DEEP LEARNING

Charantonis Anastase Alexandre

Assistant Professon, ENSIIE

Reasoning / intelligence

Deductive Reasoning

$A \rightarrow B \rightarrow C$

$C \rightarrow D$

$\Big\} \; A \rightarrow D$

Associative reasoning

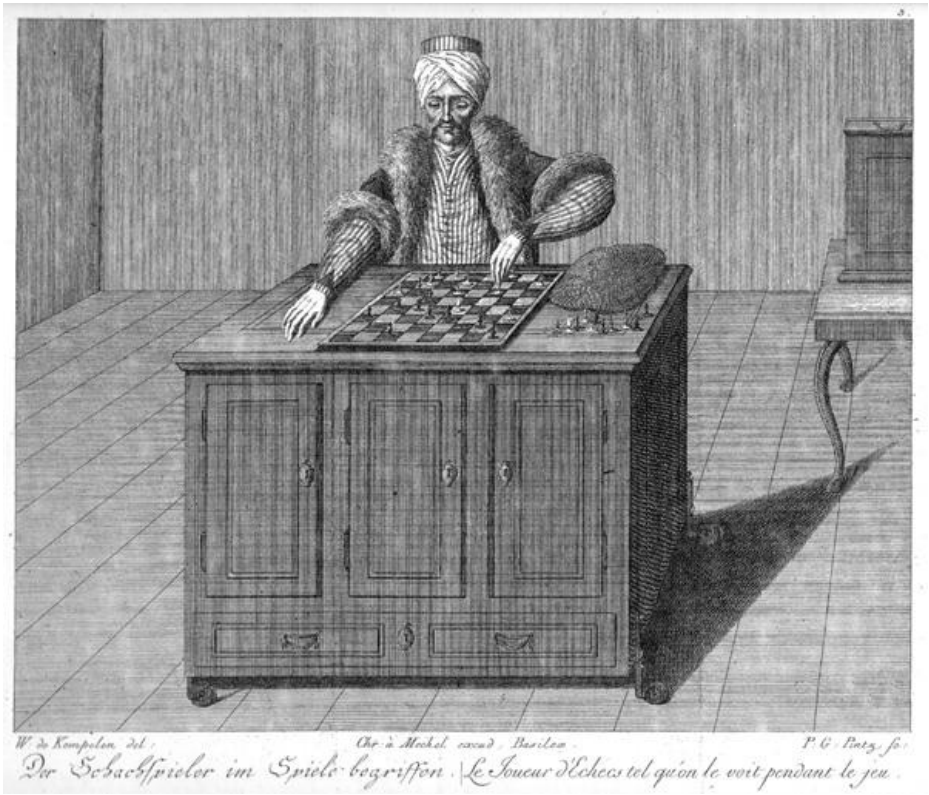
(A;D), (A;D), (A;D), (A;D)…

$A \rightarrow D$

A history of AI

Antiquité: Talos

Wikipedia



Automate des échecs:
Le Turc Méchanique

Learning

Tasks

Performance

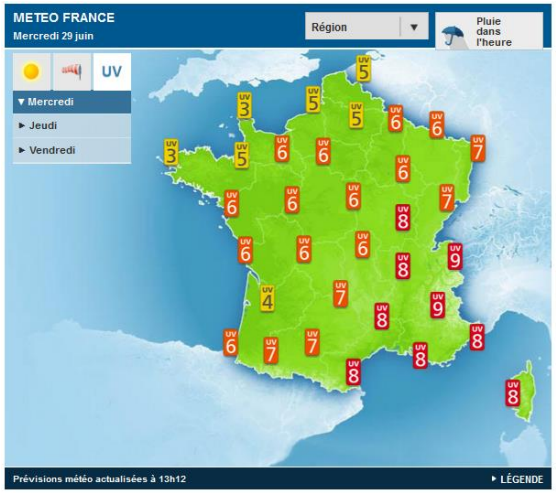
Experience

Tasks

Classification

Renée Magrit

Regression



meteofrance.fr

Tasks

Classification

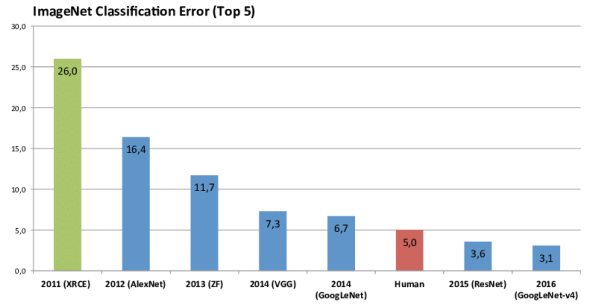
Image: Google Brain

Ancient idea(Palton)

Noticing that objects share characteristics and clustering them into groups

Basis of both mathematics and language

When is a cat no longer a cat?



Survey of neural networks in autonomous driving
July 2017

Tasks

Regression

Le Juste Prix TF1

Monty Python's Life of Brian, SONY Images

Calculate a precise output given a set of inputs.

It is used for predictions:

From weather to stock market,
From insurance to energy needs,
From «likes» to votin intentions…

In nature things have a precise value…
But observations are uncertain.

↓

How to proceed to know the actual, precise value???

Humans as a learning machine

- Task: Survival
- Performance:
  Minimization of pain,/
  Maximization of joy
- Experience:
  Personal + Transfert / Education

The cost of learning

Questions:

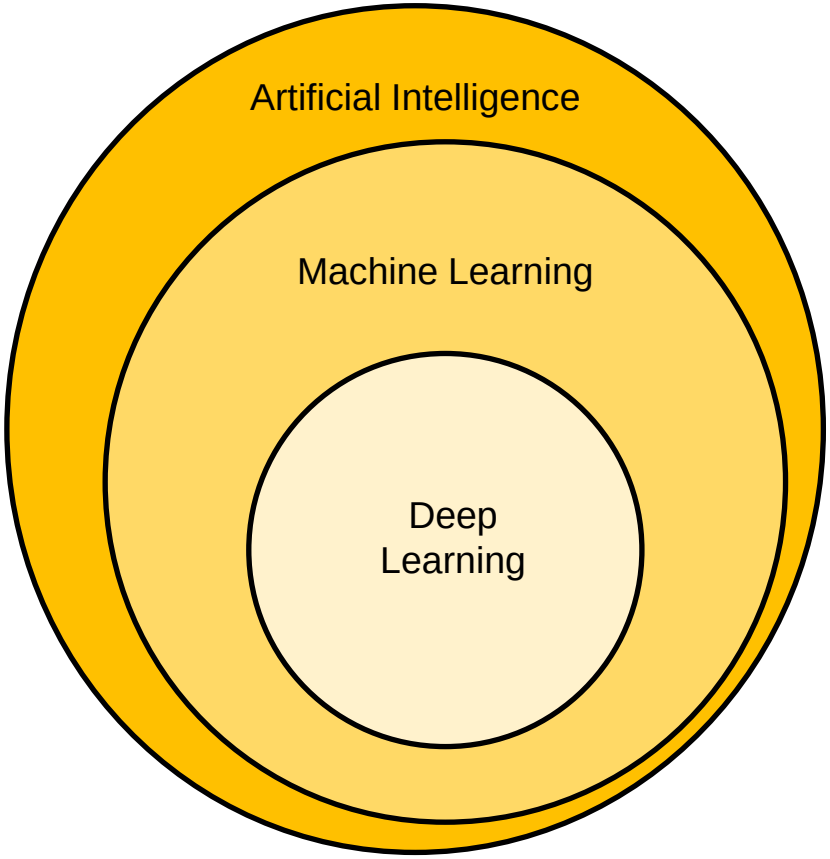
a) How many of you have cheated at least once during your studies?

Underfitting, Generalization, Overfitting

b) How many of you dislike having to learn a new programming language?

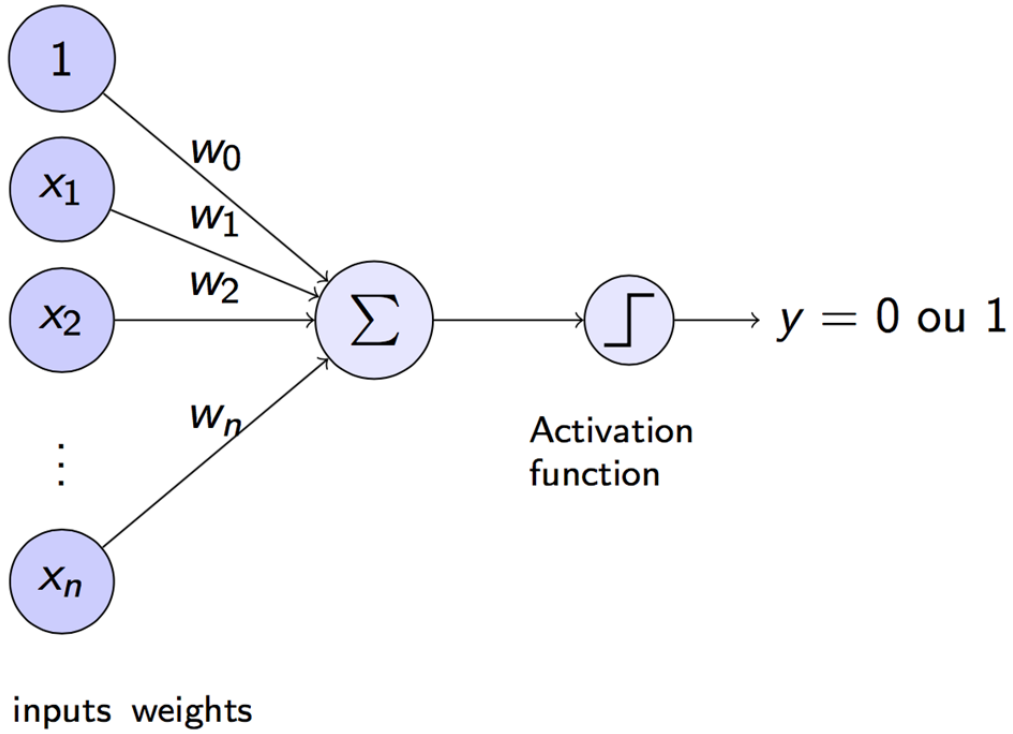
It’s normal.

From AI to Deep Learning



1950: Turing Learning Machine

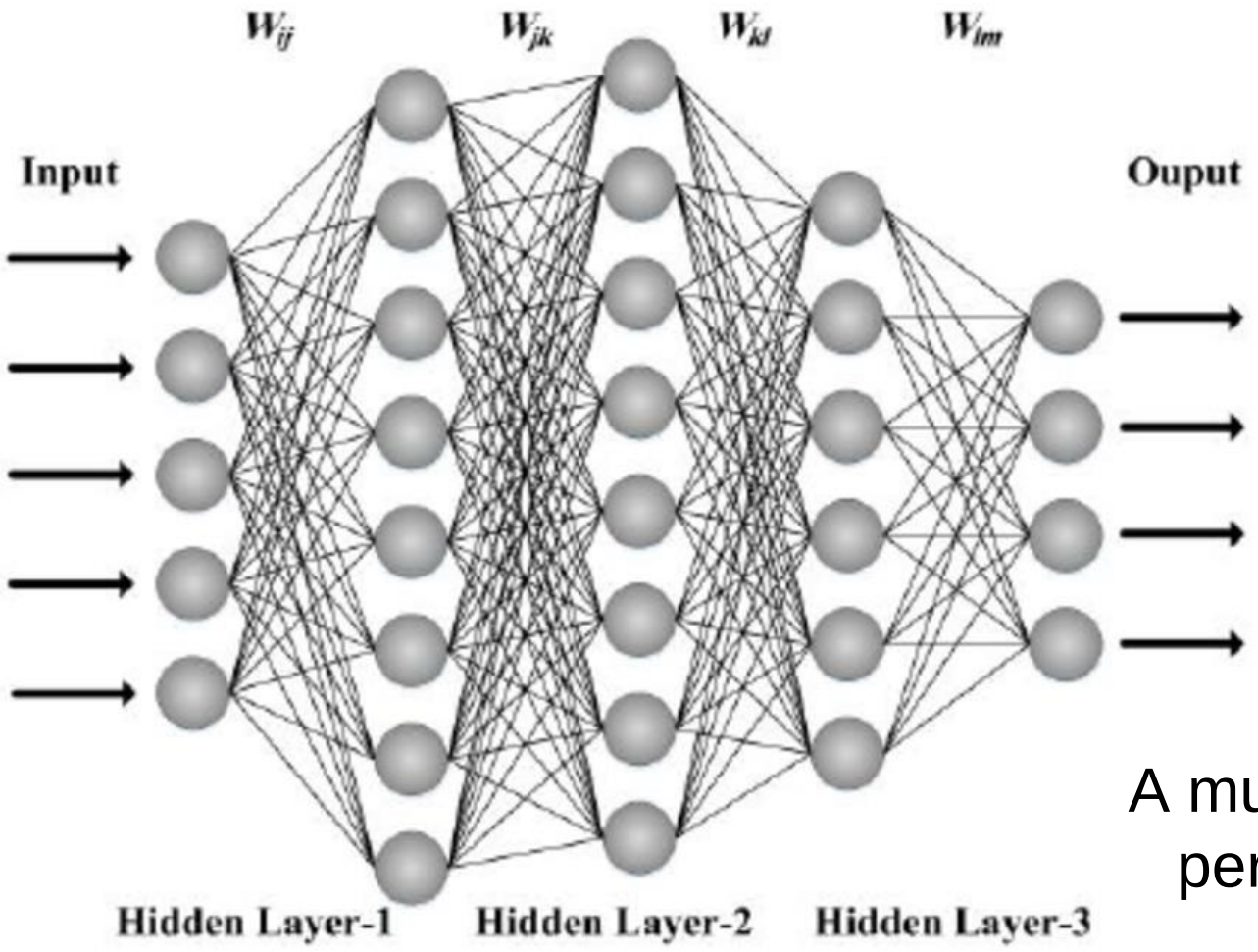
1957: Frank Rosenblatt invents the perceptron



If $w_0 + w_1 \times x_1 + w_2 \times x_2 + \cdots + w_n \times x_n < 0$, $\mathbf{y = 0}$
else $\mathbf{y = 1}$

Efficient heuristic to determine weights

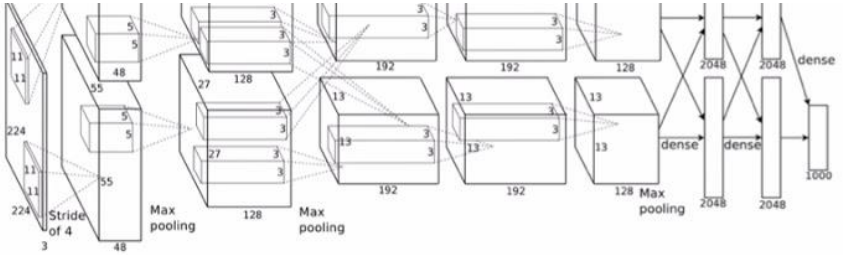
1986: The process of backpropagation is described by David Rumelhart, Geoff Hinton and Ronald J. Williams.

