L’intelligence artificielle au service de la vision automatique

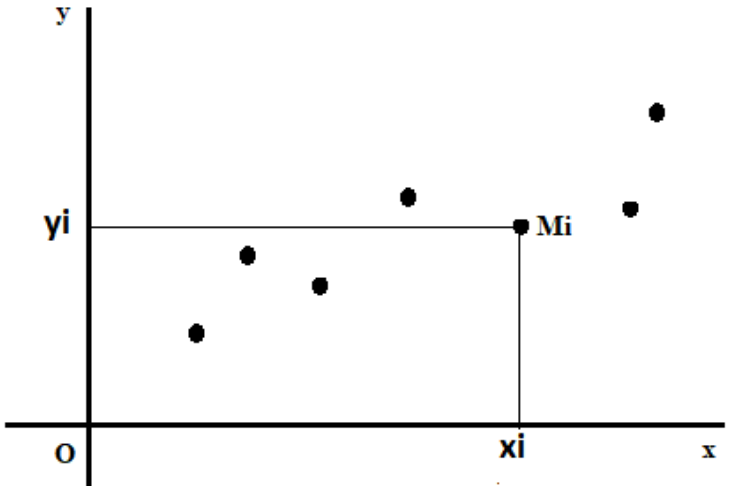
Jean CEA

Quand un ordinateur apprend à reconnaitre le contenu des images

- Si vous présentez une image à un ordinateur et vous lui demandez : que représente cette image ? Il ne saura pas vous répondre !
- Il faut, tout d'abord lui dire ce qu'est une **image** et ensuite, lui expliquer ce que **contient** cette image : il s'agit d'un chat, d'un chien, du chiffre 3 ...
- On pourrait tenter de **décrire pas à pas** le contenu des images, les courbes, les angles, les volumes, les couleurs puis faire un programme pour reconnaître tout cet attirail.
- Cette méthode est trop lourde, il y a **trop de cas particuliers**.
- On a trouvé une autre méthode qui consiste à montrer à un ordinateur **des dizaines de milliers de chiffres manuscrits, des centaines de milliers d'images différentes mais cataloguées ou étiquetées**. Il va apprendre à partir de ces exemples.
- Ensuite, on va ***tester sur d'autres images*** qu'il est capable de dire ce que contient chaque image (le chiffre 4, un chat, une auto, un verre…). **Après des tests de confirmation, l'ordinateur sera opérationnel.**
- C'est **l'apprentissage profond** des ordinateurs (ou **Deep Learning**).

1. Régressions

Un nuage de points alignés… ou presque



Représentation graphique du nombre de tortues d'une réserve en fonction du temps écoulé (en années)

En abscisse x : le nombre d'années écoulées

En ordonnées y : le nombre de tortues

Nous faisons des relevés tous les ans à partir d'une certaine date.

Il y a de nombreux problèmes semblables, aussi nous oublions l'origine et nous nous intéressons à ce qu'il y a de commun à ce genre de problème.

http://www.jybaudot.fr/Correl_regress/rls.html

On dispose de n points du plan.

Ils représentent des **variables quantitatives**, disons, qu'il s'agit de nombres réels.

On s'est limité à deux composantes : **(x_i, y_i) par point**

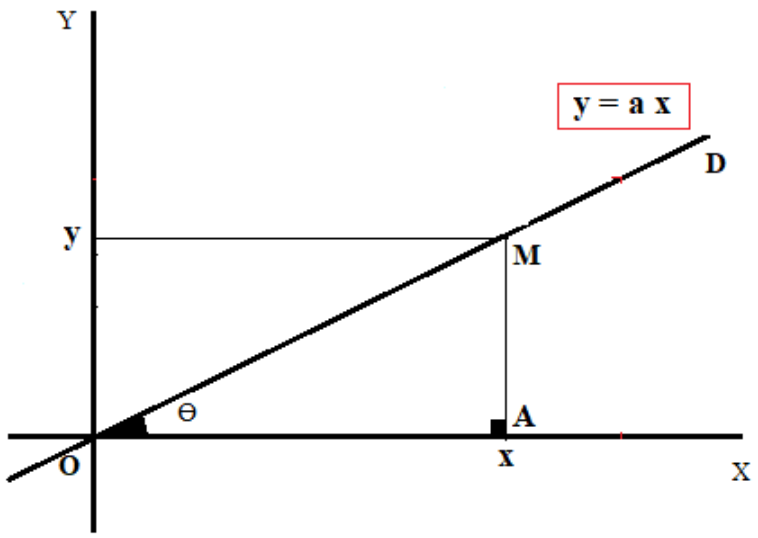
A chaque **y_i** on aurait pu associer davantage de composantes : $x_i = (x_i^1, x_i^2 \dots x_i^k \dots x_i^m)$

[Par exemple, en ce qui concerne ***les composantes x qui interviennent dans le prix y d'un appartement***, on aurait pu introduire la surface, la qualité de l'immeuble, la qualité du quartier, l'existence d'un concierge, le nombre de mètres carrés d'un jardin…]

Mais, cela ne ferait qu'alourdir les notations, alors que dans ce que nous allons faire, le raisonnement sera le même avec 1 ou 10 composantes. Donc, simplifions : ainsi, **nous disposons de n points repérés par (x_i, y_i) : <u>y est la cible, x est une variable explicative</u>.**

<u>Problème</u> : ces points semblent alignés. Peut-on trouver une droite qui passe au plus près de ces points ?

Une droite dans le plan



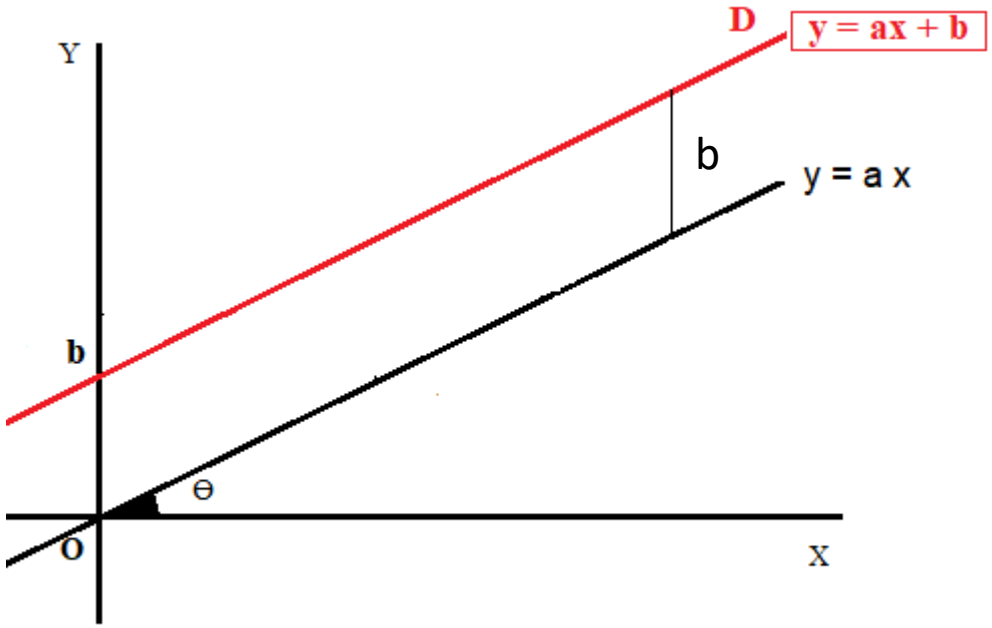
Tangente θ = côté opposé / côté adjacent

tan θ = y/x = AM/OA

Définition de a c'est la tangente de l'angle θ

a est la pente de la droite

a = y/x ou bien **y = a.x**



a = tan θ b = ordonnée à l'origine

- L'équation d'une droite dans un plan dépend de 2 paramètres : **a** et **b, la pente a** **et l'ordonnée à l'origine b**

Notre modèle

Nous choisissons comme modèle une droite. Il existe de très nombreuses courbes pour décrire certains aspects de la nature, par exemple, la parabole, une cubique, un limaçon. Selon la disposition du nuage, il faut choisir son modèle.

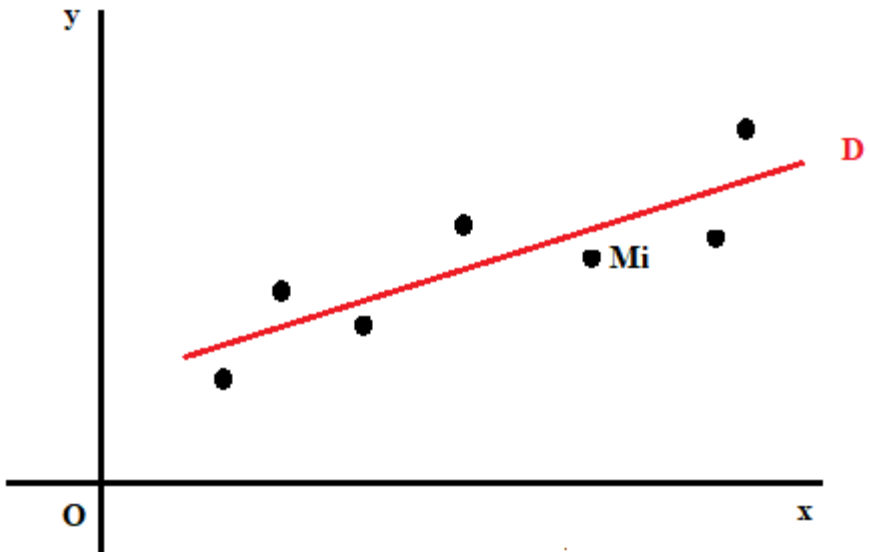
- Dans un **modèle**, il y aura toujours la **fonction f**, les **variables x**, la **cible y** et les **paramètres a, b**…

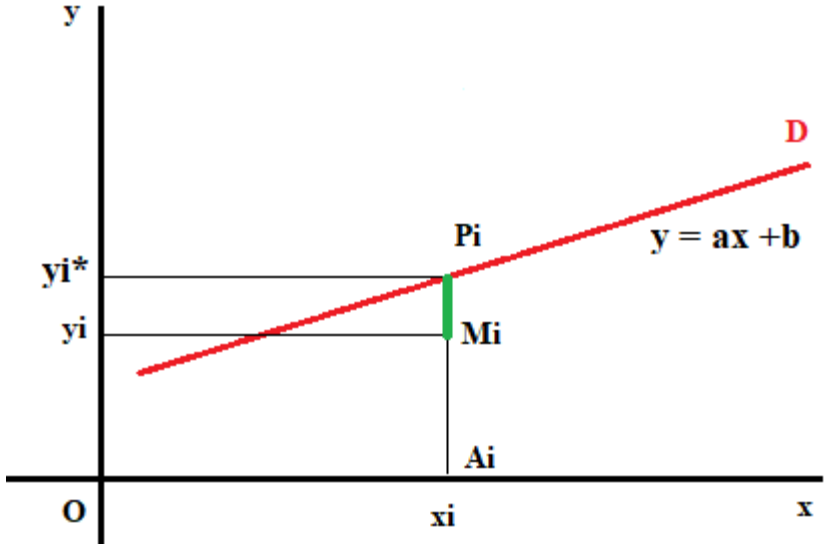
  y = f(x, paramètres) ,

pour nous, notre modèle sera :

- **y = f(x, a, b) = a x + b**

Un problème d'optimisation





La droite D semble être proche des points Mi, i allant de 1 à n. Mais **cette notion de proximité** a besoin d'être développée.
Nous supposons que le droite D a pour équation : **y = a x + b, a et b étant inconnues.**

Nous étudions à partir de cette figure la contribution dite locale de **M_i** au problème global.
On connait les coordonnées des M_i elles sont **données** : x_i et y_i . On connait aussi les coordonnées des points Pi puisque ces points sont sur la droite D : x_i est **donnée** et yi^* = a x_i + b, yi^***est calculée**
Donc : **Erreur$_i$ = M_i P_i** = A_i P_i – A_i M_i = yi^* - y_i = a x_i + b – y_i = **f(x_i , a, b) - y_i**

Si on ajoute ces erreurs locales, *les positives peuvent compenser les négatives*, le résultat n'aura aucun sens. Nous préférons **Erreur$_i$ = |a x_i + b – y_i|** et mieux **Erreur$_i$ = (a x_i + b – y_i)^2** **(dérivabilité)**
Nous posons maintenant pour la somme des erreurs : **E(a,b) = 1/n $\sum_{i=1}^{n}$ (a xi + b – yi)^2 = 1/n $\sum_{i=1}^{n}$ (f(xi , a, b) – yi)^2**
Il s'agit de l'erreur quadratique moyenne, tout est connu sauf a et b.

- **Nous cherchons a* et b* pour lesquels : E(a*,b*) <= E(a,b) pour tout a et pour tout b**

La droite optimale, prévision

- Les mathématiciens savent résoudre ce problème d'optimisation, **on le suppose résolu (*moindres carrés*)**. On dispose donc de a* et de b*. La droite D optimale a pour équation : **y = a* x + b***
- Avec seulement **2 nombres a* et b***, nous avons une idée plus précise du tableau des coordonnées des points M_i

➢ De plus, cette équation peut servir pour une **prévision** :

- Si je vous donne un x, que prévoyez-vous comme y pour rester cohérent avec les autres points (et avec le modèle choisi) ? Naturellement : **y = a* x + b***
- Le mot **« Régression »** a été proposé en 1886 par **Francis Galton**. Il constate un phénomène de « **régression vers la moyenne** » de la taille des fils en fonction de la taille des pères (Wikipedia)

De nombreux exemples de régressions

- Rappelons qu'il s'agit de **variables quantitatives**

- Le prix de vente des appartements en fonction des surfaces (et de bien d'autres caractéristiques).
- Températures en fonction de l'altitude pour 8 stations météo d'une station alpine.
- Taille et poids de n individus.
- Relations entre investissements publicitaires et ventes.
- Nombre d'heures passées à fumer et évaluation de l'humeur sur une échelle de 1 à 10.
- Pluviométrie en fonction de l'altitude dans les Alpes du nord.
- Tension en fonction de l'âge.

- ***On résume un ensemble important avec 2, 3… paramètres.***
- ***On se sert du modèle pour des prévisions.***

Récapitulation et ouverture vers les chiffres et les images

- On s'est donné n points, disons (x^i, y^i), i variant de 1 à n.
- **On a introduit une fonction du type : y = F(x, paramètres)** en choisissant les **paramètres optimaux**, pour que la somme des « erreurs », si on approche les y^i connus par les F(x^i, paramètres optimaux), soit **minimum**. Les mathématiciens savent résoudre ce problème d'optimisation (***moindres carrés***).
- On s'est limité au cas où **y est un nombre réel**. Cependant, il existe des problèmes semblables où **y est une étiquette, une catégorie**… Par exemple : y est un chiffre manuscrit 0, 1, 2… 9. y est une catégorie d'animaux : chien, chat, cheval, lion… Ou bien : auto, moto, autobus, camion… Dans ces cas, y peut prendre des valeurs autres qu'un nombre réel. Pour les 10 chiffres, on peut introduire des y de la forme $\mathbf{y = (y_0, y_1, y_2 \ldots y_9)}$ avec $\mathbf{y_i >= 0}$ pour tout i et $\sum y_i = 1$. Maintenant **y est un ensemble de probabilités !** Pour les données (donc parfaitement connues) les (x^i, y^i) contiennent : d'abord le **descriptif x^i** de l'objet en question, un appartement et maintenant aussi une image, un chiffre et puis la valeur des **probabilités y^i** pour que l'objet soit telle catégorie de chiffre ou d'image…
- Une valeur **y connue** donc **sûre** est du type **(1, 0, 0.. 0)** ou **(0, 1, 0… 0)** ou **(0, 0, … 0, 1)** s'il y a 10 catégories. **Une valeur y calculée** n'est pas certaine, elle peut comporter une petite erreur, elle peut être du type : (0,09 0,1 **0,7** … 0,02). La probabilité la plus grande indique qu'il s'agit de l'étiquette ou catégorie **2, valeur proposée à 70 %** (on commence par la catégorie 0)
- **On pourra maintenant se servir de la fonction F pour faire des prévisions dans le cas où y repère une catégorie ou étiquette. Ce sera le cas des chiffres manuscrits et des images**

➢ **Il nous faudra inventer de nouvelles fonctions** avec de nombreux x, de nombreux y et de nombreux paramètres.

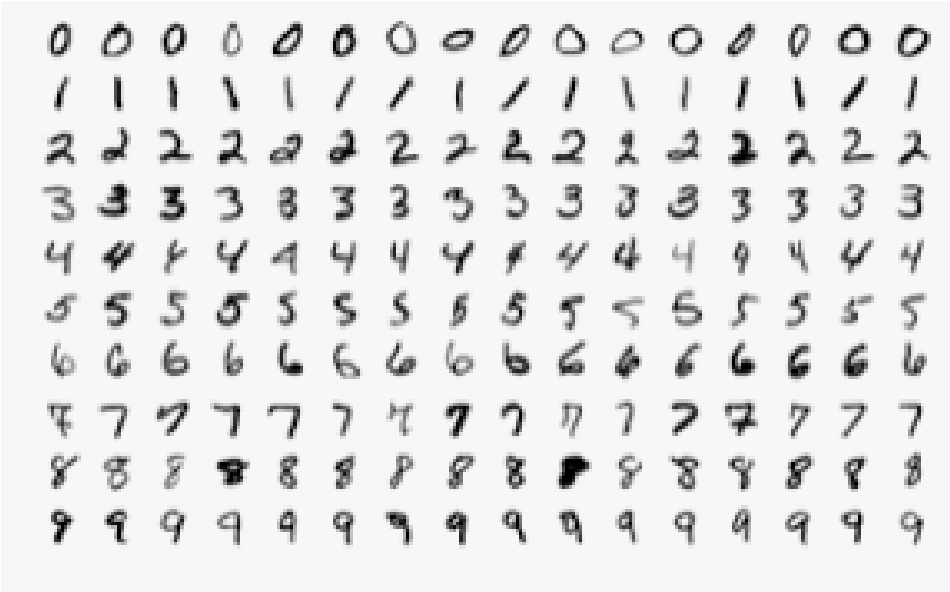
2. Le repérage d’une image

Une image de près, de loin, les PIXELS

IMAGE INFORMATIQUE : un carrelage rectangulaire, chaque carreau ayant sa propre couleur. Un carreau est appelé un **PIXEL**.
La couleur de ce pixel est repérée par un ou plusieurs nombres : **niveau de gris**, **couleurs RVB (Rouge, Vert, Bleu)**.

