

## Least-Squares Policy Iteration

Emmanuel Rachelson

RFIA, 19 Janvier 2010

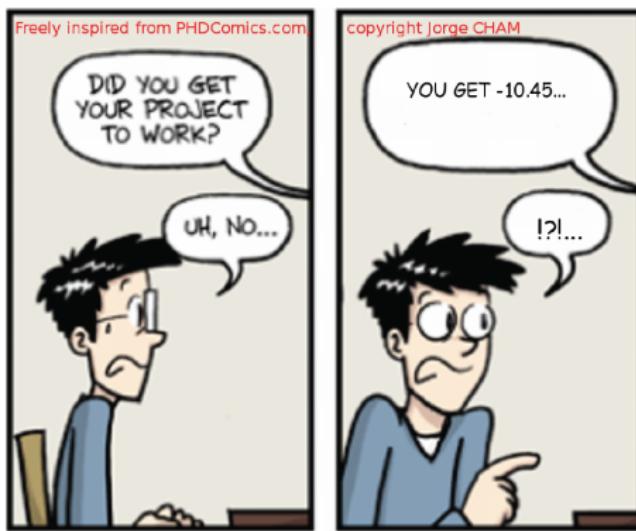
## 1 Rappels et introduction

## 2 Least Squares Temporal Differences

## 3 Least Squares Policy Iteration

## Le but

Sur la base des infos issues de l'expérience, on veut . . .



contrôler un agent de façon à maximiser un critère lié à sa trajectoire.

# Formellement

Choisir meilleure stratégie de contrôle  $\leftrightarrow$  Maximiser un critère

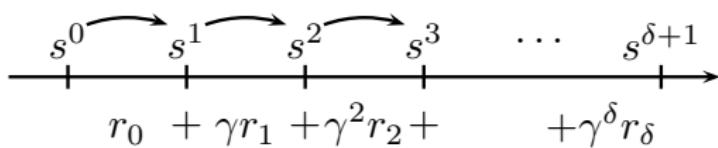
$$\forall s \in S, \quad \pi^*(s) = \arg \max_{\pi} E(C(\pi, s))$$

# Formellement

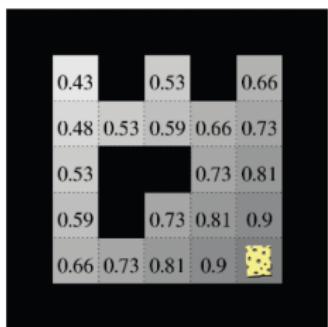
Choisir meilleure stratégie de contrôle  $\leftrightarrow$  Maximiser un critère

$$\forall s \in S, \quad \pi^*(s) = \arg \max_{\pi} E(C(\pi, s))$$

$$\forall s \in S, \quad \pi^*(s) = \arg \max_{\pi} E \left( \sum_{\delta=0}^{\infty} \gamma^{\delta} r(s_{\delta}, \pi(s_{\delta})) \middle| s_0 = s \right)$$



# Fonction de valeur

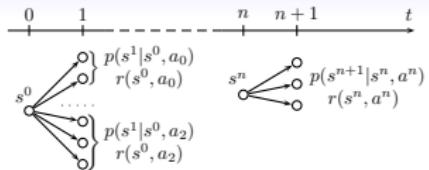


$$V^\pi(s) = E \left( \sum_{\delta=0}^{\infty} \gamma^\delta r(s_\delta, \pi(s_\delta)) \middle| s_0 = s \right)$$

Donc :

$$\forall s \in S, \quad \pi^*(s) = \arg \max_{\pi} V^\pi(s)$$

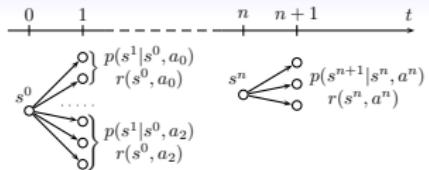
# Equation de Bellman (évaluation)



Dans un monde de Markov,

$$V^\pi(s) = E \left( \sum_{\delta=0}^{\infty} \gamma^\delta r(s_\delta, \pi(s_\delta)) \middle| s_0 = s \right) \Rightarrow V^\pi = T^\pi V^\pi$$

