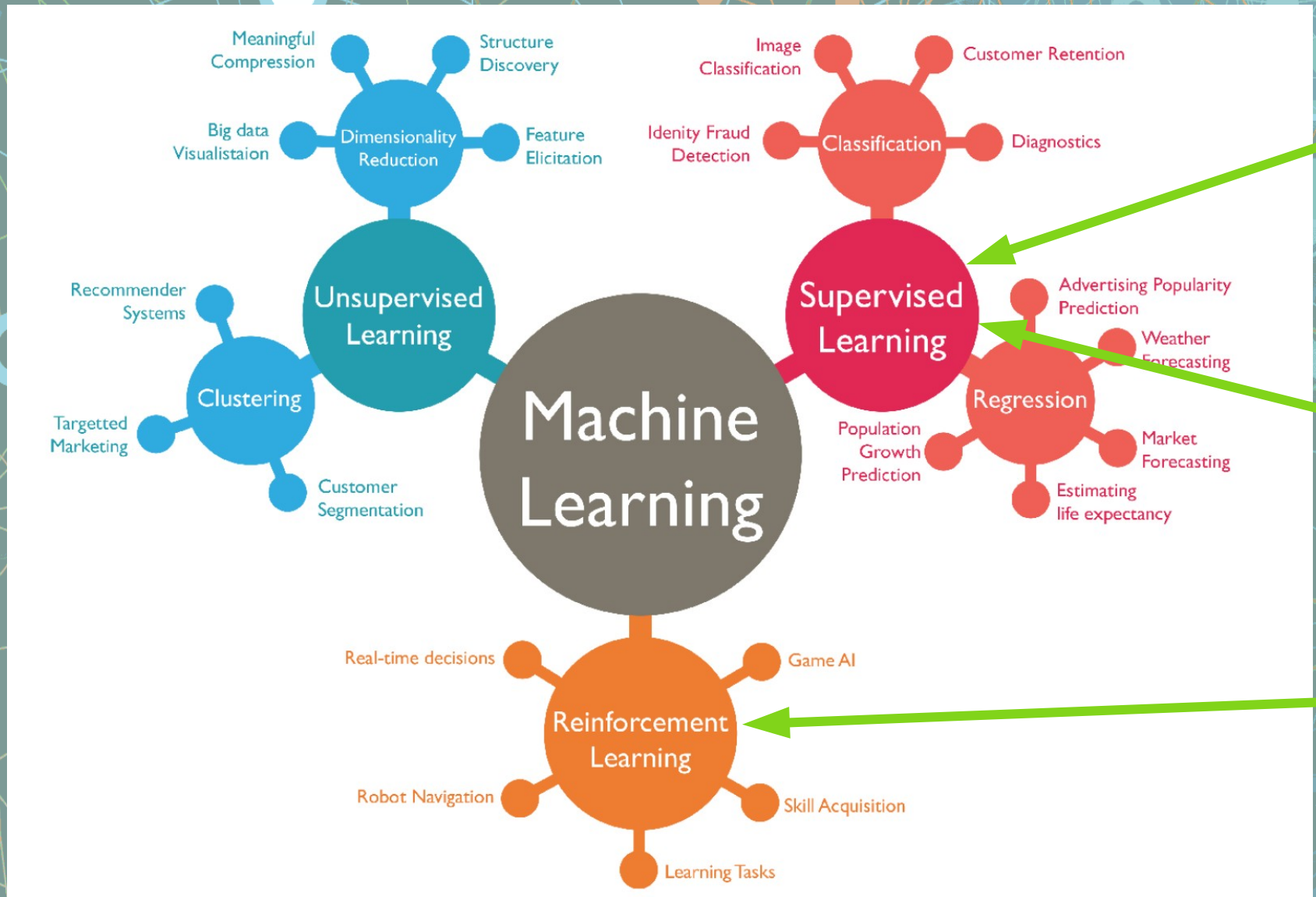
Apprentissage par Renforcement and Q-Learning

Vue d'Ensemble :



