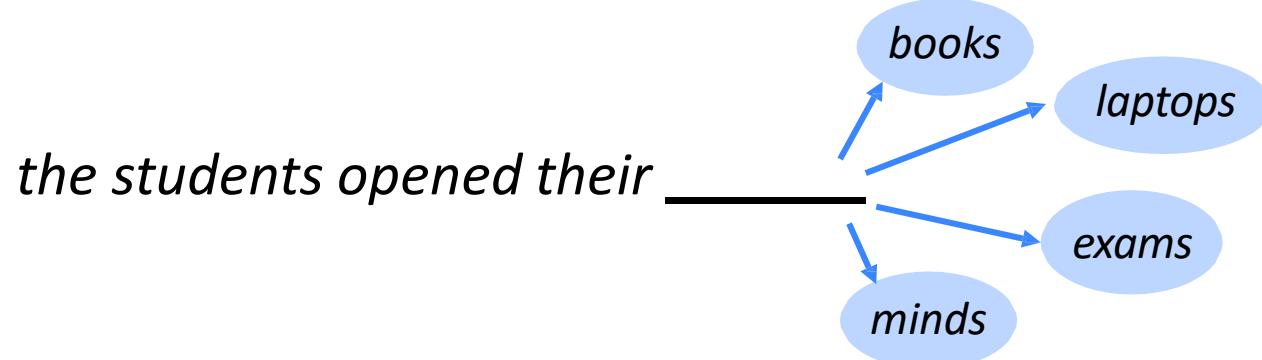
A family of neural architectures





Language Modeling

- **Language Modeling** is the task of predicting what word comes next.



- More formally: given a sequence of words $x^{(1)}, x^{(2)}, \dots, x^{(t)}$, compute the probability distribution of the next word $x^{(t+1)}$:

$$P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})$$

where $x^{(t+1)}$ can be any word in the vocabulary $V = \{w_1, \dots, w_{|V|}\}$

- A system that does this is called a **Language Model**.



n-gram Language Models

univ-cotedazur.fr

- First we make a **simplifying assumption**: $x^{(t+1)}$ depends only on the preceding $n-1$ words.

$$P(x^{(t+1)} | x^{(t)}, \dots, x^{(1)}) = P(x^{(t+1)} | \underbrace{x^{(t)}, \dots, x^{(t-n+2)}}_{n-1 \text{ words}}) \quad (\text{assumption})$$

$$\begin{aligned} \text{prob of a n-gram} &\rightarrow P(x^{(t+1)}, x^{(t)}, \dots, x^{(t-n+2)}) \\ \text{prob of a (n-1)-gram} &= P(x^{(t)}, \dots, x^{(t-n+2)}) \end{aligned} \quad (\text{definition of conditional prob})$$

- **Question:** How do we get these n -gram and $(n-1)$ -gram probabilities?
- **Answer:** By **counting** them in some large corpus of text!

$$\approx \frac{\text{count}(x^{(t+1)}, x^{(t)}, \dots, x^{(t-n+2)})}{\text{count}(x^{(t)}, \dots, x^{(t-n+2)})} \quad (\text{statistical approximation})$$



Sparsity Problems with n-gram Language Models

univ-cotedazur.fr

Sparsity Problem 1

Problem: What if “students opened their w ” never occurred in data? Then w has probability 0!

(Partial) Solution: Add small δ to the count for every $w \in V$. This is called *smoothing*.

$$P(w|\text{students opened their}) = \frac{\text{count}(\text{students opened their } w)}{\text{count}(\text{students opened their})}$$

Sparsity Problem 2

Problem: What if “students opened their” never occurred in data? Then we can’t calculate probability for *any* w !

(Partial) Solution: Just condition on “opened their” instead. This is called *backoff*.

Note: Increasing n makes sparsity problems worse.
Typically we can’t have n bigger than 5.



A fixed-window neural Language Model

univ-cotedazur.fr

output distribution

$$\hat{y} = \text{softmax}(U\mathbf{h} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

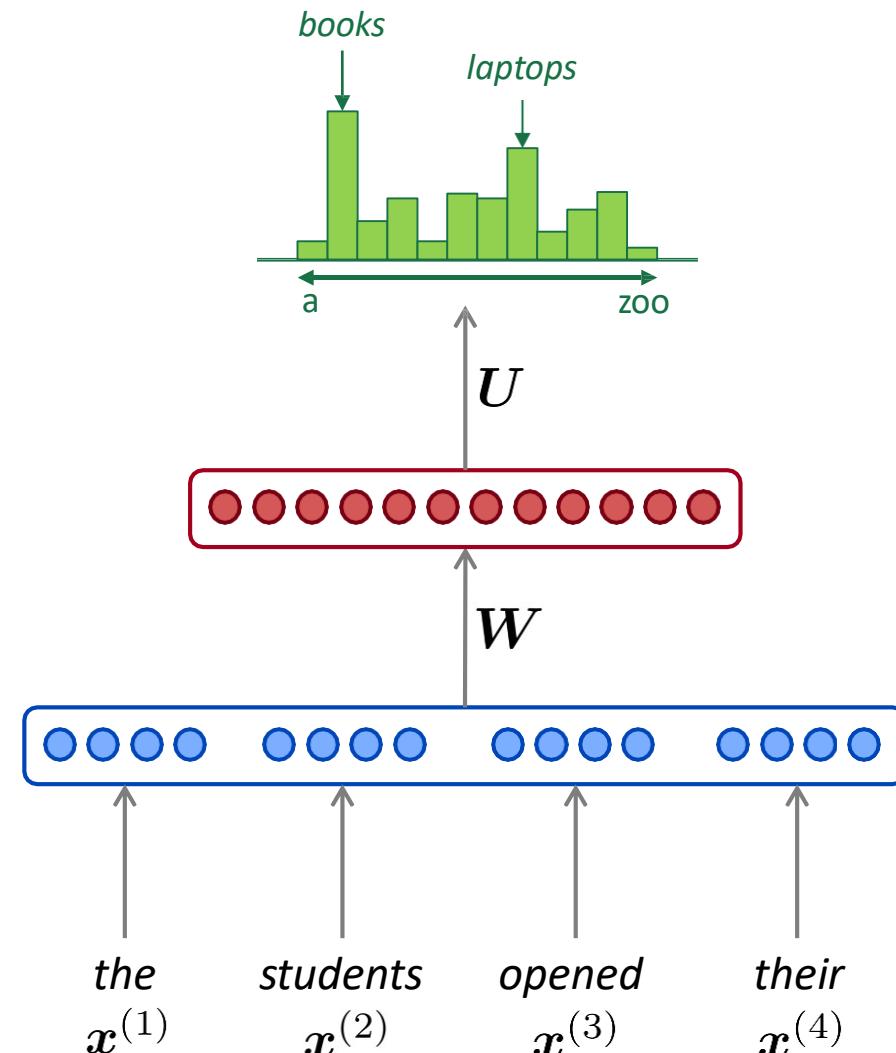
$$\mathbf{h} = f(\mathbf{W}\mathbf{e} + \mathbf{b}_1)$$

concatenated word embeddings

$$\mathbf{e} = [\mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \mathbf{e}^{(3)}; \mathbf{e}^{(4)}]$$

words / one-hot vectors

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}$$

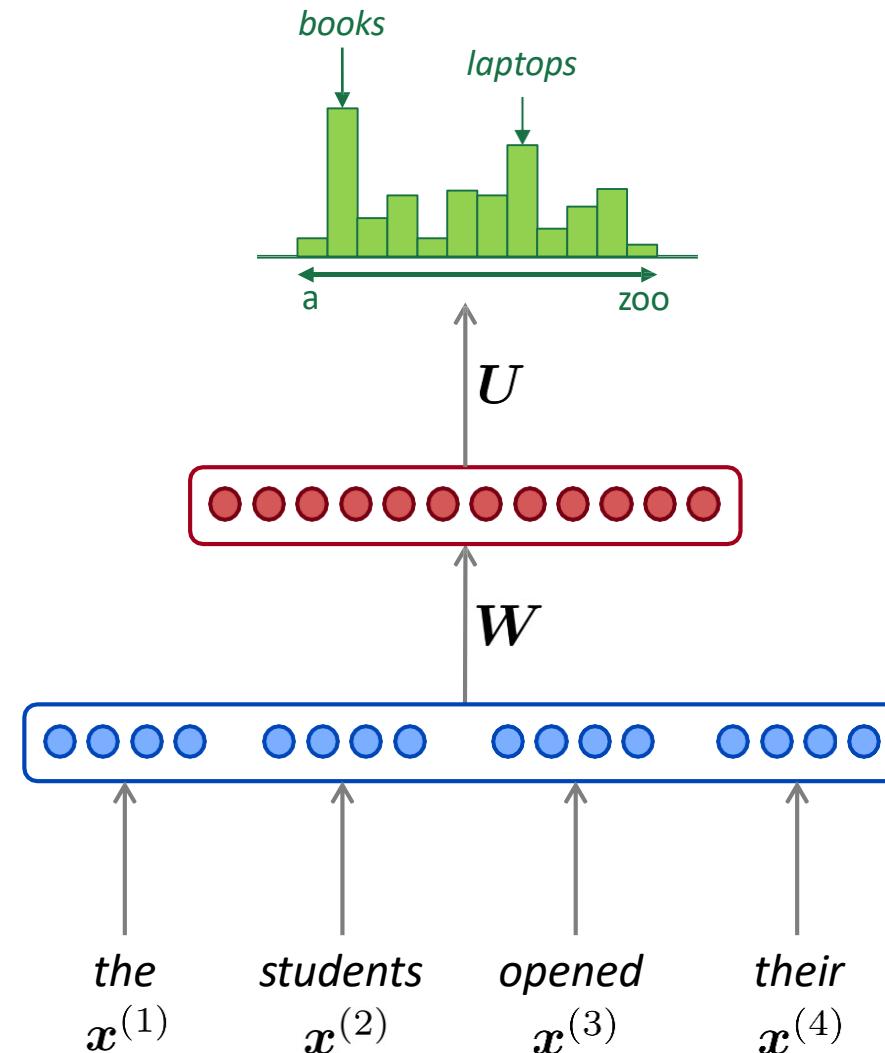


A fixed-window neural Language Model (as CNN)

Remaining **problems**:

- Fixed window is **too small**
- Enlarging window enlarges W
- Window can never be large enough!
- $x^{(1)}$ and $x^{(2)}$ are multiplied by completely different weights in W .
No symmetry in how the inputs are processed.

