

MAP 556 Challenge 2

Peng-Wei Chen

On utilise deux méthodes principales :

- Force de rappel : la base
- Optimisation finale avec les polynômes 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 a réduit encore le coût à l'ordre de 122.

2 Polynômes locaux

A partir de la force de rappel, on peut encore améliorer le résultat avec des polynômes locaux. On divise l'espace en $t = 0, t = 1, \dots, t = 9$ 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 coût à l'ordre de 120. Cette méthode n'est malheureusement pas utilisée pour le code final 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 coût 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 selon l'axe $y = -x + h(t)$ où $h(t)$ est la position moyenne au temps t . Ensuite on réutilise la force de rappel pour la fonction de base, puis optimiser les coefficients sur chaque bloc. Le coût est à 119 sur 10000 échantillons pour cette méthode.