l'**Intelligence Artificielle** : une généralisation de l'analyse de données

Assimilation de données : identifier l'état d'un système à partir d'observations
réanalyses du climat, initialisation de prévisions numériques

Apprentissage machine : identifier des relations entre données
classification, prévision statistique

vocab. avant 2010	*après 2010*
statistiques	data science
bases de données	data lake
modèle numérique	digital twin
optimisation, régression	apprentissage
réseaux de neurones	deep learning
traitement d'image	convolutional neural network
filtre numérique	recurrent neural network
traitement de données	intelligence artificielle
modèle statistique	émulateur

l'IA, une révolution technologique

Déf générale : algorithmes qui imitent le cerveau humain
Ce qui nous intéresse ici : classification et apprentissage

Pourquoi cette (r)évolution ? Depuis ~2000 :

- ***plus de données*** collectées : télédétection, objets connectés, sites web & réseaux sociaux
- ***archives*** plus profondes : moyens de stockage, accès public "plus facile"
- ***moyens de calcul*** plus puissants et moins centralisés (supercalculateurs → cloud, PC)
- ***meilleurs algorithmes*** : surtout réseaux de neurones sur processeurs GPU

Compétition actuelle entre modèles numériques à bases physiques & méthodes statistiques conceptuellement pauvres, mais scalables à de gros volumes de données.

Applications phares :

- reconnaissance d'images (=classification)
- autocorrection de texte (=prédiction)
- générateurs d'images et de texte (=classification+analogues)

Applis anciennes en météo/océano/climat :

- identification des types de temps, téléconnexions climatiques (= classification et ACP)
- extrapolation de séquences d'images radar & satellite (= extraction de contours, d'objets, flux optique)
- corrections statistiques des sorties de modèles (= débiaisage, régressions, forêts aléatoires)

Exemples d'applications actuelles de l'IA en météo/océano

Génération de données synthétiques par traitement d'images :

- images satellites de nuages → champs de pluie
- images radar → zones d'orages, de grêle
- images webcam → pseudo-obs de brouillard, de neige au sol

Remplacement de modèles numériques physiques par des émulateurs statistiques :

- émulation de modules coûteux (ex : paramétrisation du rayonnement)
- émulation de modèles complets de prévision météo (apprentissage de la fonction xa → xb)

Post-traitement de modèles :

- identification d'orages violents dans champs de modèle numérique
- apprentissage de structures fines à partir de prévisions basses résolution ("downscaling")

Principes généraux :

- identification d'analogues et d'un sous-espace utiles dans des archives d'apprentissage
- corrections statistiques par régression

Limitations structurelles:

- produits rarement explicables
- dépendance aux données d'apprentissage : pb pour événements extrêmes et changement climatique

Exemple du clustering (classification automatique non-supervisée)

But : découvrir des structures dans les données (= des liens entre elles)

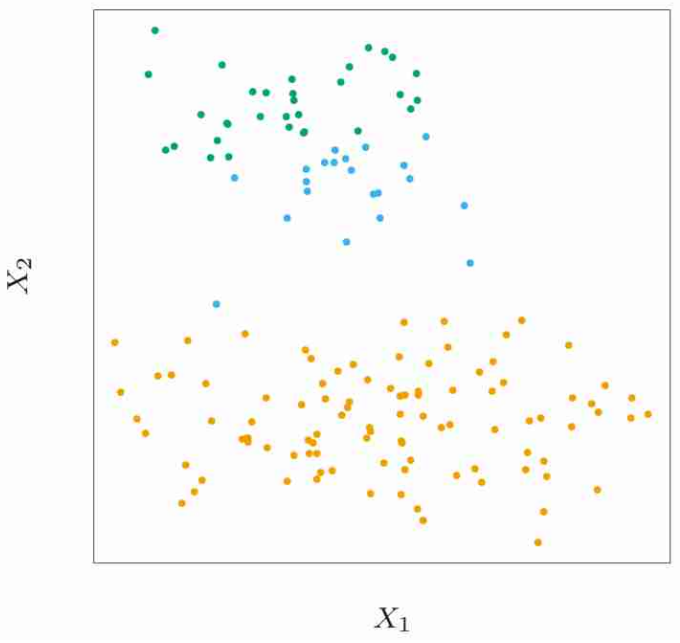
Déjà vu dans ce cours :

- diagnostics de corrélations
- ACP : analyse par composantes principales = réduire la dimension de l'espace des données

Méthodes de classification :