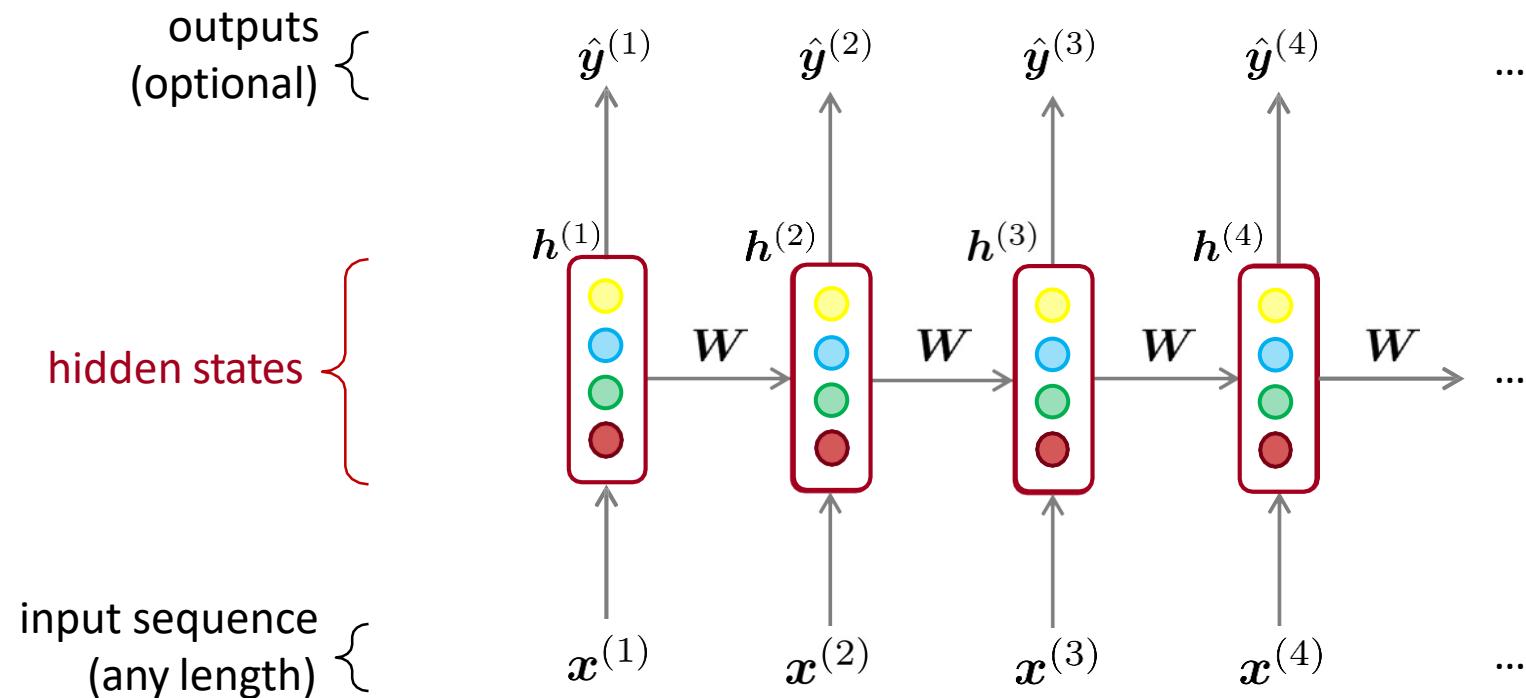
We need a neural architecture that can process *any length input*





Recurrent Neural Networks (RNN)

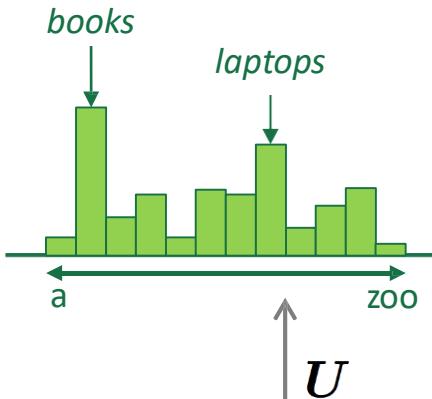
A family of neural architectures



Core idea: Apply the same weights W repeatedly

A RNN Language Model

$\hat{y}^{(4)} = P(\mathbf{x}^{(5)} | \text{the students opened their}$



output distribution

$$\hat{y}^{(t)} = \text{softmax}(\mathbf{U}h^{(t)} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma(\mathbf{W}_h h^{(t-1)} + \mathbf{W}_e e^{(t)} + \mathbf{b}_1)$$

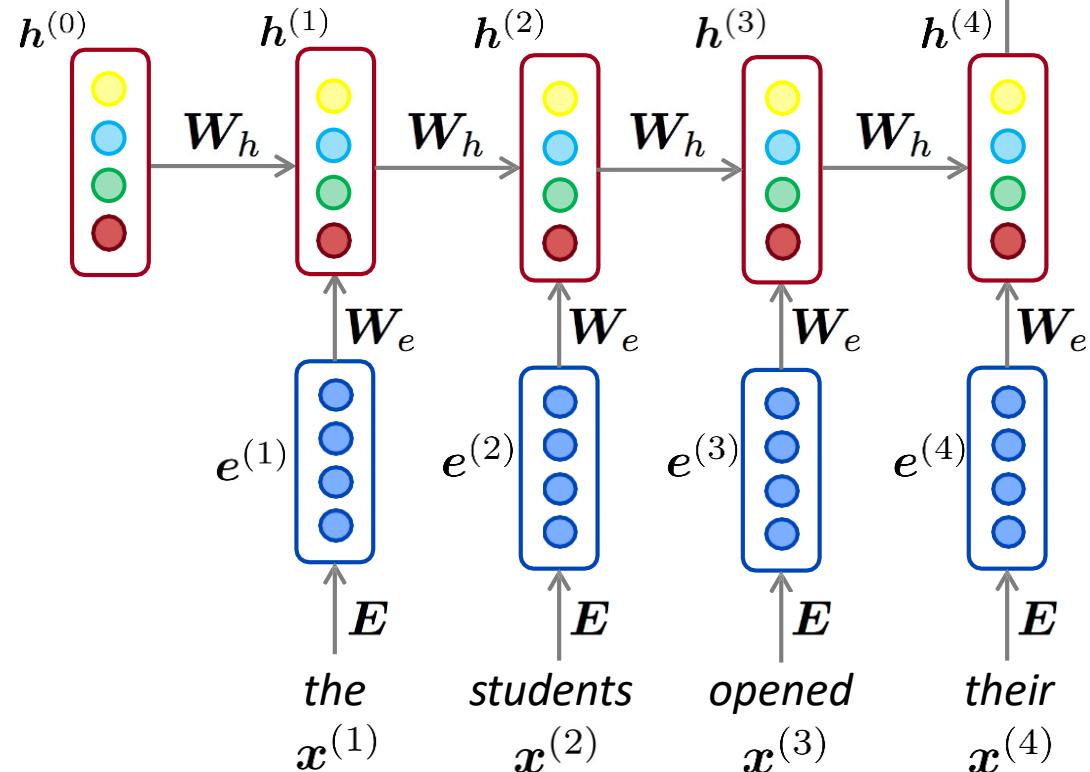
$h^{(0)}$ is the initial hidden state

word embeddings

$$e^{(t)} = \mathbf{E}x^{(t)}$$

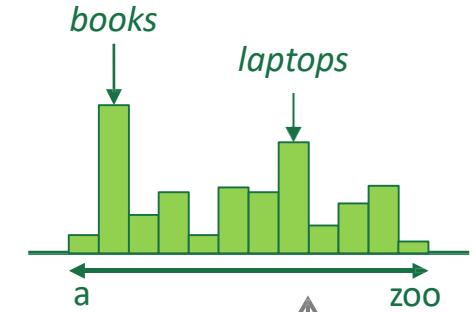
words / one-hot vectors

$$\mathbf{x}^{(t)} \in \mathbb{R}^{|V|}$$



Note: this input sequence could be much longer, but this slide doesn't have space!

A RNN Language Model

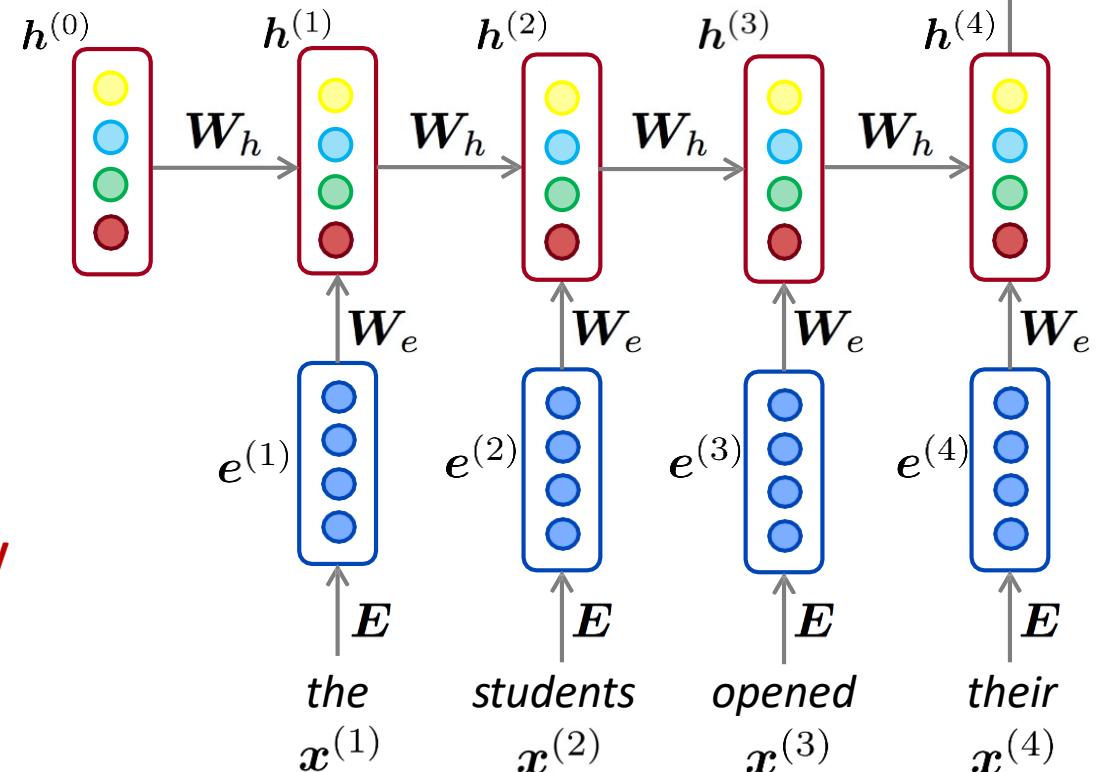
 $\hat{y}^{(4)} = P(\mathbf{x}^{(5)} | \text{the students opened their})$


RNN Advantages:

- Can process **any length** input
- Computation for step t can (in theory) use information from **many steps back**
- **Model size doesn't increase** for longer input
- Same weights applied on every timestep, so there is **symmetry** in how inputs are processed

RNN Disadvantages:

- Recurrent computation is **slow**
- In practice, difficult to access information from **many steps back**





Recall: Training a RNN Language Model

- Get a **big corpus of text** which is a sequence of words $x^{(1)}, \dots, x^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{y}^{(t)}$ **for every step t .**
 - i.e. predict probability distribution of *every word*, given words so far
- **Loss function** on step t is **cross-entropy** between predicted probability distribution $\hat{y}^{(t)}$, and the true next word $y^{(t)}$ (one-hot for $x^{(t+1)}$):

