# Journée "Evolution du Machine Learning vers le Deep Learning"

## Apprentissage automatique, modèles & évaluation

Thierry Brouard (thierry.brouard@univ-tours.fr)

12 mars 2020

Laboratoire d'Informatique Fondamentale et Appliquée de Tours
64 av. Jean Portalis, 37200 Tours

# Portrait d'Edmond de Belamy (2018)

*Portrait d'Edmond de Belamy* est une peinture signée par le collectif Obvious.

Des G.A.N.[1] (IA), ont *vu* plus de 15 000 portraits, depuis le moyen âge, de façon à apprendre les règles de ce style de peinture. Par la suite on les utilise pour fabriquer des peintures, dont celle présentée sur la page précédente.

Approché par Christie's qui a estimé la *toile* entre 7 000 et 10 000 dollars, le collectif a finalement décidé de la mettre aux enchères. Le 25 octobre 2018, elle a été adjugée, comme première œuvre d'art produite par une IA, au prix de 432 500 dollars.

1. Generative Adversarial Networks

# Introduction

- Dès 1950 (Turing [40]) avance deux options pour construire une I.A. :
  1. le modèle de connaissances est formé d'un ensemble de règles, conçues par l'humain ;
  2. le modèle de connaissances est construit d'après des cas d'exemple.
- Il avance l'idée que l'IA ne pourra pas être efficace si elle est construite manuellement : il faudra que des algorithmes d'apprentissage construisent le modèle en étant dirigés par les données.

# Introduction

- Le *machine learning* – M.L. (ou apprentissage automatique) consiste à déterminer, par calcul, ou par essai-erreur, les paramètres d'un modèle de façon à optimiser le comportement de ce modèle.
- Durant cette phase, le modèle va, en moyenne, de mieux en mieux réaliser sa tâche ; on considère donc qu'il apprend une tâche en étant entraîné à la réaliser.

# Introduction

- Les modèles de *machine learning* sont nombreux, variés, et peuvent être organisés selon différents critères. Il est donc difficile d'en dresser un panel exhaustif, ordonné, offrant un outil d'aide à la décision dans le choix d'une telle technique par rapport à un problème donné.
- Les éléments qui suivent ne sont donc que parcellaires ; ils visent surtout à donner un aperçu de techniques, accompagnées d'angles de vue permettant de se faire une petite idée des usages et de la difficulté du choix, mais aussi de l'utilisation, d'un modèle.

# En bref...

1. Trucs en vrac !
2. Un modèle, pour quels usages ?
3. Quelques applications (orientées médical/biologie)
4. Quelques modèles courants
5. Evaluation
6. Difficultés d'apprentissage
7. Le mot de la fin

# Trucs en vrac !

# Notion de *modèle*

Un *modèle* va désigner un ensemble, plus ou moins complexe (algorithmes, fonctions mathématiques), dont le but va être de simuler le fonctionnement d'un processus du monde réel.

Par simulation on entend la capacité à proposer un résultat théorique très proche de valeurs expérimentales.

*Informatiquement*, un modèle est une *boite noire*, qui produit une sortie en fonction de données d'entrées.

Le *machine learning* consiste à trouver comment régler le point de fonctionnement optimal de la boite noire.

# Entrées, sorties

La prise de décision (identifier un objet, segmenter une image, prévoir une tendance...) résulte de l'utilisation d'un modèle, construit ou appris.

Deux notions sont importantes à ce niveau :

- la notion d'observation, ou d'*exemple* ou d'*individu* : ce sont des mesures, prises dans le monde réel[2], décrivant les informations sur lesquelles on s'appuie, afin de construire le modèle ;
- la notion d'erreur, qu'on va mesurer en sortie du modèle, laquelle est souvent utilisée pour améliorer, de façon automatisée, le modèle.

2. dans certains cas, les observations sont synthétisées

# Notations courantes

Les données d'entrée forment un jeu de données $\mathbf{X}$ de $n$ individus, codés chacun par un vecteur de $m$ *features* :

$$\mathbf{X} = \{\bar{x}_1, \bar{x}_2 \dots \bar{x}_n\} \text{ où } \bar{x}_i \in \mathbb{R}^m$$

Les données à *prédire* forment un jeu de données $\mathbf{y}$ de $n$ éléments caractéristiques des individus de $\mathbf{X}$, codés chacun par un nombre (régression/prédiction) ou une classe (classification/inférence) :

$$\mathbf{y} = \{y_1, y_2 \dots y_n\}$$

avec :

- $y_i \in [0, 1]$ dans le cas d'une régression (sorties normalisées) ;
- $y_i \in \{C_1, C_2 \dots C_k\}$ pour une classification à $k$ classes.

# Types de modèles

*Apprendre* consiste alors à chercher une fonction qui permet de faire correspondre chaque $\bar{x}_i$ à son $y_i$.

Si le modèle dépend d'un jeu de paramètres $\bar{\theta}$ qui contrôle la forme de la fonction, le modèle est dit paramétrique.

$$\tilde{\mathbf{y}} = f(\mathbf{X}, \bar{\theta})$$

avec $\tilde{\mathbf{y}}$ la sortie produite, et $\bar{\theta}$ le jeu de paramètres.

D'autres modèles sont non paramétriques : $\tilde{\mathbf{y}} = f(\mathbf{X})$. Leur comportement n'est lié qu'aux exemples d'apprentissage.

# Modèle paramétrique

La forme de la fonction qui relie $\mathbf{X}$ à $\mathbf{y}$ est imposée.

Le modèle doit donc être en partie déterminé et *paramétré* par l'opérateur.

Ce type de modèle est souvent très sensible aux changements de conditions (influence sur les paramètres du modèle). La composition des données d'apprentissage est très importante (représentativité effective).

Exemple de modèle : estimateur des moindres carrés

# Rôle de l'apprentissage

Un modèle paramétrique est formé de 2 parties :

1. l'une, *statique*, qui constitue la structure du modèle ;
2. l'autre ($\bar{\theta}$), *dynamique* qui conditionne la manière exacte donc va fonctionner le modèle, et qui est l'objectif de l'apprentissage.

Si on a $p$ paramètres à régler, ils forment, ensembles, un espace à $p$ dimensions, dans lequel chaque point est lié à une hypothèse d'apprentissage et définit une hypersurface partitionnant cet espace.

Le but de l'apprentissage est de trouver le meilleur point de cet espace : erreur de prédiction minimum et capacité à généraliser maximum.

# Modèle non paramétrique

Aucune supposition concernant la forme de la fonction qui relie $\mathbf{X}$ à $\mathbf{y}$.

On attend donc de l'apprentissage qui détermine l'ensemble des paramètres du modèle.

Ces modèles nécessitent beaucoup de données afin d'estimer la fonction recherchée

Se pose aussi le problème du contrôle de la complexité du modèle, afin d'éviter le cas où le modèle apprendrait par cœur les données d'apprentissage, mais serait incapable de *généraliser* correctement.

Exemple de modèle : ANN, CNN, HMM, ...

# Modèle linéaire

Un modèle est dit *linéaire* lorsqu'il exprime une variable aléatoire **y** en fonction de variables explicatives **X** sous forme d'un opérateur linéaire sur les paramètres inconnus **B** du modèle selon la formule :

$$\mathbf{y} = \mathbf{XB} + \mathbf{U}$$

avec **B** une matrice de paramètres à estimer et **U** une matrice d'erreurs (écarts, bruits, erreurs de mesures. . . )

Le modèle linéaire le plus connu est la fonction affine : $y = a.x + b$. Schématiquement on peut dire que le processus (dans le monde réel) qui permet de produire **y** en fonction de **X** varie très peu, il suit le même *régime* de fonctionnement.

# Noyau

Dans le cas où le processus qui relie $\mathbf{y}$ à $\mathbf{X}$ ne semble pas linéaire, il est peut-être possible de projeter les mesures $\mathbf{X}$ dans un autre espace, où le problème serait linéaire.

La *fonction noyau* représente la transformation permettant de changer d'espace de représentation.

Ce concept est présent dans différentes modèles, SVM [3] et ANN [4] en particulier.

Le problème peut alors être la découverte de l'expression de la fonction noyau...

3. Support Vector Machine
4. Artificial Neural Network

# Modèle non-linéaire

Dans un certain nombre de cas, le processus réel, qui régit les observations $\mathbf{y}$, ne suit pas toujours le même régime. Sur une certaine plage, il peut être linéaire avec un premier jeu de paramètres, et sur une autre plages, il peut être linéaire avec un autre jeu de paramètres.

Le modèles linéaires sont donc tenus en échec, sauf à connaître précisément les points de changement de régime et d'utiliser une collection de modèles linéaires, chacun s'appliquant uniquement entre 2 points consécutifs.

La difficulté consiste donc à connaître le lieu des changements de régime. On utilise alors des modèles non-linéaires qui, par nature, intègrent la possibilité de séparer les observations en groupes au sein desquels la linéarité est supposée.

Cela revient à dire qu'on peut construire une courbe complexe comme la superposition, ou la concaténation, d'éléments *plus simples*.

# Modèles génératifs vs. discriminatifs

Modèles génératifs : cherche à modéliser la distribution jointe $P(\mathbf{X}, \mathbf{y})$. La classification est déduite en appliquant le théorème de Bayes. Ces modèles peuvent aussi, d'après $\mathbf{X}$, produire des $\mathbf{y}$.

Ex. réseaux Bayésiens, MM, HMM, MRF, GAN, ...

Modèles discriminatifs : modélisent directement la règle de classification $P(\mathbf{y}|\mathbf{X})$. Les paramètres sont estimés par minimisation d'un coût, de classification par exemple. Souvent très efficaces, elles sont supérieures aux méthodes génératives lorsque les distributions de $\mathbf{X}$ sont inconnues et difficilement modélisables.

Ex. SVM, ANN, Arbre de décision, k-ppv, CNN, ...

# Types d'approches pour l'apprentissage

L'idée principale est d'avoir, à sa disposition, des exemples, et de s'en inspirer (apprentissage) pour déduire quelque chose, dans le futur, d'exemples *non appris* (généralisation).

On va donc chercher à *extraire*, d'exemples de référence, des informations utiles, pour bâtir des décisions, d'après de nouveaux exemples observés.

Deux grands types d'approches sont utilisées :

- les approches non-paramétriques ;
- les approches probabilistes.

# Approche non-paramétrique

Plus deux exemples sont proches, plus ils ont des chances de faire partie de la même classe.

Ces approches reposent donc sur la notion de similarité entre exemples. Ce concept est parfois délicat à mettre en œuvre, car ce calcul est entaché de la perception du problème par l'expert, du choix des *features*, du choix du seuil au delà duquel deux exemples sont trop éloignées, entre autres.

Le choix des *features* est très souvent expérimental. Distinguer des exemples d'après ces seules caractéristiques est parfois complexe, spécialement si les caractéristiques subissent des évolutions dans le temps. Et il est possible que certains exemples, aux caractéristiques inhabituelles, tiennent en échec les règles de classification.

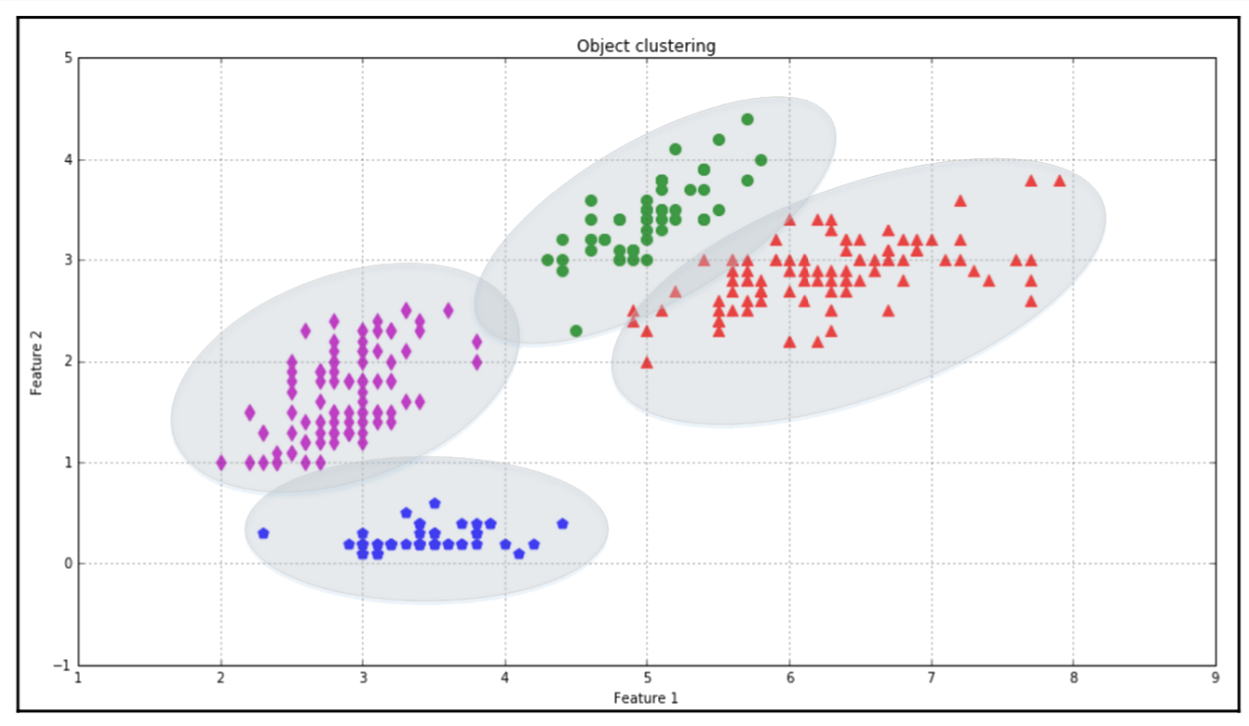
# Approche probabiliste

Au lieu de considérer l'appartenance à une classe d'après des seuils fixes, on va plutôt chercher à exprimer le fait que les valeurs prises par les *features* suivent certaines lois de probabilité.

L'objectif de l'apprentissage consiste alors à estimer les paramètres (moyenne, variance) de ces lois dans le but de maximiser, la probabilité d'observer chaque $y_i$ de $\mathbf{y}$.

De très nombreuses méthodes d'apprentissage suivant cette approche existent.

# Exemple pour un *Mixture Model*




*Source : [8]*

Dans l'utilisation de toute chose, les algorithmes de M.L. ne faisant pas exception, on a toujours, au moins, deux points de vue :

- Qu'est-ce que cela fait : on s'intéresse au résultat final, l'outil est alors une *boite noire* dont la caractéristique prépondérante est la finalité ;
- Comment ça fonctionne : le focus est placé sur les mécanismes internes, et donc sur la compréhension du calcul du résultat final.

Comme, malheureusement, on n'a pas (encore ?) d'outil alliant les deux critères, il faut s'interroger sur la motivation première à l'utilisation d'un outil.

# Un modèle, pour quels usages ?

# Critères de sélection

Précision : elle est obtenue par des modèles complexes, non linéaires, dont l'interprétation est difficile. Ces modèles fonctionnent, mais on ne sait pas toujours expliquer précisément pourquoi [5], dans le sens où la compréhension des relations liant les entrées aux sorties produites nous échappent.

