Chapitre -
Monte Carlo Tree Search (MCTS)

1. Introduction

- Toujours Apprentissage dans l'incertain
- Arbres de recherche dans les jeux:
  + jeux à information complète
  + jeux à deux adversaire
  + jeux à somme nulle

( , echecs, Go, ...)

→ Solution: Algorithme MinMax (avec l'élagage $\alpha\beta$)

+ efficace pour les arbres à faible facteur de branchement
+ nécessite de "bonnes" fonction d'heuristiques

Exemple où la technique n'est pas du tout efficace : Go
- bon 250 en moyenne
- pas de très bonne heuristique.

2. Simulation de type Monte-Carlo.
   1. Simuler le jeu en effectuant des mouvements aléatoires
   2. Donner un score à la fin du jeu
   3. Mémoriser les statistiques liées aux jeux gagnants
   4. Jouer le mouvement avec le plus grand pourcentage de gain
   5. Répéter ces actions.

- Application au MinMax :
  Utiliser les simulation comme évaluation pour les plateaux

Problèmes : - Il faut faire (vraiment) beaucoup de simulations pour chaque plateau.
⇒ MC ont été négligés pendant très longtemps pour le jeu de Go...
- Tel quel, on va avoir tendance à chercher systématiquement de nouvelles positions alors que des régions (déjà explorées) peuvent être intéressantes.

⇒ Il faut un tradeoff entre exploration et exploitation

exploration vs exploitation

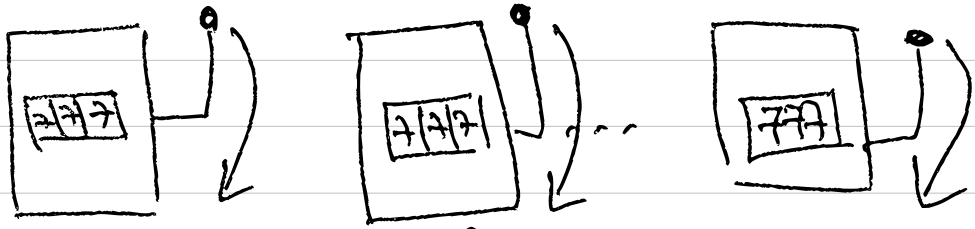
Principe : on souhaiterait utiliser le résultat de simulation déjà effectuées pour orienter la croissance de l'arbre de jeu

Exploitation : jouer le coup le plus prometteur (d'après la simulation déjà faite)

Exploration : jouer les coups pour lesquels l'incertitude est la plus haute

⇒ Théorie des bandits

Multi Armed Bandits



machine = bras = action

K machines : 1, 2, ..., K

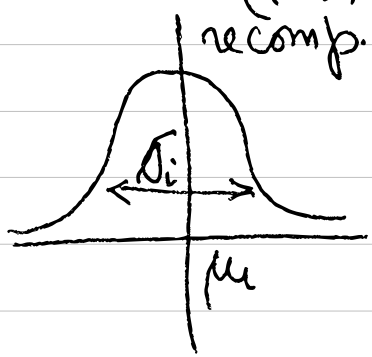
Chaque machine i a une distribution de récompense δ_i

Bernoulli $B(p)$

$\begin{cases} 1 \text{ avec proba } p \\ 0 \quad " \quad " \quad 1-p \end{cases}$

Distri. Normale

$\mathcal{N}(\mu_i, \sigma_i)$



$\mu = \mathbb{E}(\delta)$

$\sigma^2 = \mathrm{Var}(\delta)$

Hypothèse :

- On connaît le mais pas les distributions $(S_i)_{i=1,k}$

⇒ Simulations (MC) :
- à chaque étape t, on choisit un bras i et on génère une récompense $r(t) \in \mathbb{R}$ selon S_i

Objectifs (équivalents) :

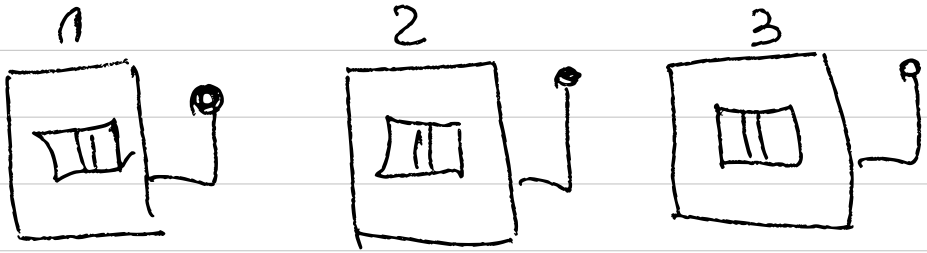
(a) Trouver le meilleur bras :

$$\arg\max_i \mu_i$$

(2) maximiser la récompense totale

$$\sum_{t \geq 1} r(t).$$

Problème: on doit faire un choix avec une information incomplète:



