

# Comment les maths peuvent-elles aider les machines à apprendre ?

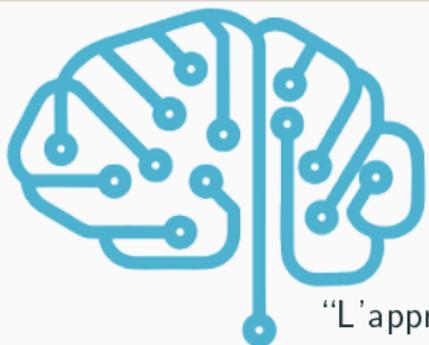
Apprentissage statistique séquentiel

---

Aurélien Garivier

1<sup>er</sup> octobre 2018

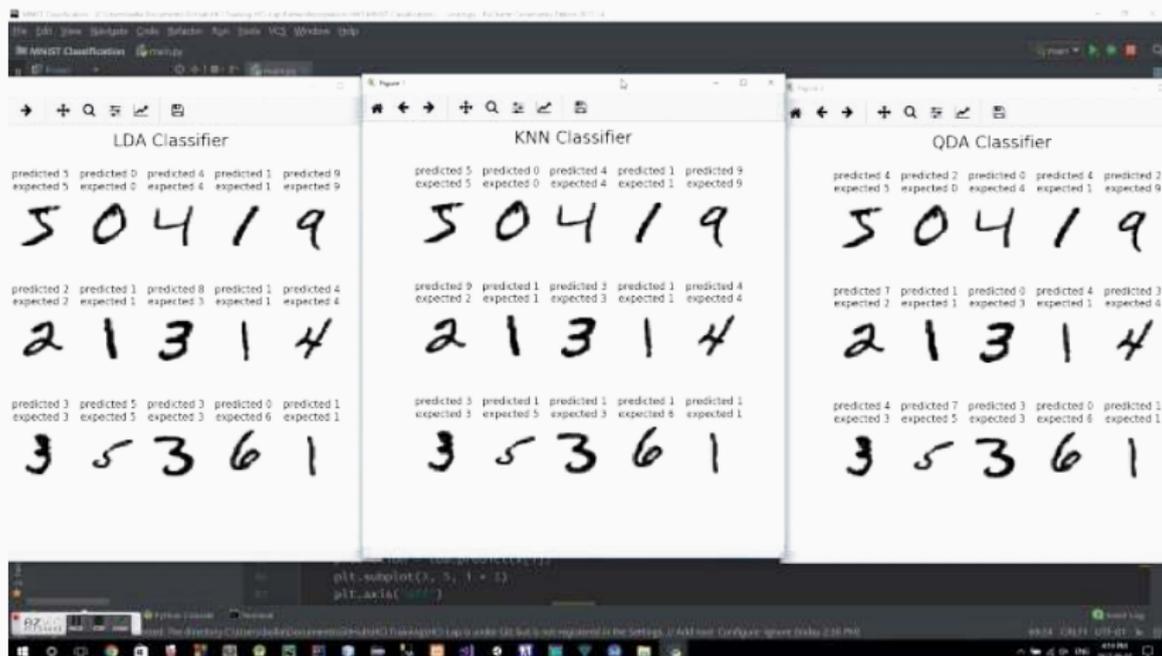
Réunion de rentrée de l'UMPA



**WIKIPÉDIA**  
*L'Encyclopédie libre*

“L'apprentissage automatique (en anglais machine learning, littéralement « l'apprentissage machine ») ou apprentissage statistique, champ d'étude de l'intelligence artificielle, concerne la conception, l'analyse, le développement et l'implémentation de méthodes permettant à une machine (au sens large) d'évoluer par un processus systématique, et ainsi de remplir des tâches difficiles ou problématiques par des moyens algorithmiques plus classiques.”