Interprétabilité : elle est clairement facilitée par les modèles linéaires, ou par des méthodes à noyaux, si ceux-ci ne sont pas trop complexes à interpréter. C'est justement parce que ces modèles sont plus simples qu'ils nous sont intelligibles. Par contre, leur performance est moindre sur des problèmes complexes.

---

5. [9] rapporte une expérience, menée chez Google, dans le but de créer une discussion sécurisée. Deux I.A., *Alice* et *Bob* ont communiqué ensemble en inventant leur propre langage et leur propre protocole de sécurité. Une 3eme I.A. a été créée pour déchiffrer la conversation. Mais les chercheurs n'ont toujours pas compris ce qu'*Alice* et *Bob* ont pu s'écrire.

# Motivation #1 : inférence

Approche *boite noire* : l'information principale est la décision rendue par l'algorithme, et non la manière dont les entrées sont utilisées pour calculer cette décision.

Exemples : déterminer si un cas est pathologique ou non ; extraire tous les pixels d'une région ; plus généralement attribuer un label à un *individu*, avec un *taux de confiance* . . .

Les modèles non-linéaires, par exemple, sont très bons dans ce type de tâche, mais difficilement interprétables, donc peu adaptés en ce qui concerne l'explication de la décision prise.

# Motivation #2 : prédiction

Approche *boite blanche* : on cherche à comprendre comment telle caractéristique d'entrée influe sur le résultat final.

Exemples : quelle sera l'évolution de la glycémie dans les 6 prochaines heure ? Quelle sera la zone touchée par l'épidémie d'ici 5 jours ?

Les modèles linéaires sont souvent privilégiés, car ils permettent de mieux appréhender les relations qui existent entre les sorties et les entrées. Leur explicabilité est meilleure (donc la détection d'anomalie dans une décision est plus facile également, donc l'amélioration du modèle de prédiction est plus aisée).

# Classification

On fait apprendre des *formes* (images, sons, vecteurs . . . ) à la machine, on attend, ensuite, de celle-ci, qu'en présence d'une forme non-étiquetée, elle puisse proposer une, ou plusieurs, classes d'appartenance. **y** est discret et/ou qualitatif.

Exemples d'applications :

- reconnaissance faciale (visages, émotions)
- reconnaissance vocale (phonèmes, sons)
- reconnaissance de l'écriture (images de mots, de lettres)
- reconnaissance d'objets pour conduite autonome
- reconnaissance gestuelle pour IHM, jeux
- identification de pathologies (images, symptômes)
- pilotage d'interfaces cérébrales
- . . .

# Régression

On fait apprendre des *séries numériques* à la machine, on attend, ensuite, de celle-ci, qu'en présence d'une série partielle, elle la complète ou la prolonge. **y** est numérique et continue.

- prévision de séries temporelles (économie, finance)
- prévision de phénomènes physiques (météorologie)
- prévision de dommages sur des conflits
- évolution de la propagation d'une épidémie
- ...

# Clustering

On fait apprendre des *formes* (images, sons, vecteur . . . ) à la machine, on attend, ensuite, de celle-ci, qu'elle forme des groupes *homogènes* de formes. Pas de contraintes sur **y** (qui peut être absent).

- création de corpus de textes
- aide à la transcription
- groupage de personnes à risques
- découverte de connaissances
- profilage d'utilisateurs (transports, téléphonie, CRM [6])

6. Customer Relationship Management

# Estimation de densité

On fait apprendre des *formes* (images, sons, vecteur . . . ) à la machine.

On étudie ensuite le modèle obtenu de façon à identifier les lois de probabilité caractérisant la distribution des $\mathbf{X}$.

Plusieurs applications, parmi lesquelles :

- validation d'hypothèses sur l'emploi d'un certain type de modèle
- restauration de signaux (dont les images)
- débruitage de signaux (dont les images)
- génération d'images (mode, art, cinéma)

# Réduction de dimension

On fait apprendre des *formes* (images, sons, vecteur . . . ) à la machine.

On étudie ensuite le modèle obtenu de façon à identifier les $\mathbf{X}$ les plus significatifs,

- soit réduire la taille des données d'entrée ;
- soit pour limiter le temps de calcul ;
- soit pour améliorer la précision (il est parfois plus facile de trouver une relation de projection si on a moins de données en entrée) ;

Parfois les trois conséquences sont observées en même temps...

# De nombreuses pratiques . . .

Il n'y a pas autant d'algorithmes d'apprentissage que de modèles ; il y en a plus !

- Pour un modèle donné, on peut trouver plusieurs approches ;
  - On peut, parfois, poser un système d'équations et le résoudre (par ex. méthode des moindres carrés) ;
  - On peut, parfois, faire évoluer, par calcul, un modèle, pour atteindre un objectif (par ex. descente de gradient) ;
  - On peut, parfois, faire évoluer, par essai-erreur, un modèle pour atteindre un objectif (par ex. algorithme génétique) ;
- Pour une approche donnée, on peut trouver plusieurs stratégies d'amélioration des paramètres, incluant différentes mesures de l'erreur commise (apprentissage supervisé) ;

# Quelques applications (orientées médical/biologie)

# MYCIN (1972)

Système expert à but de diagnostic médical concernant les infections du sang.

Conçu et réalisé en LISP par E.H.Shortliffe, à l'université de Stanford.

Utilise un moteur d'inférence basé sur environ 500 règles, capable d'expliquer son raisonnement.

Mode conversationnel : la machine pose des questions en langage naturel, établit des hypothèses, cherche à les valider, afin d'aboutir à une liste probable de conclusions.

Fiabilité : 65 % des diagnostics corrects, ce qui a été jugé très bon, à l'époque (*i.e.* supérieur à ce que des praticiens, avec peu d'expérience, obtenaient).

# DESIREE (en cours)

Partenariat entre différents laboratoires, universités et entreprises.

Financement Européen.

Plateforme Web [7] dédiée au cancer du sein :

- diagnostic, basé notamment sur l'analyse d'images ;
- aide au choix de la thérapie à appliquer ;
- chirurgie virtuelle afin d'anticiper le résultat d'une thérapie.

I.A. basée sur une technique appelée CBR (*case-based reasoning*) [24]

7. Voir le site officiel : http ://www.desiree-project.eu

# Métrologie sur des images en microscopie optique (2010)

Partenariat entre le LIFAT et l'unité INSERM 921, projet financé par la Région Centre.

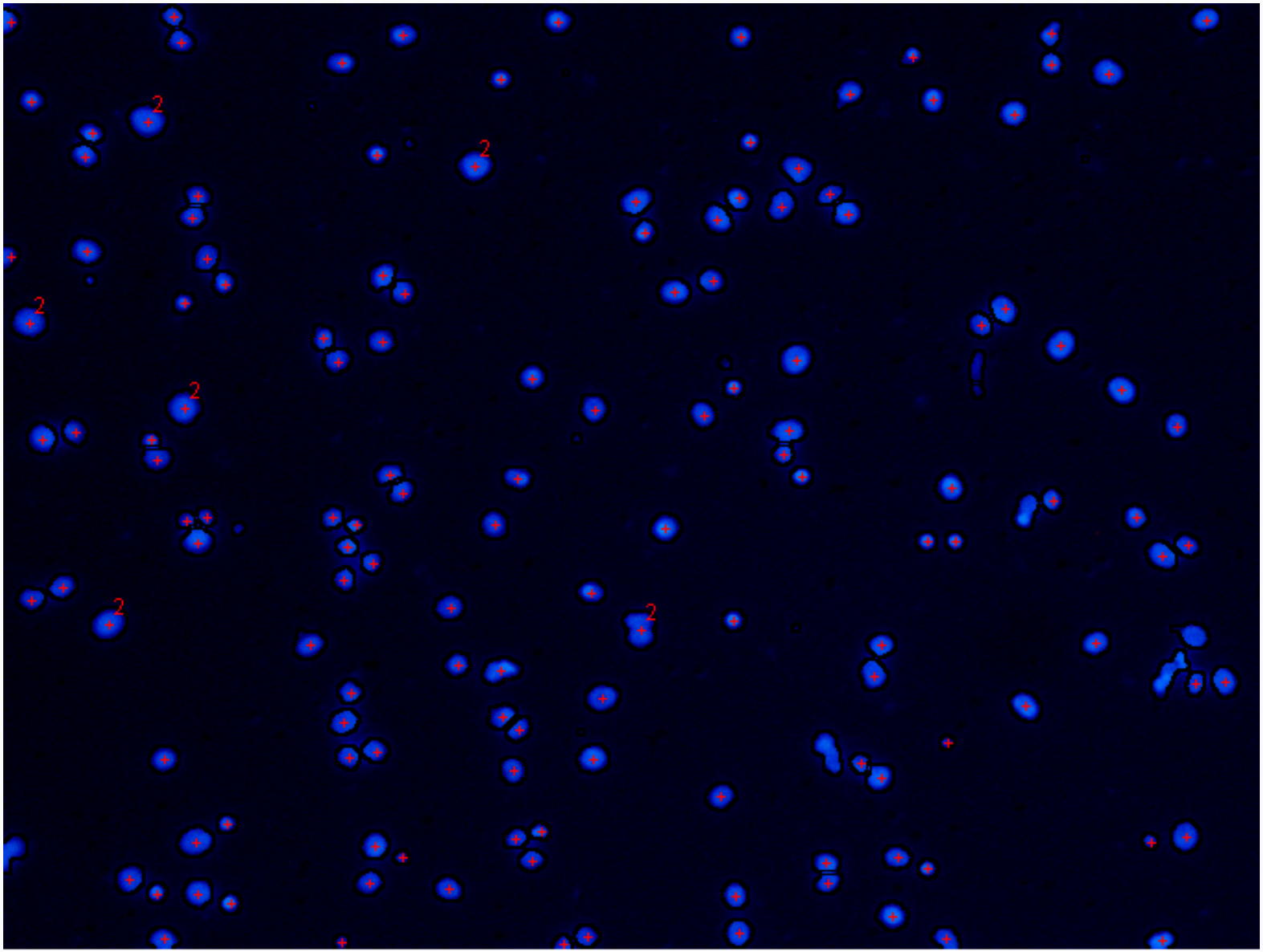
Lignées de cellules étudiées dans le cadre de la recherche contre le cancer.

But : compter les cellules vivantes présentes dans l'image : trouver les cellules, déterminer si elles sont vivantes ou non, et compter les vivantes. Suivi dans le temps pour évaluer l'influence des drogues destinées à freiner la motilité.

Images en microscopie optique à fluorescence.

En moyenne, 3 ms par image traitée, précision supérieure à 99 %.

# Brouard et *al.* [10]



# Segmentation d'images 3D (2016-)

Partenariat entre le LIFAT et l'INRA Val de Loire, UMR 85 PRC.

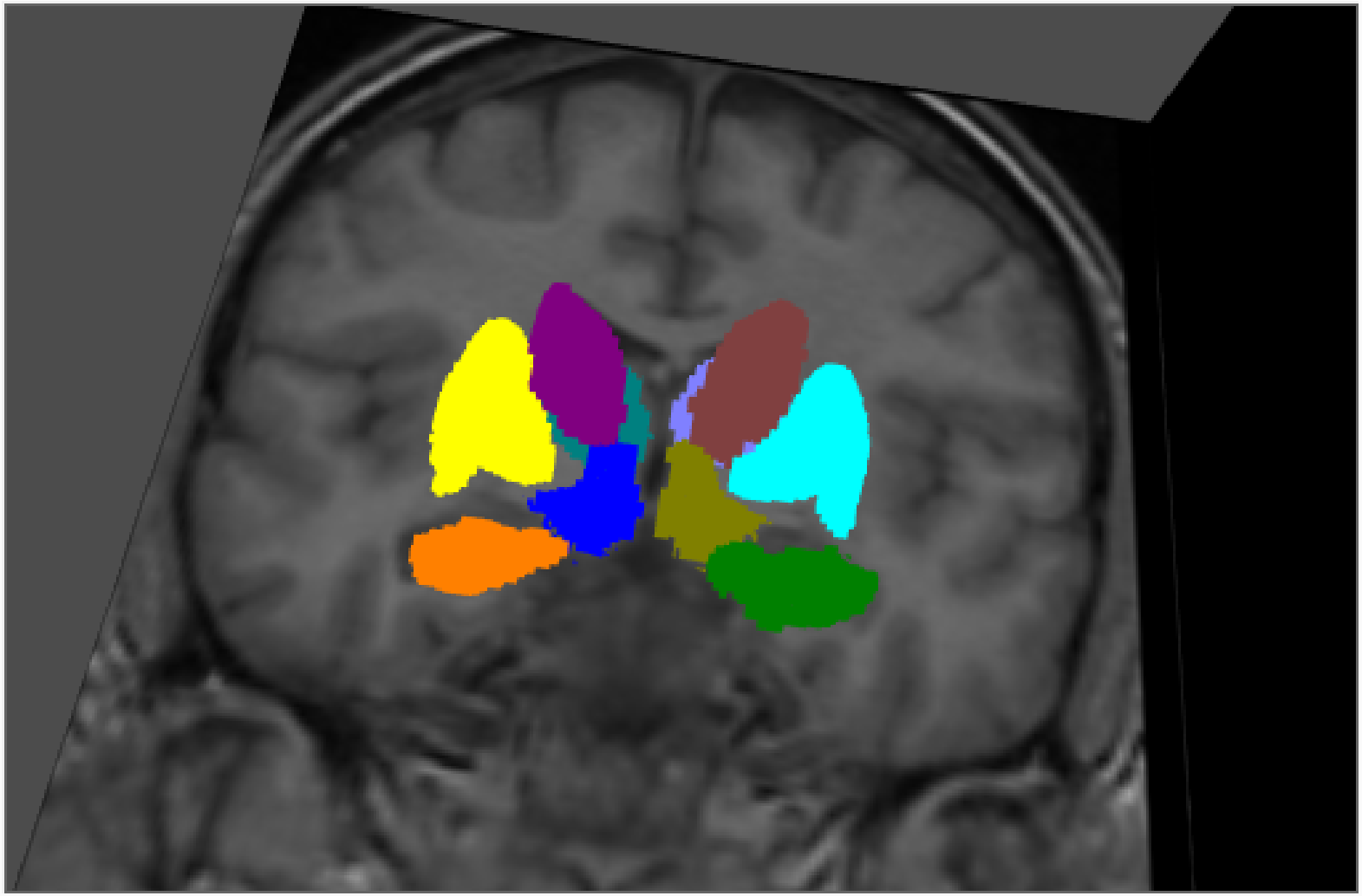
Repérage automatique de zones sur des IRM de cerveaux de brebis.

Combinaison d'outils de traitement d'images et de modèles à base de graphes, utilisation de connaissances *a priori* (atlas) pour aider la localisation automatique.

Insertion de l'expertise de l'utilisateur, qui choisit de segmenter une zone avant une autre car elle semble plus facile à obtenir.