- ***classification hiérarchique :*** grouper des points (= données) en commençant par les plus proches
- ***K-means :*** diviser un nuage de points en *n* sous-parties
- ***DBSCAN***... (pour données très structurées)

couleurs = classes



Exemple du clustering : classification hiérarchique

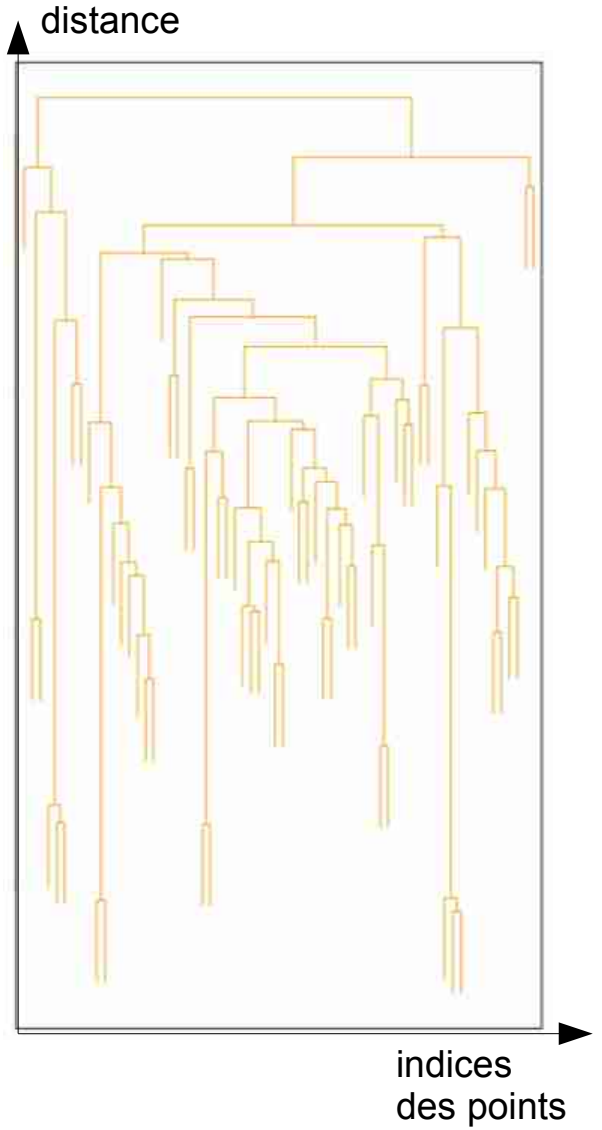
Entrée : n points dans un espace de dimension p

Prérequis :

- si tous les points sont dans un petit sous-espace, se projeter dedans
- définir une fonction distance entre tous les points ou groupes de points
  ex : moyenne des centres de gravité

Algo :

1. trouver les 2 (groupes de points) les + proches
2. les fusionner en 1 seul
3. calculer un critère de qualité pour arrêter l'algo (ex : Nb de groupes)
4. sinon, recommencer



Apprentissage supervisé : principe

Généraliser à partir d'exemples une relation entre données d'entrée (prédicteurs) et de sortie (prédictands, labels), en 2 étapes

1) **apprentissage :** optimisation de la fonction y=f(x) à partir de données d'apprentissage (xi,yi)

2) **inférence :** pour n'importe quel autre x, on "prédit" que y = f(x)

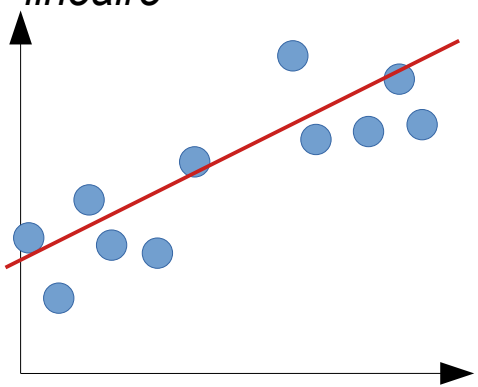
La fonction f optimisée s'appelle le **modèle** de la relation xi -> yi

Les algos d'assimilation en sont des cas particuliers :

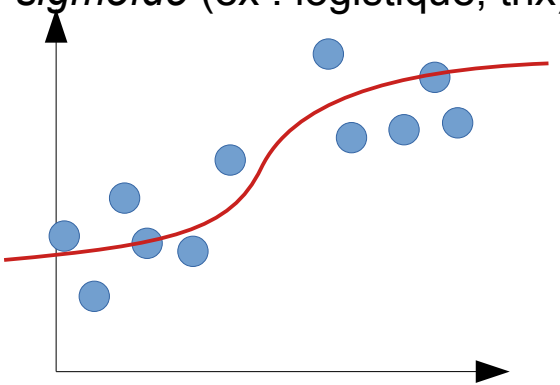
- **interpolation, analyse :** obs y → x analyse, càd champs à prédire
- **régression :** idem, en restant dans une famille de fonctions : obs y → coeffs de la fonction

