Aix Marseille Université — 2019-2020
Master IAAA — 2ème année

UE apprentissage par renforcement

Séance 2 - Bandits contextuels : LinUCB

On considère ici des bandits contextuels à K bras. Le contexte est donné par un vecteur $\mathbf{x} \in \mathbb{R}^d$ et on suppose que la récompense moyenne de chaque bras k suit un modèle linéaire en fonction du contexte : $\mathbb{E}[r|k, \mathbf{x}] = \mathbf{x}^T \boldsymbol{\theta}_k^*$ où $\boldsymbol{\theta}_k^* \in \mathbb{R}^d$ est le vecteur des coefficients de régression linéaire.
Pour simuler ce type de problème, on utilisera la classe `LinearBandits` fournie, et dont le fonctionnement repose sur les principes suivants :

- à l'initialisation du problème (constructeur de la classe), le vrai vecteur $\boldsymbol{\theta}_k^* \in \mathbb{R}^d$ de chaque bras k est généré aléatoirement.
- lorsqu'on choisit un bras k dans un contexte $\mathbf{x}$, la récompense est générée en tirant une variable aléatoire selon une loi normale $\mathcal{N}\left(\mathbf{x}^T \theta_k^*, 1\right)$ centrée en l'espérance $\mathbf{x}^T \theta_k^*$ et de variance 1 (voir méthode `step(self, a, x)`).
- l'environnement fournit un nouveau contexte via la méthode `LinearBandits.get_context()`, qui génère ici un vecteur $\mathbf{x} \in \mathbb{R}^d$ aléatoirement.

L'algorithme 1 détaille le déroulement des itérations pour linUCB.

Algorithme 1 LinUCB

Entrées: K (nombre de bras/actions), d (nombre de features), $\delta \in [0, 1]$ (niveau de confiance)

{Initialisation}
$\alpha \leftarrow 1 + \sqrt{\frac{1}{2} \ln \frac{2}{\delta}}$
pour chaque bras k **faire**
  $\mathbf{A}_k \leftarrow \mathbf{I}_d$
  $\mathbf{b}_k \leftarrow \mathbf{0}_d$
fin pour

{Itérations}
pour $t = 0, \ldots, T-1$ **faire**
  {Interaction avec l'environnement}
  Observer un nouveau contexte $\mathbf{x}$

  {Choix du bras}
  pour chaque bras k **faire**
    $\boldsymbol{\theta}_k \leftarrow \mathbf{A}_k^{-1} \mathbf{b}_k$
    $u_k \leftarrow \mathbf{x}^T \boldsymbol{\theta}_k + \alpha \sqrt{\mathbf{x}^T \mathbf{A}_k^{-1} \mathbf{x}}$
  fin pour
  Choisir le bras $\widehat{k} \leftarrow \text{argmax}_k \, u_k$

  {Interaction avec l'environnement}
  Obtenir la récompense $\widehat{r} \leftarrow r\left(\widehat{k}, \mathbf{x}\right)$

  {Mise à jour du modèle}
  Mettre à jour $\mathbf{A}_{\widehat{k}} \leftarrow \mathbf{A}_{\widehat{k}} + \mathbf{x}\mathbf{x}^T$
  Mettre à jour $\mathbf{b}_{\widehat{k}} \leftarrow \mathbf{b}_{\widehat{k}} + \widehat{r}\mathbf{x}$
fin pour

Exercice 1 (*Mise en œuvre*)

Dans la continuité des TP précédents sur les bandits,

1. complétez la classe `LinUCBAlgorithm` pour mettre en œuvre l'algorithme linUCB
2. utilisez l'environnement `LinearBandits` et réalisez une expérience, par exemple avec 30 bras, $d = 10$ et 1000 itérations.
3. tracez le regret en fonction des itérations

Exercice 2 (*Optimisation des calculs*)

Le calcul de l'inverse de $\mathbf{A}_{\widehat{k}}$ à chaque itération est coûteux, alors que la matrice a été mise à jour en ajoutant simplement une matrice de la forme $\mathbf{x}\mathbf{x}^T$, c'est-à-dire une matrice de rang 1. Il est possible d'optimiser les calculs en stockant $\mathbf{A}_{\widehat{k}}^{-1}$ au lieu de $\mathbf{A}_{\widehat{k}}$ et en utilisant une formule de mise à jour dite « de rang 1 » de $\mathbf{A}_{\widehat{k}}^{-1}$. Trouvez la formule de mise à jour en faisant une recherche sur internet et proposez une nouvelle implémentation. Mesurez et comparez les temps de calcul des deux implémentations (n'hésitez pas à faire grandir une des dimensions pour mettre en valeur le gain obtenu).