Master 2 Recherche
Apprentissage Statistique, Optimisation et Applications

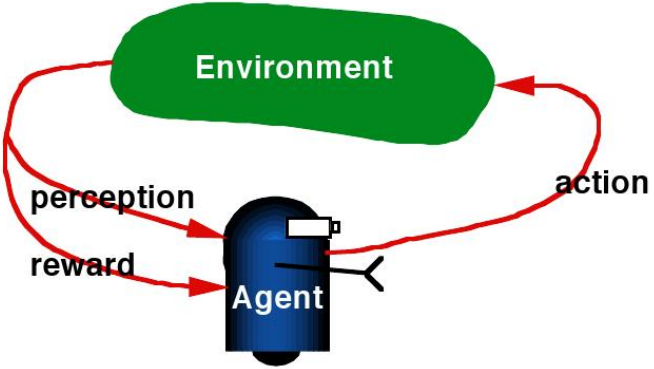
Michèle Sebag — Balazs Kégl — Anne Auger

TAO: Theme Apprentissage & Optimisation

http://tao.lri.fr/tiki-index.php

26 janvier 2011

Apprentissage par Renforcement



Cas général

- Un agent est dans le temps et dans l'espace
- L'environnement est stochastique et incertain
- Le but est d'agir sur l'environnement
- de façon à maximiser une fonction de satisfaction (reward)

Qu'est-ce qu'on apprend ?

$$\text{Une politique} = \text{une stratégie} = (\text{état} \rightarrow \text{action})$$

Apprentissage par Renforcement

Plan du cours

Apprentissage par Renforcement: Plan du cours

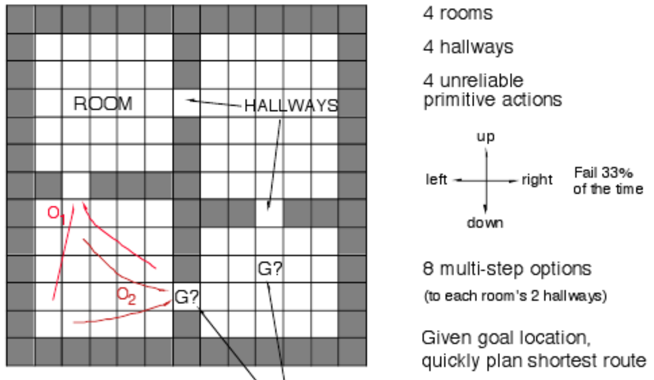
Apprentissage par Renforcement

Contexte

Le monde est inconnu.

Certaines actions, dans certains états, portent des fruits (*rewards*) avec un certain retard [avec une certaine probabilité].

Le but : trouver la politique (état $\rightarrow$ action) maximisant l'espérance de reward


ROOM
HALLWAYS
O_1
O_2
G?
G?
4 rooms
4 hallways
4 unreliable primitive actions
up
left
right
down
Fail 33% of the time
8 multi-step options
(to each room's 2 hallways)
Given goal location, quickly plan shortest route

Apprentissage par Renforcement, exemple

World	You are in state 34. Your immediate reward is 3. You have 3 actions
Robot	I'll take action 2
World	You are in state 77 Your immediate reward is -7. You have 2 actions
Robot	I'll take action 1
World	You are in state 34 (again)

Markov Decision Property: actions/rewards only depend on the current state.

Apprentissage par renforcement

Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will – others things being equal – be more firmly connected with the situation, so that when it recurs, they will more likely to recur;

those which are accompanied or closely followed by discomfort to the animal will – others things being equal – have their connection with the situation weakened, so that when it recurs, they will less likely to recur;

the greater the satisfaction or discomfort, the greater the strengthening or weakening of the link.
Thorndike, 1911.

Formalisation

Formalisation

- Espace d'états $\mathcal{S}$
- Espace d'actions $\mathcal{A}$
- Fonction de transition $p(s, a, s') \mapsto [0, 1]$
- Reward $r(s)$

But

- Trouver politique $\pi : \mathcal{S} \mapsto \mathcal{A}$

$$\text{Maximiser } E[\pi] = \text{ Espérance du reward cumulé}$$

(détails après)

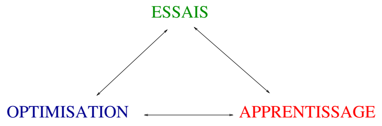
Quelques applications

- Robotique
  Navigation, football, marche, jonglage
- Jeux
  Backgammon, Othello, Tetris, Go, ...
- Contrôle
  Hélicoptère, ascenseurs, telecom, grilles de calcul, gestion de processus industriels, ...
- Recherche opérationnelle
  Transport, scheduling, ...
- Autres
  Computer Human Interfaces, ...

Position du problème

Trois problèmes

- Apprendre le monde (p, r)
- Décider
- Faire des essais



Sources

- Sutton & Barto, Reinforcement Learning, MIT Press, 1998
- `http://www.eecs.umich.edu/~baveja/NIPS05RLTutorial/`

Cas particulier

Quand on connait la fonction de transition

Reinforcement learning $\rightarrow$ Optimal control

Défis

Malédiction de la dimensionalité

- état : décrit par *taille*, *apparence*, *couleur*, ...
  $|\mathcal{S}|$ exponentiel en fonction du nombre d'attributs
- Mais tous les attributs ne sont pas toujours pertinents

Exemple:

voir	cygne	blanc	—
	cygne	noir	prendre une photo
	ours	—	fuir

Défis

Malédiction de la dimensionalité

- état : décrit par *taille*, *apparence*, *couleur*, ...
  $|\mathcal{S}|$ exponentiel en fonction du nombre d'attributs
- Mais tous les attributs ne sont pas toujours pertinents

Exemple:

voir	cygne	blanc	—
	cygne	noir	prendre une photo
	ours	—	fuir

Horizon – Rationalité limitée

- Horizon infini : on a l'éternité devant soi. JAMAIS
- Horizon fini inconnu : on veut une politique qui trouve le but aussi vite que possible
- Horizon fini : on veut une politique qui trouve le but après T pas de temps
- Rationalité limitée : on veut trouver rapidement une politique raisonnable (qui trouve une approximation du but)

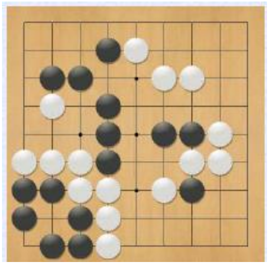
Apprentissage par Renforcement: Plan du cours

Apprentissage par Renforcement: Plan du cours

Go as AI Challenge

Features

- Number of games $2.10^{170} \sim$ number of atoms in universe.
- Branching factor: 200 ($\sim$ 30 for chess)
- Assessing a game ?
- Local and global features (symmetries, freedom, ...)



Gelly Silver 2007

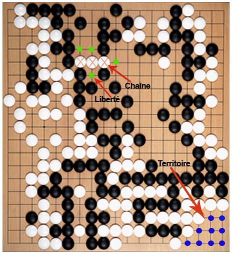
Principles of MoGo

- A weak but unbiased assessment function: Monte Carlo-based
- Allowing the machine to play against itself and build its own strategy

Weak unbiased assessment

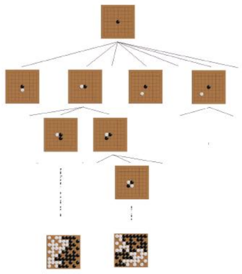
Monte-Carlo-based Brügman (1993)

1. While possible, add a stone (white, black)
2. Compute Win(black)
3. Average on 1-2



Remark: The point is to be unbiased if there exists situations where you (wrongly) think you're in good shape
then you go there and you're in bad shape...

Build a strategy: Monte-Carlo Tree Search



In a given situation:

Select a move — Multi-Armed Bandit

In the end:

1. Assess the final move — Monte-Carlo
2. Update reward for all moves

Select a move

Exploration vs Exploitation Dilemma

Multi-Armed Bandits — Lai, Robbins 1985

- In a casino, one wants to maximize one's gains *while playing*
- Play the best arms so far ? — Exploitation
- But there might exist better arms... — Exploration

Multi-Armed Bandits, foll'd

Auer et al. 2001, 2002; Kocsis Szepesvari 2006

For each arm (move)

- Reward: Bernoulli variable $\sim \mu_i, 0 \leq \mu_i \leq 1$
- Empirical estimate: $\hat{\mu}_i \pm$ Confidence (n_i) *nb trials*

Decision: Optimism in front of unknown!

$$\text{Select } i^* = \text{argmax } \hat{\mu}_i + C\sqrt{\frac{log(\sum n_j)}{n_i}}$$

Multi-Armed Bandits, foll'd

Auer et al. 2001, 2002; Kocsis Szepesvari 2006

For each arm (move)

- Reward: Bernoulli variable $\sim \mu_i, 0 \leq \mu_i \leq 1$
- Empirical estimate: $\hat{\mu}_i \pm$ Confidence (n_i) *nb trials*

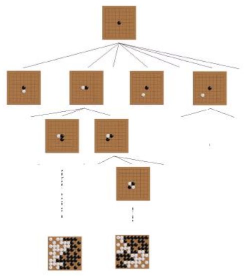
Decision: Optimism in front of unknown!