Une image, c’est quoi pour un ordinateur ?

Voici 16 façons d’écrire (à la main) les chiffres 0, 1… 9
Il y en a des milliers ! Beaucoup plus, car nous sommes des milliards sur terre !
A chacun son style.



MNIST database - Wikipedia
en.wikipedia.org

Chiffres 3 (28 X 28 = 784 pixels) et 8 (18 X 18 = 324 pixels)

Les niveaux de gris vont de 0 à 255. Pourquoi ? Parce que avec 8 bits on peut stocker un nombre de 1 à 256, ou de 0 à 255

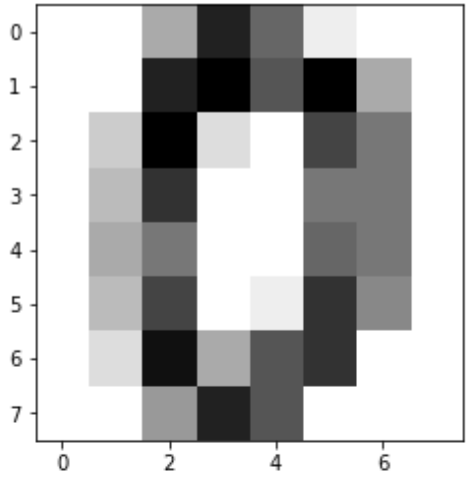
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	29	150	195	254	255	254	176	193	150	96	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	48	166	224	253	253	234	196	253	253	253	253	233	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	93	244	249	253	187	46	10	8	4	10	194	253	253	233	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	107	253	253	230	48	0	0	0	0	0	192	253	253	156	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	3	20	20	15	0	0	0	0	0	43	224	253	245	74	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	249	253	245	126	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	14	101	223	253	248	124	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	11	166	239	253	253	253	187	30	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	16	248	250	253	253	253	253	232	213	111	2	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	43	98	98	208	253	253	253	253	187	22	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	51	119	253	253	253	76	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	183	253	253	139	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	182	253	253	104	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85	249	253	253	36	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	214	253	253	173	11	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	98	247	253	253	226	9	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	42	150	252	253	253	233	53	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	42	115	42	60	115	159	240	253	253	250	175	25	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	187	253	253	253	253	253	253	253	197	86	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	103	253	253	253	253	253	232	67	1	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

L'image a été créée à l'aide du code fourni dans le livre : Deep Learning for Coders with fastai and PyTorch

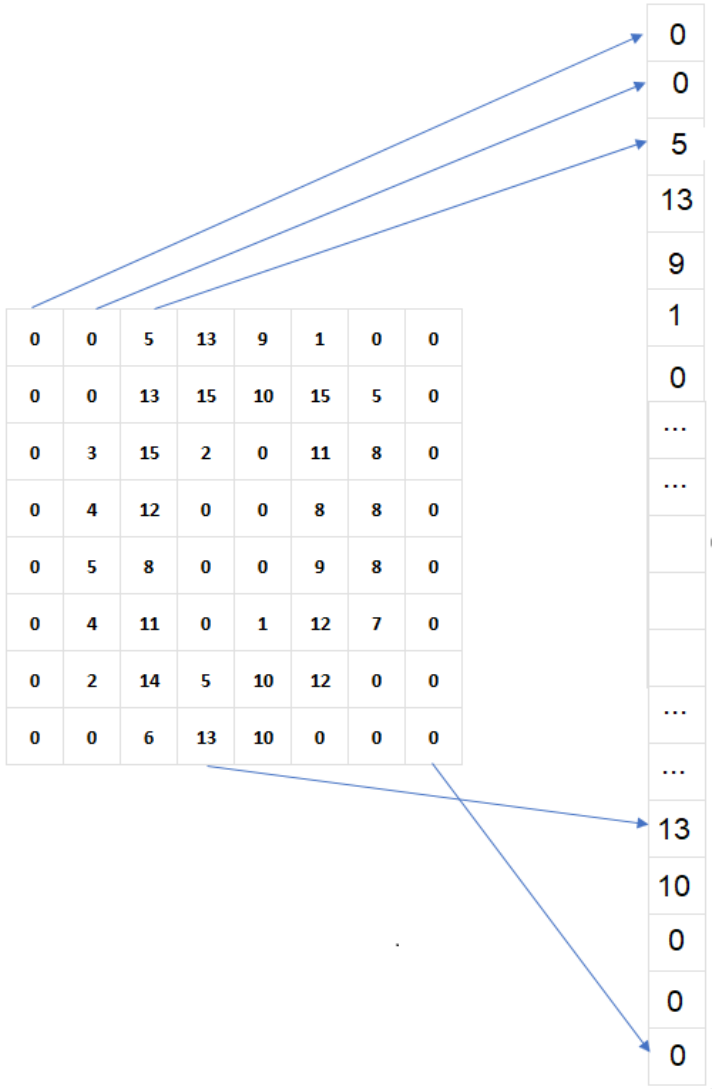
https://ichi.pro/fr/pixels-la-base-de-la-vision-par-ordinateur-151661518724346

Une image <-> ‘0’ <-> un tableau 8 X 8 <-> Un vecteur 64 X 1



'0'

0	0	5	13	9	1	0	0
0	0	13	15	10	15	5	0
0	3	15	2	0	11	8	0
0	4	12	0	0	8	8	0
0	5	8	0	0	9	8	0
0	4	11	0	1	12	7	0
0	2	14	5	10	12	0	0
0	0	6	13	10	0	0	0



Chiffre décomposé en 8 X 8 = 64 pixels, avec16 niveaux de gris

https://thedatafrog.com/fr/articles/handwritten-digit-recognition-scikit-learn/

Ou bien, avec une notation mathématique, pour une écriture horizontale :

‘0’ <--> (0, 0, 5, 13… 10, 0, 0, 0)

‘0’ € R^{64}, notre image devient un point x d’un espace mathématique :

$x \in R^{64} \Leftrightarrow x = (x_1, x_2 \ldots x_{64})$

pour tout i, $x_i \in R$, x_i nombre réel.

- **Une image est un point de l’espace R à une certaine puissance, le nombre de pixels de l’image.**

En couleurs, il faut multiplier cette puissance par 3 (Rouge, Vert, Bleu)

Quelques notations mathématiques

$x \in R^{64}$ ⟺ $x = (x_1, x_2, \dots, x_i, \dots, x_{64})$ **pour tout i de 1 à 64,** $x_i \in R$, x_i **nombre réel.**

Exemple : $x \in R^7$ x = (12, -6, 4, 125, 7, -5 , 19) (Pour des images, les niveaux de gris sont >=0)

$x \in R^n$ **n est un nombre entier positif** ⟺ $x = (x_1, x_2, \dots, x_i, \dots, x_n)$ **pour tout i de 1 à n,** $x_i \in R$, x_i **nombre réel.**

Quelques manipulations :

Somme :

x = (2, 3, 5), y = (6, 8, 12) ⟹ x + y = (2 + 6, 3 + 8, 5 + 12) = (8, 11, 17)

Produit par un nombre :

x = (2, 3, 5) ⟹ 7x = 7(2, 3, 5) = (7.2, 7.3, 7.5) = (14, 21, 35)

Norme (longueur) :

$x = (x_1, x_2, \dots, x_i, \dots, x_n)$ ⟹ $\| x \|$ = racine carrée de $(\Sigma | x_j |^2)$ ⟹ $\| x \|^2 = \sum_{i=1}^{n} |x_i|^2$ NORME de x au carré.

Mémoires… informatiques !

- **1 bit :** peut stocker 2 valeurs, **0** ou **1** (**B**inary dig**IT**).
- **1 octet** = ensemble de 8 bits, il a ***une adresse physique*** dans la mémoire. On peut y stocker 8 choix parmi les 2 nombres 0 ou 1, 2^8 choix au final, soit ***256 nombres*** (valeurs en décimales de 0 à 255). C'est le stockage classique pour des **niveaux de gris** de 0 à 255 (**images en noir et blanc**) ; on peut aussi diviser tous ces nombres par 255, on obtient ainsi des nombres qui varient de 0 à 1.
- **Images en couleurs** : on stocke séparément les 3 couleurs **RVB** (Rouge, Vert, bleu) comme le noir et blanc. 3 fois plus de mémoire !

Quelques dimensions classiques :

- Chiffres : 28 X 28 = 784 pixels

- Images : 400 X 400 = 160.000 800 X 800 = 640.000 pixels (ou 3 fois plus en couleurs), une grande image peut contenir des millions de pixels !

Les « grandes » bases de données d'images

- **MNSIT** (Mixed (ou Modified) National Institute of Standards and Technology) : les **10 chiffres**. **60.000 images annotées ou étiquetées en 10 classes**, plus **10.000** images pour les tests. Travail effectué manuellement par 125 employés et 125 étudiants. Images en 28 X 28 = 784 pixels. « **Annotée** » signifie qu'on a rajouté au stockage une « étiquette » pour rappeler de quel chiffre il s'agit, par exemple :

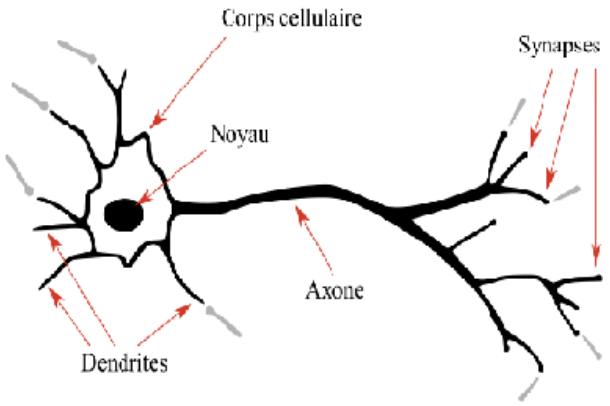
'0' = (1,0,0,0,0,0,0,0,0,0) '3' = (0,0,0,1,0,0,0,0,0,0) '9' = (0,0,0,0,0,0,0,0,0,1)

- **ImageNet** : en 2021 cette base contenait **15 millions d'images** étiquetées, classées par catégories (plus de **1000 catégories**).
- On retient le format standard suivant : 469 X 387 = **181.503 pixels**
- Depuis le 11 mars 2021, 243.198 photos ont subi un **floutage du visage** pour éviter des exploitations malveillantes !
- **Ces images sont disponibles gratuitement !**

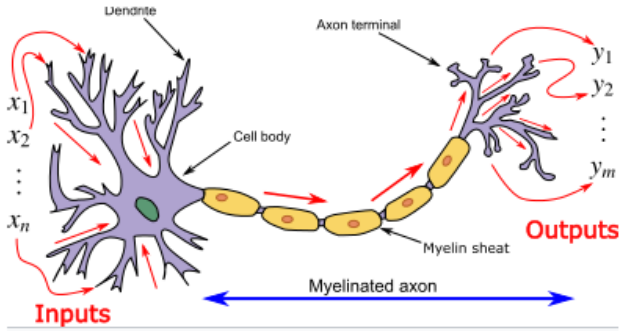
3. Les neurones biologiques et formels (ou mathématiques)

Les réseaux de neurones

Les composants d'un neurone
Associations mathématiques diverses



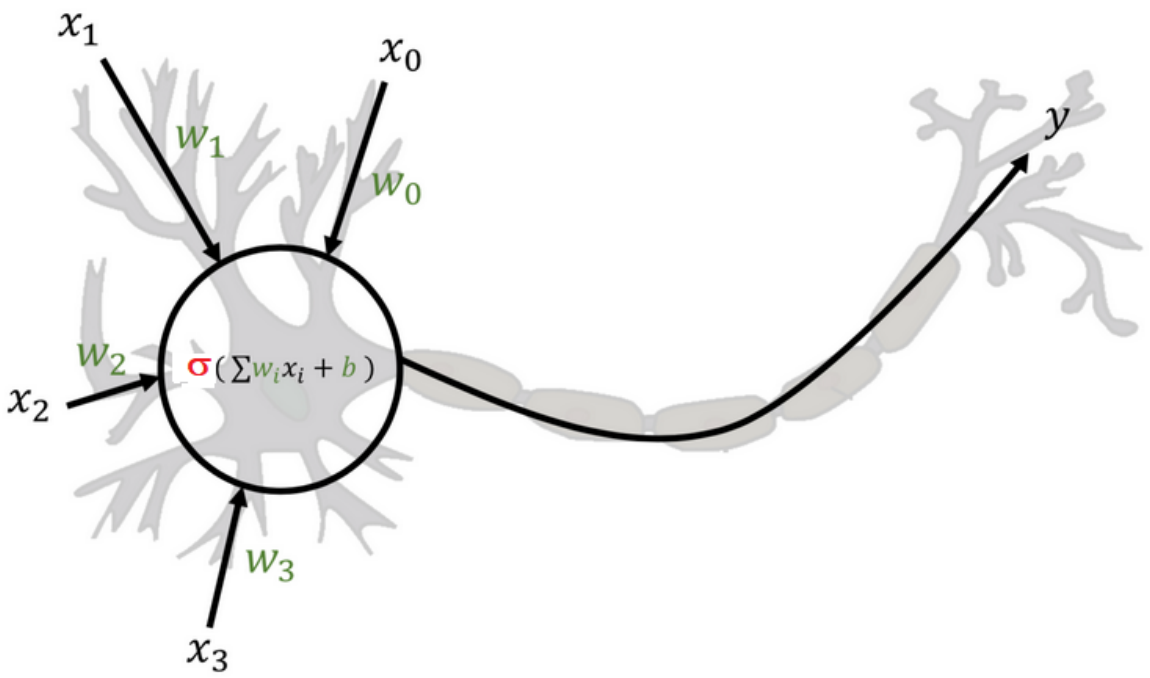
Un neurone biologique



Neuron and myelinated axon, with signal flow from inputs at dendrites to outputs at axon terminals

Entrées - Sorties d'un neurone

Figure 9. Schéma d'un neurone informatique superposé à un schéma de neurone biologique



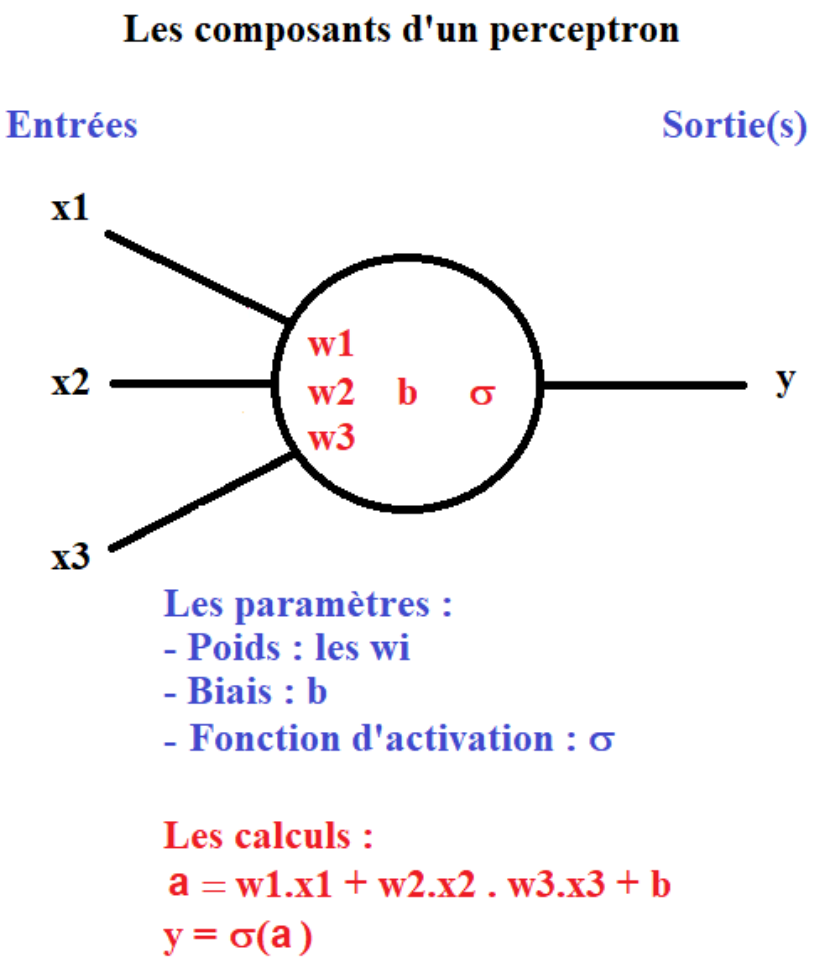
Les variables écrites en vert sont les paramètres du neurone. Source de l'image du neurone biologique : Wikipedia.

--> La fonction σ est une fonction d'activation

Merci à https://culturesciencesphysique.ens-lyon.fr/ressource/IA-apprentissage-Rousseau.xml

Un cerveau humain contient 80 milliards de neurones, et 10.000 synapses par neurone.
Nombre de connexions énorme ! Poids d'un cerveau moyen : 1,4 Kilos.

