Premier Bandit

Enseignant : L. Ralaivola
Date : 6 janvier 2014

1 Programmation d'un premier bandit

Objectif. L'objectif de cette planche est de guider la programmation d'un premier bandit à K-bras qui délivre des récompenses différentes (et aléatoires) pour chaque bras.

Modélisation mathématique. Un bandit à K-bras est une machine à sous munie de K leviers, chacun de ces leviers étant associé à une récompense « moyenne »éventuellement différente. En termes probabilistes, cela signifie que chaque levier i est associé à une loi de probabilité (fixe et inconnue) D_i d'espérance μ_i (inconnue également) : chaque tirage du levier fournit une récompense (i.e. un nombre réel) qui est une réalisation indépendante d'une variable X_i de loi D_i. On a donc : $\mu_i = \mathbb{E}[X_i]$.

Example 1. *Une manière de rendre concrètes les définitions et notations utilisées ci-avant est d'imaginer les situations suivantes :*

— *Un bras* i *ne peut prendre que* 2 *valeurs* $\{l_i, g_i\}$*, la première,* l_i*, avec une probabilité* $1-p_i$ *et la seconde,* g_i*, avec la probabilité* p_i *:* l_i *est par exemple le montant associé au fait de ne pas gagner (i.e. cela correspond à la mise du joueur) et* g_i *est le montant du gain lorsque la chance sourit au joueur (on peut donc imaginer que* p_i *est proche de* 0 *et que la probabilité de perdre* $1-p_i$ *est proche de* 1*). De manière formelle, si* X_i *est la variable aléatoire associée au tirage du bras* i*, les probabilités de perte et gain s'écrivent respectivement :*

$$\mathbb{P}(X_i = l_i) = 1 - p_i$$
$$\mathbb{P}(X_i = g_i) = p_i.$$

L'espérance de gain $\mathbb{E}[X_i]$ *associée au bras* i *est alors :*

$$\mathbb{E}[X_i] = l_i \cdot \mathbb{P}(X_i = l_i) + g_i \cdot \mathbb{P}(X_i = u_i) = l_i p_i + g_i(1 - p_i).$$

— *Un bras* i *peut prendre* n_i *valeurs* $\{v_1, \ldots, v_{n_i}\}$ *avec les probabilités* $\{p_1, \ldots, p_{n_i}\}$ *(et donc* $p_j \geqslant 0, \forall j = 1, \ldots n_i$ *et* $\sum_{j=1}^{n_i} p_j = 1$*). Formellement :*

$$\mathbb{P}(X_i = v_j) = p_j$$
$$\mathbb{E}[X_i] = \sum_{j=1}^{n_i} v_j p_j.$$

— *Un bras* i *peut prendre une infinité de valeurs. La variable aléatoire* X_i *associée est alors définie par une densité de probabilité* f_i*, avec* $f_i(v) \geqslant 0, \forall v$ *et* $\int f(v)dv = 1$

qui définit entièrement la distribution de X_i :

$$\mathbb{P}(X_i \in [a; b]) = \int_a^b f(\nu)d\nu$$
$$\mathbb{E}[X_i] = \int f(\nu)d\nu.$$

2 Des structures et des fonctions pour les bandits

Pour débuter, nous allons considérer des bandits à K bras très simples : ceux dont les bras ne peuvent prendre que deux valeurs (voir l'exemple précédemment décrit). Dans cette situation, Un bandit est donc constitué de K bras, donc chacun est lui-même associé avec 3 paramètres : la valeur d'une perte, la valeur d'un gain, et la probabilité associée à un gain.
Une manière de représenter ces éléments, c'est-à-dire un bandit et un jeu de paramètres, il est possible de définir 2 nouveaux types de données, l'un appelé bandit et l'autre appelé parametres, de la manière suivante :

```
#define MAX_BRAS 10

/** Des declarations anticipees des types          */
/** Ces renommages permettent d'alleger le code    */
typedef struct bandit bandit;
typedef struct parametres parametres;

struct bandit{
  /** Nombre de bras */
  int K;
  /** Un tableau de parametres: l'entree i correspond */
  /** aux parametres du bras i                        */
  parametres param[MAX_BRAS]
};

struct parametres{
  /** Le montant de la perte */
  double perte;
  /** Le montant d'un gain */
  double gain;
  /** La probabilite d'un gain */
  double proba_gain;
};
```

Ces types peuvent être utilisés de la manière suivante :

```
bandit b;
parametres p1, p2;
```

```
/** Les parametres pour le premier bras */
p1.perte = -1.0;
p1.gain = 10;
p1.proba_gain = 0.1;

/** Les parametres pour le second bras */
p2.perte = 0.0;
p2.gain = 2;
p2.proba_gain = 0.5;

/** Les parametres du bandit */
b.K = 2;
b.param[0] = p1;
b.param[1] = p2;
```

On peut noter que le nombre maximal de bras est fixé par MAX_BRAS, qui vaut ici 10. Lorsque les pointeurs seront introduits, nous verrons comment il est possible de ne pas limiter le nombre de bras à considérer *a priori*.
Pour utiliser les structures de données que l'on vient de définir, il est possible de définir deux fonctions, dont les fonctionnements sont décrits dans le commentaires qui les précèdent :

```
/** Fonction qui renvoie un double compris entre 0 et 1 */
double drand();

/** Renvoie la valeur de la recompense (gain ou perte)  */
/** lorsqu'on tire sur le bras i du bandit b */
double recompense(bandit b, bras i);
```

3 Travail à faire

Un premier travail à effectuer pour se familiariser avec les bandits est le suivant :

1. Programmer les fonctions drand et recompense déclarées ci-dessus (faire des recherches sur internet pour comprendre leur implémentation, si besoin).
2. Faire un programme principal (rappel : fonction **int** main()) qui fait plusieurs tirages en choisissant aléatoirement les bras choisis. Constater que la moyenne des récompenses pour chaque bras converge bien vers l'espérance de chaque bras.

On peut noter que les déclarations des types et les définitions des fonctions peuvent pour l'instant (et pour la dernière fois) toutes se trouver dans un même fichier. Nous verrons par la suite comment « séparer » un programme en plusieurs fichiers ou modules.