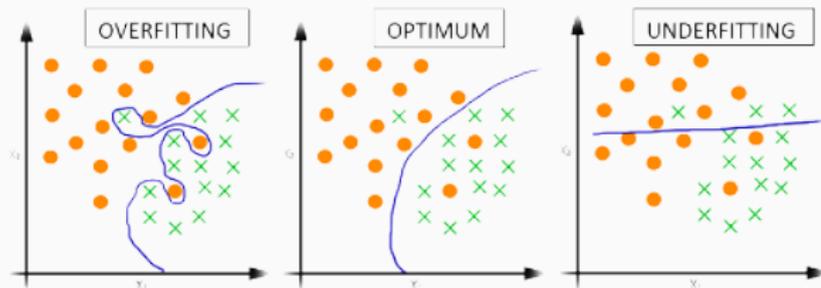
# Exemple : reconnaissance de caractères



Algorithme de classification supervisée :

- entrée : un jeu d'exemples  $(im_1, lab_1), \dots, (im_n, lab_n)$
- sortie : règle de décision = **fonction**  $im \mapsto lab$

# Pourquoi des mathématiques ?



- chercher les fonctions dans les bonnes classes  
⇒ **approximation + optimisation**
- éviter la *superstition* (=sur-apprentissage) et le *scepticisme* (=sous-apprentissage)  
⇒ séparer le **signal** et le **bruit**  
apprentissage **statistique** : marche sur des **modèles aléatoires** de bruit
- comprendre les **limites** de ce qui peut être fait par des algorithmes  
⇒ bornes inférieures

- Magistère **math-info**, agreg maths option probas-stat
- Thèse en **théorie de l'information**, avec E. Gassiat (maths) et S. Boucheron (à l'époque info)
- Travaux aussi en **statistique mathématique** (modèles markoviens)
- CR CNRS sur un poste 1  $\rightarrow$  7, en poste à Telecom Paris
- Orientation progressive vers l'apprentissage statistique

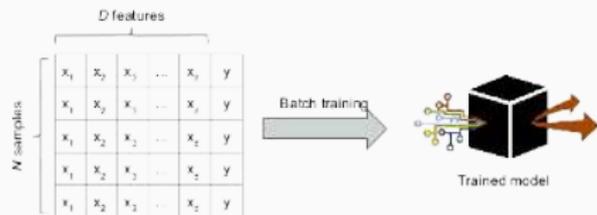
$\implies$  **Approche "modélisation aléatoire" de l'apprentissage**

$\neq$  optimisation, théorie du signal, algorithmique, logique/symbolique, fouille de données,...

# Cadre batch vs séquentiel

Cadre classique = "**batch**" :

- le jeu d'apprentissage est disponible a priori,
- on fait tourner l'algorithme d'apprentissage pour trouver la "bonne" règle de décision,
- ensuite on l'applique éternellement...

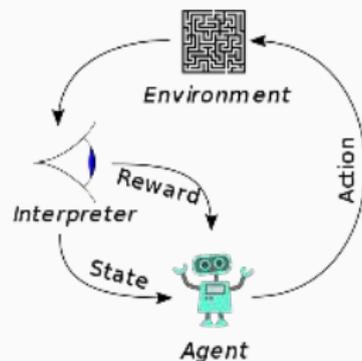


Cadre **séquentiel** :