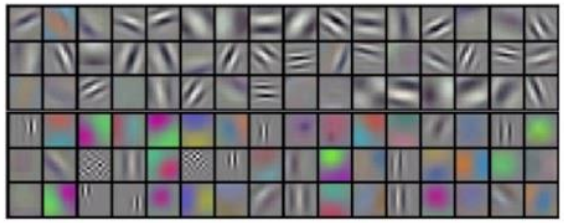


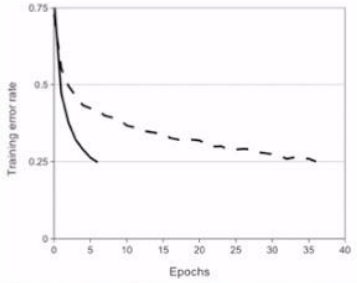
A multi-layer perceptron

1997: IBM Deep Blue Beats Kasparov

2012: AlexNet learns to recognise images on ImageNet







$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

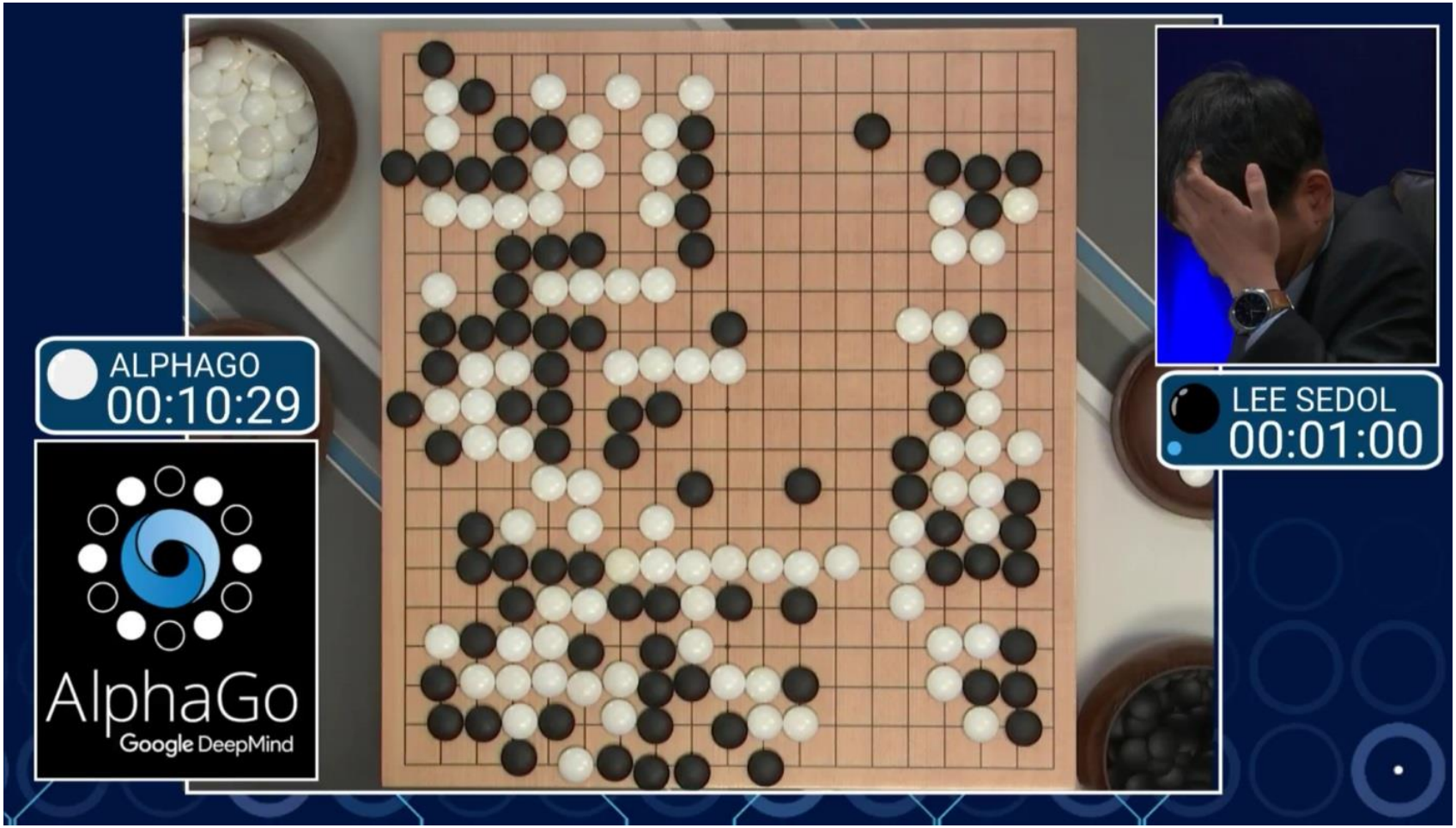
$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i}$$

$$w_{i+1} := w_i + v_{i+1}$$

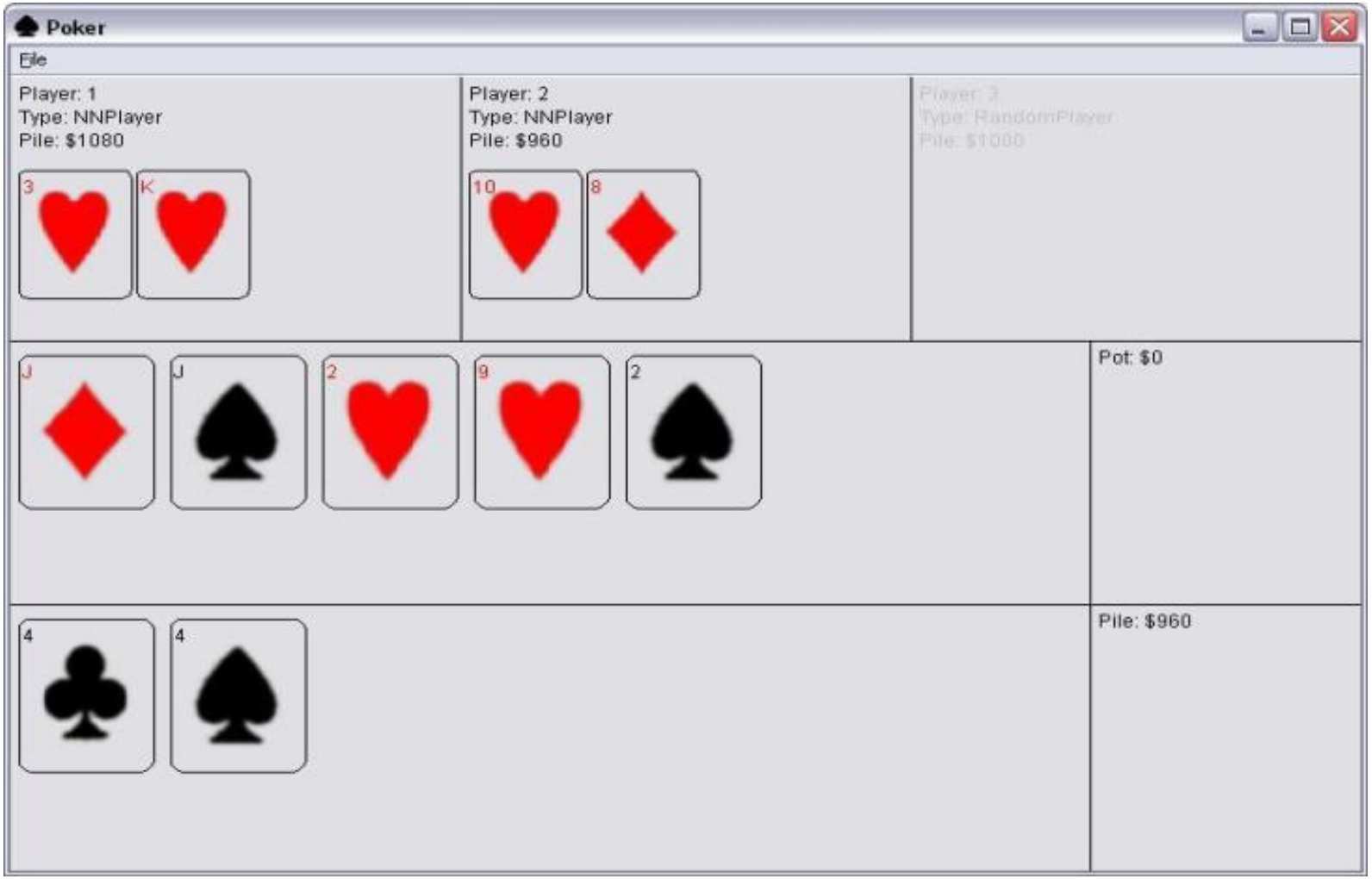
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

1. ILSVRC-2012, 1.2M img, 1000classes,top1(37.5%) top5(17%)
2. 2010ILSVRC: 1.2M train; 50K val; 150K test.
3. nw: 60M para, 650k neurons, 5conv+3fc +softmax
4. Relu+GPU+lrn+overlap pooling+dropout
5. 2013, MNIST ER<0.3%
6. imageNet=15M images+22K cate (still overfitting)
7. faster GPU, bigger dataset, better architecture…
8. Nonlinearity: Relu faster learning than tanh. (6times), no saturating
9. GPU, read and write each other without CPU.M, vis.2.streams
10. lrn: aid generalisation. lateral inhibition. certain layers after relu.
11. overlap pooling: s=2 z=3, difficult to overfit
12. conv1-relu1-lrn1-pool1-conv2-relu2-lrn2-pool2-conv3-relu3-conv4 -relu4-conv5-relu5-pool5-fc6-relu6-fc7-relu7-softmax-prob-obj-error
13. conv2,conv4,conv5 connected only to input from same GPU conv3,fc, connected to all previous neurons
14. reduce overfitting: label-preserved image augmentation
    1. image translation and flip, then crop 224x224
    2. RGB intensity modification by PCA
15. Dropout: prob=0.5, dont FF nor FB, sp@different arch, no specific dep. learn robust conn to subset neurons . @2fc layers
16. Momentum SGD learning
17. lr divide 10, if stop improving validation error,90 cycle 1.2M images
18. results: ensemble, pretrain imageNet best
19. GPU1 orientation, GPU2 color-specific. every run
20. detect offcenter objects, top5 reasonable,pic ambiguity
21. rm single conv, degrade performance.
22. infers-temporal pathway
23. large+deep nw on video

2015: AlphaGo beat a human professional Go player



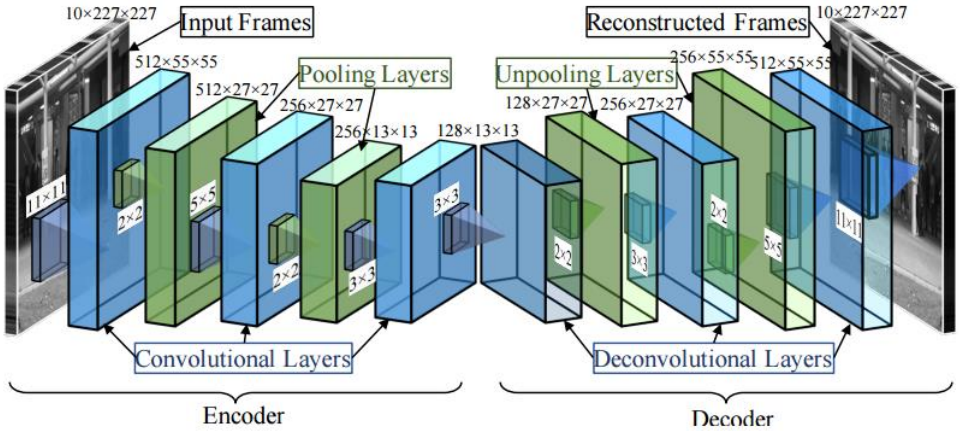
2017: DeepStack wins professional poker tournament

