MAP 556 Challenge 2

Peng-Wei Chen

On utilise deux méthodes principales :

- Force de rappel : la base
- Optimisation finale avec les pôlynomes locaux

1 Force de rappel

D'après la formule de dynamique position, on peut simplement pousser l'oiseau vers sa position où il est sensé être au temps t. C'est à dire qu'on définit la force de rappel comme :

$$F(t, position) = -\begin{bmatrix} position[0] - 10t \\ position[1] - 20t + 2t^2 \end{bmatrix}$$

On pourrait aussi rajouter l'effet de frottement (λ_1) et le fait que l'effet de vent de t=9 à t=10 ne sont pas recompensé (μ). On essaie de repartir ces phénomènes-là avec un facteur devant les composants.

$$F(t, position) = -\mu \begin{bmatrix} \lambda_1 position[0] - 10t \\ \lambda_1 position[1] - 20t + 2t^2 \end{bmatrix}$$

Finalement, on trouve $\mu = 1.59$ et $\lambda_1 = 1.0125$. Et le coût est de l'ordre de 125 sur 10000 échantillons.

Quitte à mettre des coefficients devant tous les membres de l'équation, on se retrouve avec le résultat dans la section Optimisation finale.

2 Approche polynômes locaux

A partir de la force de rappel, on peut encore améliorer le résultat avec des polynômes locaux. On divise l'espace en $t = 0, t = 1, \ldots, t = 9$ et $x_i \in [a_{i_t}, b_{i_t}], i \in [1, 2]$ où a_{i_t}, b_{i_t} sont déterminés à partir des échantillons. Pour chaque bloc, on utilise un polynôme locale de degré 3 pour simuler le contrôle en plus de la force de rappel. Puis on utilise l'équation suivante pour la méthode de descente de gradient :

$$J(u) = \mathbb{E}\left[\int_0^T u_s^2 ds + L(x_T)\right], \dot{J}(u) = \mathbb{E}\left[\int_0^T 2 * v_s u_s ds + L'(x_T)\dot{x}_T\right]$$

On utilisait 2282 variables, et on réussit à descendre le côut à l'ordre de 120. Cette méthode n'est malheusement pas utilisé pour le code finale à cause de son efficacité: cela lui prend des heures pour diminuer 0.1 en coût.

3 Approche réseaux de neurones

Après avoir lu des articles, on a décidé de faire une approche de reinforcement learning avec le modèle de TD3 (Twin delayed DDPG), qui sert spécialement à ce type de jeu. Cependant le côut ne converge pas et le minimum était de l'ordre de 200.

4 Optimisation finale

On constate qu'on a souvent $V_{t,1} \sim -V_{t,2}$ (Il suffit de regarder la matrice de covariance pour la distribution au temps t). Ainsi, cela nous fait penser à diviser le plan en plusieurs blocs selons l'axe $y = -x + h_1(t)$ et $y = x + h_2(t)$ où $h_1(t), h_2(t)$ sont la position moyenne au temps t. Ensuite on réutilise la force de rappel comme la fonction de base pour un polynôme locale de degré 1 sur chaque bloc. Le côut est à 116 sur 10000 échantillons pour cette méthode. Quelques observations par rapport à la trajectoire ainsi obtenue :

- Pour tout $t \in \{0, 1, \dots, 9\}$, nous n'avons pas une distribution de X_t avec un ecart-type petit : En effet, nous gardons les informations sur l'intensité de vent selon sa position par rapport à la position moyenne au temps t.

- Pour le contrôle à $t = 9$, le meilleur n'est pas de l'envoyer à $(100, 0)$ même si on connaît le contrôle exacte pour le faire (Avec un réseau de neuronne j'ai pu toujours envoyer l'oiseau à (100, 0) à 2m près). Cela prend souvent plus de norme en contrôle et c'est moins rentable par rapport au coût de position finale.