Ex de régressions :

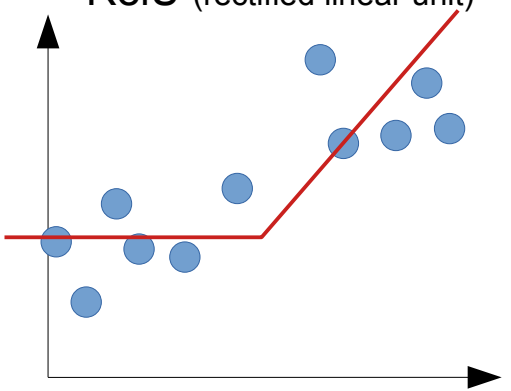
linéaire



sigmoïde (ex : logistique, thx)



ReIU (rectified linear unit)



Apprentissage supervisé et modélisation par analogues

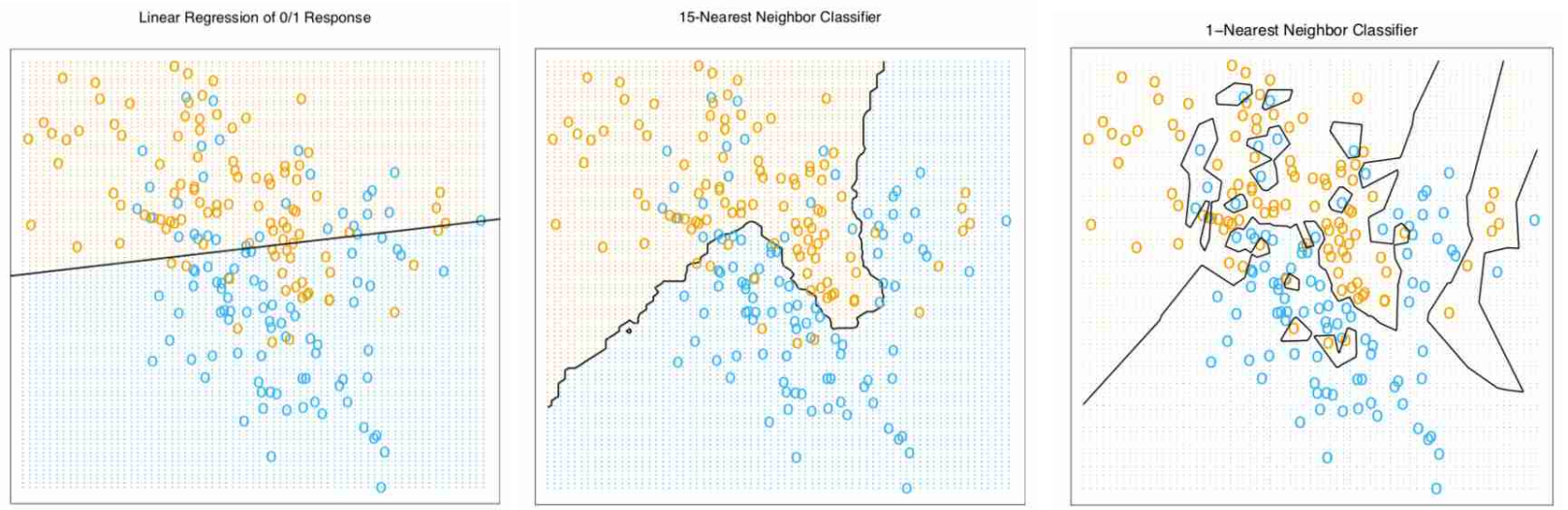
IA générative : prédictands y de grande dimension

Méthode des analogues : "*les mêmes causes produisent les mêmes effets*"

Si on a un très grand historique d'événements statistiquement reliées 2 à 2 : (ai,bi) alors on peut fabriquer un prédicteur empirique a-> b :

- donnée d'entrée : un nouveau a
- on recherche les n ai les plus ressemblants à a dans l'historique : les "analogues"
- on prend les n bi correspondants : c'est un ensemble des b les plus probablement associés à a

Exemple : a = les obs météo d'aujourd'hui, b= le temps qu'il fera demain



(Hastie & Tibishrani)

Apprentissage neuronal (1)

= régression avec plusieurs fonctions connectées entre elles ("neurones")

