# Supplément sur l'apprentissage par renforcement

IFT-17587
Concepts avancés pour systèmes intelligents
Luc Lamontagne



## Plan de la présentation

- Exemple de *TD-Learning*
- Approximation de la fonction d'évaluation du *Q-Learning*
  - Réseau de neurones
- Approximation de *Q-Learning* avec *kNN*



Rappel :

## *TD-Learning*

- Exploitation : Nous avons une politique prédéterminée $\pi : S \rightarrow A$
- Entraînement : Évaluer la politique !
  - Utiliser les transitions $s \rightarrow s'$ pour ajuster les valeurs des états observés.
  - Les valeurs devraient être cohérentes avec les équations de contraintes
  - Pas d'apprentissage de modèle de transitions.
- Mise à jour

$$U^{\pi}(s) \leftarrow (1-\alpha)U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s'))$$

  - $\alpha$ : le paramètre du taux d'apprentissage
  - $\gamma$ : le facteur d'escompte.
- *TD = temporal difference*
  - Utilise les différence d'utilités entre états successifs.



Rappel :

## *TD-Learning*

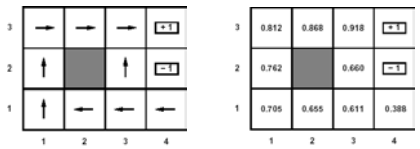
```
function PASSIVE-TD-AGENT(percept) returns an action
  inputs: percept, a percept indicating the current state s' and reward signal r'
  static: π, a fixed policy
          U, a table of utilities, initially empty
          N_s, a table of frequencies for states, initially zero
          a, s, r, the previous state, action, and reward, initially null

  if s' is new then U[s'] ← r'
  if s is not null then do
      increment N_s[s]
      U[s] ← U[s] + α(N_s[s])(r + γU[s'] − U[s])
  if TERMINAL?[s'] then s, a, r ← null else s, a, r ← s', π[s'], r'
  return a
```

Apprentissage par renforcement – *TD-Learning*:

# Exemple de la grille 3X4

- On mène 3 essais d'entraînement :
  - La récompense pour chaque état non terminal est -0.04.
  - Le taux d'apprentissage $\alpha$ = 0,5.
  - Le facteur d'escompte est $\gamma$ = 1.
- On obtient les séquences d'états suivants ainsi que les récompenses correspondantes :

$(1,1)_{-0.4} \rightarrow (1,2)_{-0.4} \rightarrow (1,3)_{-0.4} \rightarrow (1,2)_{-0.4} \rightarrow (1,3)_{-0.4} \rightarrow (2,3)_{-0.4} \rightarrow (3,3)_{-0.4} \rightarrow (4,3)_{+1}$
$(1,1)_{-0.4} \rightarrow (1,2)_{-0.4} \rightarrow (1,3)_{-0.4} \rightarrow (2,3)_{-0.4} \rightarrow (3,3)_{-0.4} \rightarrow (3,2)_{-0.4} \rightarrow (3,3)_{-0.4} \rightarrow (4,3)_{+1}$
$(1,1)_{-0.4} \rightarrow (2,1)_{-0.4} \rightarrow (3,1)_{-0.4} \rightarrow (3,2)_{-0.4} \rightarrow (4,2)_{-1}$





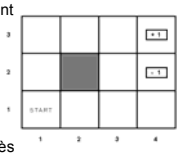

Apprentissage par renforcement – *TD-Learning*:

# Exemple de la grille 3X4

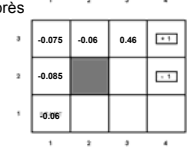
## Séquence d'entraînement #1 :