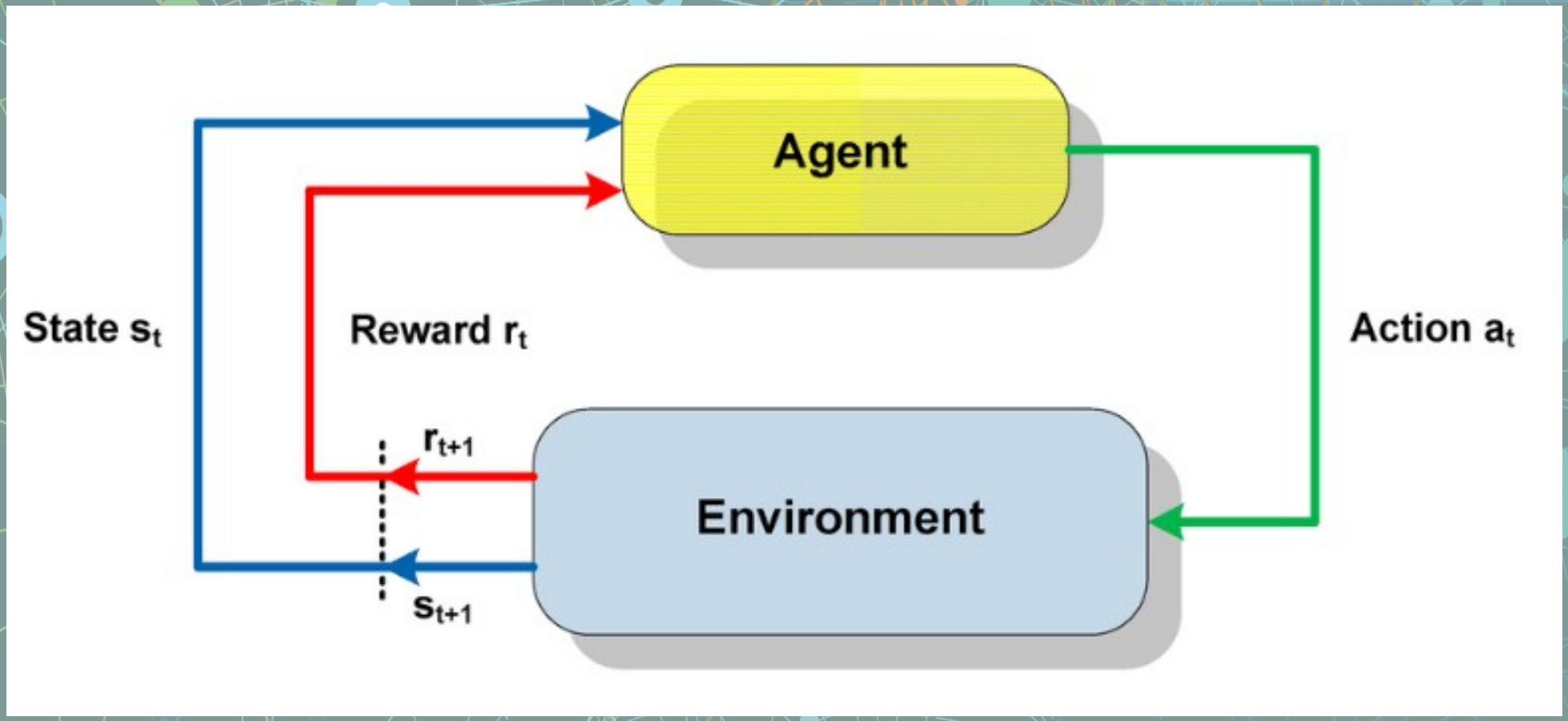
Réseaux de Neurones

Deep Learning

Q Learning

C’est quoi le but ?

Comment ça marche ?



Comment choisir nos action ?

K-Bandit Machine Problème

...

P1 = ?

P2 = ?

Pk = ?

a1

a2

ak

h tours

Agent

Stratégie = ?

Donc … Exploration vs Exploitation ?

Exploration

- Nécessaire pour découvrir les actions

- Mais pas optimisé sur le long terme

Exploitation

- Nécessaire pour optimiser nos actions

- Mais risque de s'obstiner sur une action non optimale

Il nous faut une Stratégie !