$$\text{Select } i^* = \text{argmax } \hat{\mu}_i + C\sqrt{\frac{log(\sum n_j)}{n_i}}$$

Variants

- Take into account standard deviation of $\hat{\mu}_i$
- Trade-off controlled by C
- Progressive widening

Monte-Carlo Tree Search



Comments: MCTS grows an asymmetrical tree

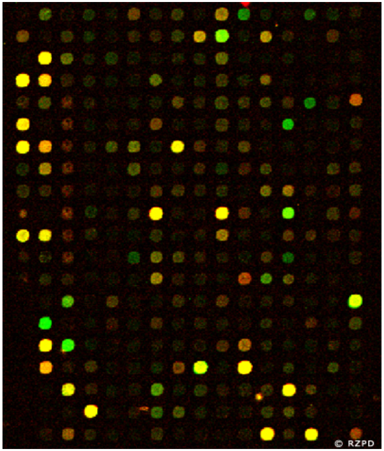
- Most promising branches are more explored
- thus their assessment becomes more precise
- Needs heuristics to deal with many arms...
- Share information among branches

MoGo: World champion in 2006, 2007, 2009
First to win over a 7th Dan player in 19×19

Apprentissage par Renforcement: Plan du cours

Quand l'apprentissage c'est la sélection d'attributs

Bio-informatique



- 30 000 gènes
- peu d'exemples (chers)
- but : trouver les gènes pertinents

Position du problème

Buts

- Sélection : trouver un sous-ensemble d'attributs
- Ordre/Ranking : ordonner les attributs

Formulation

Soient les attributs $\mathcal{F} = \{f_1, ..f_d\}$. Soit la fonction :

$$\begin{aligned} \mathcal{G} : \mathcal{P}(\mathcal{F}) &\mapsto \mathbb{R} \\ F \subset \mathcal{F} &\mapsto Err(F) = \text{ erreur min. des hypothèses fondées sur } F \end{aligned}$$

Trouver *$Argmin(\mathcal{G})$*

Difficultés

- Un problème d'optimisation combinatoire (2^d)
- D'une fonction $\mathcal{F}$ inconnue...

Approches

Filter méthode univariée

Définir $score(f_i)$; ajouter itérativement les attributs maximisant *score*

ou retirer itérativement les attributs minimisant *score*

$+$ simple - pas cher

$-$ optima très locaux

Rq : on peut bactracker : meilleurs optima, mais plus cher

Wrapping méthode multivariée

Mesurer la qualité d'attributs en rapport avec d'autres attributs :

estimer $\mathcal{G}(f_{i1}, ...f_{ik})$

$-$ cher : une estimation $=$ un pb d'apprentissage.

$+$ optima meilleurs

Méthodes hybrides.

Approches filtre

Notations

Base d'apprentissage : $\mathcal{E} = \{(x_i, y_i), i = 1..n, y_i \in \{-1, 1\}\}$

$f(x_i)$ = valeur attribut f pour exemple (x_i)

Gain d'information

arbres de décision

$$p([f = v]) = Pr(y = 1 | f(x_i) = v)$$
$$QI([f = v]) = -p \log p \; - (1 - p) \log (1 - p)$$
$$QI = \sum_v p(v) QI([f = v])$$

Corrélation

$$corr(f) = \frac{\sum_i f(x_i).y_i}{\sqrt{\sum_i (f(x_i))^2 \; \times \; \sum_i y_i^2}} \propto \sum_i f(x_i).y_i$$

Approches wrapper

Principe générer/tester

Etant donné une liste de candidats $\mathcal{L} = \{f_1, .., f_p\}$

- Générer un candidat F
- Calculer $\mathcal{G}(F)$
    - apprendre h_F à partir de $\mathcal{E}_{|F}$
    - tester h_F sur un ensemble de test $= \hat{\mathcal{G}}(F)$
- Mettre à jour $\mathcal{L}$.

Algorithmes

- hill-climbing / multiple restart
- algorithmes génétiques — Vafaie-DeJong, IJCAI 95
- (*) programmation génétique & feature construction. — Krawiec, GPEH 01

Approches a posteriori

Principe

- Construire des hypothèses
- En déduire les attributs importants
- Eliminer les autres
- Recommencer

Algorithme : SVM Recursive Feature Elimination   Guyon et al. 03

- SVM linéaire $\rightarrow h(x) = sign(\sum w_i.f_i(x) + b)$
- Si $|w_i|$ est petit, f_i n'est pas important
- Eliminer les k attributs ayant un poids min.
- Recommencer.

Limites

Hypothèses linéaires

- Un poids par attribut.

Quantité des exemples

- Les poids des attributs sont liés.
- La dimension du système est liée au nombre d'exemples.

Or le pb de FS se pose souvent quand il n'y a pas assez d'exemples

Some references

- Filter approaches [1]
- Wrapper approaches
  - Tackling combinatorial optimization [2,3,4]
    - Exploration vs Exploitation dilemma
- Embedded approaches
  - Using the learned hypothesis [5,6]
  - Using a regularization term [7,8]
    - Restricted to linear models [7] or linear combinations of kernels [8]

[1] K. Kira, and L. A. Rendell ML'92
[2] D. Margaritis NIPS'09
[3] T. Zhang NIPS'08
[4] M. Boullé J. Mach. Learn. Res. 07
[5] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik Mach. Learn. 2002
[6] J. Rogers, and S. R. Gunn SLSFS'05
[7] R. Tibshirani Journal of the Royal Statistical Society 94
[8] F. Bach NIPS'08

Feature Selection

Optimization problem

Find $F^* = \textit{argmin}\,\mathbf{Err}(\mathcal{A}, F, \mathcal{E})$

$\mathcal{F}$:	Set of features
F:	Feature subset
$\mathcal{E}$:	Training data set
$\mathcal{A}$:	Machine Learning algorithm
Err:	Generalization error

Feature Selection Goals

- Reduced Generalization Error
- More cost-effective models
- More understandable models

Bottlenecks

- Combinatorial optimization problem: find $F \subseteq \mathcal{F}$
- Generalization error unknown

FS as A Markov Decision Process

Set of features $\mathcal{F}$

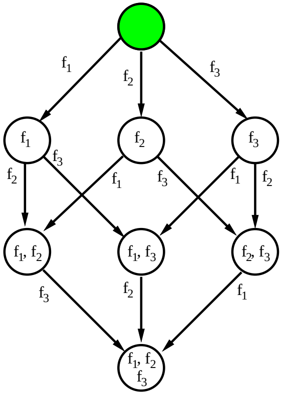
Set of states $\mathcal{S} = 2^{\mathcal{F}}$

Initial state $\varnothing$

Set of actions $A = \{\text{add } f, \ f \in \mathcal{F}\}$

Final state any state

Reward function $V : \mathcal{S} \mapsto [0, 1]$



Goal: Find $\underset{F \subseteq \mathcal{F}}{argmin}\ \mathbf{Err}\left(\mathcal{A}\left(F, D\right)\right)$

Optimal Policy

Policy $\pi : \mathcal{S} \rightarrow A$

Final state following a policy F_π

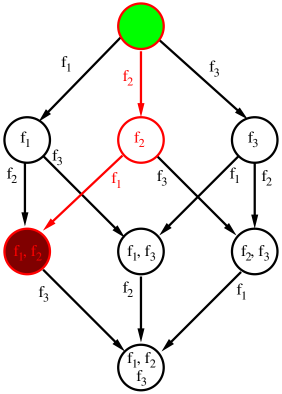
Optimal policy $\pi^\star =$

$$\underset{\pi}{argmin}\ \mathbf{Err}\left(\mathcal{A}\left(F_\pi, \mathcal{E}\right)\right)$$

Bellman's optimality principle

$$\pi^\star(F) = \underset{f \in \mathcal{F}}{argmin}\ V^\star(F \cup \{f\})$$

$$V^\star(F) = \begin{cases} \mathbf{Err}(\mathcal{A}(F)) & \text{if } final(F) \\ \underset{f \in \mathcal{F}}{min}\ V^\star(F \cup \{f\}) & \text{otherwise} \end{cases}$$



In practice

- $\pi^\star$ intractable $\Rightarrow$ approximation using UCT
- Computing $\mathbf{Err}(F)$ using a fast estimate

FS as a game

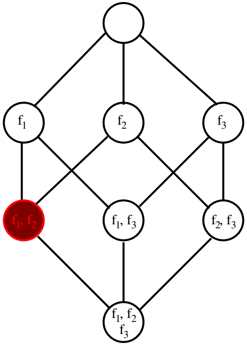
Exploration vs Exploitation tradeoff

- Virtually explore the whole lattice
- Gradually focus the search on most promising Fs
- Use a frugal, unbiased assessment of F

How ?

