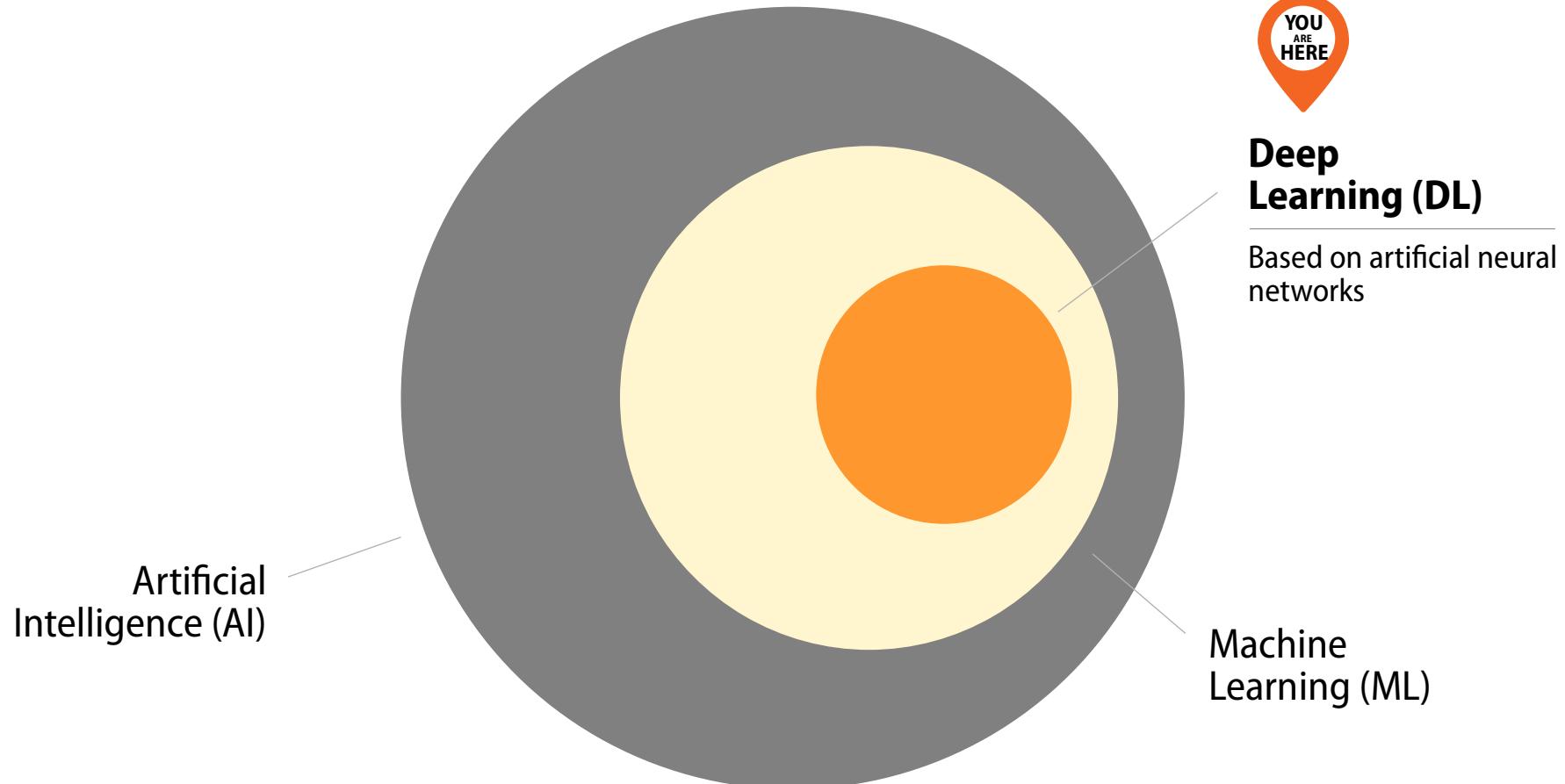


Intelligence Artificielle pour le Traitement d'Images

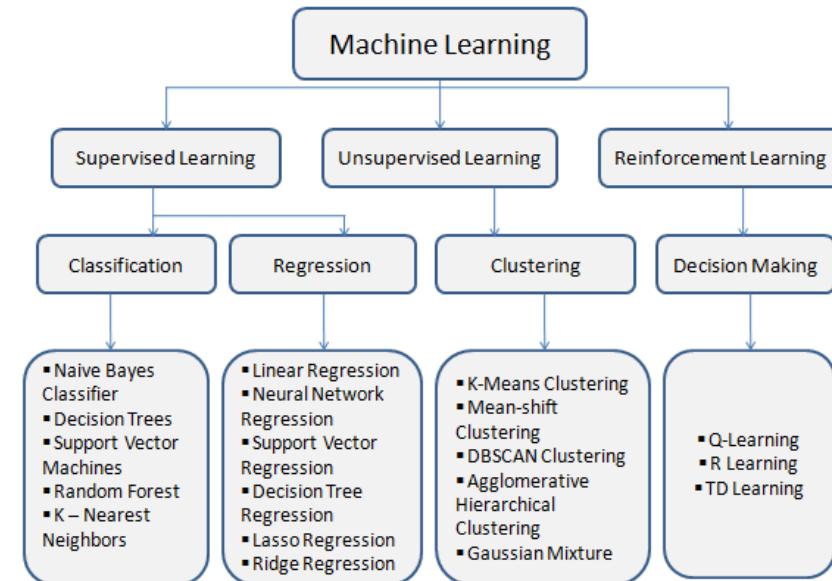
Apprentissage profond et vision par ordinateur

C'EST QUOI L'INTELLIGENCE ARTIFICIELLE ?

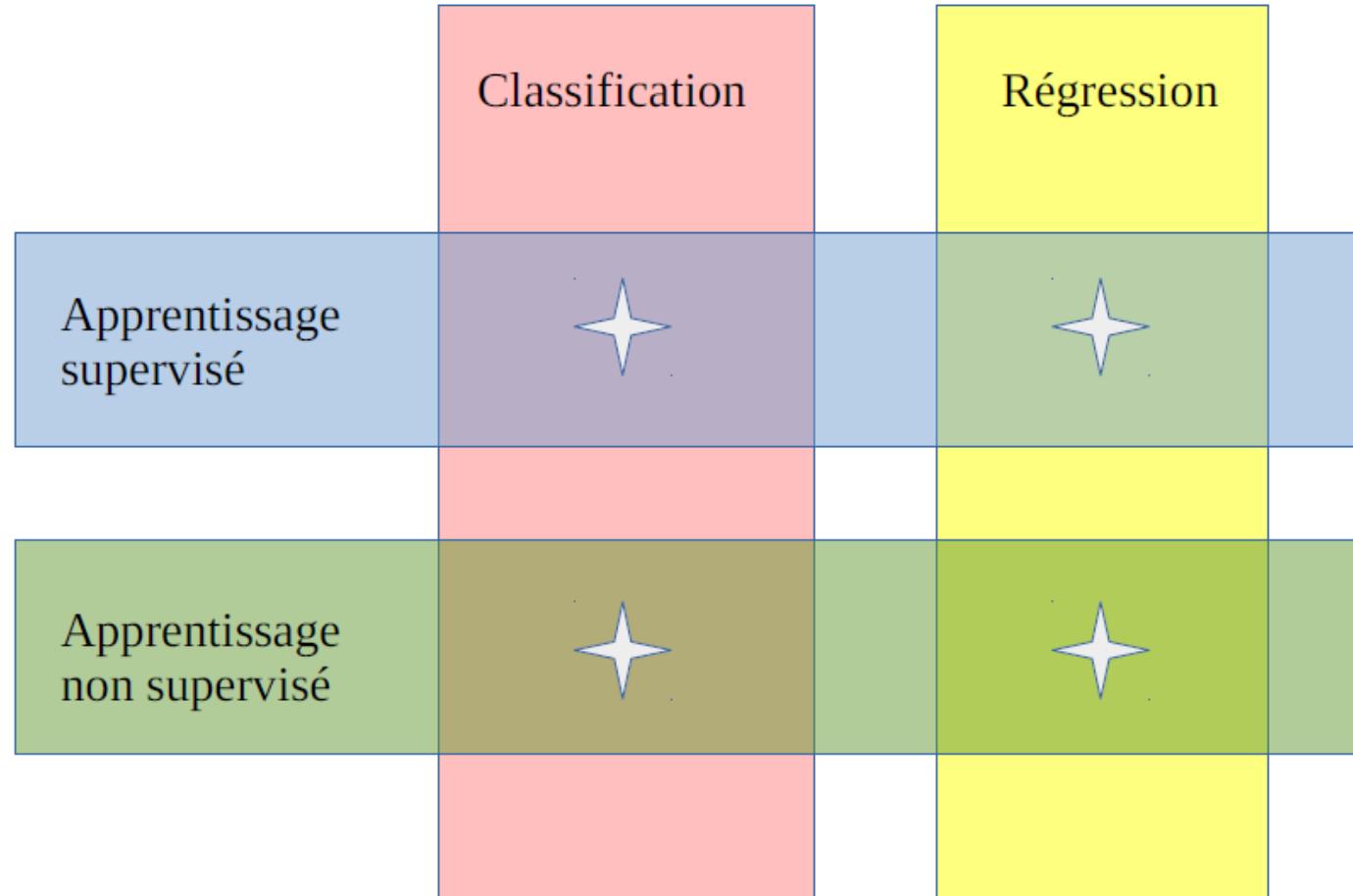


LES BASES : LE MACHINE LEARNING

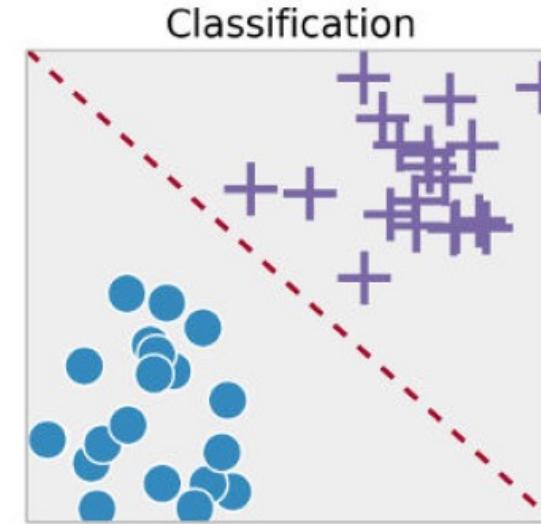
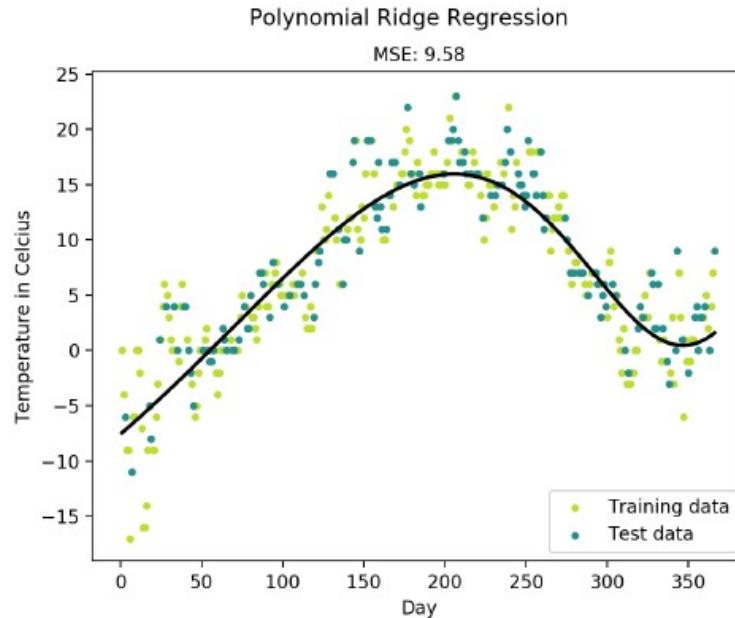
- Un modèle mathématique nécessite un certain nombre de paramètres pour fonctionner
 - Ex : le périmètre P d'un cercle de rayon r s'écrit : $P = 2 \times \pi \times r$
 - Que se passerait-il si on n'avait pas π mais juste plusieurs P et r ? Peut-on déduire une valeur de π ?
- Le ML couvre plusieurs techniques mathématiques/statistiques permettant d'apprendre des paramètres à partir des données
- Le ML "classique" comprend souvent des techniques qui s'appliquent facilement à des données numériques
 - Régression linéaire
 - Régression logistique
 - Arbres de décision
 - ...
- On regarde rapidement deux de ces méthodes



LES GRANDES FAMILLES



CLASSIFICATION VS REGRESSION

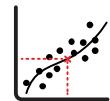


Régression

Prédire une variable quantitative

	150 K€
	400 K€
	120 K€
	100 K€

Tell me,
what's the
price ?



Classification

Prédire une classe (qualitative, discrète)



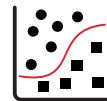
This is a cat



This is a rabbit

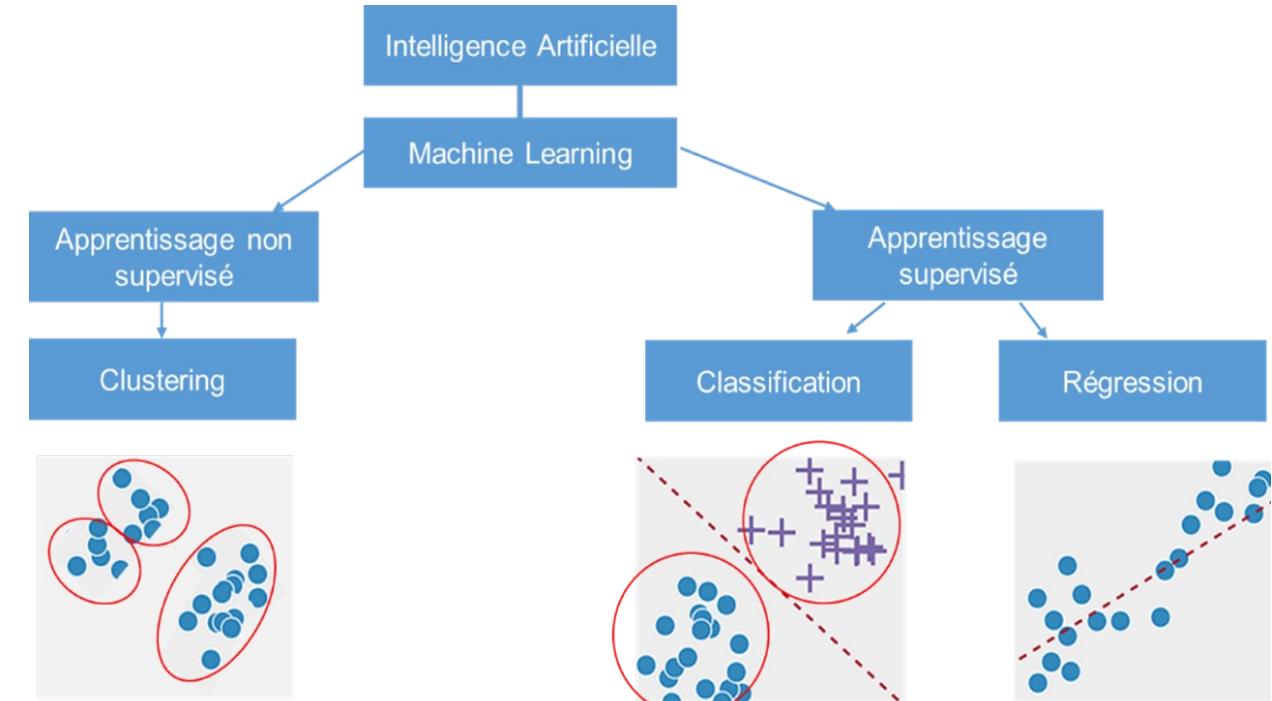


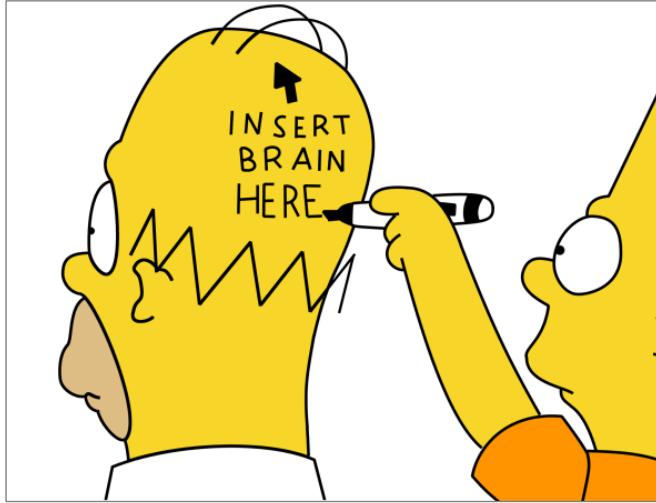
Tell me,
what is it ?



APPRENTISSAGE SUPERVISÉ VS NON-SUPERVISÉ

- Apprentissage supervisé :
 - Nécessite un jeu d'entraînement **X, y**
 - **X** : prédicteurs
 - **y** : variable à prédire
- Apprentissage non supervisé :
 - Nécessite un jeu d'entraînement X
 - Application principale : le clustering
 - Exemple : classer des situations météo en groupes homogènes





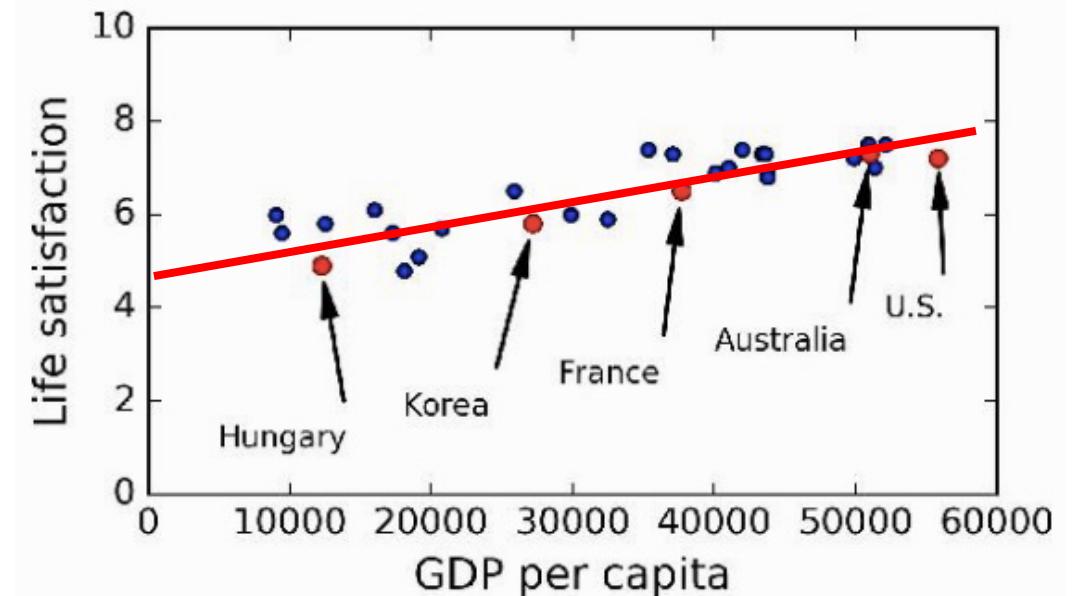
La Régression Linéaire

Une première méthode de IA

LA RÉGRESSION LINÉAIRE

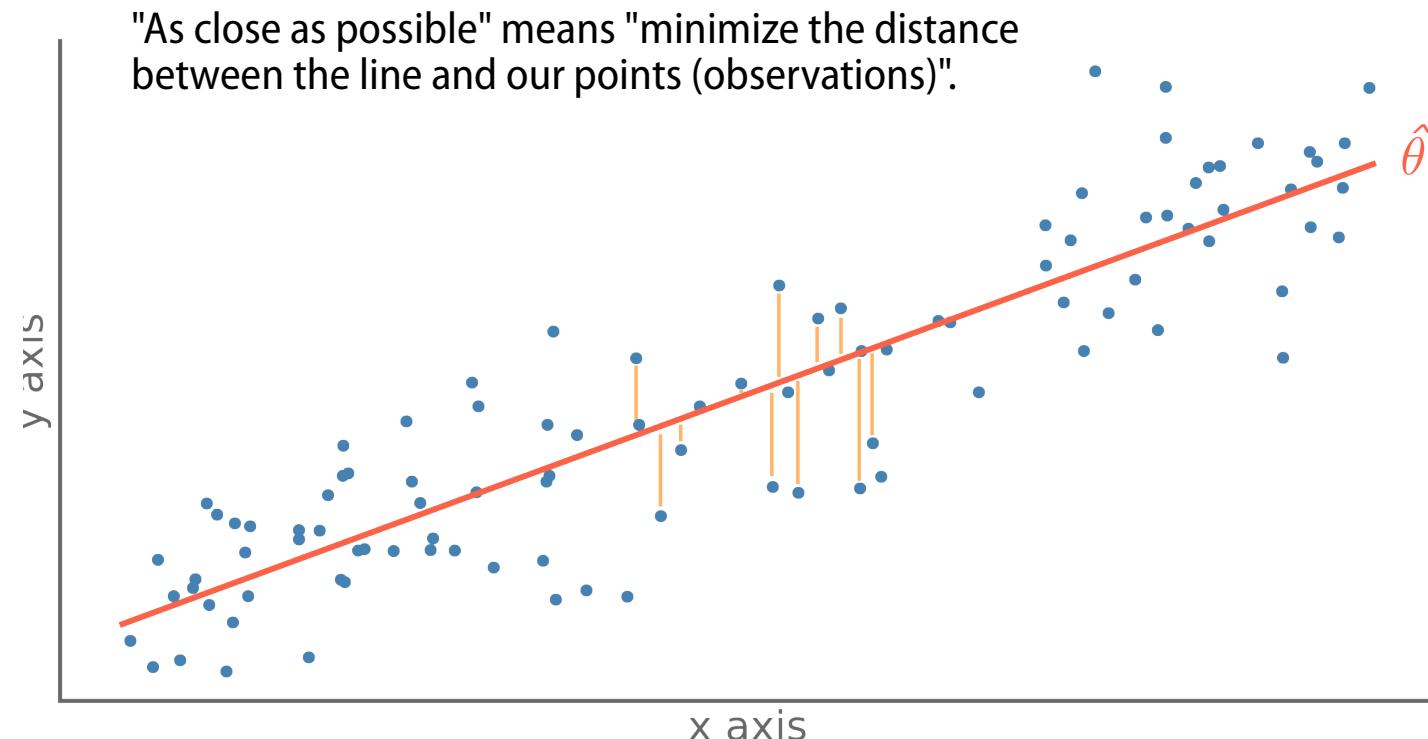
- À partir d'observations d'un phénomène, extraire un comportement "linéaire"
 - Exemple : prévoir le taux de satisfaction d'un pays en fonction de son revenu per capita
- Méthode d'apprentissage supervisé : un jeu d'entraînement X, y
 - X : Le revenu des pays
 - y : le taux de satisfaction
- Objectif : Trouver la droite qui se rapproche le plus du nuage de points

Country	GDP per capita (USD)	Life satisfaction
Hungary	12,240	4.9
Korea	27,195	5.8
France	37,675	6.5
Australia	50,962	7.3
United States	55,805	7.2



LA RÉGRESSION LINÉAIRE

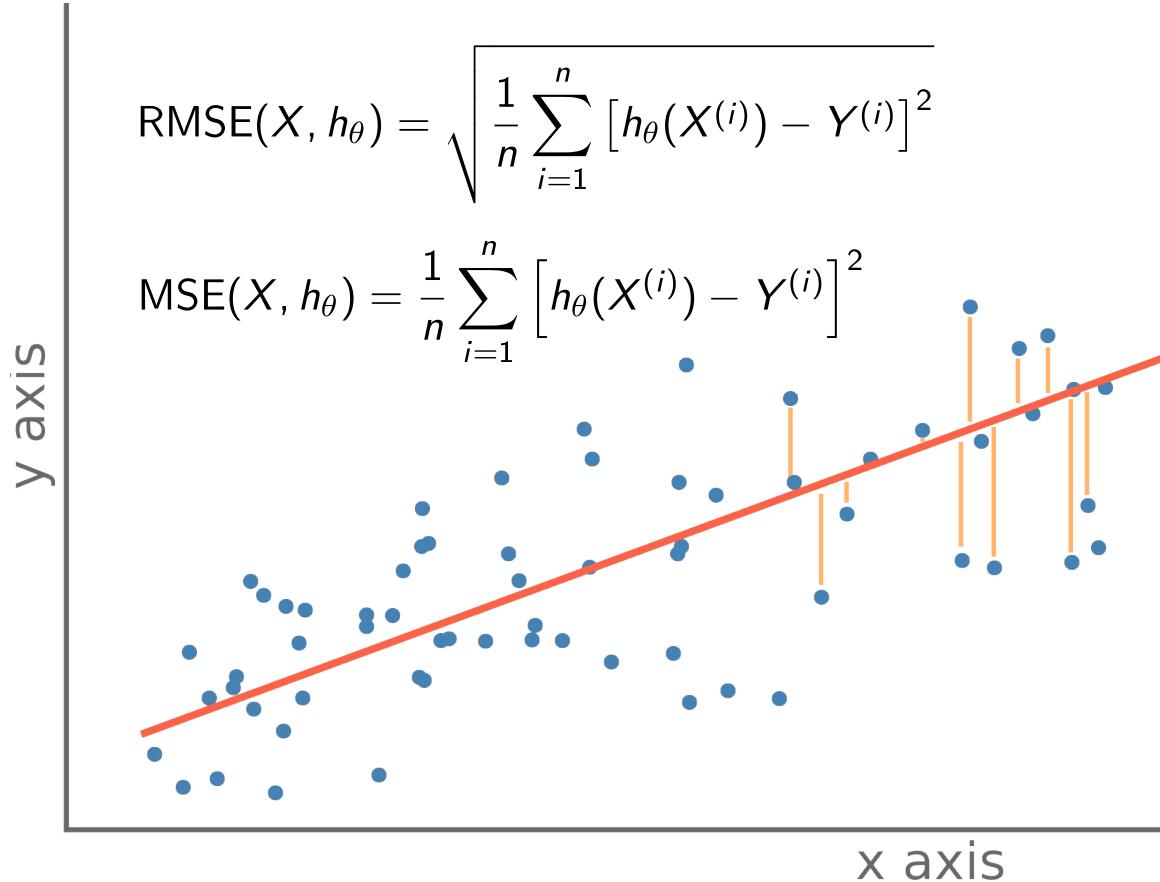
- Le modèle suppose une fonction de prédiction de forme
 - $f(x) = a_1x_1 + \dots + a_px_p + b = a \cdot x + b$
- L'apprentissage consiste à calculer les coefficients a et b qui minimisent l'erreur de prédiction (coût)
- Mais comment définir le coût ?



LA FONCTION DE COÛT (LOSS)

- C'est l'écart moyen entre les prédictions et la vérité terrain

$$MAE(X, h_{\theta}) = \frac{1}{n} \sum_{i=1}^n |h_{\theta}(X^{(i)}) - Y^{(i)}|$$



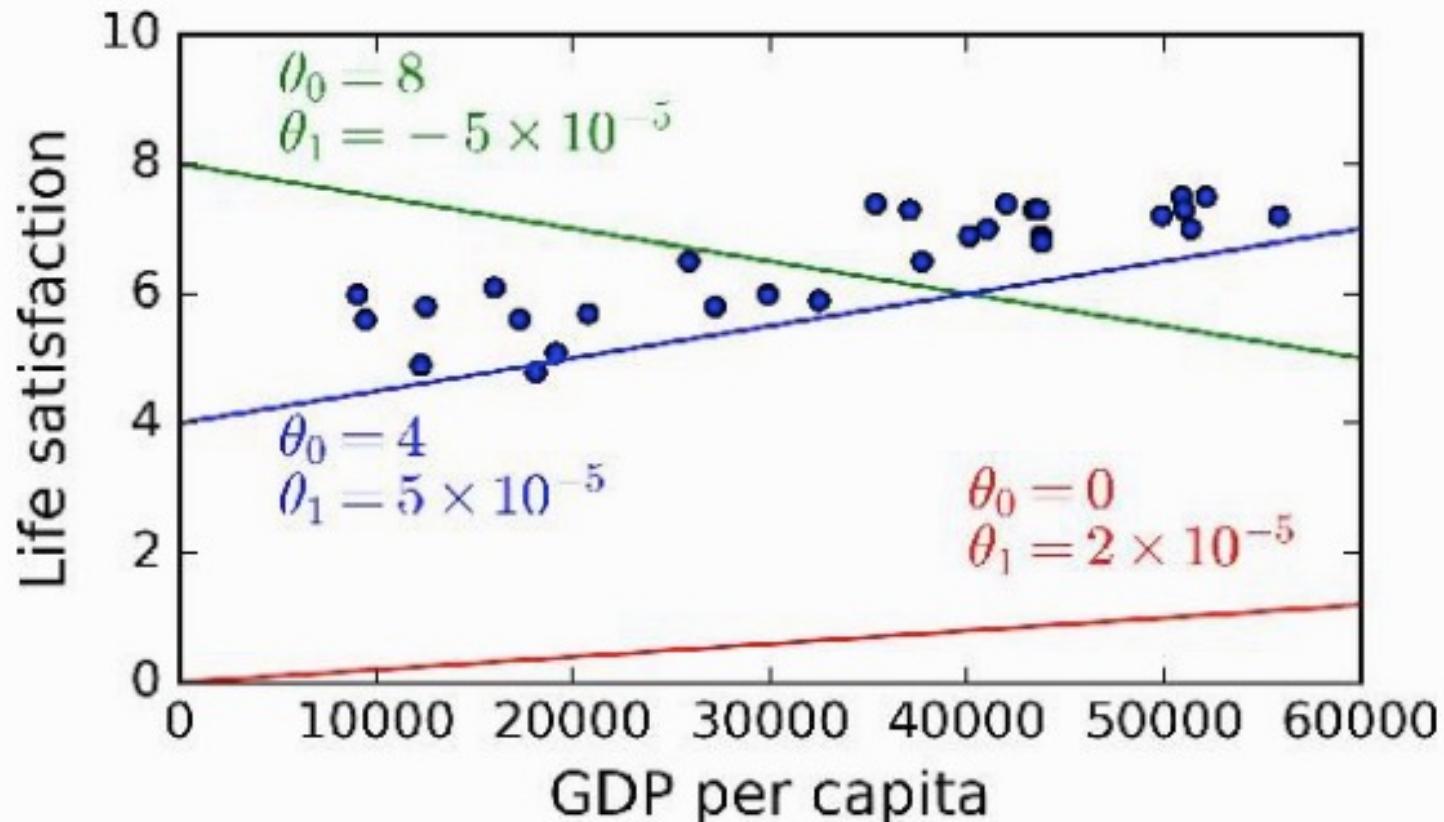
RMSE : Root Mean Square Error
Erreur quadratique moyenne

MSE : Mean Squared Error
Moyenne du carré des erreurs

MAE : Mean Absolute Error
Erreur absolue moyenne

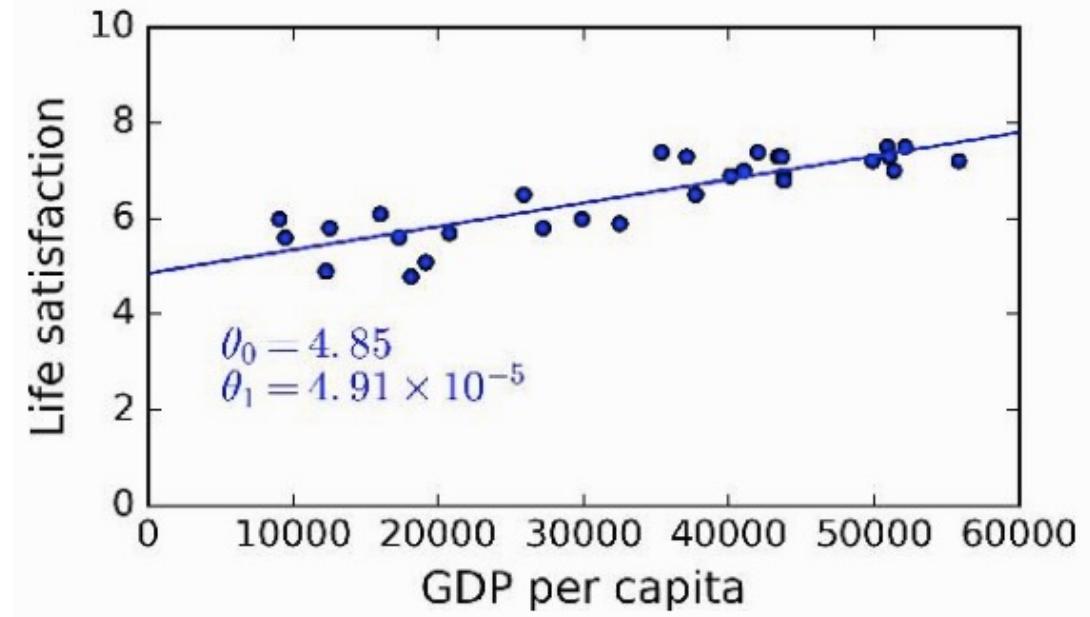
EXEMPLE : EST-CE QUE L'ARGENT REND HEUREUX ?