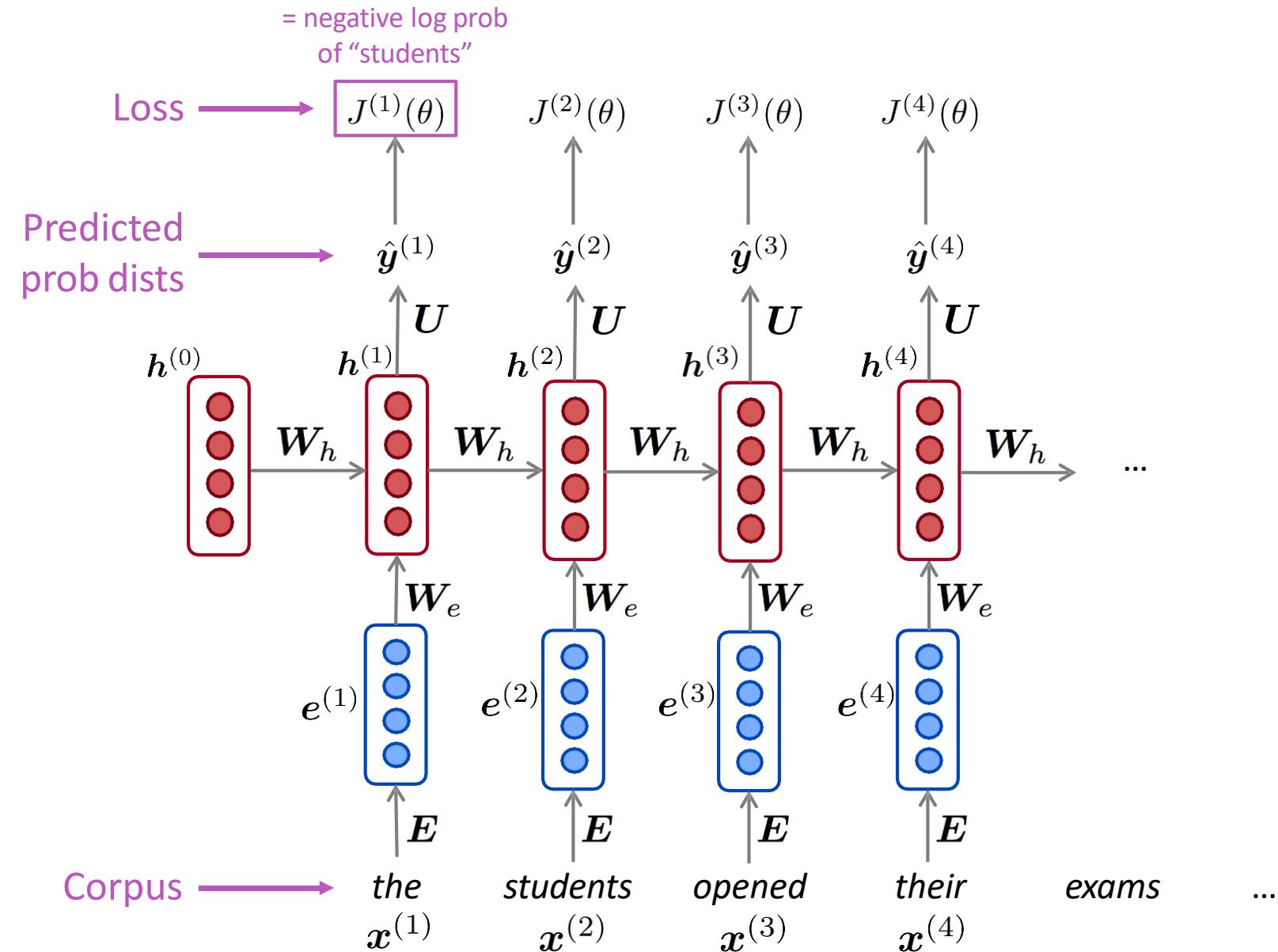
$$J^{(t)}(\theta) = CE(y^{(t)}, \hat{y}^{(t)}) = - \sum_{w \in V} y_w^{(t)} \log \hat{y}_w^{(t)} = - \log \hat{y}_{x_{t+1}}^{(t)}$$

- Average this to get **overall loss** for entire training set:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^T J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^T - \log \hat{y}_{x_{t+1}}^{(t)}$$

