Cours DQN – Section 3.3

3.3 – Échantillonnage aléatoire : casser les biais temporels

L'un des grands risques dans l'apprentissage par renforcement est d'entraîner le réseau de neurones sur des séquences d'expériences fortement corrélées, ce qui nuit à la généralisation et à la stabilité de l'agent. Pour résoudre ce problème, DQN introduit une stratégie simple mais puissante : l'**échantillonnage aléatoire** dans la mémoire de répétition (Replay Memory).

Problème : apprentissage biaisé par l'ordre des données

Sans échantillonnage aléatoire, les expériences utilisées pour l'apprentissage sont prises dans l'ordre où elles sont vécues. Cela entraîne plusieurs inconvénients :

— Les données successives sont souvent très similaires (corrélées temporellement).
— Le réseau peut suradapter ses poids à des séquences locales, perdant en robustesse globale.
— L'agent risque d'oublier les stratégies efficaces plus anciennes.

Solution : échantillonnage aléatoire dans la mémoire

Lors de chaque mise à jour du réseau de neurones, on sélectionne un **mini-batch aléatoire** d'expériences extraites de la mémoire.

$$\mathcal{B} = \{(s_i, a_i, r_i, s'_i, \text{done}_i)\}_{i=1}^{N}$$

Cet échantillonnage aléatoire permet de :

— Briser les corrélations temporelles.
— Fournir une diversité d'expériences dans chaque mise à jour.
— Favoriser une meilleure généralisation du réseau.

Analogie pédagogique

> Imaginez un étudiant qui révise uniquement le chapitre qu'il vient de lire. Il risque de négliger les autres. Avec l'échantillonnage aléatoire, il mélange toutes ses fiches de révision, et tire au hasard les notions à travailler. Résultat : sa compréhension devient plus globale et solide.

Détails techniques

— Un mini-batch contient typiquement 32 ou 64 expériences choisies au hasard.
— Chaque expérience contribue à la mise à jour des poids via la descente de gradient.
— Cette méthode est parfois améliorée par un échantillonnage *prioritaire*, mais le DQN de base utilise un tirage uniforme.

Conclusion

L'échantillonnage aléatoire est essentiel dans DQN car il permet :
— De stabiliser l'apprentissage.
— D'éviter les effets de surapprentissage séquentiel.
— De garantir une meilleure couverture de l'espace des expériences.

Grâce à cette technique, combinée à la Replay Memory et au Target Network, DQN peut apprendre de manière efficace et robuste dans des environnements complexes.