Assistance proposée par la machine a montrée que la tâche est accessible à des utilisateurs non/moins experts

# Galisot et *al.* [16]



*Vue 2D d'une IRM 3D d'un cerveau de brebis, avec des régions segmentées*

# Traitement des maladies de l'œil (2018)

Partenariat entre Google DeepMind et hôpital Moorsfield (London, UK) depuis 2016.

But : I.A. capable de détecter les maladies de l'œil.

Premiers résultats deux ans plus tard [14] :

- Analyse d'images 3D du fond de l'œil ;
- Détection de plus de 50 pathologies (*ie* diabète, dégénérescence maculaire...) ;
- Recommandation d'un traitement adapté ;
- Suggestion de soins à réaliser de manière urgente.

Taux moyen d'erreur : 5,5 % (de 6,7 % à 24,1 %) sur un échantillon de 997 patients, disponibilité immédiate.

# Détection de l'insuffisance rénale (2019)

Partenariat entre Google DeepMind et le University College of London

Résultats publiés en 2019 [38] :

- Diagnostic rapide (15 minutes, au lieu de plusieurs heures) ;
- Détection jusqu'à 48h à l'avance ;
- Réduction du coût des admissions grâce à la détection précoce ;

Taux moyen d'erreur : 3,3 % (contre 12,4 % pour les spécialistes).

# Sans oublier . . .

Chirurgie virtuelle, comme outil de formation, mais aussi d'intervention à distance, de simulation.

IHM ou robots compagnons, pour personnes âgées, ou présentant des pathologies mentales.

Métrologie pour la réalisation de prothèses, le contrôle de la réussite d'une opération.

Dissection virtuelle [35].

Segmentations d'images (toutes modalités, 2D, 3D).

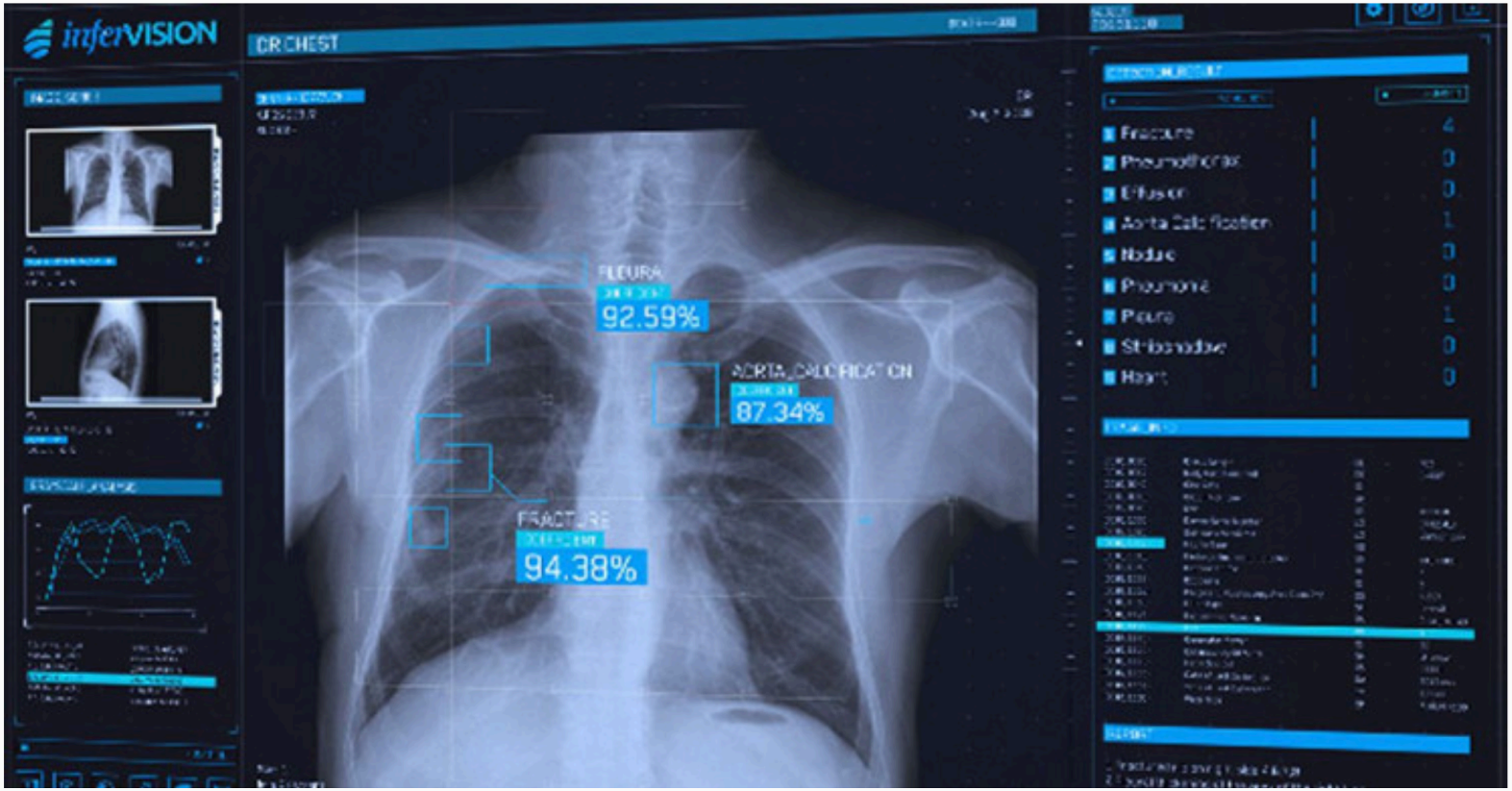
Identification de fibres nerveuses [15].

Reconnaissance de cellules tumorales, calcul du risque de récidive.

Repérage précoce d'effets secondaires médicamenteux.

. . .

# Screening & Covid-19




*inferVISION, logiciel permettant de cibler en un temps record des signes, même partiels, du Covid-19 sur les images des patients [29]*

# Quelques modèles courants

*L'apprentissage supervisé s'applique à de très nombreux modèles :*

Arbre de décision, Forêts aléatoires, Régression linéaire, Classifieur naïf de Bayes Régression logistique, Perceptron, U-matrix, Conditional Random Fields, Markov Random Fields, Mixture Models, Hidden Markov Models, Probabilistic Graphical Model, Support Vector Machine, Perceptron multicouche, cartes de Kohonen (SOM), Radial Basis Function, Bayesian Network, Feed Forward Neural Network, Réseau de Hopfield, Restreint Boltzmann Machine, k-NN, Convolutional Neural Network, Time Delay Neural Network, Spiking Neural Network ...

# Régression linéaire (apprentissage supervisé)

R.J. Boscovich ≈1755 (intuition) puis P.S. Laplace (formalisation)

But : trouver la droite qui *passe le mieux* à travers un nuage de points.

Problème de base : points de coordonnées $(x, y)$ où $x$ est la feature, et $y$ le label, trouver $(a, b)$ tq $y_i = a.x_i + b$

Facilement généralisable à $n$ features :

- $n = 1$ : droite de régression, représentation dans un espace de dimension 2 (feature, label) ;
- $n = 2$ : plan de régression, espace de dimension 3 ;
- ...
- $n = k$ : hyperplan de régression, espace de dimension $k + 1$ ;

# Régression linéaire

Utilisée en :

- prédiction : valeur probable de la sortie pour une entrée donnée ;
- classification : de quel coté de la droite/plan/hyperplan se situe-t-on dans un cas à 2 classes ?

Hypothèses :

- variables explicatives non corrélées entre elles, aux termes d'erreur ;
- termes d'erreur indépendants, de variance constante, normaux.

Plusieurs méthodes d'estimation des paramètres :

- les moindres carrés (Least Squares – LS) (A.M. Legendre≈1805, C.F.Gauss≈1809)
- le maximum de vraisemblance (Maximum Likelihood – ML) (R.A. Fisher, 1922 [37])
- inférence Bayésienne (Bayesian Inference – BI)

# Mais la réalité est souvent plus complexe...

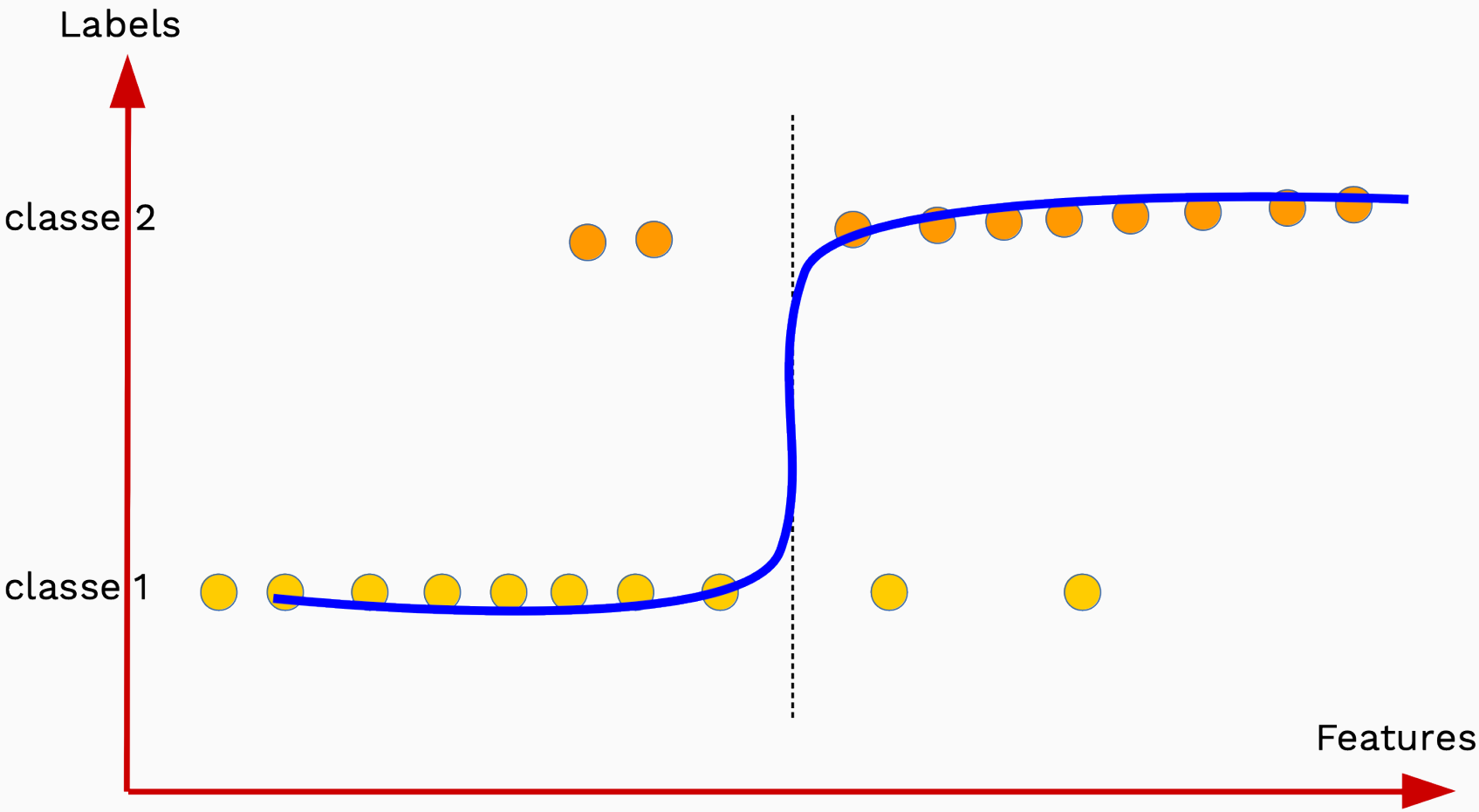
De nombreux problèmes ne sont pas linéaires :

- ajustement de fonction non linéaires ;
- labels non numériques ;
- classification d'objets ...

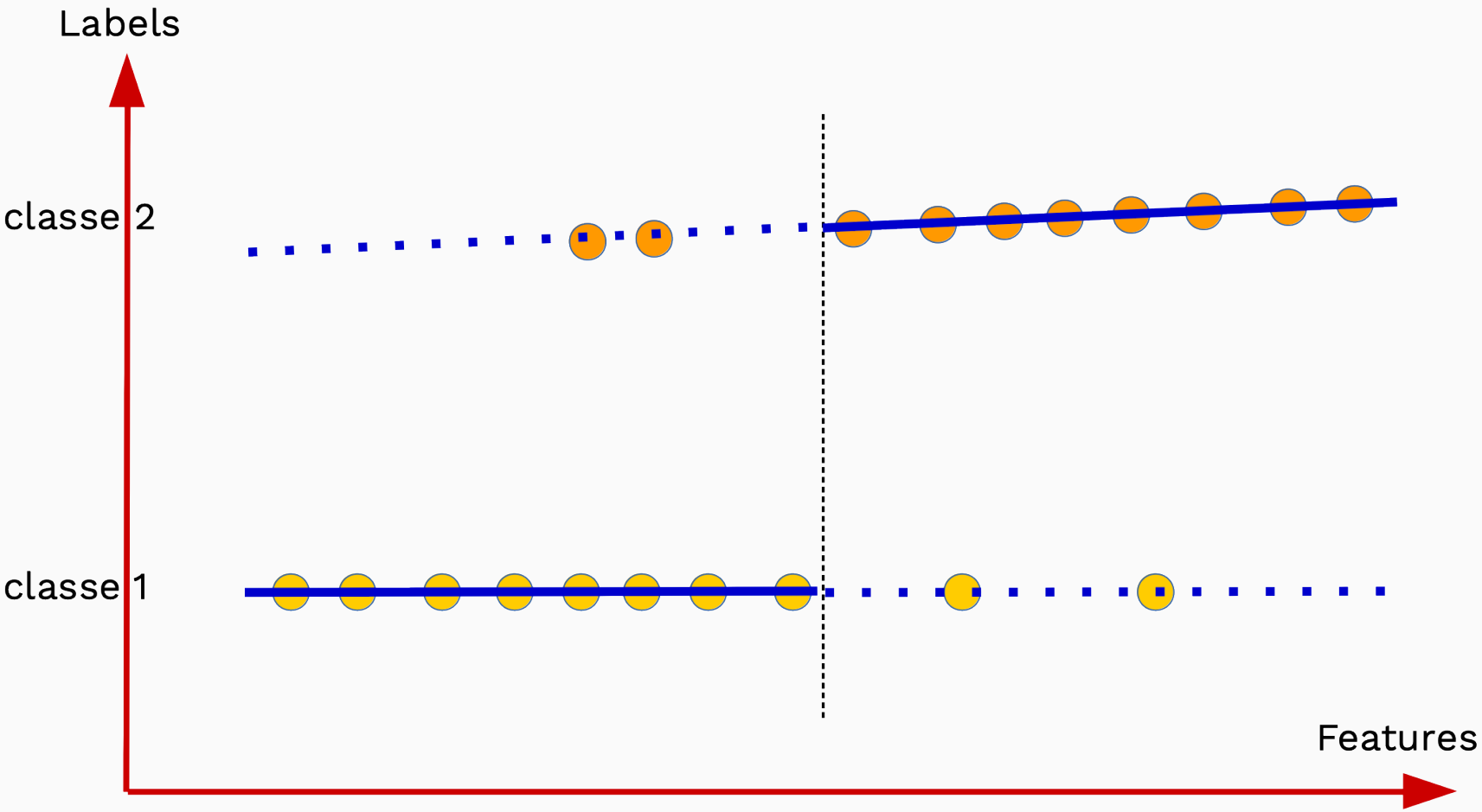
On fait alors appel à :

- des méthodes qui transforment les features (méthodes à noyaux) ;
- des méthodes qui introduisent de la non-linéarité (par ex. réseaux de neurones récurrents – ceux à propagation avant sont linéaires) ;
- l'introduction de points de ruptures dans l'espace des features pour se ramener à des portions plus ou moins linéaires ;

# Regression logistique



# Soft Support Vector Machine



# Classifieurs linéaires

La généralisation de l'introduction de points de ruptures en $n$ dimensions conduit à :

- en dimension 2, une droite de rupture qui sépare l'espace des features en 2 demi-plans
- en dimensiont 3, un plan de rupture qui sépare l'espace en 2 demi-espaces
- en dimension n, un hyperplan de rupture...