# Equation de Bellman (évaluation)



Dans un monde de Markov,

$$V^\pi(s) = E \left( \sum_{\delta=0}^{\infty} \gamma^\delta r(s_\delta, \pi(s_\delta)) \middle| s_0 = s \right) \Rightarrow V^\pi = T^\pi V^\pi$$

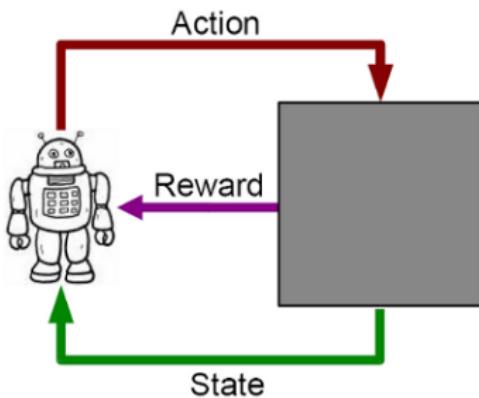
Et en termes de fonctions de valeur d'actions ( $Q$ -fonctions) :

$$Q^\pi(s, a) = r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) \sum_{a' \in A} \pi(a'|s') Q^\pi(s', a')$$

$$Q^\pi = r + \gamma P \Pi Q^\pi$$

# Confrontés à la réalité

- On ne connaît pas  $p$  et  $r$ .
- Taille des espaces de recherche.



Qu'a-t-on à disposition ?  
Echantillons  $D = \{(s, a, r, s')\}$

# Les différents algorithmes de RL

- model-based vs. model-free
- online vs. offline
- on-policy vs. off-policy
- incremental vs. batch

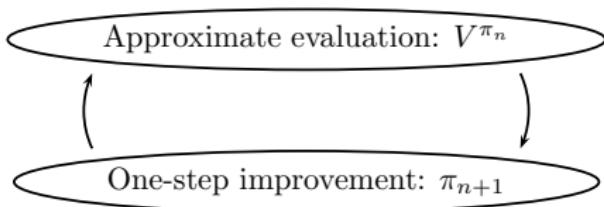
# Les différents algorithmes de RL

Dans notre cas :

- model-based vs. model-free
- online vs. offline
- on-policy vs. off-policy
- incremental vs. batch

# Les ingrédients

- Cadre actor-critic.
- Recherche directement parmi les politiques (et non les fonctions de valeur) : Policy Iteration.
- Fonction de valeur mise à jour via des échantillons : méthodes TD.
- Méthode offline : mise à jour “batch” de la fonction de valeur afin d’exploiter au mieux les échantillons et l’équation de Bellman.
- Contrer la complexité de représentation de  $V$ ,  $Q$  et  $\pi$  : abstraction par l’approximation. Ici, architecture linéaire.



## Least Squares Temporal Differences



**Boyan A. J. (2002).**

Technical Update: Least-Squares Temporal Difference Learning.  
*Machine Learning Journal* 49(2-3):233-246.

## L'idée clé

La fonction de valeur  $Q^\pi$  est solution du système linéaire :

$$Q^\pi = r + \gamma P \Pi Q^\pi$$

## L'idée clé

La fonction de valeur  $Q^\pi$  est solution du système linéaire :

$$Q^\pi = r + \gamma P \Pi Q^\pi$$

Approximation : combinaison linéaire de fonctions caractéristiques.

$$\hat{Q}(s, a, w) = \sum_{j=1}^k \phi_j(s, a) w_j$$

## L'idée clé

La fonction de valeur  $Q^\pi$  est solution du système linéaire :

$$Q^\pi = r + \gamma P \Pi Q^\pi$$

Approximation : combinaison linéaire de fonctions caractéristiques.

$$\hat{Q}(s, a, w) = \sum_{j=1}^k \phi_j(s, a) w_j$$

Ou encore :

$$\hat{Q} = \Phi w, \quad \Phi = \begin{pmatrix} \phi_1(s_1, a_1) & \cdots & \phi_k(s_1, a_1) \\ \vdots & \ddots & \vdots \\ \phi_1(s_{|S|}, a_{|A|}) & \cdots & \phi_k(s_{|S|}, a_{|A|}) \end{pmatrix}$$