Le But de la Stratégie :

Horizon Infini avec facteur de réduction :

$$E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

Stratégies formellement justifiés:

Approche Récursive => Exponentielle ...

Stratégies formellement justifiés:

Approche Récursive => Exponentielle ...

Indices d'Allocation de Gittins => Non Généralisable ...

Stratégies formellement justifiés:

Approche Récursive => Exponentielle ...

Indices d'Allocation de Gittins => Non Généralisable ...

Apprentissage avec automates => Peut converger vers une mauvaise action ...

Stratégies non-justifiés … :

Greedy Strategie => Erreur quasi-assurée

Stratégies non-justifiés … :

Greedy Strategie => Erreur quasi-assurée

Random action avec proba p,
sinon Greey Strategie ! => Le choix random n'est pas optimisé

Stratégies non-justifiés … :

Greedy Strategie => Erreur quasi-assurée

Random action avec proba p,
sinon Greey Strategie ! => Le choix random n'est pas optimisé

Random Strategie mais avec les lois de Boltzmann : $$P(a)=\frac{e^{ER(a)/T}}{\sum_{a' \in A} e^{ER(a')/T}}$$

Récompenses différés => Markov Decision Processes (MDP)

- ensemble d'états S
- ensemble d'actions A
- fonction de récompense $R : S \times A \rightarrow \mathbb{R}$
- fonction de transition $T : S \times A \rightarrow P(S)$

- stratégie : $S \rightarrow A$
- valeur d'un état : $S \rightarrow \mathbb{R}$
- probabilité de transition : $S \times A \times S \rightarrow \mathbb{R}$

Stratégie optimale connaissant le modèle :

- valeur d'un état $V : S \rightarrow \mathbb{R}$

Stratégie optimale connaissant le modèle :

- valeur d'un état $V : S \rightarrow \mathbb{R}$

$$V^{star}(s) = \max_{\pi} E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

Stratégie optimale connaissant le modèle :

- valeur d'un état $V : S \rightarrow \mathbb{R}$

$$V^{star}(s) = \max_{\pi} E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

$$V^{star}(s) = \max_{a}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{star}(s')\right)$$

Stratégie optimale connaissant le modèle :

- valeur d'un état $V : S \rightarrow \mathbb{R}$

$$V^{star}(s) = \max_{\pi} E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

$$V^{star}(s) = \max_{a}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{star}(s')\right)$$

$$\pi^{star}(s) = \operatorname*{argmax}_{a}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{star}(s')\right)$$

Donc on a un algorithme !

- initialiser V arbitrairement
- tant que la stratégie n'est pas assez précise …
  - pour tous les états s
    - pour toutes les actions a
      - $Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V(s')$
    - $V(s) = \max_a Q(s,a)$
  - fin

fin

Et quand on connaît pas le modèle ?

- valeur d'une action d'un état $Q : S \times A \rightarrow \mathbb{R}$

Et quand on connaît pas le modèle ?

- valeur d'une action d'un état $Q : S \times A \rightarrow \mathbb{R}$

$$V^{star}(s) = \max_{a} Q^{star}(s, a)$$

Et quand on connaît pas le modèle ?

- valeur d'une action d'un état $Q : S \times A \rightarrow \mathbb{R}$

$$V^{star}(s) = \max_{a} Q^{star}(s,a)$$

$$Q^{star}(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q^{star}(s',a')$$

Et quand on connaît pas le modèle ?