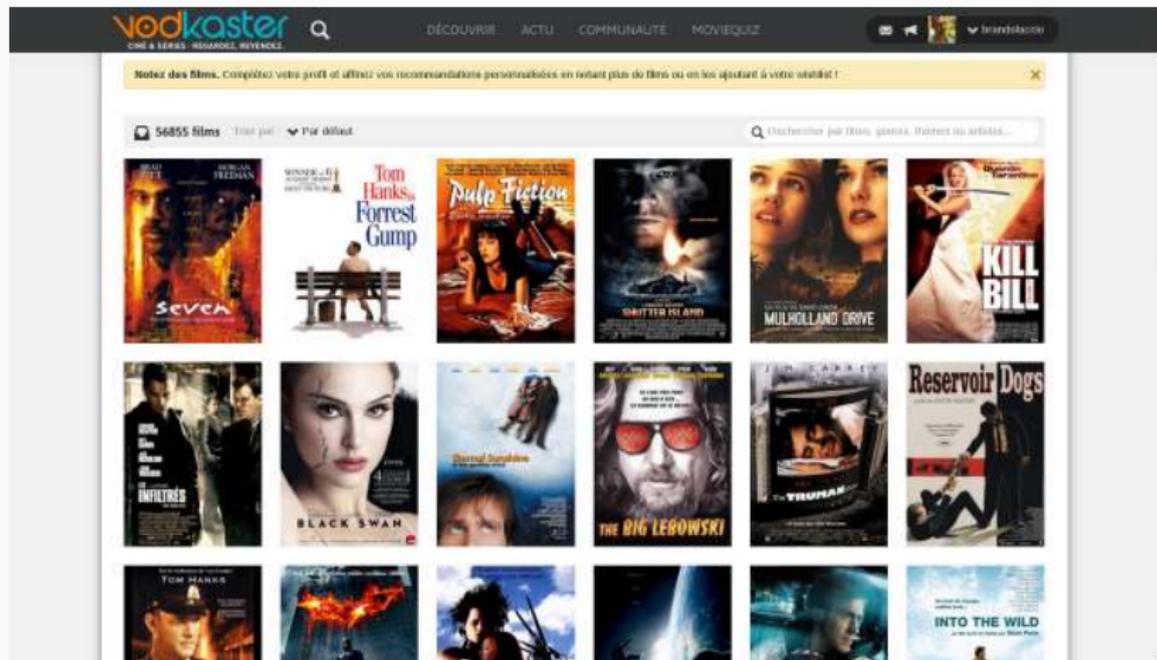
à chaque instant  $t$

- on prend une décision,
- on observe la conséquence de cette décision,

le but étant d'être efficace en moyenne à long terme OU d'être efficace à la fin.



