Douzième partie XII

Apprentissage

En bref ...

L'apprentissage automatique

Apprentissage supervisé

Apprentissage non-supervisé

Apprentissage par renforcement

Conclusion

Plan

1. Introduction à l'intelligence artificielle
2. Agents intelligents
3. Algorithmes classiques de recherche en IA
4. Algorithmes et recherches heuristiques
5. Programmation des jeux de réflexion
6. Problèmes de satisfaction de contraintes
7. Agents logiques
8. Logique du premier ordre
9. Inférence en logique du première ordre
10. Introduction à la programmation logique avec Prolog
11. Planification
12. Apprentissage

L'apprentissage automatique

Apprentissage automatique

Quelques définitions :

- Wikipédia :
  - L'apprentissage automatique fait référence au développement, à l'analyse et à l'implémentation de méthodes qui permettent à une machine (au sens large) d'évoluer et ainsi de remplir des tâches qu'il est difficile ou impossible de remplir par des moyens algorithmiques plus classiques
- Herbert Simon :
  - "L'apprentissage dénote des changements dans un système qui ... lui permet de faire la même tâche plus efficacement la prochaine fois."

Apprentissage automatique

- On dira qu'une machine apprend dès lors qu'elle change sa structure, son programme ou ses données en fonction de données en entrée ou de réponses à son environnement de sorte à ce que ses performance futures deviennent meilleures

Apprentissage automatique

- On dira qu'une machine apprend dès lors qu'elle change sa structure, son programme ou ses données en fonction de données en entrée ou de réponses à son environnement de sorte à ce que ses performance futures deviennent meilleures

→ L'objectif de l'apprentissage automatique est de concevoir des programmes pouvant s'améliorer automatiquement avec l'expérience

Pourquoi l'apprentissage automatique ?

- Certaines tâches ne sont bien définies que via un ensemble d'exemples
  - On n'est capable de spécifier des relations entre les entrées et les sorties
- Pour découvrir des relations importantes dans des données (fouille de données)
- Les machines peuvent ne pas fonctionner sur tous les environnements
  - Certains aspects des environnements peuvent être inconnus lors de la conception
- La quantité de connaissances disponibles à propos de certaines situations sont telles que le cerveau humain ne puisse les expliciter
  - L'apprentisage peut permettre de mieux exploiter ces connaissances
- L'environnement change constamment
  - L'apprentisage permet aux machines de s'adapter aux changements

Pourquoi l'apprentissage automatique ?

Exemples de systèmes apprenant :

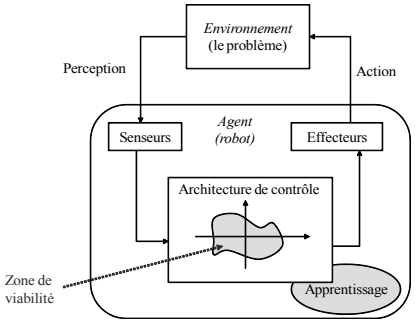
- Un robot ayant la capacité de bouger ses membres mais ne sachant initialement rien de la coordination des mouvements permettant la marche, peut apprendre à marcher.
  - Le robot commencera par effectuer des mouvements aléatoires, puis, en sélectionnant et privilégiant les mouvements lui permettant d'avancer, mettra peu à peu en place une marche de plus en plus efficace.
- La reconnaissance de caractères manuscrits est une tâche complexe car deux caractères similaires ne sont jamais exactement égaux
  - Un système d'apprentissage automatique peut apprendre à reconnaître des caractères en observant des exemples

Pourquoi l'apprentissage automatique ?

- Un des dix plus grands enjeux du XXI ème siècle (MIT Technology review) :
  - Compréhension et amélioration de l'apprentissage humain (ex : instruction assistée par ordinateur)
  - Découverte de nouvelles connaissances ou structures (ex : fouille de données)
  - Paramétrage automatique de systèmes complexes et/ou dynamiques
- Applications de l'apprentissage :
  - Traitement du langage naturel (fouille de textes), reconnaissances des formes, moteurs de recherche, diagnostic médical, bioinformatique, biochimie, finance (détection de fraude (à la carte bancaire), ?) , analyse des marchés boursiers, jeux, robotique, ...

