TP : Upper Confidence Bound

1 Présentation du sujet

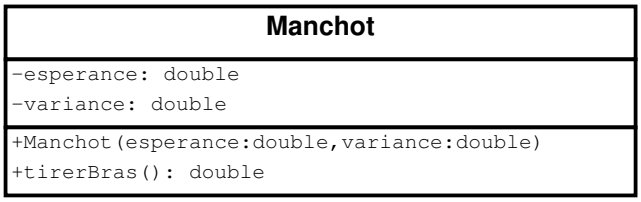
Pour ce TP nous allons créer un certain nombre de machines à sous. Chaque machine a une espérance de gain et une variance qui nous sont inconnues.

Le but est de se donner un certain nombre de tirages (représentant une somme d'argent initial) et de maximiser les gains avec cet argent.

Pour cela nous allons tester 3 algorithmes:

- Un tirage aléatoire.
- Une stratégie gloutonne (choix du meilleur).
- Un tirage selon la formule UCB vue en cours.

2 Implémentation d'une machine à sous



La fonction `tirerBras` représente le gain obtenu en choisissant cette machine. Pour plus de simplicité, je vous donne le code permettant de générer une variable aléatoire gaussienne:

tirerBras

```
double Manchot::tirerBras()
{
    double rand1 = ((rand() / (double) RAND_MAX ));
    double rand2 = ((rand() / (double) RAND_MAX ));
    return variance * sqrt(-2.0 * log(rand1)) * cos(2 * M_PI * rand2)
           + esperance;
};
```

3 Implémentation d'une stratégie aléatoire

Vous allez devoir maintenant créer une stratégie aléatoire. Créer une fonction
`double rechercheAleatoire(int nbIterations, Manchot[])`.

Dans cette fonction le choix du bras se fera aléatoirement parmi tous les bras possibles. La fonction va faire `nbIterations` tirages et va retourner la somme des gains cumulés.

4 Implémentation d'une stratégie gloutonne

Vous allez devoir maintenant créer une stratégie gloutonne. Créer une fonction
`double rechercheGloutonne(int nbIterations, Manchot[])`.

Dans cette fonction le choix du bras se fera comme suit :

1. Essayer chaque bras une fois et retenir le meilleur bras.
2. Pour toutes les itérations restantes choisir le bras retenu.

La fonction doit là encore retourner la somme des gains cumulés.

5 Implémentation d'une stratégie UCB

Vous allez devoir maintenant créer une stratégie UCB. Créer une fonction
`double rechercheUCB(int nbIterations, Manchot[])`.
Dans cette fonction le choix du bras se fera comme suit :

1. Créer deux tableaux pour stocker le nombre de tirages de chaque bras et la somme des scores de chaque bras.
2. Essayer chaque bras une fois et mettre à jour les deux tableaux précédents.
3. Pour toutes les itérations restantes choisir le bras en suivant la formule UCB vue en cours, en remplaçant la constante d'exploration ($\sqrt{2}$ dans le cours) par une variable K.

La fonction doit là encore retourner la somme des gains cumulés.

6 Tests

Créer une fonction `Manchot[] creerManchots(int nb)` qui va créer aléatoirement `nb` machines à sous. Les espérances et les variances de chaque machine seront aléatoires. Pour commencer, vous pouvez choisir l'intervalle $[-10, 10]$ pour l'espérance et $[0, 10]$ pour la variance.

Ensuite créer un programme principal qui va faire les 3 stratégies définies et comparer le score de chacune. Relancer plusieurs fois votre programme afin de regarder le comportement de chaque algorithme (pensez à initialiser la graine aléatoire, par exemple `srand(time(0))`. Faire varier:

- le nombre de machines,
- le nombre d'itérations,
- les intervalles d'espérances et de variances des machines.

Etudier le comportement d'UCB en faisant varier le paramètre K. Afin de mieux comprendre vous pouvez afficher le nombre de fois que chaque bras a été tiré (à la fin des itérations).

Voici un exemple d'exécution avec 15 machines et 15000 itérations, $K = \sqrt{2}$ et les intervalles par défaut:

execution

```
Score random -14744.2
Score Meilleur 16297
Répartition des essais UCB: 1 1 2 1 1 1 1 4 1 1 14957 1 1 6 21
Score UCB 104618
```

7 Questions

1. Comment se comporte UCB en fonction de K ?
2. Dans quel(s) cas la stratégie gloutonne est la meilleure ?
3. Dans quel(s) cas la stratégie UCB est la meilleure ?