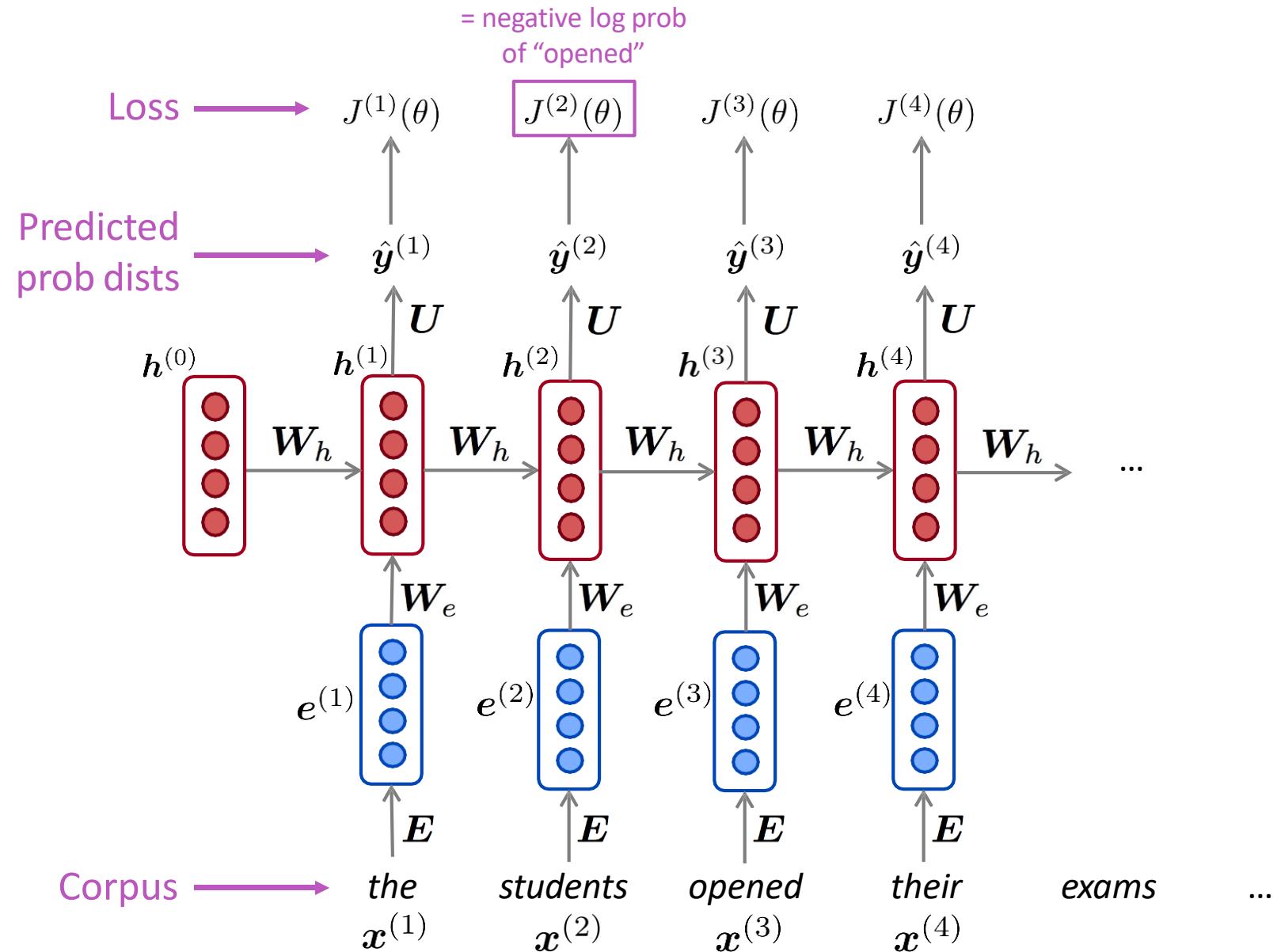


Training a RNN Language Model

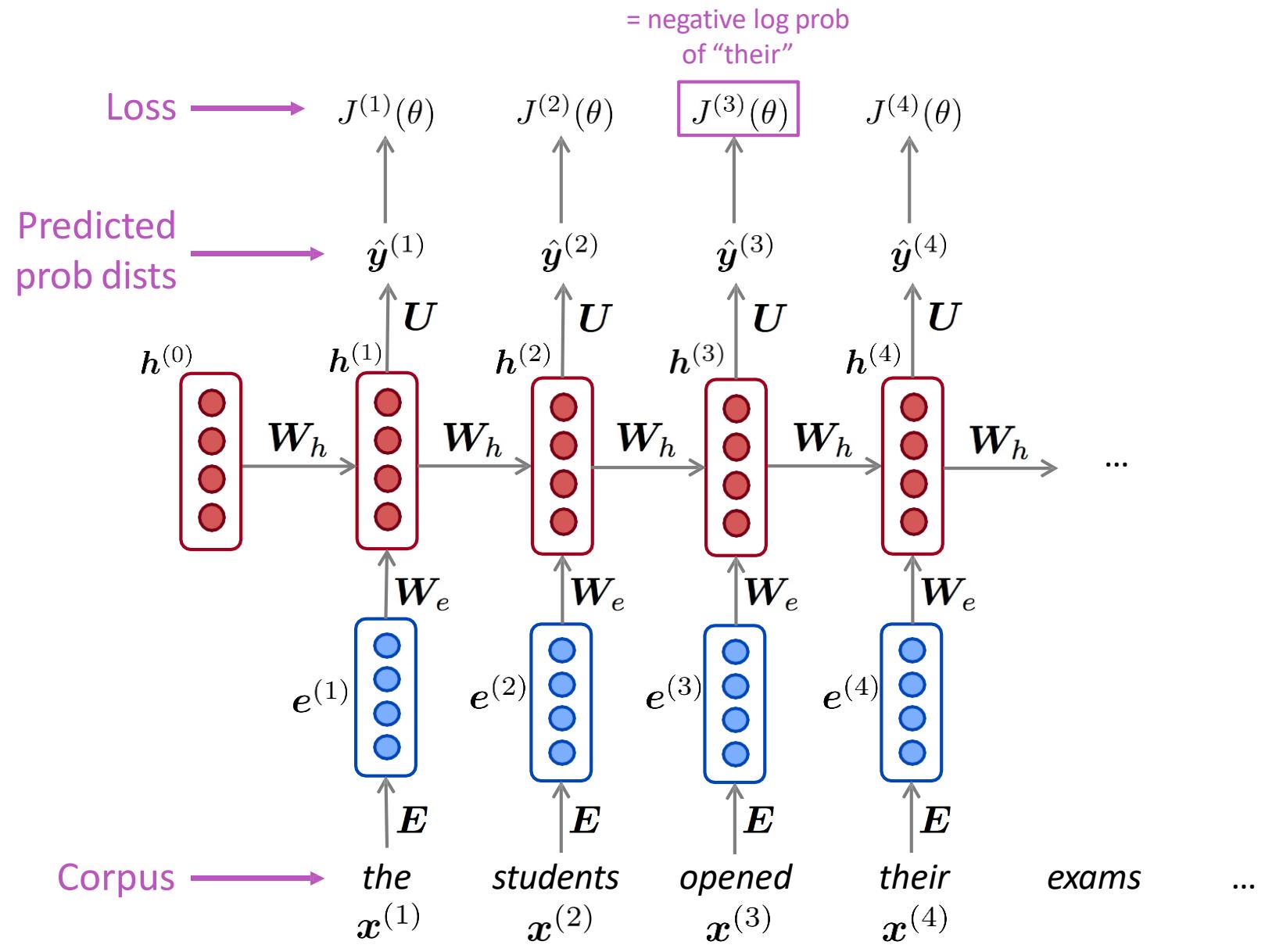




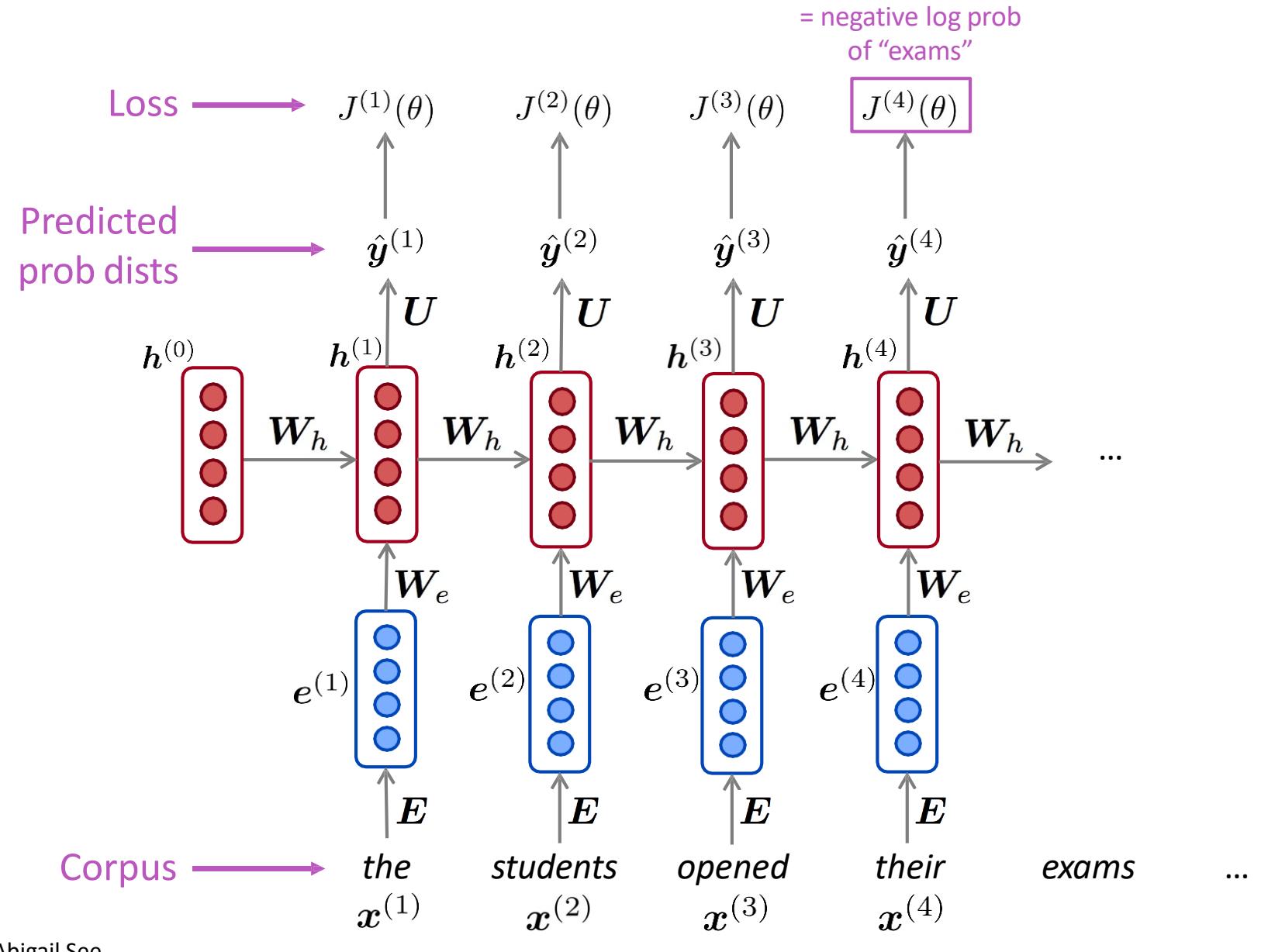
Training a RNN Language Model



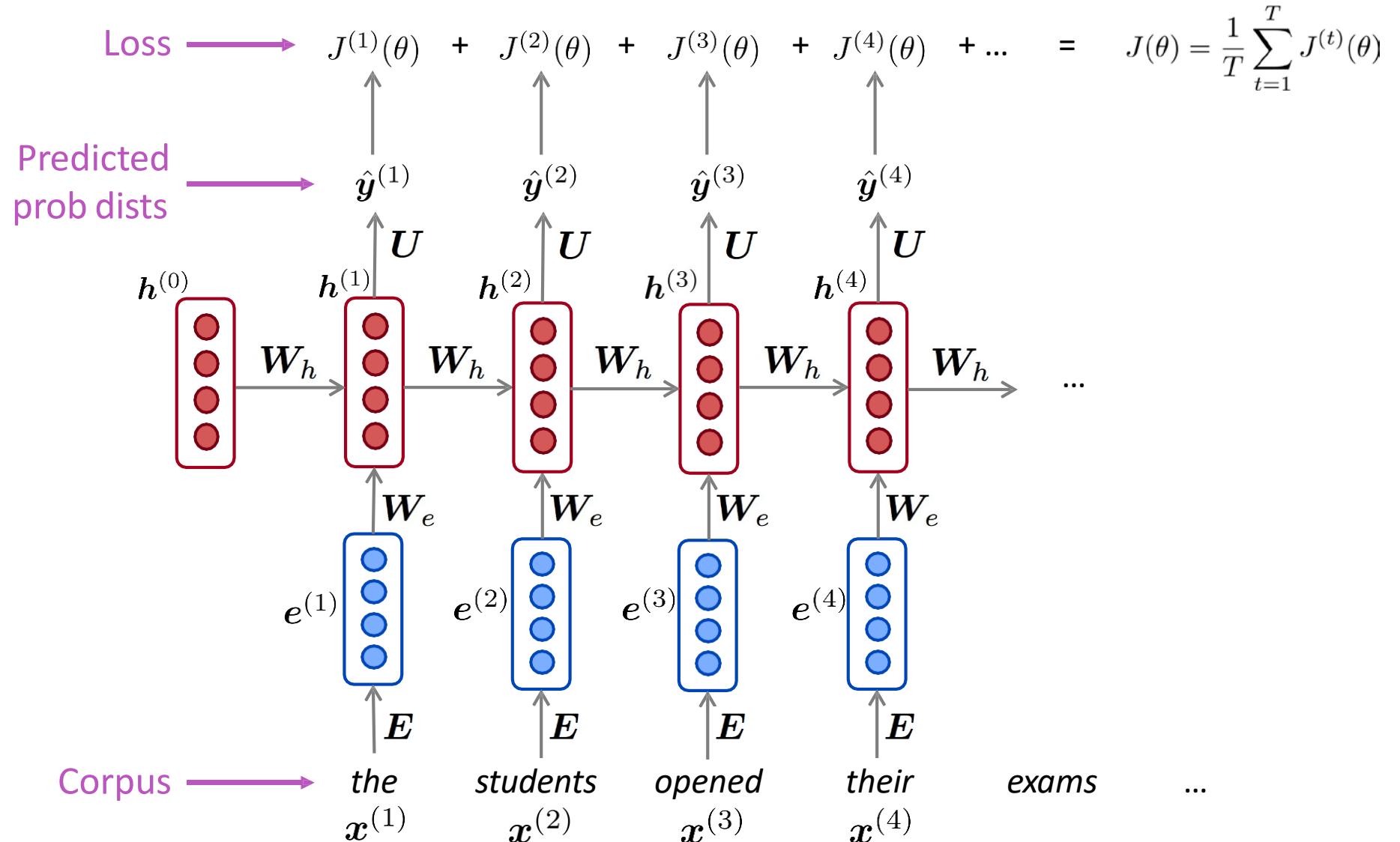
Training a RNN Language Model



Training a RNN Language Model

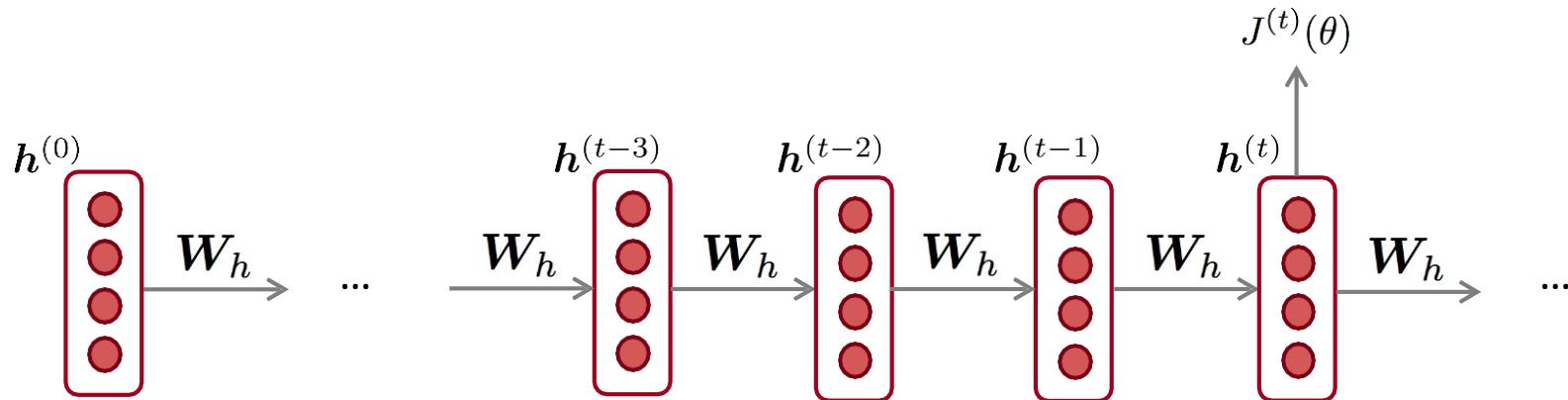


Training a RNN Language Model





Backpropagation for RNNs



Question: What's the derivative of $J^{(t)}(\theta)$ w.r.t. the **repeated** weight matrix W_h ?