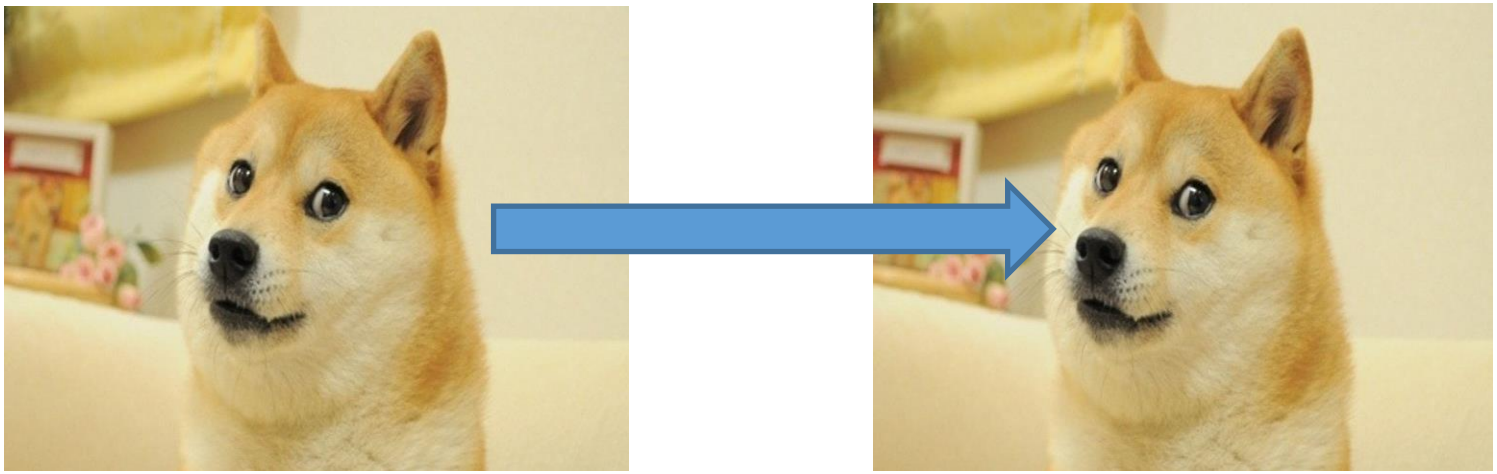


Deep Learning

Google Dream

Convolutional Auto-encoder

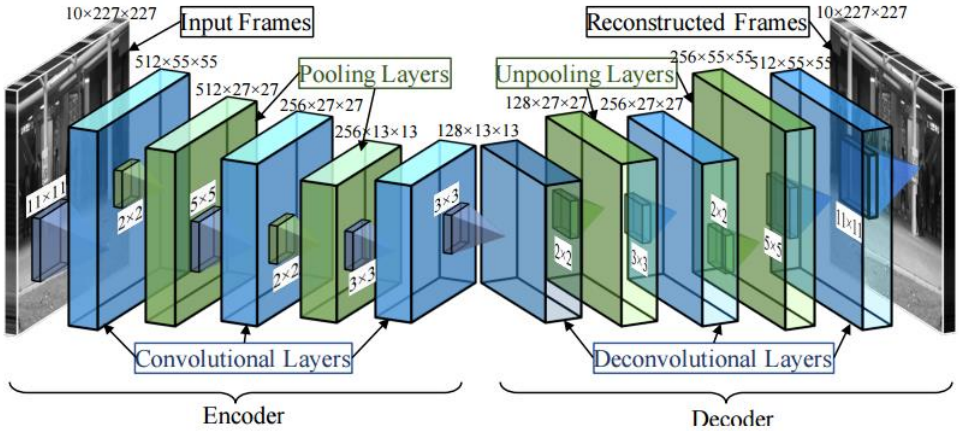




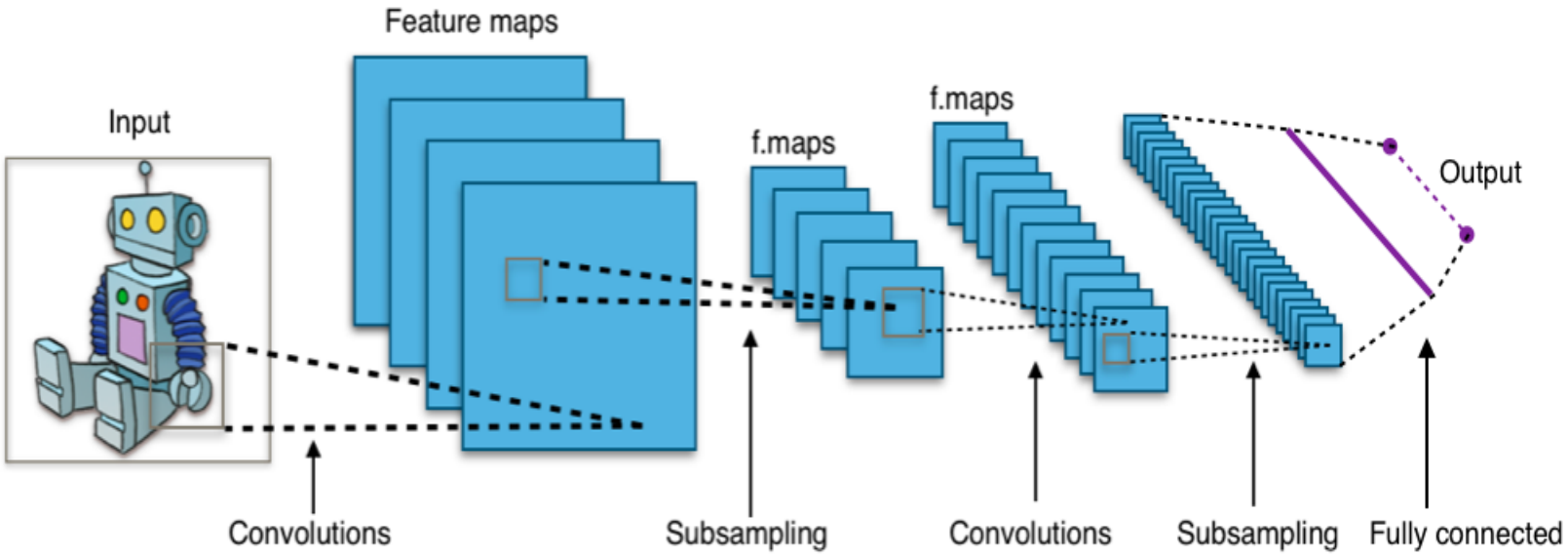
Deep Learning

Google Dream

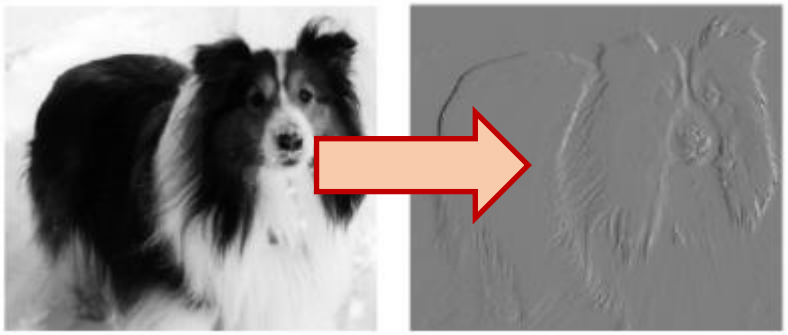
Convolutional Auto-encoder



Convolutional Neural Networks

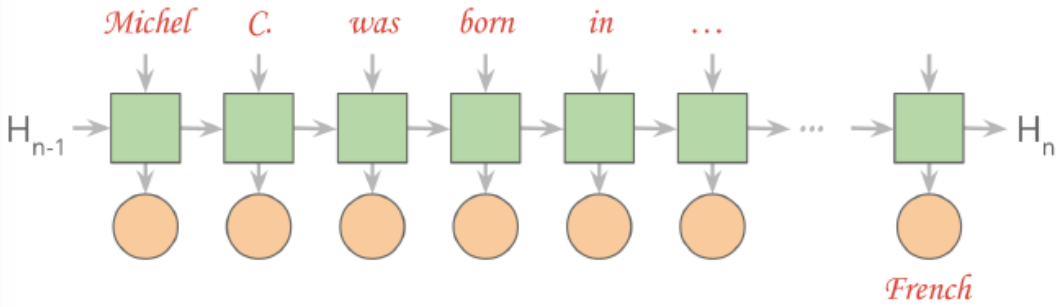


Learns a set of filters to apply on images



Recurent Neural Networks

Michel C. was born in Paris, France. He is married and has three children. He received a M.S. in neurosciences from the University Pierre & Marie Curie and the Ecole Normale Supérieure in 1987, and and then spent most of his career in Switzerland, at the Ecole Polytechnique de Lausanne. He specialized in child and adolescent psychiatry and his first field of research was severe mood disorders in adolescent, topic of his PhD in neurosciences (2002). **His mother tongue is** ?????

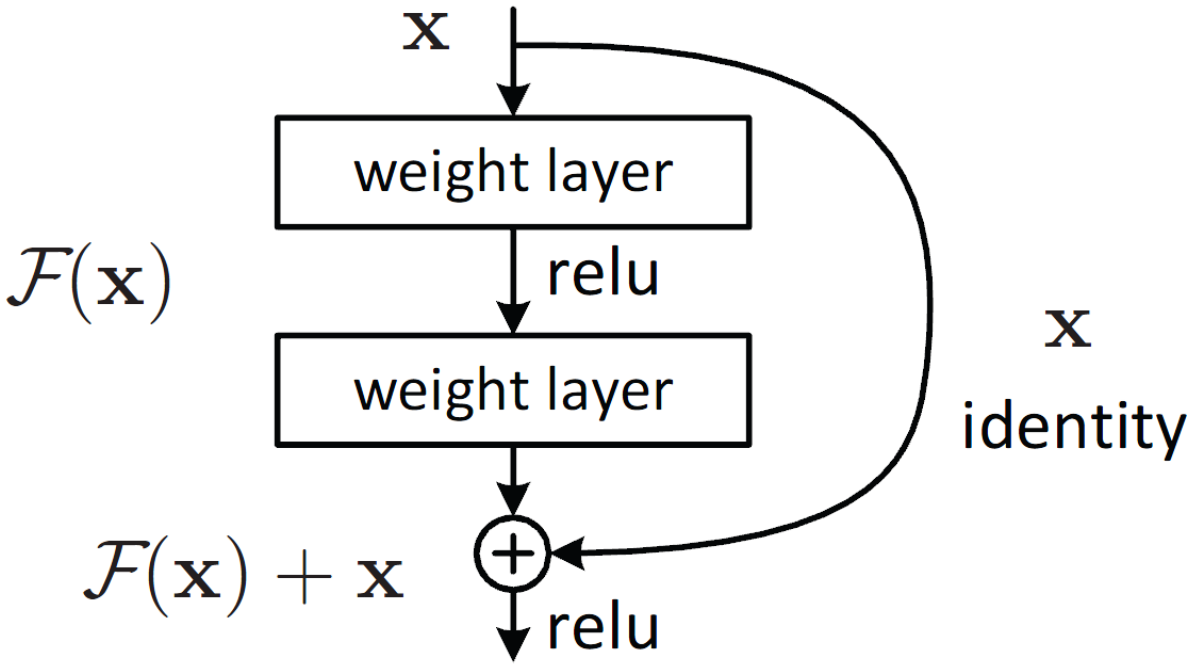


Short context → English, German, Russian, French ...

Long context → Problems...

Residual Neural Networks

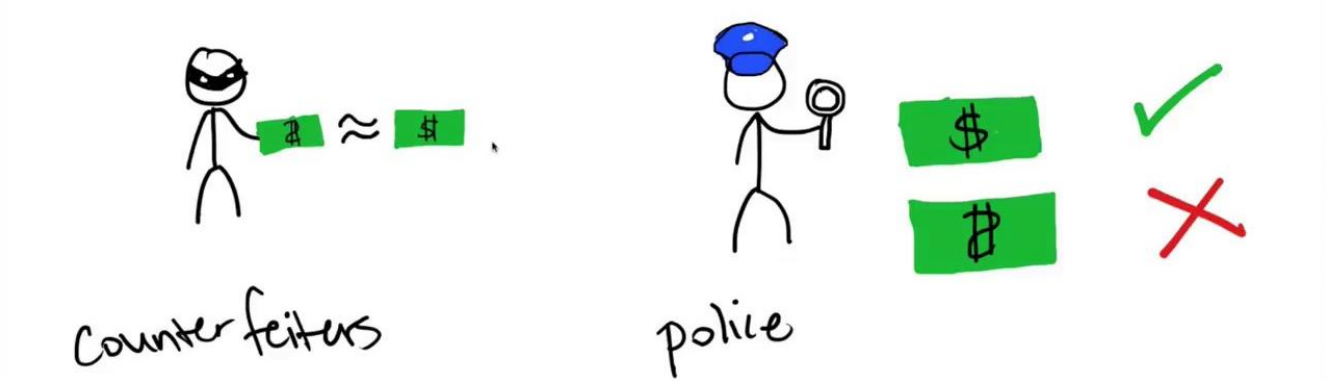
Empyrical learning of differential equations



https://neurohive.io/en/popular-networks/resnet/

Adverserial Neural Networks

Inspired by game theory:



https://academy.zenva.com/product/deep-learning-mini-degree/?zva_src=youtube-deeplearning-MD

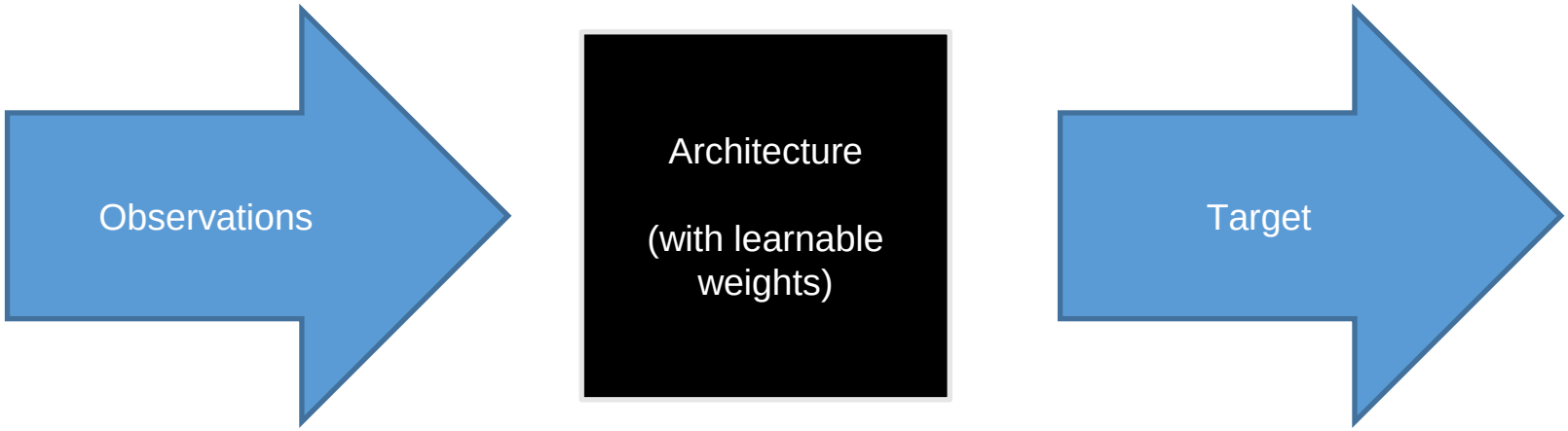
« Transfer Learning »

Problem:
training a network is expensive (time, computations, data, expertise…)

Solution:
Use another already trained network and retool it to the new data / problem

model		Alex	VGG-16	GoogLeNet	NIN
conv	layer	5	13	21	12
	weights	3.8M	15M	5.8M	7.6M
	comp.	1.1B	15.3B	1.5B	1.1B
FC	layer	3	3	1	0
	weights	59M	124M	1M	0
	comp.	59M	124M	1M	0
TOTAL	weights	62M	138M	6.8M	7.6M
	comp.	1.1B	15.5B	1.5B	1.1B
ImageNet	top-5 err.	17.0%	7.3%	7.9%	10.9%

Machine Learning



« Targets »:

$$\begin{cases} Data\ inaccessible\ (either\ costly\ to\ get, or\ exists\ in\ the future\ or\ otherwise\ unmeasurable) \\ \\ Clusters\ ou\ other\ representations\ of\ the\ data \end{cases}$$

Split your dataset in:

Training:	to learn the architecture's weights (estimation through an iterative process)
Test:	to compare different architectures
Validation:	to ensure we are not overfitting

Avantages/ Inconveniants

Massively parallelisable
(Post-Dénard Era)

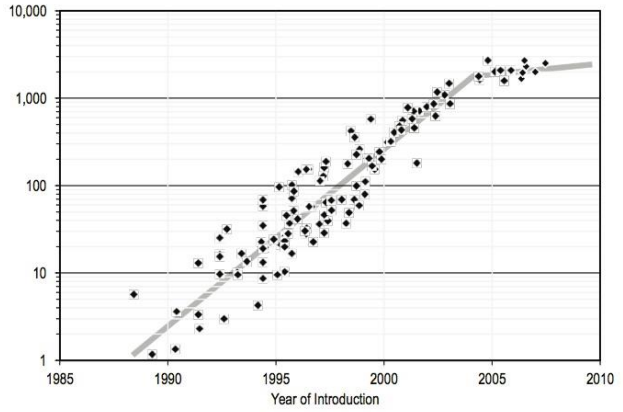


FIGURE A.2 Floating-point application performance (SPECfp2000) over time (1985-2010).