# LSTD $Q$

On cherche donc

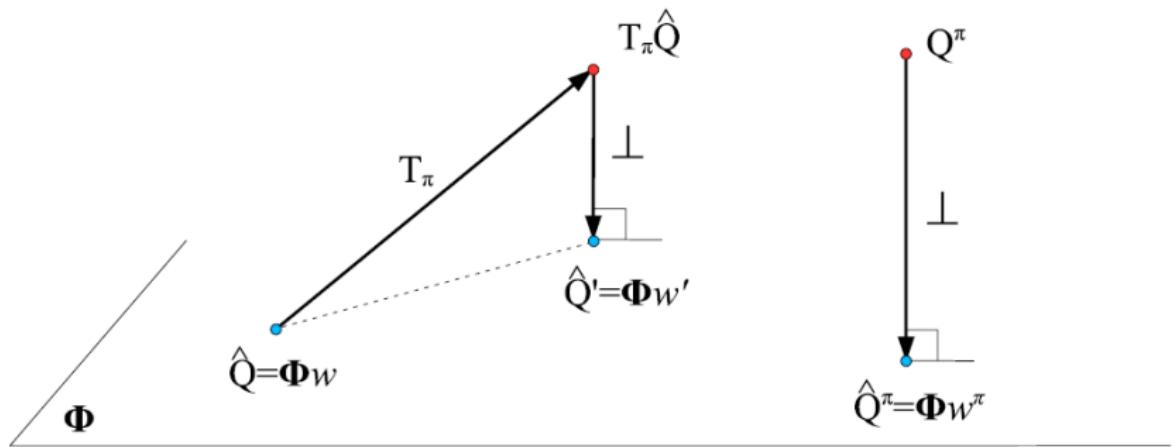
$$\hat{Q}^\pi(s, a, w) = \sum_{j=1}^k \phi_j(s, a) w_j^\pi$$

qui approche au mieux la solution de l'équation :

$$Q^\pi = r + \gamma P \Pi Q^\pi = T^\pi Q^\pi$$

Mais le point fixe de  $T^\pi$  n'appartient pas forcément au plan  $\text{span}(\Phi)$  !

# “Approcher au mieux” ?



# Minimisation de l'erreur de Bellman

Erreur de Bellman (Bellman Residual) :

$$BQ(s, a) = T^\pi Q(s, a) - Q(s, a)$$

Résolution sans approximation :  $Q^\pi$  annule l'erreur de Bellman.

## Bellman Residual Minimization (BRM)

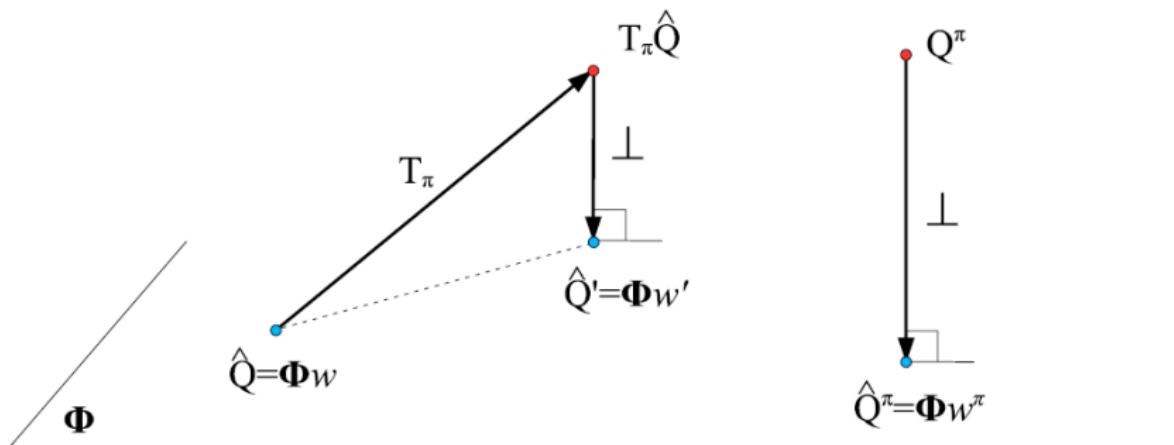
On cherche  $w$  tel que :

$$w = \arg \min_w \|BQ(s, a)\|_2 = \arg \min_w \|r + \gamma P \Pi \Phi w - \Phi w\|_2$$

C'est un problème surcontraint, type "moindres carrés".

$$w^\pi = \left( (\Phi - \gamma P \Pi \Phi)^T (\Phi - \gamma P \Pi \Phi) \right)^{-1} (\Phi - \gamma P \Pi \Phi) r$$

# “Approcher au mieux” ?



# Recherche d'un point fixe de l'opérateur $\hat{T}$

Plutôt que minimiser  $B\hat{Q}$ , on cherche un point fixe à  $T^\pi$ .