Un modèle mathématique d'un neurone :
Le PERCEPTRON : un générateur de fonctions



En **1943**, Warren **McCulloch** & Walter **Pitts**, proposent le premier modèle de neurone formel. Il est trop simple, imprégné par la logique mathématique : NON, ET, OU, OU EXCLUSIF… En 1959, ils participeront à l'écriture de l'article intitulé "*What the Frog's Eye Tells the Frog's Brain (Ce que l'oeil d'une grenouille dit à son cerveau)*

1957/58 : modèle proposé par Franck **Rosenblatt**, il est livré avec la première « règle d'apprentissage »

1969 : mise en avant des **défauts** des modèles par Marvin Minsky et Seymour Papert.

1974-1980 : Hiver !

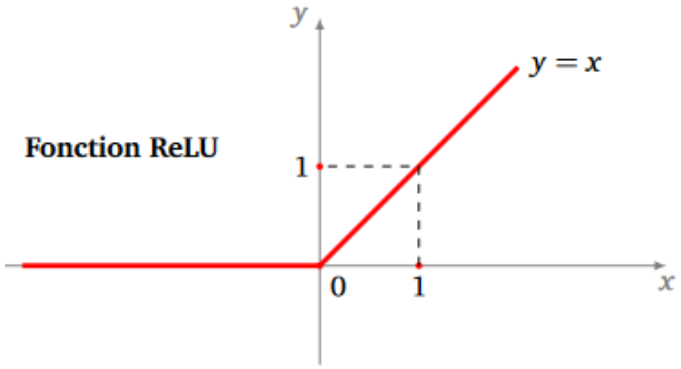
1985-1986 : Paul **Werbos**, David **Rumelhart**, Yann **Le Cun** proposent séparément le perceptron multi-couche

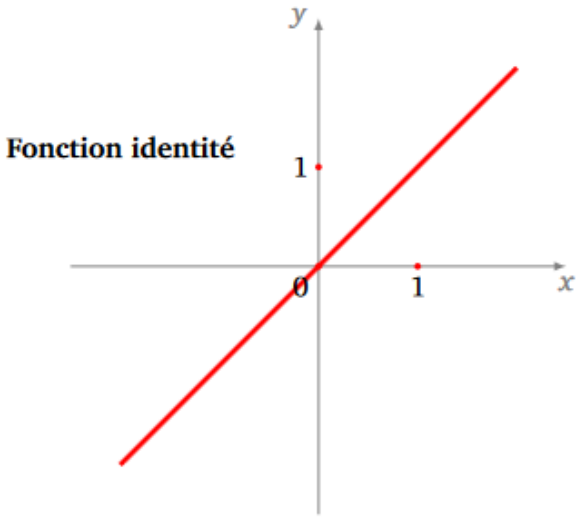
1986 : *David **Rumelhart**, Geoffrey **Hinton**, Ronald **Williams**, "Learning representations by back-propagating errors".* La preuve de l'utilisation de l'algorithme de rétropropagation pour l'entraînement de réseaux de neurones multi-couches.

2018 Prix Alan Turing : Yann **Le Cun** , Yoshua **Bengio** , Geoffrey **Hinton**

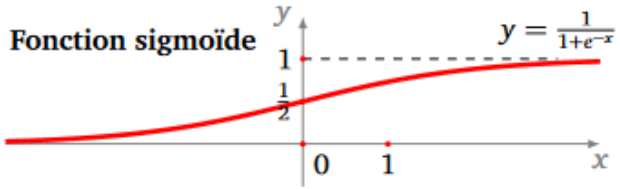
- Notons que **y = f(x,w,b) : le perceptron conduit à une fonction, x variables, w et b paramètres à adapter ! σ restera fixe.**

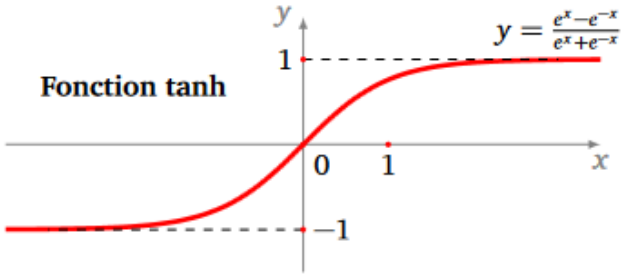
Les fonctions d'activation





La **sigmoïde** est **indéfiniment dérivable**. Valeurs entre 0 et 1. Elle est très utilisée.
La fonction **RELU** (Rectified Linear Unit ou unité linéaire rectifiée), elle permet tout simplement de **remplacer les résultats négatifs par zéro**. Elle est aussi très prisée.

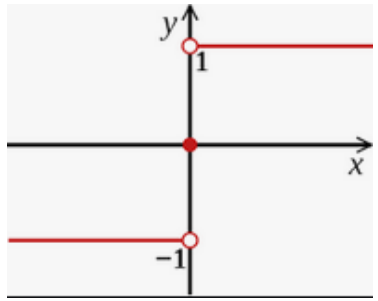




http://exo7.emath.fr/cours/livre-deepmath.pdf page 187

Les programmes pourront introduire aussi des fonctions qui transforment des nombres en **probabilités** ou **pourcentages**. Exemple :
5, 10, 15, 20 ont pour somme 50. En termes de probabilités, ils deviennent : 5/50, 10/50, 15/50, 20/50 soit
0,1 0,2 0,3 0,4 (somme = 1) ou
10 20 30 40 % (somme = 100)
La fonction Softmax traite ces diverses catégories.

Heaviside : elle fût la première fonction d'activation, puis abandonnée : discontinue !



Au sujet des premiers modèles de neurones

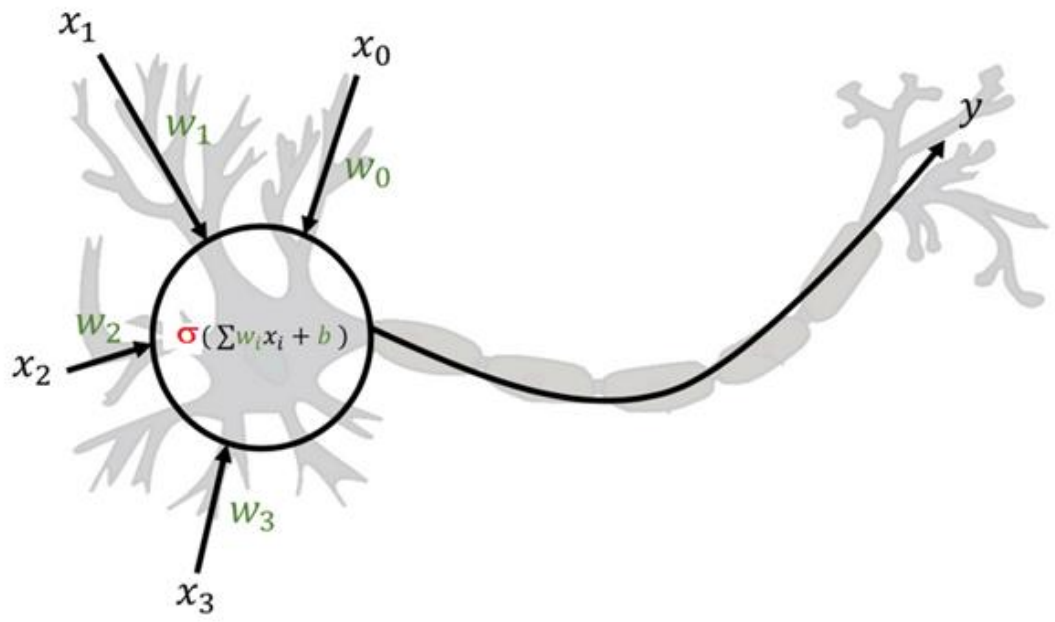
- Au début, le modèle fonctionnait avec des **nombres binaires** 0 ou 1.
- Plus tard, on généralisera aux réels.
- On utilisait la fonction de Heaviside ($\sigma(t) = 1$ si t est positif ou nul, $\sigma(t) = 0$ si t est négatif).

- La linéarité : beaucoup de phénomènes naturels sont non-linéaires. Avec les diverses **fonctions d'activation σ, on a introduit de la non-linéarité**. Mais les calculs préalables sont linéaires, économie de temps !

- Pour la partie linéaire, on utilise des cartes spéciales pour accélérer les calculs sur l'ordinateur !

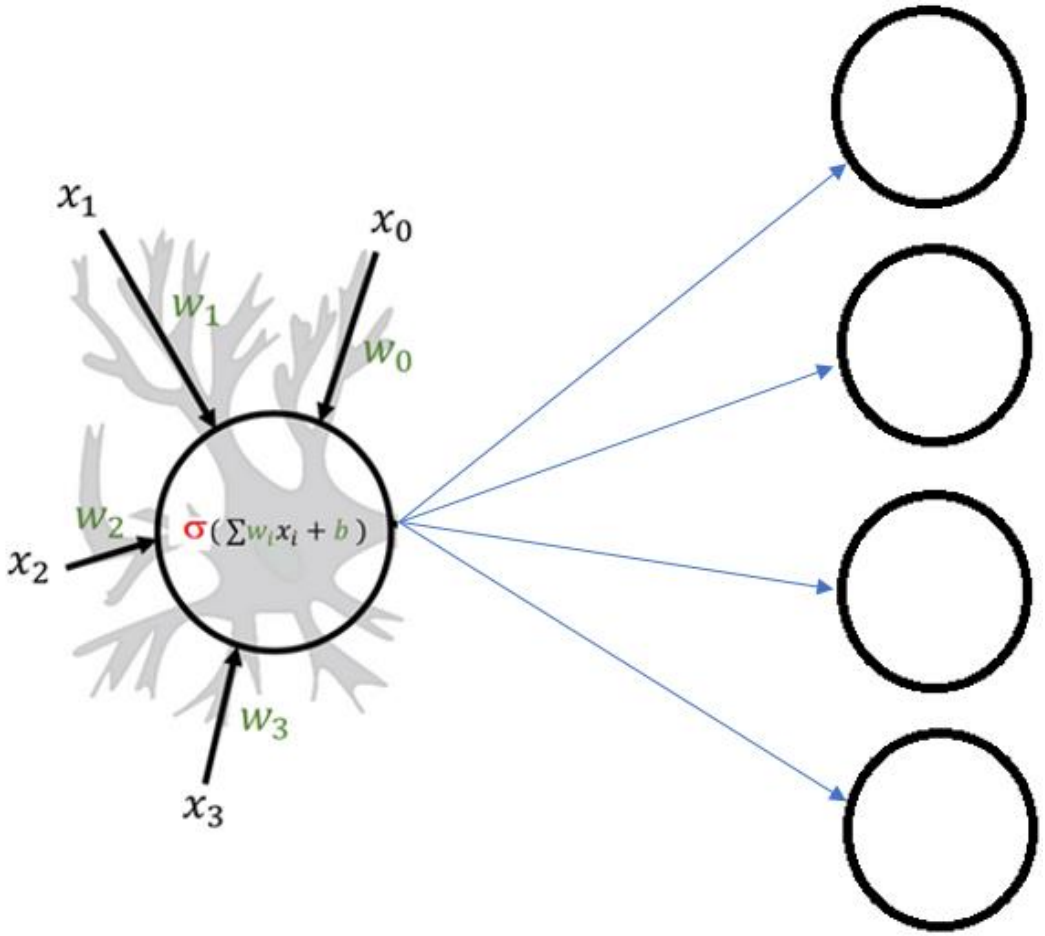
Connexions entre plusieurs neurones modélisés

Figure 9. Schéma d'un neurone informatique superposé à un schéma de neurone biologique



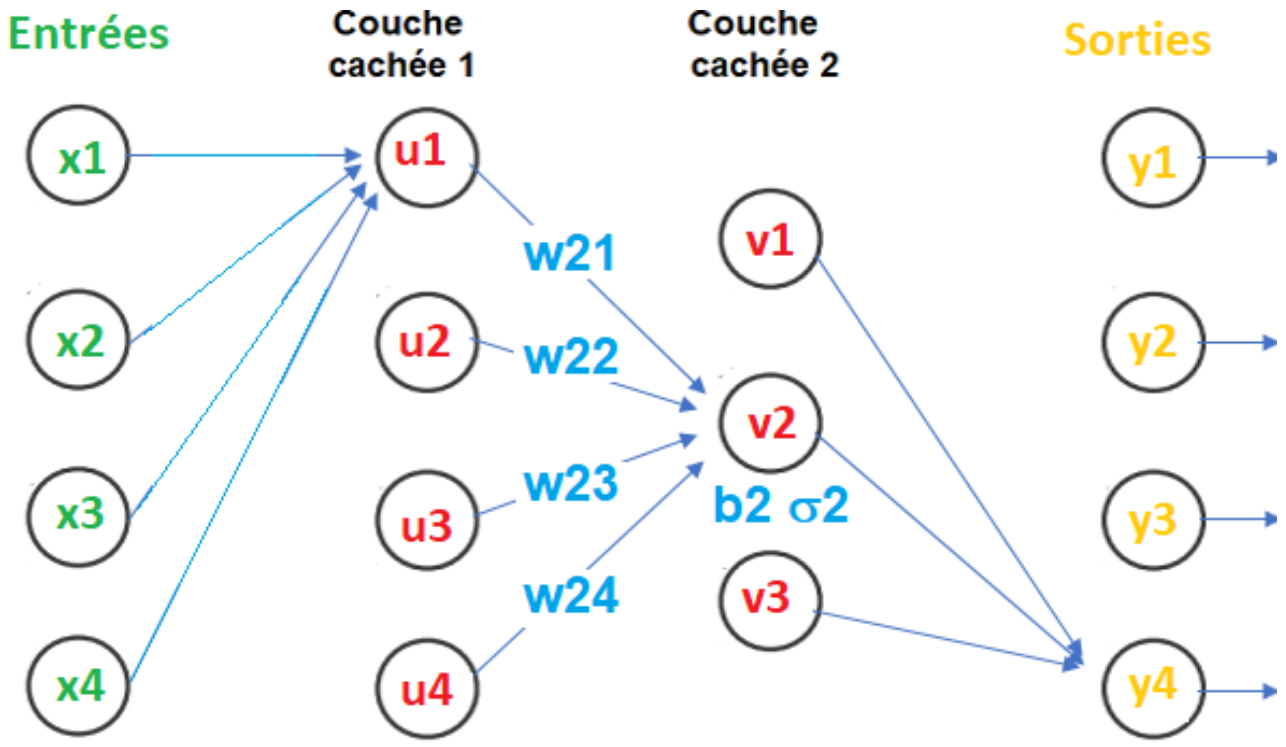
Les variables écrites en vert sont les paramètres du neurone. Source de l'image du neurone biologique : Wikipedia.

--> La fonction σ est une fonction d'activation



On supprime l'axone du neurone de gauche, on le remplace par des liens directs avec les autres neurones.

Un réseau de neurones : une fabrique de fonctions (I)



Les données (ou entrées) sont en vert.

Les paramètres en bleu.

Les résultats intermédiaires de calculs en rouge.

Les sorties en jaune.

Exemple de calcul en 2 étapes : v2 ?

1. a = w21.u1 + w22.u2 + w23.u3 + w24.u4 + b2
2. v2 = σ2(a) σ2 est la fonction d'activation du neurone 2 de la couche cachée 2

- Les sorties y dépendent de x et des paramètres w et b. Elles dépendent aussi des fonctions σ, mais ces fonctions seront fixes tout au long du procédé d'apprentissage et d'exploitation du réseau.
- On a construit une nouvelle fonction :

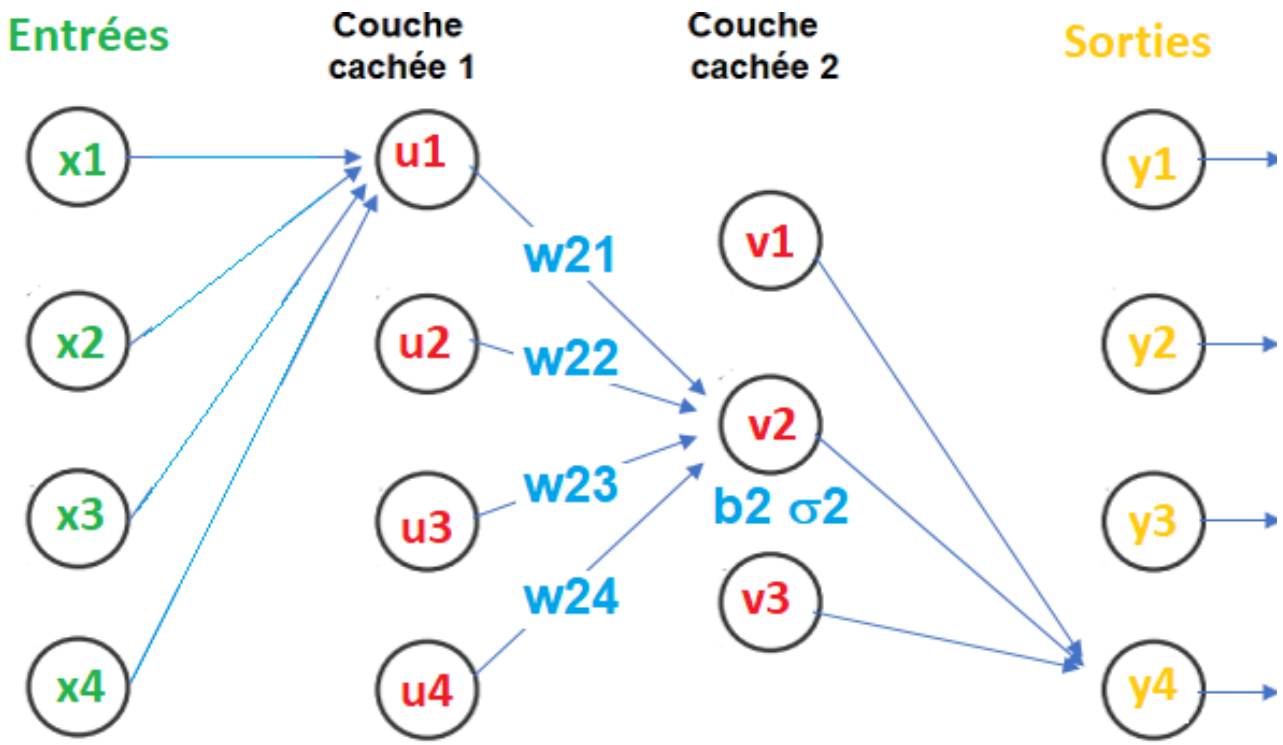
$$y = F(x, w, b),\ x \in R^4,\ y \in R^4$$

Exemples de paramètres : w21 = 3,2 W22 = 5,1 … b2 = 3 σ2 = ReLU

On a construit un réseau de neurones ou de perceptrons, avec dans notre exemple, les couches des entrées et des sorties, et deux couches dites cachées. *Le schéma n'est pas complet* : il faut ajouter toutes les liaisons (les flèches pour lier 2 ronds (les neurones !) dans 2 colonnes successives), présenter tous les paramètres (poids w et biais b), les diverses fonctions d'activations σ et faire les calculs semblables à celui de l'exemple avec v2.