$(1,1)_{-0.4} \rightarrow$ U(1,1) = -0.04 // nouveau
$(1,2)_{-0.4} \rightarrow$ U(1,2) = -0.04 // nouveau
U(1,1) = 0.5 (-0.04) + 0.5 (-0.04 - 0.04) = -0.06
$(1,3)_{-0.4} \rightarrow$ U(1,3) = - 0.04 // nouveau
U(1,2) = 0.5 (-0.04) + 0.5 (-0.04 - 0.04) = -0.06
$(1,2)_{-0.4} \rightarrow$ U(1,2) pas nouveau
U(1,3) = 0.5 (-0.04) + 0.5 (-0.04 - 0.06) = -0.07
$(1,3)_{-0.4} \rightarrow$ U(1,3) pas nouveau
U(1,2) = 0.5 (-0.06) + 0.5 (-0.04 – 0.07) = -0.085
$(2,3)_{-0.4} \rightarrow$ U(2,3) = -0.04 // nouveau
U(1,3) = 0.5 (-0.07) + 0.5 (-0.04 - 0.04) = -0.075
$(3,3)_{-0.4} \rightarrow$ U(3,3) = -0.04 // nouveau
U(2,3) = 0.5 (-0.04) + 0.5 (-0.04 - 0.04) = -0.06
$(4,3)_{+1} \rightarrow$ U(4,3) = +1 // Terminal
U(3,3) = 0.5 (-0.04) + 0.5 (-0.04 + 1) = 0.46
*S,A,r* ← null

Avant




Après





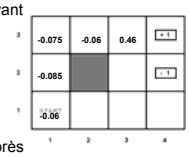

Apprentissage par renforcement – *TD-Learning*:

# Exemple de la grille 3X4

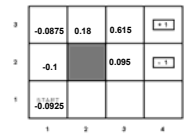
## Séquence d'entraînement #2 :

$(1,1)_{-0.4} \rightarrow$
$(1,2)_{-0.4} \rightarrow$ U(1,1) = 0.5 (-0.06) + 0.5 (-0.04 - 0.085) = -0.0925
$(1,3)_{-0.4} \rightarrow$ U(1,2) = 0.5 (-0.085) + 0.5 (-0.04 +0.075) = -0.1
$(2,3)_{-0.4} \rightarrow$ U(1,3) = 0.5 (-0.075) + 0.5 (-0.04 - 0.06) = -0.0875
$(3,3)_{-0.4} \rightarrow$ U(2,3) = 0.5 (-0.06) + 0.5 (-0.04 + 0.46) = 0.18
$(3,2)_{-0.4} \rightarrow$ U(3,2) = -0.04 // nouveau
U(3,3) = 0.5 (0.46) + 0.5 (-0.04 - 0.04) = 0.27
$(3,3)_{-0.4} \rightarrow$ U(3,2) = 0.5 (-0.04) + 0.5 (-0.04 + 0.27) = 0.095
$(4,3)_{+1} \rightarrow$ U(3,3) = 0.5 (0.27) + 0.5 (-0.04 + 1) = 0.615
*S,A,r* ← null

Avant




Après





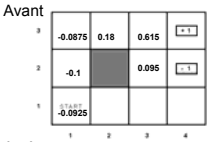

Apprentissage par renforcement – *TD-Learning*:

# Exemple de la grille 3X4

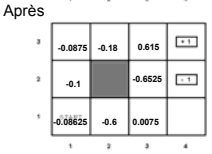
## Séquence d'entraînement #3 :

$(1,1)_{-0.4} \rightarrow$
$(2,1)_{-0.4} \rightarrow$ U(2,1) = -0.04 // Nouveau
U(1,1) = 0.5 (-0.0925) + 0.5 (-0.4 – 0.4) = -0.08625
$(3,1)_{-0.4} \rightarrow$ U(3,1) = -0.04 // Nouveau
U(2,1) = 0.5 (-0.04) + 0.5 (-0.04 - 0.04) = -0.06
$(3,2)_{-0.4} \rightarrow$ U(3,1) = 0.5 (-0.04) + 0.5 (-0.04 + 0.095) = 0.0075
$(4,2)_{-0.4} \rightarrow$ U(4,2) = -1 // Terminal
U(3,2) = 0.5 (0.095) + 0.5 (-0.4 - 1) = -0.6525
*S,A,r* ← null

Avant




Après



Apprentissage par renforcement – *TD-Learning*:

## Exemple de la grille 3X4

Faudrait-il plusieurs essais à partir de :

| | | | | |
|---|---|---|---|---|
| 3 | -0.0875 | -0.18 | 0.615 | +1 |
| 2 | -0.1 | | -0.6525 | -1 |
| 1 | -0.08625 | -0.6 | 0.0075 | |
| | 1 | 2 | 3 | 4 |

pour arriver à :