- Ce modèle a deux paramètres, θ_0 et θ_1 .
- On peut créer une formule simple et essayer plusieurs combinaisons
 - $satisfaction = \theta_0 + \theta_1 \times GDP_per_capita$

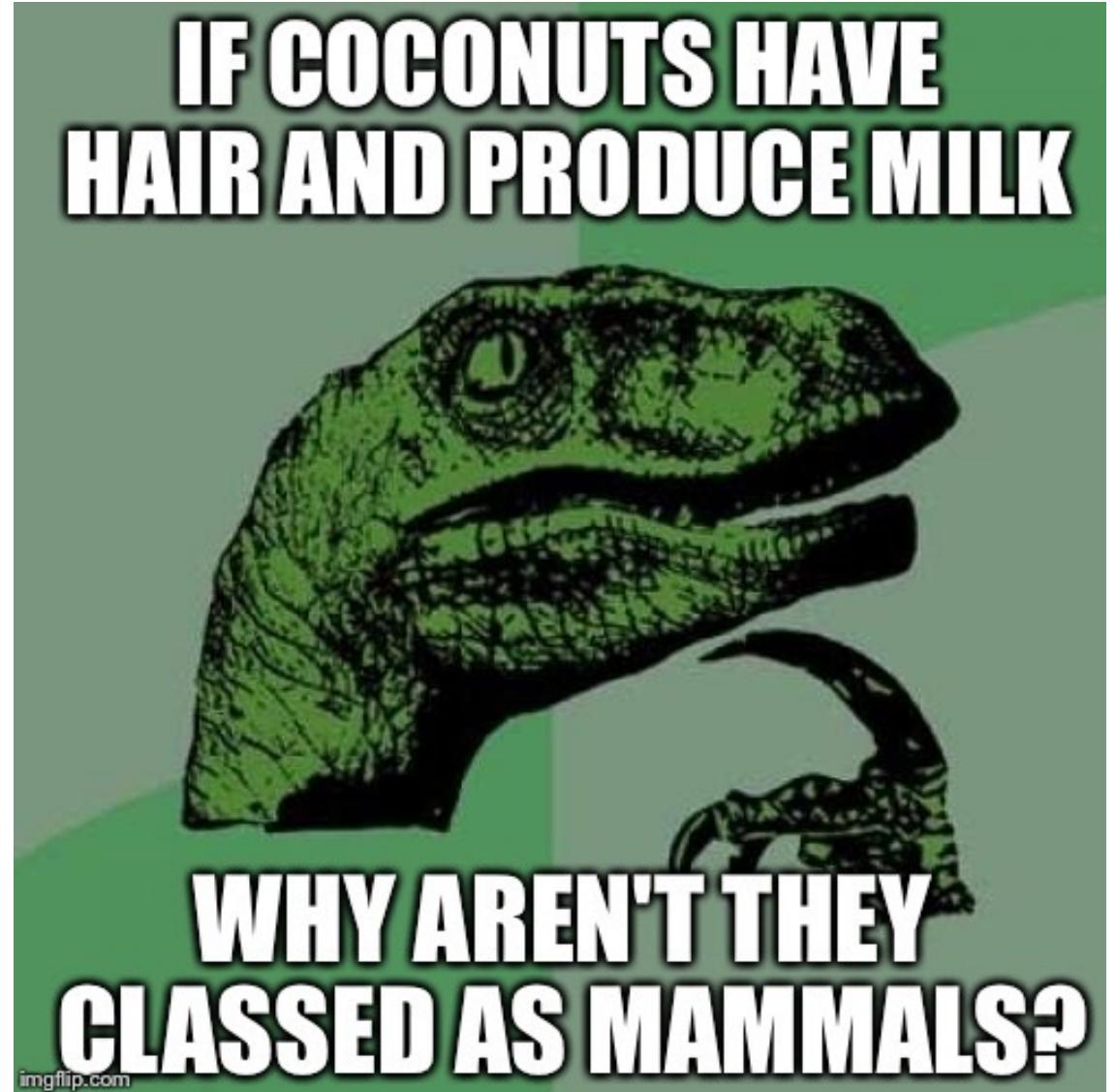


EXEMPLE : EST-CE QUE L'ARGENT REND HEUREUX ?

- Le modèle qui mieux correspond aux données est
 - $\theta_0 = 4.85$
 - $\theta_1 = 4.91 \times 10^{-5}$
- Grâce à ce modèle, on peut essayer d'estimer la satisfaction de la population de Chypre
 - `GDP_per_capita = 22875 USD`
 - `Estimation = 4.85 + 22875 * 4.91 * 10-5`
 - `Estimation = 5.96`

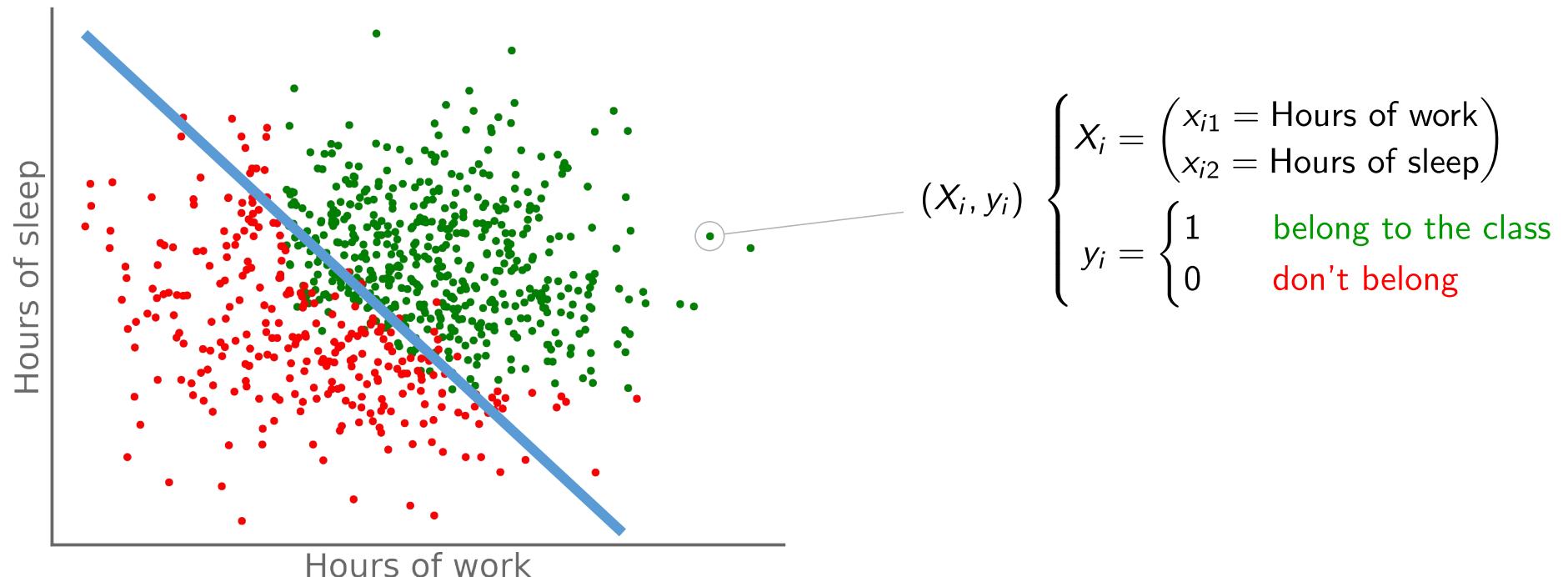


De la Régression à la Classification



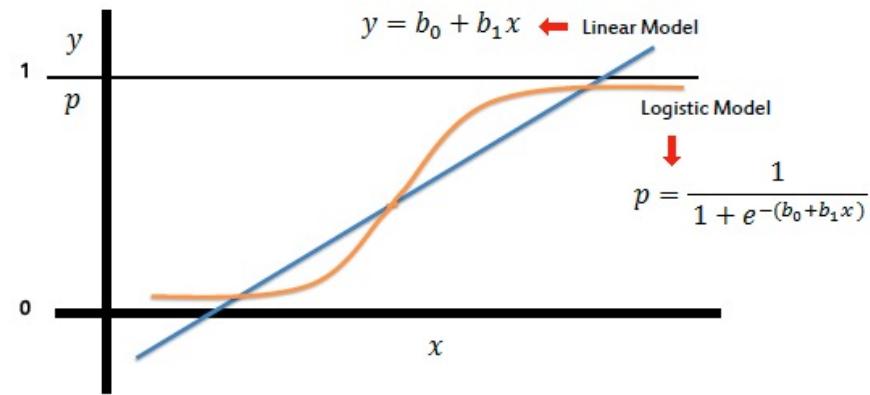
LE CAS DE LA RÉGRESSION LOGISTIQUE

- Au lieu de trouver la ligne qui est la moins éloignée des points, **on cherche la ligne qui mieux les sépare**
 - Une Régression Logistique sert à donner une probabilité d'appartenance à une classe

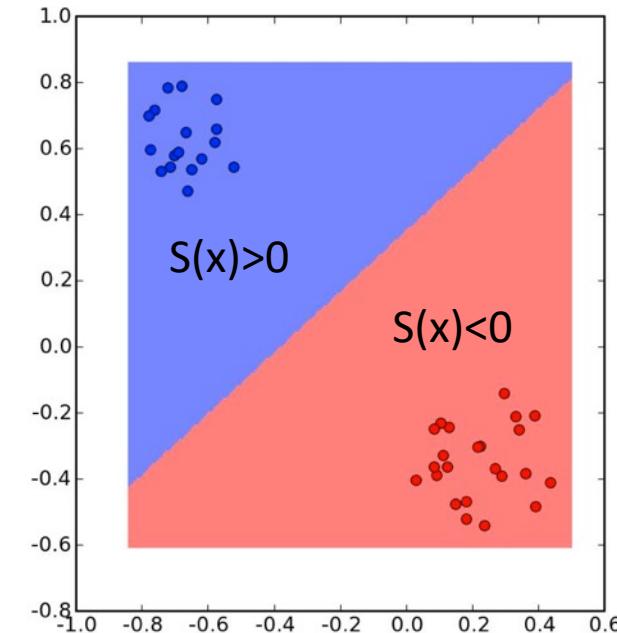


RÉGRESSION LOGISTIQUE

- Comme pour la régression linéaire, on cherche une fonction $S(x) = a_1x_1 + \dots + a_px_p$ qui doit délimiter deux groupes (classes) de données
 - Trouver des coefficients a tel que
 - $S(x)$ est positive lorsque les chances d'appartenir au groupe 1 sont grandes
 - $S(x)$ est négative si la probabilité est grande pour le groupe 0
- Il faut trouver un moyen de "recompenser" les bons scores
 - Une fonction d'interpolation $\text{logit}(S) = 1/[1 + \exp(-S)]$ est souvent utilisée pour exprimer cette probabilité

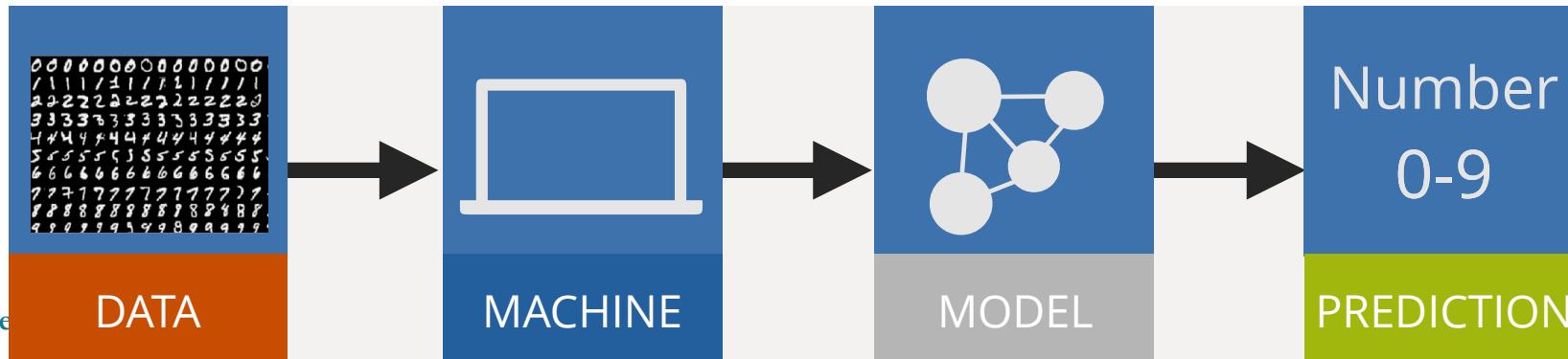


Luiz Angelo Steffenel



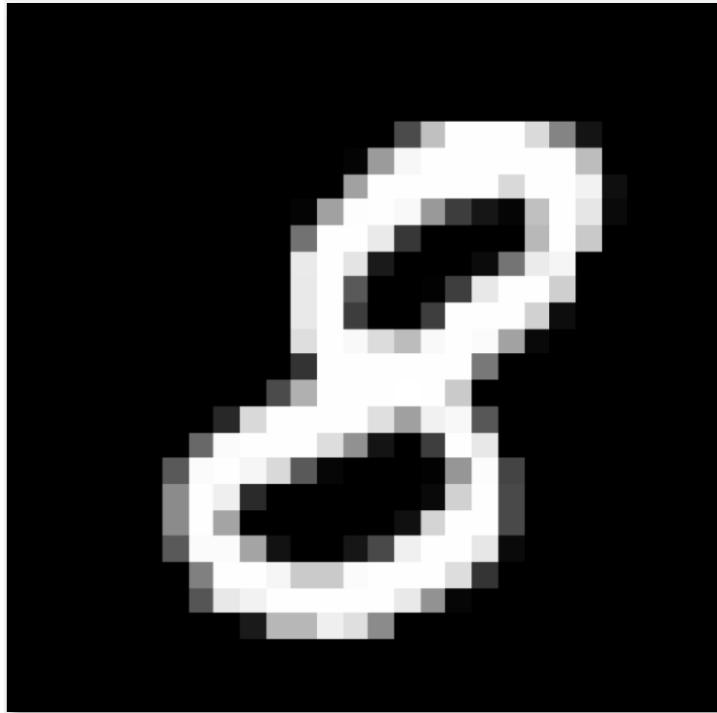
COMMENT ÇA MARCHE DANS LA PRATIQUE ?

- Exemple : le dataset MNIST
 - Ensemble de chiffres écrits à la main
- Objectif : identifier le chiffre
 - 0 à 9



MISE EN ROUTE

- Les chiffres sont des images, chaque pixel a une valeur numérique



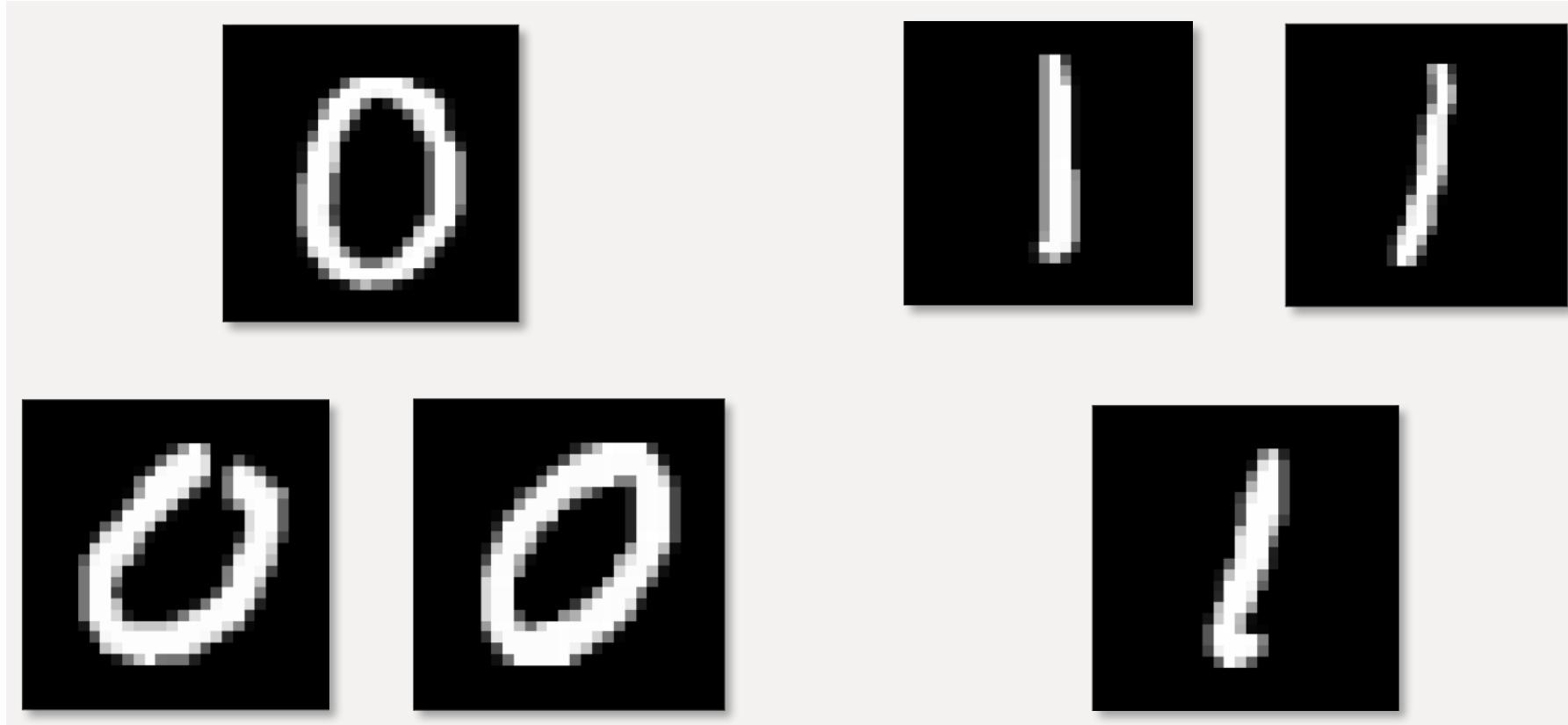
A thick black arrow pointing to the right, indicating the direction of the next section.

= X

MNIST Dataset of Handwritten Digits (Images)

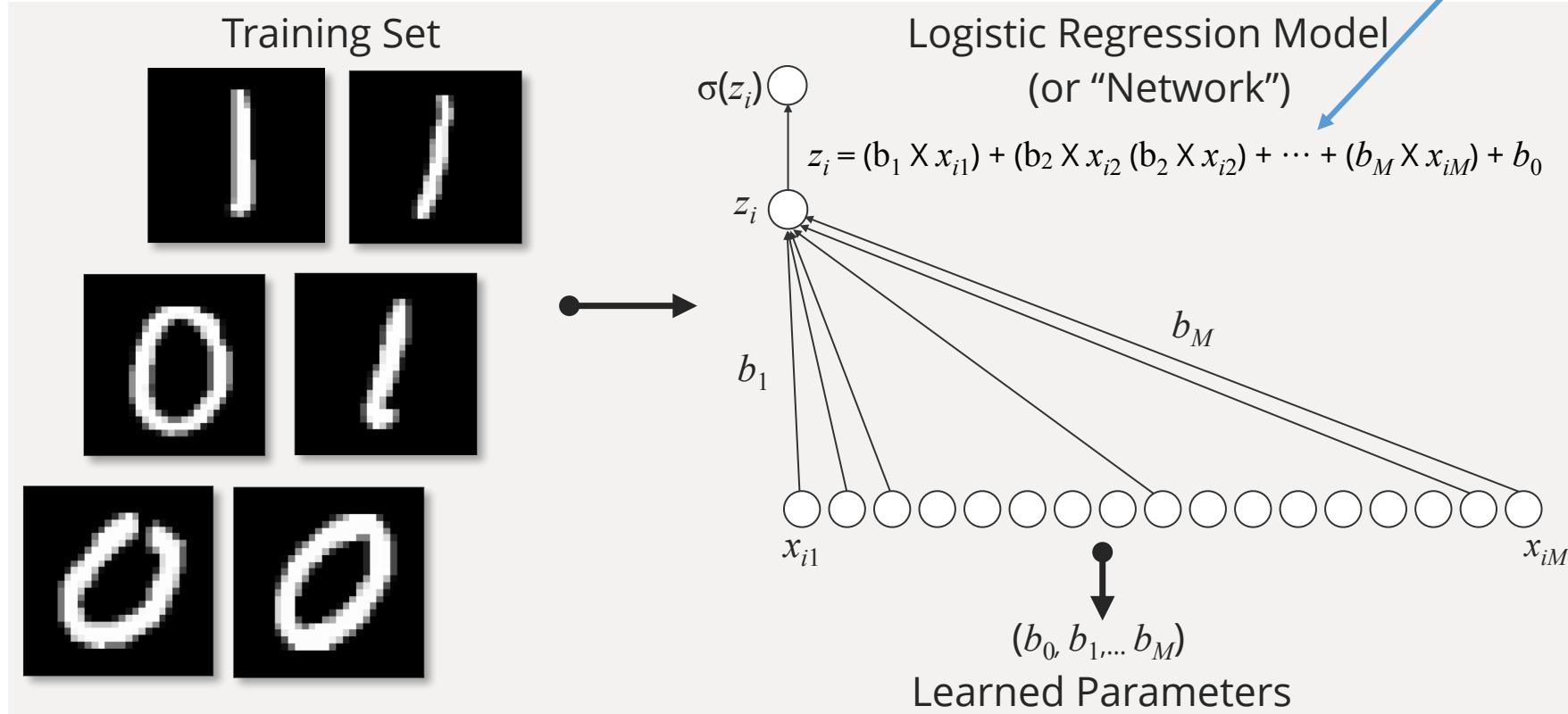
Yann LeCun (Courant Institute, NYU) and Corinna Cortes (Google Labs, New York) CC-by-SA 3.0
Luiz Angelo Steffenel ED SNI – Introduction au Deep Learning – 6 juin 2023
<http://yann.lecun.com/exdb/mnist/>

LE CAS SIMPLE : SORTIE BINAIRE



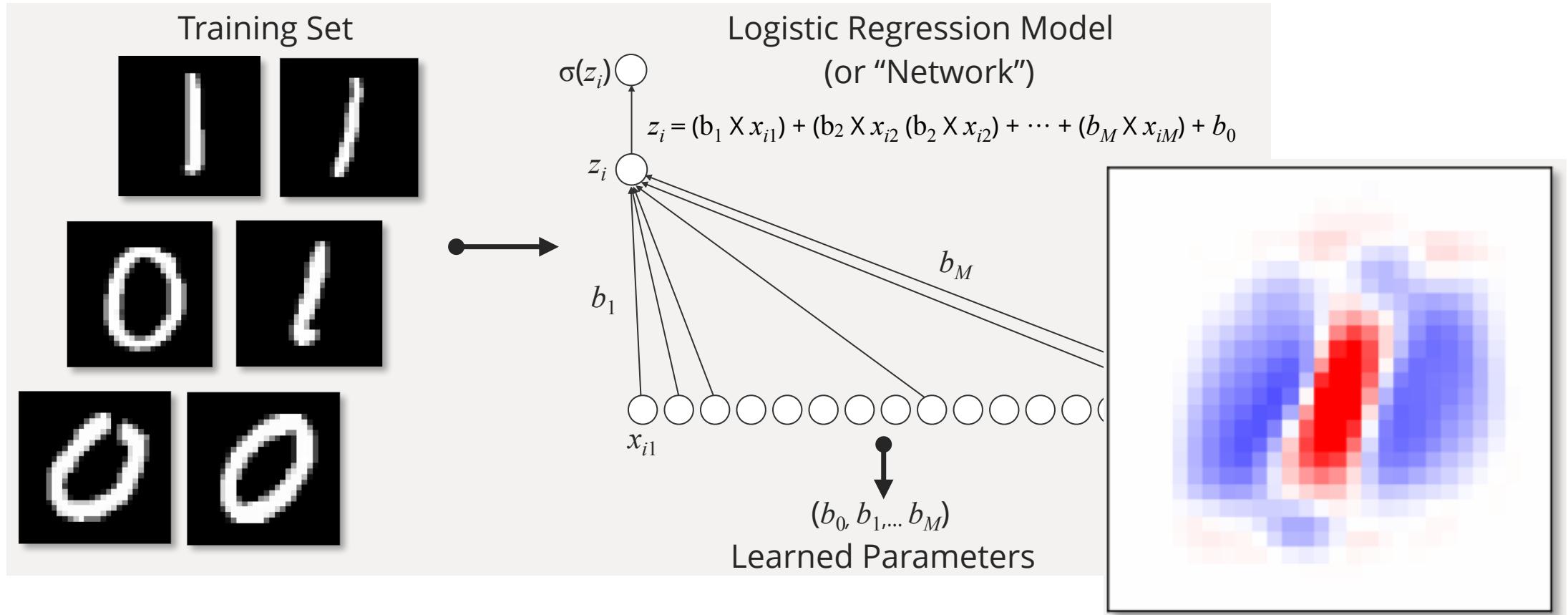
APPRENTISSAGE DU MNIST

C'est la même formule de la régression linéaire !!!



Chaque position de la matrice est étiquetée avec une valeur négative (0), positive (1) ou rien. Plus on a des 0 ou 1, plus la "case" est fortement marquée (poids de b)

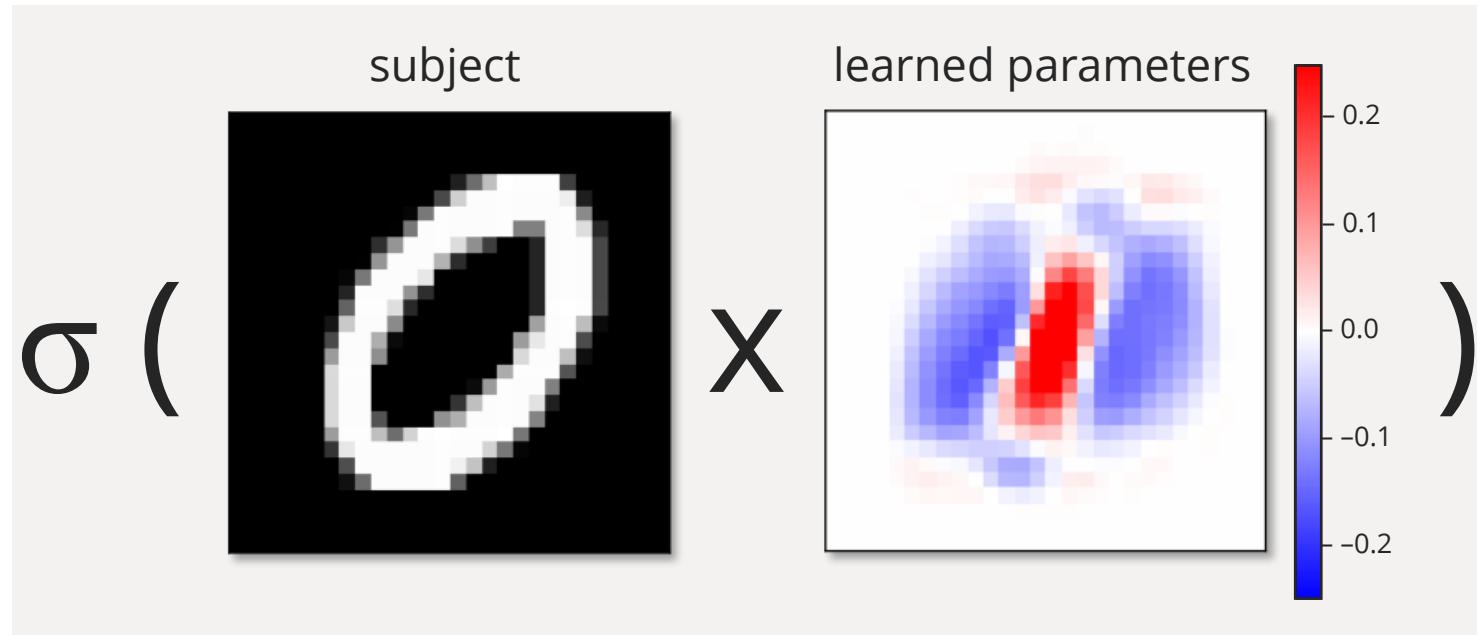
APPRENTISSAGE DU MNIST



Chaque position de la matrice est étiquetée avec une valeur négative (0), positive (1) ou rien. Plus on a des 0 ou 1, plus la "case" est fortement marquée (poids de b)

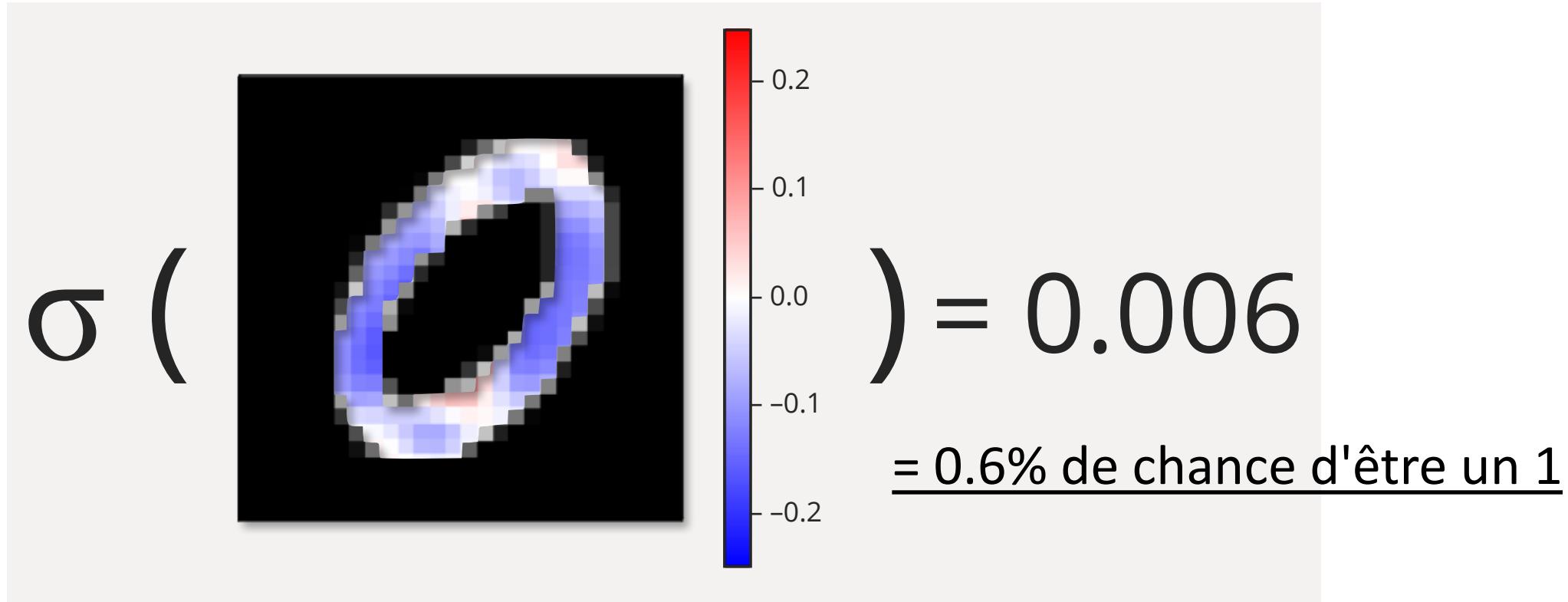
QUELLE EST LA PROBABILITÉ D'ÊTRE UN ZÉRO ?