Pour reprendre la dimension 2, on a un demi-plan par classe. Cette technique conduit donc à la construction de classifieurs linéaires. La difficulté est d'isoler les individus appartenant à la même classe dans le même sous-espace.

# Support Vector Machine

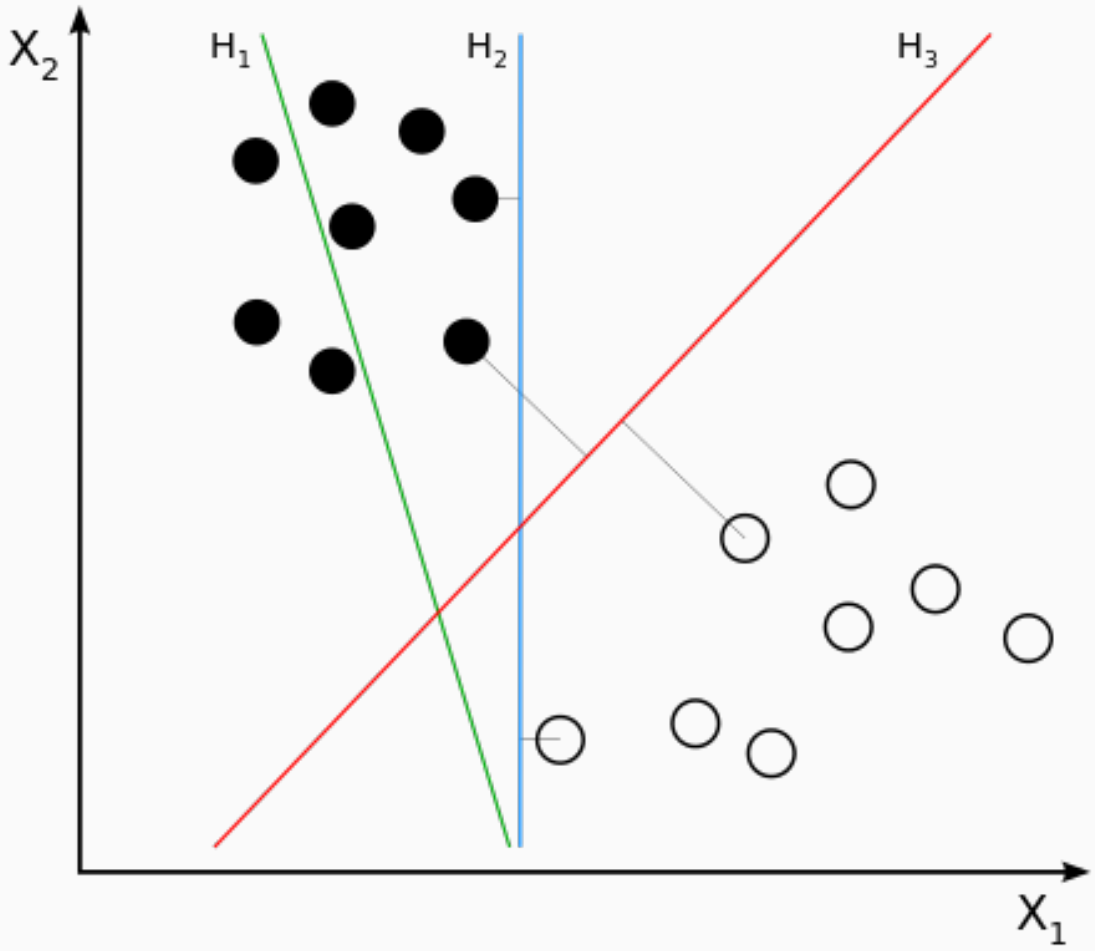
Le *séparateur à vaste marge* [12] est une généralisation du classifieur linéaire.

Il est utilisé aussi bien pour la régression que pour la classification. Certains travaux l'utilisent même pour le clustering [6] avec un algorithme d'apprentissage non supervisé.

Le principe est de trouver, dans l'espace de représentation, la droite qui sépare les deux classes en présence de la façon la plus *évidente*, avec le maximum de *sécurité*

Si on a un problème où plus de deux classes sont présentes, on en sépare une du reste ; puis, dans ce reste, une autre encore du reste, et ainsi de suite (approche hiérarchique).

# Support Vector Machine




$H_1$ ne sépare pas les deux classes, $H_2$ le fait, mais avec moins de *marge* que $H_3$.

# Support Vector Machine

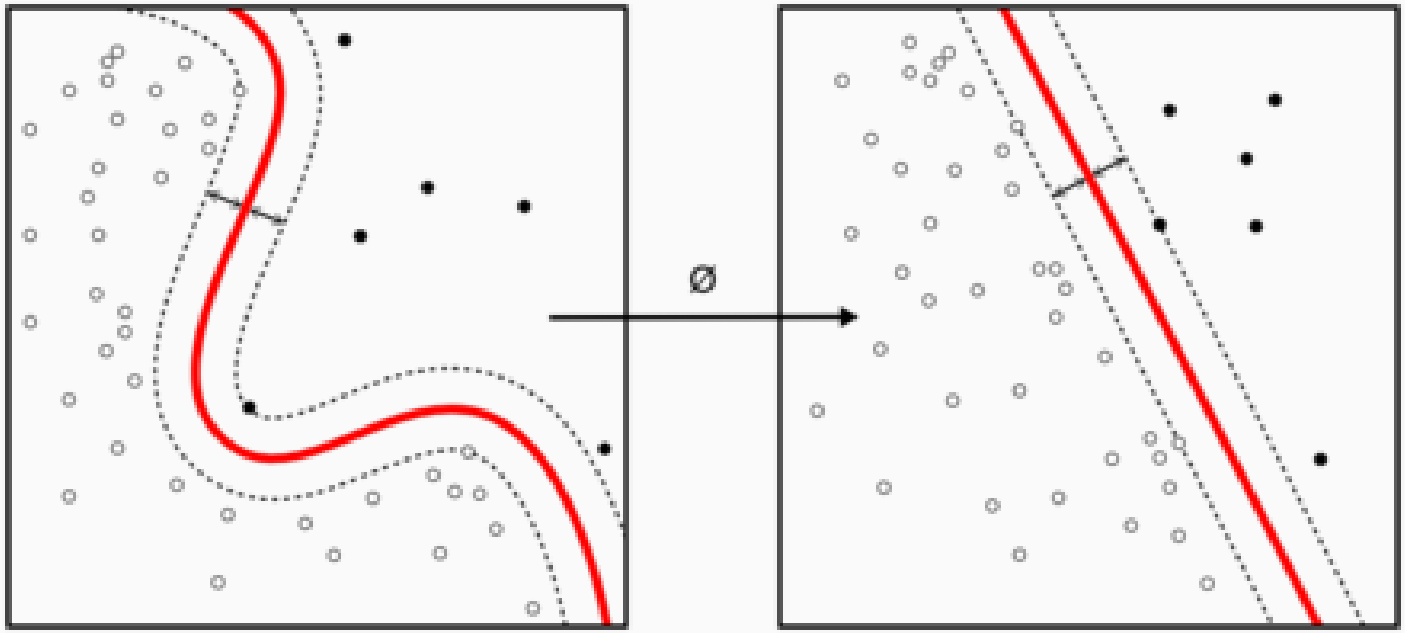
Si les données d'entrée ne sont pas linéairement séparables, on va chercher une fonction noyau qui va projeter ces données dans un espace où elle le seront (on l'espère. . . )

Une fois le problème *linéarisé*, on procède comme précédemment.

Dans la réalité, il y a toujours une marge d'erreur, prise en compte dans le modèle au moyen de fonctions de perte [33], utilisées avec ou sans noyau.

Nombreuses applications dans tous les domaines [41], y compris en *feature selection* : en analysant les vecteurs supports, on peut calculer dans quelle mesure telle caractéristique d'entrée est jugée discriminante ou non. On peut ainsi réduire le nombre de caractéristiques d'entrée, sous réserve d'évaluer l'impact sur la précision du modèle. Voir aussi [21].

# Support Vector Machine




La fonction noyau $\phi$ transforme l'espace de représentation de façon à se ramener à un problème linéaire.

# Réseau de neurones

Travaux sur le *neurone formel*, des neurologues Warren McCulloch et Walter Pitts, fin des années 1950 [25].

Un neurone ( en version simplifiée) ...

- reçoit des entrées ;
- somme ses entrées ;
- compare la somme résultante à une valeur seuil ;
- répond en émettant un signal si cette somme est supérieure ou égale à ce seuil.

De nombreux autres modèles existent, un réseau de neurones consiste à définir un graphe ou certains neurones perçoivent les entrées, d'autres les sorties des précédents, etc. jusqu'à un groupe qui contrôle la sortie du réseau. Cette sortie peut être booléenne (connu/inconnu), en $n$ exemplaires (autant que de classes à reconnaître), ou numérique (pour la régression notamment).

# Exemple de neurone formel

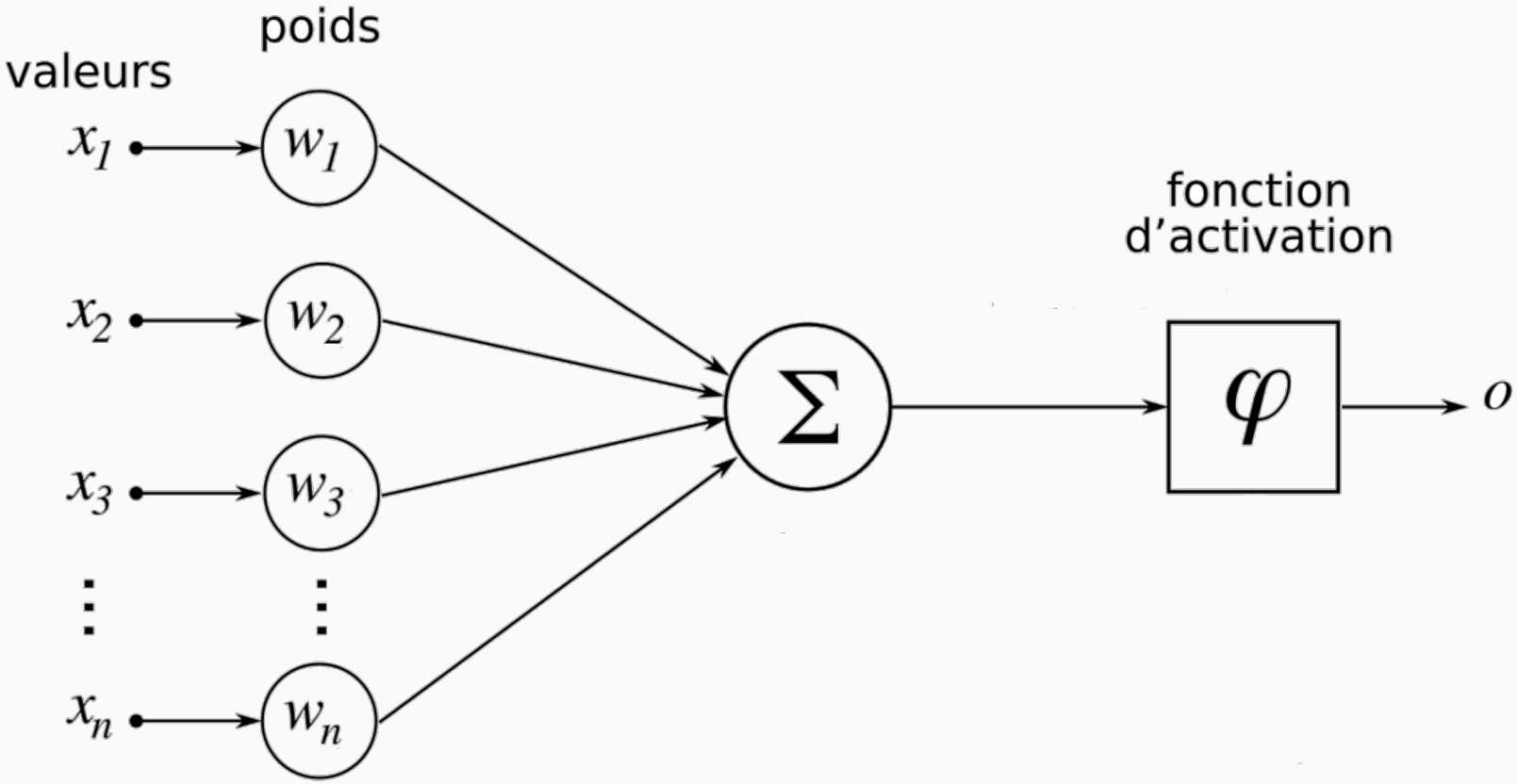



Schéma d'un neurone formel

# Perceptron

1959 : F. Rosenblatt crée le perceptron [34] (modèle linéaire), qui évoluera plus tard (1986) en MLP (*Multi-Layer Perceptron* (modèle non linéaire)