Robots et apprentissage

- Senseurs qui s'adaptent aux perceptions
  - filtrage des perceptions
  - mécanisme d'attention
- Effecteurs qui s'adaptent à l'environnement
  - détection et correction des erreurs d'exécution
  - améliorer l'exécution d'un comportement
- Une architecture de contrôle qui apprend
  - reconnaissance des couleurs, des formes
  - modélisation de l'environnement
  - organiser les connaissances et la prise de décision
  - apprendre le comportement adéquate selon la situation
  - optimiser un comportement par l'expérience



Types d'apprentissage

- Tout apprentissage s'opère à partir d'exemples de données
- Selon les informations disponibles, l'apprentissage peut prendre plusieurs formes

→ 3 grands types d'apprentissage :

- Apprentissage supervisé
  - Chaque exemple est associé à une étiquette
  - Objectif : prédire l'étiquette de chaque donnée
  - → Le système apprend à classer les données
- Apprentissage non-supervisé
  - Les exemples ne sont pas étiquetés
  - Objectif : trouver une structure aux données
  - → Le système apprend une classification des données
- Apprentissage par renforcement
  - Les exemples sont (parfois) associés à une récompense ou une punition
  - Objectif : trouver les actions qui maximisent les récompenses
  - → Le système apprend une politique de décision

Apprentissage supervisé

Apprentissage supervisé

- Aussi appelé analyse discriminante
- Les données d'apprentissage sont etiquetées
  - Un expert ou oracle doit préalablement étiqueter des exemples.
- Le processus se passe en deux phases :
  - La phase d'apprentissage (*hors ligne*) : déterminer un modèle des données étiquetées
  - La phase de test (*en ligne*) : prédire l'étiquette d'une nouvelle donnée, connaissant le modèle préalablement appris.

Définition formelle

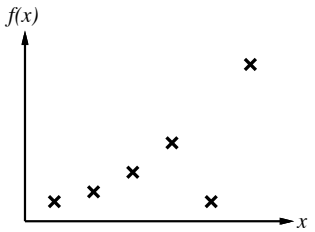
- Données d'apprentissage
  - N couples entrée-sortie $(x_n, y_n)_{1 \leq n \leq N}$ avec $x_n \in X$ et $y_n \in Y$
  - On suppose que ces données sont tirées selon une loi (de probablilité) inconnue
- Objectif de l'apprentissage
  - déterminer une fonction de prédiction $f : X \to Y$ qui soit *en accord* avec le données d'apprentissage

$\to$ Le but est de généraliser à des entrées inconnues ce qui a pu être *appris* grâce aux données déjà traitées par des experts

- On distingue deux types de problèmes :
  - $Y \subset \mathbb{R}$: problème de régression.
  - $Y = \{1, \ldots, I\}$: problème de classement

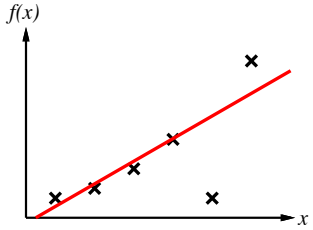
Le problème de la généralisation

- A quelle fonction correspond cet ensemble de points ?



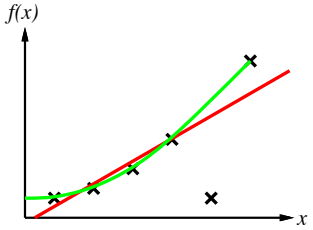
Le problème de la généralisation

- A quelle fonction correspond cet ensemble de points ?



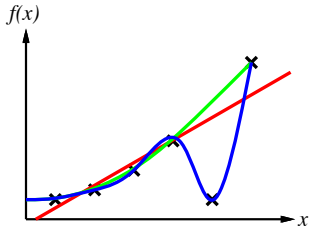
Le problème de la généralisation

- A quelle fonction correspond cet ensemble de points ?



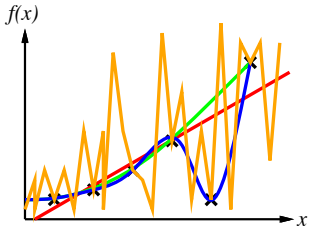
Le problème de la généralisation

- A quelle fonction correspond cet ensemble de points ?