Answer:
$$\frac{\partial J^{(t)}}{\partial W_h} = \sum_{i=1}^t \frac{\partial J^{(t)}}{\partial W_h} \Big|_{(i)}$$

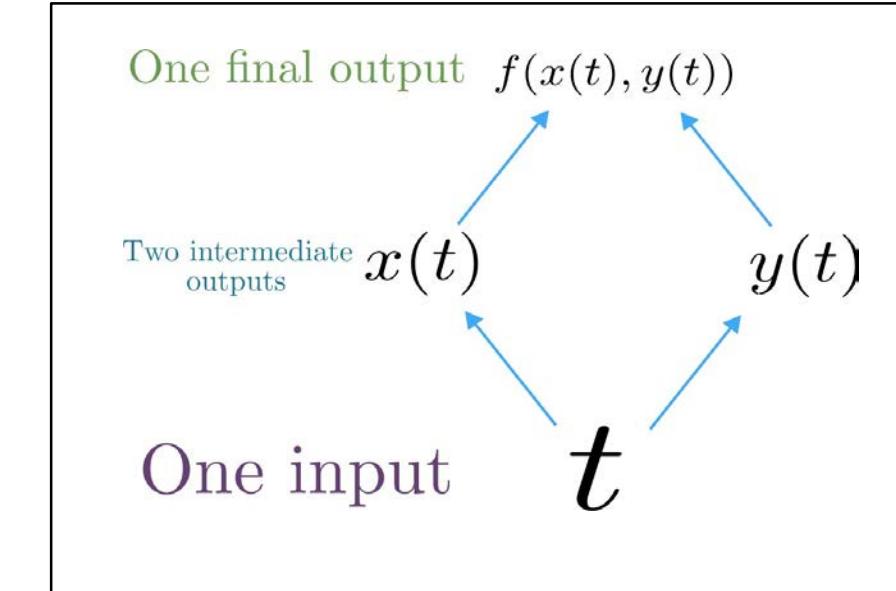
"The gradient w.r.t. a repeated weight is the sum of the gradient w.r.t. each time it appears"



Multivariable Chain Rule

- Given a multivariable function $f(x, y)$, and two single variable functions $x(t)$ and $y(t)$, here's what the multivariable chain rule says:

$$\underbrace{\frac{d}{dt} f(\textcolor{teal}{x}(t), \textcolor{red}{y}(t))}_{\text{Derivative of composition function}} = \frac{\partial f}{\partial \textcolor{teal}{x}} \frac{d\textcolor{teal}{x}}{dt} + \frac{\partial f}{\partial \textcolor{red}{y}} \frac{d\textcolor{red}{y}}{dt}$$



Source:

<https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/differentiating-vector-valued-functions/a/multivariable-chain-rule-simple-version>

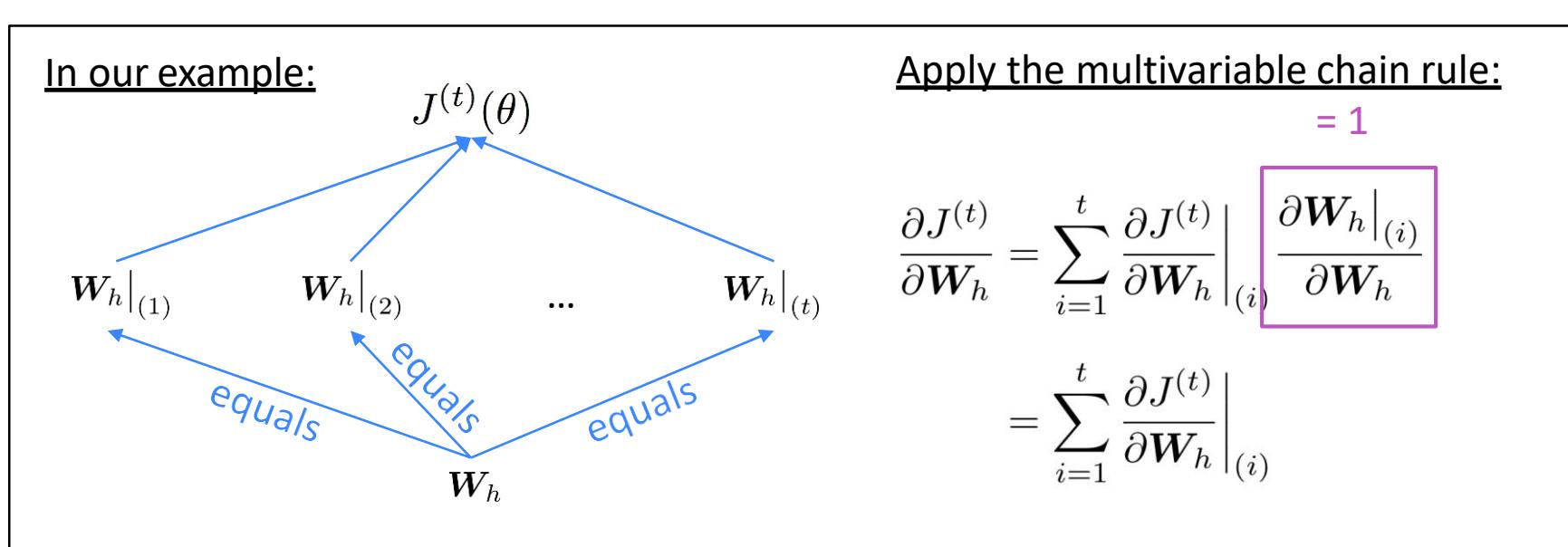


Backpropagation for RNNs: Proof sketch

univ-cotedazur.fr

- Given a multivariable function $f(x, y)$, and two single variable functions $x(t)$ and $y(t)$, here's what the multivariable chain rule says:

$$\underbrace{\frac{d}{dt} f(\textcolor{teal}{x}(t), \textcolor{red}{y}(t))}_{\text{Derivative of composition function}} = \frac{\partial f}{\partial \textcolor{teal}{x}} \frac{d\textcolor{teal}{x}}{dt} + \frac{\partial f}{\partial \textcolor{red}{y}} \frac{d\textcolor{red}{y}}{dt}$$

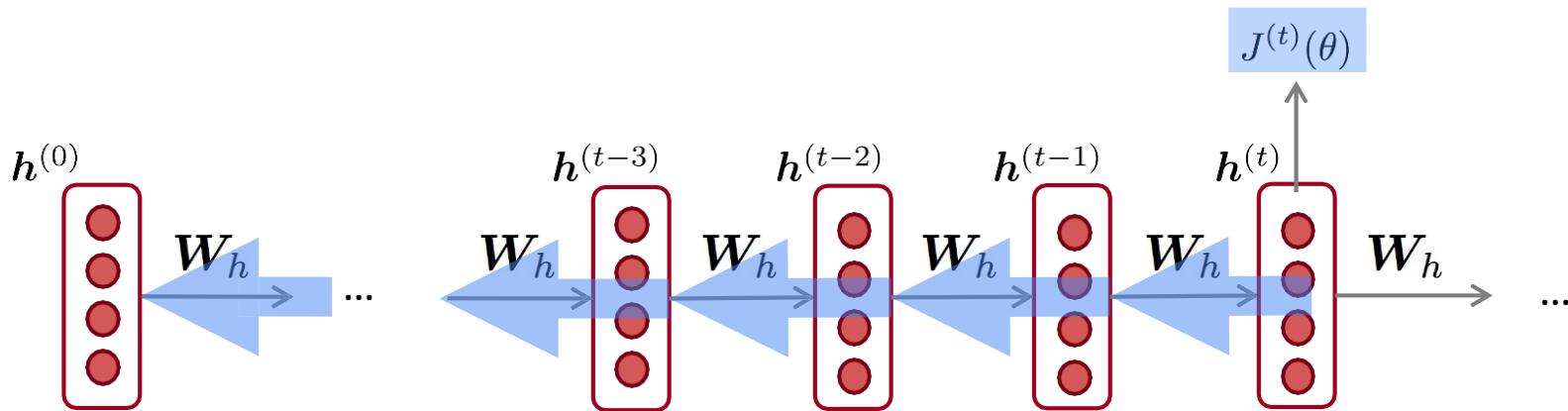


Source:

<https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/differentiating-vector-valued-functions/a/multivariable-chain-rule-simple-version>



Backpropagation for RNNs

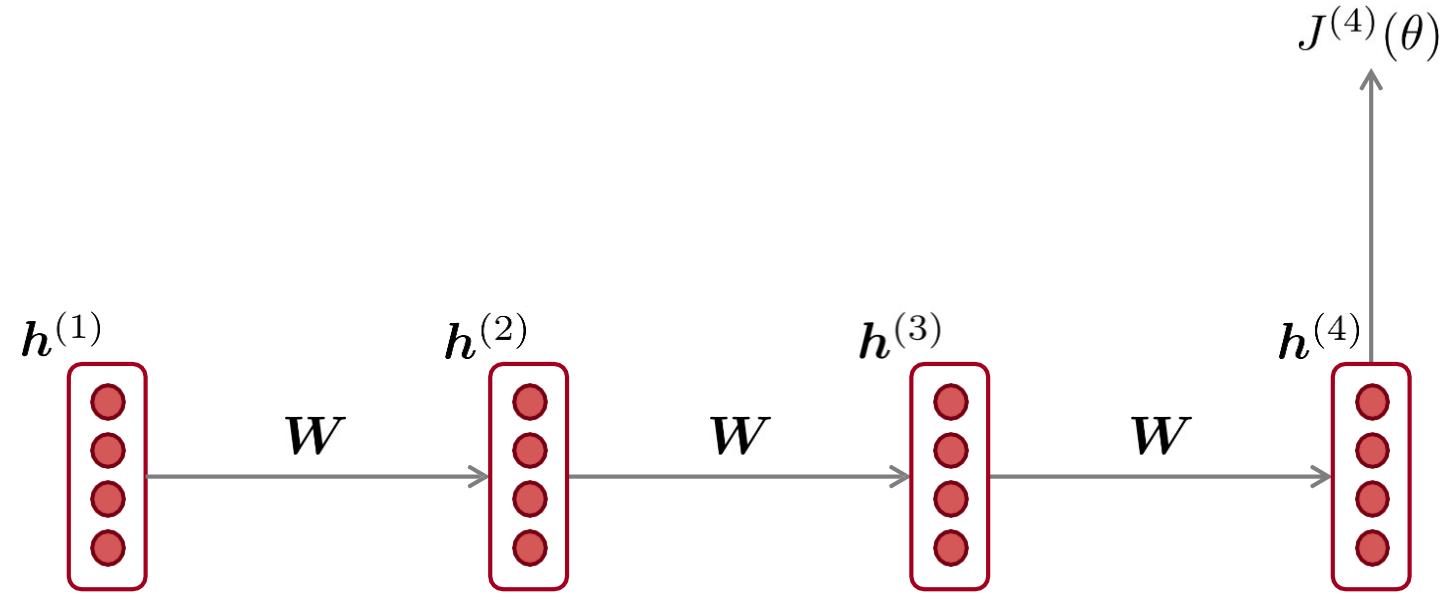


$$\frac{\partial J^{(t)}}{\partial W_h} = \sum_{i=1}^t \left. \frac{\partial J^{(t)}}{\partial W_h} \right|_{(i)}$$

Question: How do we calculate this?