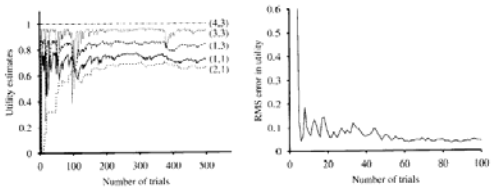
| | | | | |
|---|---|---|---|---|
| 3 | 0.812 | 0.868 | 0.918 | +1 |
| 2 | 0.762 | | 0.660 | -1 |
| 1 | 0.705 | 0.655 | 0.611 | 0.388 |
| | 1 | 2 | 3 | 4 |



Apprentissage par renforcement – *TD-Learning*:

## Exemple de la grille 3X4

- Exemple de convergence






Apprentissage par renforcement :

## *Q-Learning*

- Apprendre la politique optimale
  - la politique $\pi$ n'est pas fixe.
  - Apprendre à sélectionner la prochaine action $a_t$ en se basant sur l'état $s_t$
    $\pi : S \rightarrow A$
  - L'agent apprend une fonction *Q*.
  - La qualité d'une action pour un état donné.
- Elle représente
  la récompense immédiate obtenue en exécutant l'action *a* dans l'état *s*
  +
  la valeur obtenue en suivant la politique optimale par la suite.

$$Q(a,s) \leftarrow (1-\alpha)\ Q(a,s) + \alpha\ (R(s) + \gamma\ max_a Q(a',s'))$$

- Lien entre utilités et Q-valeurs
  $U(s) = max_a\ Q(a,s)$



Apprentissage par renforcement :

## *Q-Learning*

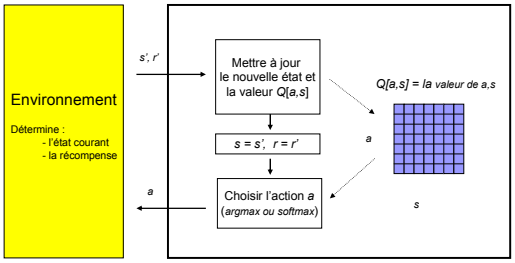
**function** Q-LEARNING-AGENT(*percept*) **returns** an action
  **inputs**: *percept*, a percept indicating the current state $s'$ and reward signal $r'$
  **static**: $Q$, a table of action values index by state and action
    $N_{sa}$, a table of frequencies for state-action pairs
    $s$, $a$, $r$, the previous state, action, and reward, initially null

  **if** $s$ is not null **then do**
    increment $N_{sa}[s, a]$
    $Q[a, s] \leftarrow Q[a, s] + \alpha(N_{sa}[s, a])(r + \gamma \max_{a'} Q[a', s'] - Q[a, s])$
  **if** TERMINAL?[$s'$] **then** $s, a, r \leftarrow$ null
  **else** $s, a, r \leftarrow s', \text{argmax}_{a'} f(Q[a', s'], N_{sa}[a', s']), r'$
  **return** $a$

Apprentissage par renforcement :

# *Q-Learning*

La fonction de valeurs est représentée par un tableau.




Que faire lorsque le nombre d'états et le nombre d'actions sont très grands ?



Apprentissage par renforcement :

# Problème de dimensionnalité

- Le tableau de valeur représente la fonction
  $Q : A,S \rightarrow \mathbb{R}$
- Par exemple, pour *Tetris*
  - Chaque état est représenté par 12 attributs :
    - 10 colonnes, le type de pièce et son orientation
  - En gros ~ $20^{11}$ états et ~ 200 actions possibles
- Pour le Backgammon
  - Le nombre de pièces à chaque position
  - Nombre d'états > $10^{20}$ , Nombre d'actions > 400
- Idée : approximer la fonction Q !