Great initial application cost but inexpensive application

Sensibility to missing data and definition of performance

Inability to predict situations/classes not present in training

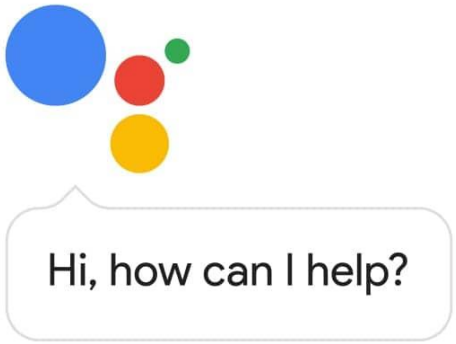
Applications

Automatic driving (car / drones)

- Object detection and tracking,
- Trajectory predicition,
- Moral Choices…

Applications

Personal Assistants



Vocal → Texte$ → Semantic

Applications

Personalised creation

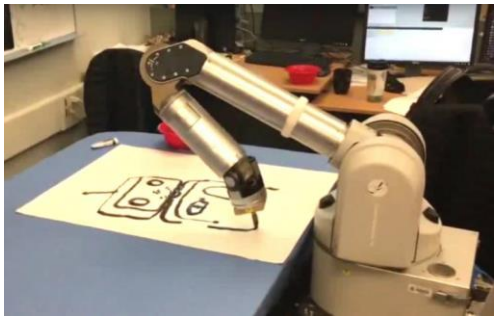
Catégorisation of tastes
High level Semantic Generators (scenarios), then cascade applications to lower levels (dialogues etc)

↓

A tailor made story!

Dangers

Obsolescence of human labor →

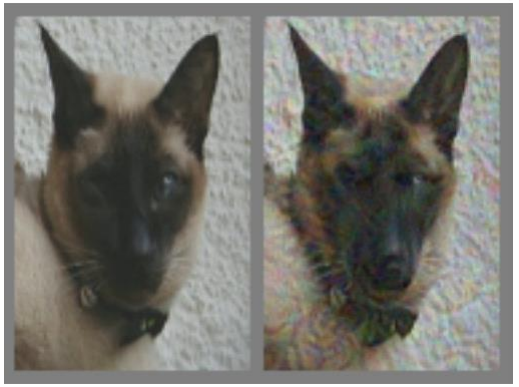


A robot attempts a self-portrait, but lacks a mirror or self-awareness. ROBOTART

Desanonymification of data

Création of algorithms that aim at « crashing » the economy

https://www.stockinvestor.com/39050/laughing-stock-market-crash/



Or to cheat other algorithms…

Dangers

Input

Output

Deep Fake

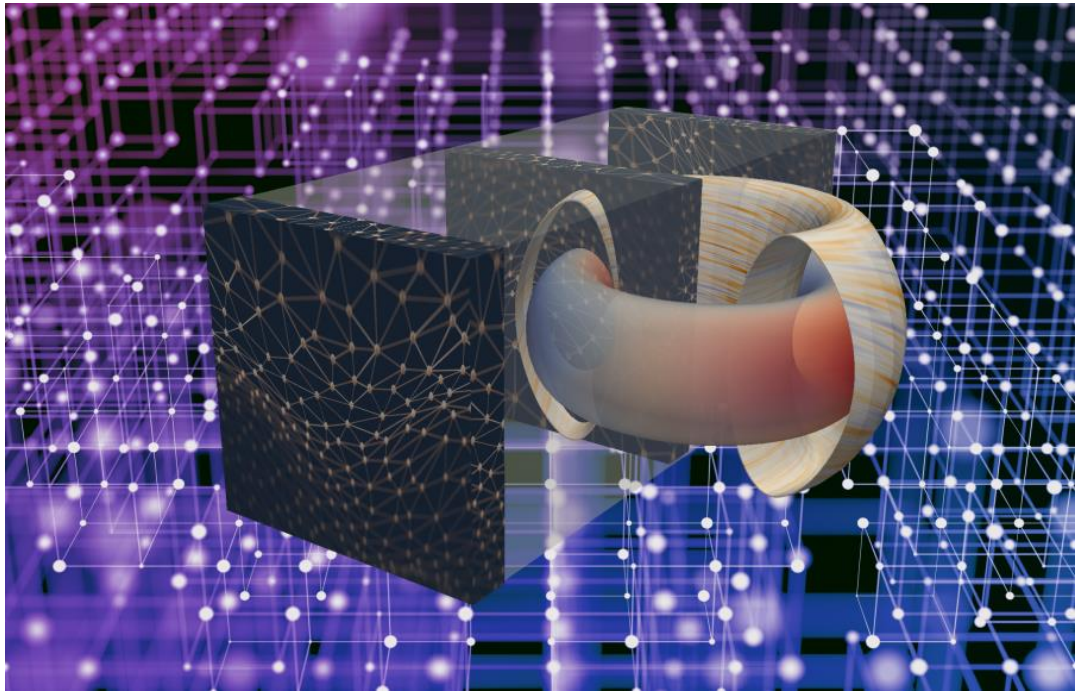
H. Kim et al., 2018

Source Sequence

Our Reenactment (Full Head)

Averbuch-Elor et al. 2017

All is not catastrophic!



Deep Fusion

Machine Learning Basics

- TASK

  Classification, Classification with missing inputs, Regression, Transcription, Machine translation, Structured output, Anomaly detection, Synthesis and sampling, Imputation of missing values, Denoising, Density estimation

- PERFORMANCE

- EXPERIENCE

Machine Learning Basics

- TASK
- PERFORMANCE
  Numerical estimation of the accuracy, or equivalently the error rate of the algorithm
  Requires a separate data set, referred to as the Test set.
- EXPERIENCE

Machine Learning Basics

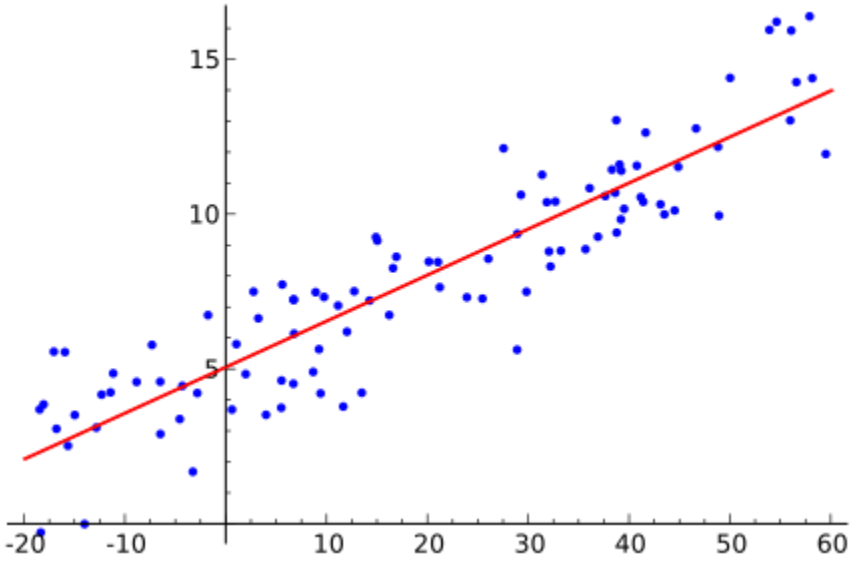
- TASK
- PERFORMANCE
- EXPERIENCE

  Allowing the alogrithm to « experience » a dataset, progressively adapting its parameters to improve its performance. This experience can be supervised or unsupervised, providing it with extra external information throughout its learning phase.

  This phase of the algorithm learning and updating cycles:

  It calculates the output of the algorithm over a dataset, using the current parameters of the model.
  It evaluates the output given the performance metric.
  It updates the parameter values of the model.
  If a condition is met, it stops.

BASIC EXAMPLE: LINEAR REGRESSION



Dataset:

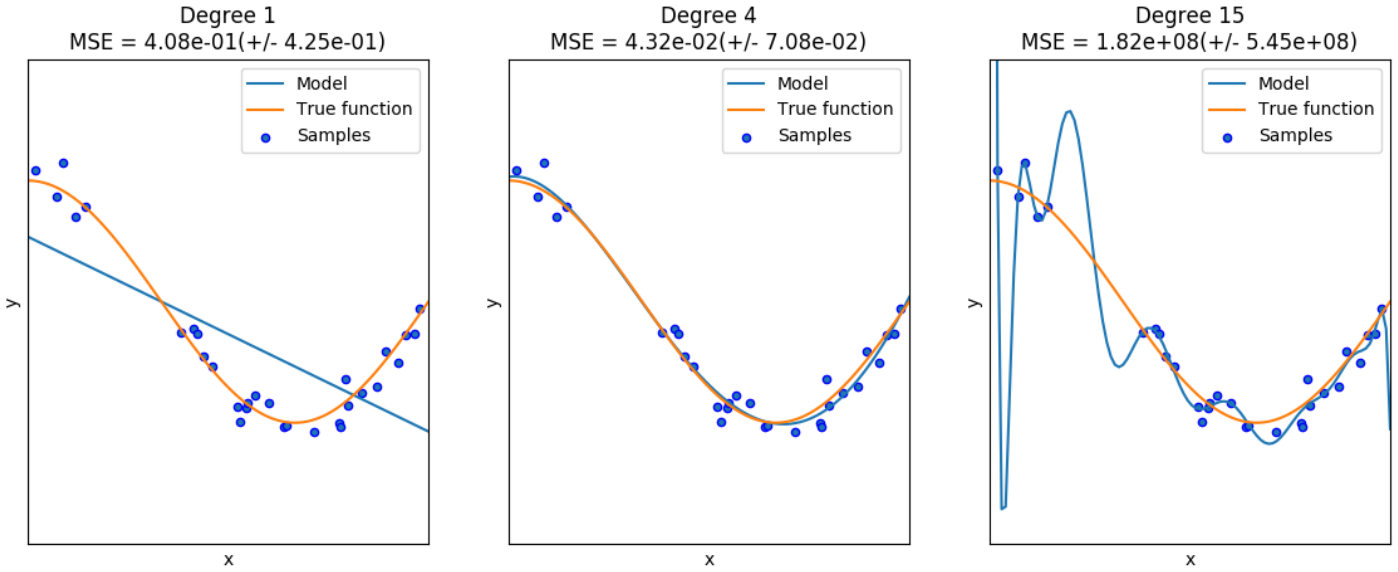
X, the explanatory variables
y, the target values

Task

Performance

Experience

Fundamental Concepts: Hyperparameters &Fitting

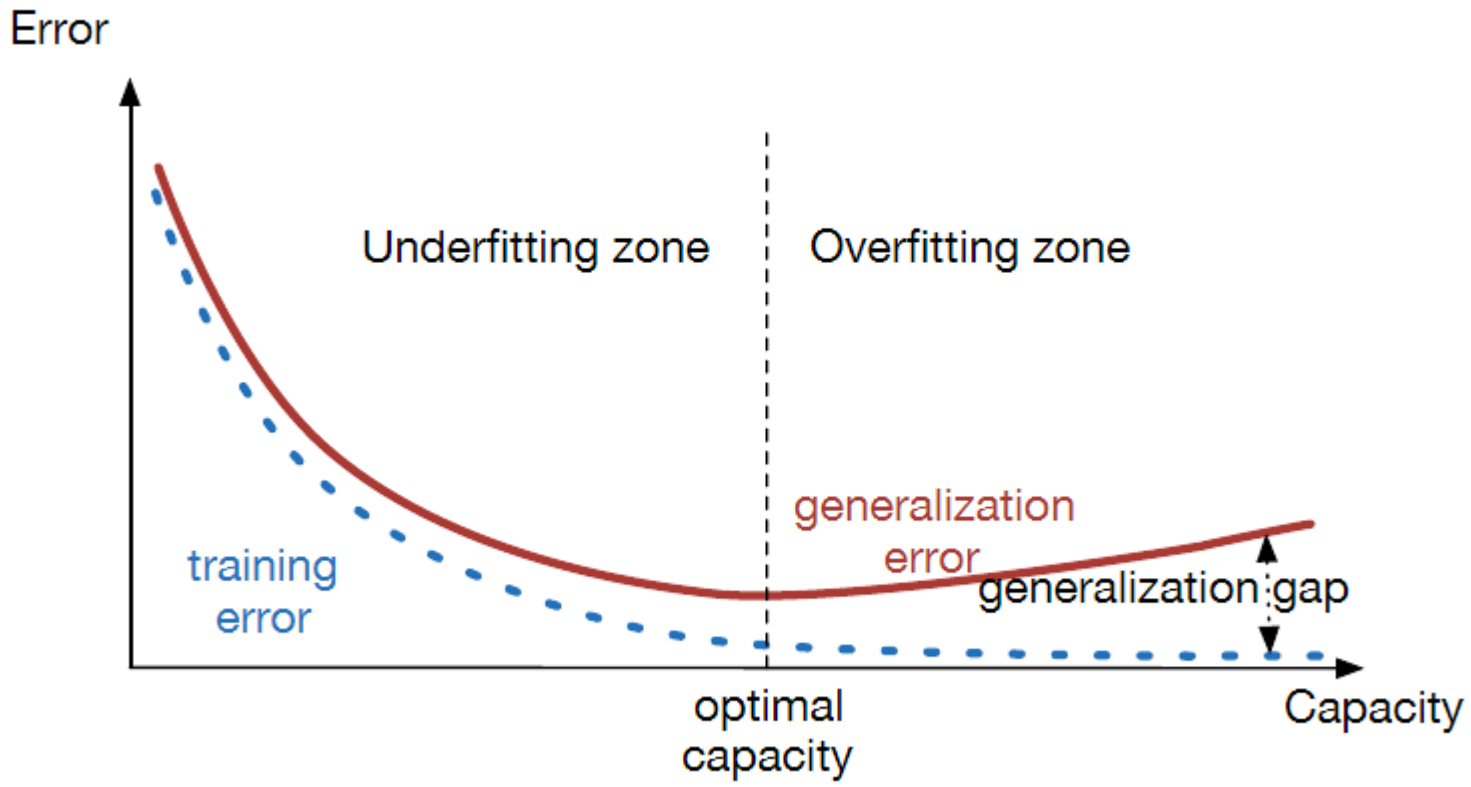


Underfitting

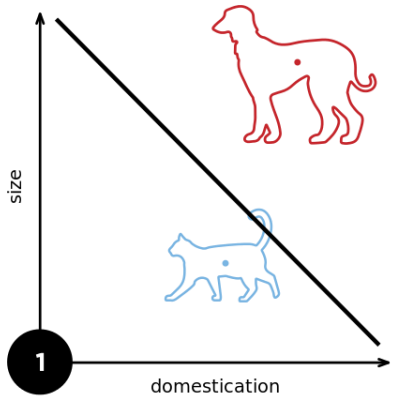
Fit

Overfitting

Fundamental Concepts: Hyperparameters &Fitting



BASIC EXAMPLE: PERCEPTRON



Dataset:

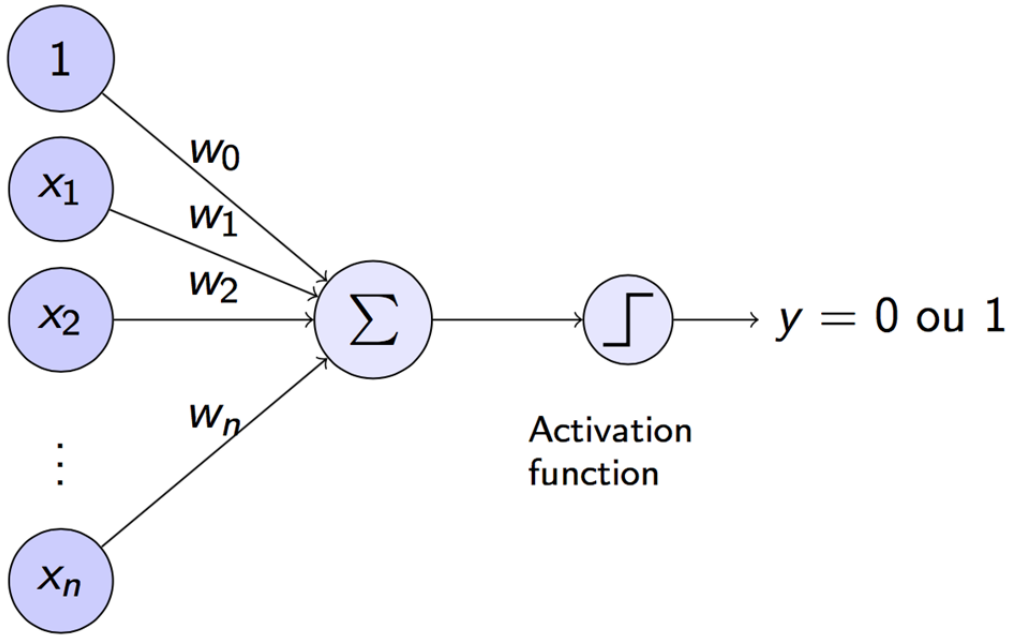
X, the explanatory variables
y, the target values

Task

Performance

Experience

BASIC EXAMPLE: PERCEPTRON



Dataset:

X, the explanatory variables
y, the target values

Task

Performance

Experience

If $w_0 + w_1 \times x_1 + w_2 \times x_2 + \cdots + w_n \times x_n < 0$, $\mathbf{y = 0}$
else $\mathbf{y = 1}$

BASIC EXAMPLE: PERCEPTRON

Definitions:

y=f(z) is the *output* from the perceptron for an input vector z

D_n is the training data-set consisting of *n* number pairs:
$\{(X_1 , trg_1) \ldots (X_i , trg_i) \ldots (X_n , trg_n)\}$

Where X_i is the m-dimensional input vector
$X_{i,j}$ is the j-th element of the vector
$X_{i,0}$ is considered to be 1
And trg_i is the target value (0 or 1) for that input

W_j is the weight of the linear regression over the j-th element

Since it is an iterative algorithm, W_j (t) symbolizes the value of the weights at iteration t. Initialise *w* to some values

Finally, η is the *learning rate*, be a small positive number (small steps lessen the possibility of destroying correct classifications)

BASIC EXAMPLE: PERCEPTRON

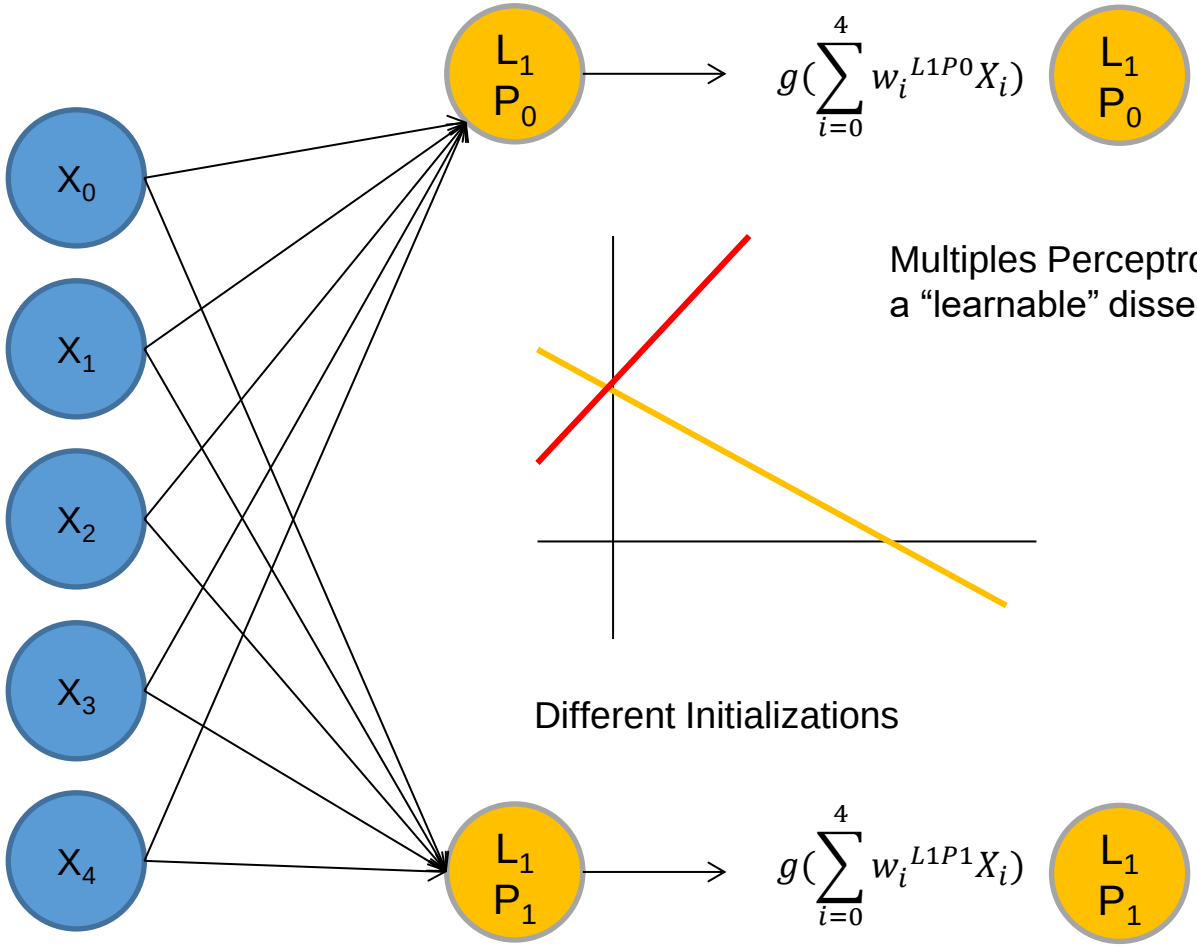
1. Select random sample from training set as input
2. Calculate the output: $y_i(t) = f(W(t) * X_i)$
3. If classification is incorrect, modify the weight vector *w* using:

$$W_j(t+1) = W_j(t) - \eta * (trg_i - y_i(t)) * X_{i,j}$$

The perceptron is a *linear classifier*, therefore it will never get to the state with all the input vectors classified correctly if the training set D is not linearly separable, i.e. if the positive examples can not be separated from the negative examples by a hyperplane.

In this case, no "approximate" solution will be gradually approached under the standard learning algorithm, but instead learning will fail completely.

BASIC EXAMPLE: MULTIPLE PERCEPTRON

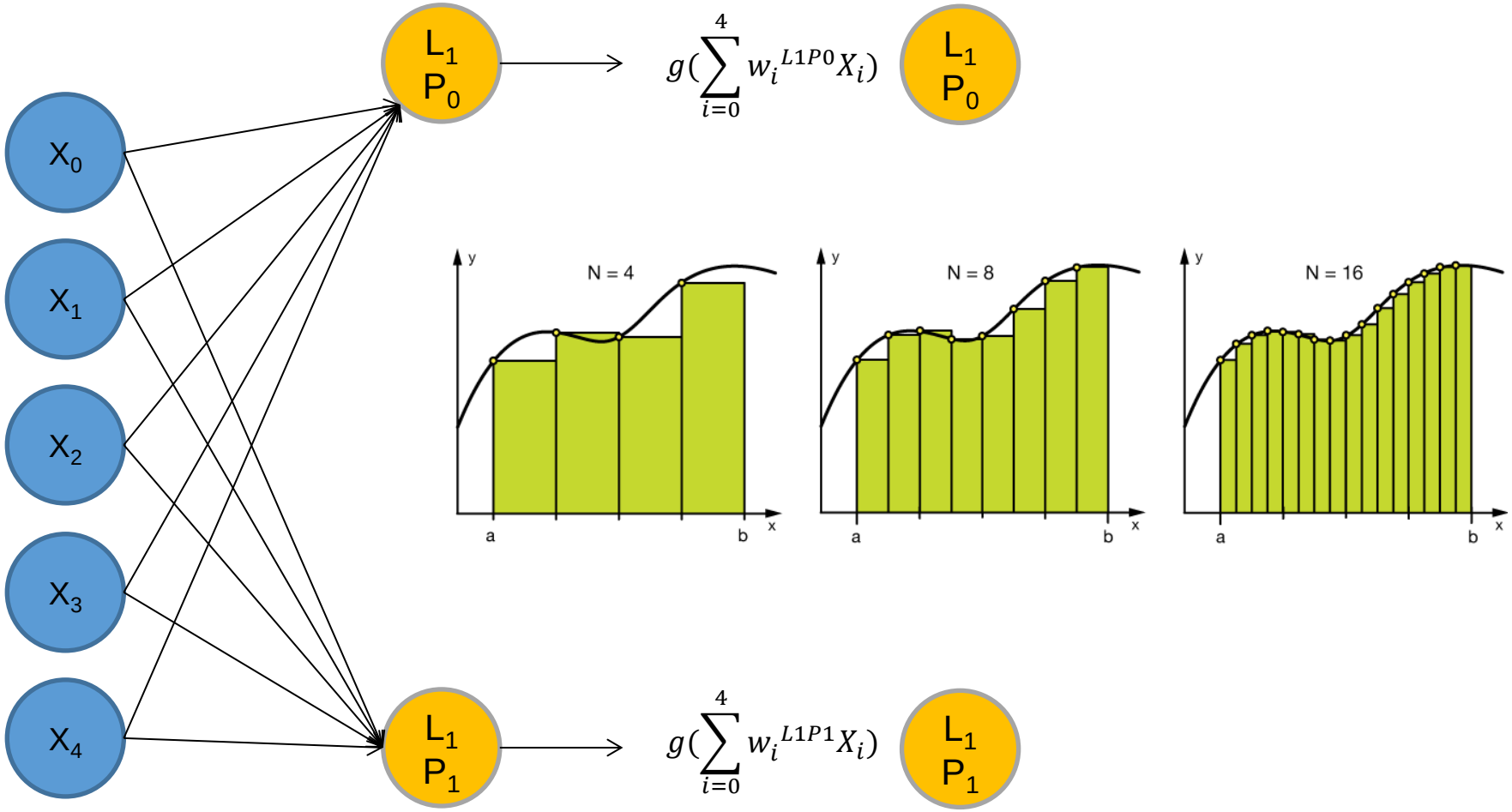


Different Initializations

Multiples Perceptrons with same inputs, allow for a “learnable” dissection of the input dimension.

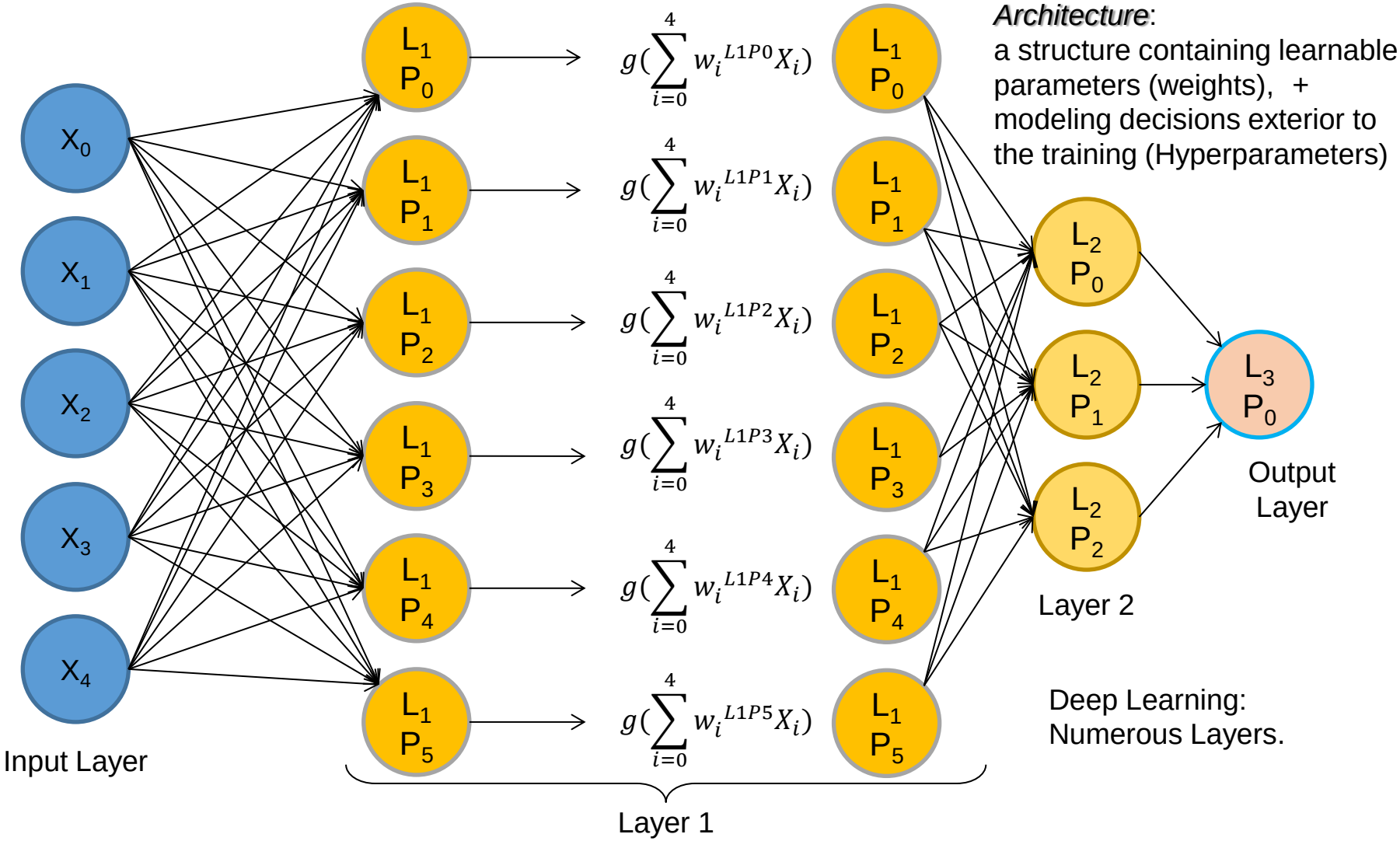
Combining the outputs of these perceptrons with other perceptrons allows us to medialize non-linear processes by « linearizing» them in a higher dimension.