$\delta_1 = \begin{cases} 1 \text{ prob. } .6 \\ 0 \text{ — } .4 \end{cases}$ $\delta_2 \begin{cases} 1 \text{ prob. } .55 \\ 0 \text{ — } .45 \end{cases}$ $\delta_3 = \begin{cases} 1 \text{ pr. } .4 \\ 0 \text{ — } .6 \end{cases}$

⇒ le meilleur choix aurait été le bras 1 mais on ne le sait pas

⇒ On cherche à minimiser le "regret" = la perte par rapport à la meilleure suite de bras (si on avait l'information).

⇒ équilibre entre exploration ↑ politique uniforme et exploitation politique gloutonne (Greedy)

The ε-greedy algorithm

Après T étapes

pour chaque machine, on dispose d'un échantillon

→ $\hat{\mu}_i(T)$ espérance empirique

Greedy : on joue $\arg\max_i \hat{\mu}_i(T)$ (exploitation)

ε-greedy : on fixe $\varepsilon \in [0,1]$

- avec prob. ε, on choisit uniformément une action (exploration)

avec prob. $1-\varepsilon$ on choisit $\arg\max_i \hat{\mu}_i(T)$ (exploitation)

Analyse du regret : une reformulation

Regret$(T) = R(T)$ = diff. entre la meilleur a posteriori et ce qu'on obtient

$$\Rightarrow R(T) = T \cdot \mu_k - \sum_{t=1}^{T} r(t)$$

(μ_i est la vraie espérance et $\hat{\mu}_i(T)$ son estimation à l'instant T)

$$\Rightarrow \min R(T) \equiv \max \sum r(t)$$

UPPER Confidence Bounds (UCB)

$r(t)$ = recomp. à l'instant t

$$r(i,t) = \begin{cases} r(t) \text{ si } i \text{ est choisi à } t \\ 0 \text{ sinon} \end{cases}$$

$$R(t) = T\mu_k - \sum_{\tau=1}^{T} r(t)$$

$$\hat{\mu}_i(T) = \frac{1}{n(i,T)} \sum_{t=1}^{T} r(i,t)$$

UCB : $\arg\max_i \hat{\mu}_i(T) + C(i,T)$

avec $C(i,T) = \sqrt{\log(T)/n(i,T)}$

Intuitions :

- Si $n(i,T)$ est petit (peu d'information) alors $C(i,T)$ est grand
  $\Rightarrow$ Il faut explorer

- Si $n(i,T)$ est grand alors $C(i,T)$ est petit, sauf quand T est très grand

Question : pourquoi $C(i,T) = \sqrt{\frac{\log(T)}{n(i,T)}}$?

Borne de Chernoff-Hoeffding :

$Y_1, Y_2, \dots Y$ iid de même loi que Y

- Loi des grands nombres

$$\frac{1}{n}\sum y_i \longrightarrow \mathbb{E}(Y)$$

- $\mathbb{P}r\left(\left|\frac{1}{n}\sum y_i - \mathbb{E}(Y)\right| \geq c\right) \leq 2e^{-2c^2n^2}$

Application :

$$\Pr\left(\left|\hat{\mu}_i(T) - \mu_i\right| \geq c(i,T)\right) \leq 2e^{-2c(i,T)^2 n(i,T)^2}$$

avec $c(i,T) = \sqrt{\frac{\log(T)}{n(i,T)}}$, la somme devient $\frac{2}{T^2} \xrightarrow[T\to\infty]{} 0$

Mise à jour :

$$\hat{\mu}_i(T) = \frac{1}{n(i,T)} \sum_{t=1}^{T} r(i,t)$$

Si on choisit le bras i :

$$\hat{\mu}_i(T+1) = \hat{\mu}_i(T) + \frac{1}{n(i,T)}\left[\hat{\mu}_i(T) - X(i,T+1)\right]$$

new = old + α (old − current)

α : step size

MCTS.

- exploration d'un arbre de jeu :

4 Etapes:

1. Selection (d'une feuille à développer)
2. Expansion
3. Simulation (roll out)
4. Back-propagation

On met à jour les valeurs des ancêtres de la feuille explorée.