- Upper Confidence Tree (UCT) [1]
    - UCT $\subset$ Monte-Carlo Tree Search
    - UCT tackles tree-structured optimization problems

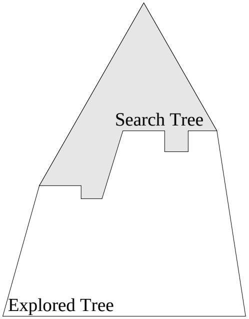
[1] L. Kocsis, and C. Szepesvári ECML'06



Apprentissage par Renforcement: Plan du cours

The UCT scheme

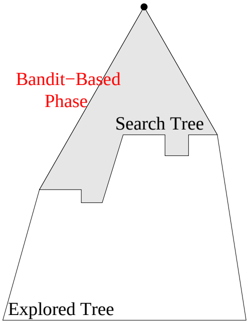
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

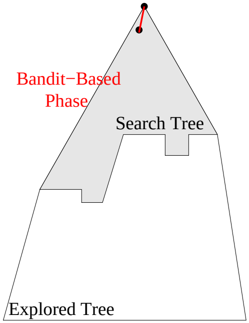
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

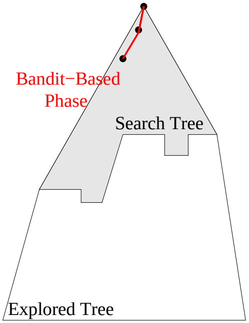
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

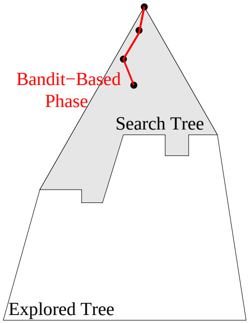
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

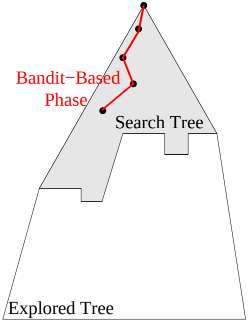
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

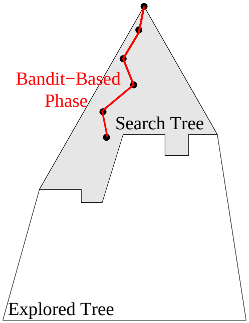
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

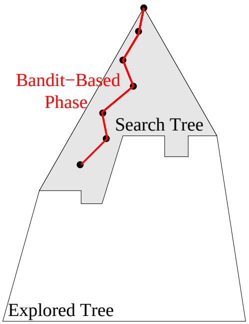
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

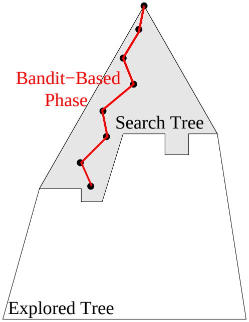
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

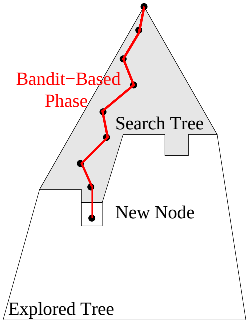
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

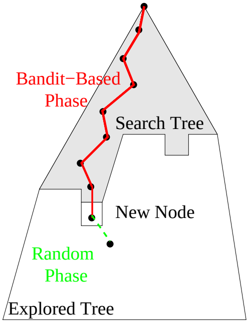
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

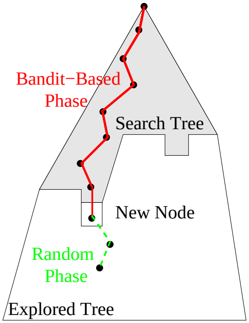
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

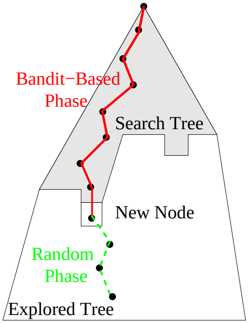
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

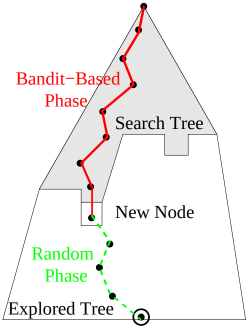
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

The UCT scheme

- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

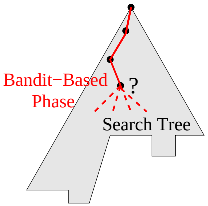


[1] L. Kocsis, and C. Szepesvári ECML'06

Multi-Arm Bandit-based phase

- Upper Confidence Bound (UCB1-tuned) [1]
  - Select $\underset{a \in A}{argmax}\ \hat{\mu}_a + \sqrt{\frac{c_e \log(T)}{n_a} min \left(\frac{1}{4}, \hat{\sigma}_a^2 + \sqrt{\frac{c_e \log(T)}{t_a}} \right)}$

- T: Total number of trials in current node
- n_a: Number of trials for action a
- $\hat{\mu}_a$: Empirical average reward for action a
- $\hat{\sigma}_a^2$: Empirical variance of reward for action a

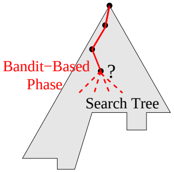


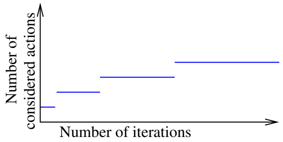
[1] P. Auer, N. Cesa-Bianchi, and P. Fischer ML'02

Apprentissage par Renforcement: Plan du cours

FUSE: bandit-based phase
The many arms problem

- Bottleneck
  - A many-armed problem (hundreds of features)
    - $\Rightarrow$ need to guide UCT
- How to control the number of arms?
  - Continuous heuristics [1]
    - Use a small exploration constant c_e
  - Discrete heuristics [2,3]: Progressive Widening
    - Consider only $\lfloor T^b \rfloor$ actions ($b < 1$)



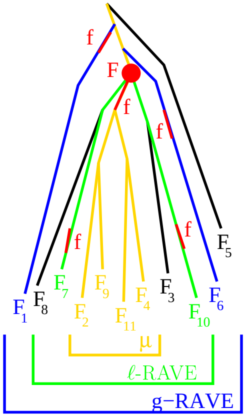


[1] S. Gelly, and D. Silver ICML'07
[2] R. Coulom Computer and Games 2006
[3] P. Rolet, M. Sebag, and O. Teytaud ECML'09

FUSE: bandit-based phase
Sharing information among nodes

- How to share information among nodes?
  - Rapid Action Value Estimation (RAVE) [1]

$$\text{RAVE}(f) = \text{average reward when } f \in F$$

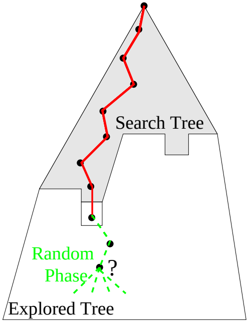


[1] S. Gelly, and D. Silver ICML'07

FUSE: random phase
Dealing with an unknown horizon

- Bottleneck
  - Finite unknown horizon
- Random phase policy

```
↱ With probability $1 - q^{|F|}$ stop
| Else • add a uniformly selected feature
|      • $|F| = |F| + 1$
⌊ Iterate
```

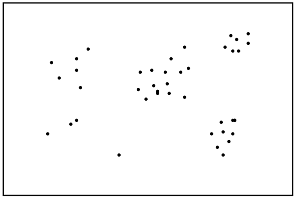


FUSE: reward(F)

Generalization error estimate

- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN like
  - $+$ AUC criterion *
- Complexity: $\tilde{O}(mnd)$

  d Number of selected features
  n Size of the training set
  m Size of sub-sample ($m \ll n$)



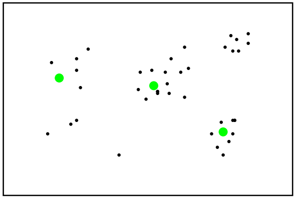
(*) Mann Whitney Wilcoxon test:

$$V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$$

FUSE: reward(F)
Generalization error estimate

- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN like
  - $+$ AUC criterion *
- Complexity: $\tilde{O}(mnd)$

  d Number of selected features
  n Size of the training set
  m Size of sub-sample ($m \ll n$)



(*) Mann Whitney Wilcoxon test:

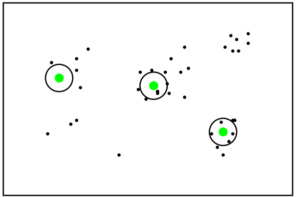
$$V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$$

FUSE: reward(F)

Generalization error estimate

- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN like
  - $+$ AUC criterion *
- Complexity: $\tilde{O}(mnd)$

d	Number of selected features
n	Size of the training set
m	Size of sub-sample ($m \ll n$)



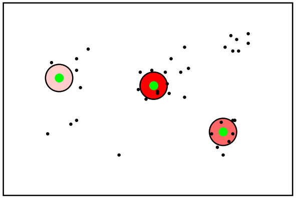
(*) Mann Whitney Wilcoxon test:

$$V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$$

FUSE: reward(F)
Generalization error estimate

- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN like
  - $+$ AUC criterion *
- Complexity: $\tilde{O}(mnd)$

  d Number of selected features
  n Size of the training set
  m Size of sub-sample ($m \ll n$)