BASIC EXAMPLE: MULTIPLE PERCEPTRON



Input Layer

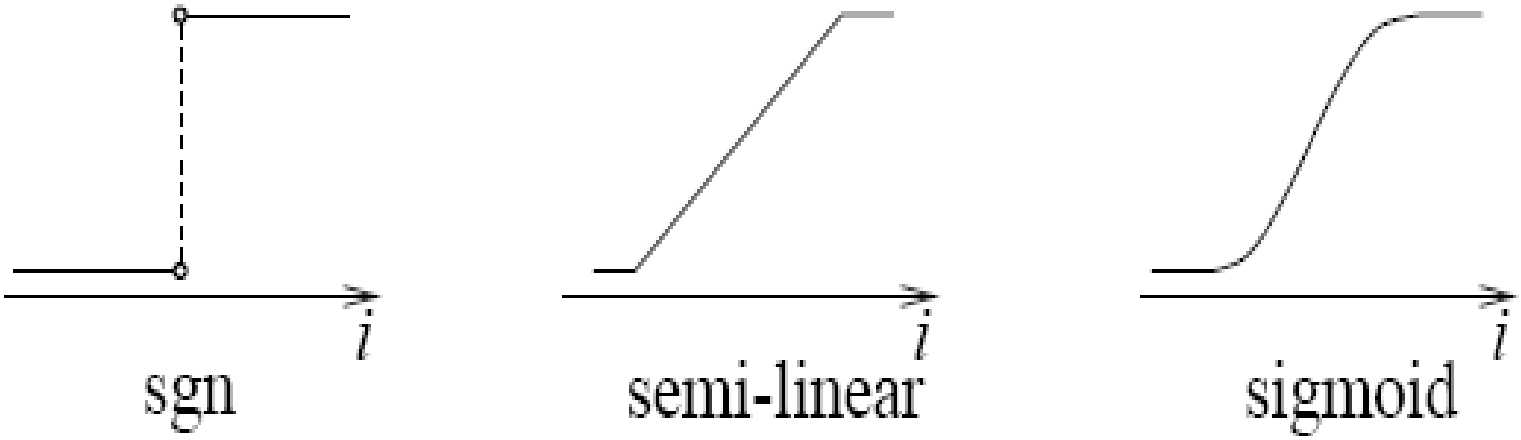
BASIC EXAMPLE: MULTIPLE PERCEPTRON



BASIC EXAMPLE: PERCEPTRON

Activation Function:

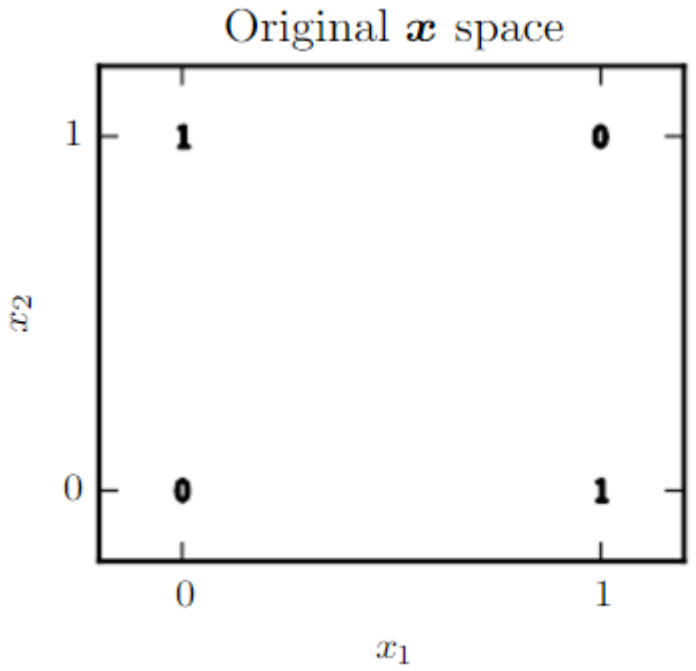
takes the total input and produces an output for the node given some threshold.



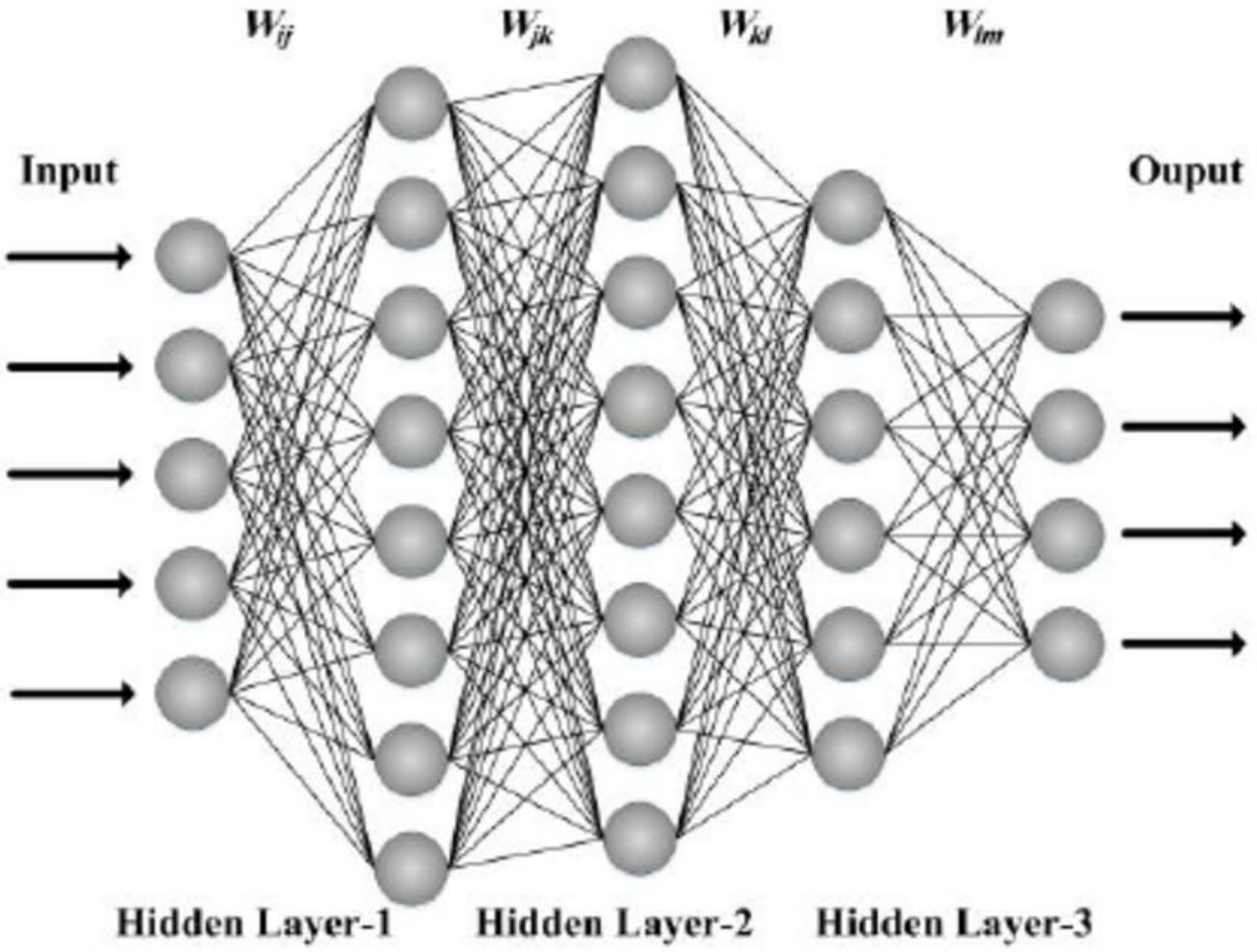
Others! (Logistic)

BASIC EXAMPLE: PERCEPTRON

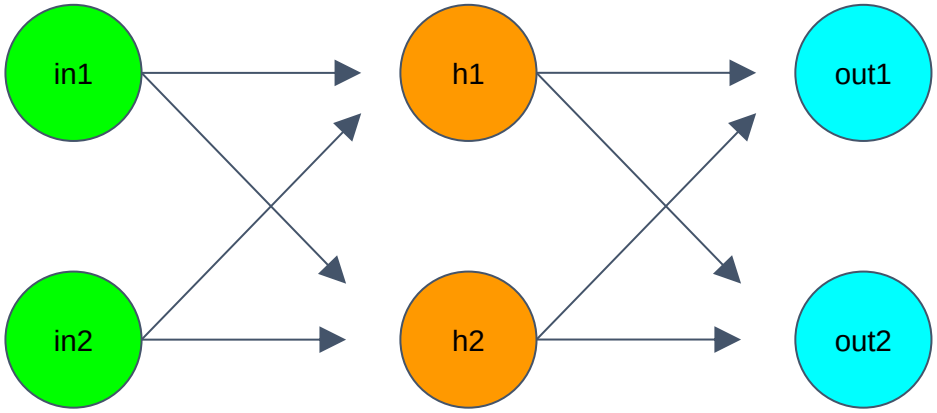
XOR: Can it be solved?