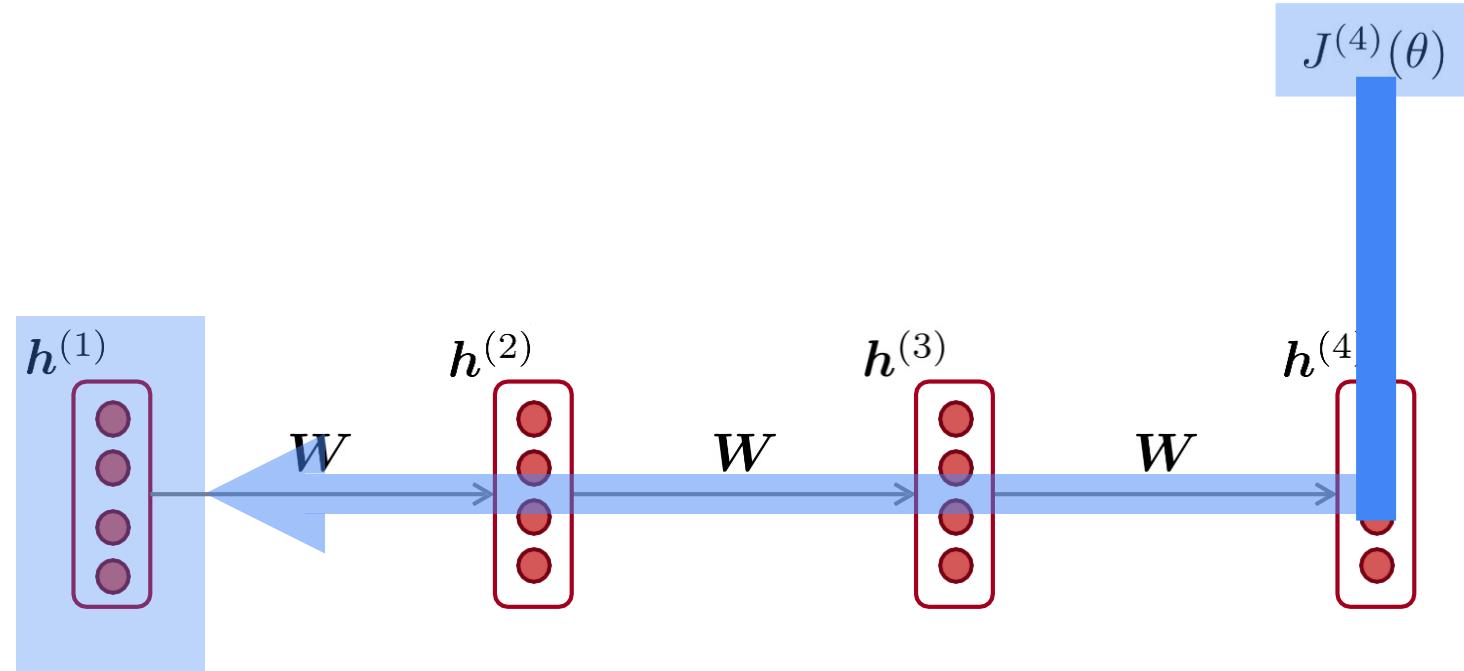
Answer: Backpropagate over timesteps $i=t, \dots, 0$, summing gradients as you go.
This algorithm is called “backpropagation through time”

Vanishing gradient intuition



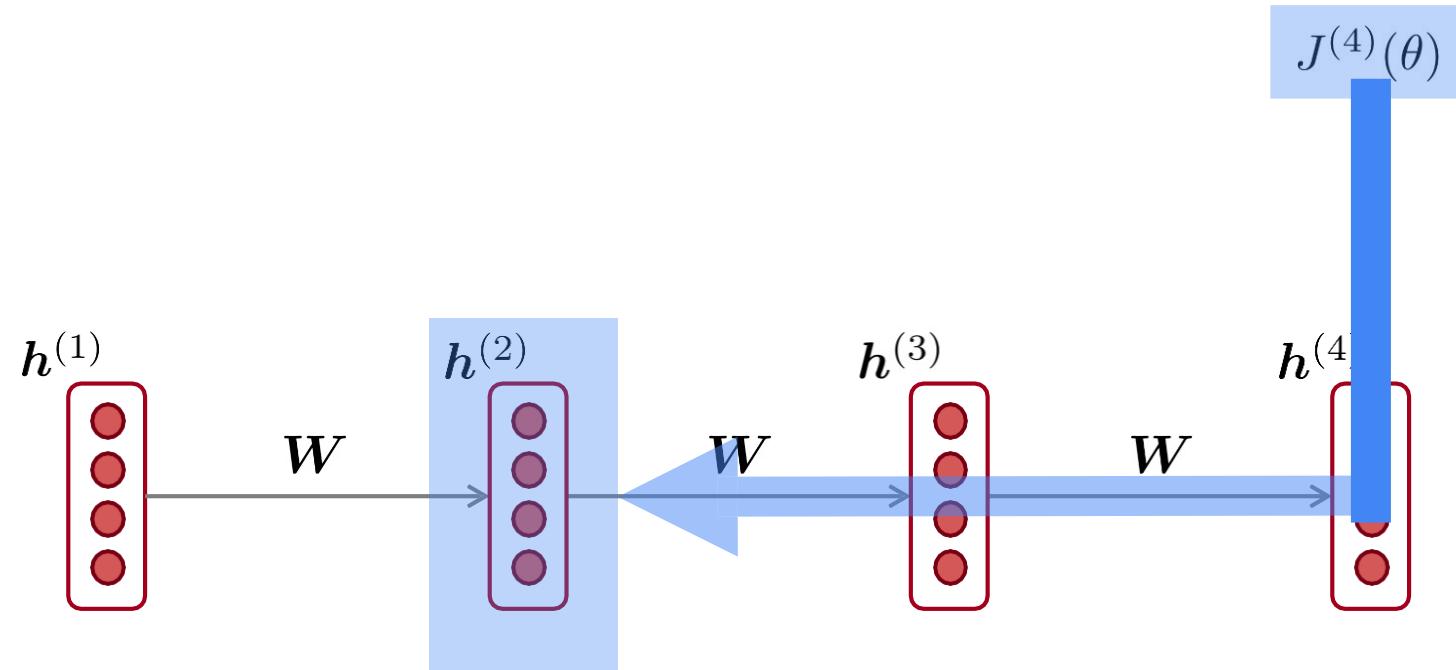


Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = ?$$

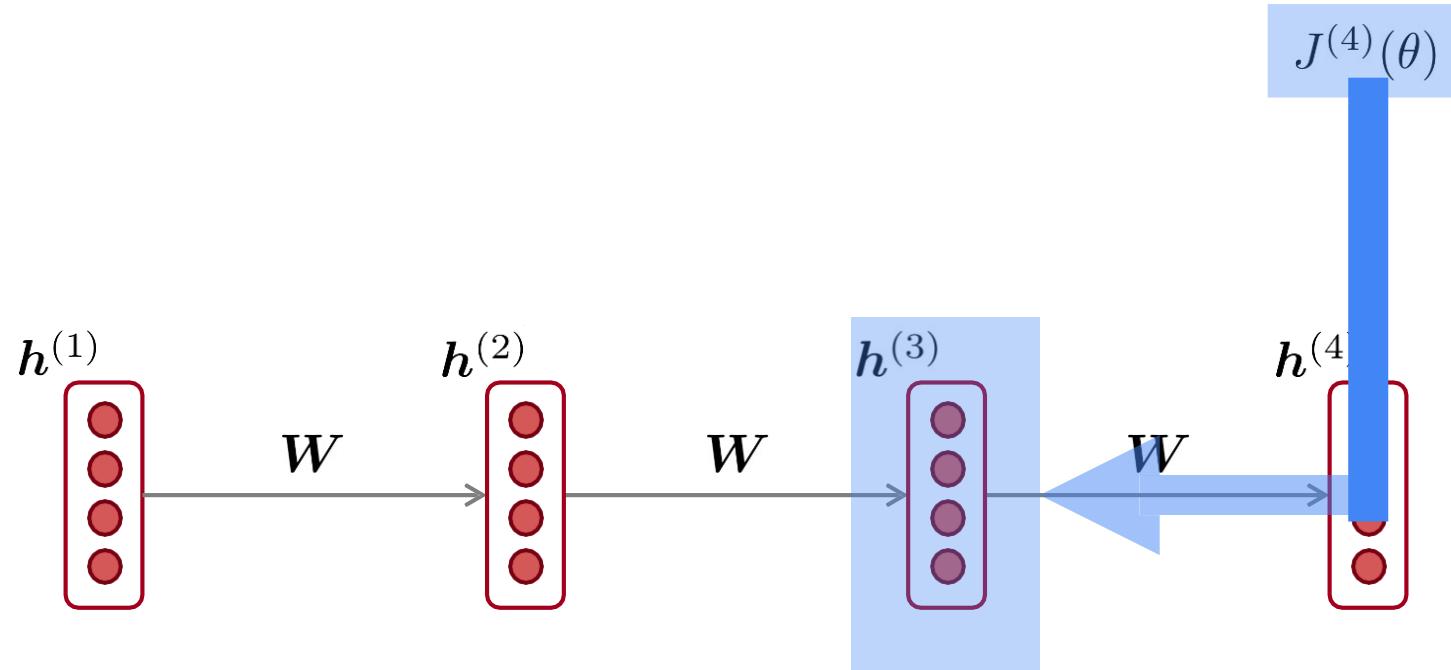
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial J^{(4)}}{\partial h^{(2)}}$$

chain rule!

Vanishing gradient intuition

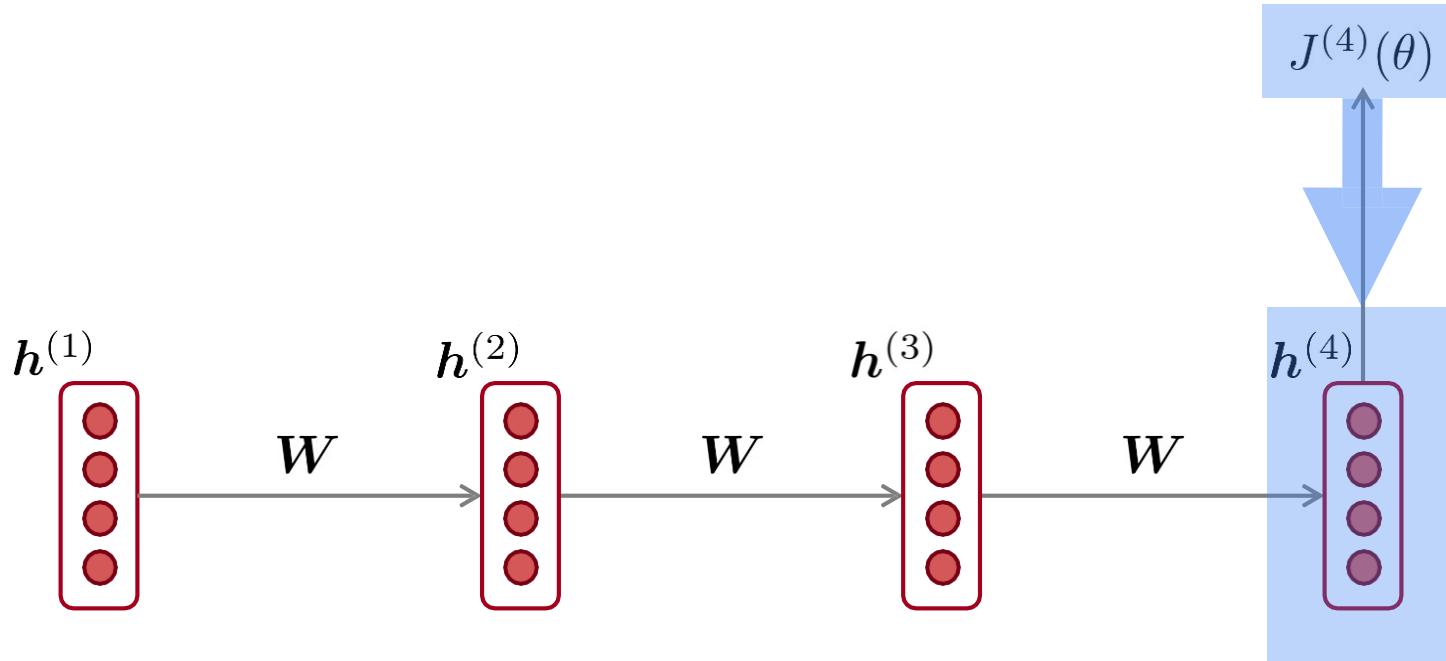


$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times$$

$$\frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial J^{(4)}}{\partial h^{(3)}}$$

chain rule!