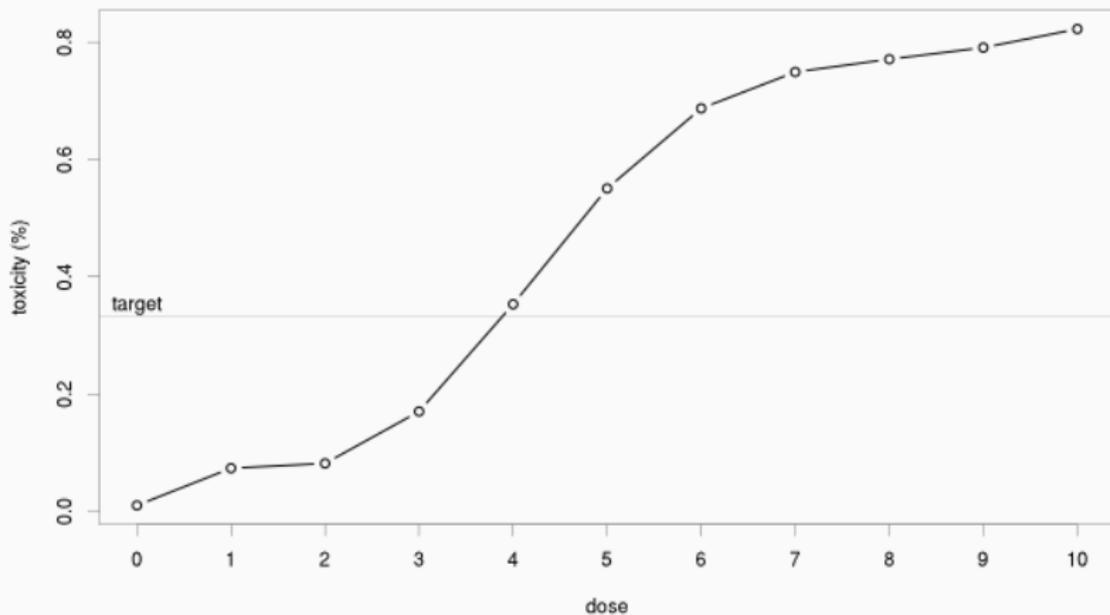
# Exemple : systèmes de recommandation



The screenshot shows the vodkaster website interface. At the top, the logo "vodkaster" is visible with the tagline "CINÉ & SÉRIES - MÉTIERS, REVIEWS". Navigation links include "DÉCOUVRIR", "ACTU", "COMMUNAUTÉ", and "MOVIEQUIZ". A search bar is present on the right. Below the navigation, a yellow banner reads "Notez des films. Complétez votre profil et obtenez vos recommandations personnalisées en regardant plus de films ou en les ajoutant à votre wishlist !". The main content area displays a grid of 18 movie posters, including titles like "Sevch", "Tom Hanks, Forrest Gump", "Pulp Fiction", "Shutter Island", "Mulholland Drive", "Kill Bill", "Infiltres", "Black Swan", "The Big Lebowski", "Reservoir Dogs", and "Into the Wild". A search bar at the top of the grid says "Rechercher par titre, genre, durée ou années...".

Modèle : **problèmes de bandits** (simples, puis riches).

# Exemple : essais cliniques (ici : phase 1)



# Exemple de résultat

## Borne inférieure

Pour trouver la distribution parmi  $\nu_1, \dots, \nu_k$  qui a la plus grande moyenne, n'importe quelle stratégie  $\delta$ -correcte (se trompant avec probabilité au plus  $\delta$ ) doit utiliser un nombre d'observations au moins égal à

$$\mathbb{E}_{\mu}[\tau_{\delta}] \geq T^*(\mu) \log \frac{1}{2.4\delta},$$

où

$$T^*(\mu)^{-1} = \sup_{w \in \Sigma_K} \inf_{\lambda \in \text{Alt}(\mu)} \left( \sum_{a=1}^K w_a d(\mu_a, \lambda_a) \right).$$

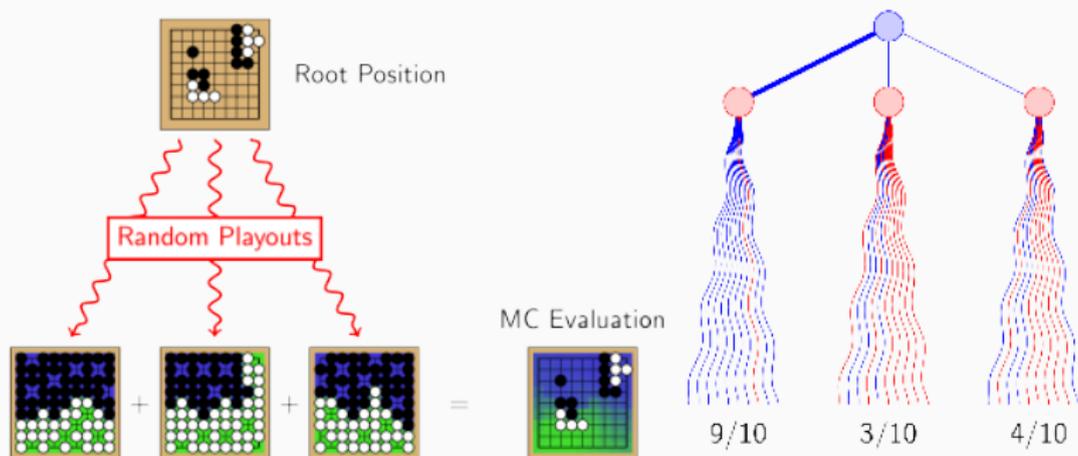
## Algorithme optimal

Nous décrivons une stratégie  $\delta$ -correcte pour laquelle

$$\limsup_{\delta \rightarrow 0} \frac{\mathbb{E}_{\mu}[\tau_{\delta}]}{\log(1/\delta)} = T^*(\mu).$$

# Résolution de jeux - MCTS

Algorithme de recherche heuristique dans les jeux profonds, utilisant des "playouts" = fin de parties aléatoires



src : <https://www.remi-coulom.fr/>