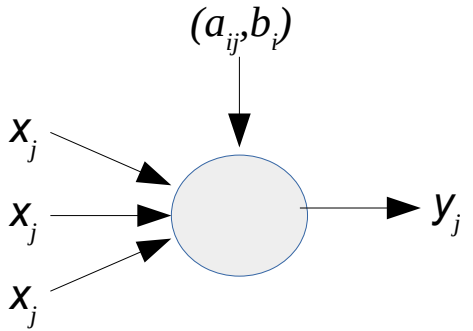
Théorème d'approximation: on peut approcher n'importe quelle fonction par sommation et composition de fonctions élémentaires appelées "neurones"

exemples de fonctions neurones :

- tangente hyperbolique : $f_i(x) = th(a_i x + b_i)$
- Relu :$f_i(x) = max(0, a_i x + b_i)$

Couche (layer) = des neurones en parallèle : $y_i = f_i(\sum_j a_{ij} x_j + b_i)$

On peut enchaîner plusieurs couches : x -> y -> z -> etc.
S'il y a beaucoup de couches, c'est le "**deep learning**"



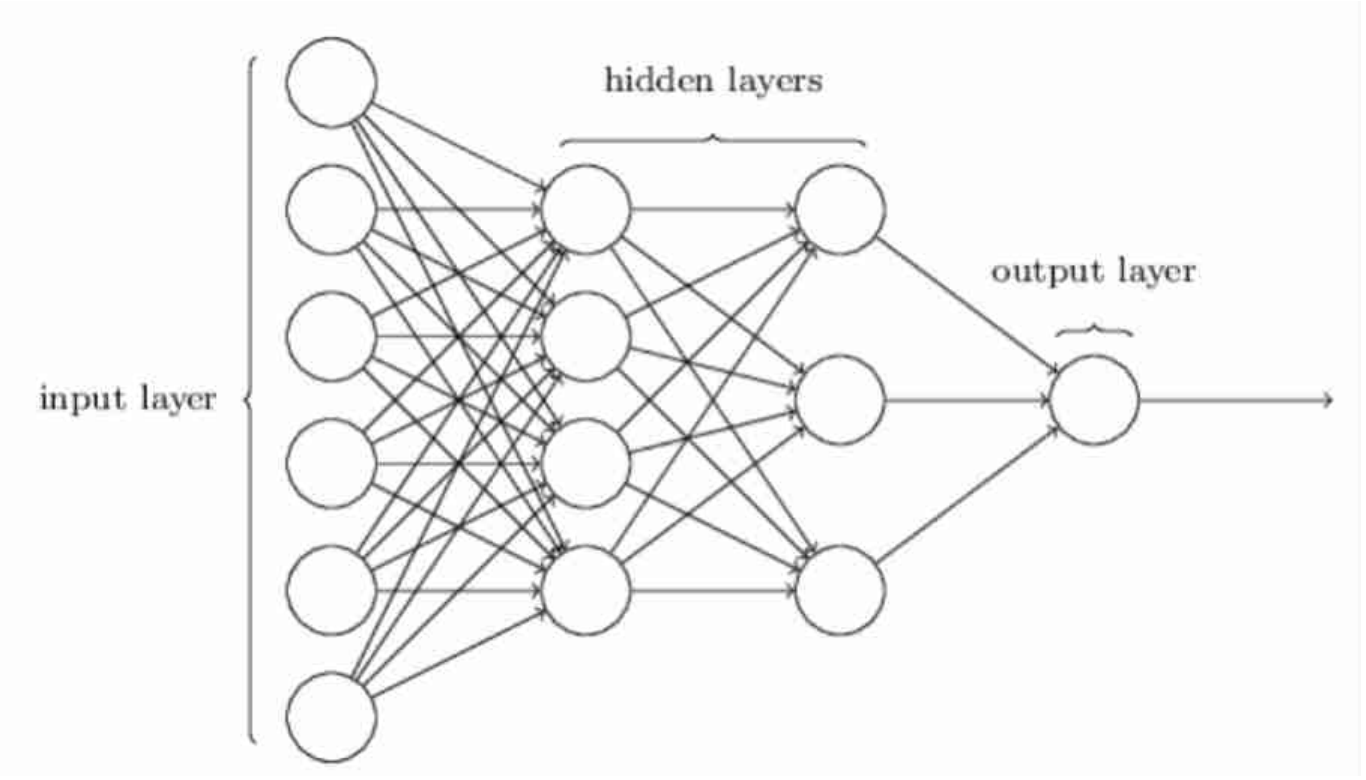
Ex d'architectures neuronales :

- CNN convolutional neural network (adapté au traitement d'image & de champ)
- recurrent network (adapté aux systèmes qui évoluent)

et U-net, GAN (Generalized Adversarial Network), LSTM (long short-term memory), etc....