- valeur d'une action d'un état $Q : S \times A \rightarrow \mathbb{R}$

$$V^{star}(s) = \max_{a} Q^{star}(s,a)$$

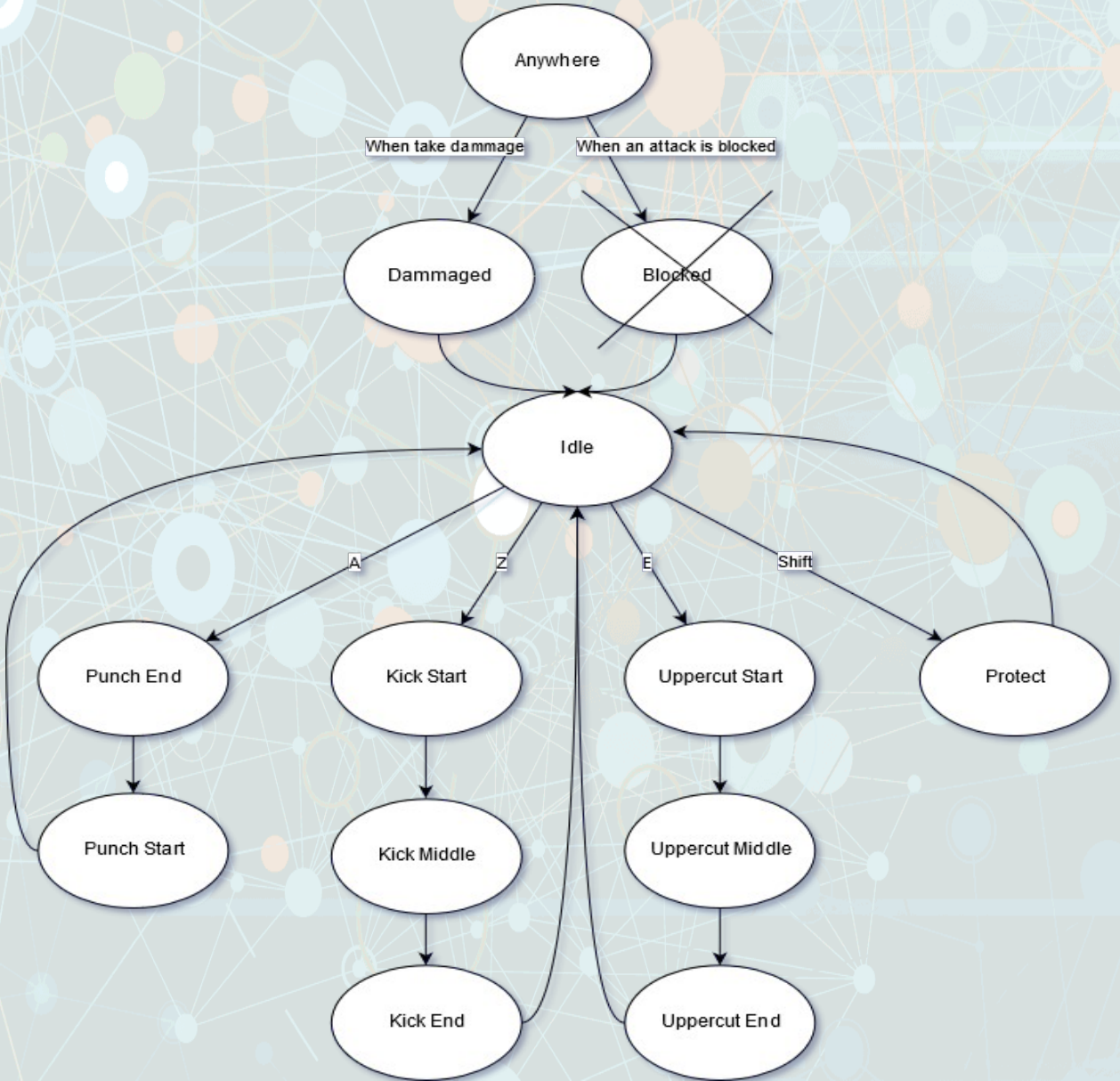
$$Q^{star}(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q^{star}(s',a')$$

THE Q-Learning rule :

$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma\left(\max_{a'} Q(s',a') - Q(s,a)\right)\right)$$

Un exemple peut-être ?

La logique du jeu :



Correspondances avec le MDP

- S = l'état du jeu
- A = les actions possibles de l'agent
- R = à nous de le récompenser pour qu'il fasse ce que l'on veut
- T = c'est la logique du jeu … mais on pas besoin de la connaître !

Pendant la phase d’exploration ...

Après un peu d'entraînement

Après 15 000 matchs d'entraînement … !