# Recherche d'un point fixe de l'opérateur $\hat{T}$

Plutôt que minimiser  $B\hat{Q}$ , on cherche un point fixe à  $T^\pi$ .

Pb :  $T^\pi Q$  est hors de  $span(\Phi)$ . Pas de solution à forme stable.  
→ Projection sur  $span(\Phi)$ .

## Recherche d'un point fixe de l'opérateur $\hat{T}$

Plutôt que minimiser  $B\hat{Q}$ , on cherche un point fixe à  $T^\pi$ .

Pb :  $T^\pi Q$  est hors de  $span(\Phi)$ . Pas de solution à forme stable.  
→ Projection sur  $span(\Phi)$ .

$$\hat{T}^\pi Q = \Phi(\Phi^T \Phi)^{-1} \Phi^T (T^\pi Q)$$

# Recherche d'un point fixe de l'opérateur $\hat{T}$

Plutôt que minimiser  $B\hat{Q}$ , on cherche un point fixe à  $T^\pi$ .

Pb :  $T^\pi Q$  est hors de  $\text{span}(\Phi)$ . Pas de solution à forme stable.  
→ Projection sur  $\text{span}(\Phi)$ .

$$\hat{T}^\pi Q = \Phi(\Phi^T \Phi)^{-1} \Phi^T (T^\pi Q)$$

## Fixed Point approximation (FP)

On cherche donc  $Q = \Phi w$  telle que

$$Q = \hat{T}^\pi Q$$

# Recherche d'un point fixe de l'opérateur $\hat{T}$

Plutôt que minimiser  $B\hat{Q}$ , on cherche un point fixe à  $T^\pi$ .

Pb :  $T^\pi Q$  est hors de  $span(\Phi)$ . Pas de solution à forme stable.  
 → Projection sur  $span(\Phi)$ .

$$\hat{T}^\pi Q = \Phi(\Phi^T \Phi)^{-1} \Phi^T (T^\pi Q)$$

## Fixed Point approximation (FP)

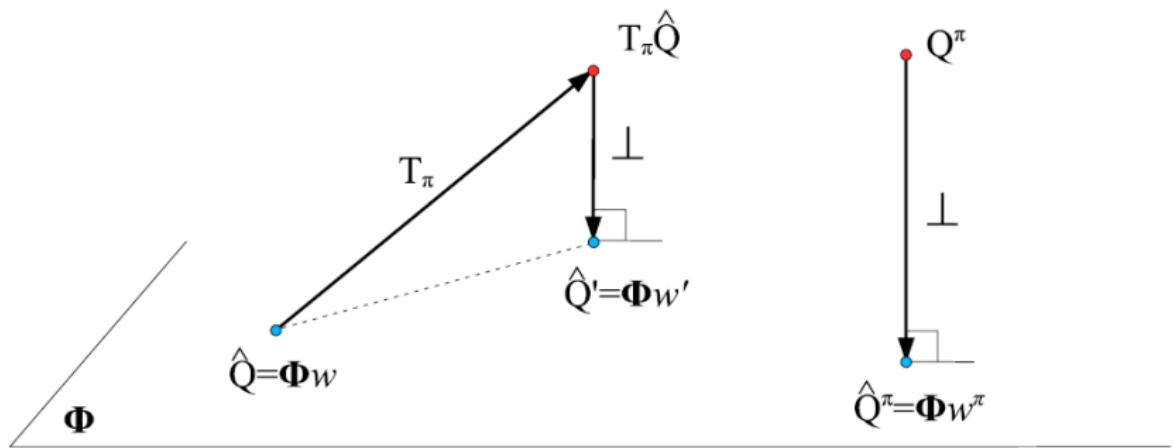
On cherche donc  $Q = \Phi w$  telle que

$$Q = \hat{T}^\pi Q$$

C'est un système linéaire de rang plein.

$$w^\pi = (\Phi^T (\Phi - \gamma P \Pi \Phi))^{-1} \Phi^T r$$

# “Approcher au mieux” ?



# Comparaison des deux méthodes

- FP conserve la propriété d'opérateur contractant.
- BRM demande deux fois plus d'échantillons que FP.
- Souvent FP est plus proche de  $Q^\pi$  que BRM.