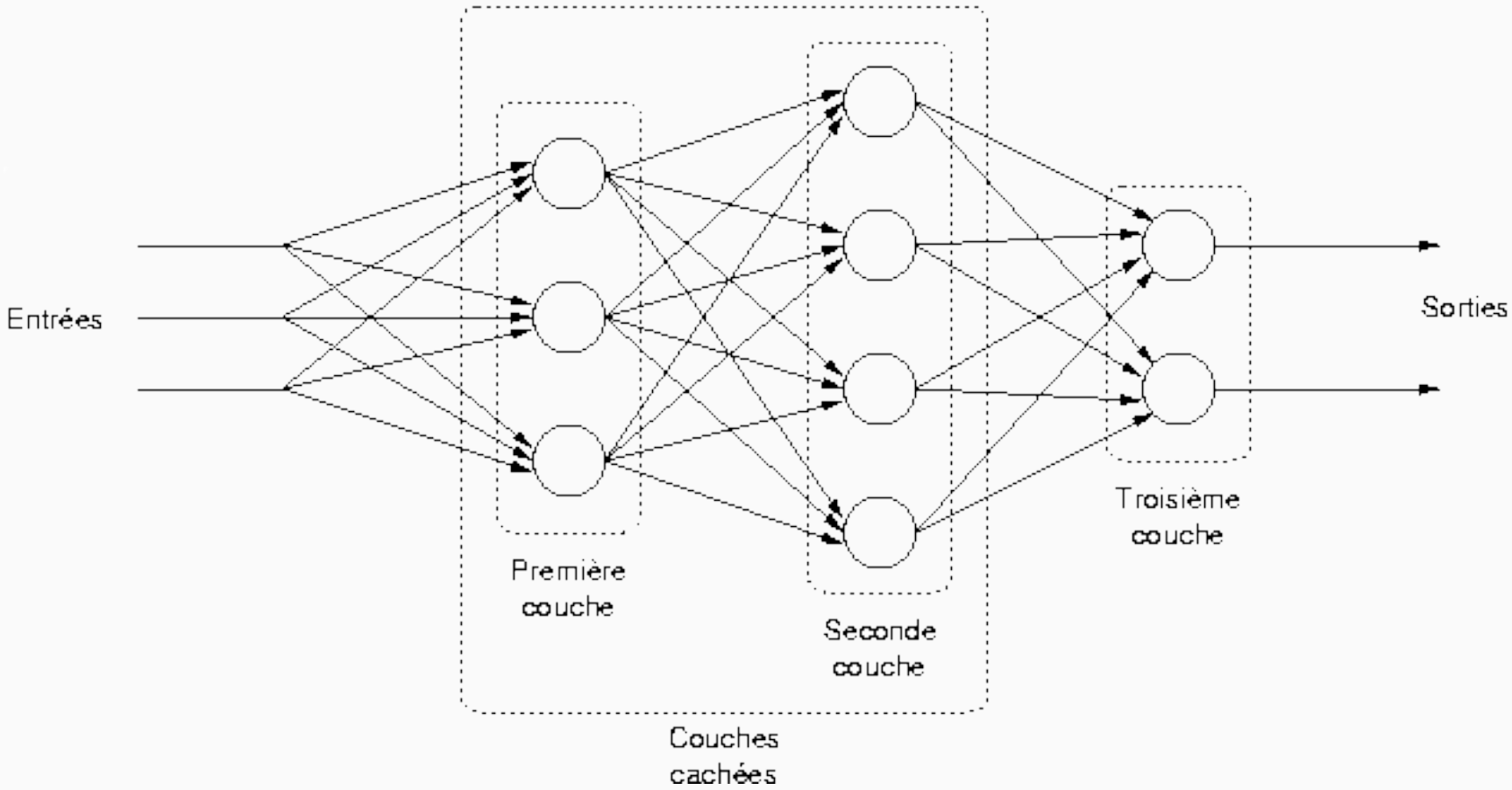
Algorithme d'apprentissage supervisé d'un réseau de neurones, à propagation avant, implémentant des classifieurs binaires.

Le cas le plus simple propose une sortie $s$ et $n$ entrées $e_1, \ldots, e_n$. On dispose d'un vecteur de $n$ coefficients *synaptiques* $w_i$, venant pondérer les entrées.

Le fonctionnement du perceptron est simple :

- On somme les entrées $e_i$ pondérées par les poids $w_i$ ;
- Puis on applique, sur ce résultat, une fonction d'activation, dont la sortie $s$ dépend d'une règle interne.

# Perceptron multicouches (MLP)



# Hidden Markov Model

Outil développé par L.E. Baum dans une série d'articles, dont [4, 5]. Voir [32]. Famille de *Mixture Models*.

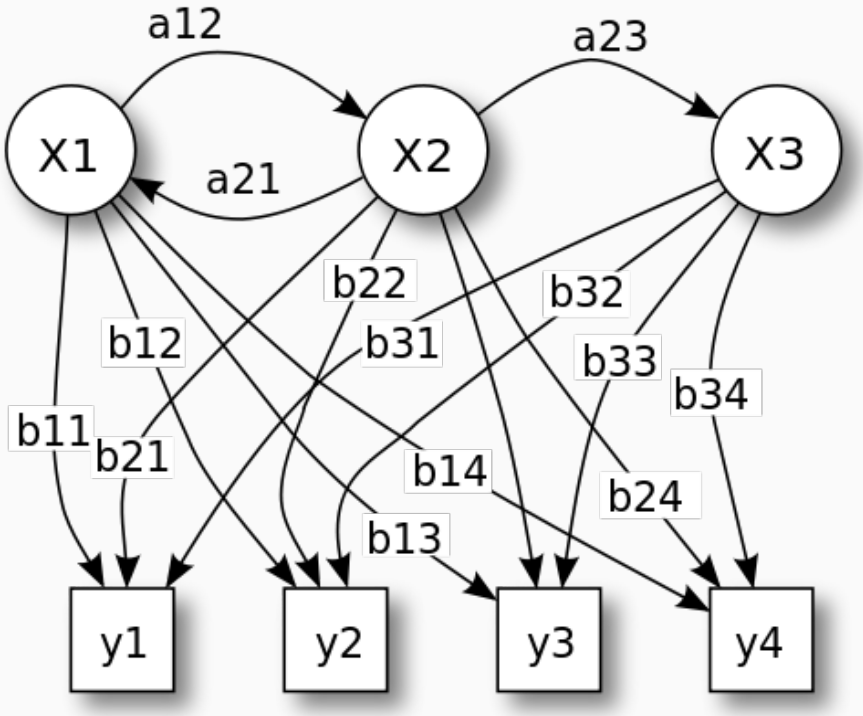
On observe la manifestation d'un processus qui nous est caché ; selon l'état de ce processus, les observations (valeurs, distributions) varient ; c'est donc un modèle non linéaire.

Trois manières d'utiliser ces modèles après apprentissage :

- calcul de la probabilité d'une séquence d'observations particulière sachant un modèle (*forward*)
- calcul de la séquence d'états cachés la plus probable pour une séquence d'observations (*viterbi*)
- trouver le meilleur modèle possible pour une séquence d'observations donnée (*baum-welch*)

Nombreux domaines d'application : reconnaissance vocale, de l'écriture, faciale, gestuelle, alignement de gènes, prévision de conflits, composition musicale, catégorisation et décodage de documents ...

# Hidden Markov Model




$M = \{A, B, \pi\}$, $A$ probabilités de transition de l'automate, $B$ probabilités de génération des symboles observés, $\pi$ probabilités de l'état initial

# Hidden Markov Model :: exemple modèle discriminant

On veut pouvoir détecter la présence d'une maladie, sur de l'imagerie. On a donc rassemblé un lot d'images "malade" et un lot d'images "sain".

Les observations sont les images, une image donne une séquence d'observations.

*Baum-Welch* va permettre d'entrainer deux modèles : un, noté $M_m$, pour "malade" et un, noté $M_s$, pour "sain".

Etant donnée une image $E$ inconnue, *Forward* va donner la probabilité que cette image puisse être générée par $M_m$, notée $P_m = P(E|M_m)$ et la probabilité que cette image puisse être générée par $M_s$ : $P_s = P(E|M_s)$.

Interprétation des états : on regarde les distributions de probabilité dans chaque état. *Viterbi* permet d'obtenir une séquence d'états la plus probable pour une image $E$, on peut reconstruire l'image d'après les états (et non d'après les observations) pour segmenter en régions.

# Hidden Markov Model :: exemple chemin discriminant

Mêmes conditions que précédemment.

*Baum-Welch* va permettre d'apprendre un modèle, d'après le lot "malade" et le lot "sain".

*Viterbi* va permettre de calculer les chemins d'états associés aux observations des lots appris

*Viterbi* va permettre de calculer le chemin d'états associé à $E$, qui sera ensuite comparé aux chemins d'états empruntés par $M_m$ et par $M_s$.

La classe de l'exemple, dont la distance au chemin suivi par $E$ est minimale, est la décision rendue par le modèle pour $E$.

Interprétation des états : même principe que précédemment, mais on aura des régions "saines" et des régions "pathologiques".

# k-NN – k-Nearest Neighbours

k-NN [2] (ou k-ppv – k-plus proches voisins) compare un exemple inconnu à une base d'exemples connus et étiquetés.

Classification : l'exemple inconnu est classé dans la catégorie majoritairement représentée parmi ses k plus proches voisins, au sens d'une fonction de distance

Régression : l'exemple inconnu prend la valeur moyenne, pondérée par la distance, de ses k plus proches voisins, toujours au sens d'une fonction de distance

Méthode non-paramétrique basée instance : il n'y a pas d'apprentissage, c'est la base des exemples et la fonction de distance qui remplissent ce rôle.

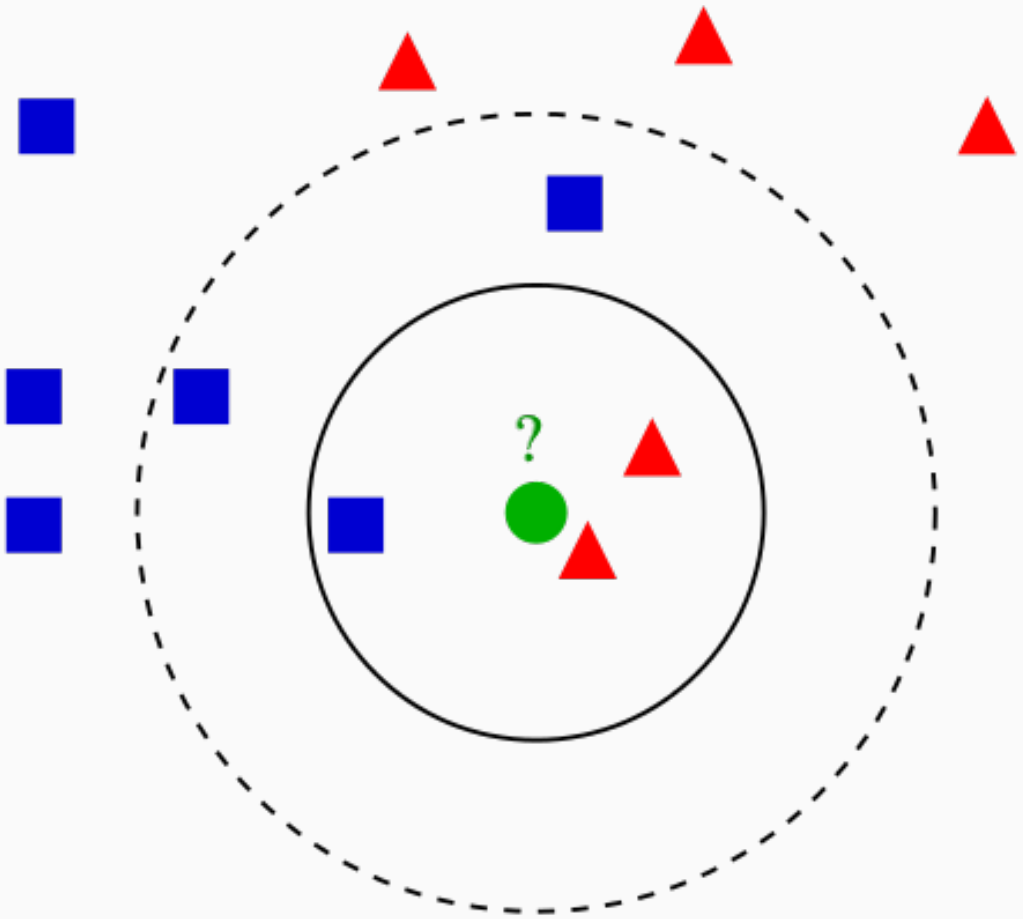
# k-NN – k-Nearest Neighbours

Difficultés

- définir une fonction de distance suffisamment pertinente
  - distance euclidienne, ou MAHALANOBIS[8] [28] pour des variables continues ;
  - distance de HAMMING [22] pour des variables discrètes ;
  - coefficients de PEARSON et SPEARMANS [23] ;
  - possibilité d'apprendre la distance également (NCA [20], LMNN [42])
  - ...
- choix de k (détermination expérimentale)

8. cette distance tient compte de la variance et de la corrélation des séries de données

# k-NN – k-Nearest Neighbours




Selon la valeur de k (le rayon du cercle) et la distance choisie (position des points les uns par rapport aux autres), la décision sera différente.

# Tout est affaire de distance . . .

Distance de Minkowsky d'ordre $p$ entre deux individus $\bar{x}_i$ et $\bar{x}_j$ :

$$d_p(\bar{x}_i, \bar{x}_j) = \Big(\sum_{k=1}^{m} |x_i^{(k)} - x_j^{(k)}|^p\Big)^{\frac{1}{p}}$$

où $x_i^{(k)}$ est le $k$-ième feature de l'individu $\bar{x}_i$

Le type de distance dépend de $p$ :

- distance de Manhattan : $p = 1$
- distance Euclidienne : $p = 2$
- distance de Chebyshev
  - $\lim_{p \to +\infty} d_p(\bar{x}_i, \bar{x}_j) = \max_{k=1}^{m} |x_i^{(k)} - x_j^{(k)}|$
  - $\lim_{p \to -\infty} d_p(\bar{x}_i, \bar{x}_j) = \min_{k=1}^{m} |x_i^{(k)} - x_j^{(k)}|$

# Algorithme des k-NN

**Données** :

- *etiquettes* : étiquettes des individus de la base de référence
- *individu* : exemple à classer
- $k$ : nombre de voisins à prendre en compte

**Résultat** :

- *classe* : étiquette de la classe proposée

**début**

  **pour** *chaque individu* $I_r \in$ *dans base d'apprentissage* **faire**

    dist[r] $\leftarrow$ distance (individu, $I_r$)

  **end**

  Trier conjointement (*dist*, *etiquettes*) sur la valeur de *dist* croissante

  *classe* $\leftarrow$ étiquette majoritaire dans *etiquettes*$[1 : k]$

  **retourner** *classe*

**end**

# k-NN : modèle non paramétrique

Il n'y a pas de paramètre à ajuster, en dehors de $k$. Cependant, en modifiant le contenu de la base d'apprentissage, on agit sur les performances du classifieur.

On procèdera donc selon :

- Construction / Choix de la distance sur la base d'apprentissage ;
- Choix de $k$ en comparant l'évolution de la performance pour différentes valeurs sur la base de validation ; et ajustement du contenu de la base d'apprentissage ;
- Evaluation de la performance sur la base de test.

Théorème de Cover-Hart [13] : soit $E$ le taux d'erreur optimal de prédiction, le taux d'erreur de la règle des k-ppv est $\leq 2.E$.

# Pourquoi ne pas s'arrêter là ?

Modèle non paramétrique, donc mémorise tout.

$\longrightarrow$ Problème de taille de stockage : trouver des modèles plus compacts

Complexité d'un $k$-ppv de taille $n$ est en général en $O(n.d + n.k)$ où $d$ est la complexité du calcul de la distance

$\longrightarrow$ Trouver des méthodes plus rapides

Point de fonctionnement : dans un espace de dimension $d$, il faut au minimum $2^d$ individus

$\longrightarrow$ Ainsi, pour 32 features (peu), il faut plus de 4 milliards d'individus.