- *Notons la propagation de la gauche vers la droite !*

Un réseau de neurones : une fabrique des fonctions (II)



- On a construit une nouvelle fonction :

$$y = F(x, w, b),\ x \in R^4,\ y \in R^4$$

- Les couches cachées peuvent avoir des nombres de neurones différents, elles peuvent dépendre de très nombreux paramètres. On peut ainsi construire des fonctions du type :

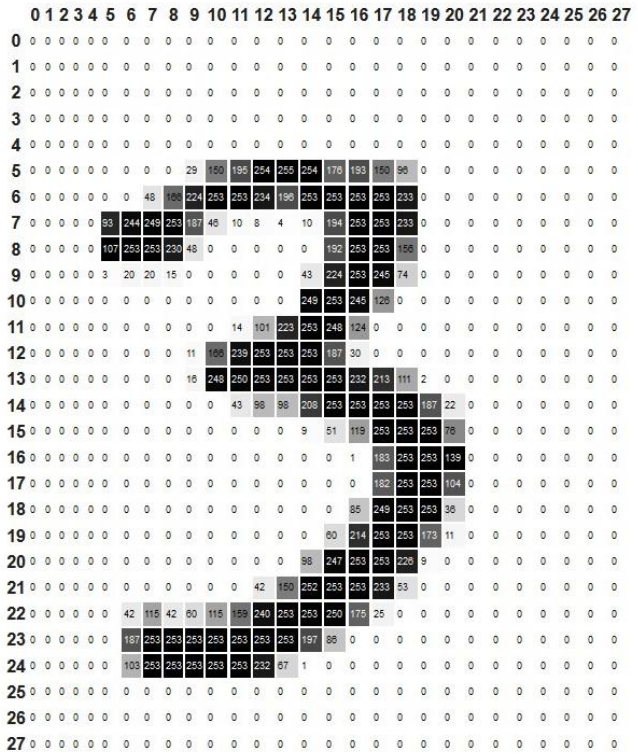
$$\boxed{y = F(x, w, b),\ x \in R^p,\ y \in R^q}$$

p et q nombres entiers positifs, avec autant de paramètres w et b que l'on veut.

Imaginez des xi pour i de 1 à 600.000 (comme les pixels d'une image) ! ***Imaginons 2 couches cachées avec 1.000 neurones chacune, puis 10 sorties y. Comptez le nombre de paramètres !***

4. L’apprentissage supervisé

La reconnaissance des chiffres : l’apprentissage profond



L’image a été créée à l’aide du code fourni dans le livre : Deep Learning for Coders with fastai and PyTorch

https://ichi.pro/fr/pixels-la-base-de-la-vision-par-ordinateur-151661518724346

- ***Si on donne à un ordinateur une image de 784 pixels, si on lui demande quel chiffre elle représente, évidemment, l’ordinateur ne saura pas répondre.*** ***Répétons :***

➢ **L’ordinateur doit au préalable apprendre à reconnaitre les chiffres**.

- ***On pourrait analyser chaque chiffre, déterminer quelles sont ses caractéristiques et exploiter toutes les conclusions en un programme long, complexe et incertain ! Mais cela sera compliqué car la saisie des chiffres est manuscrite et il y a des variations énormes entre elles. Autant de personnes, autant de 0, 1, 2, 3… manuscrits.***

➢ **Un autre système s’est révélé très efficace : le « *Deep Learning* » ou Apprentissage profond avec plus particulièrement l’utilisation des réseaux profonds de neurones. C’est une généralisation de la régression.**

La reconnaissance des chiffres

- Au début, il y avait des difficultés, car **une image, c'est deux choses à la fois** :

  - La liste des pixels *(colorés en Noir et Blanc ou en couleurs Rouge, Vert, Bleu), par exemple* $\boldsymbol{x \in R^{784}}$**.**

  - Le nom du chiffre représenté : *c'est un 0, ou bien un 1, ou bien un 2…*

- ***On a donc introduit la notion d'étiquette ou de classe associée à une image : ainsi, les 60.000 images*** des chiffres 0, 1, 2 … 9, manuscrits de la base de données **MNIST** ont été étiquetées « **à la main** ». Ces images sont partagées en **10 classes** : '0', '1'… '9'.
- La classe **' i '** contient toutes les images qui expriment le chiffre **' i '**, i de 0 à 9.

L'apprentissage

- Nous savons comment créer une fonction avec un réseau de neurones :

$$y = F(x, w, b),\ x = (x_1, x_2, x_3 \ldots x_p) \in R^p,\ y = (y_1, y_2, y_3 \ldots y_q) \in R^q.$$

- Si maintenant, **l'entrée x est une image représentée par l'ensemble des niveaux de gris des pixels d'un chiffre manuscrit quelconque écrit en noir et blanc, alors nous avons $x \in R^{784}$**. Quelle sera **la sortie y ? Quel est notre but ?**

➢ **A partir de l'image pixélisée, on souhaite trouver la classe à laquelle appartient l'image, elle indique le nom du chiffre : '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'.** C'est la sortie **y** qui doit nous permettre d'atteindre ce nom. Comme nous ne sommes pas sûrs d'annoncer à 100 % « la classe est '2' » ou « bien '7' »…, nous utiliserons des probabilités : **Y contiendra *10 probabilités* : $y \in R^{10}$** (10 parce que 10 classes)

➢ Voici un exemple de réponse, de **y calculée via F** :

$y = (0{,}1\ \ 0{,}0\ \ 0{,}6\ \ 0{,}0\ \ 0{,}1\ \ 0{,}1\ \ 0{,}0\ \ 0{,}0\ \ 0{,}0\ \ 0{,}1),\ y \in R^{10}$, la somme des composantes vaut 1, chaque composante est positive ou nulle. Dans cet exemple, la probabilité pour que le chiffre en entrée soit 0 est 0,1 - celle pour que le chiffre soit 1 est 0,0 - celle pour que le chiffre soit 2 est **0,6**… - celle pour que le chiffre soit 9 est 0,1. En termes de probabilités, le nom du chiffre favori est **2**, la plus grosse probabilité : **0,6** ! **Donc y nous propose la classe '2', le chiffre 2 avec une probabilité de 0,6 ou 60% de chances de succès.**

Merci à ReLU (sorties positives) et à la fonction Softmax (sorties du type probabilités).

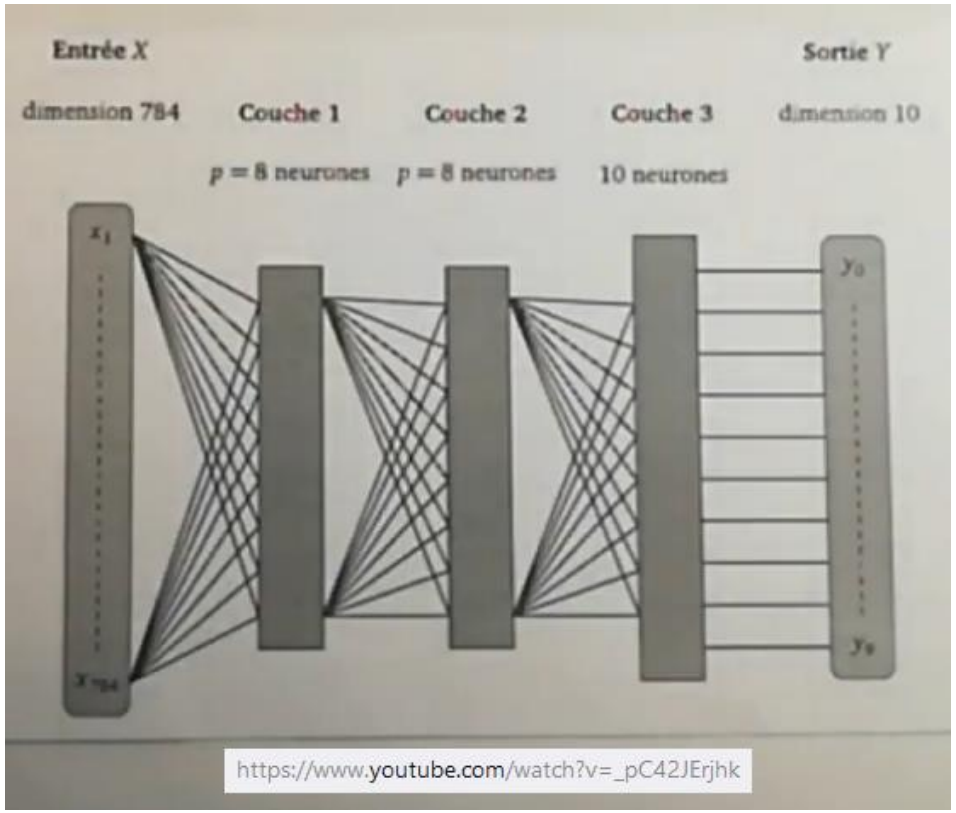
Un réseau de neurones pour reconnaître les 10 chiffres

Il faut donc construire un réseau de neurones qui conduise à

$y = F(x, w, b)$, $x \in R^{784}$, $y \in R^{10}$, $y = (y_0, y_1, y_2, \dots y_9)$, $y_i >= 0\ \forall i$, $\Sigma y_i = 1$

Les entrées et les sorties sont connues, il reste à rajouter quelques couches cachées !

Voici un réseau, vous pouvez en trouver bien d'autres sur l'internet, et pourquoi pas en construire un !



Arnaud BODIN, François RECHER présentent ce réseau et un programme en Python de moins de 60 lignes.

Ils obtiennent des résultats allant jusqu'à 96 % de succès.

Ils utilisent des standards dans l'apprentissage des réseaux :

TensorFlow est un outil open source d'apprentissage automatique développé par Google.

Python, un langage populaire utilisé dans l'apprentissage des réseaux.

Keras est une bibliothèque open source écrite en python.. Elle permet d'interagir avec les algorithmes de réseaux de neurones profonds et d'apprentissage automatique pour simplifier leur utilisation.

Bien entendu, ils ont bénéficié de la base de données **MNIST**.

Le programme est facilement lisible. Il étale sa richesse sur le Deep Learning.

Bravo et merci.

https://www.youtube.com/watch?v=_pC42JErjhk

[Deepmath] 10.1. Reconnaissance de chiffres - tensorflow - MNIST

- ***Combien de paramètres w et b, il y a dans ce réseau ? Comptons !***
- ***784 X 8 + 8 X 8 + 8 X 10 = 8.316 plus les biais.***

La phase Apprentissage *Supervisé* des chiffres

Dans cette phase d'apprentissage supervisé : on montre à l'ordinateur de très nombreux exemples, on va lui soumettre les données d'apprentissage, **60.000 images** des chiffres 0, 1, 2 … 9, manuscrits (c'est la base de données **MNIST**) **avec** leurs **étiquettes**.

Ces images sont **étiquetées**, disons, partagées en **10 classes** : '0', '1'… '9'. La classe ' i ' contient toutes les images qui expriment le chiffre ' i ', i de 0 à 9.

Quand vous envoyez une image de chiffre à un réseau de neurones, vous lui envoyer en même temps son étiquette, le réseau saura de quel chiffre il s'agit.

- **Voila l'image et voici son étiquette qui vous dit que c'est un 0 ou 1 ou 2… Faites en un bon usage !**

➢ **On va chercher des paramètres w et b « optimaux », pour que les étiquettes envoyées « y » soient voisines des étiquettes calculées « F(x, w, p) », et cela pour toutes les images ! Comme pour les régressions !**

La Reconnaissance des chiffres

- Remarque : **on a facilité le calcul des erreurs en donnant le <u>même paramétrage aux classes et aux sorties y</u>.** Exemples : ‘0’ = (**1**, 0, 0 … 0), ‘1’ = (0, **1**, 0 … 0), ‘2’ = (0, 0, 1, 0 … 0) … ‘9’ = (0, 0 … 0 , **1**). Il y a des probabilités… avec une certitude ! Dans ‘2’ la probabilité qu’il s’agisse du 2 est bien 1 !
- Puisque dans la phase apprentissage, le réseau dispose de l’image x et de la classe de x, puis qu’il calcule cette classe avec la fonction F, il pourra se rendre compte <u>qu’il y a une **erreur** entre la **classe donnée et la classe calculée**.</u>
- Pour chaque x, on trouve une erreur $\mathbf{E_x(w,b)}$; on fait ensuite **la somme des erreurs <u>pour toutes les images de la base</u>**. On obtient l’erreur globale : **E(w,b).** Signalons qu’il y plusieurs façons de définir l’erreur.
- Il nous reste à trouver les **paramètres optimaux** w* et b* qui minimisent E(w,b)
  E(w*,b*) <= E(w,b) pour tout w et pour tout b
- **<u>C’est un problème d’optimisation que les mathématiciens savent résoudre</u>**.
- Dans la phase suivante, dite ***phase de tests*** **notre fonction F utilisera w* et b* comme paramètres.**

La phase TESTS

- ***L'apprentissage, c'est la recherche des paramètres optimaux w* et b* pour que E(w,b) soit minimum.*** **Écart minimum entre Classes réelles et Classes calculées.**
- **Une fois ce problème résolu, on se sert de F pour donner le chiffre qui correspond à x, une image manuscrite du chiffre y = F(x, w*, b*),** y sous forme de probabilités ou de pourcentages indique quel est le chiffre **espéré**.

Dans la base MNIST (à côté des 60.000 images pour l'apprentissage) il y a **10.000 autres images pour les tests**.

- **Tester l'algorithme, c'est étudier le nombre d'erreurs que va faire cet algorithme avec ces nouvelles données, (indépendantes des précédentes).**
- ***Si ce nombre est inacceptable, il faudra, soit agrandir la base de données pour l'apprentissage, soit changer de méthode !***

La phase Exploitation

- Après la phase **Apprentissage**, puis la phase **Test**, si les résultats sont acceptables, on passe à la phase **Exploitation** ou **prévision**.

➢ **Pour toutes les images de la base (MINIST), on a minimisé la somme totale des erreurs, si on ajoute une nouvelle image, l'erreur devrait être petite ! C'est là la belle astuce de cette méthode, trouver les paramètres optimaux pour un très grand ensemble des données (ça n'est pas un théorème !)**

- Pour le plus grand plaisir du service **bancaire** ou du service **postal**.
- Le service bancaire va relever automatiquement les **montants** des chèques.
- Le service postal va relever automatiquement les **codes postaux** et faciliter le tri du courrier.

Yann Le Cun (I)

- **Yann Le Cun (1960) est un chercheur en intelligence artificielle et vision artificielle (robotique). Il est considéré comme l'un des inventeurs de l'apprentissage profond. Il reçoit le prix Turing le 27 mars 2019, partagé avec Yoshua Bengio et Geoffrey Hinton.**
- Yann Le Cun est diplômé de l'ESIEE Paris en **1983**, il part ensuite à **l'université Pierre-et-Marie-Curie** effectuer un DEA puis un **doctorat** qu'il soutient en **1987**. Il s'oriente rapidement vers la recherche sur l'apprentissage automatique et il propose pendant sa thèse une variante de l'algorithme de rétropropagation du gradient, très utile pour résoudre *le problème d'optimisation*. Il permet depuis le début des années 1980 l'apprentissage des réseaux de neurones. Il réalise son post-doctorat au sein de l'équipe de Geoffrey Hinton.
- **Yann Le Cun travaille depuis les années 1980 sur l'apprentissage automatique *(machine learning)* et l'apprentissage profond *(deep learning) très à la mode aujourd'hui !*** C'est la capacité d'un ordinateur à reconnaître des représentations (images, textes, vidéos, sons) à force de les lui montrer, de très nombreuses fois.
- En 1987, Yann Le Cun rejoint l'Université de Toronto et en 1988 les laboratoires AT&T, pour lesquels il développe des méthodes d'apprentissage supervisé.
- Yann Le Cun s'intéresse ensuite à la conception des algorithmes de compression du format d'archivage puis la reconnaissance automatique de chèques bancaires.
- Yann Le Cun est professeur à l'université de New York où il a créé le *Center for Data Sciences*. Il travaille notamment au développement technologique des voitures autonomes.