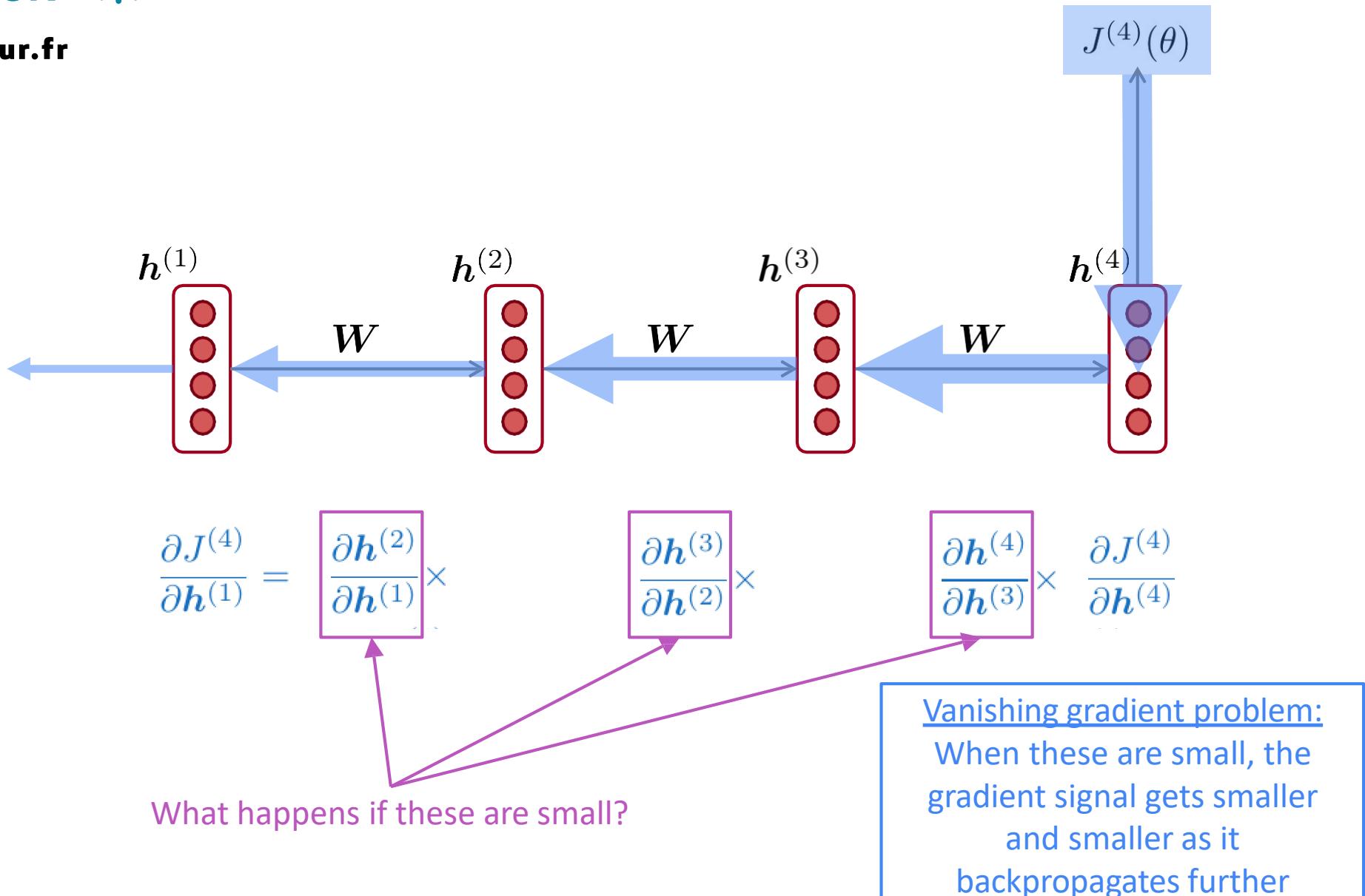
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

chain rule!

Vanishing gradient intuition





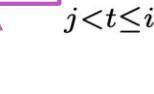
Vanishing gradient proof sketch

univ-cotedazur.fr

- Recall: $\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1)$
- Therefore: $\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} = \text{diag}\left(\sigma'\left(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1\right)\right) \mathbf{W}_h$ (chain rule)
- Consider the gradient of the loss $J^{(i)}(\theta)$ on step i , with respect to the hidden state $\mathbf{h}^{(j)}$ on some previous step j .

$$\frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(j)}} = \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \prod_{j < t \leq i} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} \quad (\text{chain rule})$$

$$= \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \boxed{\mathbf{W}_h^{(i-j)}} \prod_{j < t \leq i} \text{diag}\left(\sigma'\left(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1\right)\right) \quad (\text{value of } \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}})$$



If \mathbf{W}_h is small, then this term gets vanishingly small as i and j get further apart

Source: "On the difficulty of training recurrent neural networks", Pascanu et al, 2013. <http://proceedings.mlr.press/v28/pascanu13.pdf>



Vanishing gradient proof sketch

- Consider matrix L2 norms:

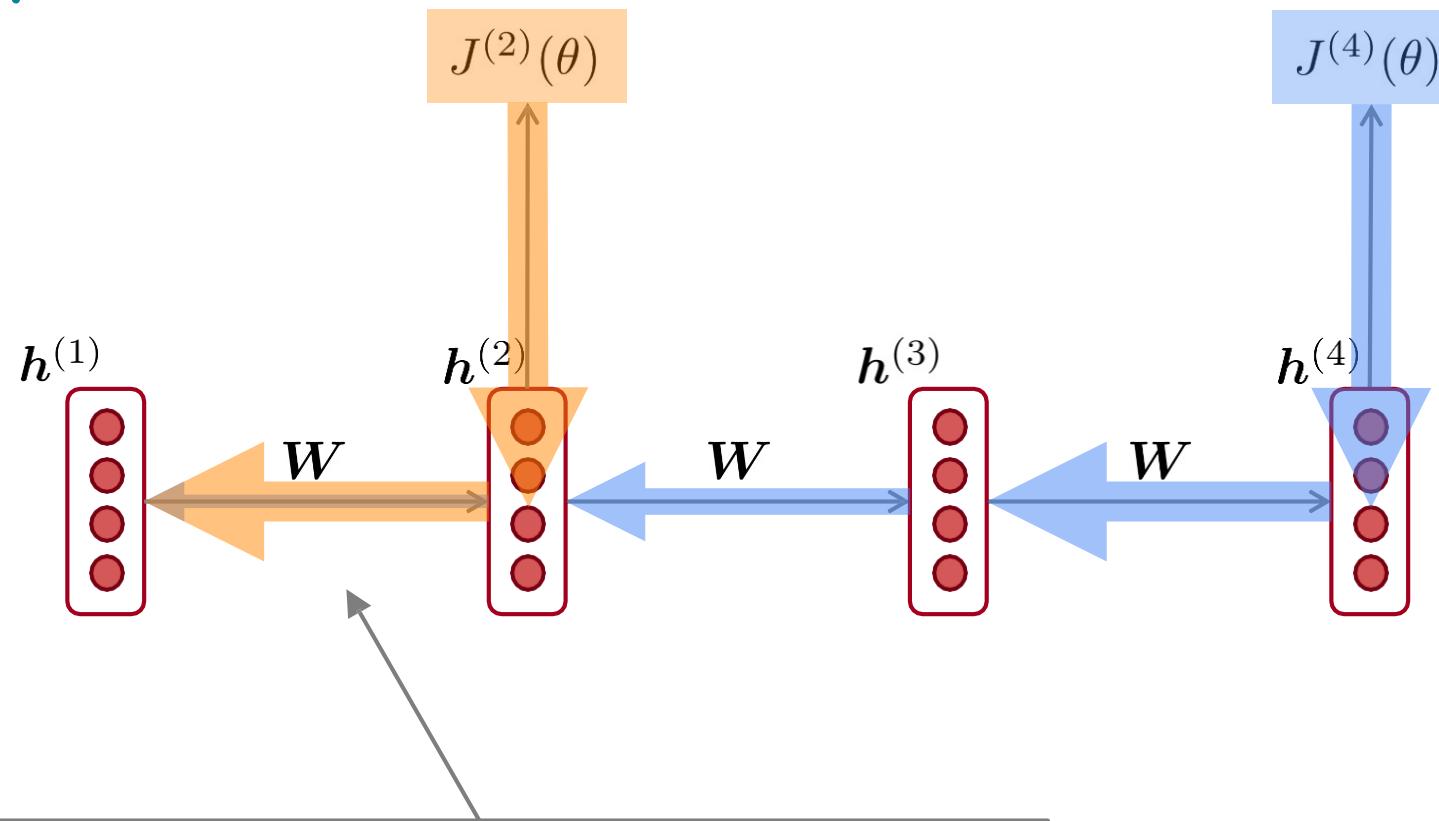
$$\left\| \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(j)}} \right\| \leq \left\| \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \right\| \|\mathbf{W}_h\|^{(i-j)} \prod_{j < t \leq i} \left\| \text{diag} \left(\sigma' \left(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1 \right) \right) \right\|$$

- Pascanu et al showed that if the largest eigenvalue of \mathbf{W}_h is less than 1, then the gradient $\left\| \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(j)}} \right\|$ will shrink exponentially
- There's a similar proof relating a largest eigenvalue >1 to exploding gradients

Source: “On the difficulty of training recurrent neural networks”, Pascanu et al, 2013.

<http://proceedings.mlr.press/v28/pascanu13.pdf>

Why is vanishing gradient a problem?