Avantages et Limites

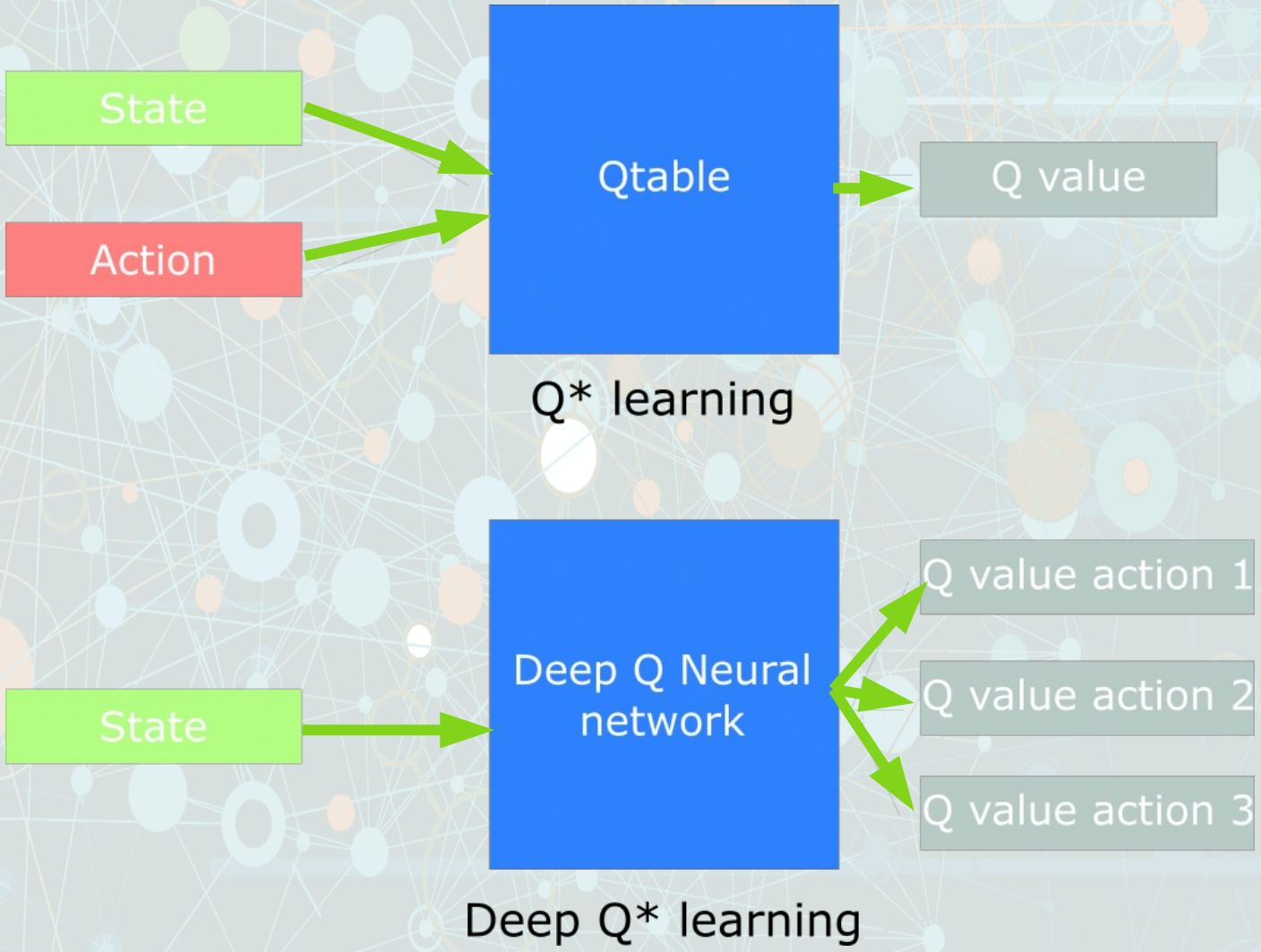
Avantages

- Pas besoin de coder la logique du jeu
- Pas besoin de données
- Très performant
- Facile d'adapter la performance de l'agent
- Extrêmement rapide une fois en jeu
- Continue de s'adapter tout au long de sa vie