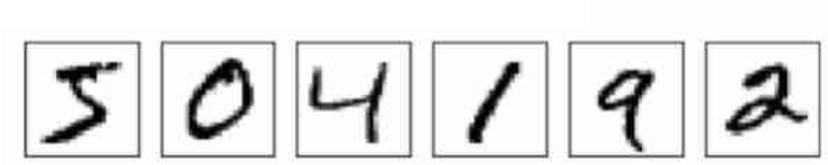
Apprentissage neuronal : exemples d'architectures

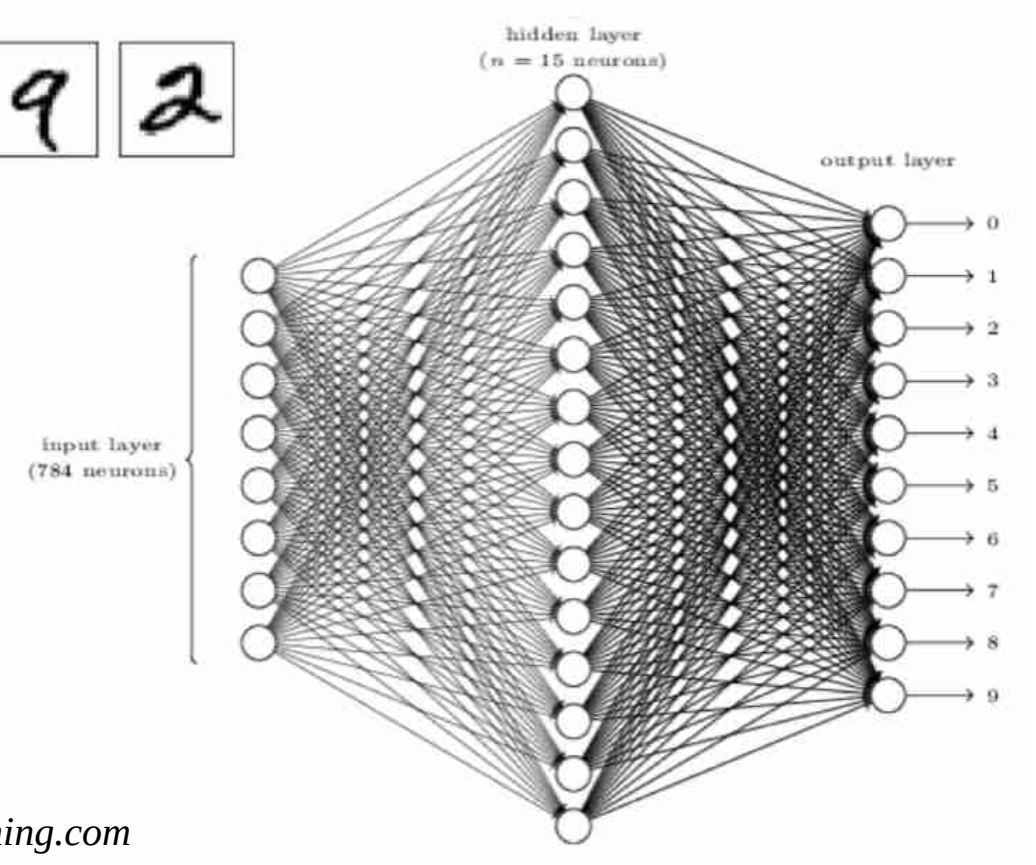
"multilayer perceptron"



Apprentissage neuronal : exemples d'architectures

classifieur (reconnaissance de caractères)