(*) Mann Whitney Wilcoxon test:

$$V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$$

FUSE: reward(F)

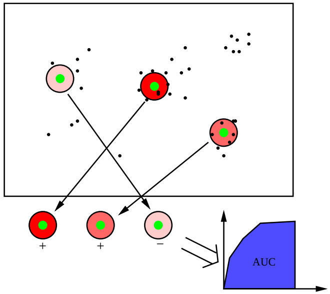
Generalization error estimate

- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN like
  - $+$ AUC criterion *
- Complexity: $\tilde{O}(mnd)$

  d Number of selected features

  n Size of the training set

  m Size of sub-sample ($m \ll n$)

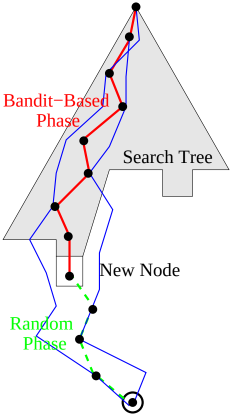


(*) Mann Whitney Wilcoxon test:

$$V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$$

FUSE: update

- Explore a graph
  - $\Rightarrow$ Several paths to the same node
- Update only current path



The FUSE algorithm

- N iterations:
  each iteration i) follows a path; ii) evaluates a final node
- Result:

$$\begin{tikzcd}
\text{Search tree (most visited path)} \arrow[r, leftrightarrow] \arrow[d, Rightarrow] & \text{RAVE score} \arrow[d, Rightarrow] \\
\text{Wrapper approach} & \text{Filter approach}
\end{tikzcd}$$

FUSE **FUSE**^{R}

- On the feature subset, use end learner $\mathcal{A}$
  - Any Machine Learning algorithm
  - Support Vector Machine with Gaussian kernel in experiments

Apprentissage par Renforcement: Plan du cours

Experimental setting

- Questions
  - FUSE vs FUSE^R
  - Continuous vs discrete exploration heuristics
  - FS performance w.r.t. complexity of the target concept
  - Convergence speed
- Experiments on

Data set	Samples	Features	Properties
Madelon [1]	2,600	500	XOR-like
Arcene [1]	200	10,000	Redundant features
Colon	62	2,000	"Easy"

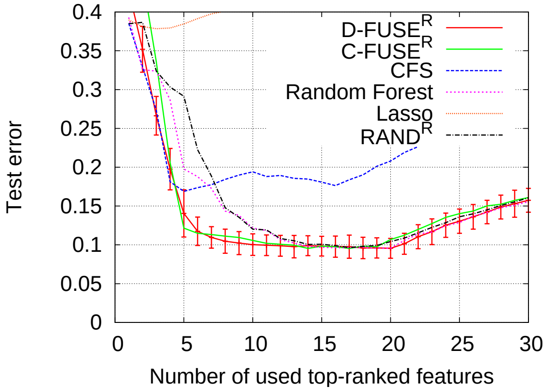
[1] NIPS'03

Experimental setting

- Baselines
  - CFS (Constraint-based Feature Selection) [1]
  - Random Forest [2]
  - Lasso [3]
  - RAND^R: RAVE obtained by selecting 20 random features at each iteration
- Results averaged on 50 splits (10 $\times$ 5 fold cross-validation)
- End learner
  - Hyper-parameters optimized by 5 fold cross-validation

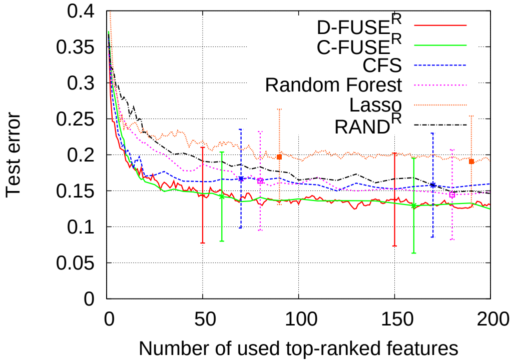
[1] M. A. Hall ICML'00
[2] J. Rogers, and S. R. Gunn SLSFS'05
[3] R. Tibshirani Journal of the Royal Statistical Society 94

Results on Madelon after 200,000 iterations



- Remark: FUSE^R $=$ best of both worlds
  - Removes redundancy (like CFS)
  - Keeps conditionally relevant features (like Random Forest)

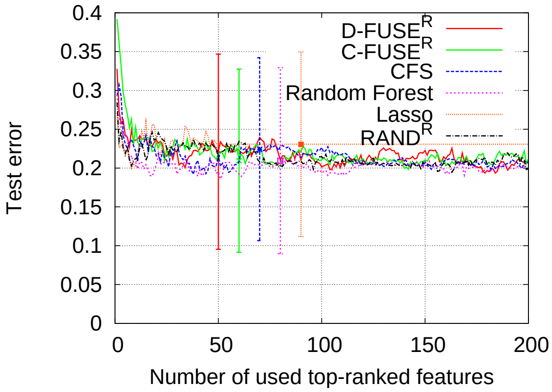
Results on Arcene after 200,000 iterations



- Remark: FUSE^R = best of both worlds
  - Removes redundancy (like CFS)
  - Keeps conditionally relevant features (like Random Forest)

[0] T-test "CFS vs. FUSE^R" with 100 features: p-value=0.036

Results on Colon after 200,000 iterations



- Remark
  - All equivalent

NIPS 2003 Feature Selection challenge

- Test error on a disjoint test set

DATABASE	ALGORITHM	CHALLENGE ERROR	SUBMITTED FEATURES	IRRELEVANT FEATURES
MADELON	FSPP2 [1]	6.22% (1st)	12	0
	D-FUSE^R	6.50% (24th)	18	0
	BAYES-NN-RED [2]	7.20% (1st)	100	0
ARCENE	D-FUSE^R(ON ALL)	8.42% (3rd)	500	34
	D-FUSE^R	9.42% 500 (8th)	500	0

[1] K. Q. Shen, C. J. Ong, X. P. Li, E. P. V. Wilder-Smith Mach. Learn. 2008
[2] R. M. Neal, and J. Zhang Feature extraction, foundations and applications, Springer 2006

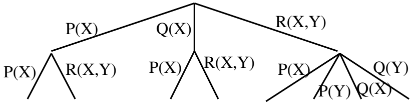
Conclusion

Contributions

- Formalization of Feature Selection as a Markov Decision Process
- Efficient approximation of the optimal policy (based on UCT)
  $\Rightarrow$ Any-time algorithm
- Experimental results
  - State of the art
  - High computational cost (45 minutes on Madelon)

Perspectives

- Other end learners
- Revisit the reward see (Hand 2010) about AUC
- Extend to Feature construction along [1]



[1] F. de Mesmay, A. Rimmel, Y. Voronenko, and M. Püschel ICML'09

Apprentissage par Renforcement: Plan du cours

Active Learning, position of the problem

Supervised learning, the setting

- Target hypothesis h^*
- Training set $\mathcal{E} = \{(x_i, y_i), i = 1 \ldots n\}$
- Learn h_n from $\mathcal{E}$

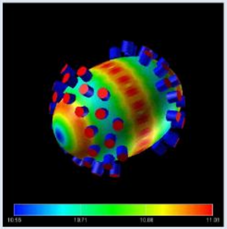
Criteria

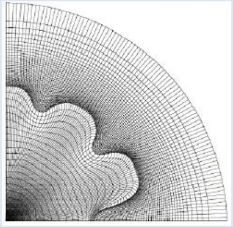
- Consistency: $h_n \to h^*$ when $n \to \infty$.
- Sample complexity: number of examples needed to reach the target with precision ϵ

$$\epsilon \to n_\epsilon \; s.t. \; ||h_n - h^*|| < \epsilon$$

Motivations

- Given *x*, obtaining *h*(x)* is costly
- Goal: reduce *sample complexity* while keeping *generalization error* low
- Motivating application: numerical engineering





=> Learn simplified models with only ~ 100 examples

Active Learning, definition

Passive learning

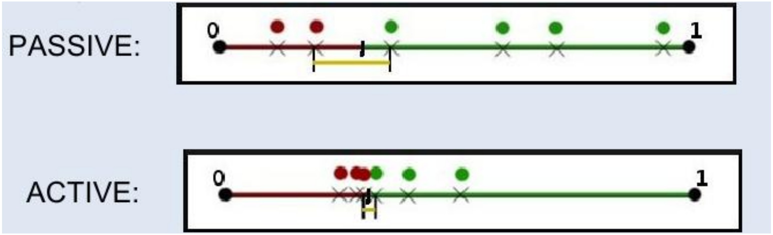
iid examples

$$\mathcal{E} = \{(x_i, y_i), i = 1 \dots n\}$$

Active learning

x_{n+1} selected depending on $\{(x_i, y_i), i = 1 \dots n\}$

In the best case, exponential improvement:



State of the art

Let H be the hypothesis space.
Realizable assumption: $h^* \in H$
Then, exponential improvements. Freund et al. 1997; Dasgupta 2005; Balcan et al. 2010.

Noisy case: improvement depends on noise model
Balcan et al. 2006; Hanneke 2007; Dasgupta et al. 2008.