- Pour un chiffre donné, on compare sa matrice avec la matrice entraînée :



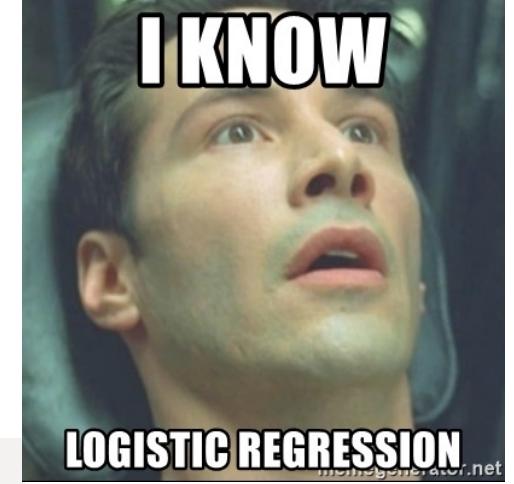
PROBABILITÉ D'UN ZÉRO

- La "superposition" donne une note pour les parties communes

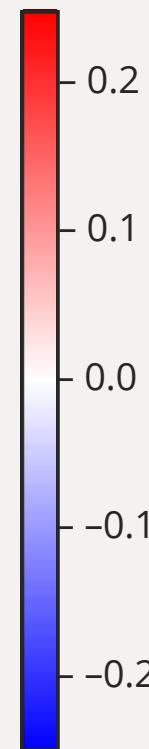
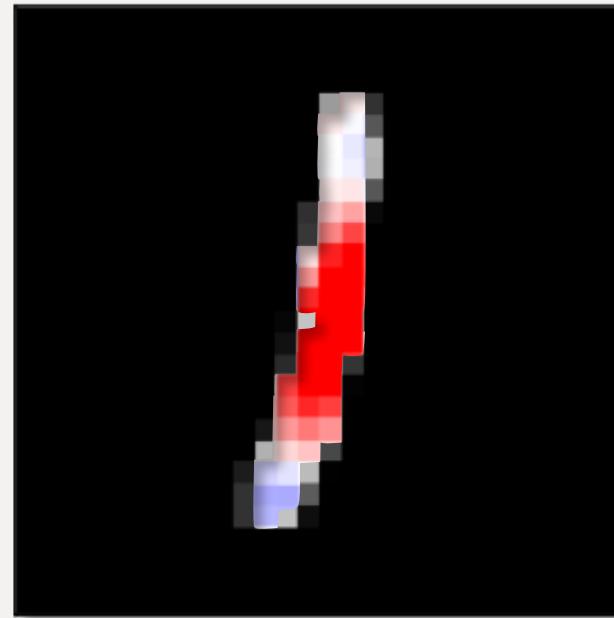


PROBABILITÉ D'UN UN

- La "superposition" donne une note pour les parties communes



$\sigma ($



) = 0.991

= 99,1% de chance d'être un 1

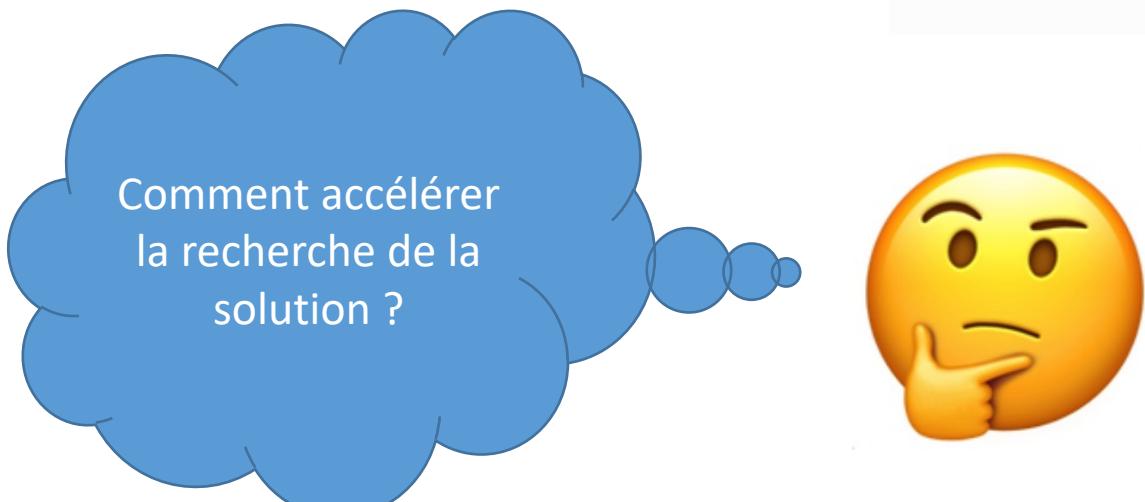
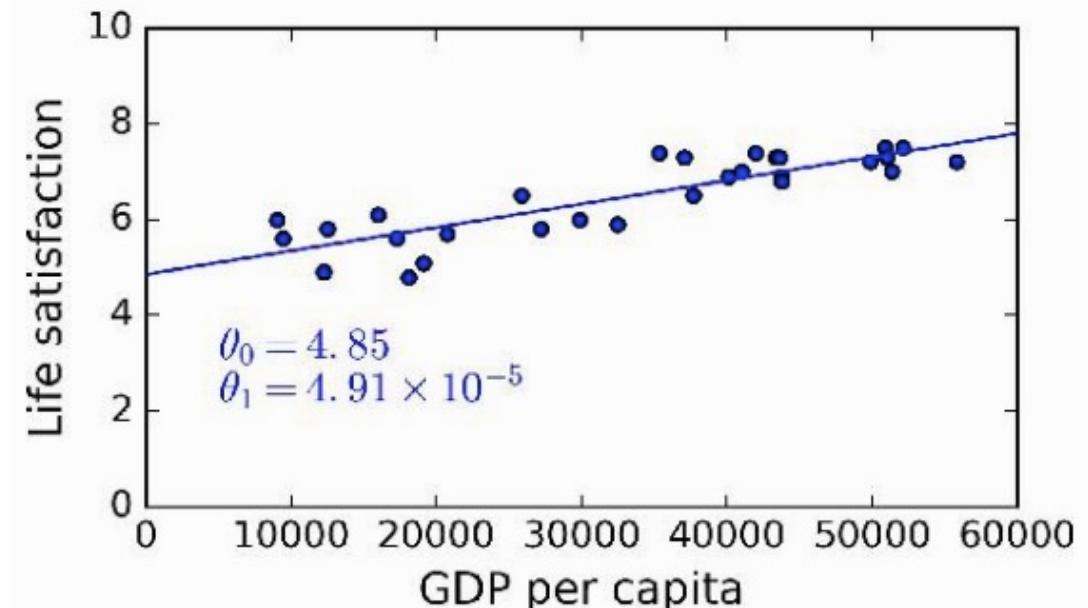
Comment entraîner un modèle ?

Un peu de jargon technique



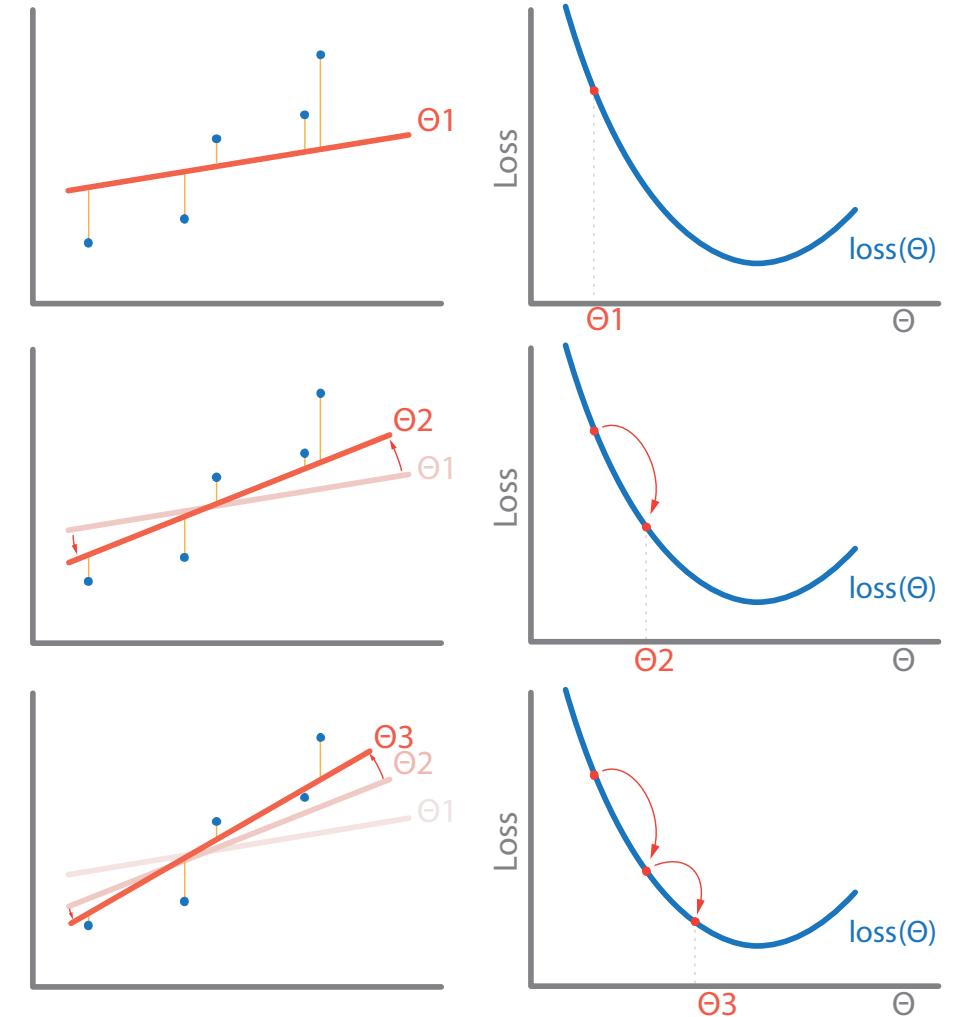
RAPPEL : EST-CE QUE L'ARGENT REND HEUREUX ?

- Le modèle qui mieux correspond aux données avait
 - $\theta_0 = 4.85$
 - $\theta_1 = 4.91 \times 10^{-5}$
- Mais comment trouver les bons paramètres ?
 - Tester **plein de combinaisons** de θ_0 et θ_1
- Problème : complexité $O(n^3)$



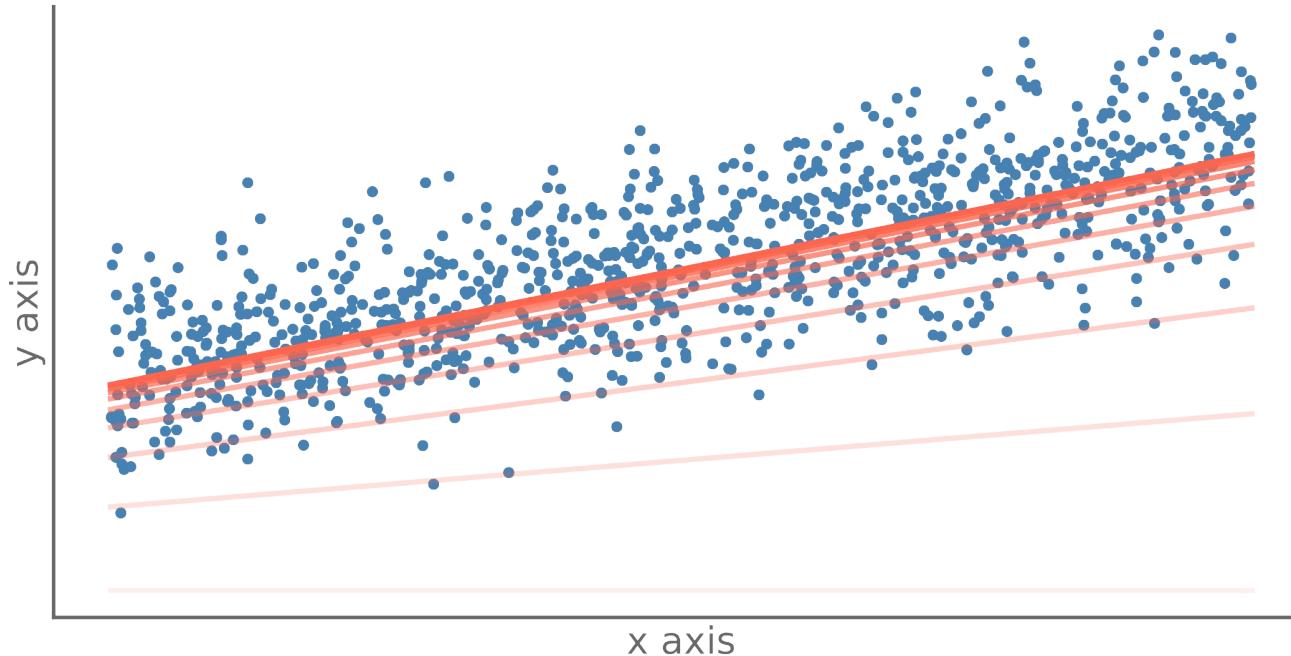
AVANCER EN FAISANT FACE À LA DESCENTE

- La descente de gradient permet de prendre l'élan tant que la pente nous est favorable
 - Avancer par des petits sauts, répéter plusieurs fois



EXEMPLE DE DESCENTE DE GRADIENT

- La solution itérative passe par $\Theta \leftarrow \Theta - \eta \cdot \nabla_{\Theta} MSE(\Theta)$ où η est le taux d'apprentissage (**learning rate**)



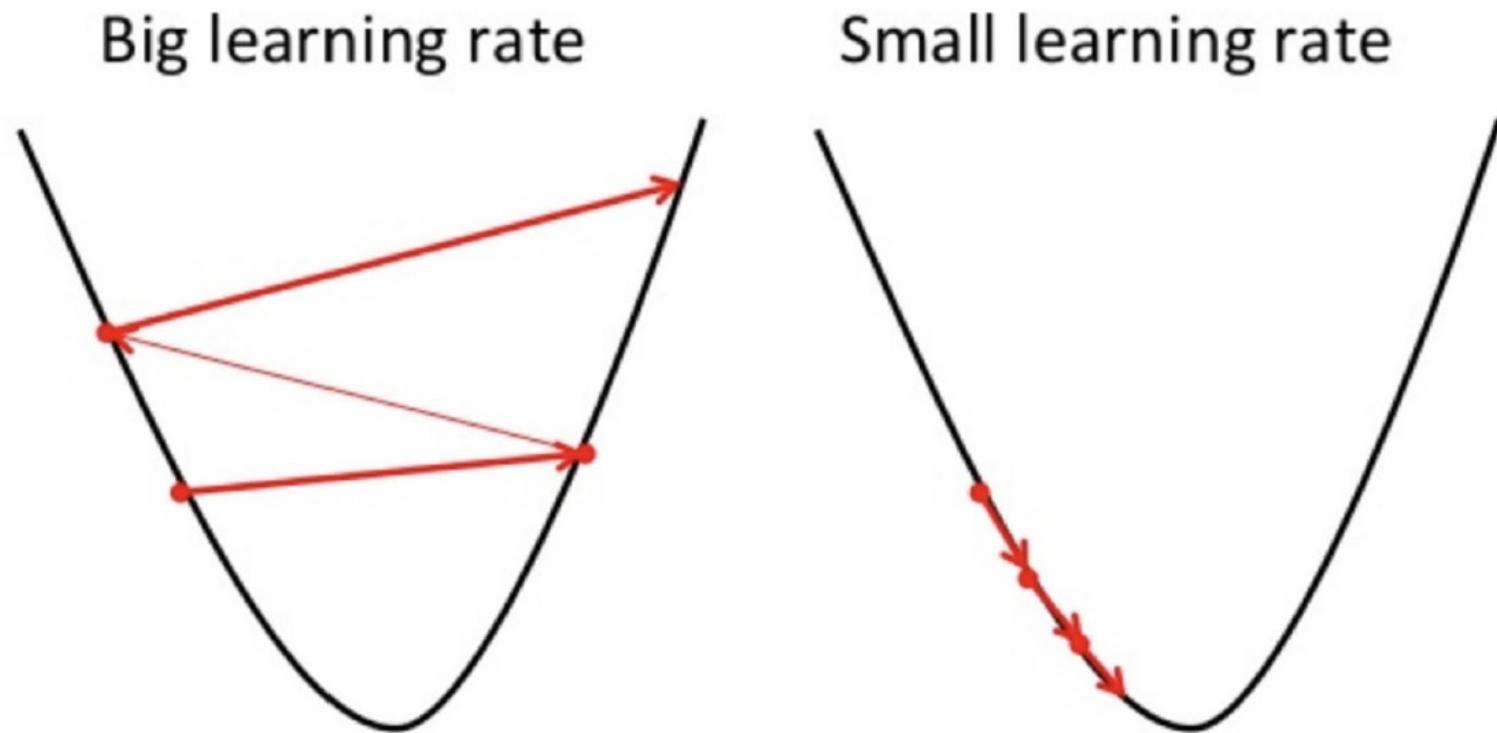
#i	Loss	Gradient	Theta
0	+12.481	-6.777	-3.388
20	+4.653	-4.066	-2.033
40	+1.835	-2.440	-1.220
60	+0.821	-1.464	-0.732
80	+0.455	-0.878	-0.439
100	+0.324	-0.527	-0.263
120	+0.277	-0.316	-0.158
140	+0.260	-0.190	-0.095
160	+0.253	-0.114	-0.057
180	+0.251	-0.068	-0.034
200	+0.250	-0.041	-0.020

$$MSE(X, h_{\theta}) = \frac{1}{n} \sum_{i=1}^n \left[h_{\theta}(X^{(i)}) - Y^{(i)} \right]^2$$

$$\nabla_{\theta} MSE(\Theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} MSE(\Theta) \\ \frac{\partial}{\partial \theta_1} MSE(\Theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} MSE(\Theta) \end{bmatrix} = \frac{2}{m} X^T \cdot (X \cdot \Theta - Y)$$

INFLUENCE DU PAS CHOISI (LEARNING RATE)

- Le choix du pas peut influencer le temps de convergence mais aussi le résultat
- Pas trop grand : risque de "sortir" du minimum
- Pas trop petit : risque de rester coincé dans un minimum local



EVALUATION D'UN MODÈLE DE CLASSIFICATION

- Dans une régression linéaire, on utilise la distance (MSE/RMSE, MAE)
- Dans le cas d'une classification (régression logistique), comment évaluer un modèle ?
- Plusieurs possibilités
 - L'accuracy (taux de réussite, justesse)
 - Le recall (rappel)
 - Le F1 score
 - Une matrice de confusion
 - ...
- L'accuracy est l'un des critères permettant d'évaluer les modèles de classification
 - Désigne la proportion des prédictions correctes effectuées par le modèle
- Attention, il y a aussi une métrique Precision (\neq accuracy)

$$Accuracy = \frac{\text{nombre de prédictions correctes}}{\text{Nombre total de prédictions}} = \frac{TN + TP}{TN + TP + FN + FP}$$

- Precision évalue la capacité à ne pas indiquer un label lorsqu'il est faux

$$Precision = \frac{TP}{TP + FP}$$

EVALUATION D'UN MODÈLE DE CLASSIFICATION

- Le **rappel (Recall)** : un pourcentage à maximiser
 - Le rappel est l'un des critères permettant d'évaluer la sensibilité du modèle
 - Le rappel correspond au nombre de documents correctement attribués à la classe i par rapport au nombre total de documents identifiés comme appartenant à la classe i

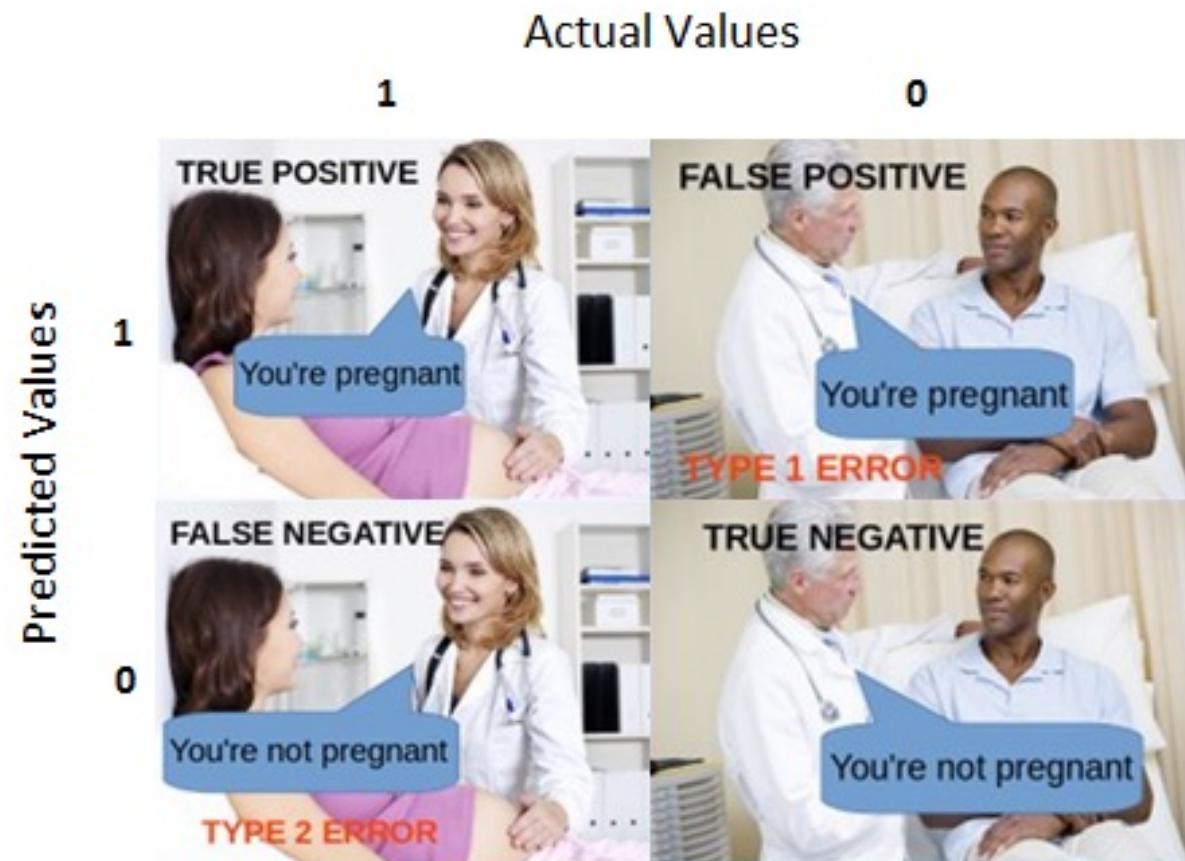
$$Rappel = \frac{\text{nombre de vrai positifs}}{\text{Nombre de vrai positifs} + \text{faux négatifs}} = \frac{TP}{TP + FN}$$

- Le **F1-score** : la moyenne harmonique entre la accuracy et le rappel
 - Favorise les cas où nous avons une haute accuracy et haut recall

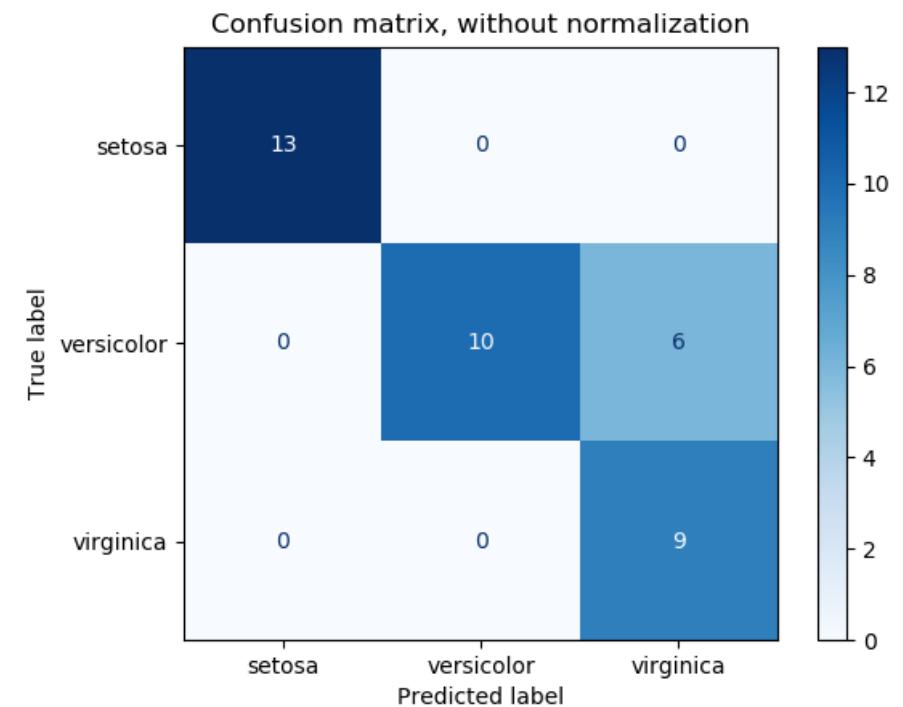
$$F1 = 2 * \frac{\text{accuracy} * \text{rappel}}{\text{accuracy} + \text{rappel}}$$

MATRICE DE CONFUSION

Cas binaire (deux classes seulement)



Cas multi-classe



QUELLE MÉTRIQUE FAVORISER ?

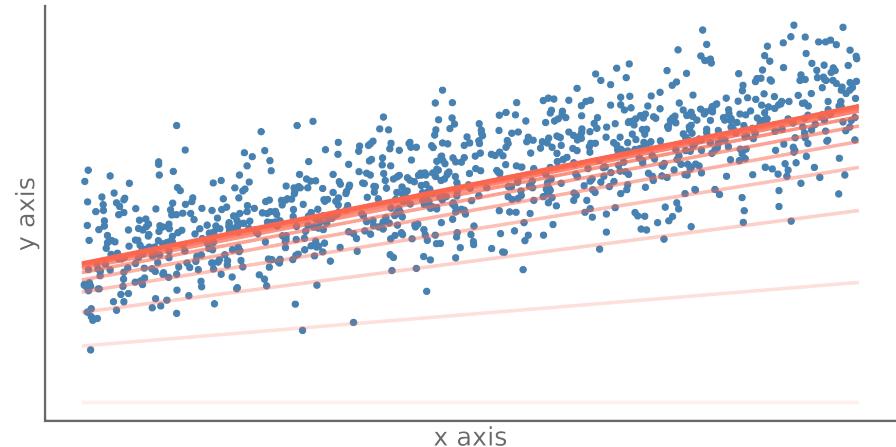
- Tout dépend de l'objectif
- Une messagerie pourra laisser passer quelques spams mais devra éviter au maximum de classer en tant que spam un mail qui n'en n'est pas un (faux negatif)
 - L'utilisateur se chargera d'effacer les spam qui passent le filtre
- Un algorithme de reconnaissance de cancer préférera se tromper et détecter un cancer (faux positif)
 - Le patient aura donc des examens supplémentaires et un avis éclairé du médecin
- Il y a aussi d'autres métriques spécifiques pour certains types de problème
 - IOU (segmentation sémantique), mAP (détection d'objets)...