Yann Le Cun (II)

- Le 9 décembre **2013**, Yann Le Cun est invité par **Mark Zuckerberg** à rejoindre **Facebook** pour créer et diriger le laboratoire d'intelligence artificielle **FAIR** (« Facebook Artificial Intelligence Research ») à New York, Menlo Park et depuis 2015 à Paris **(le FAIR-Paris)**, notamment pour travailler sur la reconnaissance d'images et de vidéos. Il avait précédemment refusé une proposition similaire de la part de Google. *Développement considérable à Paris, gros investissements*.
- En 2016, Yann Le Cun est le titulaire pour l'année de la chaire « Informatique et sciences numériques » du **Collège de France**.
- 2020 : Ordre national de la **Légion d'honneur**.
- En avril 2021, il est élu à **l'Académie nationale des sciences des Etats-Unis** .
- Le 22 mai 2017, Yann Le Cun décline une invitation de l'université des sciences et technologies du roi Abdallah en Arabie saoudite, en opposition à une déclaration de 2014 du ministre de l'Intérieur définissant comme terroriste tout athée remettant en question les fondements de la religion islamique. Se considérant comme avocat de l'athéisme, du rationalisme, de l'humanisme, il déclare ne pas pouvoir « accepter l'invitation d'un pays qui [le] voit comme un ennemi ».
- https://fr.wikipedia.org/wiki/Yann_Le_Cun

La reconnaissance des images

- Avec la reconnaissance des chiffres, nous avons appris à reconnaître une forme parmi dix familles de formes différentes, disons 10 catégories, étiquettes, classes.
- **Le traitement d'un ensemble d'autres images est identique**. Traitons un exemple avec seulement (pour être simple) **3 classes**, nous disposons de :

- **Milliers** d 'images de **chats**, ce sera notre 1ère classe, notée **(1,0,0).**
- **Milliers** d 'images de **chiens**, ce sera notre 2ème classe, notée **(0,1,0).**
- **Milliers** d 'images de **lions**, ce sera notre 3ème classe, notée **(0,0,1). Cette dernière classe pourrait être la classe « Autres ».**
- **La dimension des images est de** 469 X 387 = **181.503 pixels**
- **Nous visons la fonction : $y = F(x, w, b)$, $x \in R^{181503}$, $y \in R^3$**
- **Il nous reste à <u>construire</u> le réseau de neurones qui conduit à cette fonction**
- **<u>On recommence exactement le même raisonnement qu'avec la reconnaissance des chiffres.</u> <u>Le nombre de couches cachées fait partie de l'apprentissage des … « humains » (c'est le métier !) et dépend des ordinateurs disponibles !!!</u>**

La reconnaissance des images (résumé)

- **1. On construit le réseau**
- **2. Phase apprentissage** : on envoie au réseau des données d'**apprentissage**, **images et Classes** ou **étiquettes.**
- **3. Le réseau calcule alors l'erreur pour chaque binôme image-classe.** Pour cela, **il calcule y** avec **$y = F(x, w, b)$, $x \in R^{181503}$, $y \in R^3$** puis la différence (y calculée – y envoyée via la classe de l'image). C'est **l'erreur individuelle** pour un x soumis ; en fait, on prend la valeur absolue ou le carré de cet écart. **Ensuite, calcul de l'erreur moyenne globale**, somme des erreurs individuelles divisée par le nombre d'images.
- 4. **Le problème d'optimisation** pour trouver les paramètres optimaux **w*, b*.** Notre modèle est du type $y = F(x, w^*, b^*)$, $x \in R^{181503}$, $y \in R^3$.
- **5. Phase des tests : on passe à l'ensemble des images de tests pour vérifier que le modèle est satisfaisant.**
- **6. Phase exploitation : on soumet de nouvelles images sans étiquette.**

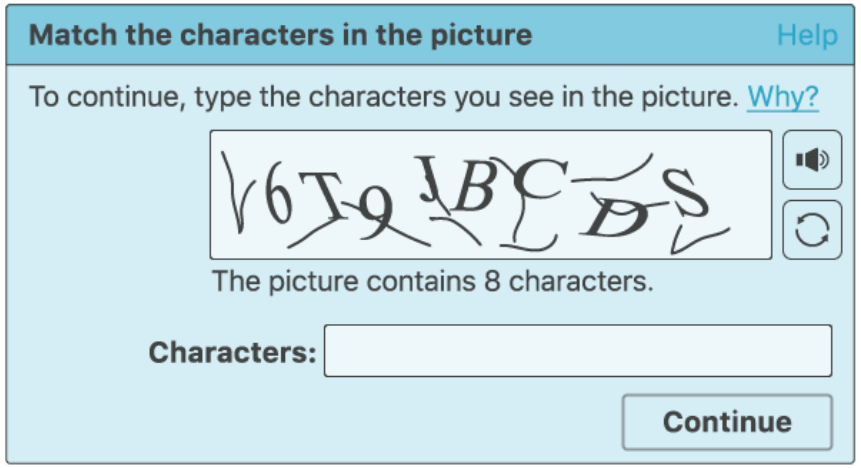
Quelques conclusions

- **Le Machine Learning (apprentissage machine) est un domaine de l'intelligence artificielle dans lequel on crée un système capable d'apprendre de manière autonome à partir d'un échantillon de données d'entraînement. Une fois l'entraînement réalisé et testé, le système est utilisé pour effectuer des prédictions concernant de nouvelles données, jamais observées auparavant.**
- **Cette approche permet de résoudre des problèmes complexes, difficiles voire impossibles à traiter par une approche traditionnelle de la programmation sur ordinateur.**

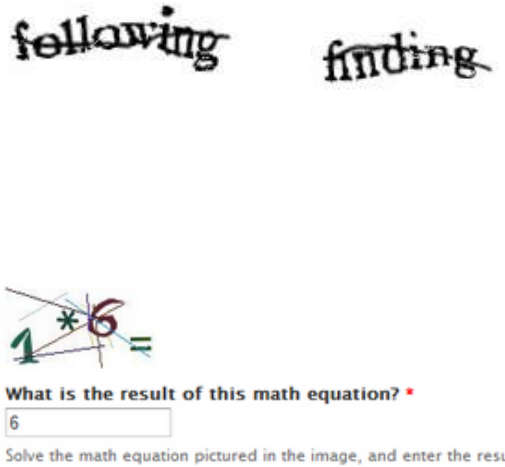
Pour le fUN : les CAPTCHAS contre le Machine Learning

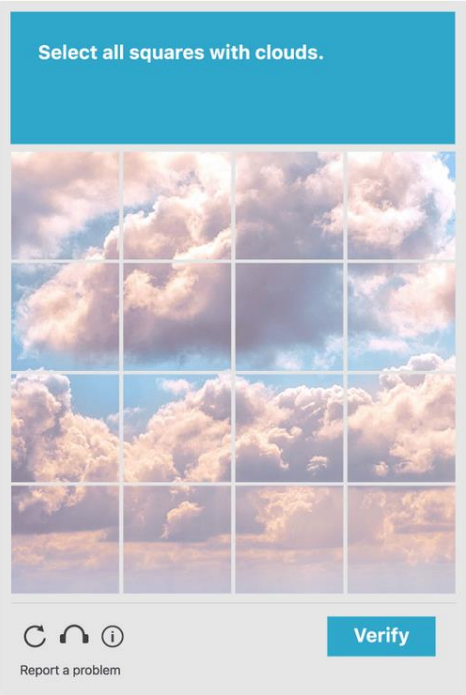
- Le propriétaire d'un site internet ne veut pas que son site soit visité par des robots, par contre il est très favorable à la visite d'êtres humains. Il va afficher un test que seul un être humain peut comprendre et auquel il peut répondre correctement. Un robot se trompera et n'aura pas accès au site. C'est ce que pense le propriétaire du site, mais il peut se tromper.
- **Le mot CAPTCHA recouvre une famille de tests : la vérification du test utilise généralement la capacité d'analyse d'image ou de son de l'être humain**. Un CAPTCHA usuel requiert ainsi que l'utilisateur saisisse au clavier une courte séquence improbable d'une demi-douzaine de lettres et/ou de chiffres visibles sur une image distordue (voir illustrations) qui apparaît à l'écran. Certains sites web préfèrent afficher une image qui contient une question mathématique. D'autres jouent avec des images à choisir.
- **Un robot qui veut visiter un site** doit donc être capable **d'identifier des chiffres** assez tordus, **des lettres** (idem), … **des images.** Ce que l'humain sait faire facilement. **Sinon, il sera interdit de visite ! Mais, attention, le machine learning fait des progrès, et un jour peut être…**
- L'invention des captcha est une des méthodes astucieuses que les data scientists ont inventé pour faire de la labellisation gratuite. Si 90 % des gens disent qu'il y a un feu rouge. Le label va être : il y a un feu rouge. Mais attention : il peut y avoir plus de robots que d'humains à chercher à se connecter.
- (https://fr.wikipedia.org/wiki/CAPTCHA)

Exemples de Captchas



https://www.cloudflare.com/fr-fr/learning/bots/how-captchas-work/





https://www.cloudflare.com/fr-fr/learning/bots/how-captchas-work/

Pour passer le Captcha KittenAuth, il faut cliquer sur trois images représentant un chat. KittenAuth

K Means

5. La convolution

LIMITES DES RN ordinaires

- **Dans les réseaux denses, chaque neurone d'une couche est connecté à tous les neurones des couches suivantes. Le nombre de paramètres peut devenir important dès que la dimension des images est grande. D'autant plus important qu'il y a de couches. Voyons rapidement un exemple : avec une image de 469 pixels par 387 pixels et 3 couleurs (rouge, vert et bleu), ce nombre devient très grand. Imaginons un réseau avec 544.509 entrées (469 * 387 * 3) connectées à une couche cachée de 1000 neurones. Avec la couche cachée, le nombre de paramètres est 544.509 X 1.000 = 544.509.1000 soit plus de 544 millions de paramètres ! Sans compter les couches suivantes et les biais ! Dans les RN, on peut rapidement arriver au milliard de paramètres !**

- **En comparaison, en 2012, AlexNet (Krizhevsky et al., 2017) un réseau de neurones à convolution (Voir Réseau de neurones à convolution) a gagné la compétition Large Scale Visual Recognition Challenge (ILSVRC), une compétition qui existe depuis 2010 mettant à l'épreuve plusieurs algorithmes de reconnaissance d'images (ImageNet, 2017), et le nombre de poids total du réseau était de 60 millions.**

- **Encore plus impressionnant, en 2014, Google avec son réseau GoogleNet a réussi à gagner la compétition en utilisant un réseau composé de 22 couches mais qui contient seulement 4 millions de poids au total (Szegedy et al., 2015).**

- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiws_mexLH0AhWwz4UKHeZXBXEQFnoECBoQAQ&url=https%3A%2F%2Fdoc.rero.ch%2Frecord%2F323738%2Ffiles%2FTDB_BarreiroLindoFl_vio_ReseauxDeNeurones.pdf&usg=AOvVaw2QrXBLUpK1XaQ0PRtJVOOD

➢ **Pour lutter contre la malédiction de la dimension, il a fallu inventer de nouveaux réseaux de neurones !**

➢ **Dans la thèse de Flávio BARREIRO LINDO. Félicitations !**

Sur la convolution

- **La convolution est une transformation mathématique très utilisée**. Elle demande des connaissances assez ardues. C'est une question de professionnels. Heureusement, dans notre cas, on peut y échapper, en montrant ce à quoi conduit cette transformation locale. Avec des petits dessins !

- Une autre façon de la présenter consiste à utiliser des réseaux de neurones très particuliers. On va créer une suite d'images, une par couche. avec des couches qui lient 1 pixel d'une nouvelle image à, par exemple, 9 pixels voisins dans l'image précédente. Une nouveauté : **on utilise toujours les mêmes 9 paramètres**, ceux du noyau**.**
- **C'est là que se fait l'économie de paramètres, c'est fondamental.**

Les réseaux de neurones de convolution

Ces nouveaux réseaux vont nous permettre de lutter, répétons le, contre la malédiction de la dimension des réseaux de neurones classiques. Ils nous permettent un changement de dimensions.

- **Ce qu'on va faire :**

A. On va construire une image en répétant plusieurs fois les opérations suivantes :

1. La convolution
2. Le ReLU (éventuellement intégré à la convolution)
3. Le Maxpooling

B. On envoie ensuite l'image construite, tous les pixels calculés (un point de R^n avec un n convenable) à un réseau de neurones classiques.

C. On exécutera alors la phase **Apprentissage supervisé**. La classification devient, en général plus performante parce qu'avec la convolution, on a récupéré les parties les plus importantes (selon Barreiro Lindo). Puis la phase **Test,** et enfin, on deviendra **opérationnel**.

Les calculs d'une convolution

Image au départ : 6 colonnes et 5 lignes. Image obtenue par convolution avec le noyau : 5 colonnes et 4 lignes.

Le noyau est formé ici d'une tableau 2 X 2, avec les 4 nombres (à choisir) 1, 2, 3, 0. Très souvent, le noyau est 3 X 3.

Image départ

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

X

1	2
3	0

Noyau

Image Convolution

11				

X

1	2
3	0

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

11	17			

X

1	2
3	0

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

11	17	9		

X

1	2
3	0

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

11	17	9	7	

X

1	2
3	0

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

11	17	9	7	13

X

1	2
3	0

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

11	17	9	7	13
8				

X

1	2
3	0

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

11	17	9	7	13
8	15			

On continue !

.....

Image départ

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

X

1	2
3	0

Image Convolution

11	17	9	7	13
8	15	0	6	19
14	4	9	17	7
5	9	9	5	7

Explications des calculs

Image départ

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

X

1	2
3	0

Noyau

Image Convolution

11				

Calculs : on multiplie terme à terme, les nombres du bloc 2X2 de l'image départ par ceux dans la même position dans le noyau.

Exemple : 1 X 1 + 2 X 2 + 2 X 3 + 3 X 0 = 11

On affecte le résultat 11 à un pixel de l'image Convolution

X

1	2
3	0

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

11	17			

On continue à déplacer notre bloc vers la droite et à créer un nouveau pixel de l'image de Convolution

X

1	2
3	0

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

11	17	9	7	13

Quand on est arrivé au bout des lignes 1 et 2, on passe aux lignes 2 et 3. Et ainsi de suite, jusqu'à la fin. A chaque pavé de 4 nombres (comme le noyau), on associe un nouveau pixel de l'image convolution.

X

1	2
3	0

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

11	17	9	7	13
8				

X

1	2
3	0

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

11	17	9	7	13
8	15	0	6	19
14	4	9	17	7
5	9	9	5	7

Passage à 1 dimension

Présentation sous forme d'un réseau de neurones classiques avec la fonction d'activation ReLU

Noyau

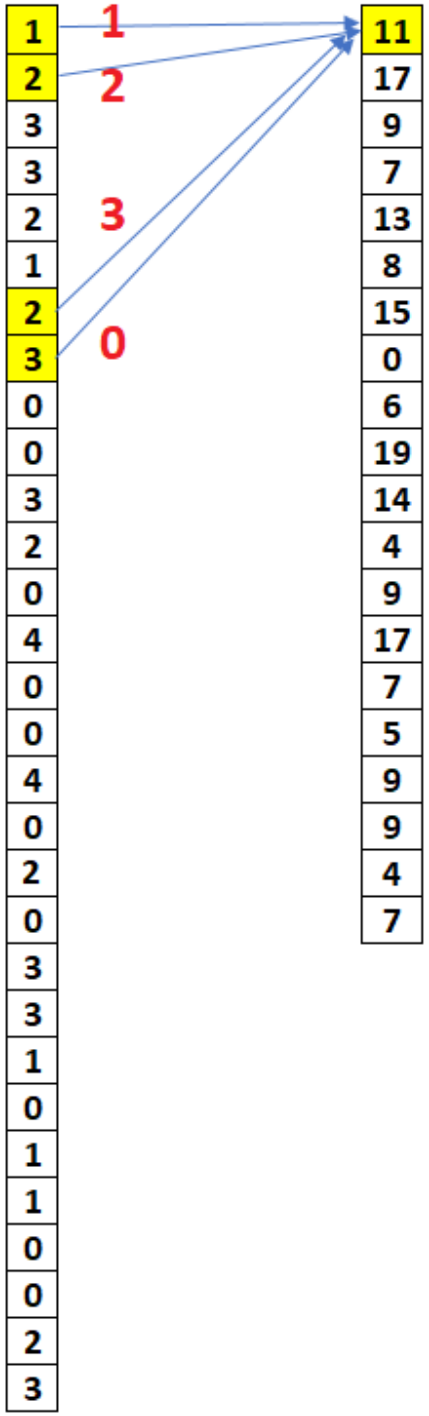
1	2
3	0

Image départ

1	2	3	3	2	1
2	3	0	0	3	2
0	4	0	0	4	0
2	0	3	3	1	0
1	1	0	0	2	3

Image Convolution

11	17	9	7	13
8	15	0	6	19
14	4	9	17	7
5	9	9	5	7



Calculs :

1 X 1 + 2 X 2 + 3 X 2 + 0 X 3 = 11

ReLU(11) = 11

Les poids sont les mêmes pour tous les liens 4 pixels --> 1 pixel

Grande économie

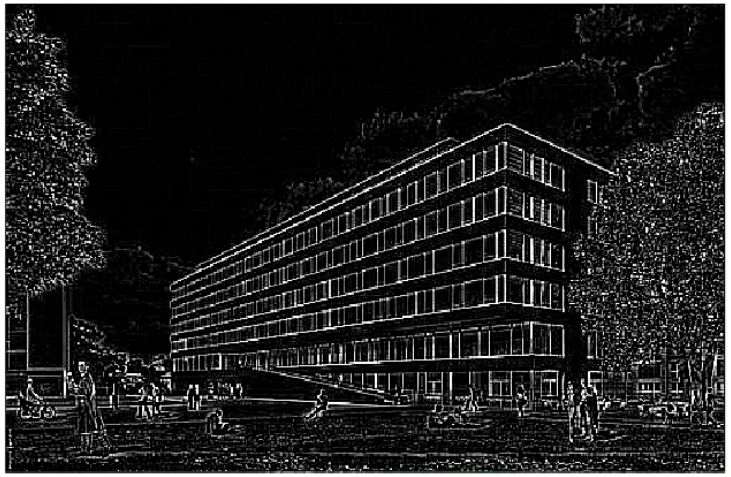
Le réseau n'est pas dense : tout pixel d'une couche n'est pas connecté à un pixel de l'autre couche

Détection de caractéristiques d'une image

Figure 22 : Campus Batelle – Bâtiment B Haute École de Gestion de Genève

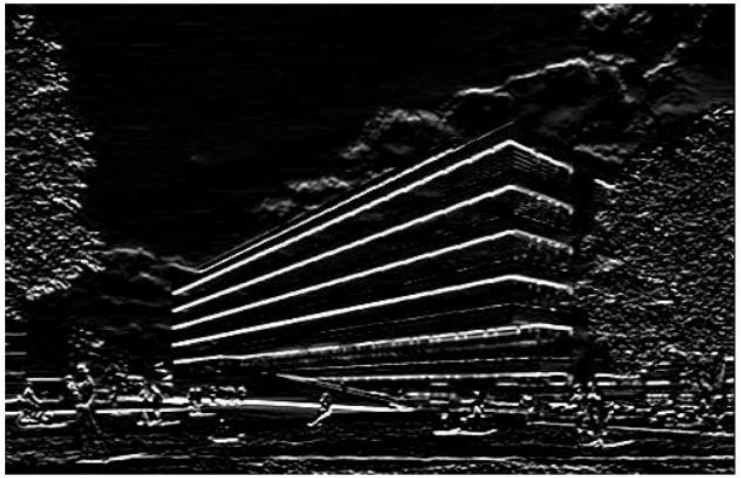
(Genève, 2014)

Figure 23 : Effet de l'application d'un filtre pour détection de bordures



(Genève, 2014)

Figure 24 : Application de filtre pour détection de lignes horizontales



(Genève, 2014)

-1	-1	-1
-1	8	-1
-1	-1	-1

-1	-2	-1
0	0	0
1	2	1

Une même image traitée avec 2 noyaux de convolutions différents : Détection de bordures et détection de lignes horizontales.
La base du convolutif est de préserver la structure d'image.