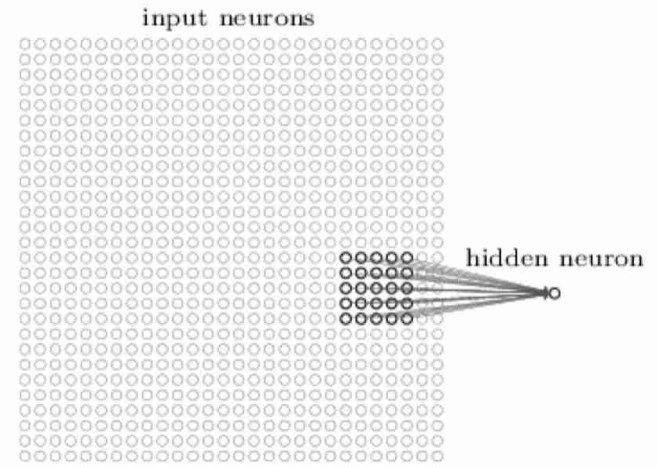
504192

https://neuralnetworksanddeeplearning.com

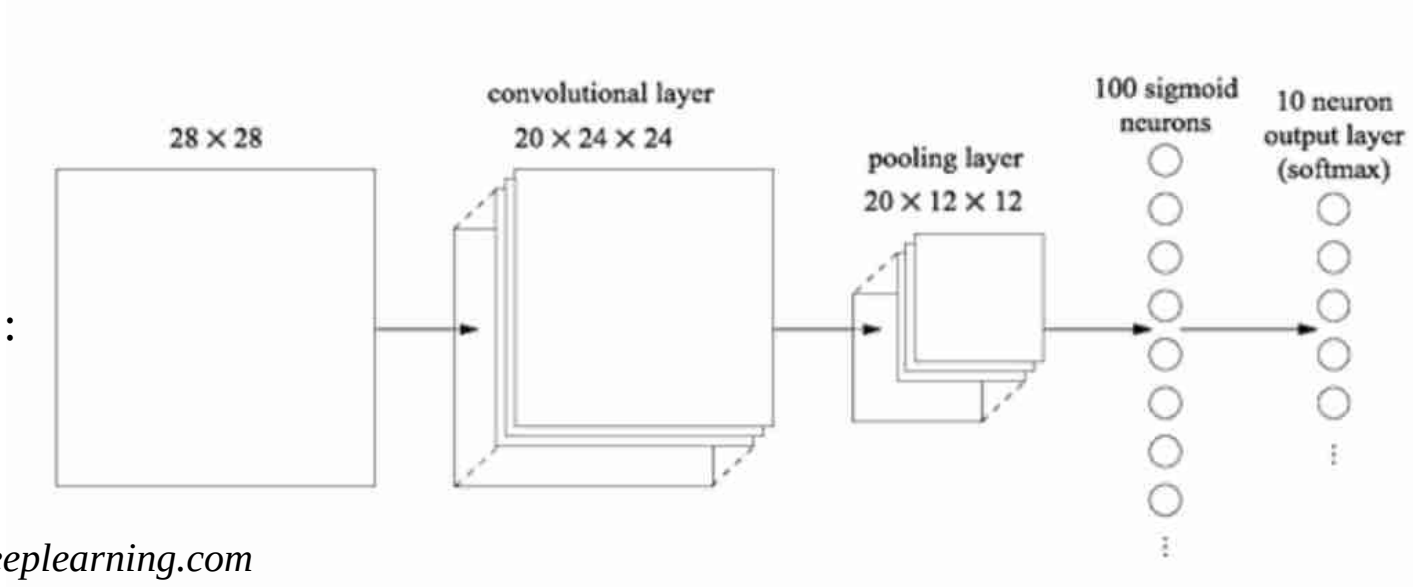
Apprentissage neuronal : exemples d'architectures

CNN (convolutional neural network) : classifieur d'image ('encodeur')

1 couche convolutionnelle :



architecture :