AUTRE PROBLÈME : LA TAILLE DES DONNÉES

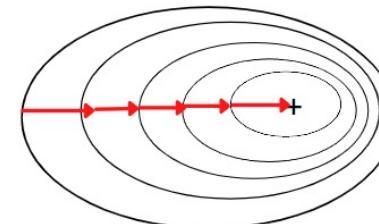
- Que faire lorsqu'on a trop de données ?
 - Si n est très grand et X de dimension élevée, alors calculer la somme devient très long, voire interminable...

$$MSE(X, h_\theta) = \frac{1}{n} \sum_{i=1}^n \left[h_\theta(X^{(i)}) - Y^{(i)} \right]^2$$

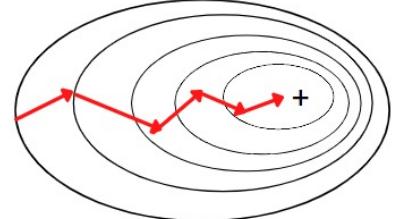
- Solution : la descente de gradient stochastique
 - On itère sur les échantillons un par un
 - Lent et convergence plus chaotique
- Un compromis : la descente par mini-batch
 - On estime le gradient sur k échantillons à la fois (par exemple 32 échantillons)
 - C'est la solution utilisée dans la pratique



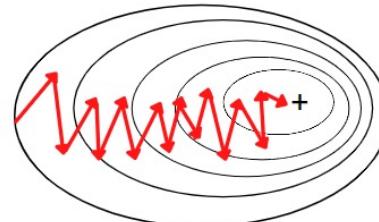
Batch Gradient Descent



Mini-Batch Gradient Descent

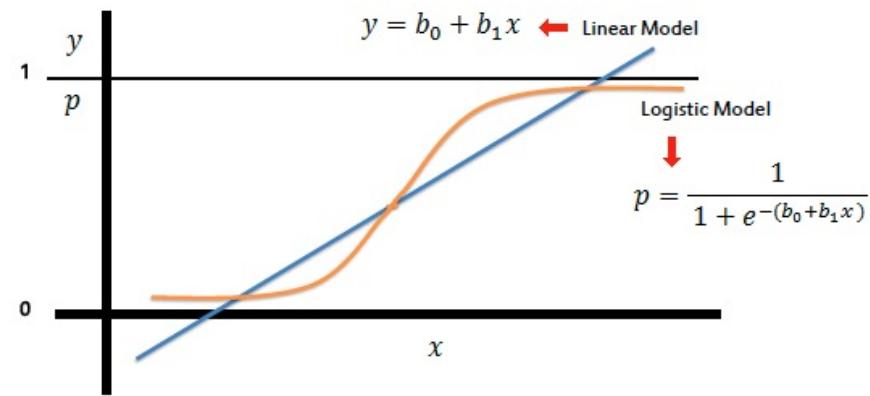


Stochastic Gradient Descent

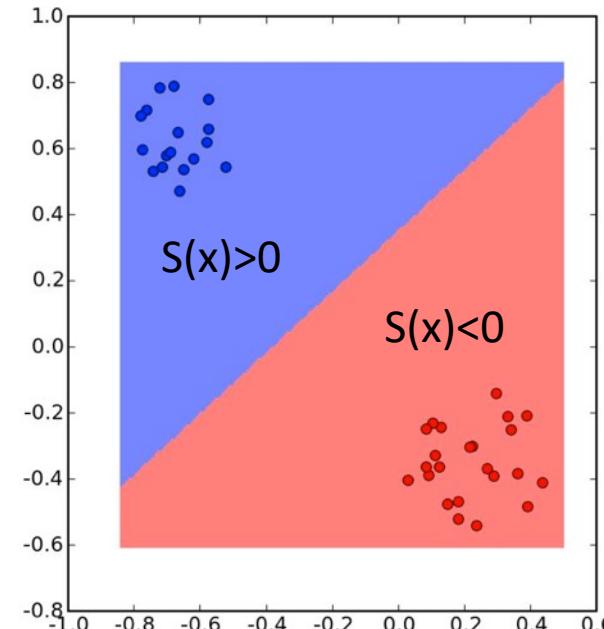


FONCTIONS D'ACTIVATION

- Dans une régression logistique, on cherche une fonction $S(x) = a_1x_1 + \dots + a_px_p$ qui sépare les deux groupes
- Il faut trouver un moyen de récompenser les bons scores
 - Une fonction linéaire ne suffit pas
 - Une fonction d'interpolation $\text{logit}(S) = 1/[1 + \exp(-S)]$ est souvent utilisée pour exprimer cette probabilité
- Selon la nature des données (ou des résultats attendus), d'autres fonctions d'activation sont utilisées

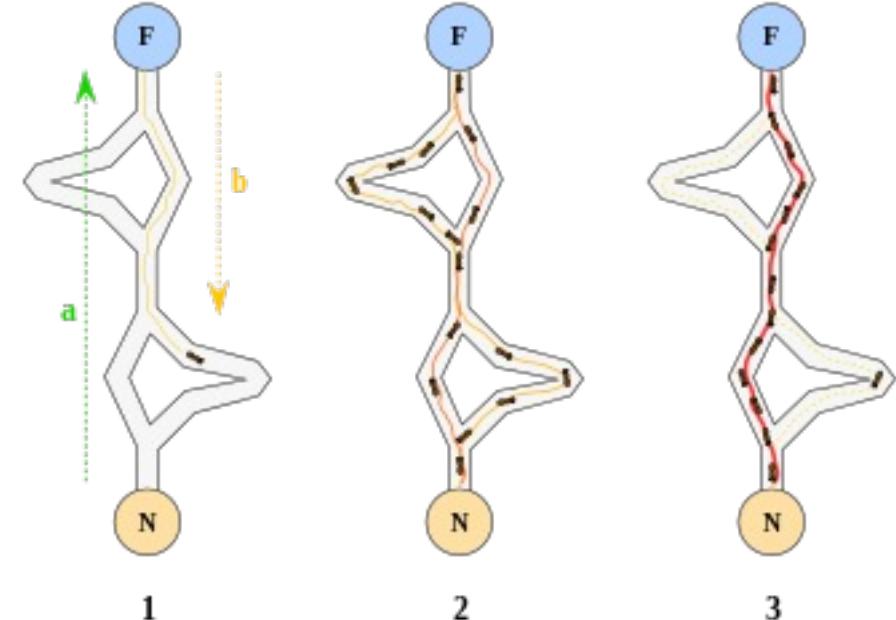


Luiz Angelo Steffenel



LA NOTION D' EPOQUE

- Dans la descente de gradient, on estime le « gradient »
 - échantillon par échantillon ou
 - par mini-batches de quelques échantillons
- Une passe complète sur le jeu de données s'appelle :



- Souvent un entraînement fait plusieurs passes sur les données
 - Sert à affiner les poids des modèles, un peu comme les traces de phéromones des fourmis

LES HYPERPARAMÈTRES

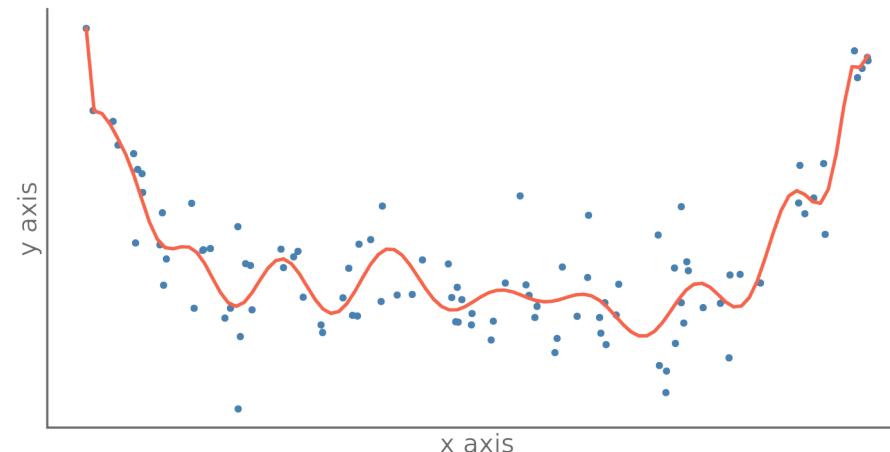
- Que peut-on modifier dans un modèle de Machine Learning ?
 - Le type et la complexité du modèle
 - La régression linéaire est un modèle simple, mais on peut le complexifier : polynôme de degré n, random forest, réseaux de neurones...
 - Certains paramètres spécifiques du modèle
 - Pour un réseau de neurones : nombre de couches, nombre de neurones par couche...
 - Le learning rate
 - La taille des mini-batches
- Comment choisir ces hyperparamètres ? C'est tout un problème...

PRÉPARATION DES DONNÉES

- Première idée : utiliser toutes les données

Jeu de données d'entraînement

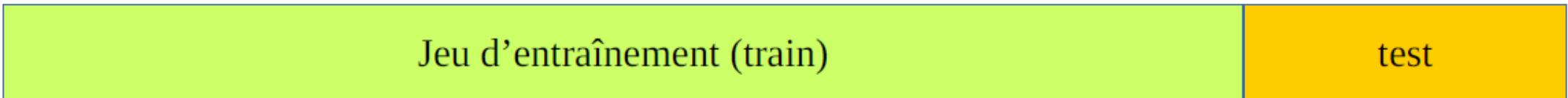
- Pas bon : on risque de surentraîner (apprendre par cœur) et le modèle ne pourra pas généraliser



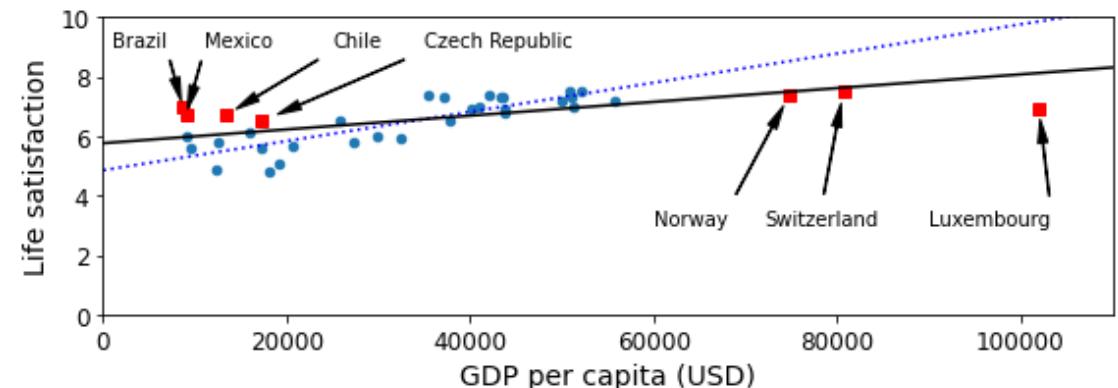
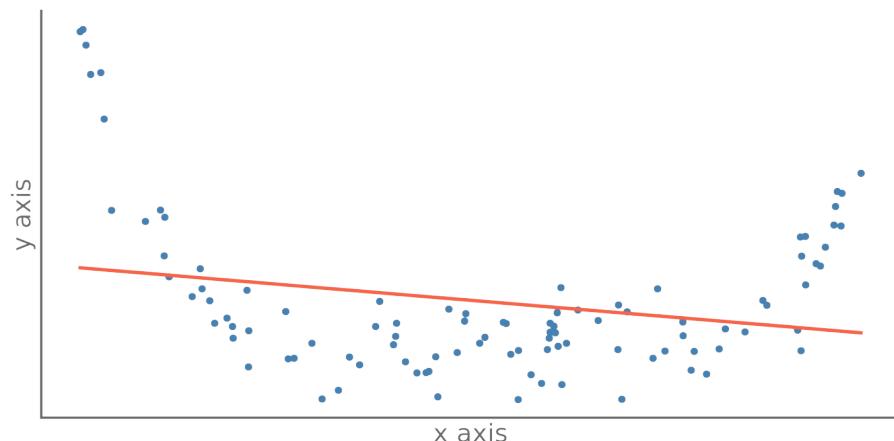
Overfitting

PRÉPARATION DES DONNÉES

- Deuxième idée : séparer les données en deux parties (train/test) et vérifier régulièrement avec le jeu de test

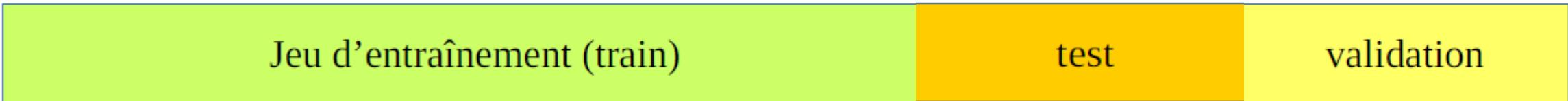


- **Pratique courante**, mais qui a quelques risques
- Problème : Aucune garantie que l'algorithme fonctionnera bien sur de nouvelles données
 - Underfitting : modèle trop simple pour expliquer la variance



PRÉPARATION DES DONNÉES

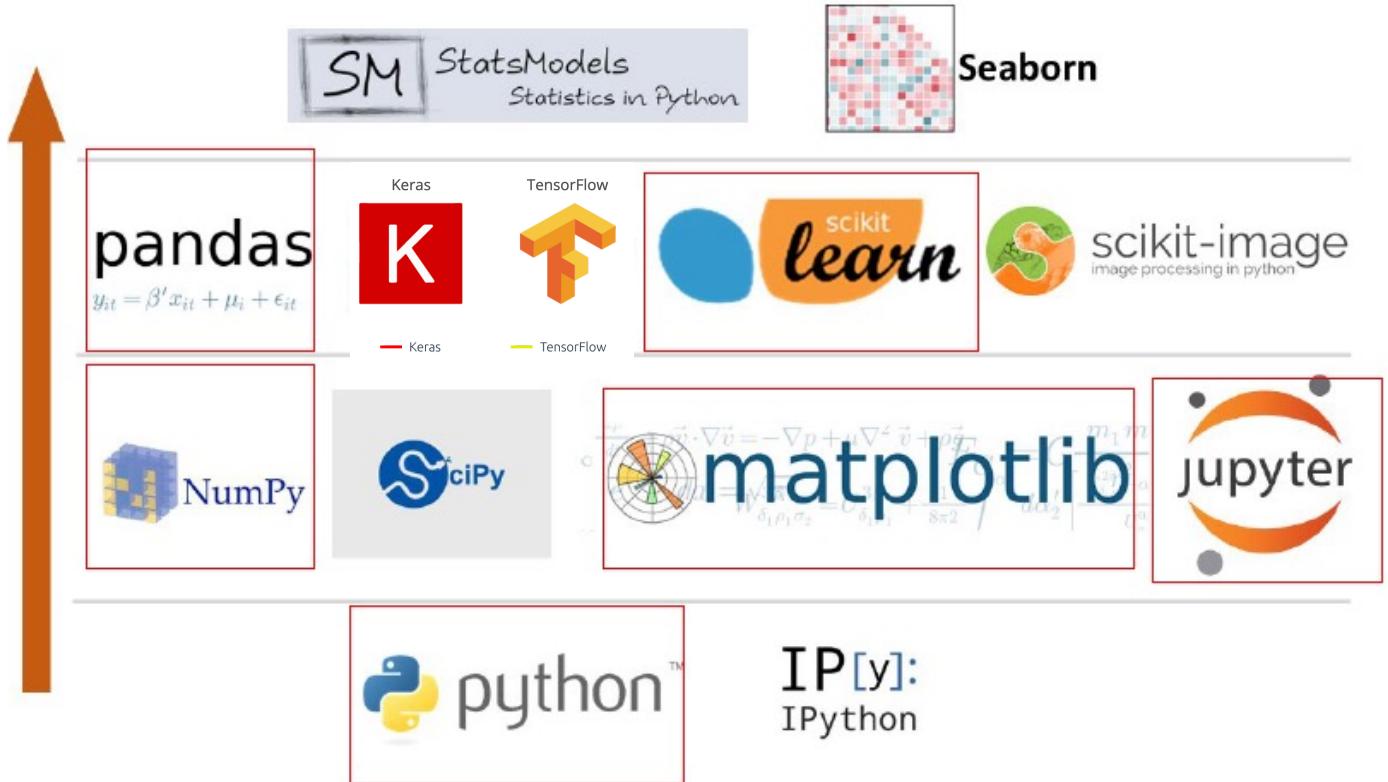
- Troisième idée :
 1. entraîner sur le jeu d'entraînement,
 2. choisir les hyper-paramètres qui fonctionnent le mieux sur un jeu de test,
 3. puis une fois le modèle réglé, l'évaluer sur un jeu de validation



Premiers pas

TOUR DE TABLE

- Pour démarrer notre exploration du machine learning, nous avons besoin de quelques outils
 - Notebooks Jupyter
 - Les bibliothèques Pandas, Scikit-Learn et Tensorflow



NOTEBOOK JUPYTER

- C'est quoi un Notebook Jupyter ?
 - Document « actif » contenant des **blocs de texte** (en format « *markdown* ») et des **blocs de code Python**

The screenshot shows a Jupyter Notebook interface with the following components:

- Bloc de texte Texte libre et formaté**: A green box pointing to the first text cell.
- Bloc de code Code Python et résultat d'exécution**: A green box pointing to the second code cell.
- Series**: A text cell containing a brief description of the Series class from Pandas.
- Entrée [4]:** A code cell containing Python code to create a Series object and print it.
- Résultat d'exécution**: The output of the code cell, showing the Series data:

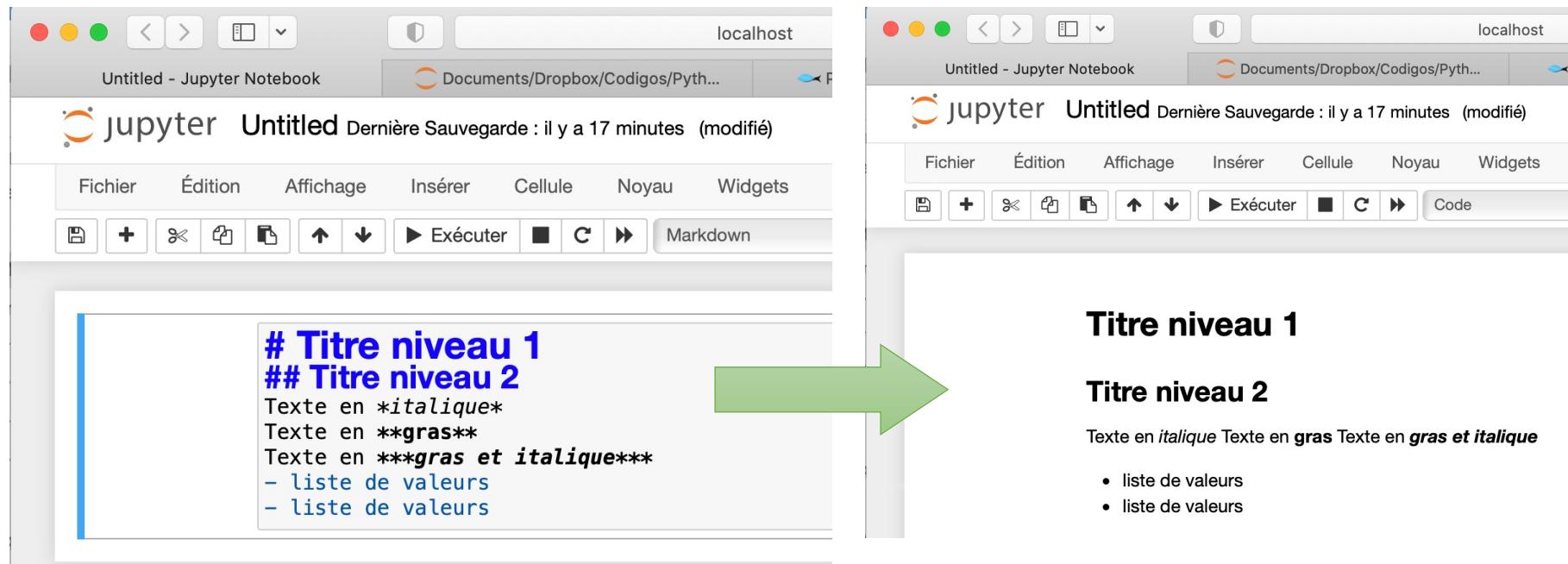
Suisse	8
France	70
USA	320
Chine	1200

- Entrée [5]:** A code cell containing Python code to import numpy and create a Series object from a NumPy array.
- Résultat d'exécution**: The output of the code cell, showing the Series data:

A	-0.96850466
B	-1.99478449
C	-1.6655539
D	-1.22318014
E	-0.35159315

NOTEBOOK

- Texte en « markdown »
 - Langage de marquage simple pour formater le texte



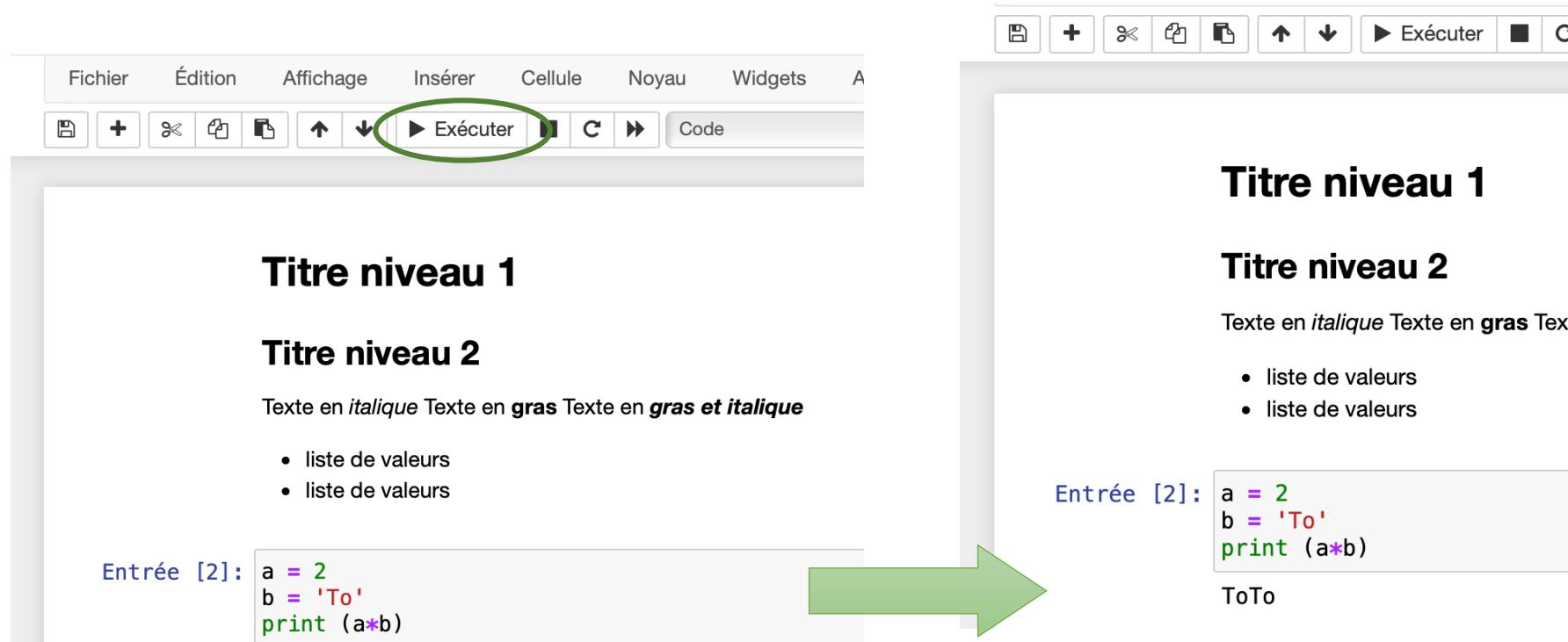
NOTEBOOK

■ Bloc de code Python

- Des petits **blocs de code** qu'on peut exécuter

- Equivalent au mode « itératif »

Attention : ça dépend de l'ordre dans laquelle on exécute les blocs,
pas nécessairement de l'ordre des blocs



The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with various icons for file operations like saving, opening, and executing cells. Below the toolbar is a menu bar with options like Fichier, Édition, Affichage, Insérer, Cellule, Noyau, Widgets, and A. The main area contains two code cells. The first cell, labeled "Entrée [2]:", contains the following Python code:

```
a = 2
b = 'To'
print (a*b)
```

A large green arrow points from this cell to the second cell. The second cell, also labeled "Entrée [2]:", contains the output of the previous command:

```
ToTo
```

The notebook interface also includes sections for "Titre niveau 1" and "Titre niveau 2" with some descriptive text and bullet points.

Google Collab



Accessible avec un simple compte Google

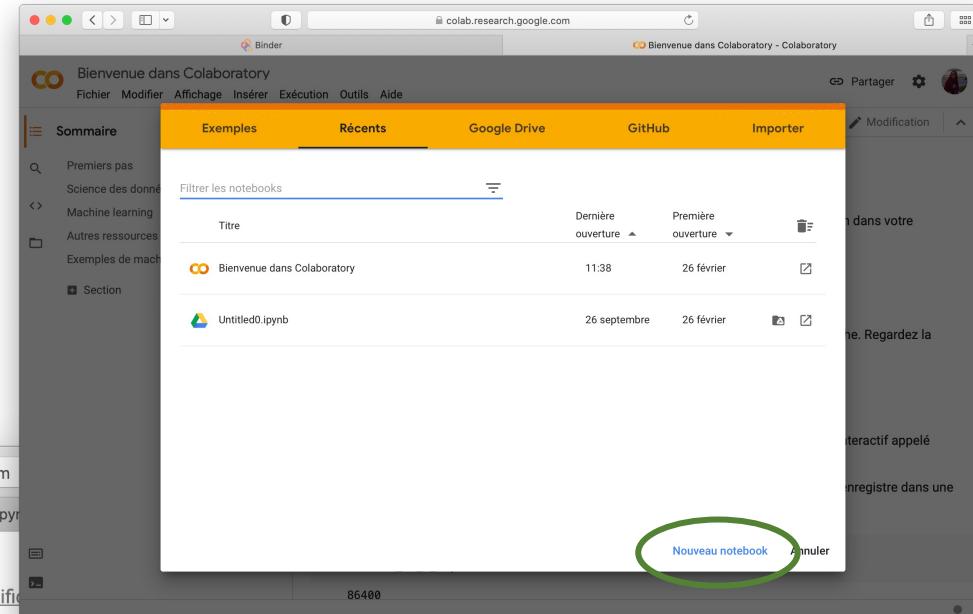
<https://colab.research.google.com>

The screenshot shows the Google Colab interface. At the top, there are tabs for 'Anaconda Navigator', 'Running the Noteb...', 'Installing Anacond...', and 'Untitled.ipynb'. The main window displays a notebook titled 'Untitled0.ipynb'. The sidebar on the left shows a tree view with a single node labeled 'Introduction'. The main content area contains the following text and code cell:

```
print('hello')
print(2+2)
```

Output:

```
hello
4
```



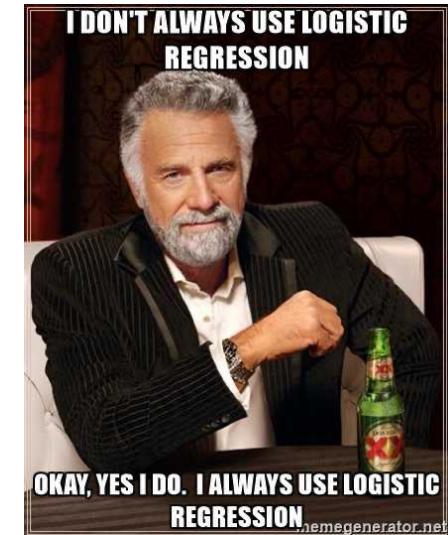
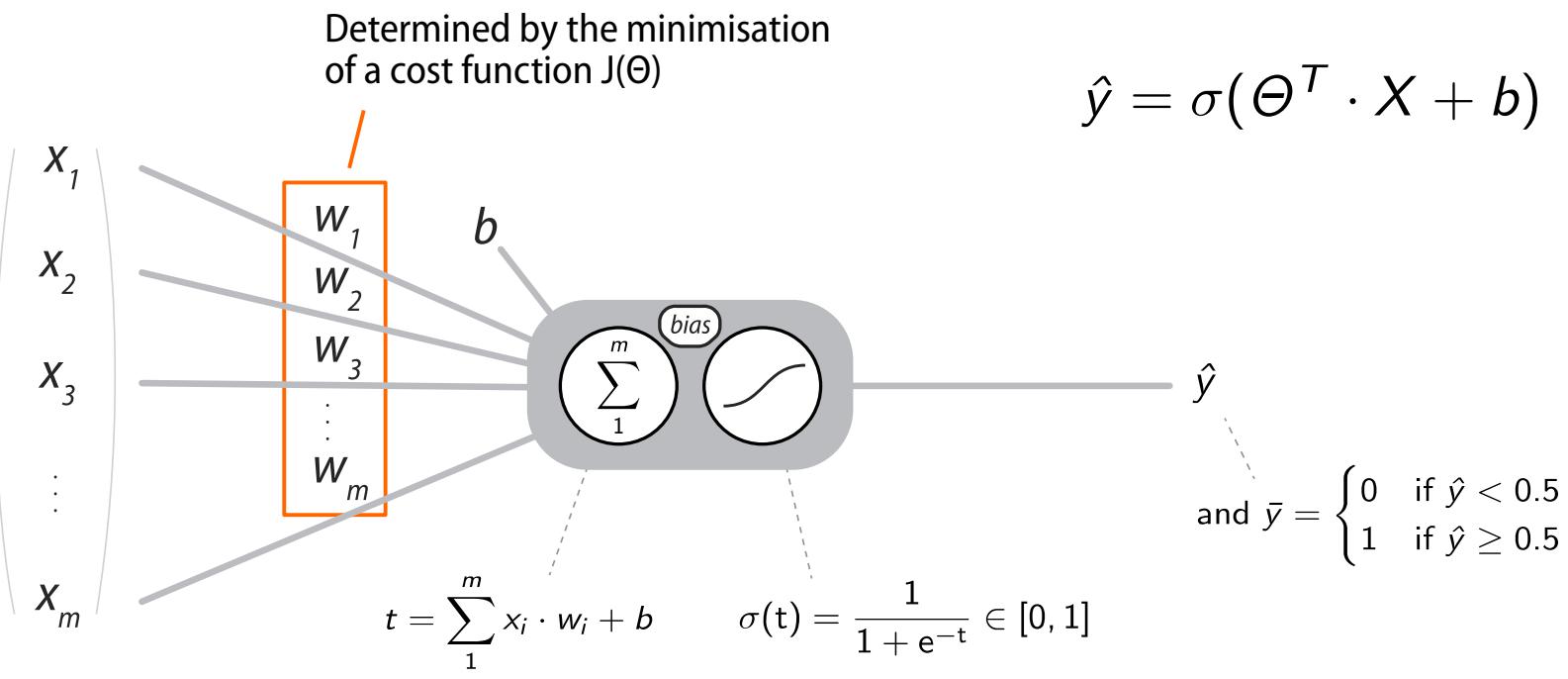
- On peut créer ses Notebooks et les enregistrer sur son Google Drive
- Partage d'un Notebook avec d'autres personnes est aussi possible
- Surtout, possibilité d'utiliser des machines avec GPU

Partie 2 :