Choix de FP en général.

# Oui, mais si on danse ?

$$w^\pi = (\Phi^T(\Phi - \gamma P \Pi \Phi))^{-1} \Phi^T r$$

Mais on n'a pas  $P$  et  $r$  !

$$\begin{aligned} A &= \Phi^T(\Phi - \gamma P \Pi \Phi) \\ b &= \Phi^T r \end{aligned}$$

- Plutôt que d'inférer  $P$  et  $r$  à partir des échantillons, on construit directement les matrices  $A$  et  $b$ .
- Plutôt que d'énumérer les couples états-actions, on approche  $A$  et  $b$  avec les éléments de  $D$ .

# De $A$ à $\tilde{A}$ , de $b$ à $\tilde{b}$

$$A = \Phi^T (\Phi - \gamma P \Pi \Phi)$$

$$\begin{aligned} &= \sum_{s \in S} \sum_{a \in A} \phi(s, a) \left( \phi(s, a) - \gamma \sum_{s' \in S} p(s, a, s') \phi(s', \pi(s')) \right)^T \\ &= \sum_{s \in S} \sum_{a \in A} \sum_{s' \in S} p(s, a, s') \left[ \phi(s, a) (\phi(s, a) - \gamma \phi(s', \pi(s'))) \right]^T \end{aligned}$$

$$b = \Phi^T r$$

$$\begin{aligned} &= \sum_{s \in S} \sum_{a \in A} \phi(s, a) \sum_{s' \in S} p(s, a, s') r(s, a, s') \\ &= \sum_{s \in S} \sum_{a \in A} \sum_{s' \in S} p(s, a, s') [\phi(s, a) r(s, a, s')] \end{aligned}$$

# Equations d'apprentissage

Etant donné  $D = \{(s_i, a_i, r_i, s'_i) | i = 1, \dots, L\}$ ,

$$\tilde{A} = \frac{1}{L} \sum_{i=1}^L \left[ \phi(s_i, a_i) (\phi(s_i, a_i) - \gamma \phi(s'_i, \pi(s'_i)))^T \right]$$

$$\tilde{b} = \frac{1}{L} \sum_{i=1}^L [\phi(s_i, a_i) r_i]$$

# Algorithme

Init:  $D = \{(s, a, r, s')\}, \phi, \pi.$

Init:  $\tilde{A} \leftarrow 0_{k \times k}.$

Init:  $\tilde{b} \leftarrow 0_{k \times 1}.$

**for**  $(s, a, r, s') \in D$  **do**

$$\tilde{A} \leftarrow \tilde{A} + \phi(s, a) (\phi(s, a) - \gamma \phi(s', \pi(s')))^\top$$

$$\tilde{b} \leftarrow \tilde{b} + \phi(s, a) r$$

$$\tilde{w}^\pi = \tilde{A}^{-1} \tilde{b}$$

**return**  $\tilde{w}^\pi$

# Bilan

## Intérêts de LSTDQ

- Estime directement la fonction  $Q^\pi$
- S'appuie sur une architecture linéaire aisément explicable
- Réutilisabilité de la base d'apprentissage  $D$ . On peut faire fonctionner l'algo sans jamais avoir besoin de nouveaux échantillons !
- Indépendance de la méthode de collecte de  $D$  et de la méthode d'optimisation.

## Least Squares Policy Iteration



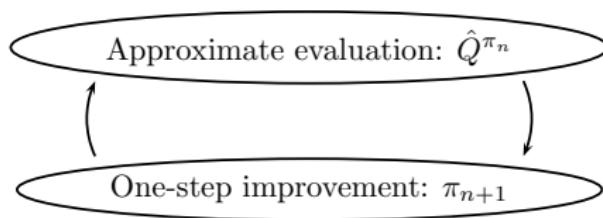
**Lagoudakis, M., and Parr, R. (2003).**

Least-Squares Policy Iteration.

*Journal of Machine Learning Research* 4:1107–1149.

# L'idée clé

Un politique gloutonne vis-à-vis de  $Q^\pi(s, a)$  est nécessairement meilleure que  $\pi$ .