# Néanmoins, il reste utilisable . . .

Algorithme simple à comprendre et à programmer

Notion de similarité facile à manipuler

À utiliser avec un petit nombre de features

Possibilité d'utiliser des techniques de réduction de dimensionnalité (P.C.A. par ex.)

Apprentissage possible de la métrique, notamment par *deep learning*

Travaux de 2012 [18] montrent que pour plusieurs classes de fonctions, 1-ppv est très performant.

*L'apprentissage non supervisé est applicable sur un nombre de modèles plus restreint :*

Clustering hiérarchique, k-means, k-medians, Algorithme EM, DBSCAN, OPTICS, Fuzzy C-Means, Self-Organizing Maps (SOM), ...

Ces algorithmes/modèles ont pour but de définir automatiquement des groupes d'individus. Seul $\mathbf{X}$ est connu.

# k-means

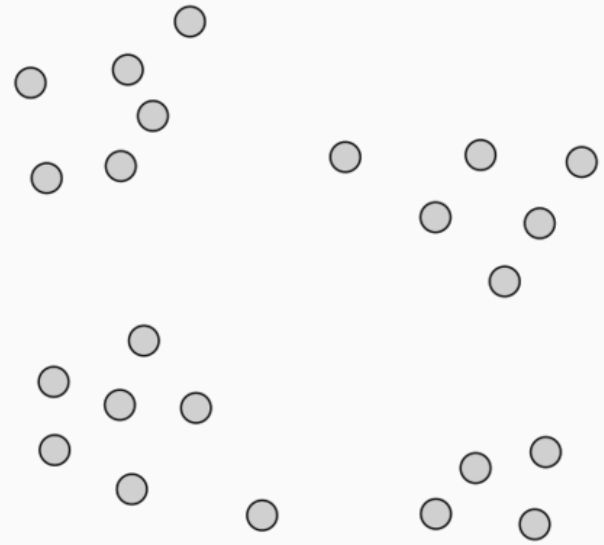
Idées posées en 1956 [36], algorithme en 1957, terminologie de 1967 [27], première publication *officielle* du papier de 1957 en 1982 [26] seulement.

Étant donné un ensemble de points et un entier $k$, comment partitionner cet ensemble en $k$ groupes, en minimisant une certaine fonction (par exemple le carré de la distance entre le centre d'un groupe et tous ses constituants) ?

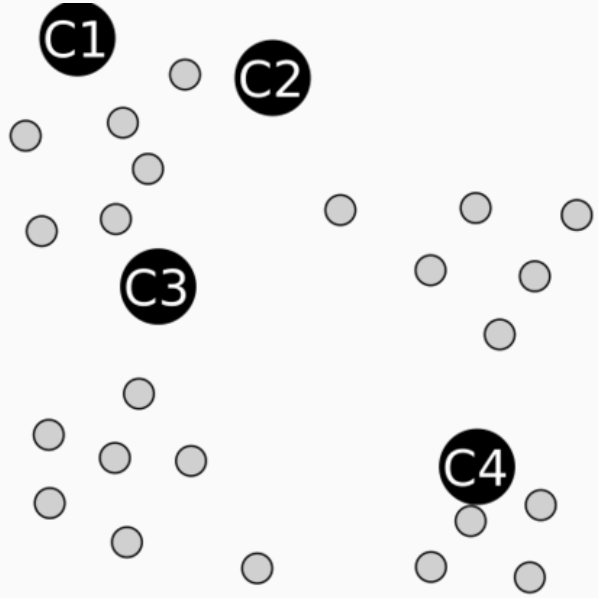
Très utilisé, bien que pas de garantie de l'optimalité, ni de la convergence. Généralisé par une autre méthode, appelée *nuées dynamiques*.

Très sensible au choix des germes (exemples choisis comme représentatifs d'une classe), à la fonction de similarité. La connaissance *a priori* de $k$ est aussi parfois un obstacle.

# k-means

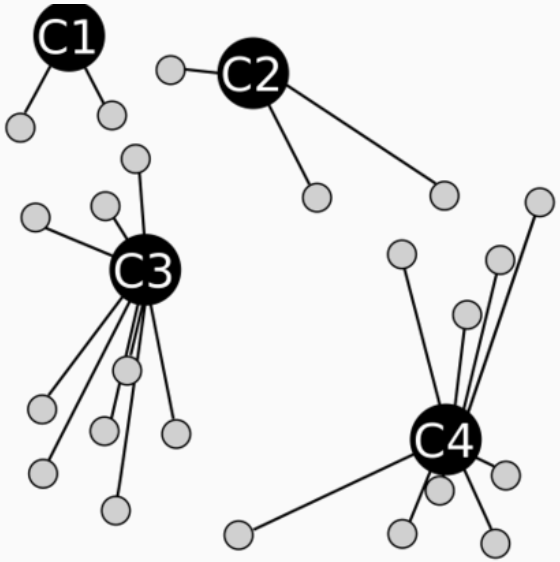


0a. Données  d'entrée

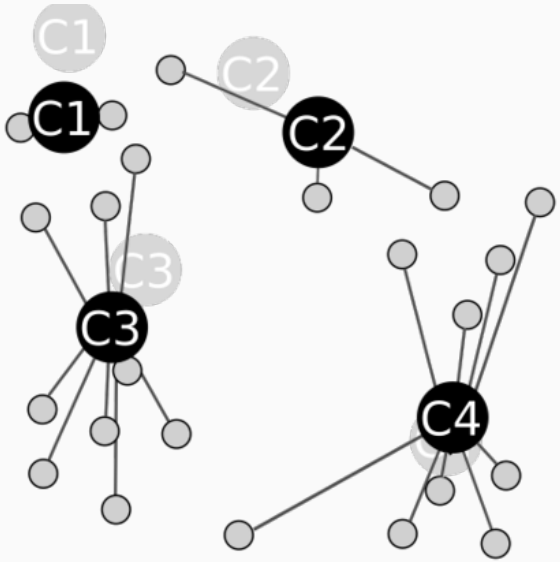



0b. intialisation

# k-means

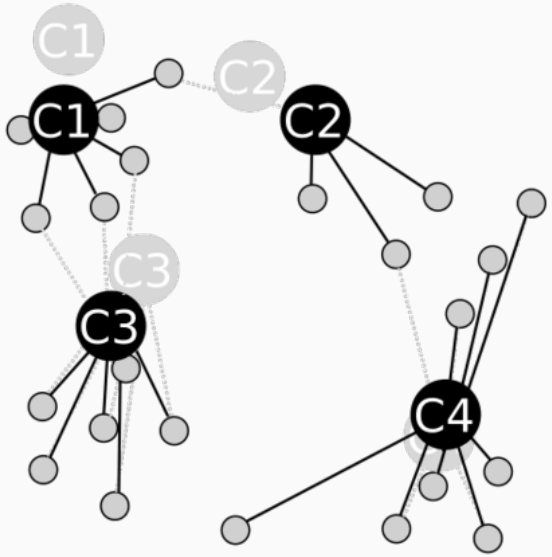



1a. assignation

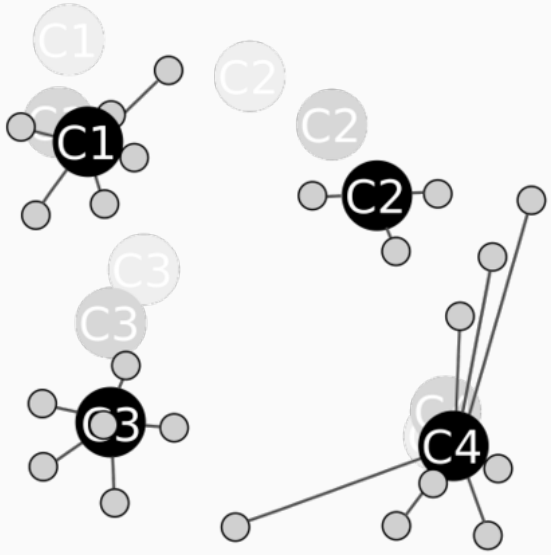



1b. calcul des points moyens

# k-means




2a. assignation




2b. calcul des points moyens

On réitère les phases d'assignation et de recalcul des barycentres jusqu'à stabilisation.

# Evaluation

# Protocole d'évaluation

Les données d'entrée, $E$, sont formées par un ensemble d'individus, mesurés sur le terrain [9]

On a construit notre système d'après un sous-ensemble de $E$ appelé base d'apprentissage, noté $B_A$.

Un second sous-ensemble de $E$, disjoint de $B_A$, sert à *régler* l'algorithme d'apprentissage : c'est la base de test, $B_T$.

Un dernier sous-ensemble de $E$, disjoint des deux précédents, permet d'évaluer les performances. C'est la base d'évaluation $B_E$.

Les trois bases sont disjointes, tout en étant représentatives [10]

9. Il arrive parfois qu'on synthétise des individus lorsque les mesures sont rares.
10. Le choix des constituants des bases constitue un biais important

# Protocole d'évaluation

On procède par validation croisée (ici en *tuning*) :

1. on choisit un nombre d'essais noté $k$ (typiquement 5) ;
2. on tire aléatoirement $1/k$ des individus de $B_T$ pour former $I$ ;
3. on calcule $T = \{B_T / I\}$
4. on fait l'apprentissage du modèle avec $T$
5. on évalue le modèle avec $I$
6. on répète les points 2-4 $k$ fois

Situation extrême : *leave-one-out*, où $k$ vaut la cardinalité de $B_T$.

# Erreur de prédiction

Chaque prédiction $\tilde{y}_i$ diffère de la valeur caractéristique $y_i$ de l'individu appris ; bien qu'on espère l'écart faible.

On apprécie cet écart au moyen d'une mesure d'erreur $e_m$, toujours positive ou nulle.

On évalue alors l'*erreur de prédiction* avec :

$$ERR_{\bar{\theta}} = \sum_{n}^{i=1} e_m(\tilde{y}_i, y_i)$$

Minimiser l'erreur est souvent difficile, car cela dépend de $\bar{\theta}$ qui influe lui-même sur l'hypersurface de classification.

# Erreur quadratique & *loss function*

Erreur quadratique moyenne (mean square error) :

$$MSE_{\bar{\theta}} = \frac{1}{n} \sum_{n}^{i=1} (\tilde{y}_i - y_i)^2$$

Plus généralement, on appelle *fonction de perte* (loss function) la relation qui exprime l'erreur commise par un modèle $\theta$ (à $B_A$ donnée).

Le M.L. vise à rechercher $\theta$ qui minimise cette fonction.

On fait aussi parfois un parallèle avec l'énergie cinétique :

$$E = \frac{1}{2} \sum_{n}^{i=1} e_m (\tilde{y}_i - y_i)^2$$

Le but de l'apprentissage est alors de minimiser cette énergie.

# Matrice de confusion

Quand les labels sont discrets, on compte le nombre de bonnes réponses.

On catégorise les réponses $\tilde{y}_i$ :

- On a $C$ classes apprises (et donc *a priori* $C$ classes à reconnaître) ;
- on consigne, dans un tableau, le nombre de réponses : classe $y_i$ en colonnes, classe $\tilde{y}_i$ en ligne.

On considère alors chaque décision prise par le modèle :

- $y_i$ attendu, $\tilde{y}_i$ obtenu : $+1$ sur la diagonale (ligne $i$/colonne $i$) ;
- $y_i$ attendu, $\tilde{y}_j$ obtenu : $+1$ à l'intersection de la ligne $j$ et de la colonne $i$.

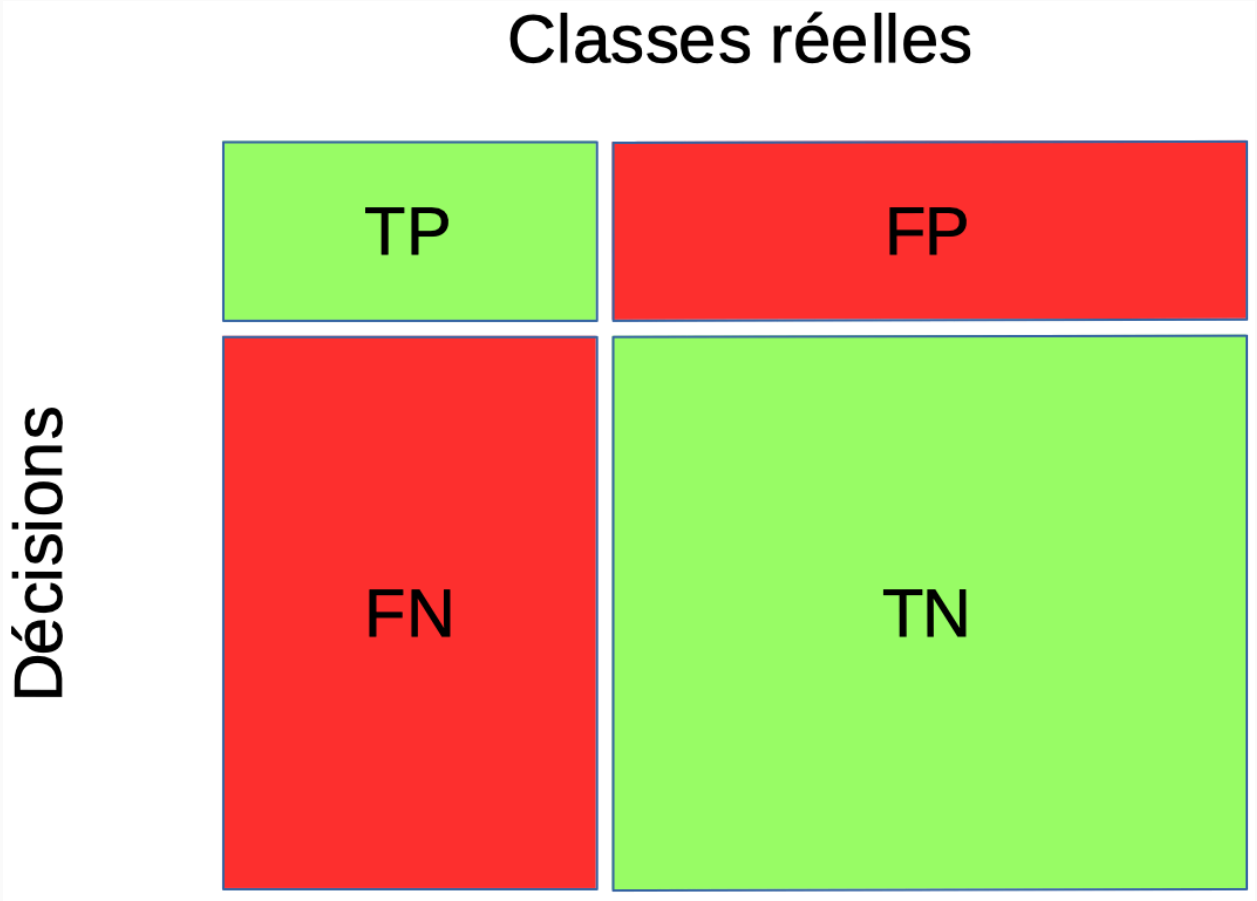
# Matrice de confusion

Si on considère maintenant une seule classe $y_i$ à reconnaître :

- sur la diagonale, intersection de la ligne $i$ et de la colonne $i$, le modèle a bien trouvé ce qu'on cherchait : *true positive* – $TP$ ;
- sur la ligne $i$, hors diagonale, le modèle s'est trompé en répondant la classe qu'on cherchait : *false positive* – $FP$ ;
- sur la colonne $i$, hors diagonale, le modèle s'est trompé en ne répondant pas la classe qu'on cherchait : *false negative* – $FN$ ;
- tout le reste, le modèle a bien répondu en ne répondant pas $\tilde{y}_i$ : *true negative* – $TN$ ;

# Matrice de confusion

Pour une $y_i$ à reconnaître :




Généralisation : moyenne des résultats

# Rappel - Précision

Rappel : Pourcentage d'exhaustivité lors de l'annonce d'une classe

$$R = \frac{TP}{TP + FN}$$

Précision : Pourcentage de vérité lors de l'annonce d'une classe

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

Le deux grandeurs varient en sens inverse, avec beacoup de rappel, on risque de citer des classes par erreur, donc précision moindre. *A contrario*, avec une forte précision on oublie les individus les plus tangents (ceux en marge des lois de probabilité) et donc le rappel est faible.