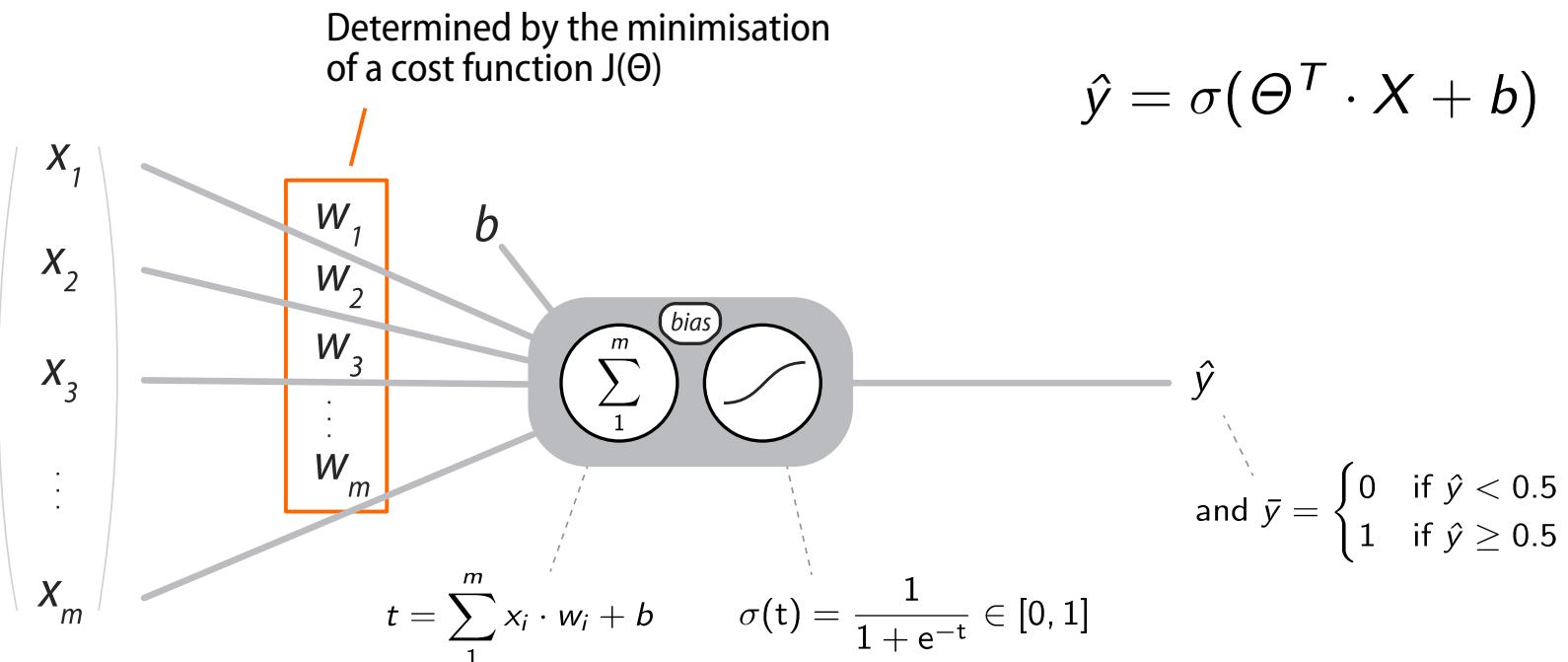
Ecole Doctorale SNI

SCHÉMA D'UNE RÉGRESSION LOGISTIQUE



Input	Bias / Weight	Activation function	Output
X	Θ	$\sigma(t)$	\hat{y}

SCHÉMA D'UNE RÉGRESSION LOGISTIQUE

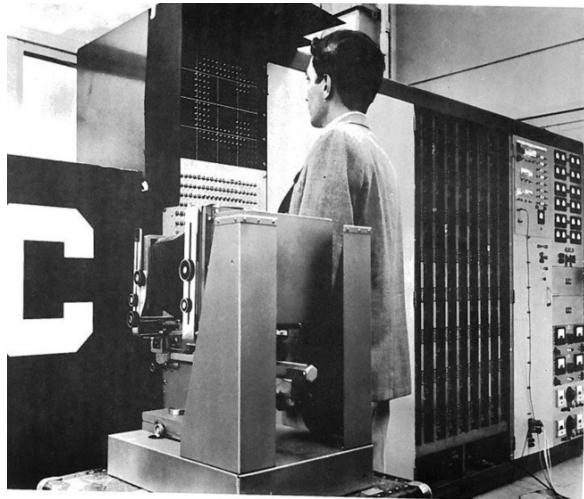
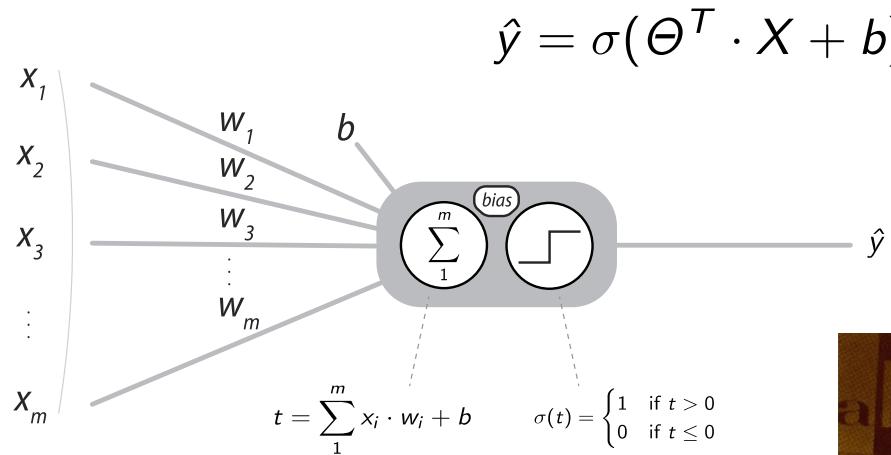


Input	Bias / Weight	Activation function	Output
X	Θ	$\sigma(t)$	\hat{y}

Surprise !!! Ceci est le schéma d'un neurone artificiel (perceptron). On a un réseau d'un neurone !!

L'HISTOIRE DES RÉSEAUX DE NEURONES

- Création du "Perceptron" : Frank Rosenblatt (1958)



THE PERCEPTRON

389

se sets of
ich are
tend to
t sets of
ve and/
stimuli
y facil-
itation of

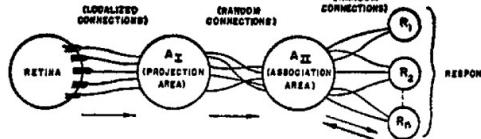


FIG. 1. Organization of a perceptron.

The cells in the projection area each receive a number of connections from

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

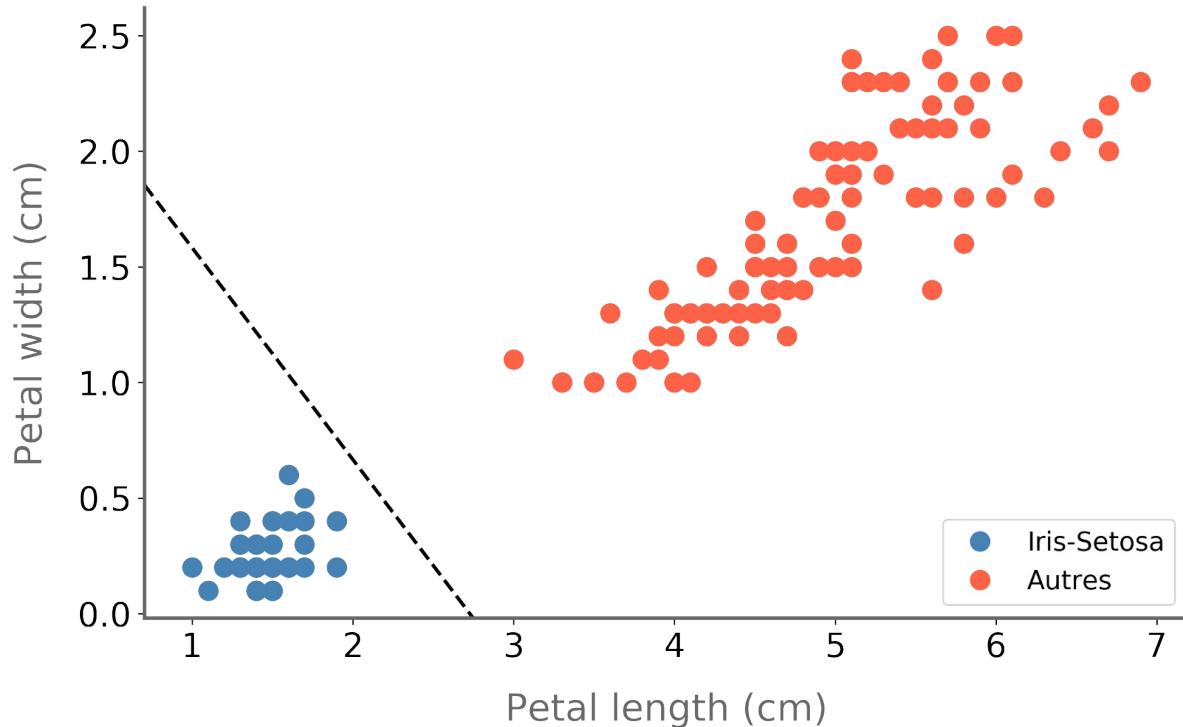
Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptrons will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

LE PERCEPTRON MARCHE (LINÉAIREMENT) !

Iris plants dataset

Dataset from : Fisher, R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936)



Length	Width	Iris Setosa (0/1)
x_1	x_2	y
1.4	1.4	1
1.6	1.6	1
1.4	1.4	1
1.5	1.5	1
1.4	1.4	1
4.7	4.7	0
4.5	4.5	0
4.9	4.9	0
4.0	4.0	0
4.6	4.6	0
(...)		

LES RÉSEAUX DE NEURONES PROFONDES

- Un perceptron peut apprendre tant que les résultats sont "linéairement séparables" :
- Ex ET logique

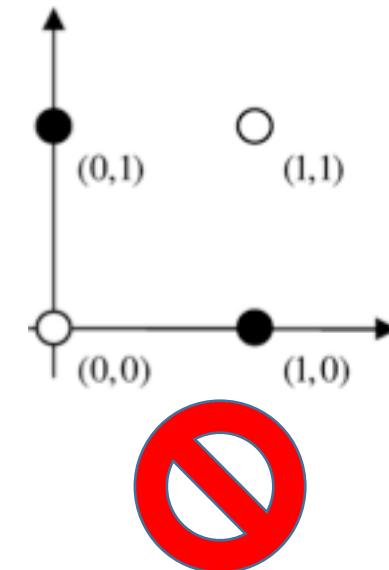
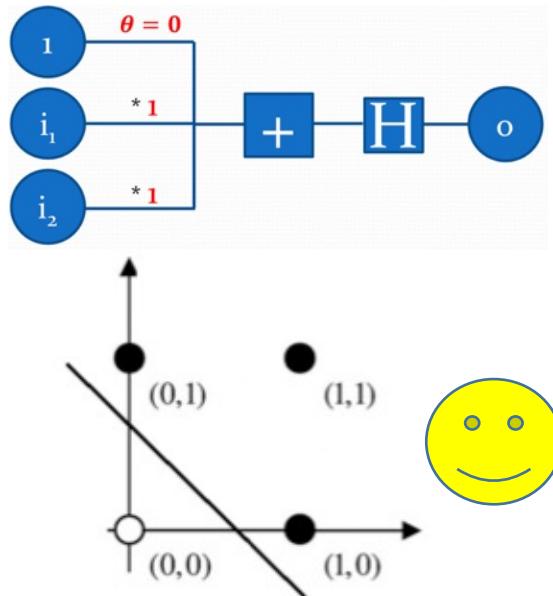
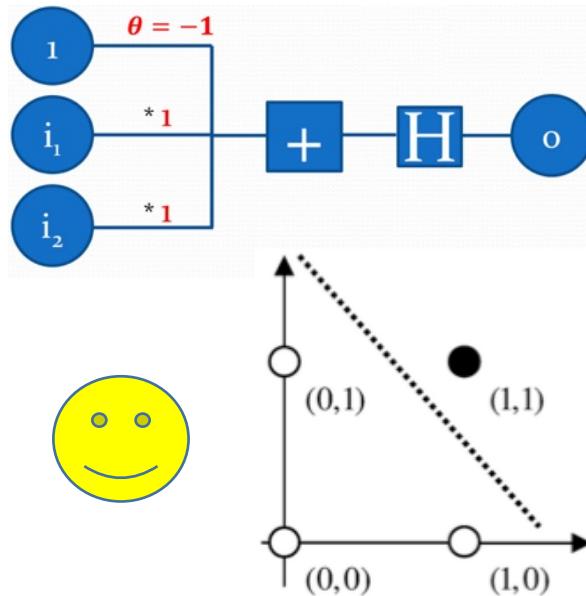
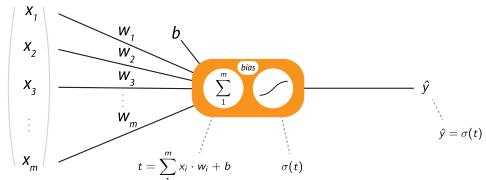
i1	i2	o
0	0	0
0	1	0
1	0	0
1	1	1

Ex OU logique

i1	i2	o
0	0	0
0	1	1
1	0	1
1	1	1

Ex XOU logique

Ex XOU logique



EXERCICE 1

- Pour ce premier exercice, vous allez utiliser un neurone simple (perceptron) pour faire la classification sur un dataset historique : IRIS
 - IRIS est un dataset créé originellement en 1936
 - Il regroupe des mesures (largeur, longueur) des pétales et sépales de trois espèces de la fleur Iris
 - Iris versicolor
 - Iris virginica
 - Iris setosa
 - Le perceptron sera utilisé pour séparer les plantes Iris setosa des autres
 - Nous allons utiliser la bibliothèque Scikit Learn
 - Accéder ce site : <https://t.ly/jek-x>



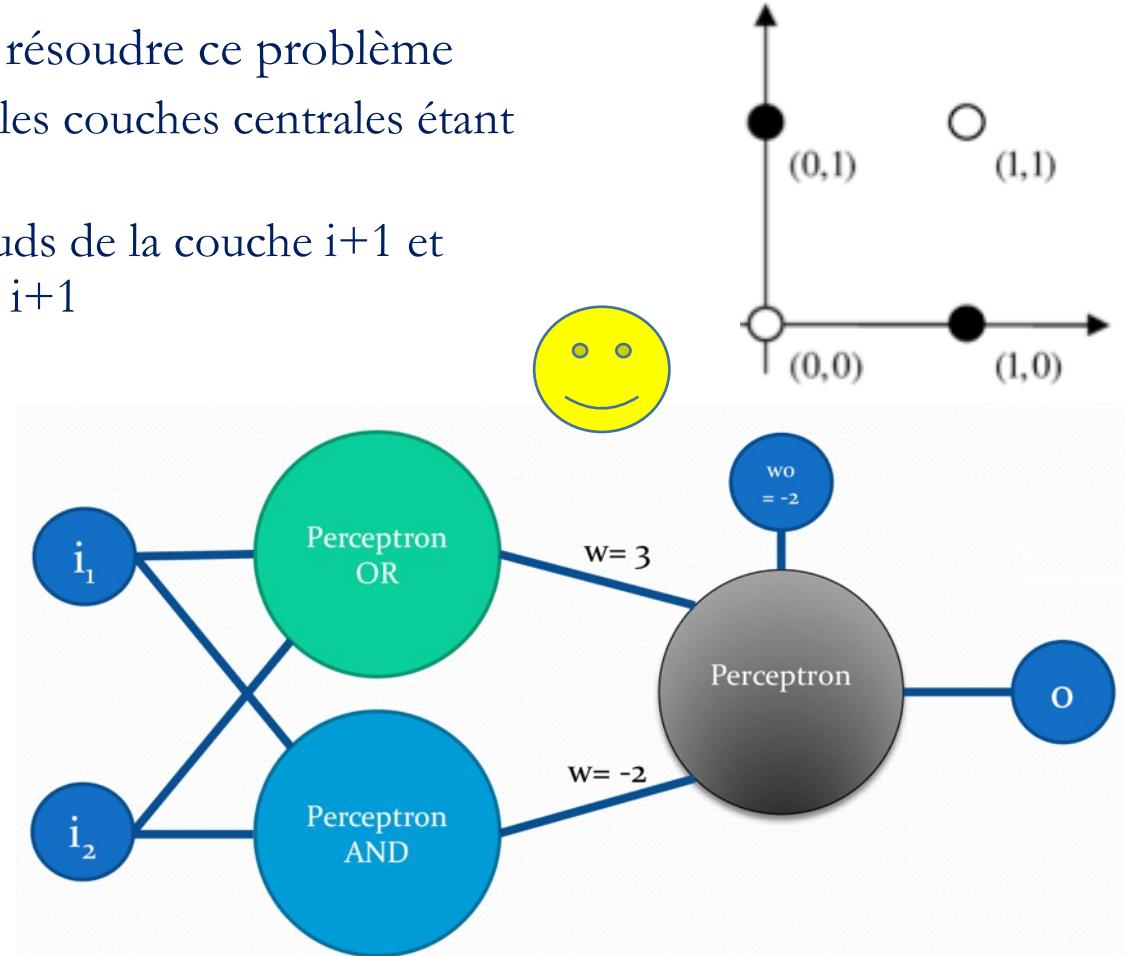
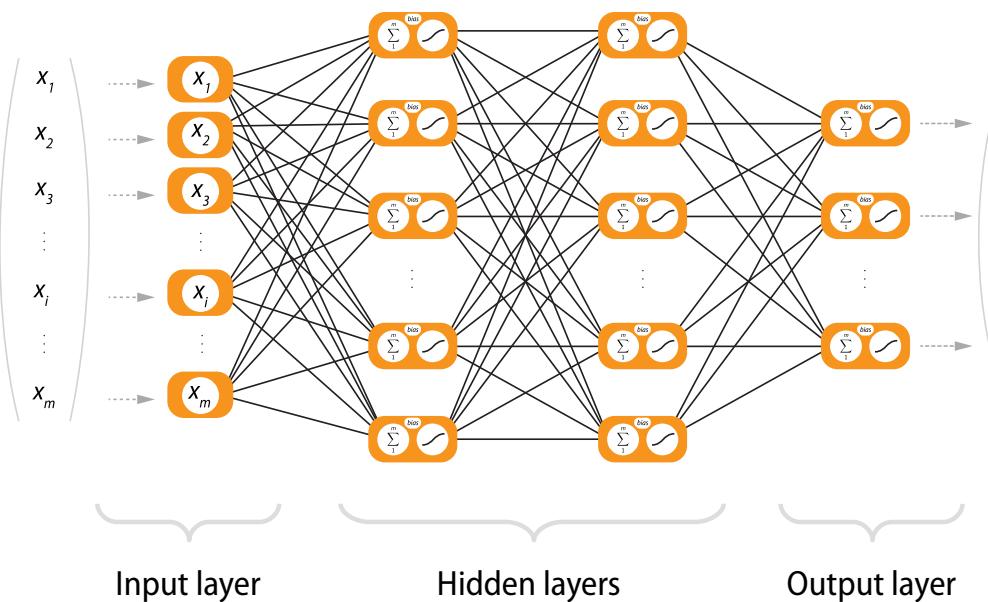
You are not a data scientist..



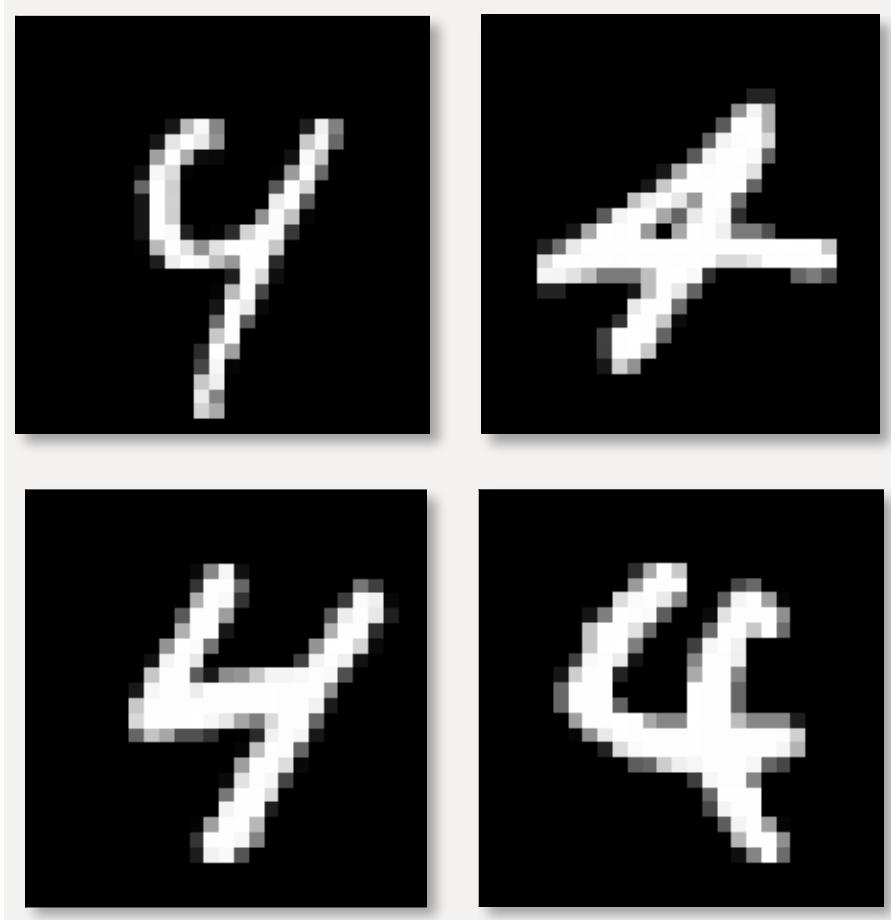
if you don't know this flower

RÉSEAUX DE NEURONES

- Le concept de perceptron multicouche (MLP) permet de résoudre ce problème
 - On utilise plusieurs perceptrons en couche successives, les couches centrales étant dites cachées
 - Chaque nœud de la couche i est connecté à tous les nœuds de la couche $i+1$ et l'information circule toujours de la couche i à la couche $i+1$

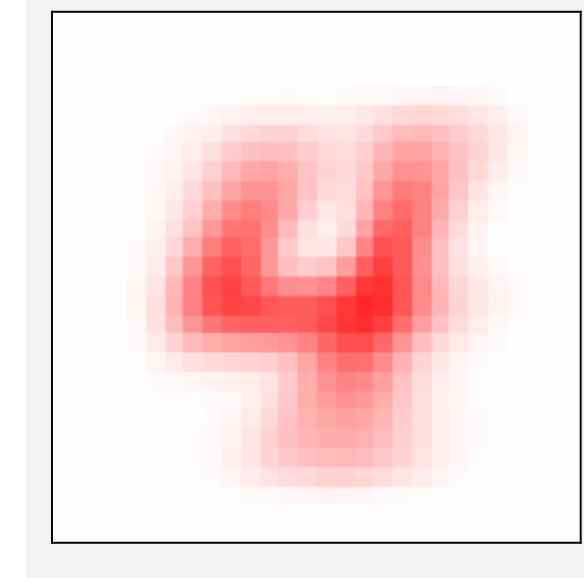


MNIST : PLUSIEURS FAÇONS D'ÉCRIRE 4

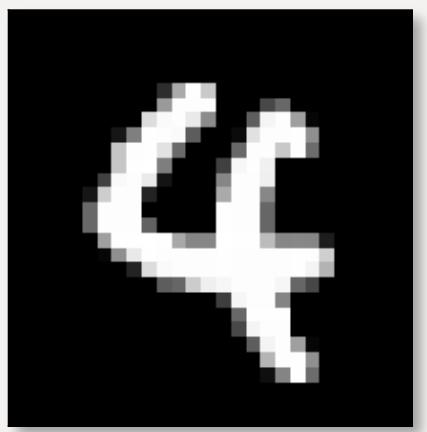
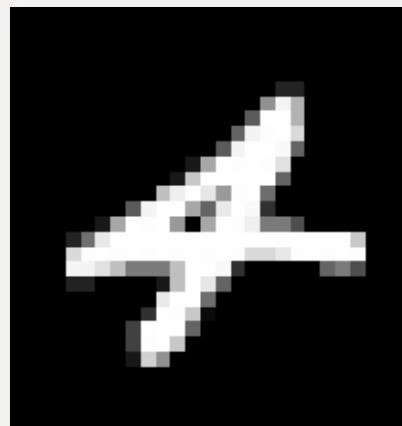
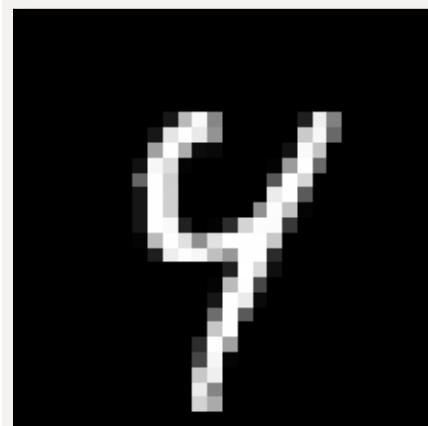


Filtre unique

- pas très précis, adapté juste au cas moyen

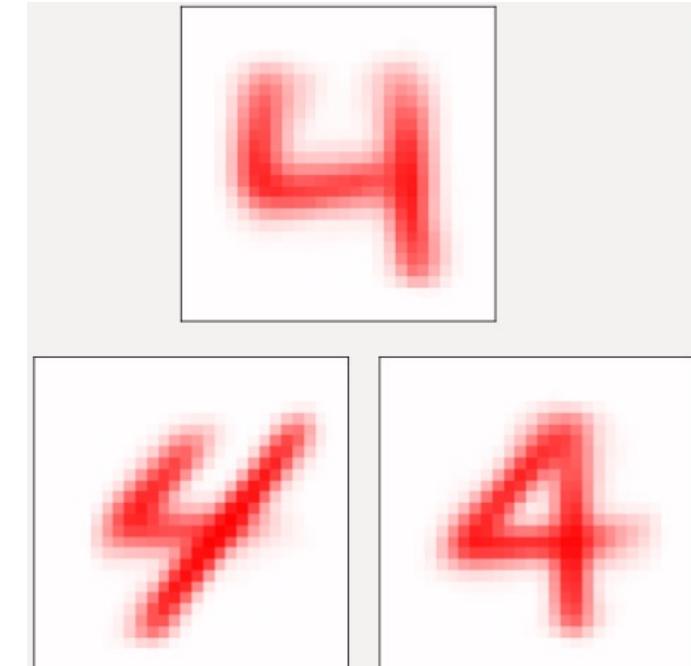


PLUSIEURS FAÇONS D'ÉCRIRE 4

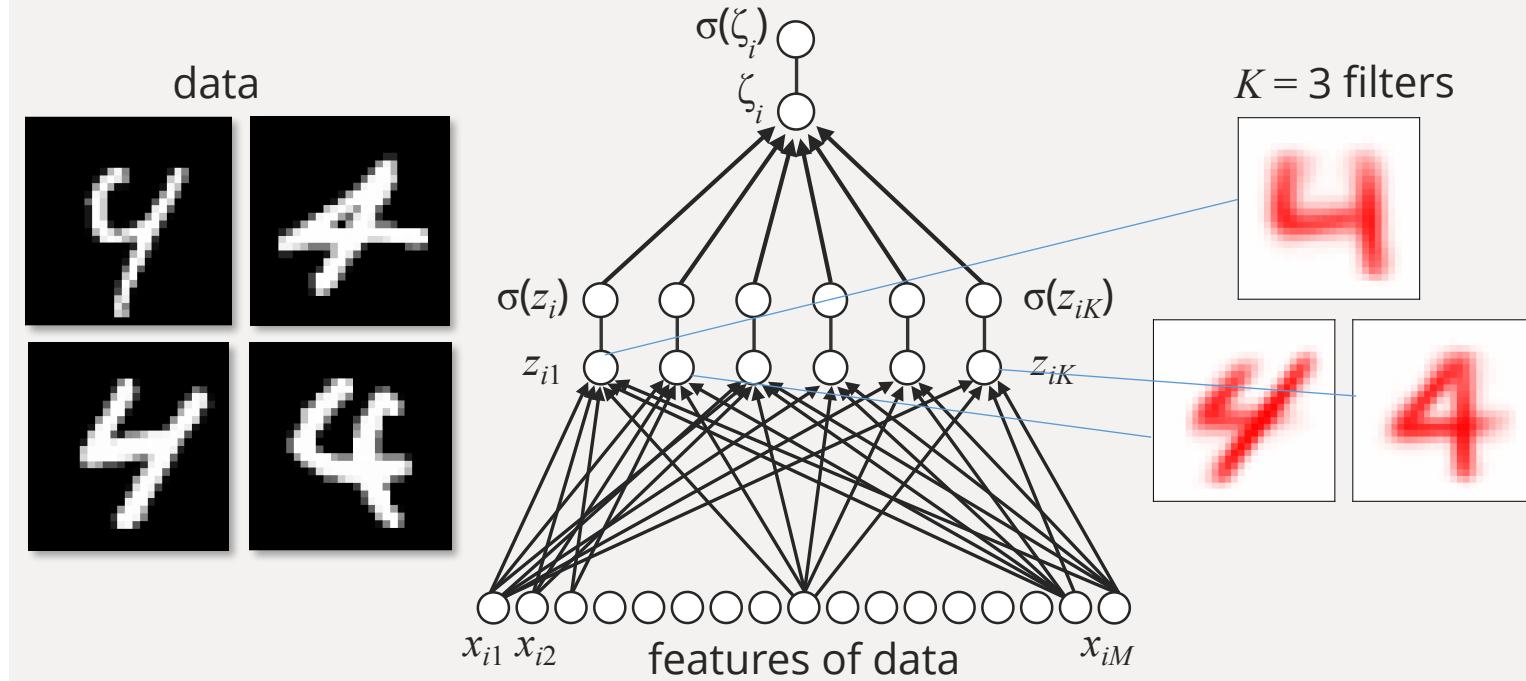


Et si on avait plusieurs filtres ?