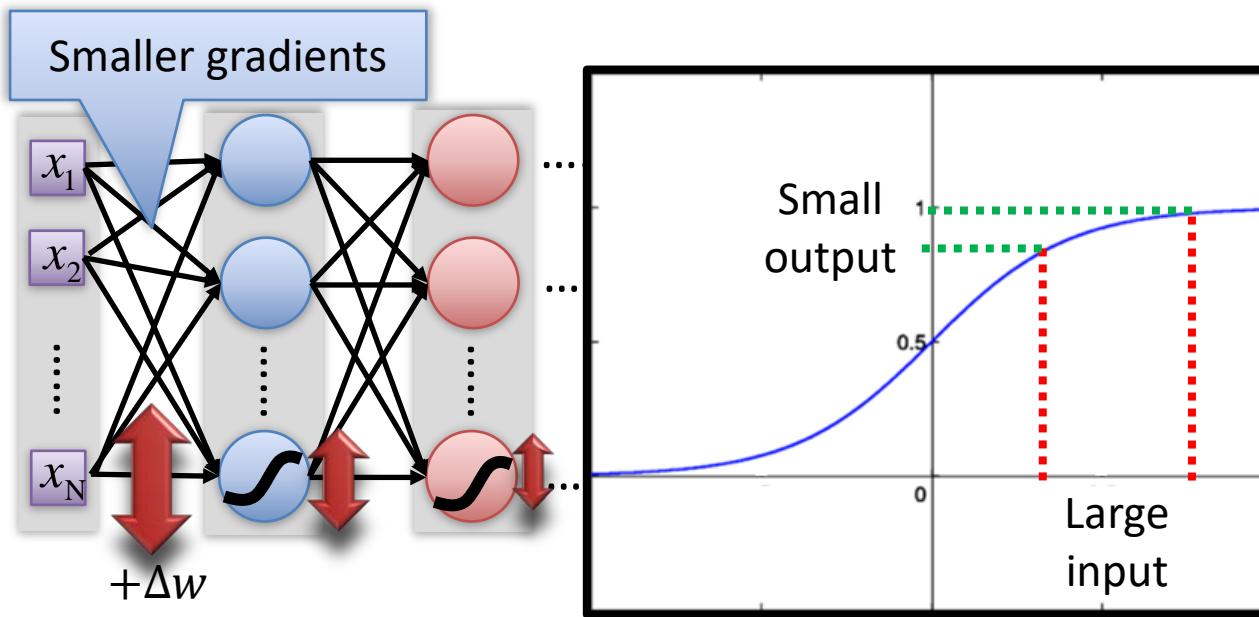
Gradient signal from faraway is lost because it's much smaller than gradient signal from close-by.

So model weights are only updated only with respect to near effects, not long-term effects.



Recipe of Deep Learning

- *Vanishing Gradient Problem*

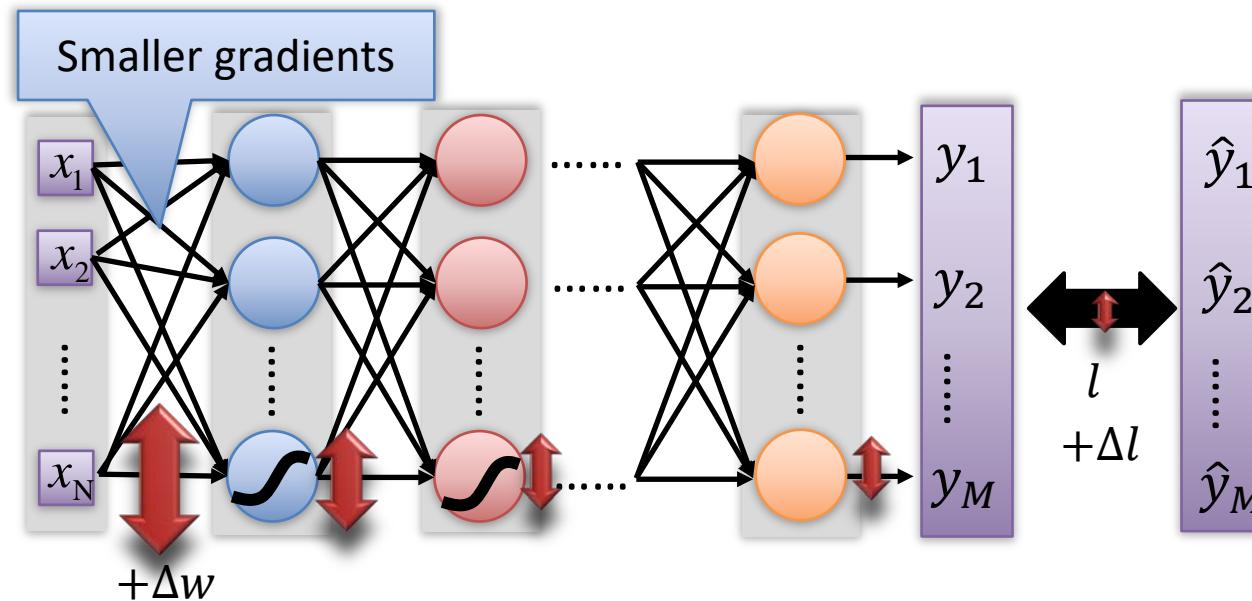


Intuitive way to compute the derivatives ...

$$\frac{\partial l}{\partial w} = ? \frac{\Delta l}{\Delta w}$$

Recipe of Deep Learning

- *Vanishing Gradient Problem*

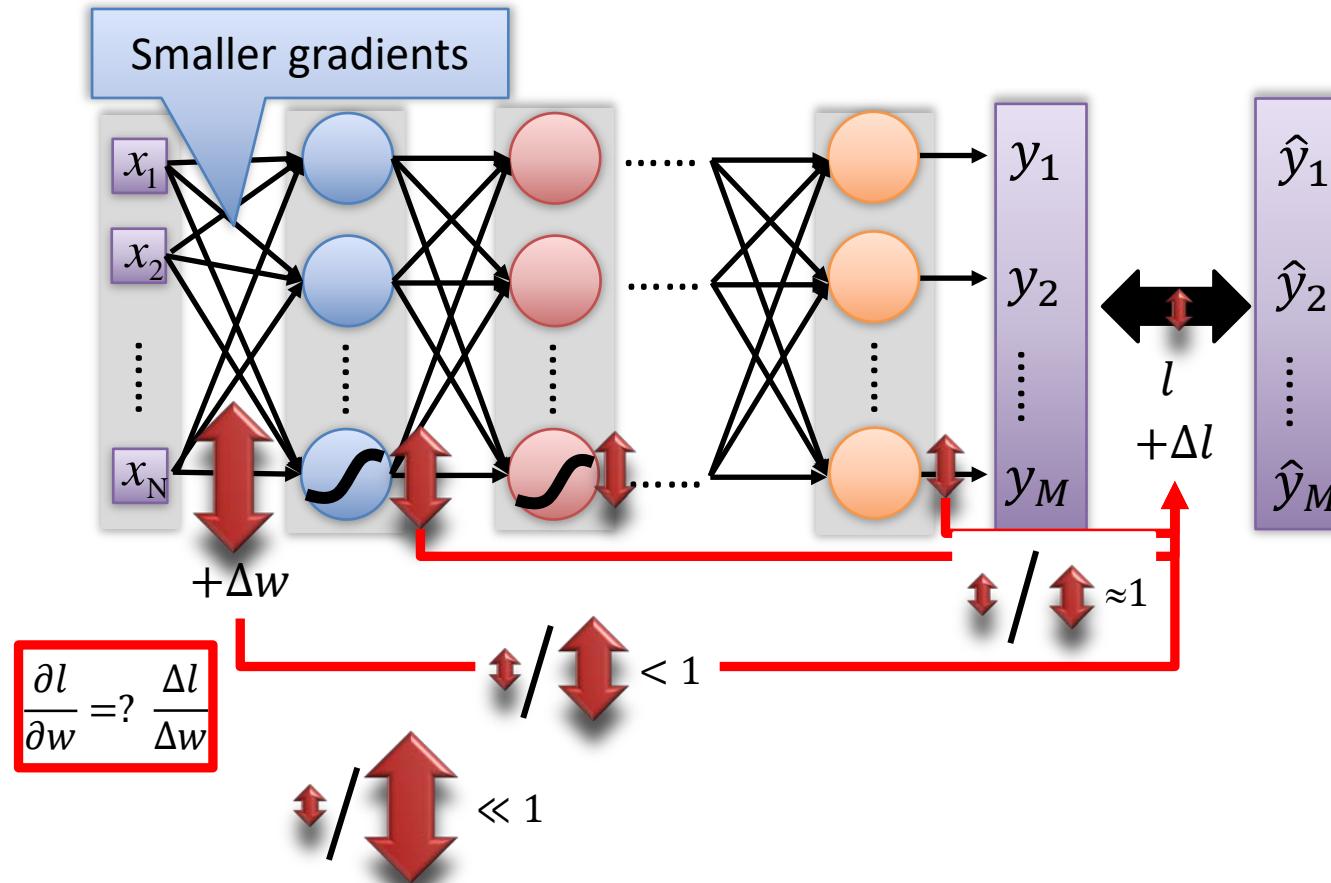


Intuitive way to compute the derivatives ...

$$\frac{\partial l}{\partial w} = ? \frac{\Delta l}{\Delta w}$$

Recipe of Deep Learning

- *Vanishing Gradient Problem*





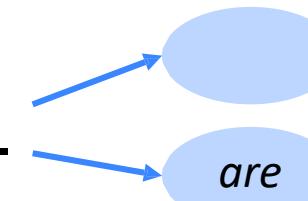
Effect of vanishing gradient on RNN-LM

- **LM task:** *When she tried to print her _____, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her tickets.*
- To learn from this training example, the RNN-LM needs to **model the dependency** between “tickets” on the 7th step and the target word “tickets” at the end.
- But if gradient is small, the model **can't learn this dependency**
 - So the model is **unable to predict similar long-distance dependencies** at test time



Effect of vanishing gradient on RNN-LM

is

- LM task: *The writer of the books* 
- Correct answer: *The writer of the books is planning a sequel*
- Syntactic recency: *The writer of the books is* (correct)
- Sequential recency: *The writer of the books are* (incorrect)
- Due to vanishing gradient, RNN-LMs are better at learning from sequential recency than syntactic recency, so they make this type of error more often than we'd like [Linzen et al 2016]

Why is exploding gradient a problem?

- If the gradient becomes too big, then the SGD update step becomes too big:

$$\theta^{new} = \theta^{old} - \overbrace{\alpha \nabla_{\theta} J(\theta)}^{\text{gradient}}$$

learning rate

- This can cause **bad updates**: we take too large a step and reach a bad parameter configuration (with large loss)
- In the worst case, this will result in **Inf** or **NaN** in your network (then you have to restart training from an earlier checkpoint)

Gradient clipping: solution for exploding gradient

- Gradient clipping: if the norm of the gradient is greater than some threshold, scale it down before applying SGD update

Algorithm 1 Pseudo-code for norm clipping

```
 $\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$ 
if  $\|\hat{\mathbf{g}}\| \geq \text{threshold}$  then
     $\hat{\mathbf{g}} \leftarrow \frac{\text{threshold}}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$ 
end if
```

- Intuition: take a step in the same direction, but a smaller step