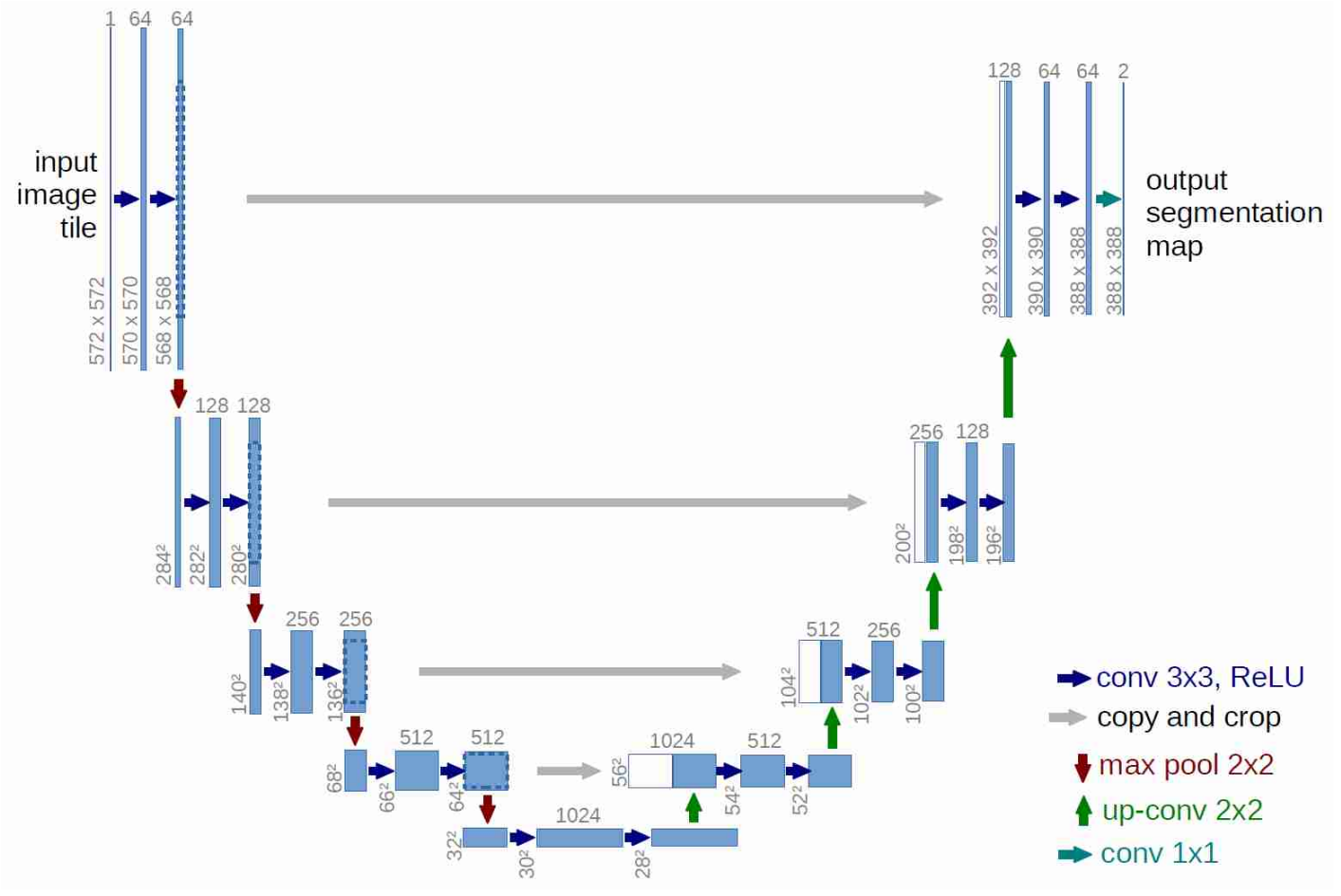


https://neuralnetworksanddeeplearning.com

Apprentissage neuronal : exemples d'architectures

U-net : transformation d'image (encodeur+decodeur)

architecture :

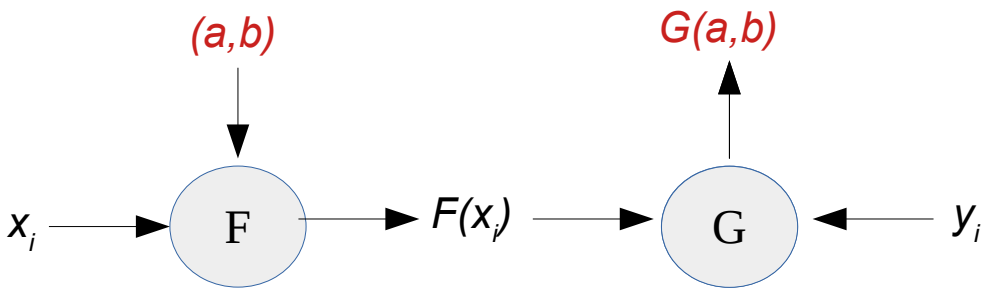


Apprentissage neuronal (2)

Le réseau de neurones définit une fonction modèle (qui peut être très complexe et de grande dimension)
F : x -> y

Pour mesurer sa qualité on définit une fonction objectif G, par ex :
$G(y) = \sum_i || F(x_i) - y_i ||^2$ où la somme i se fait sur toutes les données (x_i,y_i)

Apprentissage : trouver les (a,b) qui minimisent G(a,b) étant donné un groupe de (x_i,y_i)



Apprentissage neuronal (3) : comparaison avec le 4DVar

En assimilation :

- entrée : ébauche et obs (xb,y)
- inconnue : analyse xa
- objectif : fonction-coût J
- contraintes : opérateur B, R, H, M ; minimiseur quasi-quadratique

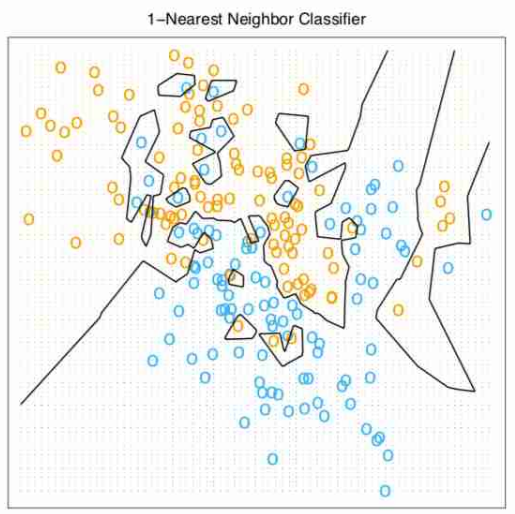
En apprentissage :