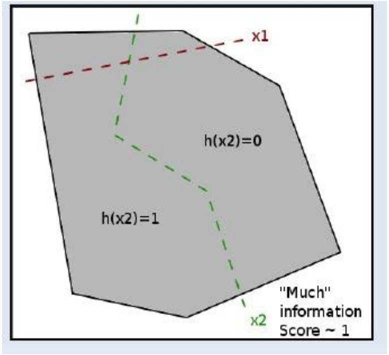
Realizable batch case
PhD Philippe Rolet, 23 dec. 2010.

How it works

Principle

- Design a measure of the information brought by an instance
- Iteratively select the best instance

Example: query by committee Seung et al. 92



Active Learning

Optimization problem

- T: time horizon (number of instances to select)
- States $s_t = \{(x_i, h^*(x_i)), i = 1 \ldots t\}$
- Action: select x_{t+1}
- $\mathcal{A}$: Machine Learning algorithm
- **Err**: Generalization error

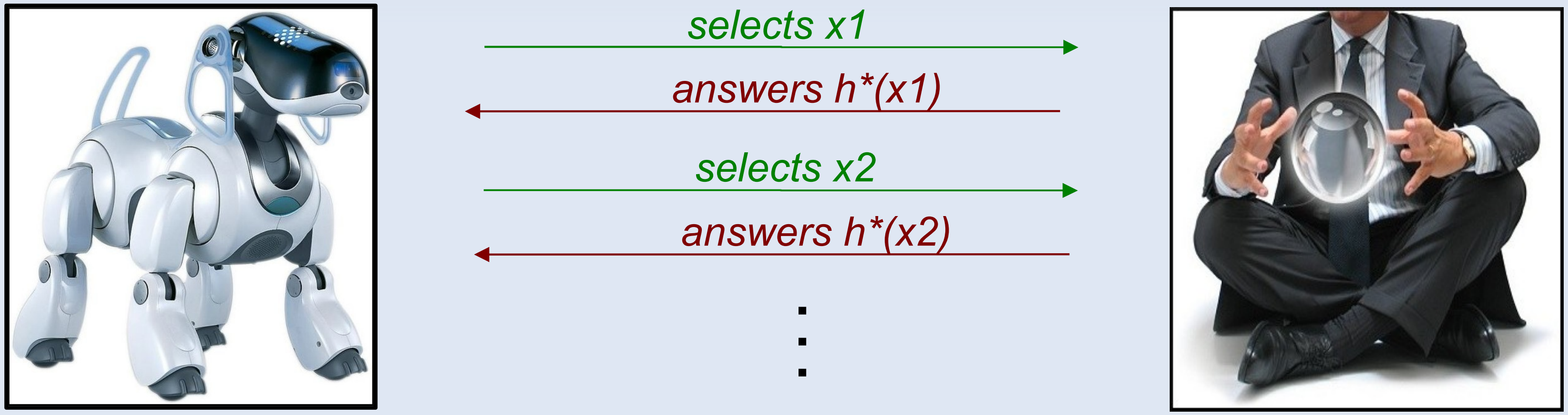
Find Sampling strategy S minimizing $\mathbb{E}\mathbf{Err}(\mathcal{A}(S_T(h^*), h^*)$

Bottlenecks

- Combinatorial optimization problem - in a continuous space
- Generalization error unknown

Optimal Strategy for AL

- Learning algorithm $\mathcal{A}$
- Finite Horizon T
- Sampling strategy S_T
- Target concept h^*



Learner $\mathcal{A}$

T-size training set $S_T(h^*)$
$\{(x_1,h^*(x_1)), \ldots , (x_T,h^*(x_T))\}$

Target Concept h^*
(a.k.a. Oracle)

- **Goal:** $\mathrm{argmin}\ E[\, Err(\mathcal{A}(S_T(h^*)), h^*)]$

Optimal Strategy for AL

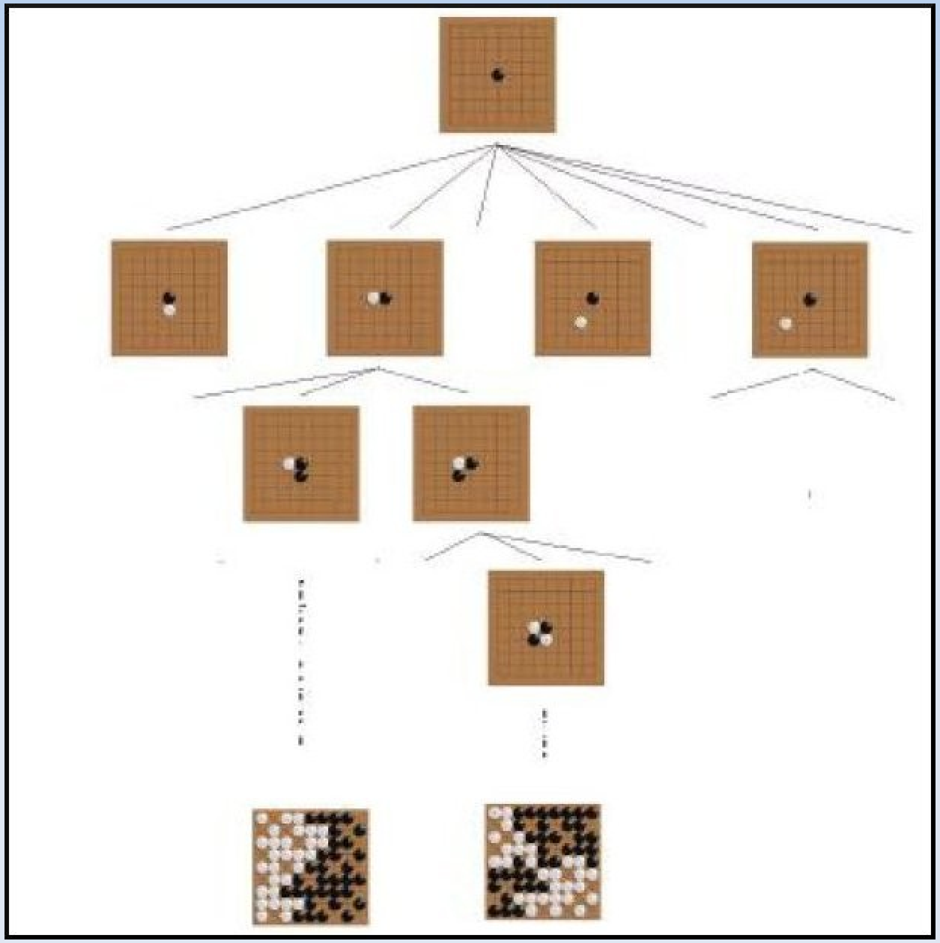
- AL modeled as a Markov decision process:
  - State space: all possible training sets of size $\leq T$
  - Action space: instances x available for query
  - Transition function: $P(s_{t+1} | s_t , x)$
  - Reward function: gen. err. $Err(\mathcal{A}(S_T(h)),h)$

- Optimal policy $\pi^* \longrightarrow$ Optimal AL strategy

Active Learning: a 1-Player Game

- Bottlenecks:
  - Large state space
  - Large action space
  - Cannot use h* directly
- Approx. sol. inspired from Go: AL as a game

  Coulom 06, Chaslot et al. 06, Gelly&Sliver 07
- Browse game tree
- Estimate move values with *Monte-Carlo* simulations



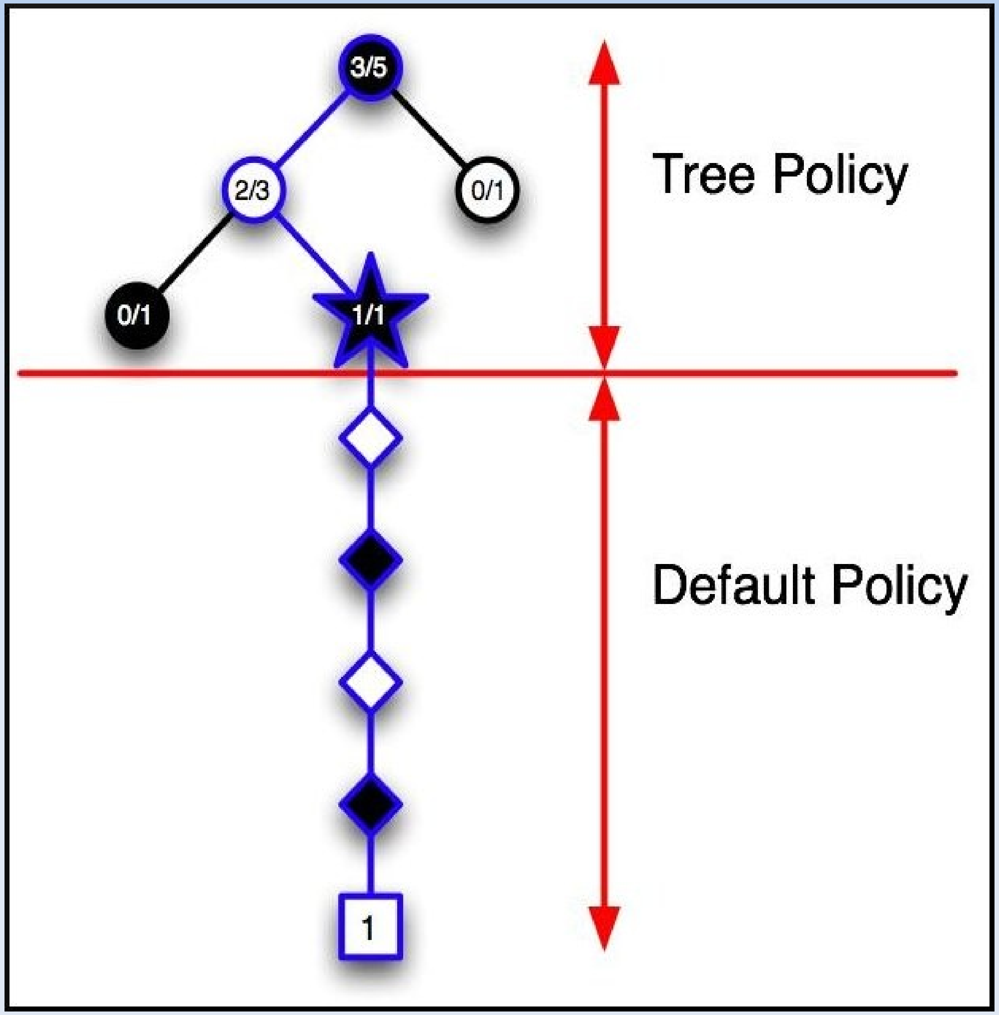
Apprentissage par Renforcement: Plan du cours