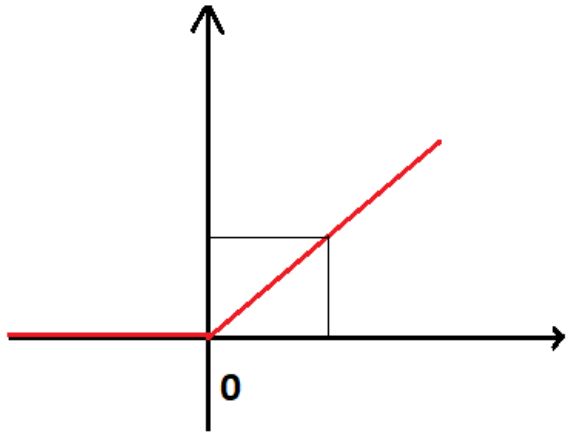
Les images et les noyaux se trouvent dans l'excellent travail de **Barreiro Lindo Flavio**.
Mes remerciements.

TDB_BarreiroLindoFl_vio_ReseauxDeNeurones.pdf
https://fr.scribd.com/document/465996869/TDB-BarreiroLindoFl-vio-ReseauxDeNeurones-pdf

La couche ReLU *(Rectified Linear Units)*

La fonction **non linéaire** ReLU (***Rectified Linear Units***) est définie par :

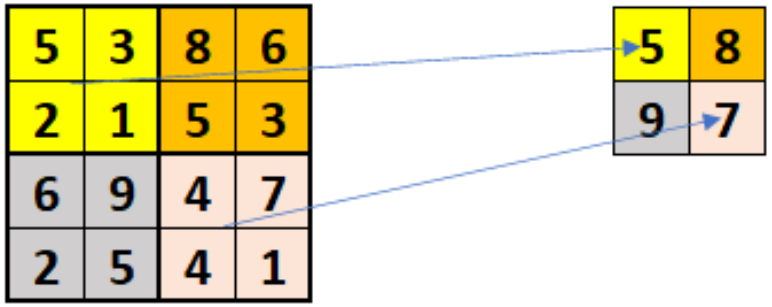
ReLU(x) = max (x,0)



ReLU est la fonction d'activation qui est appliquée ici.

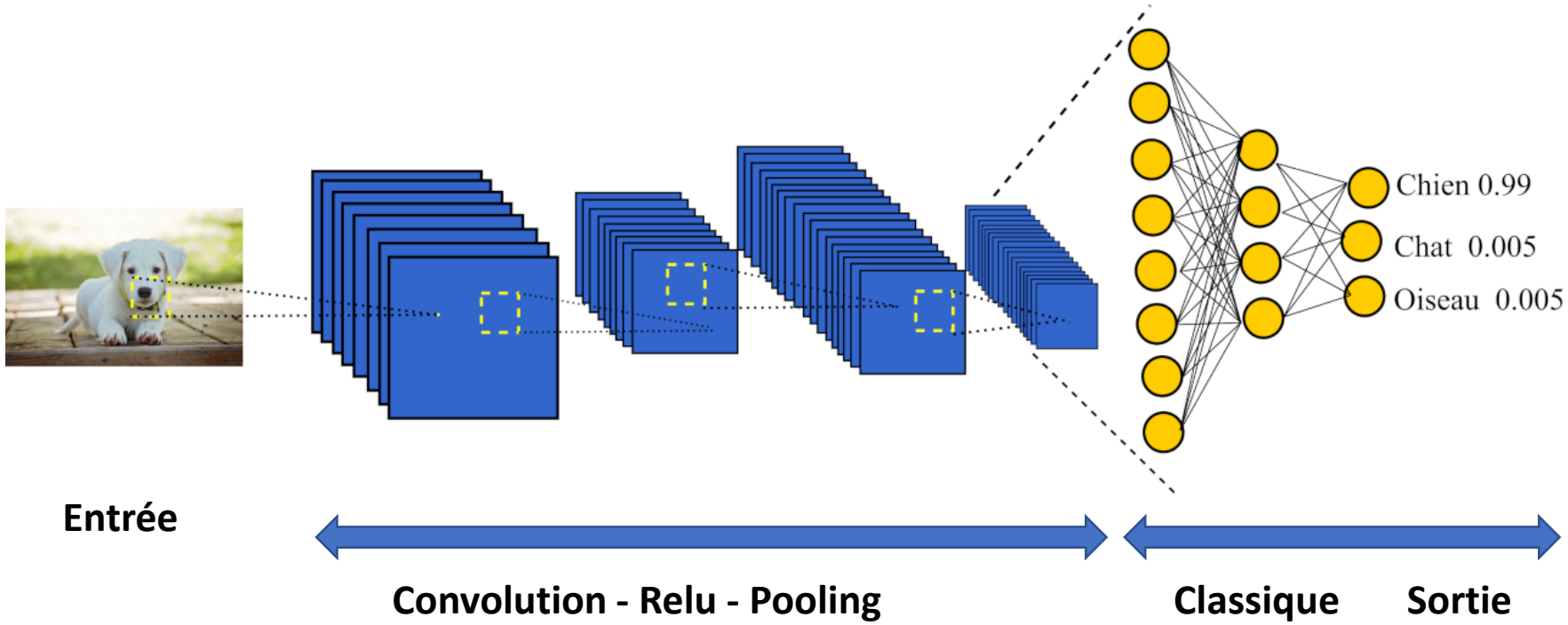
Après avoir appliqué la fonction ReLU à tous les termes utilisés, on ne travaille plus qu'avec des nombres positifs ou nuls, comme le contenu des pixels d'une image. De plus, on retrouve la non-linéarité.

Maxpooling



On partage la matrice (ou le tableau carré) en blocs de 4 cases, par exemple, les 4 cases jaunes adjacentes avec les nombres 5, 3, 2, 1. On retient le maximum de ces 4 nombres, on le reporte sur le nouveau tableau.
Et ainsi de suite : on fait la même chose pour tous les blocs.
L'image initiale est divisée par 4. On garde cependant les caractéristiques essentielles de l'image.

Une mise en œuvre efficace :
Début : un réseau de neurones convolutifs
Fin : un réseau de neurones classiques



- **La convolution va réduire considérablement les calculs !**

https://kongakura.fr/article/R%C3%A9seaux-de-neurones-convolutionnels

6. Apprentissage non-supervisé
K-Moyennes (K-means)

Un cluster, c'est quoi ?

- **On dispose de n observations** ou **points d'un ensemble**. **On veut partager notre ensemble en k sous-ensembles (sous-groupes, grappes) de façon que, dans chaque sous-ensemble les éléments qui le composent aient des affinités, des similitudes, des ressemblances.**
- Un sous-groupe est appelé en anglais « **Cluster** », son centre appelé « **Centroïde** » cristallise les propriétés de ses composants.
- La séparation en clusters se fera en utilisant des **distances**, qui dépendent des applications.
- Par exemple, si on traite des images, en pourra utiliser la notion de distance suivante qui tient compte de <u>la position des points mais aussi des couleurs</u> : un pixel est du type (x, y, r, v, b), les coordonnées des points, les niveaux des 3 couleurs Rouge, Vert, Bleu. La distance entre 2 points P et Q pourrait être :

$$\mathbf{Distance(P,Q)^2 = \lambda.(x^P - x^Q)^2 + \lambda.(y^P - y^Q)^2 + a.(r^P - r^Q)^2 + b.(v^P - v^Q)^2 + c.(b^P - b^Q)^2}$$

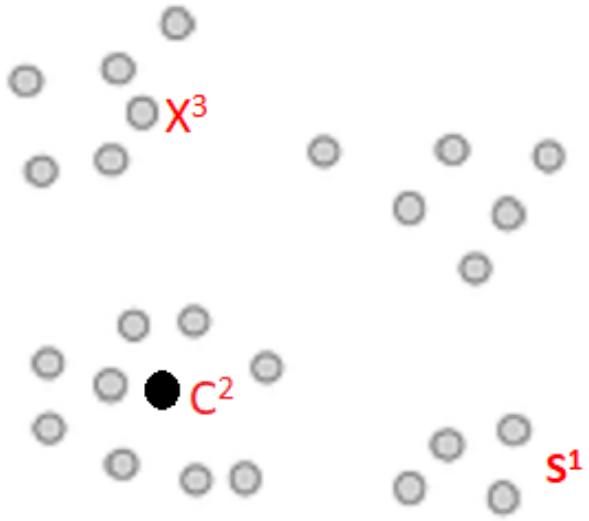
où les 4 nombres λ, a, b, c sont choisis >=0 et dépendent du type de problème traité.

- **Cette séparation en clusters est utilisée en de nombreuses situations, en particulier dans la segmentation d'images.**

K-*means* ou *K*-moyennes

Il s'agit d'une méthode de <u>partitionnement</u> de données.
On donne un ensemble de m points de $\mathbf{R}^n$.
<u>Notations</u> : les points x^i **i** variant de 1 à m. Les composantes des x^i sont les suivantes : $x^i = (x^i_1, , x^i_j , , x^i_n)$. On aura à faire des opérations sur ces points. Ainsi :
$x^p - x^q = (\ldots, x^p_i - x^q_i, \ldots)$
Distance $(x^p , x^q) = \| x^p - x^q \|$ = racine carrée de $(\Sigma \, | x^p_j - x^q_j |^2)$, dans la somme, j varie de 1 à m. Il s'agit de la distance euclidienne entre 2 points de $\mathbf{R}^n$.

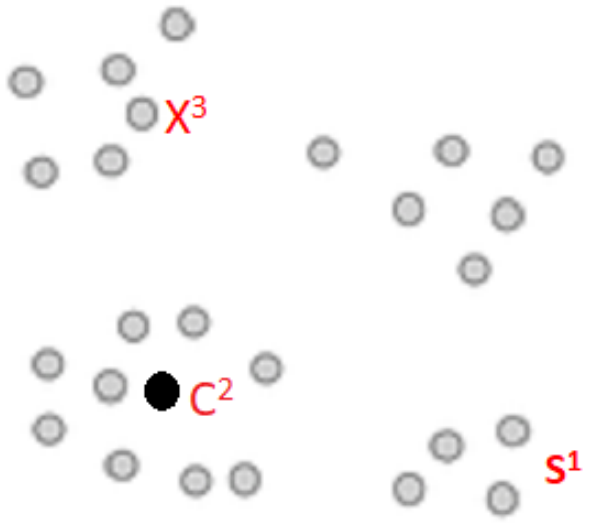
- **On va partitionner l'ensemble des points du nuage en k sous-ensembles notés S^1 … S^j … S^k appelés clusters ou groupes.**



Cela va se faire en plusieurs étapes, d'abord l'initialisation, puis, les itérations :

Commençons par l'initialisation :
1. On va choisir un nombre entier k, on en a l'intuition, ou bien, on le détermine progressivement après quelques essais.
2. On choisit k points appelés centroïdes (initiaux) C^1 , C^2 , C^3 … C^k . Ils sont choisis au hasard ou bien on prend k points du nuage.

K-*means* ou *K*-moyennes



Les itérations

1. Construction des clusters : c'est la **Phase Assignation**. **On dispose des centroïdes. On va affecter chaque point x^i du nuage au centroïde C^j le plus proche**. A la fin de cette phase, tous les points du nuage sont assignés, les clusters sont construits.

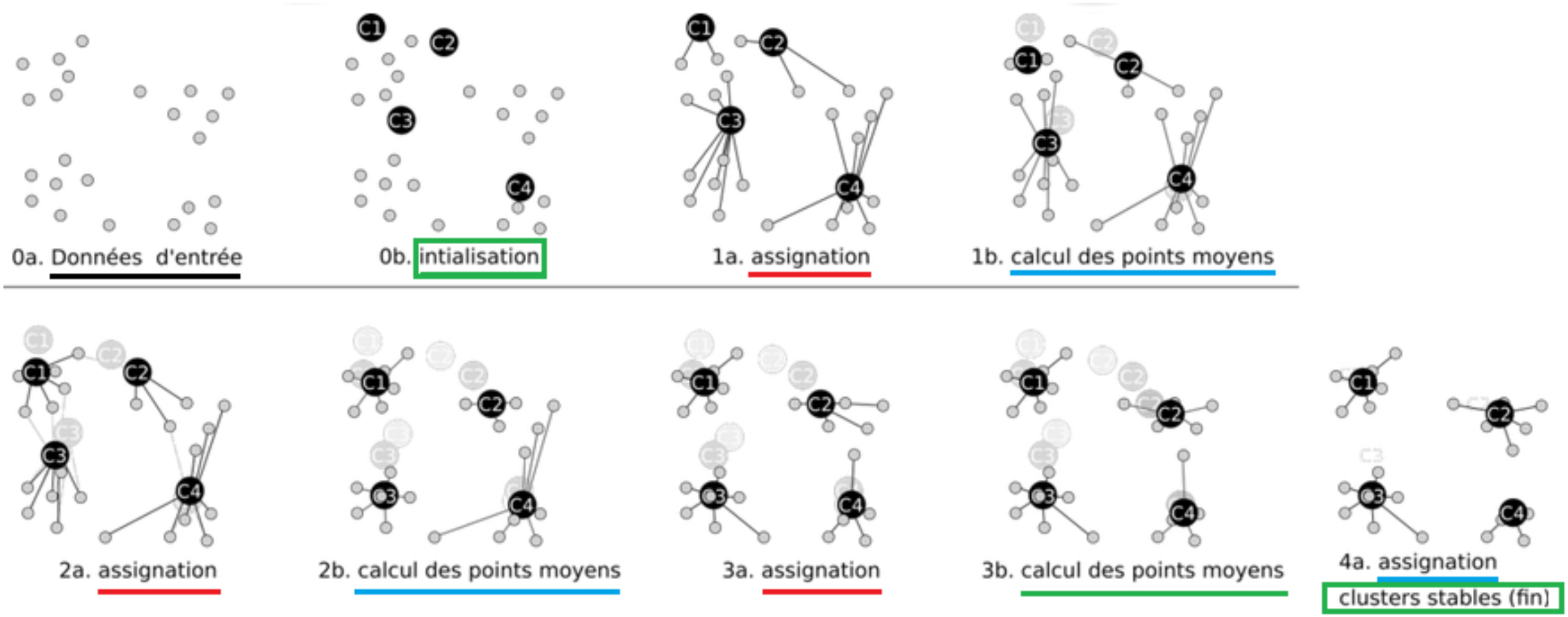
2. Changement des centroïdes : c'est la **Phase déplacement.** On dispose des clusters. A chaque cluster on va associer un **nouveau centroïde** : il s'agit de son « **point moyen** ». Ses composantes sont les moyennes des points du cluster. Exemple : si j'ai 3 points x^p , x^q , x^r **de** $\mathbf{R}^n$ **,** la moyenne de ces 3 points est : (…, $(x^p_i + x^q_i + x^r_i)/3$, …), on fait la moyenne de chaque composante. Idem pour une autre quantité de points. On dispose alors des nouveaux centroïdes.