# Autres indicateurs

- True Positive Rate (TPR), Sensibilité : $\frac{TP}{TP+FN}$
- False Positive Rate (FPR) : $\frac{FP}{FP+TN}$
- True Negative Rate (TNR), Spécificité : $\frac{TN}{FP+TN}$
- Accuracy : $\frac{TP+TN}{\Sigma population}$
- F1 : $\frac{2}{\frac{1}{P}+\frac{1}{R}}$
- Prevalence : $\frac{TP+FN}{\Sigma population}$
- False Discovery Rate (FDR) : $\frac{FP}{TP+FP}$
- False Omission Rate (FOR) : $\frac{FN}{TN+FN}$
- ...

# Courbe ROC

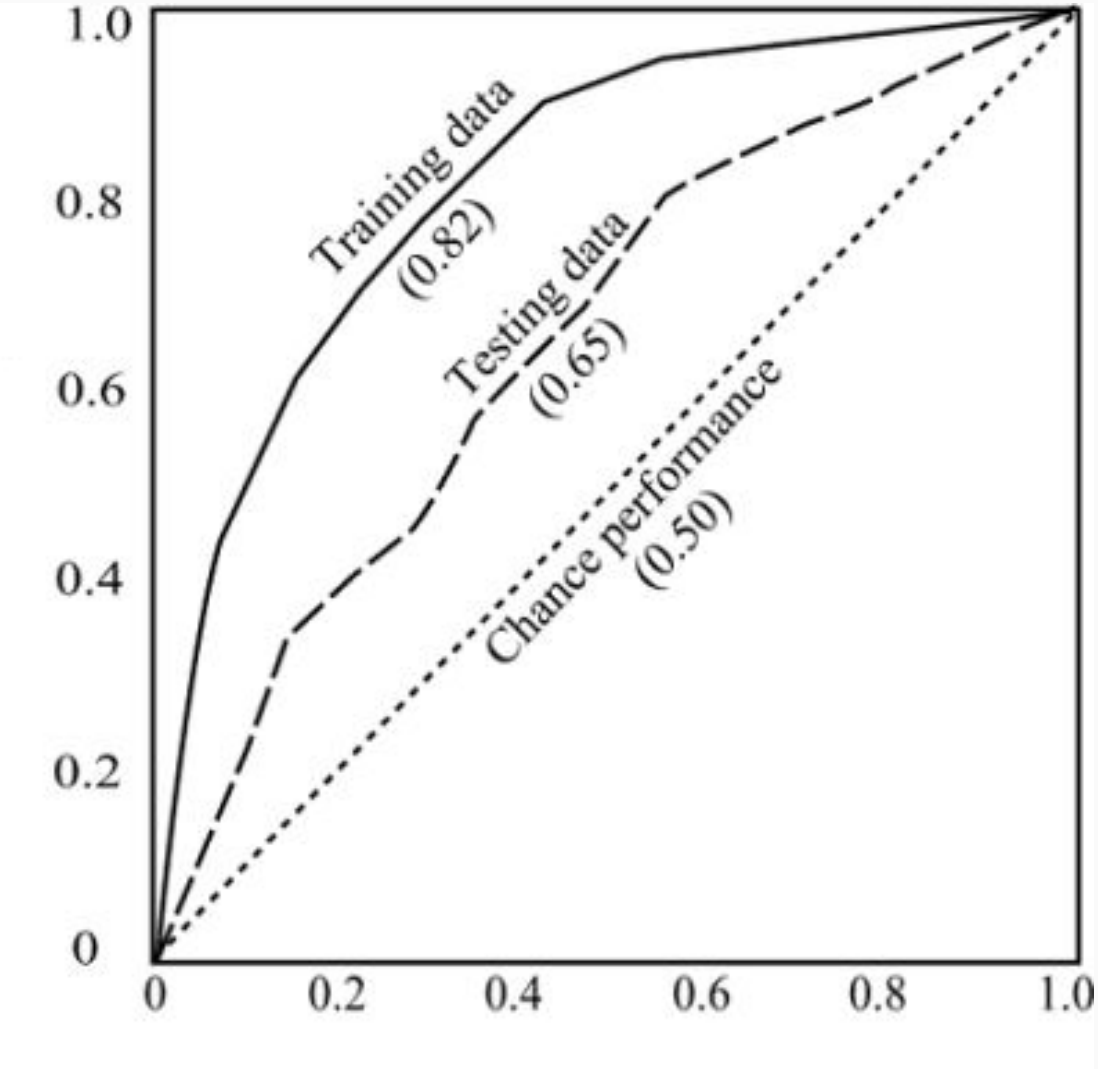
ROC : receiver operating characteristic

Origine : 2nde guerre mondiale, séparation de signaux radar / bruit de fond

Graphe qui représente $TPR = f(FPR)$

Hasard = 1ere diagonale, la courbe doit se situer au dessus, être croissante, et avoir dès le début une forte altitude.

# Courbe ROC


1.0
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1.0
Training data
(0.82)
Testing data
(0.65)
Chance performance
(0.50)

# Difficultés d'apprentissage

# La machine compte en base 2

À la base, l'information est de nature électrique. Dans la machine, *les données se déplacent ensembles*, dans des groupes de fils électriques appelés *bus*.

Donc, à un instant donné, dans un fil donné du bus, soit il y a du courant, soit il n'y en a pas [11].

Cette situation est représentée de façon théorique par un état à deux valeurs possibles : 0 ou 1. Contraction de *binary digit*, le *bit* est le plus petit élément d'information dans la machine. On les utilise par groupe de 8 (appelé *octet* ou *byte*).

Les nombres, entiers ou réels, ne sont donc qu'une combinaison, selon une convention donnée, d'un certain nombre de ces bits.

---

11. tout cela évoluera sans doute dans le futur avec des machines totalement quantiques, aussi bien au niveau processeur qu'au niveau mémoire [7], le 9 mars 2020, @GoogleAI tweetait sur la sortie de TensorFlow Quantum

# La machine peut, à peu près, compter en base 10

*Nos* entiers (en base 10) se déduisent bien de la représentation en base 2.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| En binaire : | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| Poids en décimal : | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| En décimal : | $2^7$ + | $2^6$ + | | + $2^4$ + | | + $2^2$ + | $2^1$ + | |

128 + 64 + 16 + 4 + 2 = 214

Mais l'ensemble de définition est limité par la largeur des registres du processeur. On ne peut donc pas représenter, facilement, un nombre aussi grand que l'on veut.

# La machine a plus de mal avec les réels [19]

Les réels sont codés, entre autres, à partir du même principe, mais les quantités pondérées par les bits sont portées au dénominateur d'une fraction.

On a donc une quantité définie par $\frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} \ldots$

Pour les entiers, chaque nombre peut être codé exactement, il suffit d'avoir la quantité de bits suffisante.

Pour les réels, intuitivement, on s'aperçoit qu'il va y avoir des valeurs, et *rien* entre ces valeurs : la limitation du nombre de bits a comme conséquence l'impossibilité de représenter un nombre exactement.

Les algorithmes de calcul intégrés au processeur réalisent donc des arrondis, assimilables à des incertitudes de mesure. La difficulté étant parfois que ces incertitudes se neutralisent au lieu de se s'accumuler.

# Overflow et Underflow

On appelle :

- *overflow* le dépassement de la capacité de représentation de la machine, vers la plus grande valeur (en valeur absolue).
  Impossibilité de calcul quand les nombres tendent vers $+\infty$ ou $-\infty$.
  Selon les plateformes, le résultat du calcul est *+inf, -inf, NaN*...
- *underflow* le dépassement de la capacité de représentation de la machine, vers la plus petite valeur (en valeur absolue).
  Impossibilité de calcul quand les nombres tendent vers 0.
  Le résultat du calcul est *0*

# Présentation des calculs

La difficulté peut venir de la manière dont les calculs sont écrits.

Par exemple : $2 \times \frac{1}{3} - \frac{2}{3}$ est calculé comme :

$$2 \times 0,33\ldots33 - 0,66\ldots67 = -0,00\ldots01$$

L'exemple est basique, mais laisse présager bien d'autres problèmes.

La réalisation informatique des algorithme demande donc une grande connaissance du codage des nombres [19] et de la manière dont la machine va réaliser les opérations.

# Stratégies d'évitement

Une première approche consiste à corriger le calcul tout en conservant *sa logique* (avec le risque qu'il sera peut-être impossible de projeter le résultat dans l'espace initial).

Exemples :

- Dans l'algorithme *forward*[12] (HMM), on peut normaliser le calcul, en divisant, à chaque étape, le résultat par une quantité donnée, de façon à ce que le calcul reste dans une fourchette acceptable. On obtient finalement un résultat et un coefficient de correction.
- On peut aussi faire le calcul dans un espace logarithmique, et rester dans cet espace pour comparer les résultats entre eux.

12. calcul de la vraisemblance d'une observation

# Stratégies d'évitement

Une autre approche consiste à utiliser un framework de calcul [13], dit *en précision arbitraire*

- représentation des nombres venant se substituer à celle de la machine, opérateurs adaptés ;
- temps de calcul allongé ;
- résultat est numériquement stable, reproductible, et juste.

Il est parfois impossible d'éviter l'*underflow*. On met alors en place un mécanisme de détection et on peut instaurer une valeur plancher en dessous de laquelle une quantité ne peut descendre [32].

13. scipy en Python, MPFR en C/C++, BigDecimal en Java

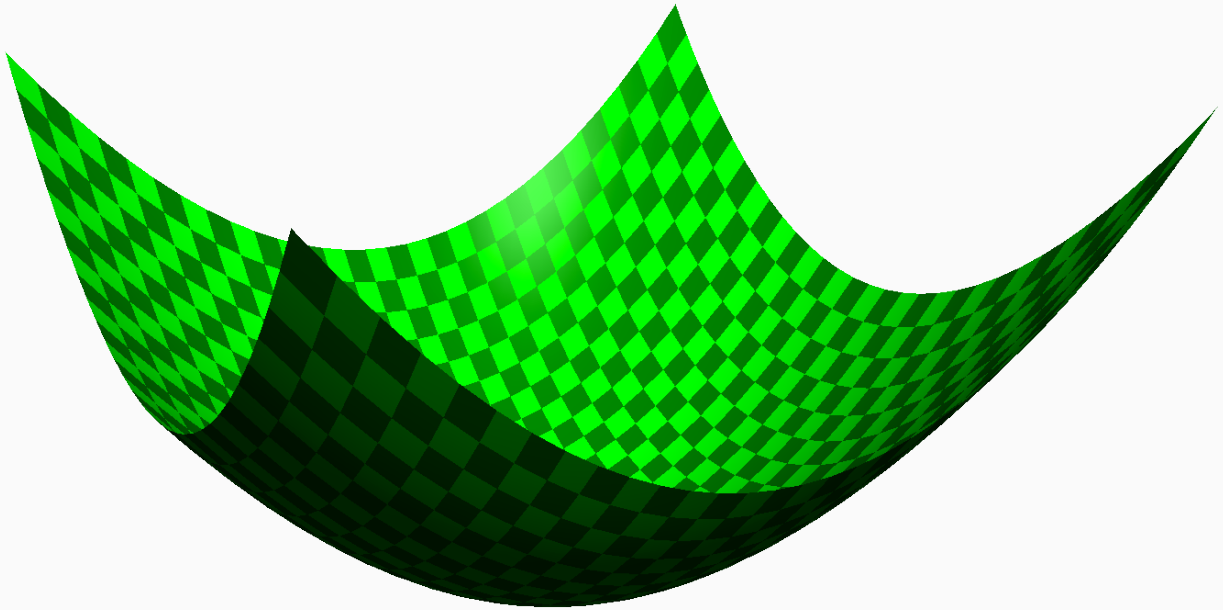
Bien que les techniques de recherche de solution soient performantes, elles n'en restent pas moins imparfaites.

Cela tient aussi au fait que la fonction à optimiser (*loss function*) peut être particulièrment complexe.

Il n'est donc pas toujours facile de trouver une bonne solution dans un espace très vaste.

# Et si la *loss function* était convexe ...

- $\forall x, y, \mu, \lambda, \mu + \lambda = 1 \Longrightarrow \lambda f(x) + \mu f(y) \geq f(\lambda x + \mu y)$
- la descente de gradient converge vite vers l'optimum



*Exemple* : $f(x, y) = x^2 + y^2$

# La réalité . . .

- la fonction est (on l'imagine) très complexe, et donc sans doute pas convexe ;
- pleine de creux et de bosses ;
- impossible à se représenter (un seul neurone avec 2 entrées utilise un espace de dimension 3) ;

La descente de gradient (ou tout autre technique) a toutes les chances de *tomber* dans un minimum local.

De plus, trouver le minimum global est un problème NP-difficile.

On se contente donc de chercher un *bon* minimum local.

# Dimensionnalité importante

Le nombre de paramètres dans $\theta$ joue aussi un rôle clé :

- beaucoup de paramètres $\Leftrightarrow$ performance espérée importante ;
- beaucoup de paramètres $\Leftrightarrow$ un grand nombre de directions à emprunter pour la descente de gradient.

Il est donc très probable ; qu'étant donné la complexité de la fonction de perte, peu de chemins mènent à un optimum, même local.

La progression de la qualité [14] du modèle est alors lente et faible.

14. M.S.E. faible

# Apprendre à relativiser !

Des travaux de 2015 [11] ont montré, sur une catégorie de CNN, que :

- les minimas locaux sont assez localisés dans l'espace de recherche ;
- la descence de gradient stochastique ou le recuit simulé permettaient d'atteindre ces points
- la performance du modèle, dans cette zone, était très correcte ;
- le minimum global était difficile à atteindre, et menait souvent à l'overfitting.