Politique implicite : politique gloutonne sur  $Q^\pi$ .

# L'algorithme LSPI

$$\pi' = \pi_0$$

**repeat**

$$\pi \leftarrow \pi'$$

$$w^\pi \leftarrow \text{LSTDQ}(\pi)$$

$$\pi' \leftarrow \text{greedy}(\Phi w^\pi)$$

**until**  $\pi' \approx \pi$

**return**  $\pi$

# Convergence

Théorème de convergence de l'approximate policy iteration.

## Théorème: Convergence de l'Approximate PI

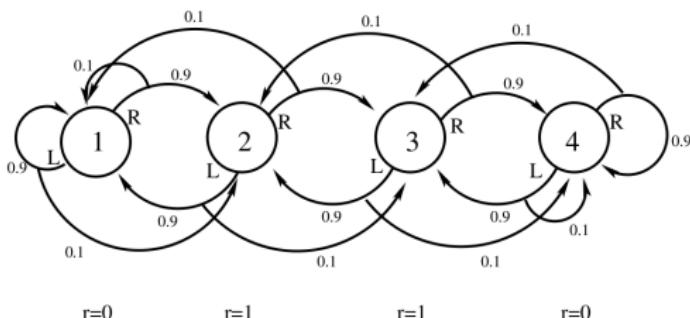
Si  $\forall m = 1, 2, \dots, \|\hat{Q}^{\pi_m} - Q^{\pi_m}\|_\infty \leq \varepsilon$ , alors :

$$\limsup_{m \rightarrow \infty} \|\hat{Q}^{\pi_m} - Q^*\|_\infty \leq \frac{2\gamma\varepsilon}{(1-\gamma)^2}$$

LSPI converge vers un voisinage de  $Q^*$ .

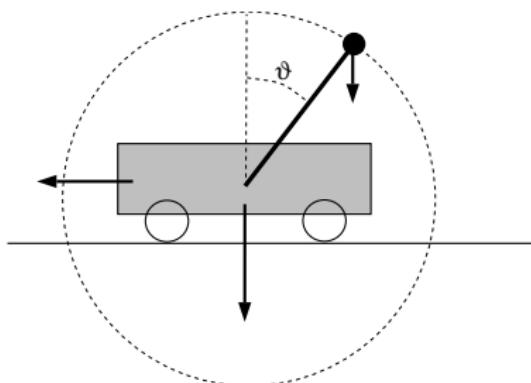
# Expériences

- Chainwalk
- Inverted Pendulum
- Bicycle



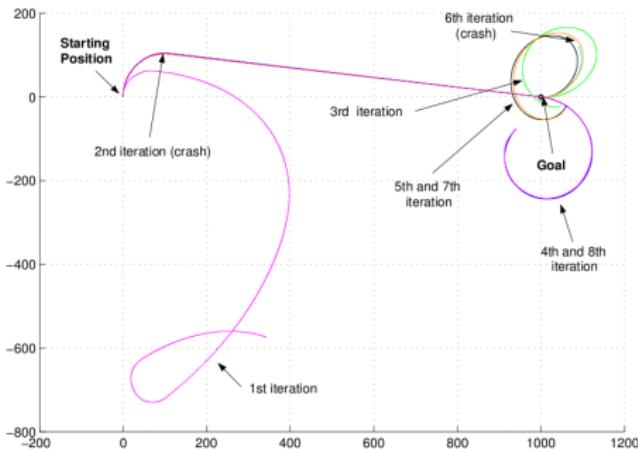
# Expériences

- Chainwalk
- Inverted Pendulum
- Bicycle



# Expériences

- Chainwalk
- Inverted Pendulum
- Bicycle



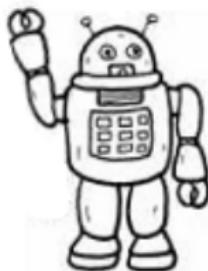
# Conclusion

Ce qu'il faut retenir.

$$\text{LSPI} = \text{LSTDQ} + \text{PI}$$

- Méthode model-free, offline, off-policy, batch.
- Architecture d'approximation linéaire.
- Algorithme simple.
- Usage maximum des échantillons.
- Applicabilité.

## Questions ?



Merci à O. Sigaud et M. G. Lagoudakis pour les illustrations !