3. **On recommence une itération (1. et 2.) :** c'est la **Phase nouvelle itération**. Et ainsi de suite. A un moment donné, les centroïdes ne bougent plus. C'est la fin du partitionnement. **STOP**

L'algorithme est une version simplifiée. On peut, par exemple, faire intervenir les couleurs (RVB) :
https://fr.wikipedia.org/wiki/K-moyennes#Algorithme_classique

K-*means* : un exemple de Wikipedia

Quand les points sont donnés dans un plan, il est facile de « voir » combien de clusters il y a, donc, on trouve immédiatement la bonne valeur de k. Dans les autres cas, la situation est bien plus difficile, heureusement, il y a quelques méthodes pour nous aider à trouver une valeur raisonnable de k.



https://fr.wikipedia.org/wiki/K-moyennes#Algorithme_classique

Modification d'une image en utilisant le K-means

Le K-means peut être utilisé dans de nombreuses situations. Ici, nous allons l'utiliser pour **réduire le volume** des informations qui concernent une image et donc pour faciliter les calculs ultérieurs.
Pour calculer la distance entre 2 points P et Q, on utilise la formule suivante :

$$\mathbf{Distance(P,Q)^2 = \lambda.(x^P - x^Q)^2 + \lambda.(y^P - y^Q)^2 + a.(r^P - r^Q)^2 + b.(v^P - v^Q)^2 + c.(b^P - b^Q)^2}$$

où les 4 nombres λ, a, b, c sont choisis >=0

Le choix de ces nombres dépend du problème traité : par exemple, si nous voulons augmenter l'importance de la couleur bleue, nous pouvons prendre un nombre c plus grand que a et b.

A titre d'exemple, que peut-on faire si nous voulons réduire la mémoire consacrée au stockage des 3 couleurs ?

Économie de mémoire

Rappel : **1 octet** = ensemble de 8 bits, il a une adresse physique dans la mémoire. On peut y stocker 2^8 nombres, soit ***256 nombres*** (valeurs en décimales de 0 à 255). C'est le stockage classique pour des **niveaux de gris ou de couleurs R, V, B** de 0 à 255. Il faut **3** octets ou **24 bits** pour stocker les 3 couleurs R, V, B d'un pixel.

- On voudrait **réduire l'ensemble des couleurs à 16**, donc stockables sur 4 bits puisque $2^4 = 16$. Un stockage de 4 bits au lieu de 24 bits ou 3 octets : nous ferons une **économie de mémoire de 6**.

Comment faire ?

On va mettre en œuvre l'algorithme des *k*-moyennes avec **k = 16** et la distance suivante qui prend en charge les couleurs de deux pixels P et Q :

$$\textbf{Distance}(P,Q)^2 = (r^P - r^Q)^2 + (v^P - v^Q)^2 + (b^P - b^Q)^2$$

A la fin de l'algorithme, on associe à chaque pixel de chaque cluster, les 3 couleurs du centroïde de ce cluster. On peut construire la table des 16 couleurs des clusters (numcentroïde, r, v, b) et stocker pour chaque pixel sa position x,y et le numcentroïde de son centroïde. On retrouve ainsi grâce à la table ses 3 couleurs.

http://mpechaud.fr/scripts/kmeans/kimage.html

7. Applications, Éthique, Conclusion, bibliographie

Applications : une petite histoire

- **Applications dans la finance, le marketing, l'industrie, l'Internet des objets, et la vie de tous les jours. Les équipes de statisticiens, d'actuaires et de quants des banques et assurances ont été parmi les premiers à exploiter leurs données avec le Machine Learning afin d'évaluer les risques financiers, de lutter contre la fraude ou pour des applications marketing, la maîtrise de la connaissance clients.**
- **Aujourd'hui les industries françaises s'intéressent de plus en plus à la valeur de leurs données : la maintenance prédictive, l'optimisation des chaînes de production, la construction de véhicules autonomes sont autant d'applications du ML qui pourront générer des gains économiques significatifs.**
- **La reconnaissance d'image avec Facebook, Google, et bien d'autres a été très médiatisée car elle nous touche tous.**
- **Enfin, la médecine(1), la recherche pharmaceutique, la ville intelligente, les transports, le BTP, la modernisation de l'action publique, la cybersécurité et encore bien d'autres domaines tirent déjà des bénéfices notables de l'analyse des données et du Machine Learning.**

(1) Cancers des seins, de la peau… reconnaissance 4 ans avant le spécialistes ! Hum ! Problèmes !

- https://www.quantmetry.com/blog/une-petite-histoire-du-machine-learning/

Éthique (I)

- **Récemment, plusieurs milliers de signatures ont été recueillies sur une lettre ouverte (28 juillet 2015) assez médiatisée, et notamment celles de grands noms de l'intelligence artificielle et du machine learning comme Yoshua Bengio, Yann LeCun, Geoffrey Hinton, Stuart Russell, Peter Norvig.**
- **Cette lettre suit une autre lettre ouverte parue fin 2014 qui, bien que très positive sur les avancées de l'intelligence artificielle et ses bénéfices pour l'humanité, avertissait déjà sur ses écueils potentiels. Ici, les signataires affirment que la construction d'armes autonomes capables de sélectionner et d'attaquer des cibles sans intervention humaine sera possible d'ici quelques années. Faisant part de leur inquiétude et de leur refus de participer à une course aux armements, ils appellent à un bannissement des armes offensives autonomes.**
- https://www.quantmetry.com/blog/une-petite-histoire-du-machine-learning/
- **http://futureoflife.org/AI/open_letter_autonomous_weapons**
- **http://futureoflife.org/AI/open_letter**

Éthique (II)

- **Un dernier aspect, moins souvent médiatisé, est la consommation énergétique de ces méthodes d'IA. Il est désormais connu que le stockage des données sur les data centers représente une part importante (plus grande que celle de l'aviation civile par exemple) de la contribution de l'activité humain à l'effet de serre. Les méthodes d'apprentissage profond qui sont, comme on l'a expliqué, gourmandes en données sont également de grandes consommatrices d'énergies.**
- **Le ML pose donc des problématiques dont il faut se saisir : l'éthique, la protection de la vie privée, la cybersécurité, le coût en énergie et la consommation de matières premières non renouvelables.** https://www.quantmetry.com/blog/une-petite-histoire-du-machine-learning/

Conclusions (I)

- **Depuis 2010, les réseaux de neurones profonds provoquent la même perturbation au sein des communautés informatiques traitant du signal, de la voix, de la parole ou du texte. Une méthode d'apprentissage proposant *le traitement le plus « brut » possible des entrées, évacuant toute modélisation explicite des caractéristiques des données et optimisant la prédiction à partir d'énormes échantillons d'exemples, produit de spectaculaires résultats.* Une manière simple de figurer ce renversement est de le caractériser comme le passage d'une machine hypothético-déductive à une machine inductive (figure 1).**

Extrait de : La revanche des neurones. L'invention des machines inductives et la controverse de l'intelligence artificielle. Dominique Cardon, Jean-Philippe Cointet, Antoine Mazières https://www.cairn.info/revue-reseaux-2018-5-page-173.htm

Figure 1

MACHINE HYPOTHÉTICO-DÉDUCTIVE (1) ET MACHINE INDUCTIVE (2)

— Source : auteurs.

Conclusions (II)

- **Un aspect peu discuté est que le niveau de qualification pour développer ces algorithmes n'est pas forcément très élevé. En effet, les codes des algorithmes d'apprentissage profond sont largement disponibles. Le travail du data-scientist consiste d'abords à se débarrasser d'un travail fastidieux : annoter les données.**
- Il a sa disposition plusieurs méthodes :
  - En général les images médicales viennent avec leurs annotations :
    L'image scanner des poumons vient avec une information supplémentaire, le patient a le covid ou pas.
    Plusieurs images à des intervalles réguliers permettent de déduire l'évolution de la pathologie.
  - Les outils de simulation pour créer des images de synthèse.
- **Il doit aussi régler les paramètres. Le temps de développement est considérablement réduit par rapport aux approches classiques où l'algorithme était programmé. Une solution logicielle peut ainsi être obtenue en moins d'une semaine là où il fallait plus d'un an au début des années 2000**

  https://culturesciencesphysique.ens-lyon.fr/ressource/IA-apprentissage-Rousseau.xml
- **Les pionniers ont vu juste : création d'outils, de bases de données, d'un langage spécialisé et simple… Le tout facilement disponible et gratuit. D'où des développements considérables.**
- **Il en était ainsi au début de la création des sites Internet, il fallait des mois pour créer un site alors qu'aujourd'hui avec les outils existants (et gratuits) vous pouvez faire un petit site dans la journée.**
- **Un autre aspect de la discipline est que le nombre de théorèmes étant faible, l'expérience et le savoir-faire sont importants.**
- **Qu'importe puisque ça marche !**

Merci

FIN

Honneur aux jeunes vulgarisateurs
A chacun sa chaine de vulgarisation !

- **Guillaume Saint-Cirgue, *Machine Learnia***
- **Lê Nguyen Hoang, *Chaine You Tube Scien4All***
- **Arnaud Bodin, François Recher, *Deep Math***
- **David Louapre, *Science étonnante***
- **Thibault Jouannic, *Miximum***

- **Gabriel Peyré, Médaille d'argent CNRS 2021, historique par un mathématicien,** https://mathematical-tours.github.io/book-basics-sources/neural-networks/NeuralNetworksFR.pdf

Bibliographie (I)

MNIST

- https://www.youtube.com/watch?v=_pC42JErjhk
- https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_MNIST

Le Cun

- https://fr.wikipedia.org/wiki/Yann_Le_Cun
- http://yann.lecun.com
- http://yann.lecun.com/exdb/mnist/
- https://www.college-de-france.fr/site/yann-lecun/course-2015-2016.htm

Historique

- https://www.cairn.info/revue-reseaux-2018-5-page-173.htm
- https://interstices.info/la-revolution-de-lapprentissage-profond/
- https://www.quantmetry.com/blog/une-petite-histoire-du-machine-learning/

Bibliographie (II)

Apprentissage supervisé

- http://neuralnetworksanddeeplearning.com/chap1.html
- https://machinelearnia.com/apprentissage-supervise-4-etapes/
- https://www.youtube.com/watch?v=reINnwhqFzY
- http://cedric.cnam.fr/vertigo/Cours/ml2/docs/coursDeep1.pdf
- https://culturesciencesphysique.ens-lyon.fr/ressource/IA-apprentissage-Rousseau.xml
- https://www.cairn.info/revue-reseaux-2018-5-page-173.htm
- http://exo7.emath.fr/cours/livre-deepmath.pdf
- https://fr.scribd.com/document/465996869/TDB-BarreiroLindoFl-vio-ReseauxDeNeurones-pdf
- https://medium.com/@davidroman0O/r%C3%A9seaux-neuraux-pour-les-nuls-a7f5f63b1c10
- https://mathematical-tours.github.io/book-basics-sources/neural-networks/NeuralNetworksFR.pdf
- https://www.cairn.info/revue-reseaux-2018-5-page-173.htm

Bibliographie (III)

Dangers

- http://futureoflife.org/AI/open_letter_autonomous_weapons
- http://futureoflife.org/AI/open_letter
- https://futureoflife.org/background/benefits-risks-artificial-intelligence-french/

Perceptrons

- https://datascientest.com/perceptron

K-means

- https://fr.wikipedia.org/wiki/K-moyennes#Algorithme_classique
- http://mpechaud.fr/scripts/kmeans/kimage.html
- http://www.lama.univ-savoie.fr/mediawiki/index.php/Clustering_par_K-means,_segmentation_d%27image
- https://fr.acervolima.com/segmentation-dimage-a-laide-du-clustering-k-means/

Bibliographie (IV)

IA

- https://www.inria.fr/fr/intelligence-artificielle

Convolution

- https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5082166-quest-ce-quun-reseau-de-neurones-convolutif-ou-cnn

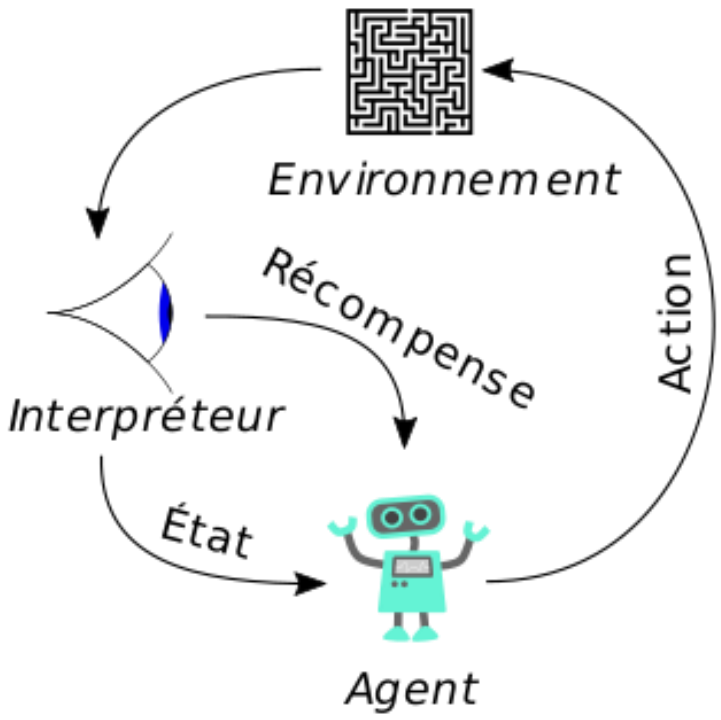
Apprentissage par renforcement

- https://fr.wikipedia.org/wiki/Apprentissage_par_renforcement

Le BigData

- https://www.lebigdata.fr/13-competences-necessaires-devenir-data-scientist

Apprentissage par renforcement (I)



L'**apprentissage par renforcement** consiste, pour un **agent autonome** (robot, etc.), à apprendre les **actions à prendre**, **à partir d'expériences**, **de façon à optimiser une récompense quantitative au cours du temps**.
L'agent est plongé au sein d'un environnement, et prend ses décisions en fonction de son état courant. En retour, l'environnement procure à l'agent une récompense, qui peut être positive ou négative. L'agent cherche, au travers d'expériences itérées, un comportement décisionnel (appelé *stratégie* ou *politique*, et qui est une fonction associant à l'état courant l'action à exécuter) optimal, en ce sens qu'il maximise la somme des récompenses au cours du temps.
L'apprentissage par renforcement est utilisé dans **plusieurs applications** : **robotique**, **gestion de ressources**, **vol d'hélicoptères**, **chimie**. Cette méthode a été appliquée avec succès à des problèmes variés, tels que le contrôle robotique, le pendule inversé, la planification de tâches, les télécommunications, le backgammon et les échecs.

https://fr.wikipedia.org/wiki/Apprentissage_par_renforcement

Quand les pigeons reconnaissent des œuvres de Van Gogh et de Chagall !

Dans un article publié en 2001 par la revue *Animal Cognition*, Shigeru Watanabe explique comment il a entraîné des columbidés à différencier quatre tableaux de chacun des deux peintres, visualisés sur un écran de télévision. En session Van Gogh, les oiseaux reçoivent des graines en picorant un bouton quand une des œuvres du Néerlandais apparaît à l'écran, mais n'ont rien s'ils le font quand un tableau de Chagall s'affiche. Et vice versa en session d'entraînement Chagall.
Une fois que le taux de réussite dépasse les 90 %, le chercheur japonais commence son expérience en intégrant à sa collection trois tableaux (deux de Van Gogh et un de Chagall) que les pigeons n'ont jamais vus. Et les oiseaux n'éprouvent pas de difficulté particulière à les classer correctement.
De la même manière, si les œuvres apparaissent en noir et blanc ou si la moitié du tableau est cachée, leurs performances ne sont pas beaucoup diminuées.

Extrait de : https://www.lemonde.fr/sciences/article/2013/02/28/chagall-van-gogh-et-les-pigeons-experts_1840802_1650684.html

- Watanabe, S. Van Gogh, Chagall et les pigeons : discrimination des images chez les pigeons et les humains. *Anim.Cogn.* 4, 147-151 (2001). https://doi.org/10.1007/s100710100112
- Texte complet dans : https://www-ncbi-nlm-nih-gov.translate.goog/pmc/articles/PMC1334394/?_x_tr_sl=en&_x_tr_tl=fr&_x_tr_hl=fr&_x_tr_pto=sc

Apprentissage par renforcement (II)

- La boucle d'échange scientifique entre **neurobiologistes, psychologues et chercheurs en intelligence artificielle** n'est pas terminée puisque actuellement, des chercheurs prennent inspiration du cerveau pour raffiner les algorithmes d'apprentissage par renforcement et essayer ainsi de mettre au point des robots plus autonomes et adaptatifs que ceux existants.
- La formalisation des problèmes d'apprentissage par renforcement s'est aussi inspirée de théories de psychologie animale, comme celles analysant comment un animal peut apprendre par **essais-erreurs** à s'adapter à son environnement. Ces théories ont beaucoup inspiré le champ scientifique de l'intelligence artificielle et ont beaucoup contribué à l'émergence d'algorithmes d'apprentissage par renforcement au début des années 1980.
- En retour, le raffinement actuel des algorithmes d'apprentissage par renforcement **inspire les travaux des neurobiologistes et des psychologues** pour la compréhension du fonctionnement du cerveau et du comportement animal.
- Dans cet apprentissage, on retrouve tous **les comportements humains** : glouton, calculateur, risque-tout… derrière des théories avec des probabilités et des statistiques.

Le robot Big Dog de Boston Dynamics

Un exemple d'apprentissage par renforcement : l'algorithme est conçu pour chercher à optimiser à tout prix une récompense quantitative, positive ou négative, à partir d'expériences correspondant à différentes situations.

Un exemple des plus actuels et des plus impressionnants d'apprentissage par renforcement est **le robot Big Dog de Boston Dynamics**.

C'est un robot à quatre pattes, capable de marcher, courir, grimper et de porter des charges lourdes.