Ne pas oublier, non plus, que le paysage de la fonction est déterminé par les données : l'optimum global est donc une position relative aux données, destiné à se déplacer, car les systèmes performants sont conçus pour intégrer de nouvelles données après un premier apprentissage.

L'important est surtout de (bien) progresser dans l'apprentissage

# Underfitting & Overfitting

Dans l'idéal, le modèle représente bien les lois en présence dans la base d'apprentissage ; et la base est très représentative de la réalité.
Cependant ...

overfitting : le modèle est trop précis, il a une très bonne performance en apprentissage, mais il ne classe pas correctement de nouveaux points car l'hypersurface qu'il modélise ne suit pas la tendance des données, elle ne fait que passer par des points remarquables (ceux de l'apprentissage).
Corolaire : le modèle explique trop parfaitement des données, même incertaines ; il a donc tendance à trouver une explication pour des données produites *au hasard*.

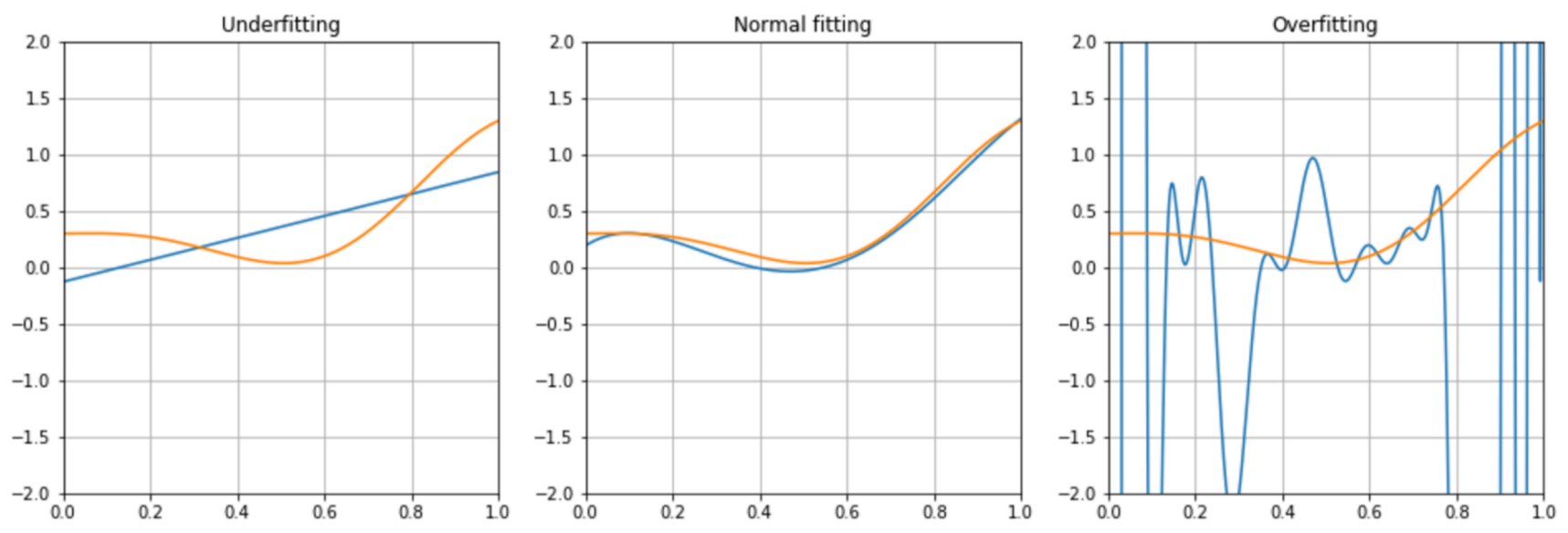
underfitting : le modèle est trop simple pour capturer la dynamique des $X$ ; sont taux d'erreur est élevé dès l'apprentissage.

# Compromis biais - variance

Biais : la quantité d'erreur introduite par approximation du réel ; c'est donc un manque de représentativité des données ; peut être *compensé* par plus de features (une approximation moins grossière), ou un modèle plus complexe (mais qui peut entrainer l'overfitting) ;

Variance : traduit la variation du résultat en fonction des données d'entrée ; c'est donc le manque de stabilité du modèle face aux données d'entrée, la manière dont la *loss function* est transformée lorsqu'on change les données d'entrée ;

# Underfitting & Overfitting




(source : [8]) Fort biais = underfitting, forte variance = overfitting

# Eviter cela ?

Plusieurs voies sont privilégiées :

- On a besoin de beaucoup d'individus, au regard des features à apprendre : on considère (cela dérive d'un théorème) que si on a $p$ features, il faut $n$ individus tel que $n \geq 100.p$ (avec la difficulté qu'il est parfois impossible d'avoir $n$ aussi grand issu du terrain).
- On multiplie les individus en introduisant des variations sur les features (on fabrique de l'incertitude sur les mesures)
- On multiplie les avis : on entraine plusieurs classifieurs, dans diverses conditions, et on se fie à l'avis résultant (Random Forest – 2001)
- On entraine un modèle pour une tâche, et un second est entrainé à évaluer cette tâche (Generative Adversarial Network – 2014)

# Le mot de la fin

# L' I.A. actuelle reste faible...

L'I.A. n'a pas d'*a priori*. Les machines réalisent des calculs et obtiennent des résultats. On n'a donc pas à avoir peur des machines en tant que telles.

Par contre, il est fondamental de contrôler comment elles sont alimentées, comment elles sont programmées, et ce qu'on fait de ces résultats. L'Homme, même en pensant faire bien, reste notre plus grand danger !

Les algorithmes ne sont pas neutres, ils reflètent l'orientation choisie par ceux qui les ont conçus. Les biais sont engendrés par ces choix [30, 39, 1].

# Mais elle tend à *progresser*...

La recherche progresse chaque jour, dans sa compréhension des mécanismes de la nature (bio-inspiré), mais aussi dans l'invention de nouveaux mécanismes.

Certains pensent que l'I.A. forte sera une réalité, mais pas en tant qu'aboutissement de l'I.A. faible [9].

D'autres pensent qu'on ira plutôt vers une I.A. dont l'intelligence sera comparable à celle de l'Homme, mais que cela résultera d'une approche scientifique, et non technologique [17].

Les conséquences sur l'Humanité sont multiples, certains redoutent sa disparition, d'autres parient plutôt sur une disparition de l'*individu autonome* [3].

# Conclusion

Les I.A. actuelles apprennent ce qu'on leur enseigne. Si on choisi mal les exemples, elles n'auront pas l'effet escompté. C'est donc toujours l'humain qui donne la direction, et les biais ! L'I.A. n'est pas, actuellement, comparable à l'intelligence humaine.

Il est donc important que les systèmes basés I.A. soient transparents et traçables, car ils sont de plus en plus complexes, ce qui limite leur compréhension de la part du grand public, et génère un sentiment de défiance.

Les systèmes basés sur l'I.A. doivent rester sous notre contrôle : l'Humain ne peut pas être exclu, en l'état actuel des choses, de processus de décision importants. Relisez [31]

# Questions ?

# Licence



# References i


[1] Marc schoenauer : évolution darwinienne et ia, March 2020.

[2] N. S. Altman.
**An introduction to kernel and nearest-neighbor nonparametric regression.**
*The American Statistician.*, 46(3) :175–185, 1992.

[3] K. Badeau.
**Intelligence artificielle : peur sur le moi.**
*Les Echos*, 2019.

# References ii


[4] L. E. Baum and J. A. Eagon.
**An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology.**
*Bulletin of the American Mathematical Society*, 73(3) :360–363, 1967.

[5] L. E. Baum and T. Petrie.
**Statistical inference for probabilistic functions of finite state markov chains.**
*The Annals of Mathematical Statistics*, 37(6) :1554–1563, 1966.

[6] A. Ben-Hur, D. Horn, H. Siegelmann, and V. N. Vapnik.
**Support vector clustering.**
*Journal of Machine Learning Research*, 2 :125–137, 2001.

# References iii


[7] M. Benoit.
**Des physiciens ont réussi à mesurer le spin d'un électron sans le "détruire" : un pas de plus vers l'ordinateur quantique ?, April 2020.**

**[8]** G. Bonaccorso.
***Machine Learning Algorithm*.**
Packt, 35 Livery Street, Birmingham, UK, 2017.

[9] N. Bostrom.
***Superintelligence*.**
ISBN 978-2100764860, 2017.

[10] T. Brouard.
***Object Modeling, Algorithms and Applications*, chapter 1, pages 3–15.**
Research Publishin, ISBN 978-981-08-5465-2, 2010.

# References iv

[11] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun.
**The Loss Surfaces of Multilayer Networks.**
In G. Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 192–204, San Diego, California, USA, 09–12 May 2015. PMLR.

[12] C. Cortes and V. N. Vapnik.
**Support-vector networks.**
*Machine Learning*, 20(3) :273–297, 1995.

[13] T. Cover and P. Hart.
**Nearest neighbor pattern classification.**
*IEEE Transactions on Information Theory*, 13 :21–27, 1967.

[14] J. De Fauw, J. R. Ledsam, B. Romera-Paredes, S. Nikolov, N. Tomasev, S. Blackwell, H. Askham, X. Glorot, B. O'Donoghue, D. Visentin, G. van den Driessche, B. Lakshminarayanan, C. Meyer, F. Mackinder, S. Bouton, K. Ayoub, R. Chopra, D. King, A. Karthikesalingam, C. O. Hughes, R. Raine, J. Hughes, D. A. Sim, C. Egan, A. Tufail, H. Montgomery, D. Hassabis, G. Rees, T. Back, P. T. Khaw, M. Suleyman, J. Cornebise, P. A. Keane, and O. Ronneberger.
**Clinically applicable deep learning for diagnosis and referral in retinal disease.**
*Nature Medicine*, 24(9) :1342–1350, 2018.

# References vi

[15] C. Destrieux, I. Zemmoura, B. Serres, G. Venturini, D. Bourry, S. Velut, F. Andersson, and J.-P. Cottier.
**FIBRATLAS : a novel method for visualization of white matter tracts dissection. Preliminary result.**
In *16th Annual Meeting of the Organization for Human Brain Mapping*, pages –, Barcelone, Spain, June 2010.

[16] G. Galisot, T. Brouard, J.-Y. Ramel, and E. Chaillou.
**Image segmentation using local probabilistic atlases coupled with topological information.**
In *12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, volume 4, pages 501–508, 2017.

[17] B. Georges.
**Yann le cun : "les machines vont arriver à une intelligence de niveau humain".**
*Les Echos*, 2019.

[18] T. Gneiting.
**On the coverhart inequality : What's a sample of size one worth ?**
*Stat*, 1(1) :12–17, 2012.

[19] D. Goldberg.
**What every computer scientist should know about floating-point arithmetic.**
*ACM Computing Surveys*, 23(1) :5–48, 1991.

# References viii

[20] J. Goldberger, G. Hinton, S. Roweis, and R. Salakhutdinov.
**Neighbourhood components analysis.**
*Advances in Neural Information Processing System*, 17 :513–520, 2005.

[21] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik.
**Gene selection for cancer classification using support vector machines.**
*Machine Learning*, 46(1-3) :389–422, 2002.

[22] R. Hamming.
**Error-detecting and error-correcting codes.**
*Bell System Technical Journal*, 29(2) :147–160, 1950.

[23] P. A. Jaskowiak and R. J. G. B. Campello.
**Comparing correlation coefficients as dissimilarity measures for cancer classification in gene expression data.**
In *Brazilian Symposium on Bioinformatics*, 2011.

[24] J.-B. Lamy, B. Sekar, G. Guezennec, J. Bouaud, and B. Séroussi.
**Explainable artificial intelligence for breast cancer : A visual case-based reasoning approach.**
*Artificial Intelligence in Medicine*, (94) :42–53, 2019.

[25] J. Lettvin, H. Maturana, W. McCulloch, and W. Pitts.
**What the frog's eye tells the frog's brain.**
In *Proceedings of the IRE*, volume 47, pages 1940–51, 1959.

[26] S. P. Lloyd.
**Least squares quantization in pcm.**
*IEEE Transactions on Information Theory*, 28(2) :129–137, 1982.

[27] J. B. MacQueen.
**Some methods for classification and analysis of multivariate observations.**
In U. of California Press, editor, *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[28] P. C. Mahalanobis.
**On the generalised distance in statistics.**
*Proceedings of the National Institute of Sciences of India*, 2(1) :49–55, 1936.

[29] C. McFadden.
**Doctors in china are using ai to screen covid-19 patients, March 2020.**

**[30]** C. O'Neil.
***Algorithmes : la bombe à retardement*.**
ISBN 978-2352049807, 2018.

[31] A. Orsini.
**Les "23 principes d'asilomar" veulent encadrer le développement de l'intelligence artificielle.**

**[32]** L. R. Rabiner.
**A tutorial on hidden markov models and selected applications in speech recognition.**
*Proceedings of the IEEE*, 77(2) :257–286, Feb. 1989.

[33] L. Rosasco, E. D. De Vito, A. Caponnetto, M. Piana, and A. Verri.
**Are loss functions all the same ?**
*Neural Computation.*, 16(5) :1063–1076, 2004.

[34] F. Rosenblatt.
**The perceptron : a probabilistic model for information storage and organization in the brain.**
*Psychological Review*, 65(6), 1958.

[35] B. Serres, I. Zemmoura, F. Andersson, C. Tauber, C. Destrieux, and G. Venturini.
**Brain Virtual Dissection and White Matter 3D Visualization.**
In *The Medicine Meets Virtual Reality Conference*, volume 184, pages 392–396, San Diego, United States, Feb. 2013.

[36] H. Steinhaus.
**Sur la division des corps matériels en parties.**
*Bull. Acad. Polon. Sci.*, 4(12) :801–804, 1957.

[37] S. M. Stigler.
**The epic story of maximum likelihood.**
*Statistical Science*, 22(4) :598–620, 2007.

[38] N. Tomašev, X. Glorot, J. W. Rae, M. Zielinski, H. Askham, A. Saraiva, A. Mottram, C. Meyer, S. Ravuri, I. Protsyuk, A. Connell, C. O. Hughes, A. Karthikesalingam, J. Cornebise, H. Montgomery, G. Rees, C. Laing, C. R. Baker, K. Peterson, R. Reeves, D. Hassabis, D. King, M. Suleyman, T. Back, C. Nielson, J. R. Ledsam, and S. Mohamed.
**A clinically applicable approach to continuous prediction of future acute kidney injury.**
*Nature*, 572(7767) :116–119, 2019.

[39] M. Tual.
**Cathy O'Neil : Les algorithmes exacerbent les inégalités, Nov. 2018.**

**[40]** A. M. Turing.
**Computing machinery and intelligence.**
*Mind*, 49 :433–460, 1950.

[41] V. Vapnik.
***Statistical Learning Theory*.**
John Wiley and Sons, 1998.

[42] K. Q. Weinberger, J. C. Blitzer, and L. K. Saul.
**Distance metric learning for large margin nearest neighbor classification.**
*Advances in Neural Information Processing Systems*, 18 :1473–1480, 2006.