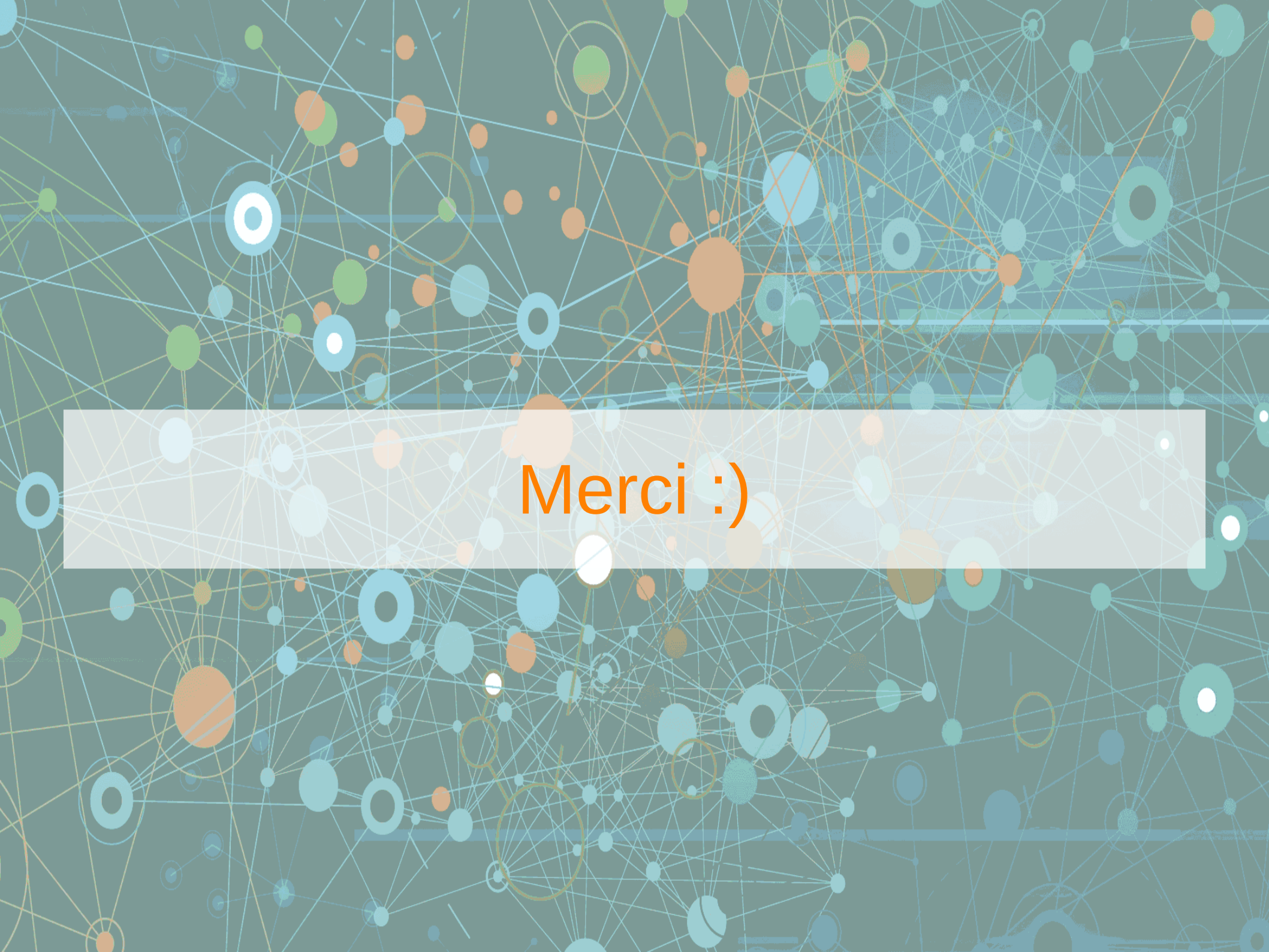
Limites

- Temps de pré-calculs très importants
- Espace mémoire complètement astronomique
- Demande un univers discret
- Doit pouvoir être simulé, peu adapté à la vie réelle

Vers une amélioration … !



Exemple d’utilisations dans des jeux :


Merci :)