The BAAL Algorithm

=> Bandit-based Active Learner

- Simulation planning with Multi-armed bandits
- Asymetric tree growth

  More exploration for promising moves



BAAL: Exploration v. Exploitation

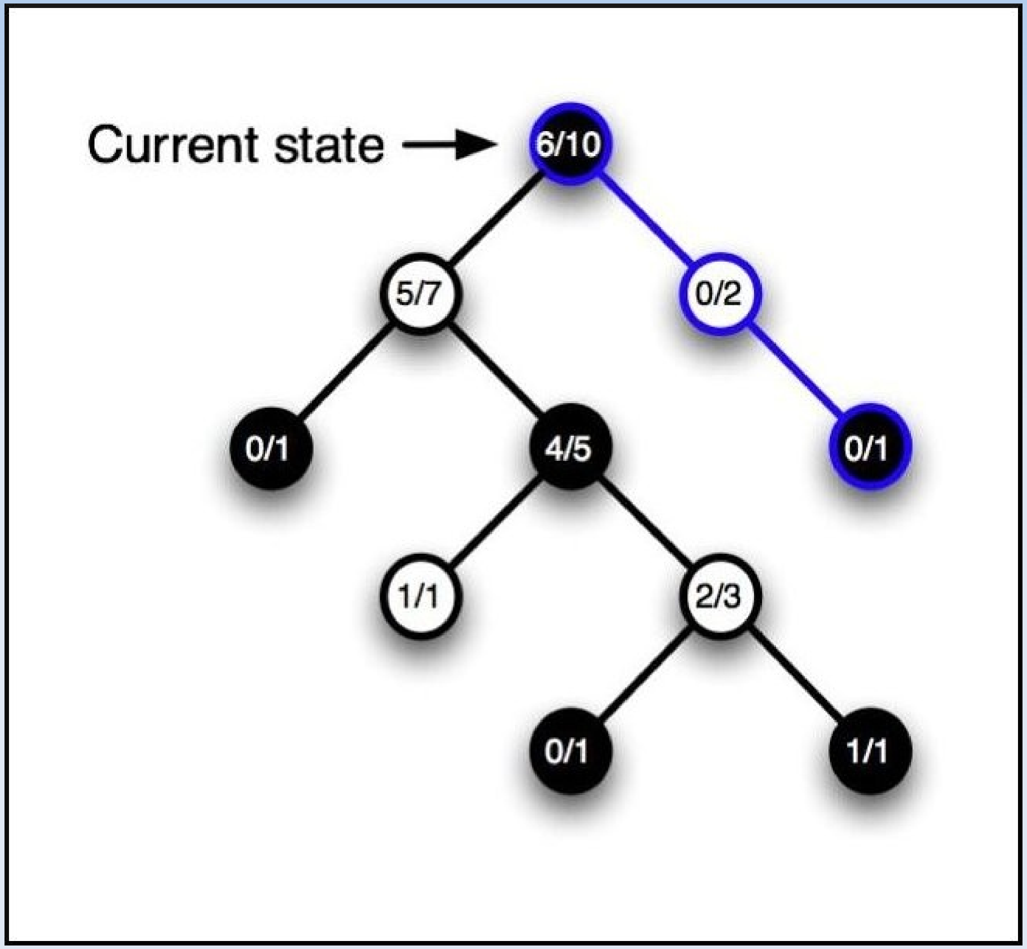
- UCB: balance exploration and exploitation