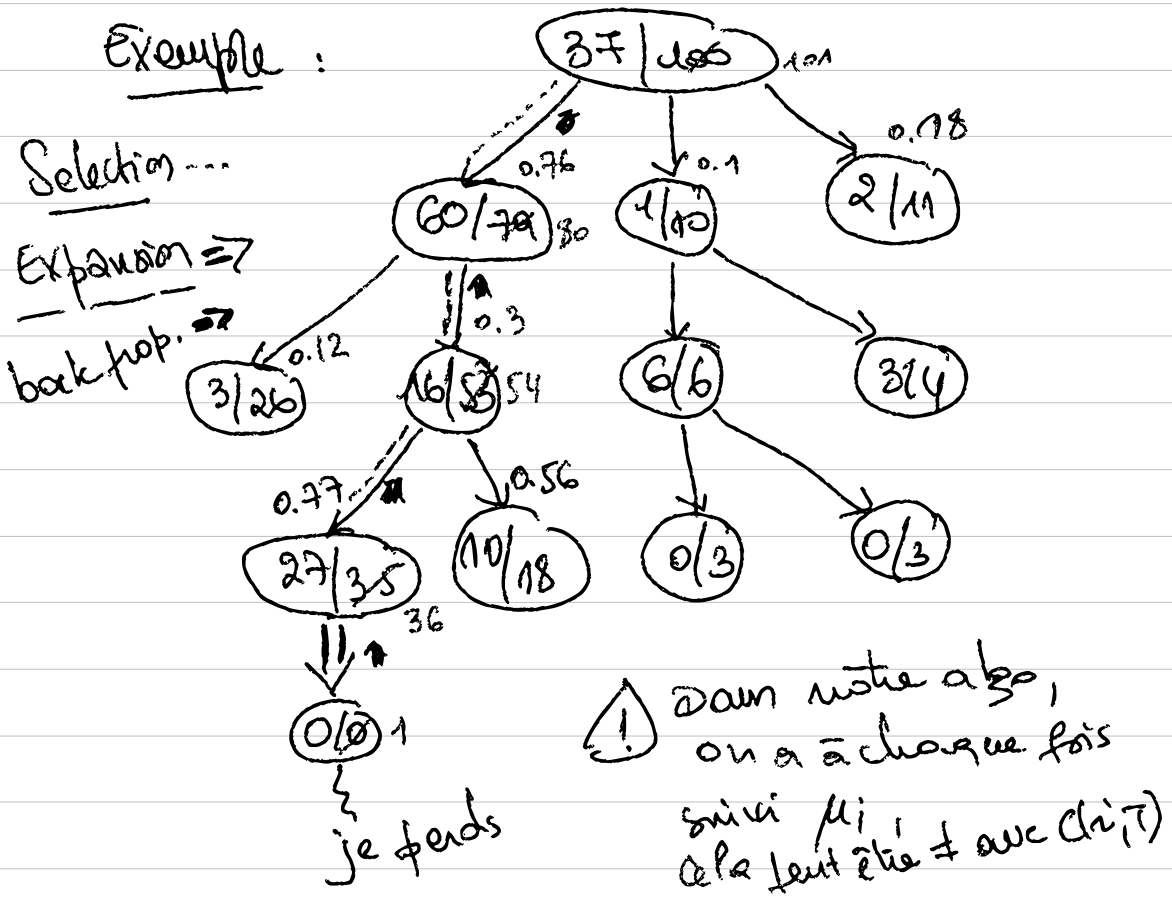
Upper Confidence Tree = MCTS + UCB

Comment faire la Selection ?
réponse : UCB.

UCB : $\underset{i}{\text{argmax}}\ \hat{\mu}_i(T) + c(i,T)$

$$\mu_i(T) = \frac{1}{n(i,T)} \sum_{t=1}^{T} r(i,t)$$

Exemple :



Selection ...

Expansion ⇒

backprop. ⇒

je perds

⚠ dans notre algo, on a à chaque fois suivi μ_i cela peut être ≠ avec $c(i,T)$

Simulation (roll out)

- Répéter
  1. a_i = random(action(s_i))
  2. s_i = simuler(a_i, s_i)

  jusqu'à s_i est terminal

- retourner valeur(s_i)

Application à l'apprentissage par renforcement :

Ce qu'on veut apprendre
- v-values = $v(s)$ pour chaque états
- q-values = $q(s,a)$ pour chaque paire (état, action) = (s,a)

Structure de données

plays : $S \longrightarrow \mathbb{N}$

plays(s) combien de simulations contiennent l'état s ?

$v : S \longrightarrow \mathbb{R}$

$v(s)$ = estimation de $v^*(s)$

$q : S, A \longrightarrow \mathbb{R}$

$q(s,a)$ = estimation de $q^*(s,a)$

Exercice : - MCTS pour V
- MCTS pour q.