- Chacun est spécialisé sur une forme d'écrire 4
- Une correspondance "forte" avec un filtre est un bon signe



RÉSEAU DENSE DE NEURONES (DNN)



- Tous les pixels de l'image sont associés à chacun des neurones
- Les neurones sont initialisés avec des poids aléatoires (filtres)
- L'entraînement permet à chacun des neurones d'observer l'image avec un filtre différent
- C'est la somme de ces points de vue qui donne une sortie

PREMIER PROBLÈME DES DNN - SCALABILITÉ

For a fully connected layer of (only)
1000 neurons, we would need to



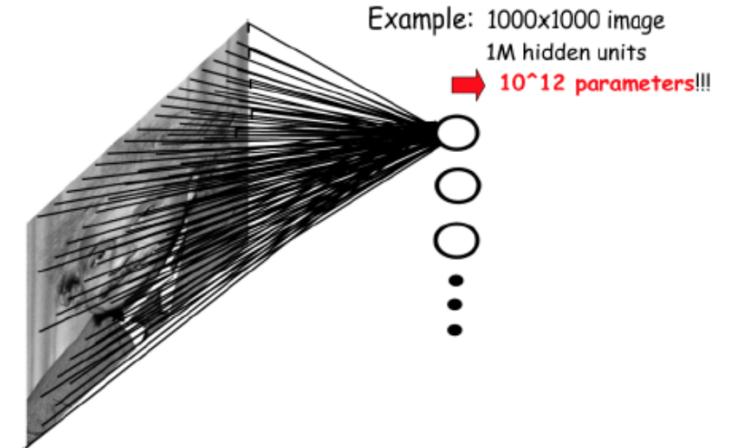
0.0008 M pixels
28x28, 8 bits

785.000 params



24 M pixels
(r,v,b) 3x8 bits

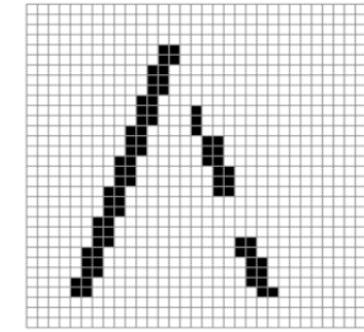
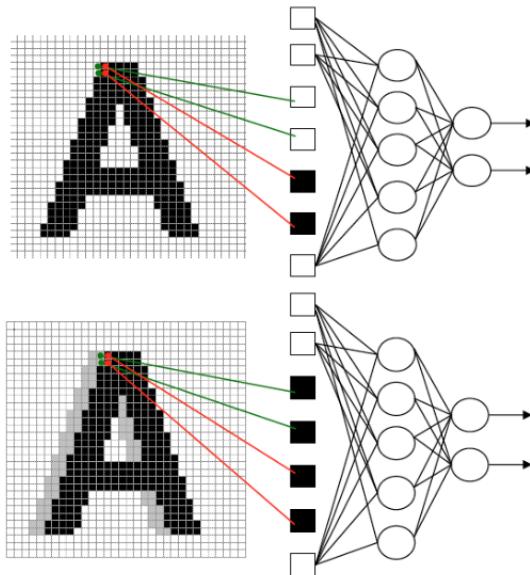
72. 10E9 params...



DEUXIÈME PROBLÈME

- Invariance et Stabilité

- **Invariance & stability**
- Expectations:
 - Small deformation \Rightarrow similar representations
 - Large deformation \Rightarrow dissimilar
- Translation invariance difficult with Fully Connected Networks \sim local scale, rotation, deformations, etc



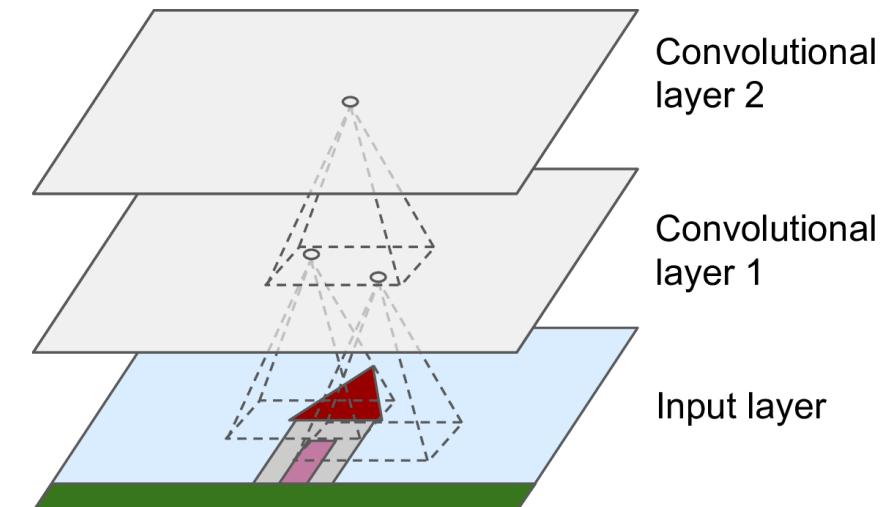
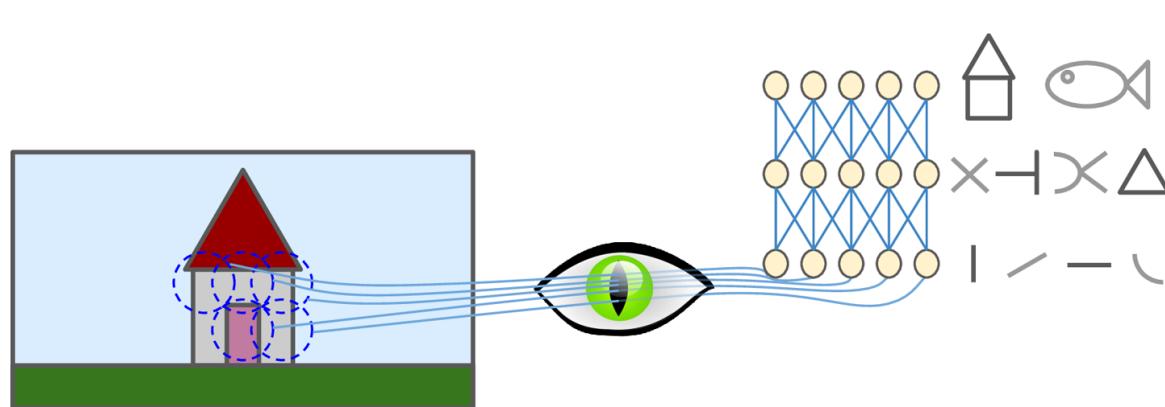
154 input change
from 2 shift left
77 : black to white
77 : white to black

@LeCun



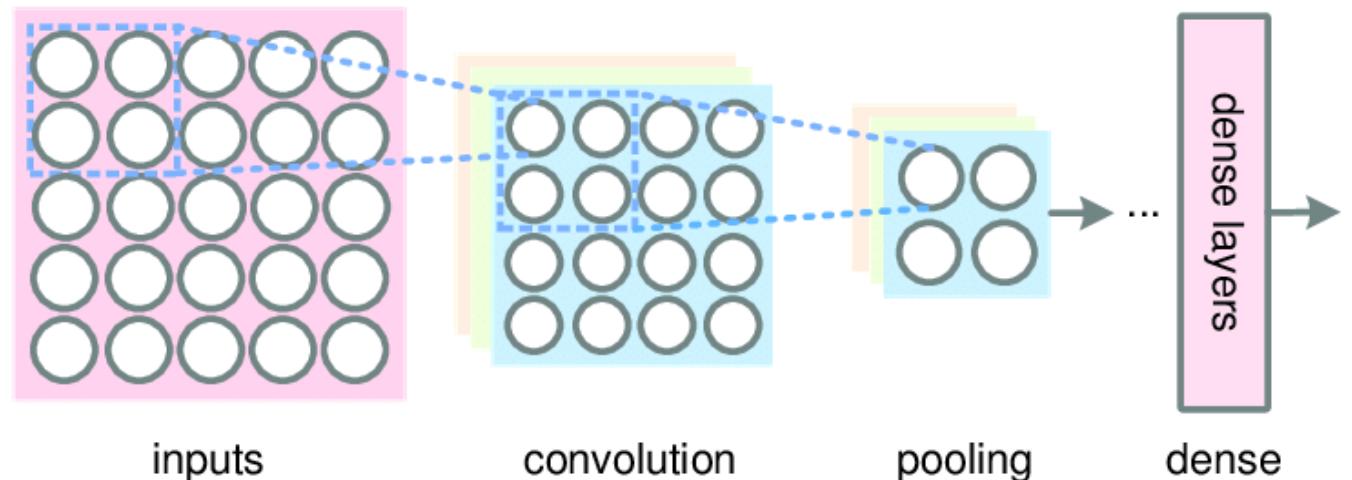
L'ARCHITECTURE DU CORTEX VISUEL

- Des travaux faits par des biologistes/neurologistes dans les années 50 ont montré que plusieurs neurones ont une champ d'activation limité
 - Ne réagissent qu'à des certaines zones du champ visuel
 - La réceptivité de ces neurones peut se superposer
 - Certains neurones ne répondent qu'à certains motifs : ex : lignes horizontales
 - D'autres étaient plus "avancés", combinant de signaux des neurones plus basiques



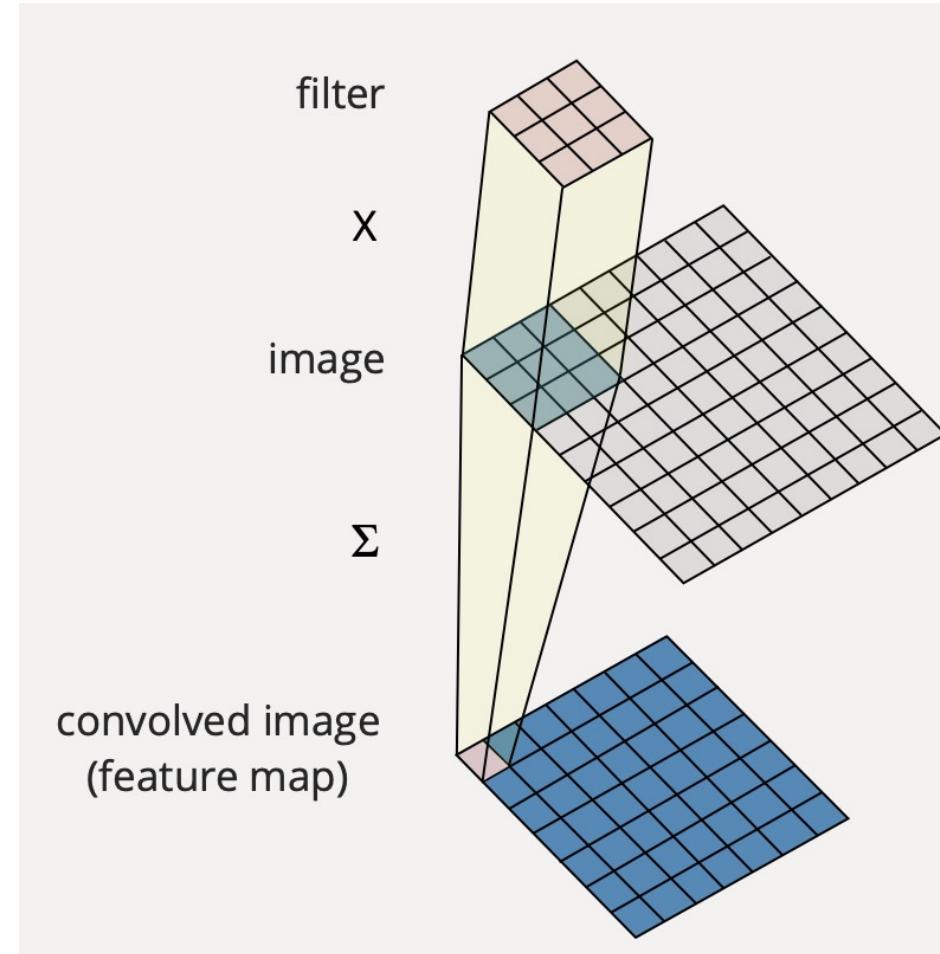
LES RÉSEAUX CONVOLUTIFS (CNN)

- Introduits par Yan Lecun en 1998 dans l'architecture LeNet-5
 - Utilisé encore pour la reconnaissance de l'écriture manuelle
- Utilisation d'éléments déjà connus :
 - Réseaux totalement connectés
 - Fonctions d'activation (sigmoid, notamment)
- Introduction de deux nouveaux éléments :
 - Les couches convolutionnelles
 - Les couches de pooling

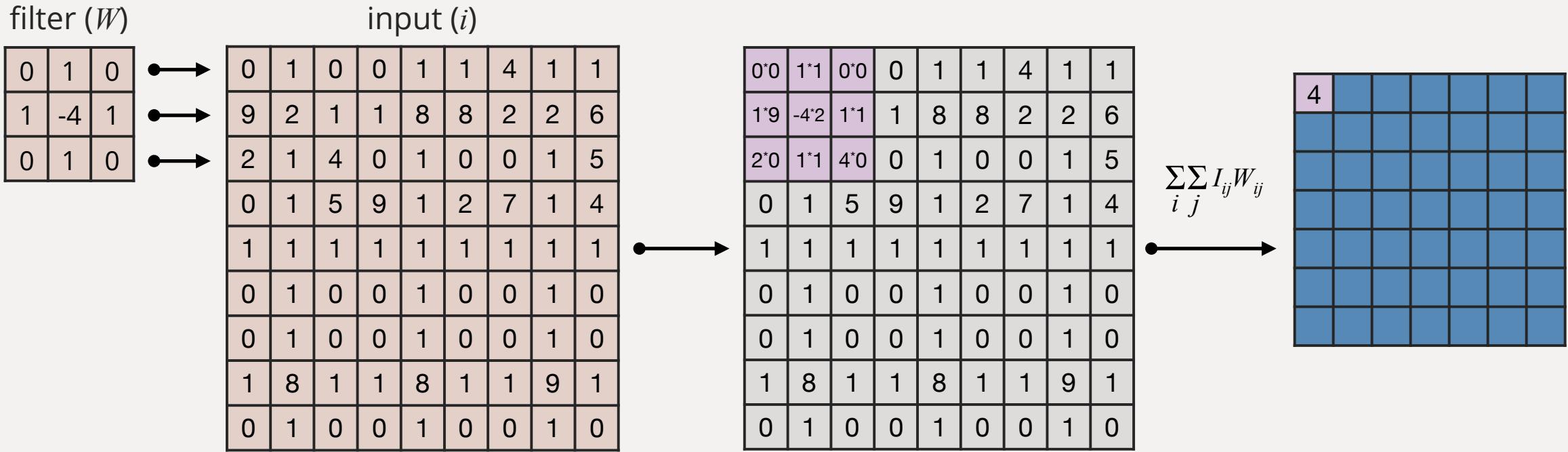


LE CAS DE LA CONVOLUTION 2D

- Le même principe s'applique dans le cas 2D
- Un filtre peut avoir différentes tailles
 - Ex : 5×5 , 3×3 , 4×6 ...



LE CAS DE LA CONVOLUTION 2D



LE CAS DE LA CONVOLUTION 2D

filter (W)

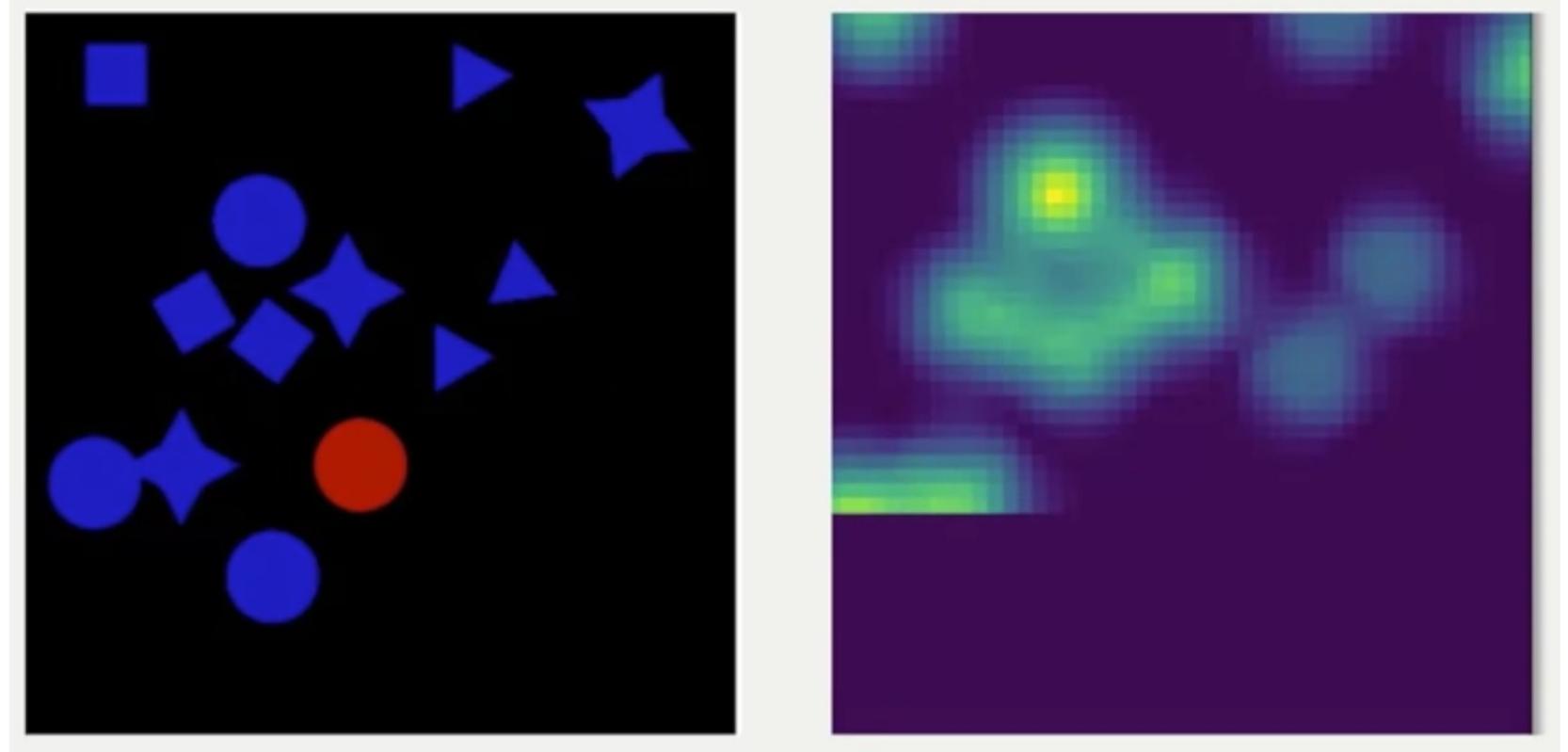
0	1	0
1	-4	1
0	1	0

input (i)



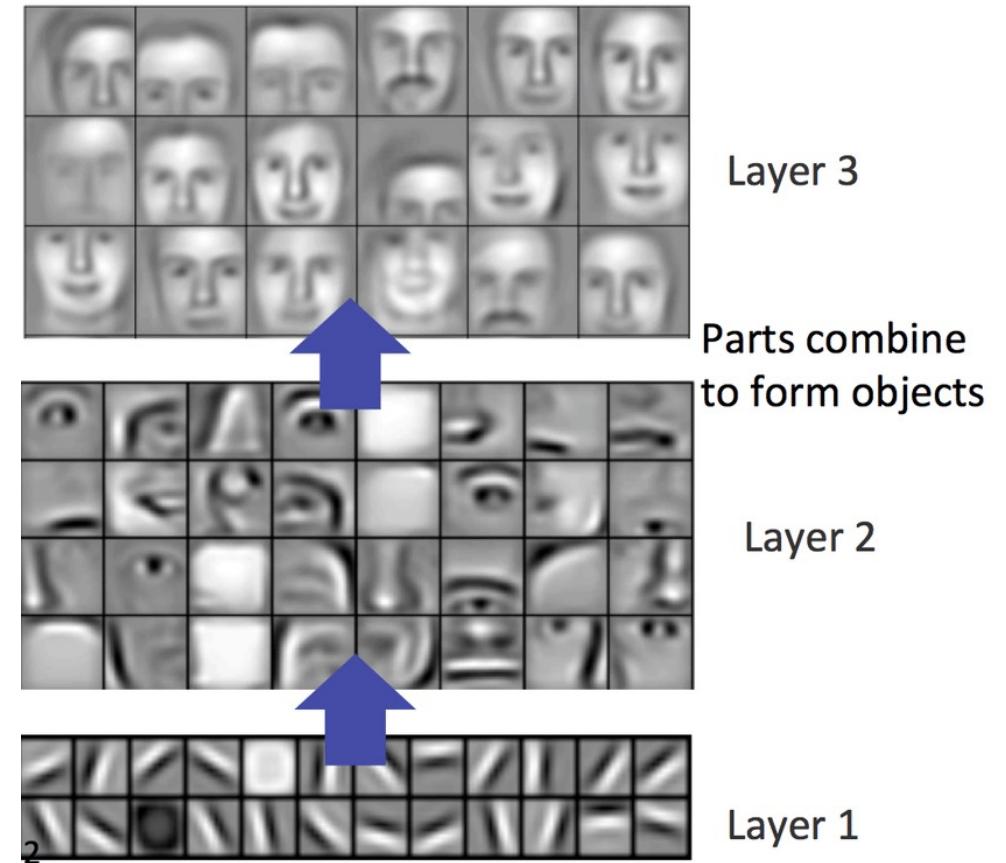
CHAQUE FILTRE RENFORCE LE POINTS "CHAUDS"

- Le fait d'appliquer des "filtres" petits (convolution) crée une espèce de détecteur de motifs
- Quand appliqués sur une image, les convolutions sont activées ou pas, créant une carte de points chauds



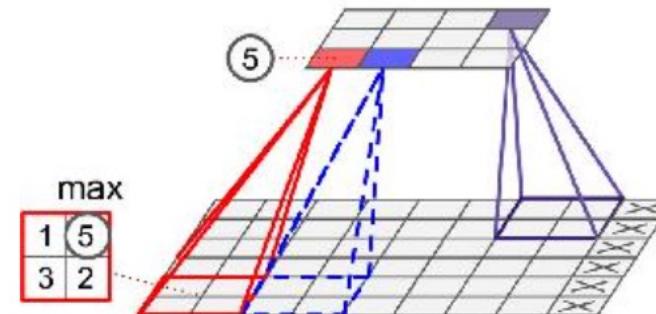
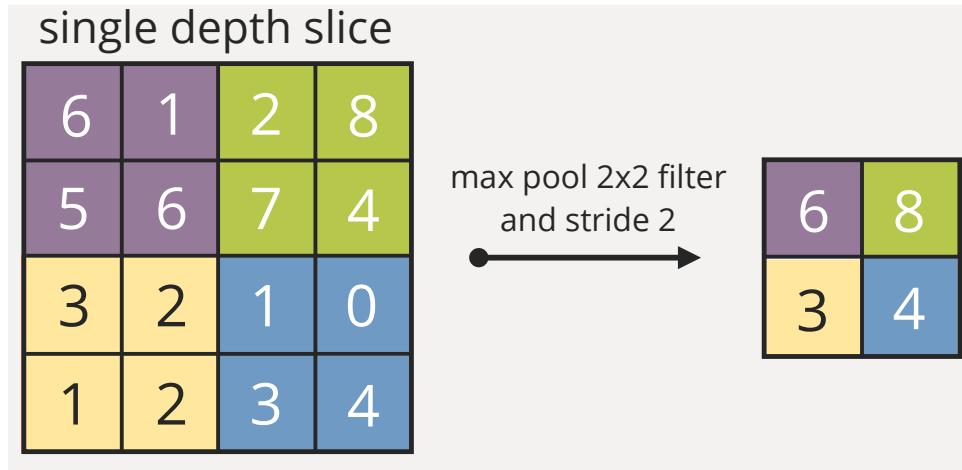
EST-CE SI SIMPLE ?

- Comme on a vu avant, il faut un nombre important de filtres pour représenter différentes features
- Les neurones pour des feature maps différents utilisent des poids différents
- Un DNN ne peut reconnaître le motif là où il l'a appris
- Un CNN apprend à identifier un motif, il est capable d'identifier ce motif partout dans l'image

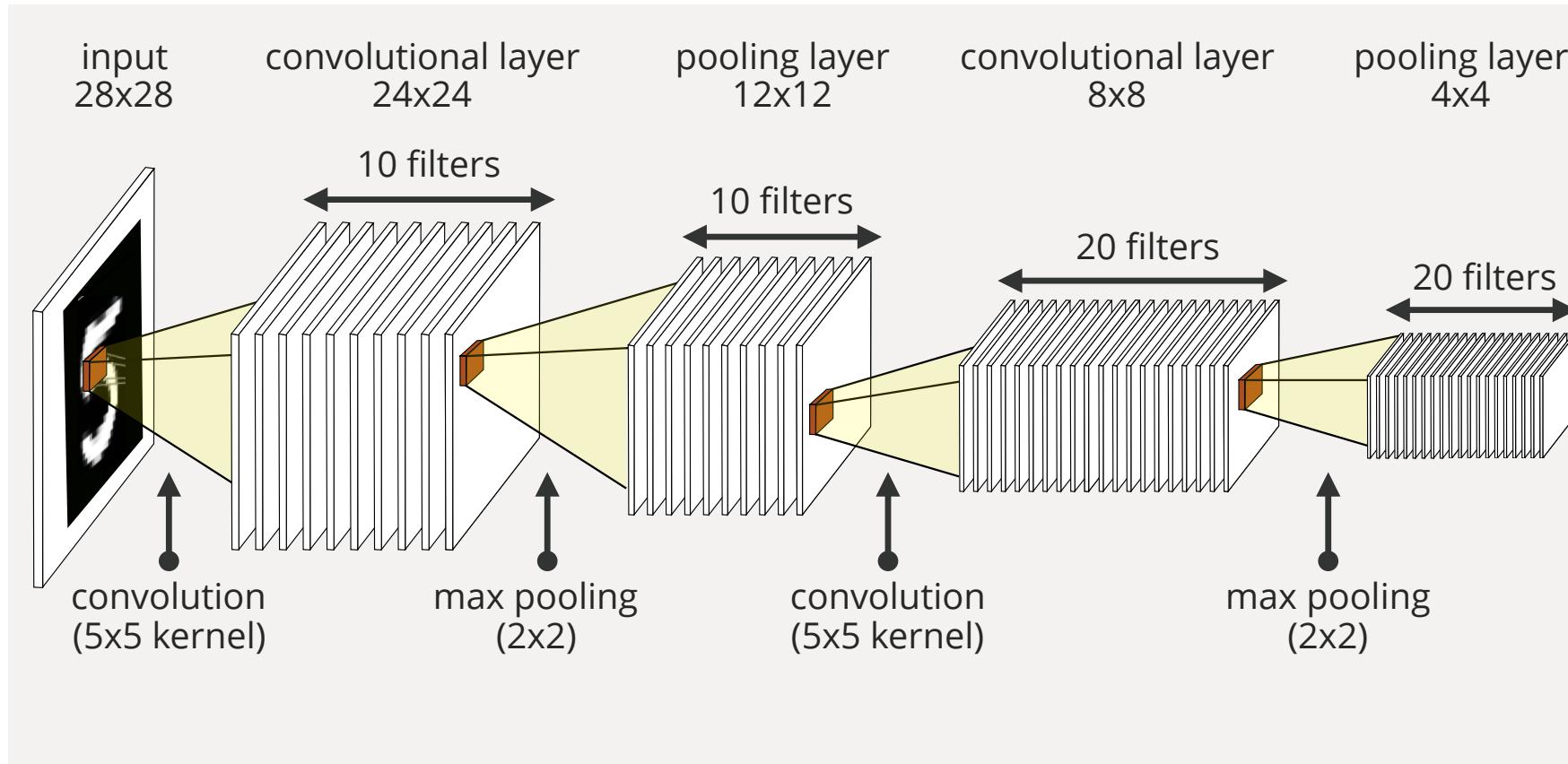


LA COUCHE POOLING

- Similaire à une convolution, sert à réduire la taille de la matrice (et donc la complexité du modèle)
- On utilise typiquement un filtre "max" (*maxpooling*)
 - Permet de réduire la taille des matrices tout en gardant les valeurs plus importantes
- On peut également utiliser la "moyenne"

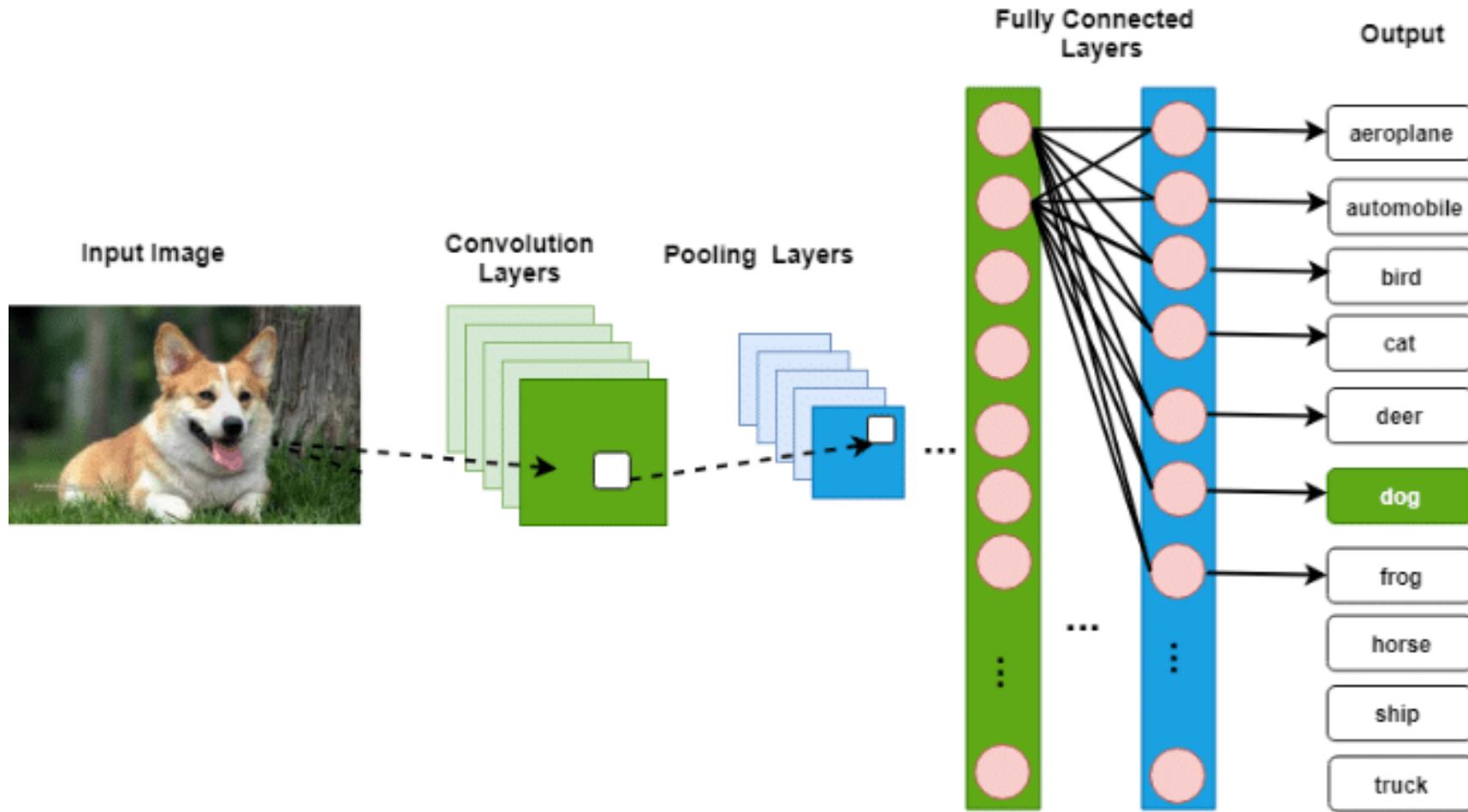


ET SI ON ENCHAÎNAIT PLUSIEURS TRANSFORMATIONS?