  Auer, 2002

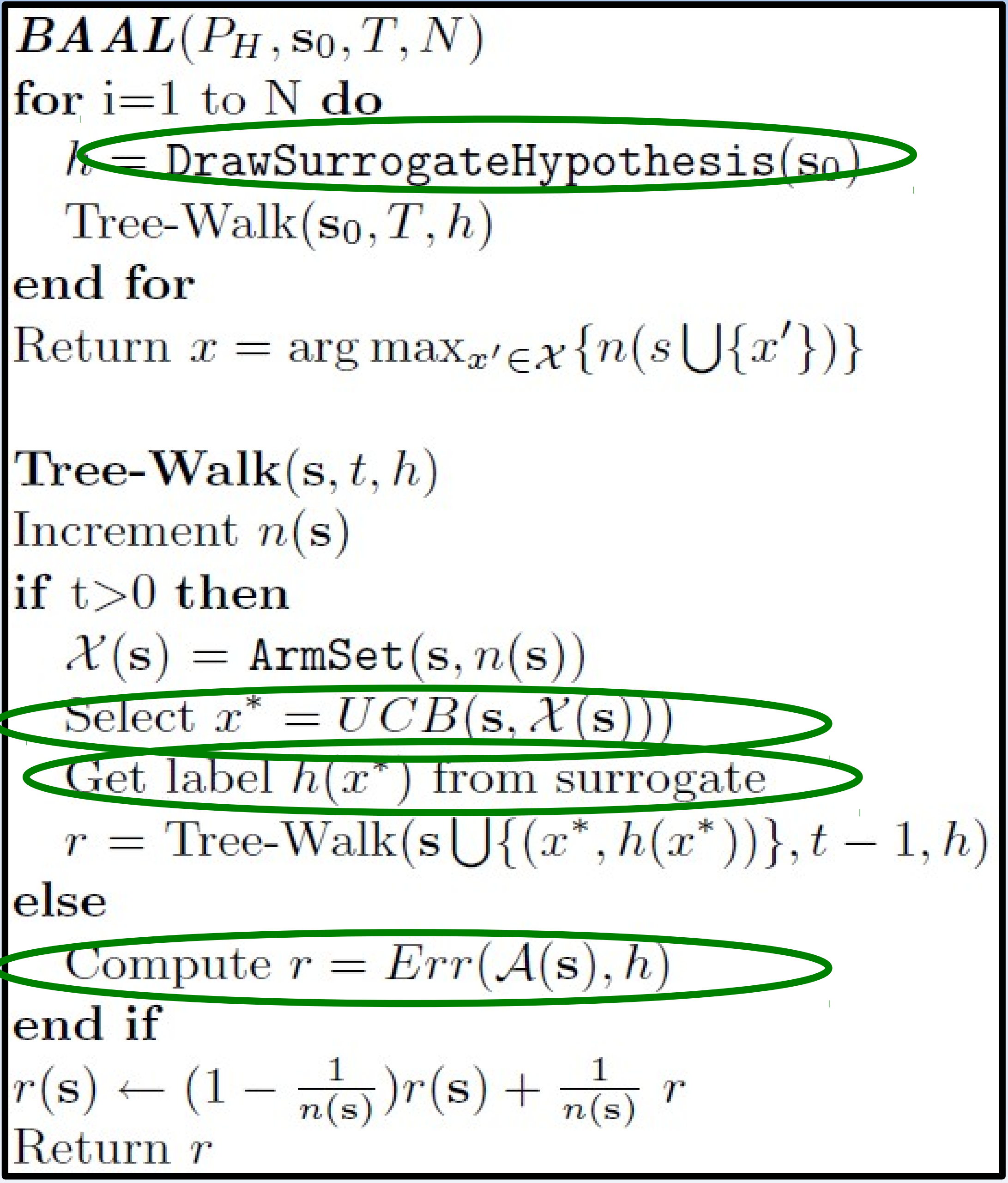
- UCT = UCB for trees

  Kocsis&Szepesvari, 2006

$$\hat{\mu}_i + C\sqrt{\frac{\log(\sum_j n_j)}{n_i}}$$

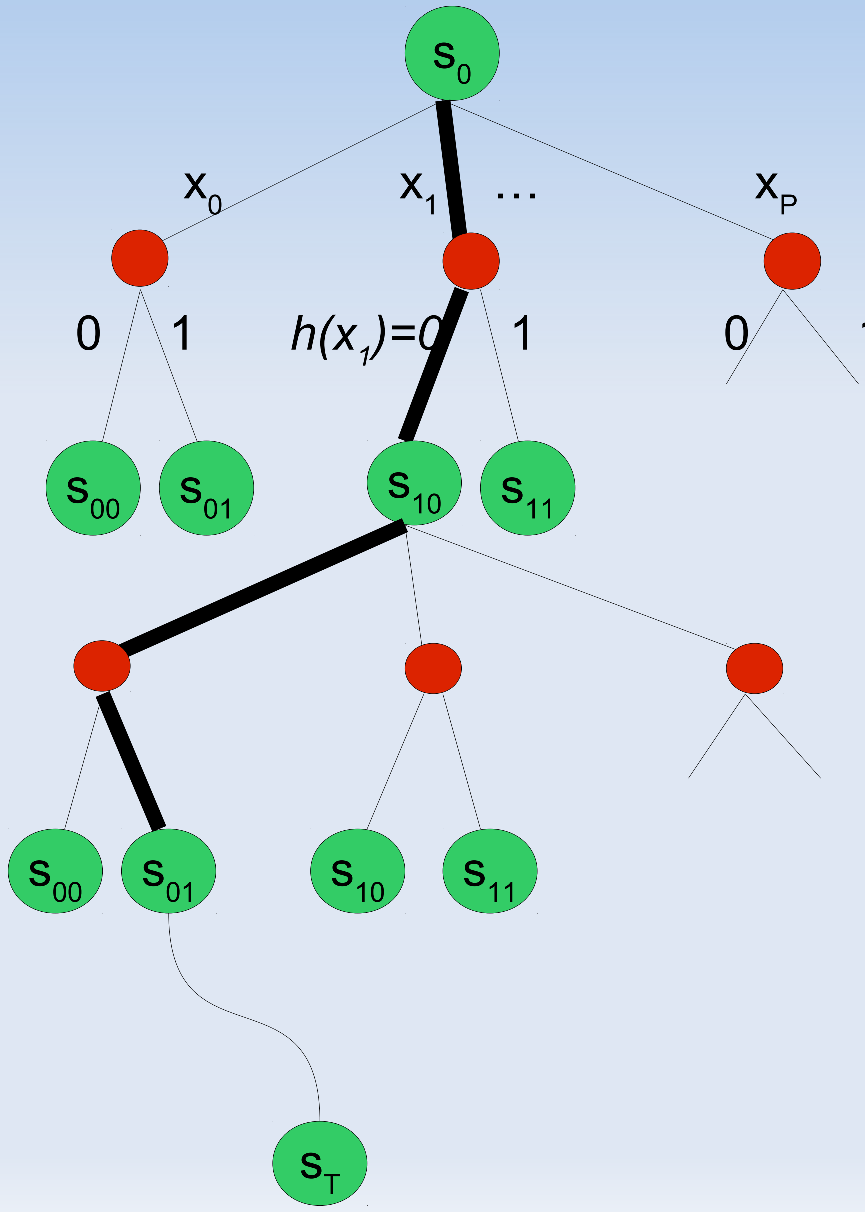


BAAL: Outline



```
$BAAL(P_H, s_0, T, N)$
for i=1 to N do
    $h$ = DrawSurrogateHypothesis($s_0$)
    Tree-Walk($s_0, T, h$)
end for
Return $x = \arg\max_{x' \in \mathcal{X}} \{n(s \bigcup \{x'\})\}$

Tree-Walk($s, t, h$)
Increment $n(s)$
if t>0 then
    $\mathcal{X}(s)$ = ArmSet($s, n(s)$)
    Select $x^* = UCB(s, \mathcal{X}(s))$
    Get label $h(x^*)$ from surrogate
    $r$ = Tree-Walk($s \bigcup \{(x^*, h(x^*))\}, t-1, h$)
else
    Compute $r = Err(\mathcal{A}(s), h)$
end if
$r(s) \leftarrow (1 - \frac{1}{n(s)}) r(s) + \frac{1}{n(s)} r$
Return $r$
```



Baal: Continuous action space

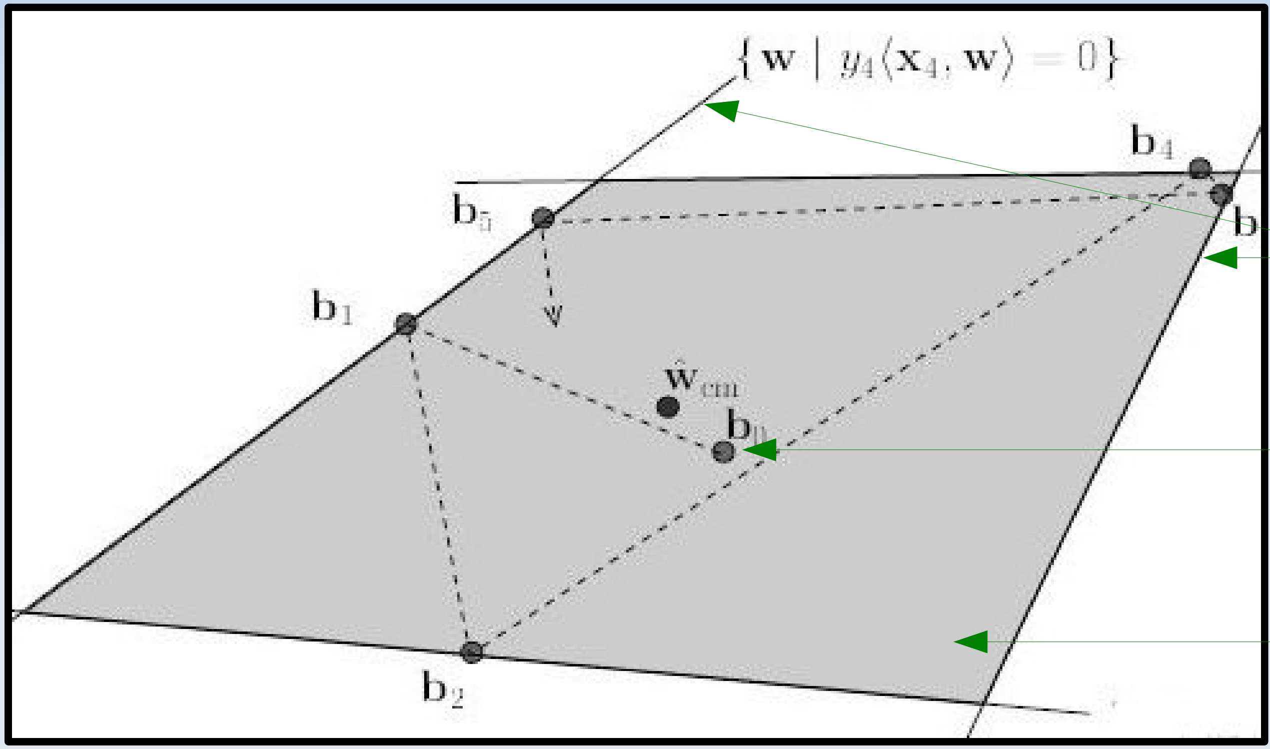
- UCB is designed for finite action spaces
- AL: action space = R^D
- Control the number of arms: *progressive widening* *# instances ~ (# visits)$^{1/4}$*

  Coulom, 2007
  Wang, Audibert, Munos, 2008
- Select new instances
  - In a random order
  - Following a given heuristic (e.g. QbC heuristic)

Baal: draw surrogate hypotheses

- *Billiard* algorithms

Rujan 97,
Comets et. al. 09, ...