Le robot est entraîné à marcher **par renforcement : sa récompense est forte et positive s'il reste debout, sa récompense est négative s'il tombe**.

Il est programmé pour explorer les différents mouvements qu'il peut effectuer et faire ses propres expériences. C'est ainsi qu'il apprend seul à monter une pente et à optimiser la manière de poser ses pattes (vitesse, fréquence, angle, etc.) d'une manière très similaire à un animal ou un humain.

https://www.quantmetry.com/blog/une-petite-histoire-du-machine-learning/

https://www.youtube.com/watch?v=wE3fmFTtP9g **(vidéo sur le robot, à voir !)**

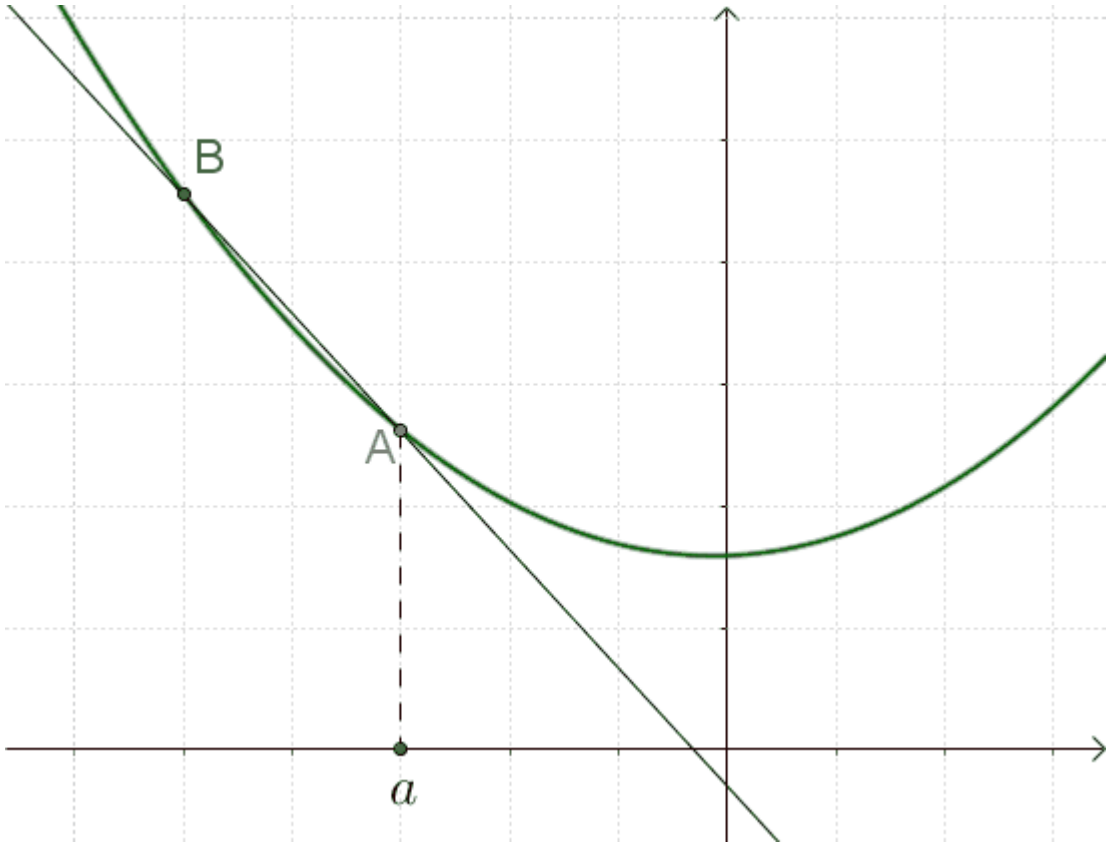
https://www.youtube.com/watch?v=wE3fmFTtP9g

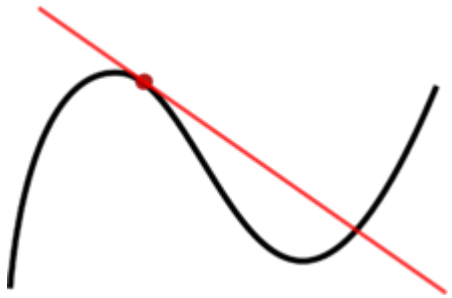
9. Dérivée
Méthode du gradient

Notion de dérivée

- Une voiture quitte Nice en direction de Paris via l'autoroute. Elle part au temps noté 0. Au temps t, la distance parcourue est notée g(t). Regardons la situation entre 2 temps voisins, **t et t+Δt, Δt** assez petit :
- la **distance parcourue** entre ces 2 temps est égale à : $g(t + \Delta t) - g(t)$
- Le **temps écoulé** est : $t+\Delta t - t = \Delta t$
- **La vitesse moyenne est : distance parcourue / temps écoulé : $(g(t + \Delta t) - g(t)) / (\Delta t)$**
- Imaginons plusieurs Δt qui se **rapprochent** de 0, la vitesse moyenne tend à se rapprocher de ce que l'on appelle la **vitesse instantanée** au temps t, **celle qui est affichée au compteur** ! Les mathématiciens ont donné un autre nom à la vitesse instantanée : **la dérivée de la fonction g, notée g'(t) ou dg/dt(t).** On peut donner le nom que l'on veut à la variable t, x, u, v...
- D'une façon générale, on pose : $g'(t) = \lim_{\Delta t \to 0} (g(t + \Delta t) - g(t))/\Delta t$
- Si j'ai besoin d'une dérivée, mais j'ai oublié la table des fonctions avec leurs dérivées (elle est loin la terminale !), qu'est-ce que je peux faire ? C'est simple, je calcule $(g(t + 0{,}01) - g(t))/0{,}01$. Variation de la fonction / Variation de la variable. Si ça manque de précision, j'essaie avec un écart plus petit : 0,001…

Une tangente comme position limite

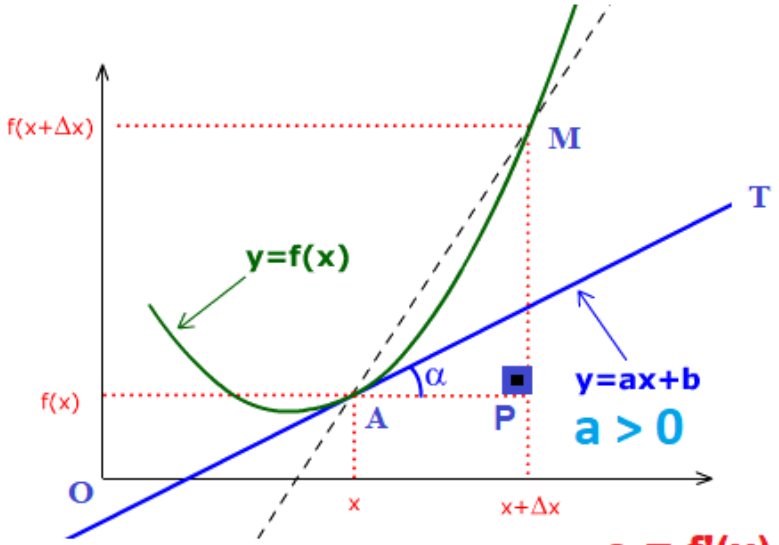


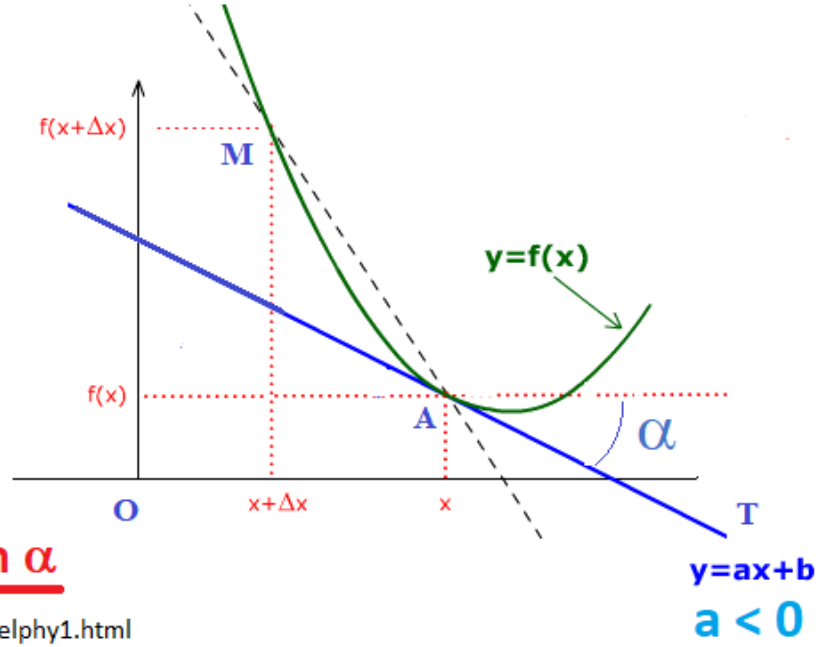


La tangente en un point peut recouper la courbe un peu plus loin.

Selon Wikipedia

Tangente à une courbe et dérivée





a = f'(x) = tan α

http://ww2.cnam.fr/physique/DOCUMENTS/POLYS/ELEMPHY/resumelphy1.html

Triangle MAP : tan (AP,AM) = PM/AP = (f(x + Δx) – f(x))/Δx ⟶ f'(x) = tan (α) = a

Autour de A : si x augmente un peu alors y aussi
F'(x) > 0 : fonction dite **croissante**

Autour de A : si x augmente un peu alors y descend !
F'(x) < 0 : fonction dite **décroissante**

- **Au minimum (ou maximum) on a f'(x) = 0 si la dérivée existe**
- **Près de la courbe, autour du point A, on peut confondre la courbe et la tangente, l'écart est très petit.**

Quelques dérivées

- $g(t) = a\,\mathbf{t} + b \rightarrow g'(t) = a$
- $g(t) = a\,\mathbf{t^2} \rightarrow g'(t) = 2\,a\,t$
- **Une fonction composée** de deux fonctions :

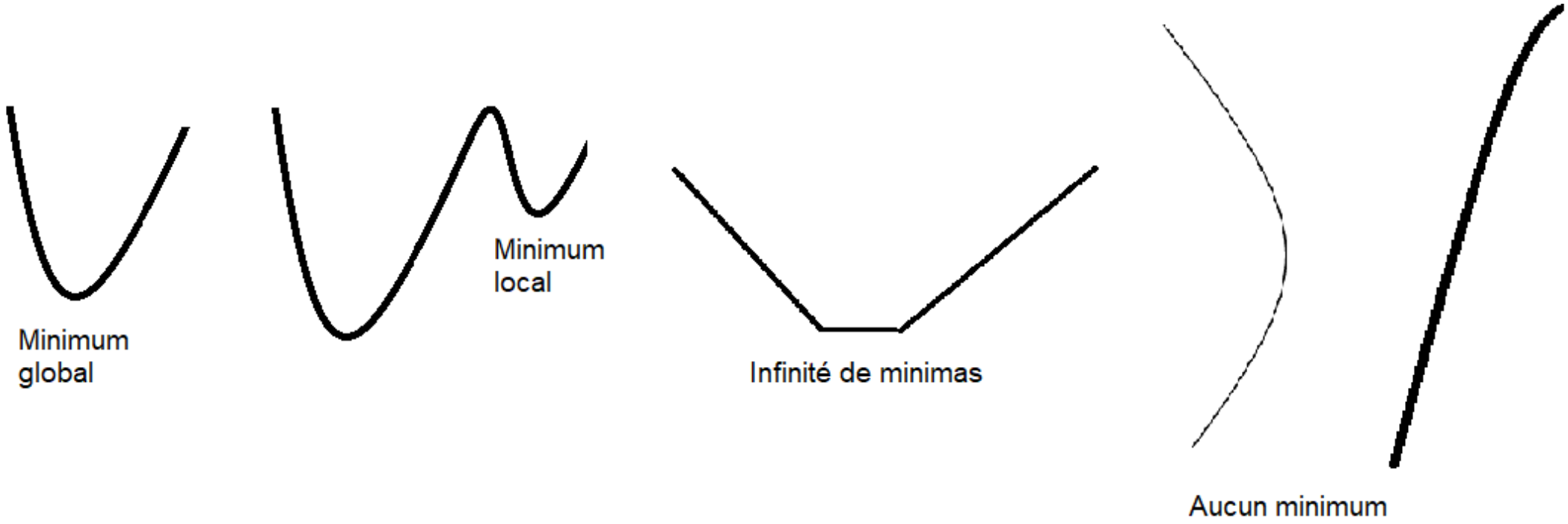
  si $g(u) \rightarrow g'(u)$ $u = u(t) \rightarrow u'(t)$ **alors** $g(u(t)) \rightarrow g(u(t))' = \mathbf{g'_u}(u(t)).\mathbf{u'_t}(t)$
- Exemple :

  $g(t) = ½\,(\mathbf{3t+4})^2$ $u(t) = \mathbf{3t+4}$ $g(u(t)) = \mathbf{½\,u(t)^2}$ $u'(t) = \mathbf{3}$ $\mathbf{g'(u) = u}$

  $g'(t) = \mathbf{g'_u}(u(t)).\mathbf{u'_t}(t) = (3t+4).3 = 9t + 12$

Optimisation : des situations très diverses, encore plus complexes avec davantage de dimensions.

Des problèmes : minimum local, minimum global, Existence, unicité, calcul numérique… (Idem pour maximum).



Minimisation

- Posons $\varepsilon(t, \Delta t) = (g(t + \Delta t) - g(t))/\Delta t - g'(t)$ Nous savons que $\lim_{\Delta t \to 0} \varepsilon(t, \Delta t) = 0$. Cela entraîne :
- $\varepsilon(t, \Delta t) + g'(t) = (g(t + \Delta t) - g(t))/\Delta t$
- $\Delta t . \varepsilon(t, \Delta t) + \Delta t . g'(t) = g(t + \Delta t) - g(t)$
- $g(t + \Delta t) - g(t) = \Delta t . g'(t) + \Delta t . \varepsilon(t, \Delta t)$ et finalement :

$$(*) \quad g(t + \Delta t) = g(t) + \Delta t . g'(t) + \Delta t . \varepsilon(t, \Delta t) \quad \text{avec} \quad \lim_{\Delta t \to 0} \varepsilon(t, \Delta t)$$

Si Δt est « assez » petit, le terme bleu est un infiniment petit par rapport au terme vert, on a l'équivalence suivante :

$$(**) \quad g(t + \Delta t) \sim g(t) + \Delta t . g'(t)$$

Un problème : On cherche le minimum de la fonction g (s'il existe).

On suppose connu g(t), **comment choisir Δt pour que $g(t + \Delta t)$ diminue ?** Il suffit que **$\Delta t . g'(t)$ soit négatif, donc Δt et $g'(t)$ de signes opposés, Δt assez petit. Une façon simple est la suivante : on choisira $\Delta t = - \rho . g'(t)$ où ρ est un nombre fixe positif, assez petit.** $\Delta t. g'(t) = - \rho.g'(t).g'(t) = -\rho. g'(t)^2$

S'il y a plusieurs variables, plutôt que gérer les signes, il est bien plus simple d'utiliser le formule précédente pour chaque variable.

La méthode ou algorithme du gradient

Méthode du gradient :

- on donne t_0 arbitraire, on calcule $g'(t_0)$. Si ce nombre est nul : STOP, f est minimum ou maximum en t_o. Sinon, on continue.

- on construit une suite itérative t_1, t_2… t_n de la manière suivante :

$\mathbf{t_{n+1} = t_n - \rho.g'(t_n)}$ où ρ est un nombre fixe **positif** assez petit (expérience !).

On construit ainsi une suite décroissante.

- **Sous certaines hypothèses, on peut démontrer que le problème d'optimisation a une solution unique et que la suite construite converge vers cette solution.**

- **En réalité, un problème d'optimisation (à plusieurs variables) n'est pas facile à résoudre. Il demande des connaissances mathématiques et du métier pour choisir les paramètres et les tests d'arrêts des itérations.**

Dérivation dans le cas de plusieurs variables

- Supposons que le fonction g dépende de plusieurs variables : $x_1, x_2 \dots x_n$
- **$g(x) = g(x_1, x_2 \dots x_n)$, $x \in R^n$**
- Pour ce genre de fonction aussi, on peut obtenir un résultat du type

$$(*)\quad g(x + \Delta x) = g(x) + \Delta x \,.\, g'(x) + \Delta x \,.\, \varepsilon(x, \Delta x) \quad \text{avec} \quad \lim_{\Delta x \to 0} \varepsilon(x, \Delta x)$$

C'est un peu plus complexe :

$$g(x + \Delta x) = g(x) + \sum_{i=1}^{n} \Delta x_i \,.\, g'_{xi}(x) + \text{infiniment petit d'ordre supérieur…}$$

où $g'_{xi}(x)$ est la dérivée de g considérée comme si elle n'avait qu'une variable : x_i . On note aussi cette dérivée : $g'_{xi}(x) = \delta g/\delta xi(x)$. L'ensemble des n dérivées $\delta g/\delta xi(x)$ constitue le gradient de la fonction g au point x.

Méthode du gradient dans R^n : la même méthode

- on donne $x^0 = (x^0_1, x^0_2 \dots x^0_n)$ arbitraire dans R^n,
- on construit une suite itérative $x^1, x^2 \dots x^k$ dans R^n de la manière suivante :

$$x^{k+1}_i = x^k_i - \rho \,.\, g'_{xi}(x^k) \quad \text{pour tout i}$$

où ρ est un nombre fixe **positif** assez petit (expérience !).

➢ Mêmes conclusions que dans **R.**