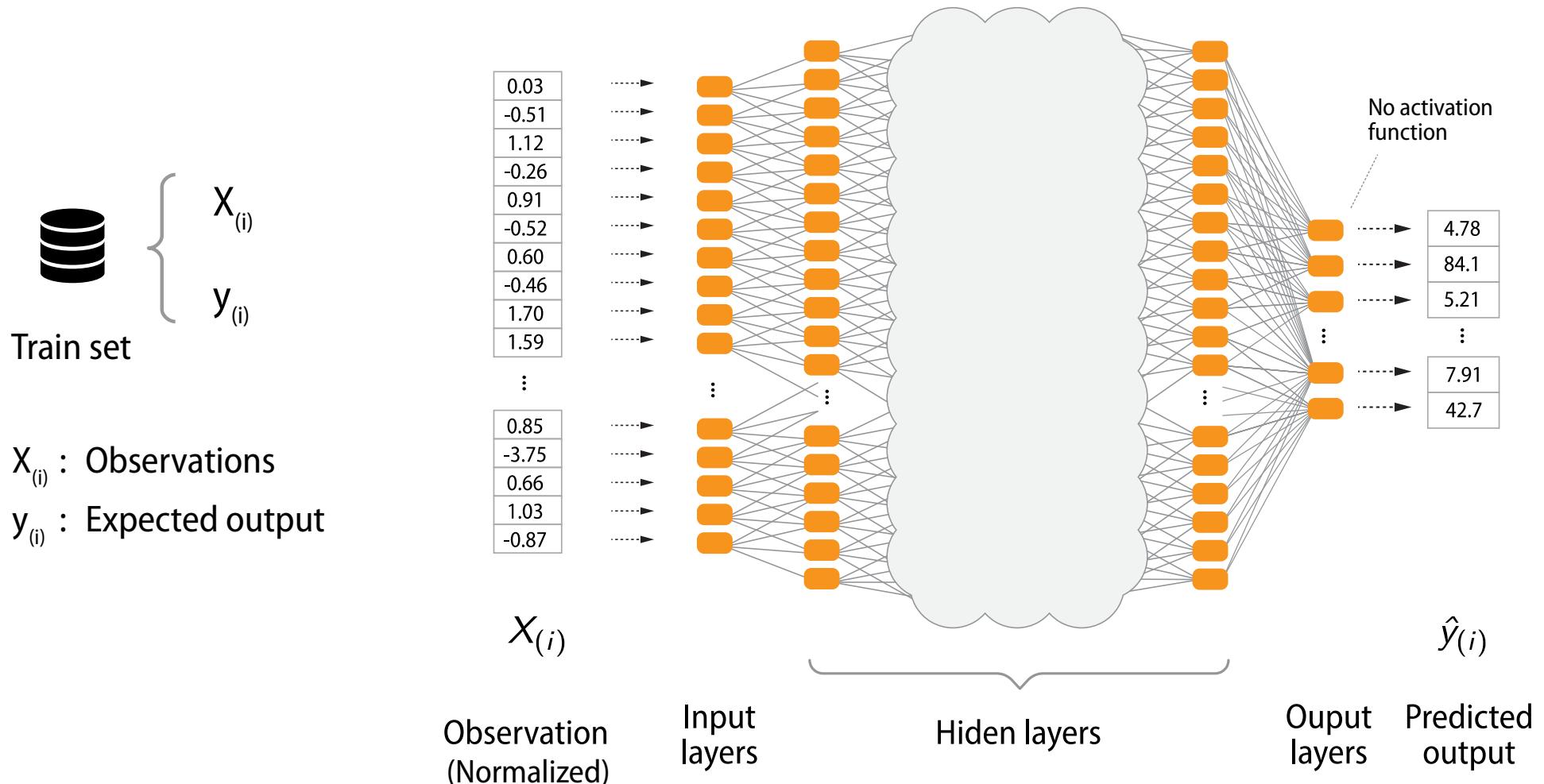


On arrive à un point où il devient impossible de faire d'autres convolutions (2x2)
Par contre, on a une pile de couches importante. Comment transformer ça pour la sortie ?

ON APLATIT TOUT ET HOP, UN COUP DE RÉSEAU DENSE POUR FAIRE LA CLASSIFICATION



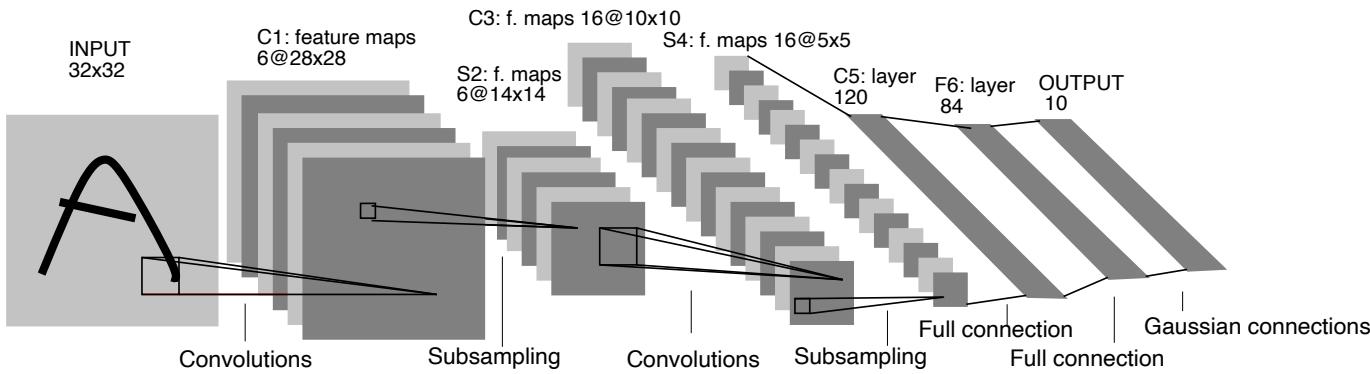
DES PERCEPTRONS AU DEEP LEARNING



Exemples de réseaux et utilisations

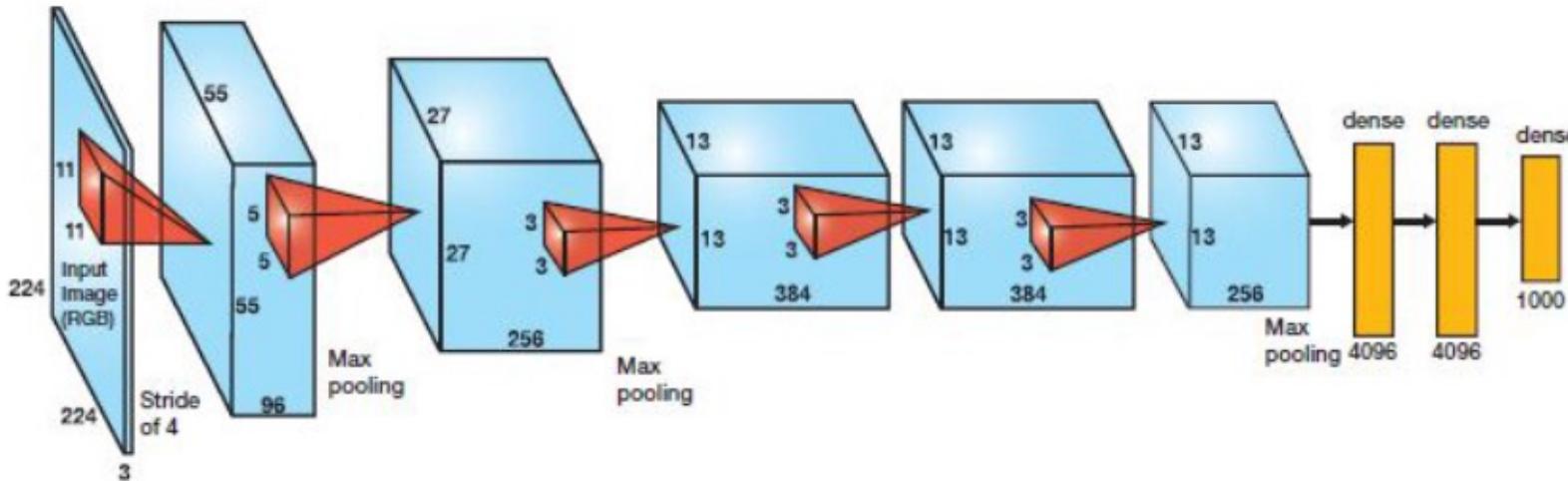
LENET, LE PIONNIER

- Modèle proposé par Yan Lecun en 1998 dans le but de reconnaître le l'écriture (MNIST)
- Peu de convolutions, peu de couches

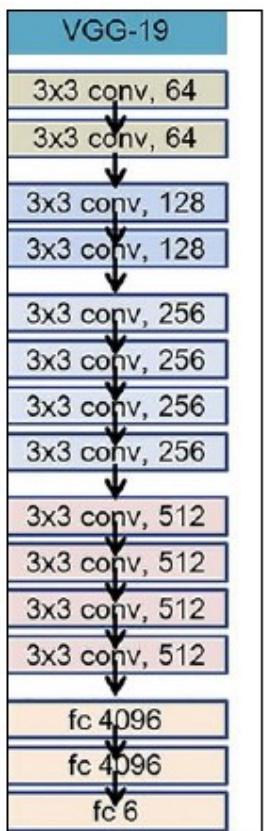


Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully connected	–	10	–	–	RBF
F6	Fully connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

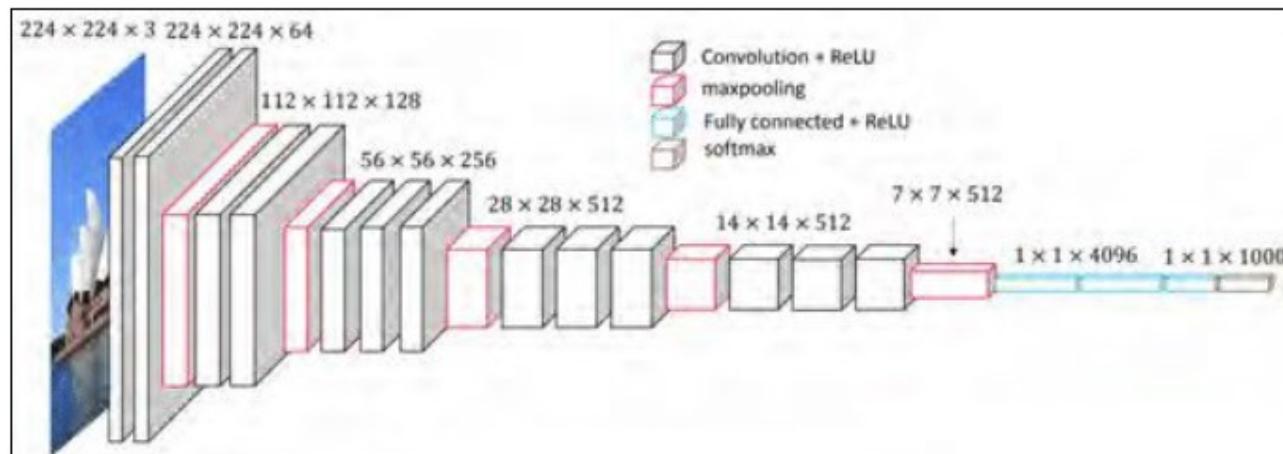
AlexNet



VGG



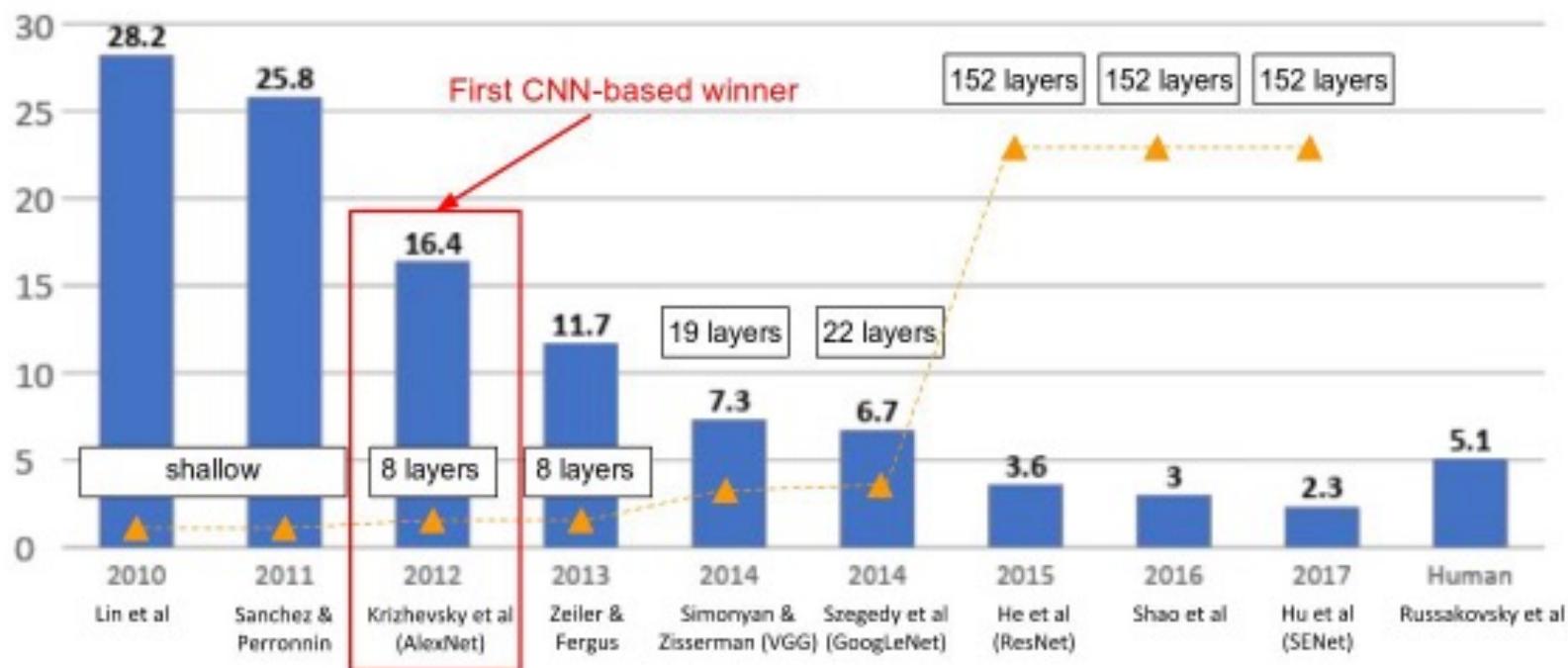
- Gagnant ImageNet 2014
- encore largement utilisé pour le transfert learning



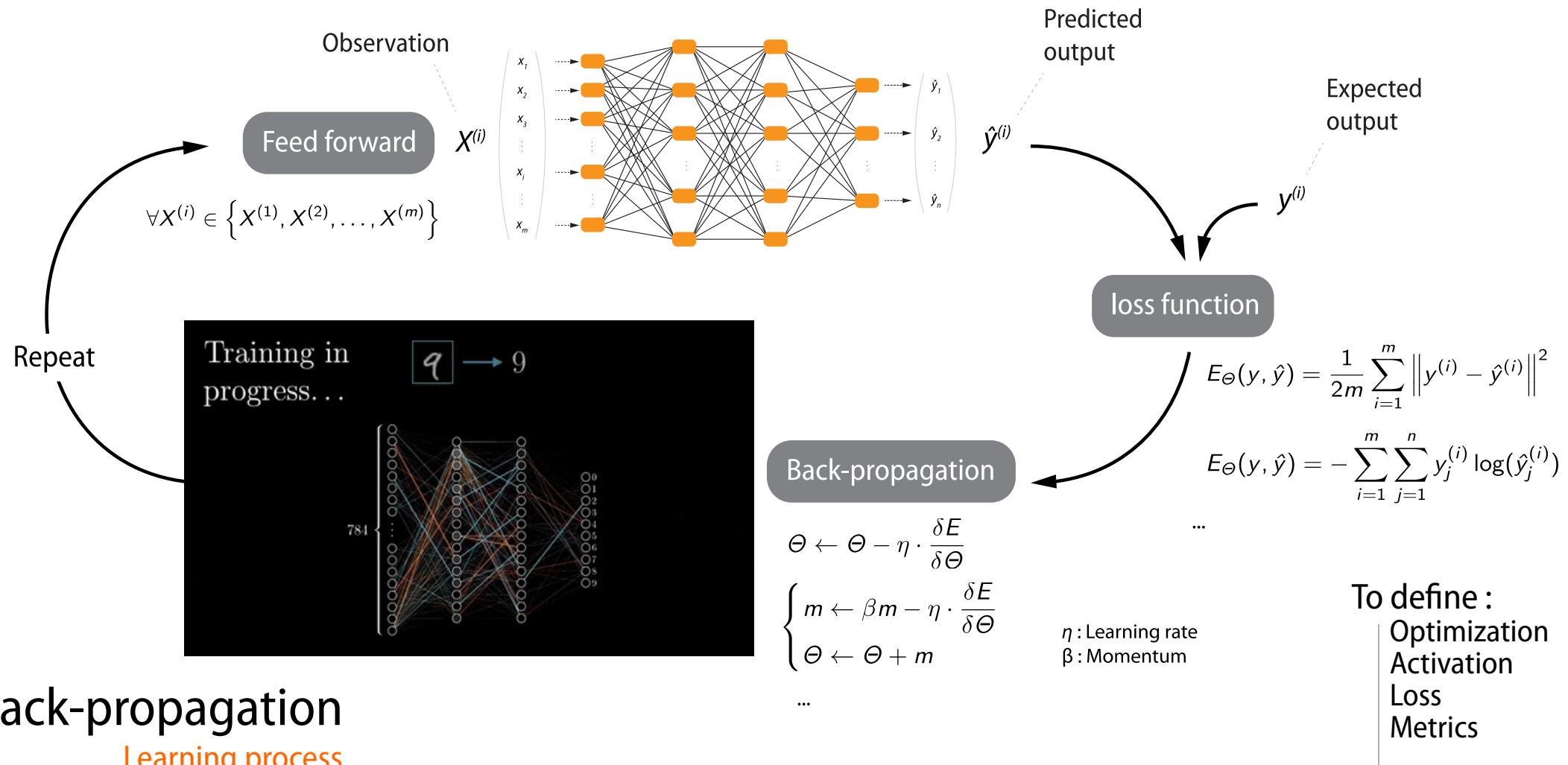
LA RÉVOLUTION DU DEEP LEARNING

- Depuis 2012, des modèles DL se sont imposés dans les challenges de classification d'images

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



ENTRAÎNEMENT PAR RETROPROPAGATION (1986)



Back-propagation
Learning process

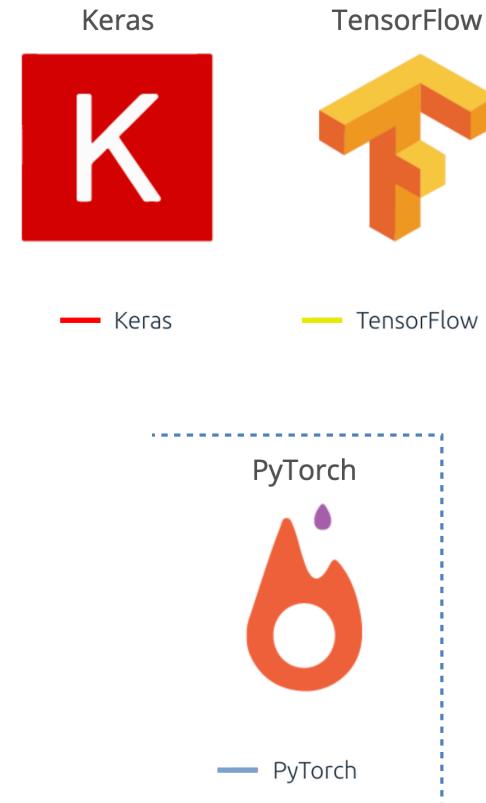
WE'RE GONNA NEED

A GPU

Comment programmer un
Deep Learning ?

DEUX GRANDS FRAMEWORKS

- Scikit Learn est une référence pour des algorithmes machine learning classiques
 - Régressions, arbres de décision, SVM
- Il n'a pas été développé pour explorer les réseaux de neurones
- En ce moment, deux frameworks dominent le "marché" du Deep Learning
- Tensorflow
 - Initialement développé par Google
 - Bien testé et disposant d'une base importante d'outils
 - Dispose d'une API "haut niveau" : Keras
- PyTorch
 - Initialement développé par Facebook/Meta
 - Plus récent, support croissant
 - Quelques APIs permettent un usage haut niveau
 - Fast.ai, PyTorch Lightning



KERAS : API SEQUENTIAL VS API FUNCTIONAL

- En Keras on a souvent utilisé l'API Sequential, qui est un simple empilement de couches
- Cependant, des architectures plus avancées (comme les variational AE) requièrent un dédoublement/concaténation de couches
- Ex : API Sequential

```
model = keras.Sequential()
model.add( keras.Input(shape=(28,28)) )
model.add( keras.layers.Flatten() )
model.add( keras.layers.Dense( hidden1, activation='relu' ) )
model.add( keras.layers.Dense( hidden2, activation='relu' ) )
model.add( keras.layers.Dense( 10, activation='softmax' ) )
```

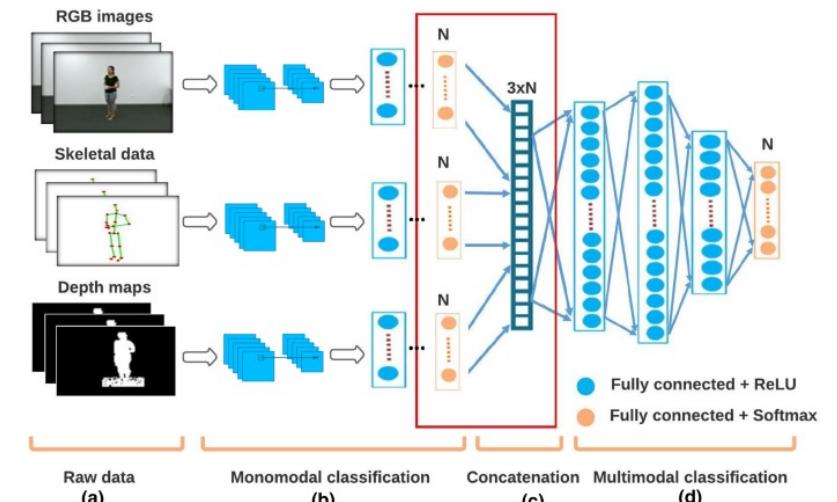
KERAS : API SEQUENTIAL VS API FUNCTIONAL

- En Keras on a souvent utilisé l'API Sequential, qui est un simple empilement de couches
- Cependant, des architectures plus avancées requièrent un dédoublement/concaténation de couches
- Ex : API Functional

```
inputs = keras.Input(shape=(28,28))

x      = keras.layers.Flatten()(inputs)
x      = keras.layers.Dense( hidden1, activation='relu')(x)
x      = keras.layers.Dense( hidden2, activation='relu')(x)
outputs = keras.layers.Dense( 10,      activation='softmax')(x)

model = keras.Model(inputs=inputs, outputs=outputs)
```



KERAS : ENTRAÎNEMENT

- Pour la compilation et l'entraînement, on doit spécifier au moins une fonction de loss

```
model.compile( optimizer = 'adam',
                loss      = [ categorical_crossentropy, mse ]
                loss_weights = [ 0.25, 1. ] )
```

- Également, on doit passer une liste lorsqu'on a plusieurs entrées

```
model.fit( [ x_train1, x_train2], [y_train1, y_train2],
            epochs = 10, batch_size = 64 )
```

EXERCICE 2

- Ici, nous allons apprendre à programmer avec Keras en reproduisant l'expérience d'Yan Lecun en 1998
 - Le dataset MNIST
- Utilisation de réseaux denses (DNN) et convolutionnels (CNN)
- Accéder le site <https://t.ly/5WWz>



EXERCICE 3

- Et si on faisait de la classification de vêtements ?
 - Le dataset Fashion-MNIST (Zalando)
- Utilisation de réseaux convolutionnels (CNN)
- Accéder le site <https://t.ly/a0lY>

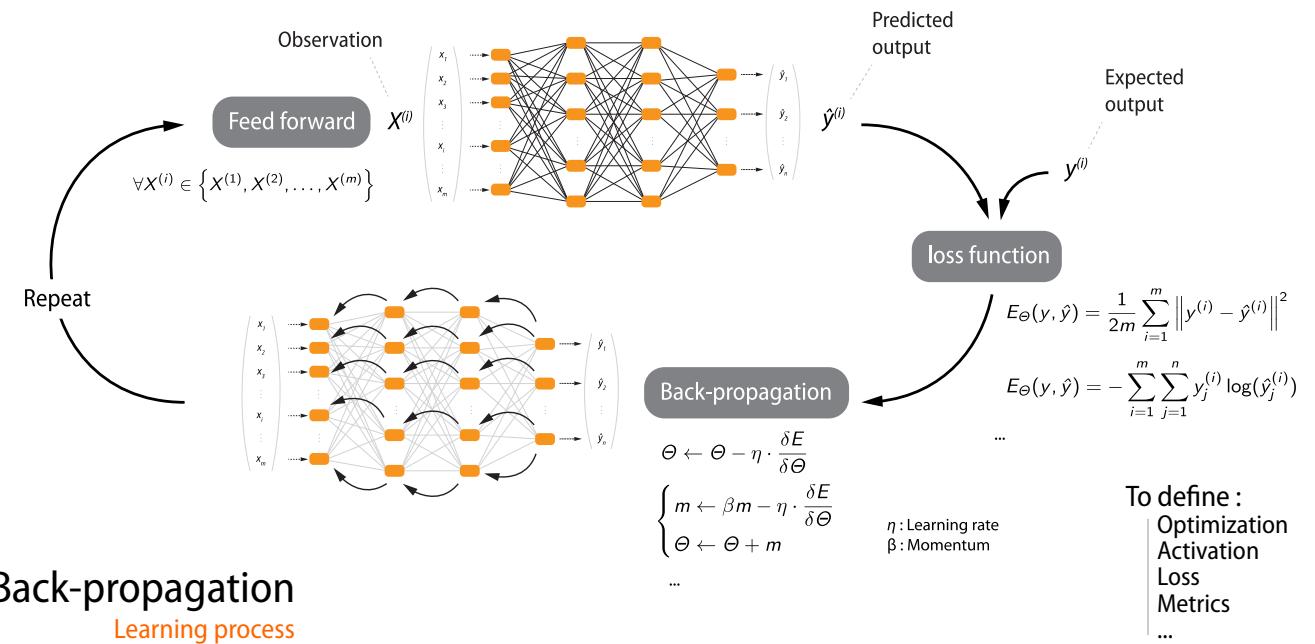


Reposer sur les
épaules
des géants



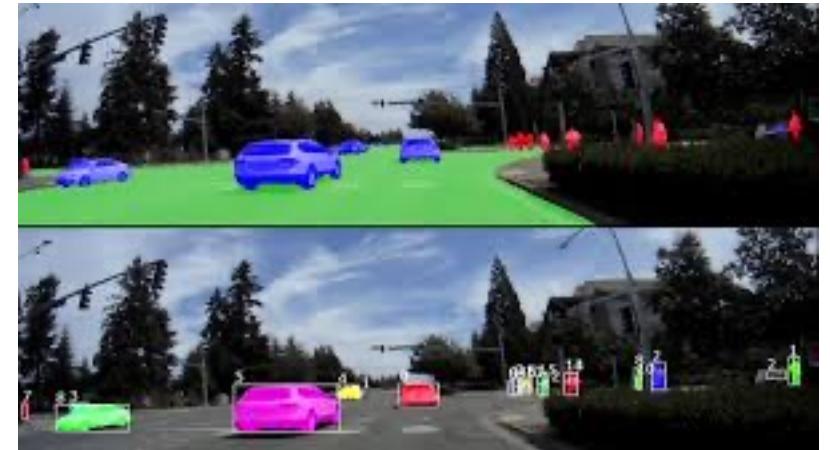
COÛT D'UN ENTRAÎNEMENT COMPLET

- La rétropropagation s'effectue sur l'ensemble des couches
- Opération lourde, surtout si le réseau est profond et le dataset grand et complexe
 - Très lent sans une GPU
- La question cependant est :
 - Faut-il entraîner un modèle du zéro à chaque fois ??
 - Après tout, les couches inférieures détectent des "patrons" bien connus



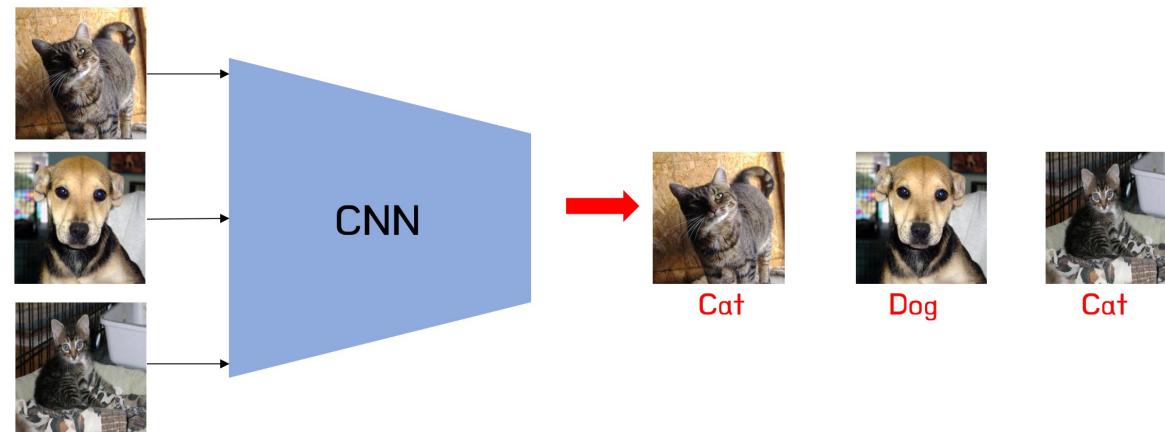
OPTION 1 : UTILISER UN MODÈLE PROCHE

- Plusieurs auteurs proposent des modèles extensivement entraînés sur des datasets publics
 - Ex : ImageNet -> 1000 classes, des millions d'images
- On peut facilement "emprunter" un modèle prêt et juste utiliser les classes qui nous intéressent
- Plusieurs sources disponibles
 - Tensorflow Hub
 - Nvidia NGC
 - Github
 - Une recherche sur Google...