Constraints = labeled instances

Point = hypothesis

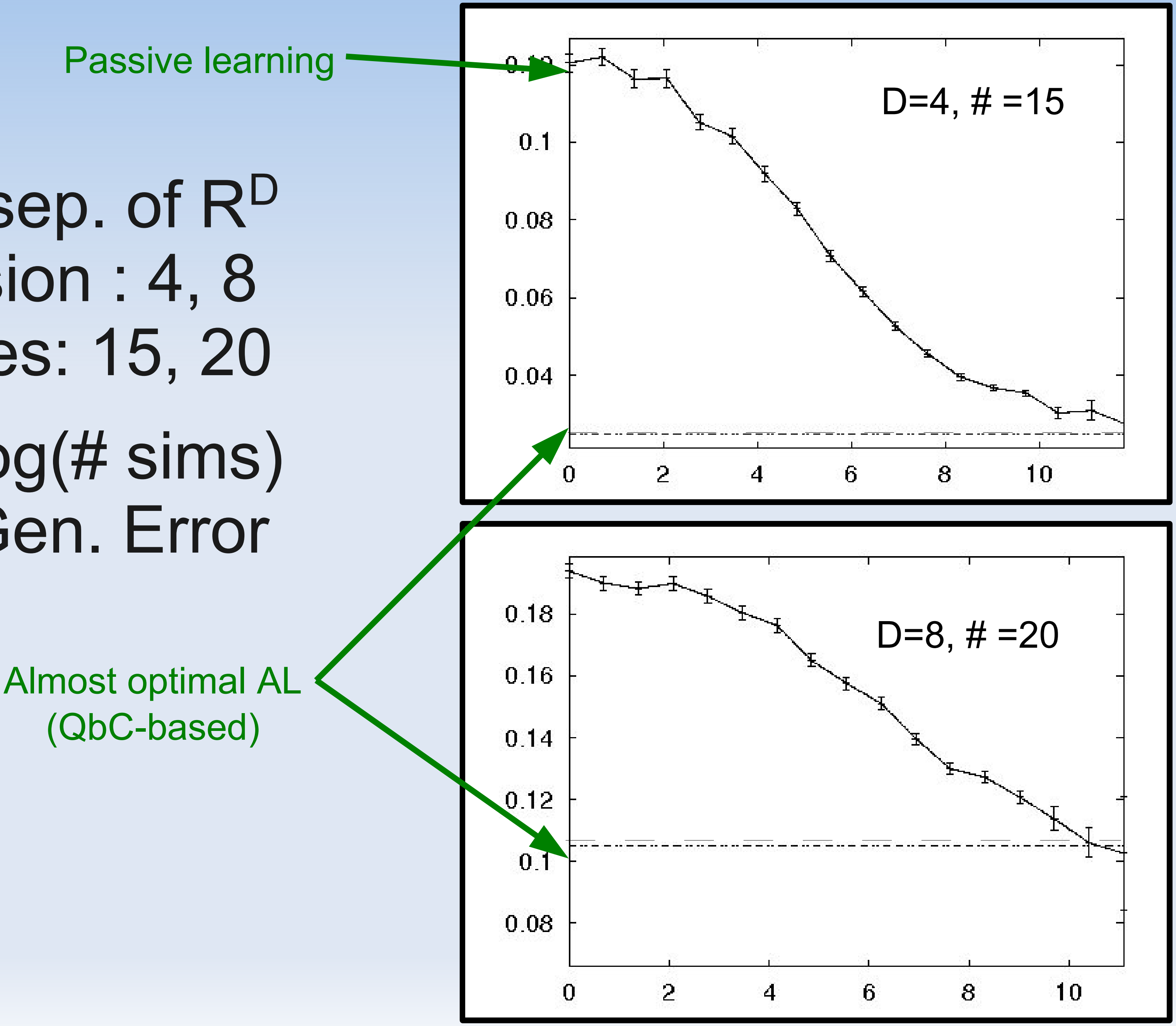
Domain = version space

- **Sound:** provably converges to uniform draw
- **Scalable** w.r.t. dimension, # constraints

Apprentissage par Renforcement: Plan du cours

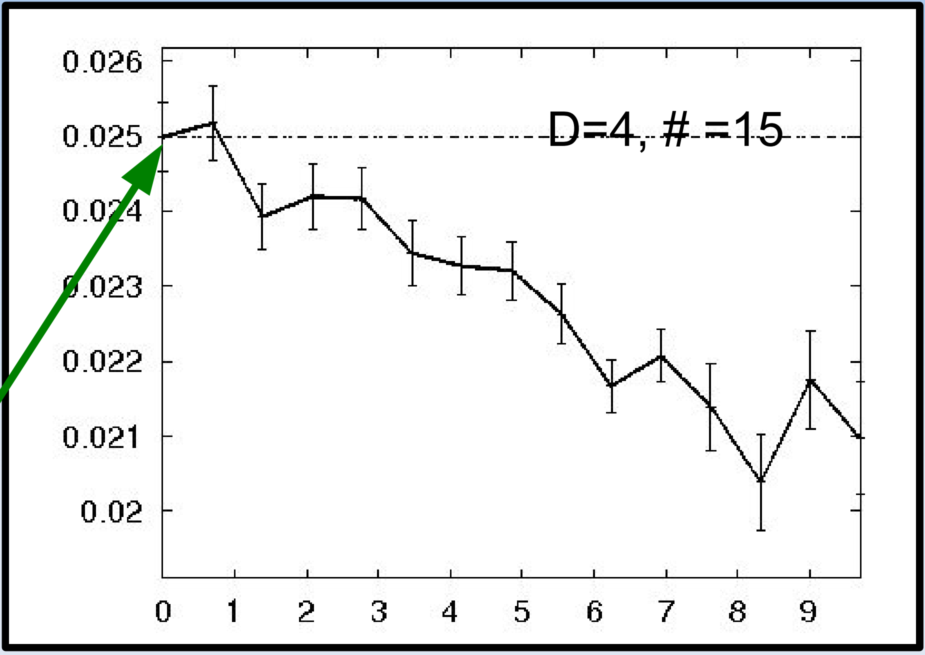
Some results

- Setting:
  - Linear sep. of R^D
  - Dimension : 4, 8
  - # queries: 15, 20
- X-axis: log(# sims)
  Y-axis: Gen. Error

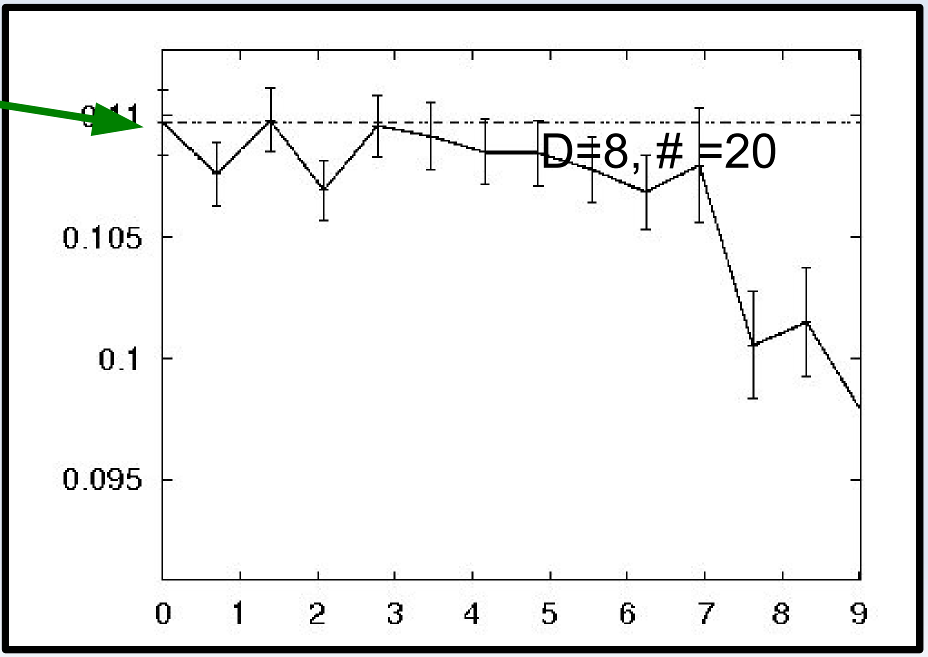


Some results

- Combining with AL criteria (inspired from QbC)
- Best of both worlds!



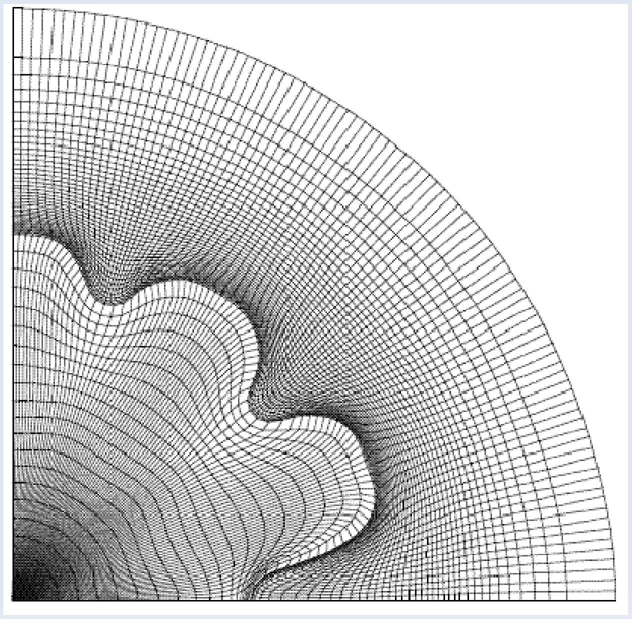
Almost optimal AL (QbC-based)



Partial Conclusion on BAAL

- A new approach to AL: *AL as a Game*
- Boosts heuristic to optimal strategy *(provably)*
- *Anytime* algorithm
- Straightforward extension to Optimization

Rolet, Sebag, Teytaud, 2009b



- **Perspectives:**
  - Kernelized Baal
  - Numerical engineering application

Apprentissage par Renforcement: Plan du cours

KDD 2009 – Orange

Targets

1. Churn
2. Appetency
3. Up-selling

Core Techniques

1. Feature Selection
2. Bounded Resources
3. Parameterless methods