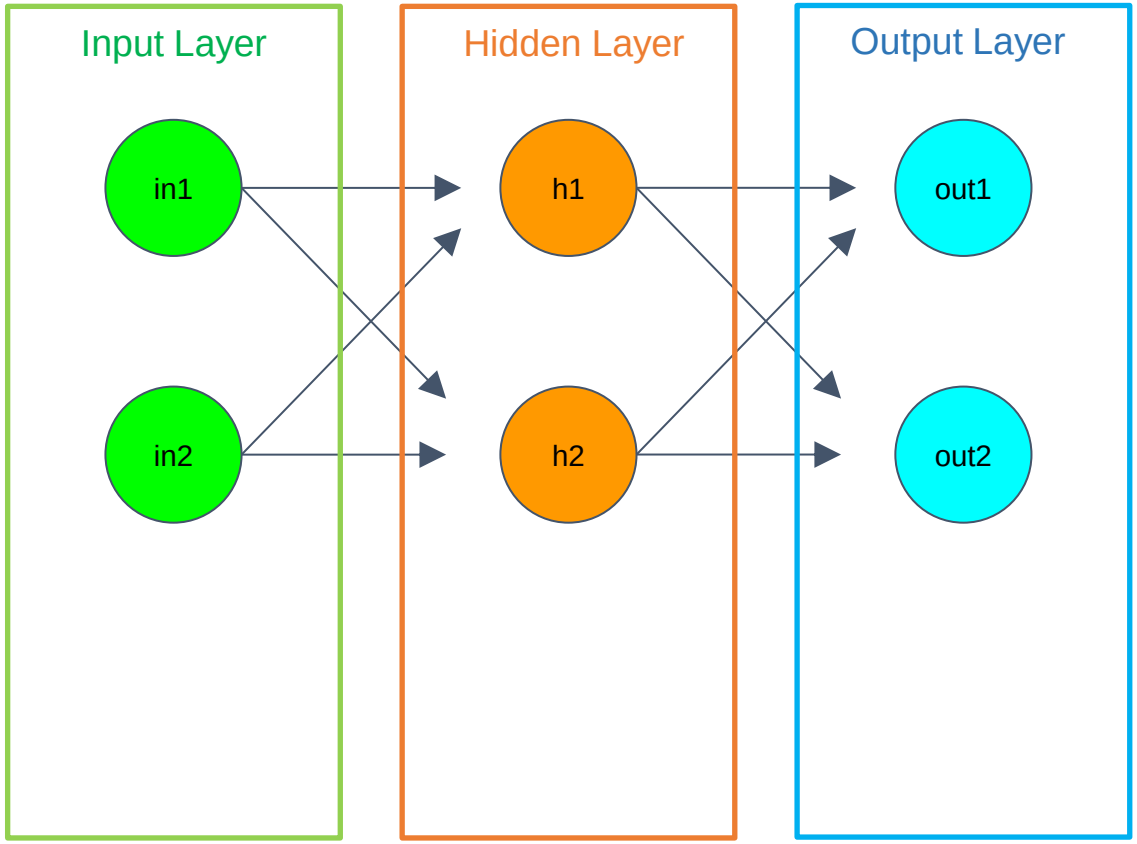
DEEP LEARNING BASICS: MLP



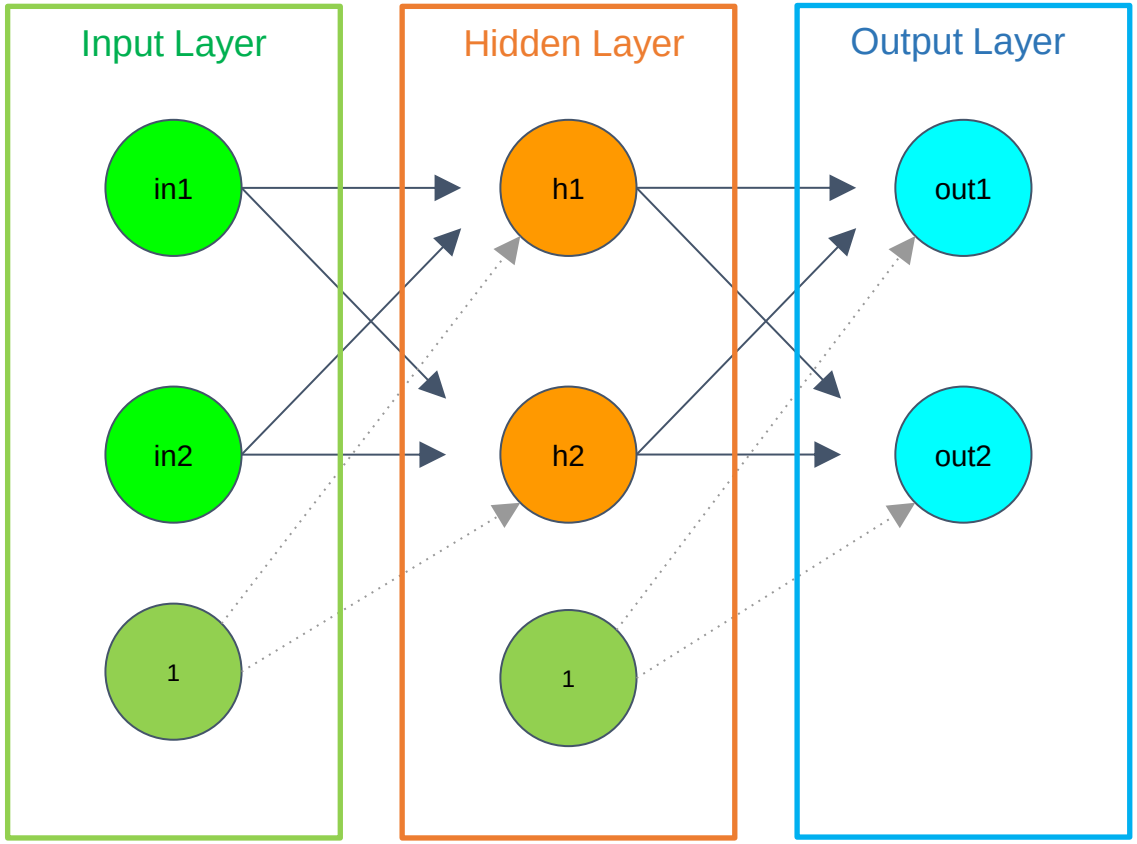
DEEP LEARNING BASICS: MLP

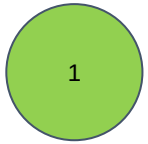


DEEP LEARNING BASICS: MLP



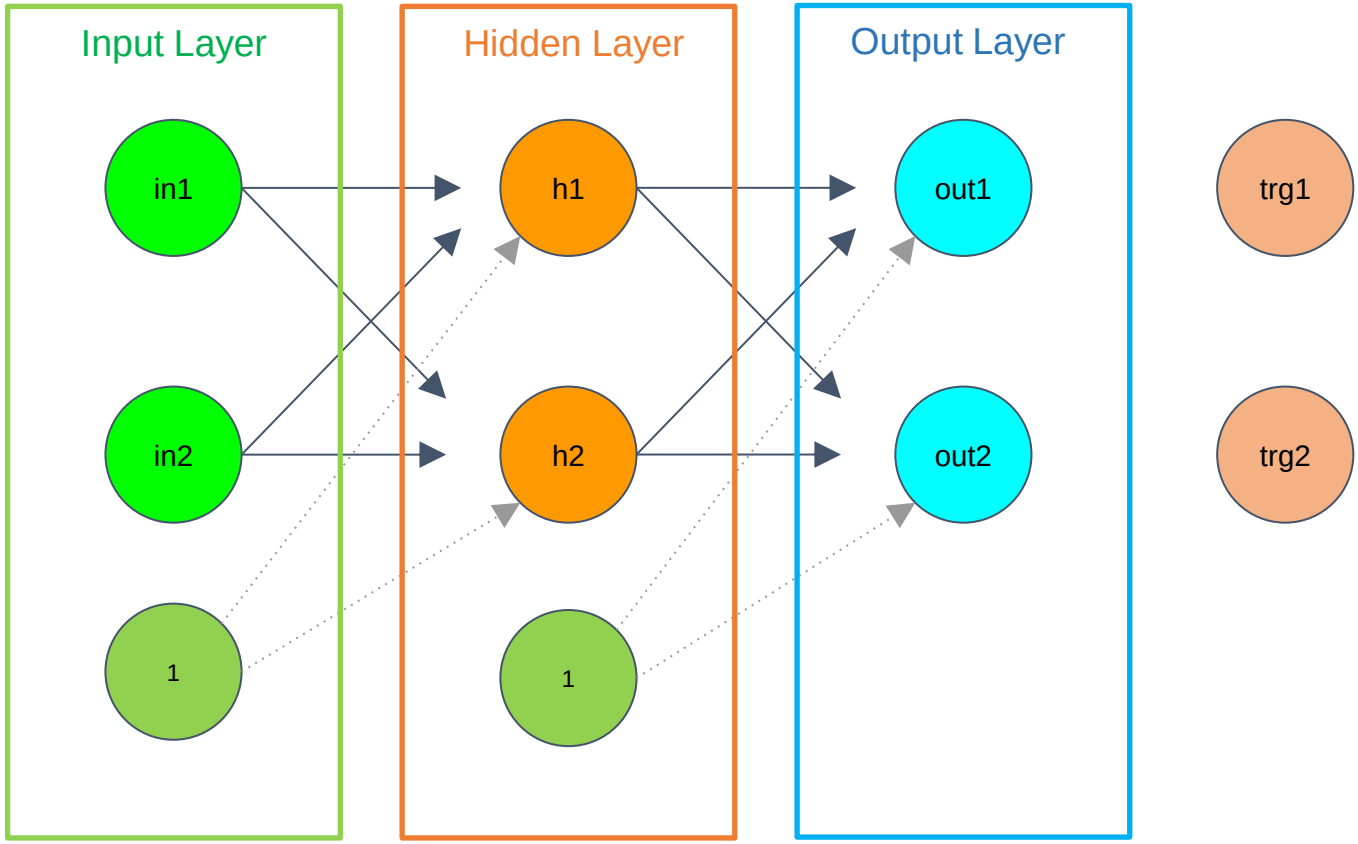
DEEP LEARNING BASICS: MLP



Why 

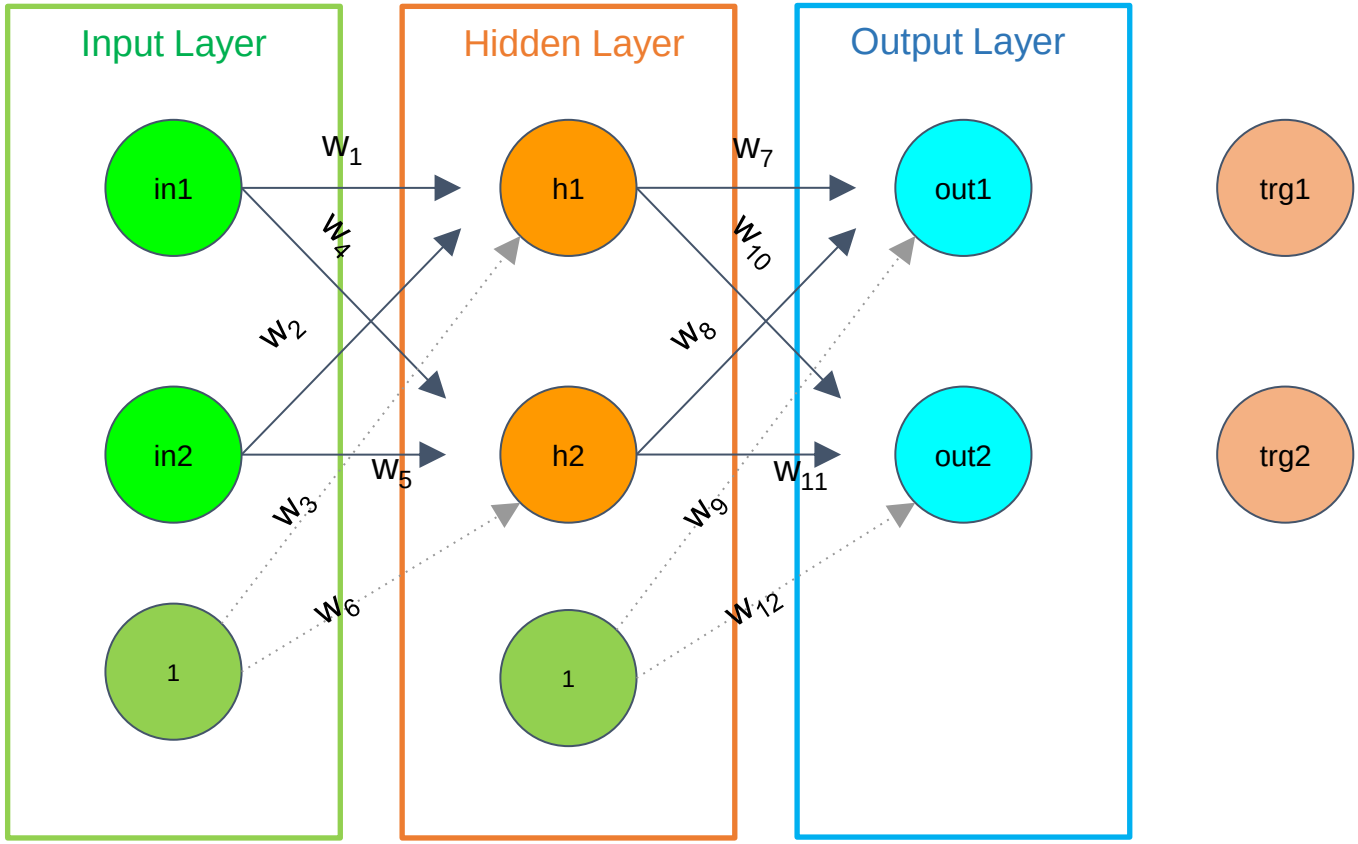
(hint: linear regression)

DEEP LEARNING BASICS: MLP



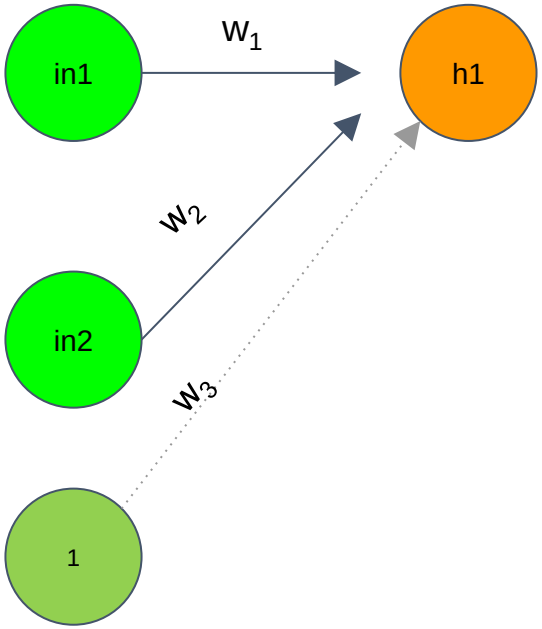
Example: $in_1 = 1;\ in_2 = 2; trg_1 = 0.1;\ trg_2 = 0.7$

DEEP LEARNING BASICS: MLP



Example: $in_1 = 1;\ in_2 = 2;\ trg_1 = 0.1;\ trg_2 = 0.7$

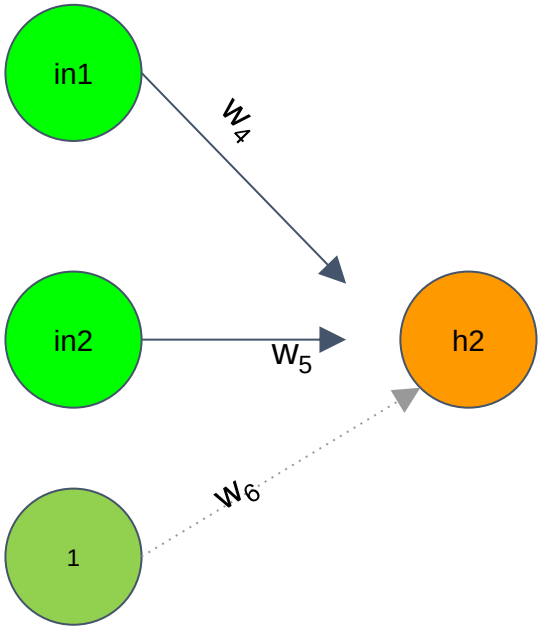
DEEP LEARNING BASICS: MLP



$$net_{h_1} = in_1 * w_1 + in_2 * w_2 + w_3$$

$$h_1 = af\left(net_{h_1}\right) =$$
$$= af\left(in_1 * w_1 + in_2 * w_2 + w_3\right)$$

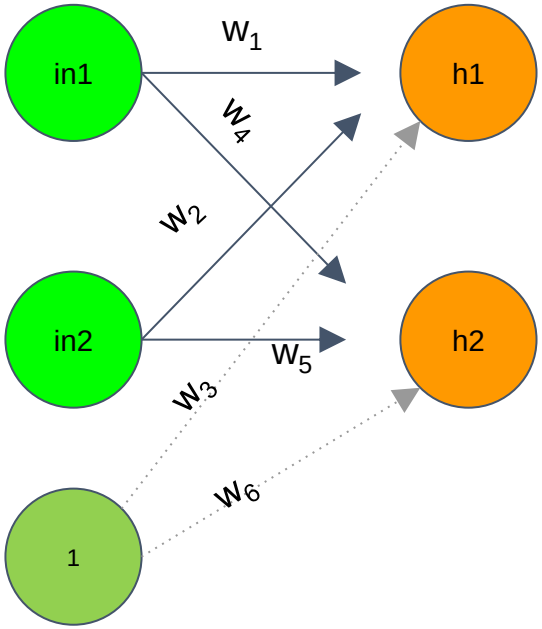
DEEP LEARNING BASICS: MLP



$$net_{h_2} = in_1 * w_4 + in_2 * w_5 + w_6$$

$$h_2 = af\left(net_{h_2}\right) =$$
$$= af\left(in_1 * w_4 + in_2 * w_5 + w_6\right)$$

DEEP LEARNING BASICS: MLP



$$in_1 = 1;\ in_2 = 2;$$
$$w_1 = -1;\ w_2 = -0.5; w_3 = 2.1;$$
$$w_4 = 1;\ w_5 = 2; w_6 = -4;$$