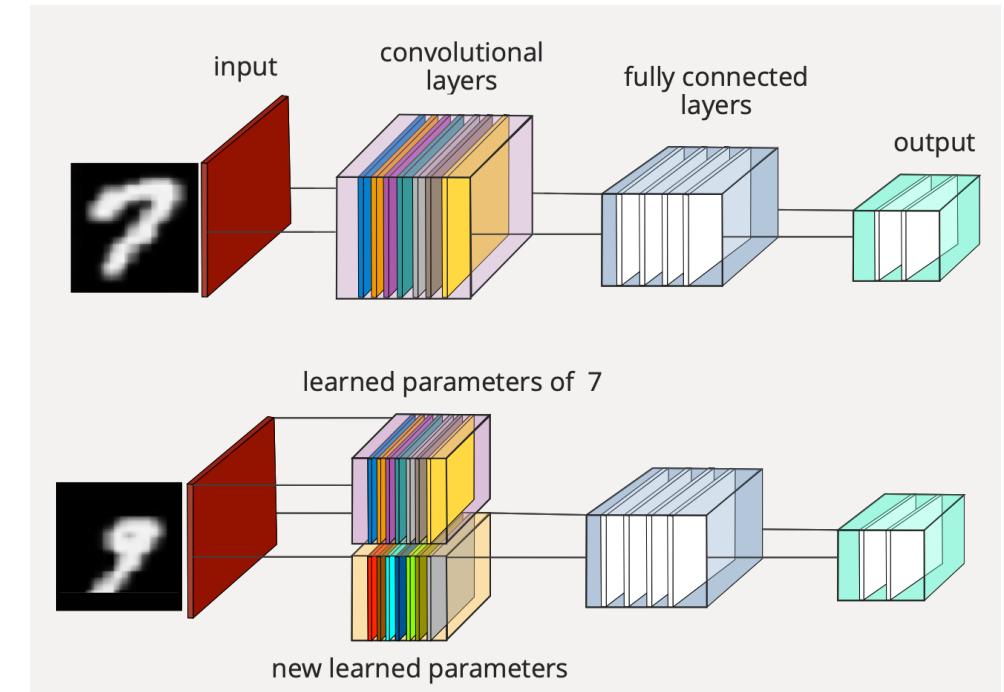
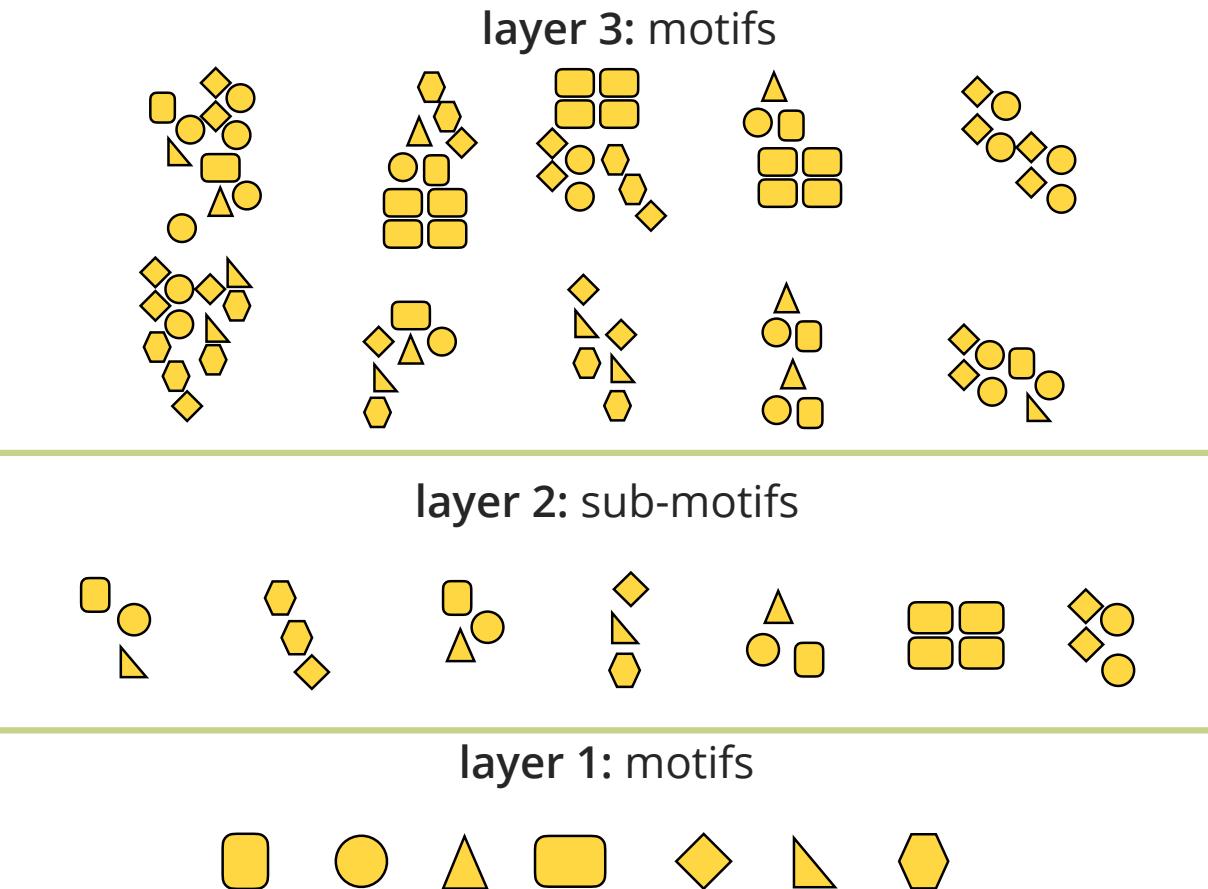
EXERCICE 4

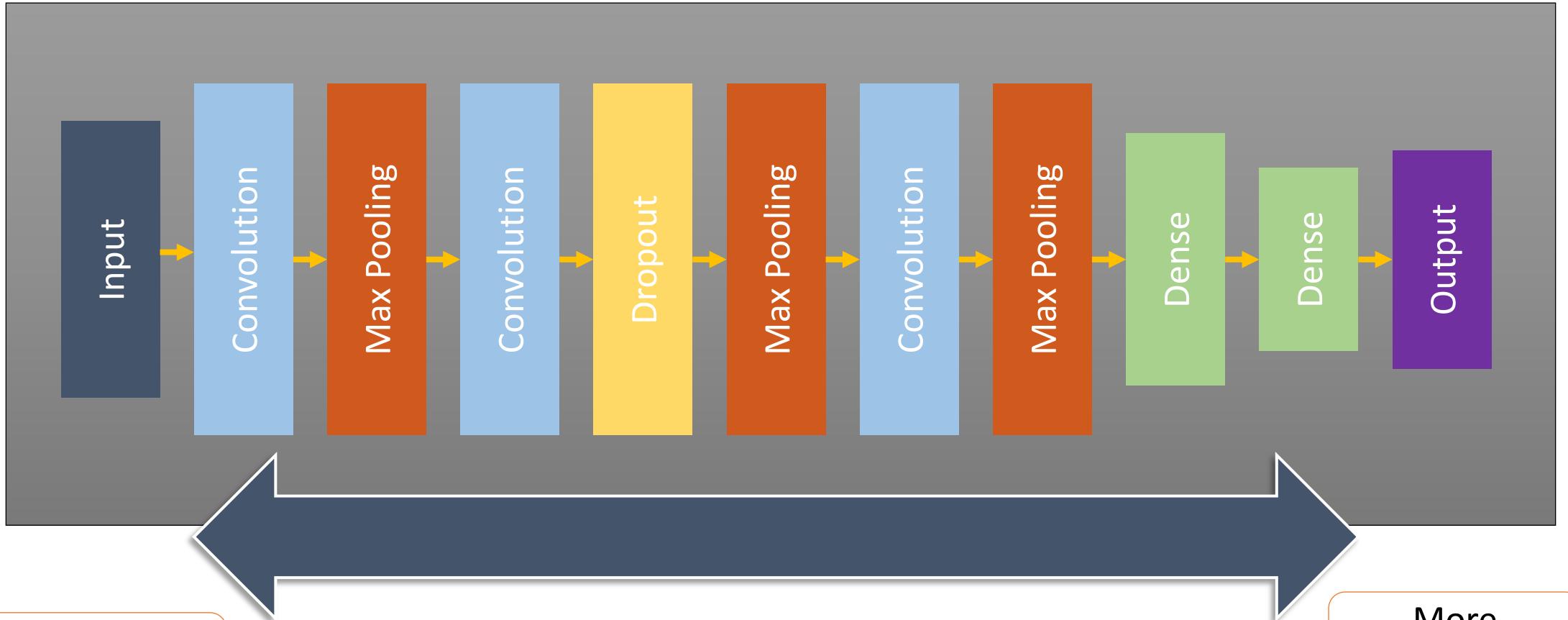
- Utiliser un modèle existant pour identifier Chiens x Chats
- Utilisation du modèle VGG 16
- Accéder le site <https://t.ly/ygyG>



OPTION 2 - TRANSFER LEARNING

- Les avantages d'une organisation hiérarchique des features

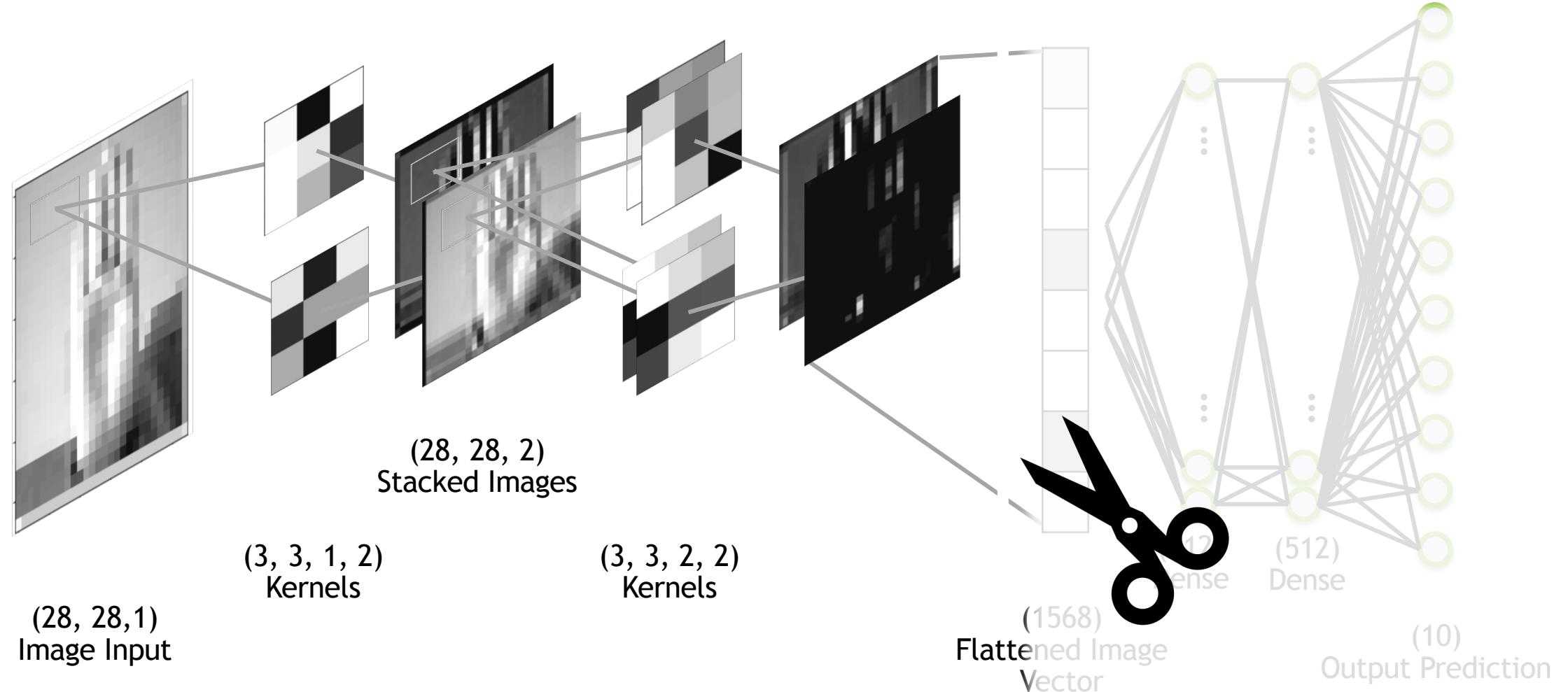




More
Generalized

More
Specialized

TRANSFER-LEARNING À PARTIR DE MODÈLES PRÉ-ENTRAÎNÉS



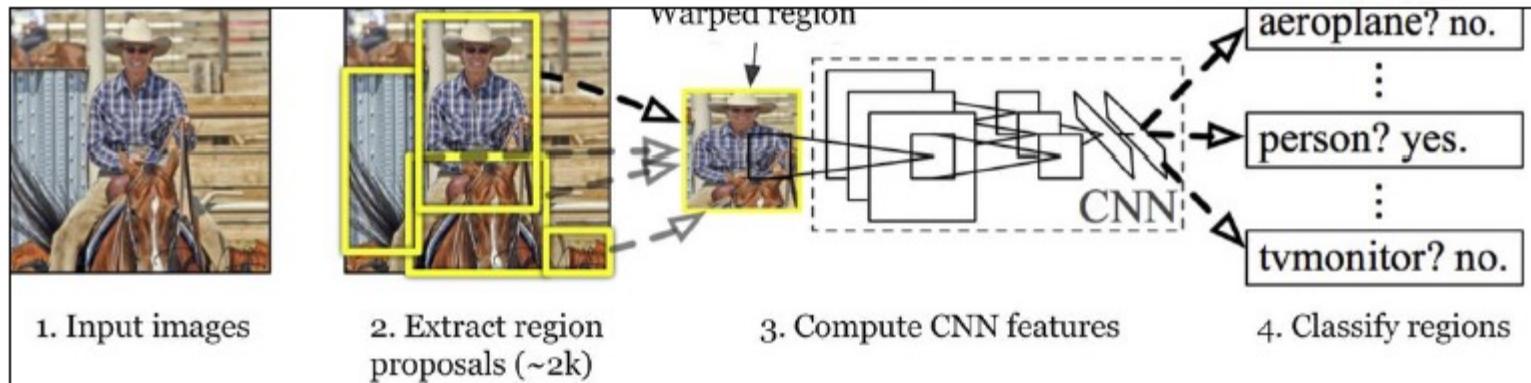
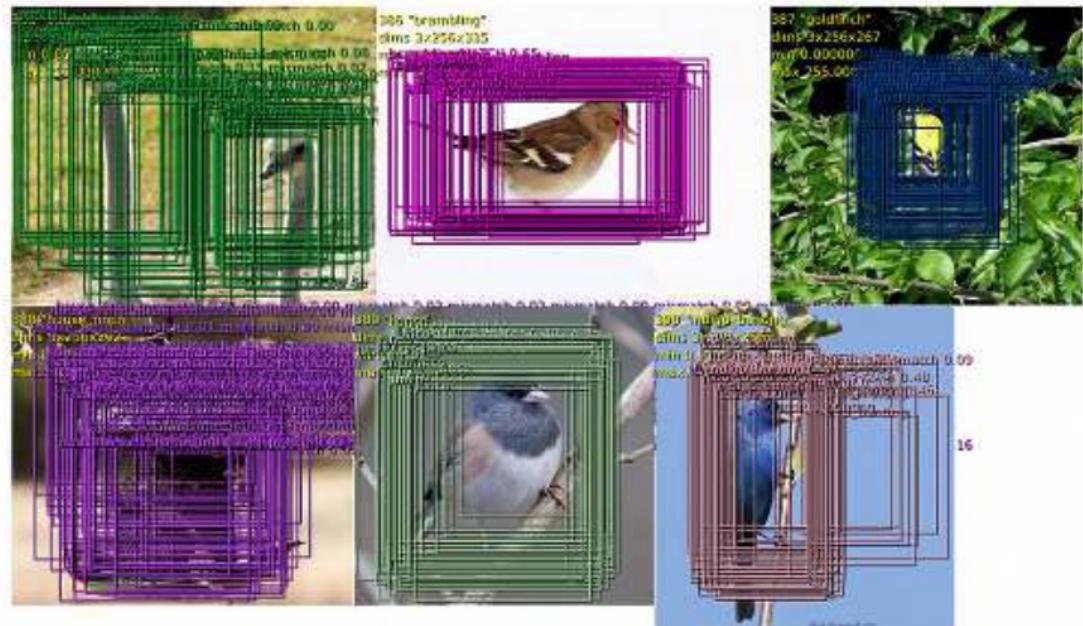
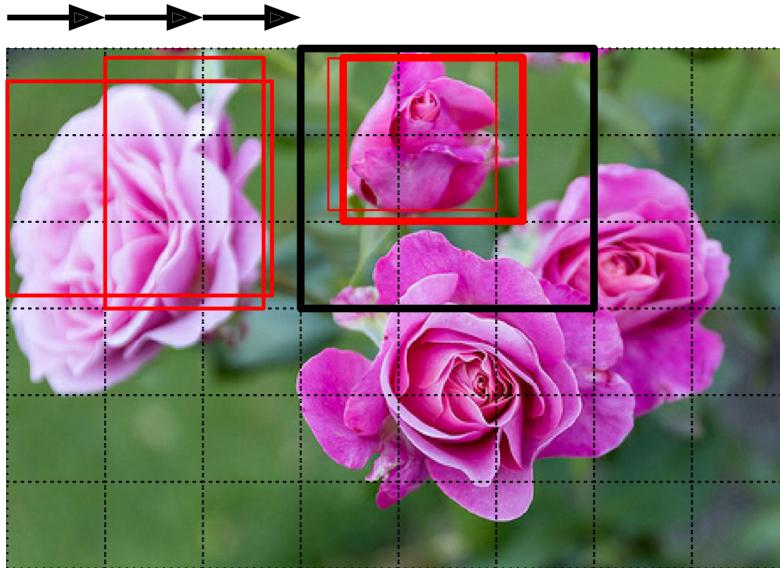
EXERCICE 5

- Mieux qu'identifier Chiens vs Chats, détecter un chien spécifique !!
 - Bo, le chien de Barack Obama
- Accéder le site <https://t.ly/axam->

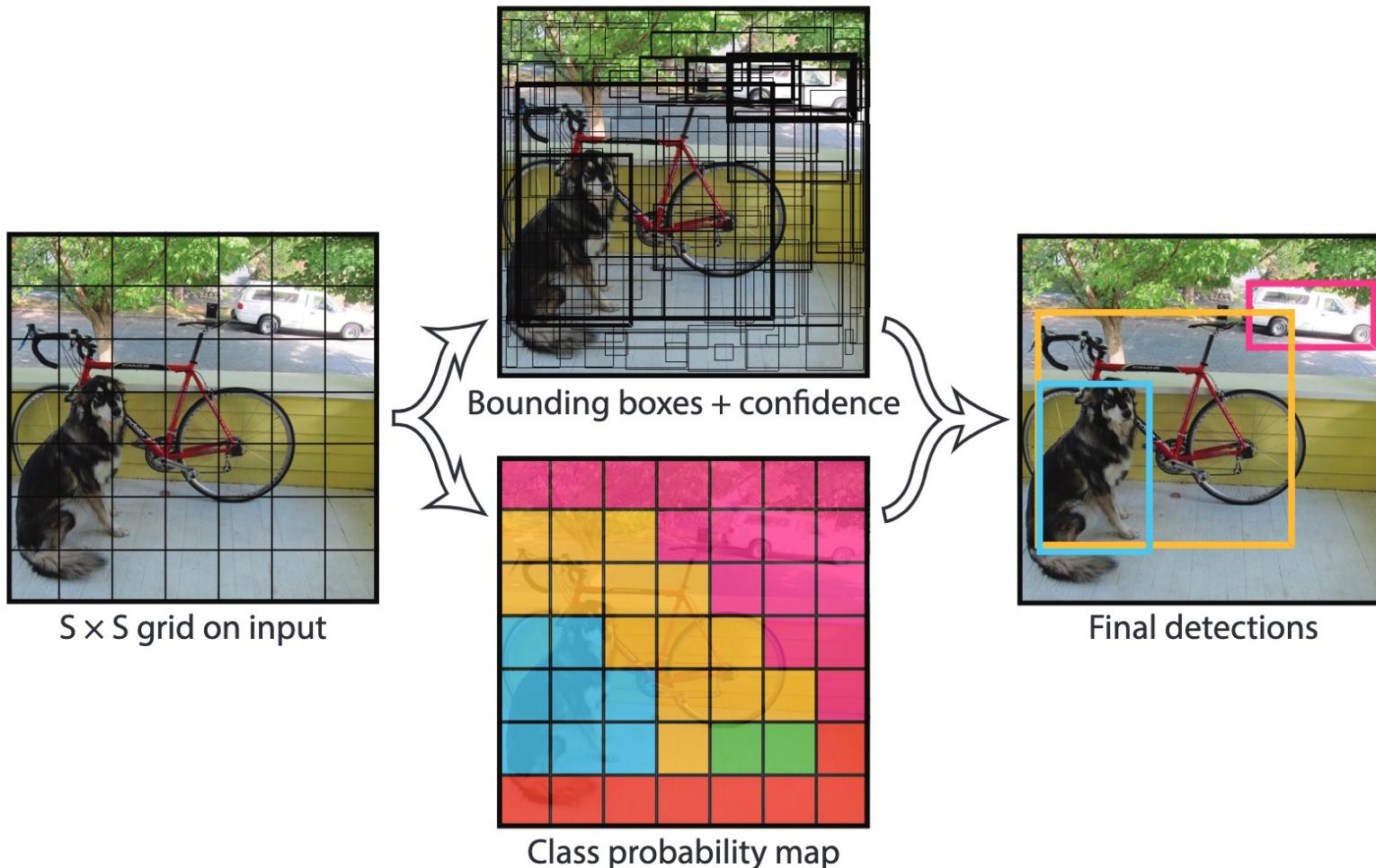


CLASSIFICATION, MAIS PAS SEULEMENT

- Si les CNN excellent dans la classification d'images, on peut les détourner à d'autres usages
- Ex : La détection d'objets



Ex : YOLO (YOU ONLY LOOK ONCE)

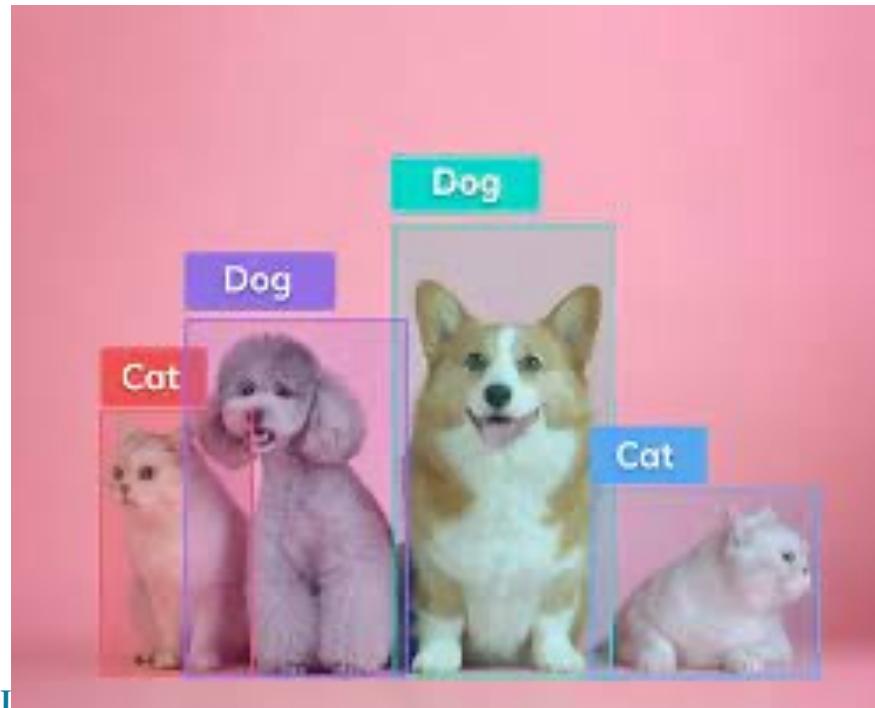


(Redmon et al., 2015)



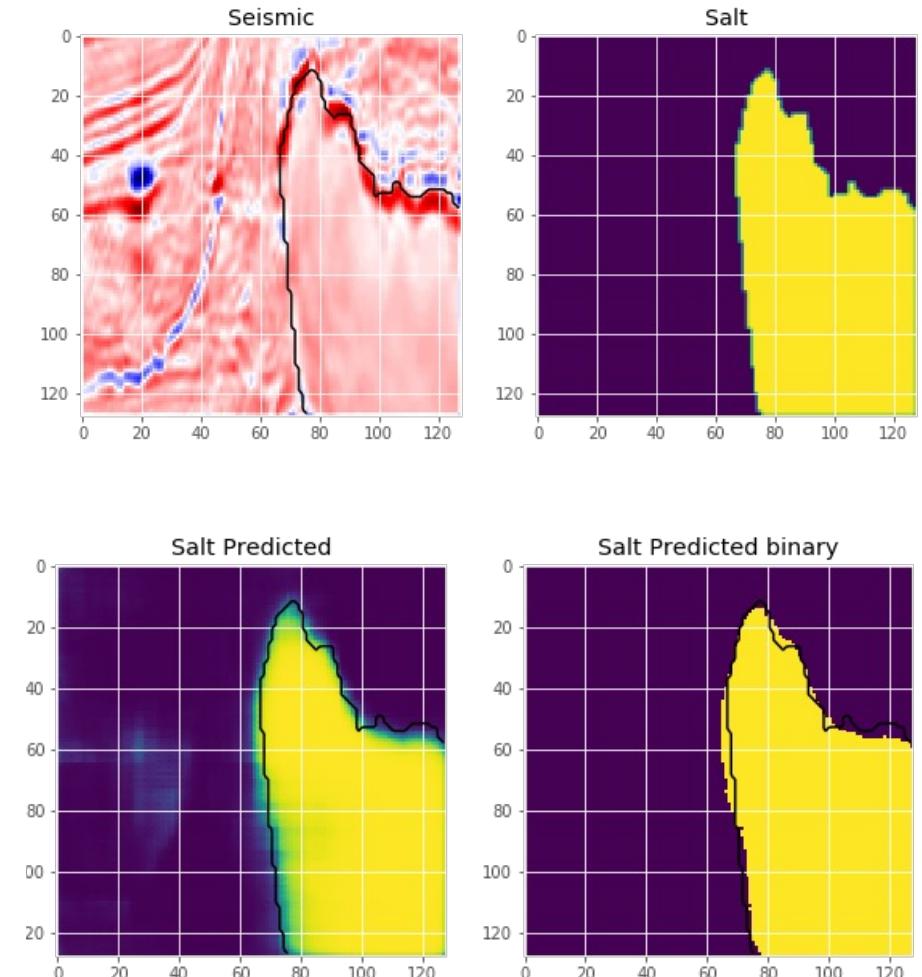
EXERCICE 6

- Faire de la détection d'objets
- Accéder le site <https://t.ly/RE9M>

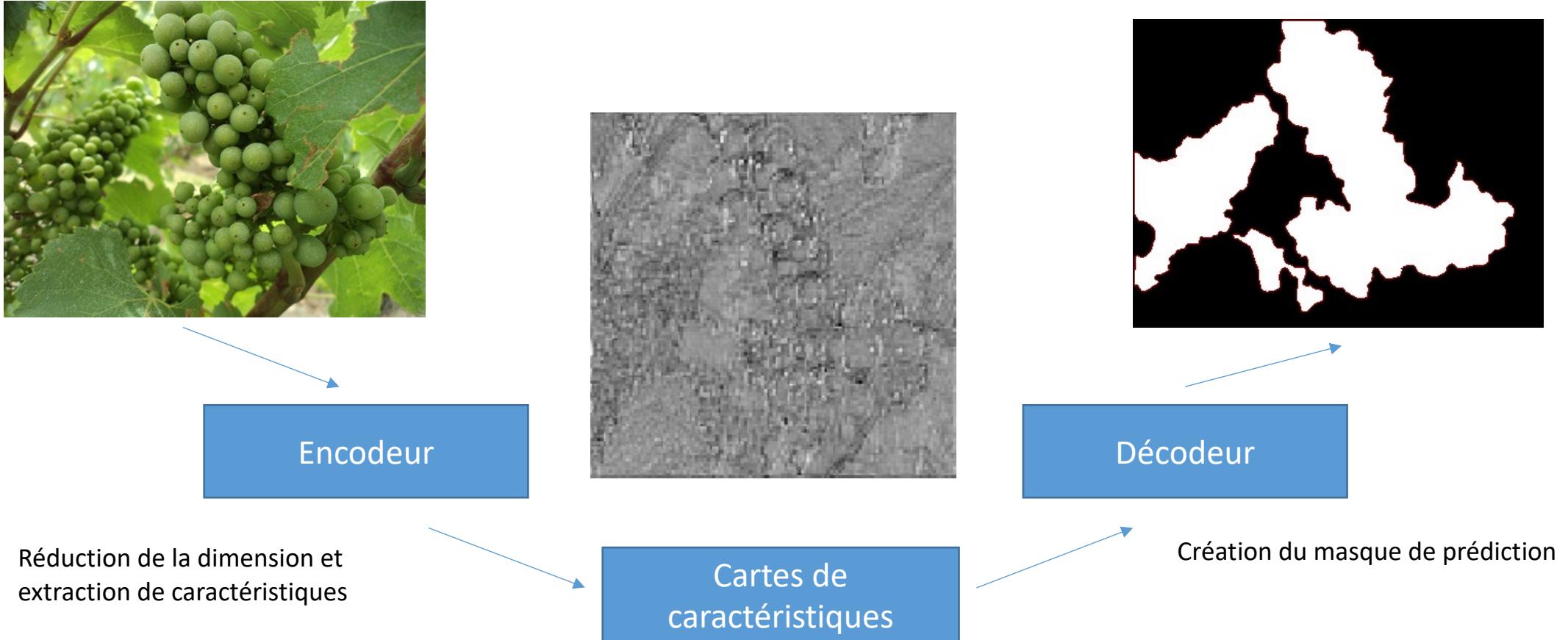


SEGMENTATION SÉMANTIQUE

- Chaque pixel est classé individuellement
- Les labels sont des images avec masques (classes détournées)



ENCODEUR-DÉCODEUR POUR LA SEGMENTATION SÉMANTIQUE



EXERCICE 7

- Un peu de segmentation sémantique
- Accéder le site <https://t.ly/XNVb>



EXERCICE FINAL (DEVOIR ?)

- Aider à organiser la défense civile
- Grâce à des images aériennes, identifier des zones touchées par un ouragan
- Comparer la performance de différents modèles de classification
- Accéder le site <https://t.ly/M5Bs1>



C'EST LA FIN POUR AUJOURD'HUI

- On espère que cela vous a plu
- Il y encore beaucoup de choses qu'on n'a pas couvert
 - Séries temporelles (prévisions)
 - Génération de données (GAN, diffusion stable)
 - Modèles linguistiques (ChatGPT)
- N'hésitez pas à nous contacter si vous voulez essayer quelque chose avec vos données de thèse
 - Le LICIIS a des chercheurs qui peuvent vous aiguiller
 - ROMEO a tout le matériel nécessaire pour tourner vos expériences
- Tout le matériel de ce cours restera disponible sur :
 - <https://github.com/lsteffenel/ED-SNI-IntroDL>