Apprentissage par renforcement :

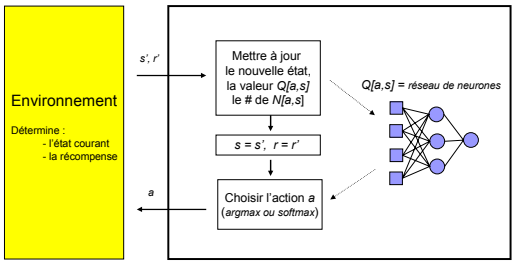
# *Q-Learning avec approximation*

- Approximation → généralisation
  - Représentation plus concise.
  - On peut utiliser l'apprentissage supervisé pour déterminer les paramètres du modèle (hypothèse).
- Options
  - Régression linéaire
    - Pas étudiée dans le cours.
    - Difficile lorsque les attributs d'un état ne sont pas numériques.
  - Arbre de décision
    - Pas bien adaptée aux classes numériques.
    - Il existe un équivalent numérique : *CART* (pas étudié…).
  - Apprentissage bayésien naïf
    - Même problème.
  - Réseaux de neurones
    - Ah ah!



Apprentissage par renforcement :

# *Q-Learning avec approximation*

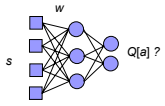



On remplace le tableau par le réseau de neurone. Et après ?

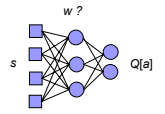
Apprentissage par renforcement :

## *Q-Learning avec réseau de neurones*

- Réseau de neurones
  - Raisonnement : *Eval-Net*
    - Obtenir une estimation de *Q* pour chaque action *a* étant donné un état *s*.
    - Propagation vers l'avant.




  - Apprentissage : *Backpropagation*
    - Déterminer les poids du réseau pour un état *s*, une action *a* et sa valeur *Q*[*a*].
    - Propagation vers l'arrière.






Apprentissage par renforcement :

## *Q-Learning avec réseau de neurones*

**fonction** Q-Learning-Agent (*percepts*) **retourne** une action
  inputs : *percepts* = *s'*, *r'*
  static : *Network* = un réseau de neurones avec poids initialisée arbitrairement
    *s*, *a*, *r* = l'état précédent, l'action et la récompense (nuls au début)

  **if** *s'* is not null **then do**
    *Q*[] ← Eval-net(*Network*, *s'*)  // Estimation de *Q* pour chaque action

  **if** *s* is not null **then do**
    Backpropagation(*Network*, *s*, *a*, $r + \gamma \max_j Q[j]$, $\alpha$)  // Mise à jour de l'état *s*

  **if** Terminal?[*s'*] **then** *s*, *a*, *r* ← null

  **else** *s*, *a*, *r* ← s', $\text{argmax}_j Q[j]$, *r'*  // Choisir l'action -- celle avec le Q maximum
    // On pourrait faire de l'exploration avec softmax

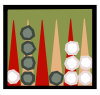
  **retourne** *a*

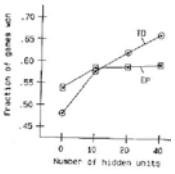


Apprentissage par renforcement :

## *Q-Learning avec réseau de neurones*

- Application pour *le Backgammon*
  - Les états sont les configurations du plateau.
  - Des attributs supplémentaires sont ajoutés pour représenter des éléments de stratégies.
- Entraînement
  - Les poids du réseau sont attribués aléatoirement.
  - Le système joue contre une copie de lui-même.
  - Premiers milliers de parties
    - Apprentissage de stratégies et tactiques élémentaires.
  - Meilleur résultat après 200 000 parties (40 neurones cachées)
- Compétition
  - Du même niveau que le champion du monde (*Robertie*)








# Conclusion

- L'apprentissage par renforcement s'appuie sur le paradigme MDP.
- Il est utile lorsqu'on ne connaît pas le modèle de l'environnement.
- L'apprentissage passif de type *temporal difference* (*TD-Learning*) s'applique pour des politiques fixes.
  - Elle permet de faire converger les estimations de valeurs sans connaître le modèle de l'environnement.
  - D'autres méthodes, comme la programmation dynamique adaptative, permettent d'apprendre le modèle de transition.
- L'apprentissage actif de type *Q-Learning* guide le choix des actions pour un état donné.
  - On apprend donc la politique.
- Pour les espaces d'états très grands, il faut des techniques qui approximent les estimations d'utilités.
  - Réseaux de neurones.
  - Raisonnement à base d'exemples (*kNN*).