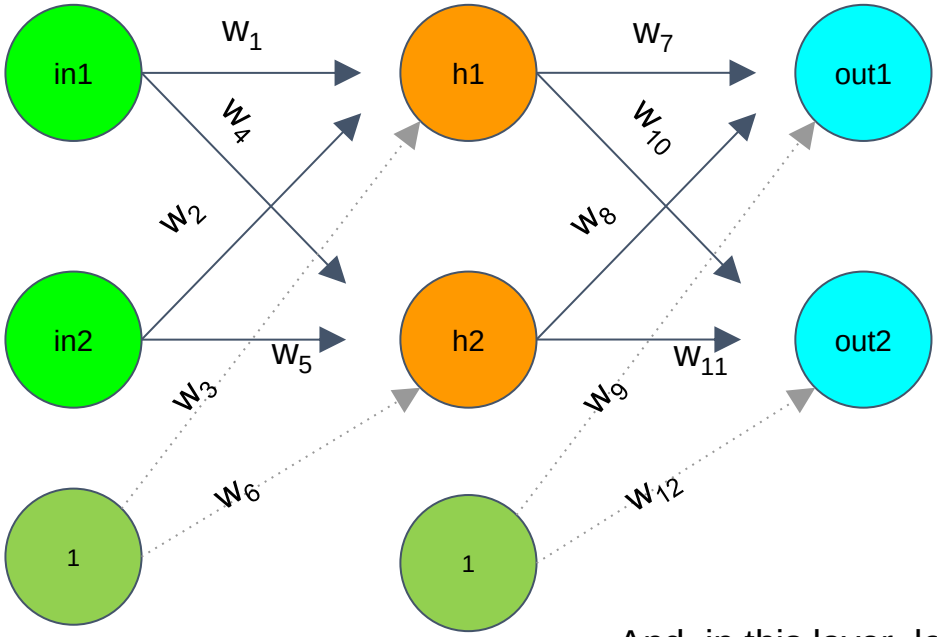
$$h_1 = af\left(net_{h_1}\right) =$$
$$= af(in_1 * w_1 + in_2 * w_2 + w_3)$$

$$h_2 = af\left(net_{h_2}\right) =$$
$$= af(in_1 * w_4 + in_2 * w_5 + w_6)$$

And, in this layer, let $af(x) = ReLu(x) = \begin{cases} x, & if\ x > 0; \\ 0, elsewhere. \end{cases}$

Calculate h_1 , h_2

DEEP LEARNING BASICS: MLP



$$h_1 = 0.1;\ h_2 = 1;$$
$$w_7 = 2;\ w_8 = -0.4; w_9 = -0.3;$$
$$w_{10} = -1;\ w_{11} = 1.5; w_{12} = 0.6;$$

$$out_1 = af\left(net_{out_1}\right) =$$
$$= af(h_1 * w_7 + h_2 * w_8 + w_9)$$

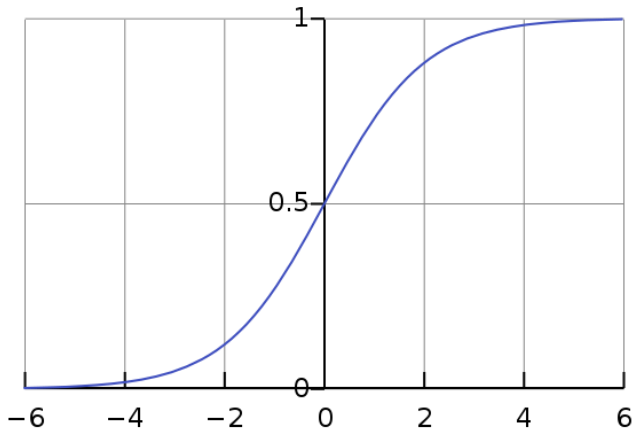
$$out_2 = af\left(net_{out_2}\right) =$$
$$= af(h_1 * w_{10} + h_2 * w_{11} + w_{12})$$

And, in this layer, let $af(x) = \frac{1}{1+e^{-x}}$, the logistic function

Calculate out_1 , out_2

DEEP LEARNING BASICS: MLP

$f(x) = \frac{1}{1+e^{-x}}$, logistic activation function



The standard logistic function has an easily calculated derivative:

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

$$\frac{d}{dx}f(x) = \frac{e^x \cdot (1+e^x) - e^x \cdot e^x}{(1+e^x)^2}$$

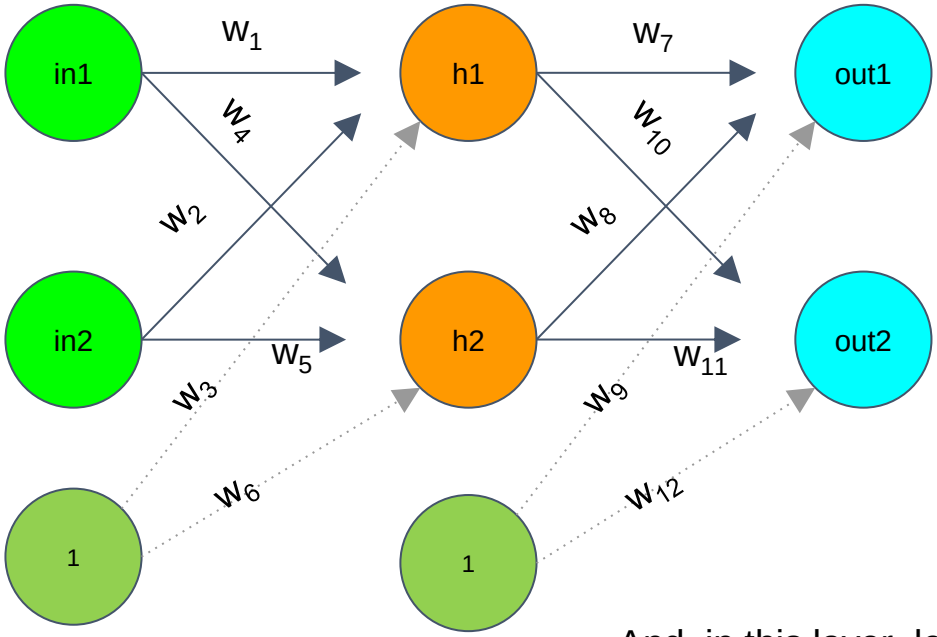
$$\frac{d}{dx}f(x) = \frac{e^x}{(1+e^x)^2} = f(x)(1-f(x))$$

$f(0.5) = 0.62245$ $f(1) = 0.73105$ $f(3) = 0.95257$

$f(0) = 0.5$ $f(2) = 0.88079$

$f(-3) = 0.04742$ $f(-1) = 0.26894$ $f(-0.5) = 0.37754$

DEEP LEARNING BASICS: MLP



$$h_1 = 0.1;\ h_2 = 1;$$
$$w_7 = 2;\ w_8 = -0.4; w_9 = -0.3;$$
$$w_{10} = -1;\ w_{11} = 1.5; w_{12} = 0.6;$$

$$out_1 = af\left(net_{out_1}\right) =$$
$$= af(h_1 * w_7 + h_2 * w_8 + w_9)$$

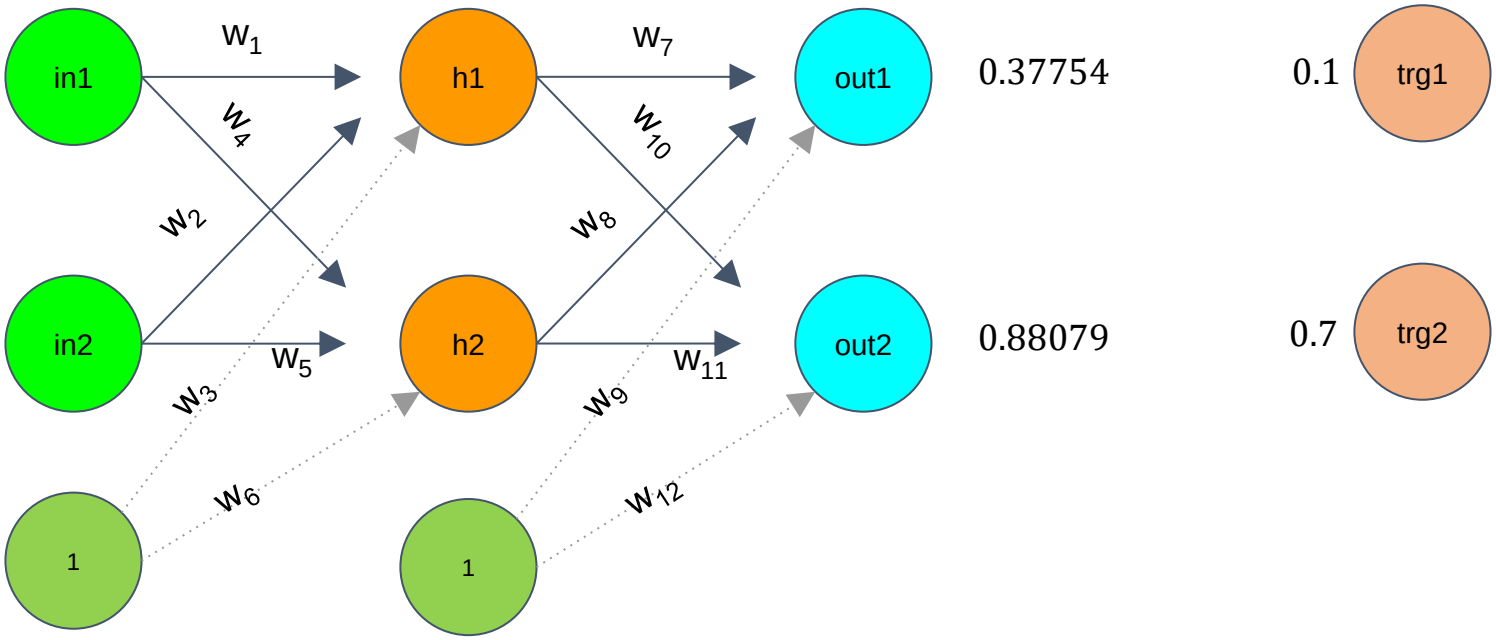
$$out_2 = af\left(net_{out_2}\right) =$$
$$= af(h_1 * w_{10} + h_2 * w_{11} + w_{12})$$

And, in this layer, let $af(x) = \frac{1}{1+e^{-x}}$, the logistic function

Calculate out_1, out_2

Tip: keep the values of net_{out_1}, net_{out_2}

DEEP LEARNING BASICS: MLP



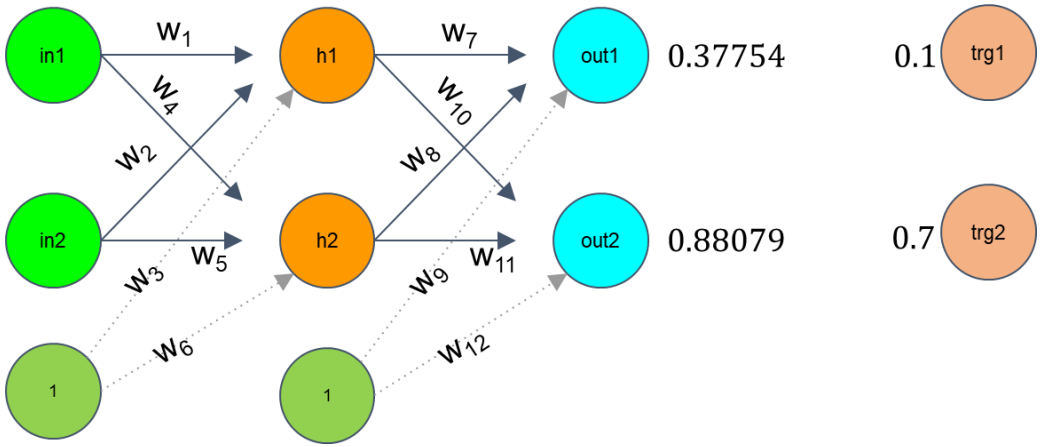
$$Total\ Error: J(\tilde{w}) = \frac{1}{2}\sum_{i=1}^{2}(out_i - trg_i)^2 = \frac{E_1 + E_2}{2} = 0.05485$$

E_1=0.0770284516

E_2=0.0326850241

DEEP LEARNING BASICS: MLP

$Total\ Error$: 0.05485



Backwards Pass

How much of the error is due to W7?

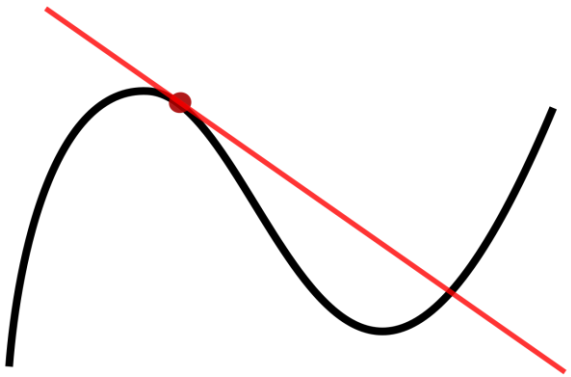
$$\frac{\partial E_{total}}{\partial W7} = ?$$

Chain rule!

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{d(g(x))} * \frac{d(g(x))}{d(x)}$$

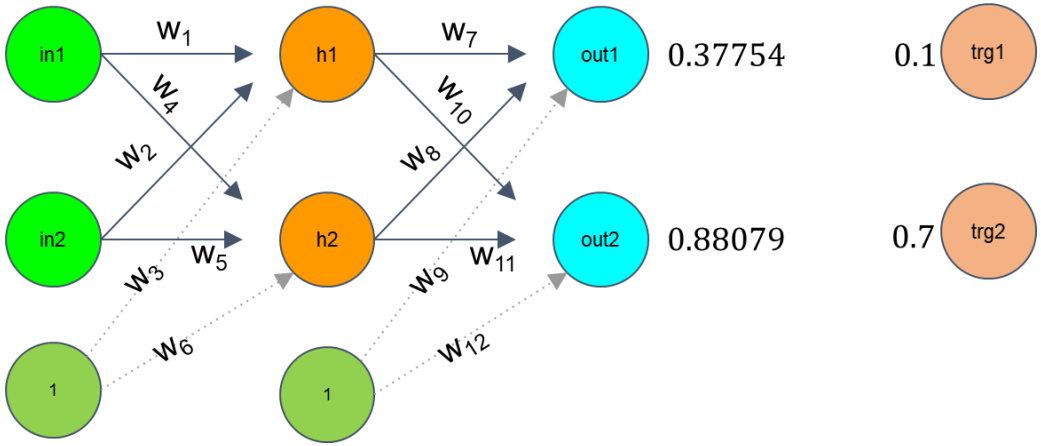
Reminder: Derivatives

Calculate the rate of change of a function based at any given point on its curve



Here it is the rate the error changes as a function of each weight of the network that interesses us.

DEEP LEARNING BASICS: MLP



$Total\ Error$: 0.05485

Backwards Pass

How much of the error is due to W7?

$$\frac{\partial E_{total}}{\partial W7} = ?$$

Chain rule!

$$\frac{\partial E_{total}}{\partial W7} = \frac{\partial E_{total}}{\partial out_1} * \frac{\partial out_1}{\partial net_{out1}} * \frac{\partial net_{out1}}{\partial W7}$$

Next iteration $W7_{it1} = W7_{it0} - \eta * \frac{\partial E_{total}}{\partial W7}$, with η known as « learning rate »

Let's get to some code: **tinyurl.com/8js9m8mn**