- **entrée :** données d'apprentissage (xi,yi)
- **inconnue : "modèle"** = l'ensemble des paramètres (a,b) du réseau de neurone: $f_i(x) = th(a_i x + b_i)$
- **objectif :** fonction-perte G(a,b) qui mesure la distance entre F(xi) et yi
- **contraintes :** architecture du réseau + pour accélérer l'apprentissage on ajoute un terme de régularisation ~ terme d'ébauche :

  $$G(a,b) = || f_i(x) - y ||^2 + || a ||^2 + || b ||^2$$

- minimisation de fonction somme de nombreux termes similaires : minimiseur adapté aux problèmes non-linéaires de grande dimension : **gradient stochastique**, calculateur GPU massivement parallèle (Nvidia : valeur x3 en 2023 -> valeur en bourse 10^{12}\$)

Apprentissage supervisé : préparation des données



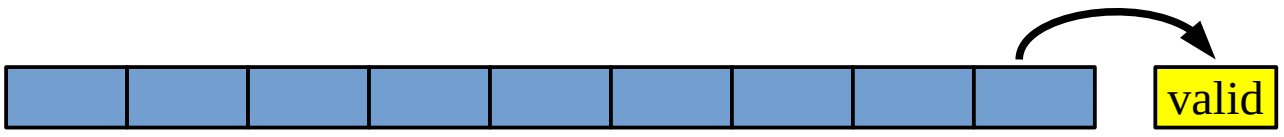
Objectif : éviter le surapprentissage

On divise les données en 2 :

- **d'apprentissage :** optimiser les paramètres du modèle en collant à ces données
- **de validation :** mesure indépendante de la qualité de ce modèle

Comment répartir les données entre apprentissage et validation ?

- on veut garder le + possible de données pour l'apprentissage (~90 % du total)
- on veut un max. d'indépendance par rapport aux données de validation



Méthode du bootstrap par bloc circulaire : (par ex sur séries temporelles) :

- on divise la séquence de données en ~10 blocs consécutifs
- on en prend 9 pour l'apprentissage, 1 pour la validation
- on recommence en décalant 10 fois

Compromis variance-biais :

- ***variance :*** modèle qui est sensible à la sélection de données
- ***biais :*** modèle qui ne "colle" pas bien aux données

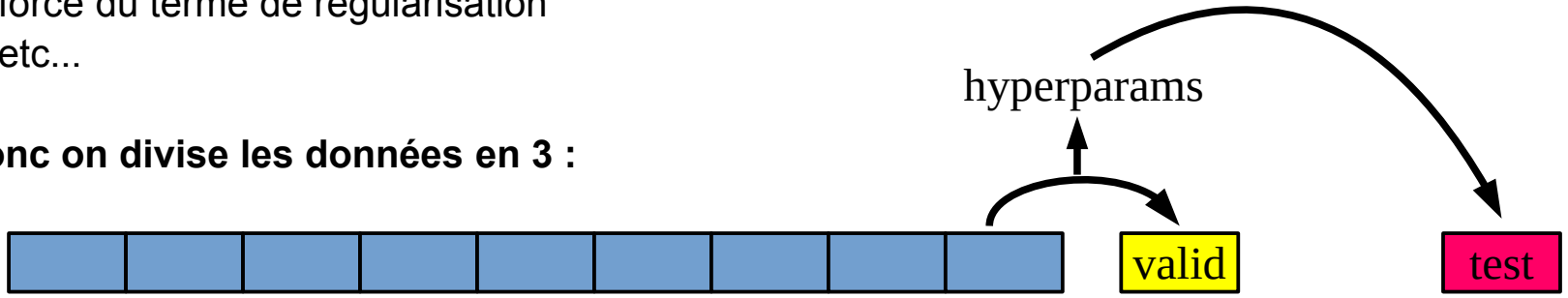
Apprentissage supervisé : préparation des données (2)

Objectif : accélérer aussi l'apprentissage

-> en plus de F on veut optimiser des '**hyperparamètres**' du système, par ex :

- taux d'apprentissage (~taille des itérations)
- force du terme de régularisation
- etc...

Donc on divise les données en 3 :



- **d'apprentissage :** optimiser les paramètres du modèle en collant à ces données
- **de validation :** mesure indépendante de la qualité de ce modèle
- **de test :** mesure finale à la toute fin de l'optimisation

Conclusion : l'apprentissage neuronal est très empirique, beaucoup de "recettes de cuisine" à adapter au problème

1 librairie (= algèbre linéaire parallèle + différenciation automatique (pytorch, keras...)