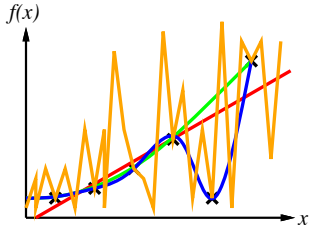
Le problème de la généralisation

- A quelle fonction correspond cet ensemble de points ?



Le problème de la généralisation

- A quelle fonction correspond cet ensemble de points ?



- Le rasoir d'Ockham

  "les hypothèses suffisantes les plus simples sont les plus vraisemblables"

→ Un principe heuristique fondamental en science

Algorithme des K plus proches voisins

- Méthode simple et intuitive
- Principe de l'algorithme :
  - On souhaite classer $x \in X$
  - On dispose de N exemples $(x_n, y_n)_{1 \leq n \leq N}$ dans $X \times Y$

  1 Regarder les classes des K exemples les plus proches

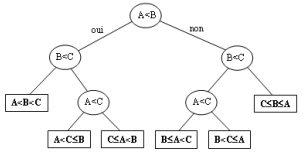
  2 Affecter la classe majoritaire au nouvel exemple
- Il faut choisir une mesure de distance pour trouver les exemples proches
- Il faut choisir K

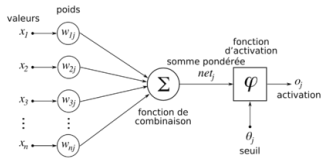
Algorithme des K plus proches voisins

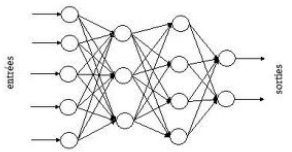
- Comment choisir K ?
  - K grand :
    - Moins sensible au bruit
    - Une grande base d ?apprentissage permet une plus grande valeur de K
  - K petit :
    - Rend mieux compte de structures fines
    - Nécessaire pour des petites bases d ?apprentissage
- Quelle décision prendre en cas d'égalité ?
  - Augmenter la valeur de K de 1 pour trancher. L'ambiguïtée peut persister
  - Tirer au hasard la classe parmi les classes ambigues.
  - Pondération des exemples par leur distance au point x

Méthodes d'apprentissage supervisé

- Les arbres de décision
- Les réseaux de neurones
  - Perceptron à une couche
  - Perceptron multi-couches
  - Réseaux de neurones récurrents
- Les machines à vecteur support
- ...







Apprentissage non-supervisé

Apprentissage non-supervisé

- Aussi appelé classification automatique ou clustering
- Les données d'apprentissage ne sont pas étiquetées
  - Aucun expert n'est requis
  - On parle d'observations plutôt que d'exemples
  - Impossible de calculer un taux d'erreurs pour évaluer une potentielle solution
- Trouver les structures cachées dans les données
  - Classer les données en groupes homogènes
  - Regrouper les données selon leur *similarité*
  - C'est ensuite à l'opérateur d'associer ou déduire du sens pour chaque groupe
  - Le but est de faire ressortir de l'information à partir des données

Apprentissage non-supervisé

- Exemples
  - Dans un ensemble assez large de victimes de cancers du foie tenter de faire émerger des hypothèses explicatives (origines géographique, génétique, habitudes ou pratiques de consommation, expositions à divers agents potentiellement ou effectivement toxiques, etx...)

Algorithme des K-moyennes (K-means)

- Partitionner les observations dans K ensembles $S_1, S_2, \ldots, S_K$ afin de minimiser la distance entre les points à l'intérieur de chaque partition
- Il faut disposer d'une mesure de distance $\|.\|$ sur X
- Il faut choisir K

Algorithme (K-means (K, X, $\{x_n\}_{1 \leq n \leq N}$))

Choisir K points $(\mathbf{m}_1, \ldots, \mathbf{m}_K)$ de X qui représentent la position moyenne des partitions $S_1, S_2, \ldots, S_K$ (par exemlpe au hasard)

répéter

Assigner chaque observation à la partition la plus proche :

$$S_i = \left\{\mathbf{x}_j : \left\|\mathbf{x}_j - \mathbf{m}_i\right\| \leq \left\|\mathbf{x}_j - \mathbf{m}_{i^*}\right\| \text{ pour tout } i^* = 1, \ldots, K\right\}$$

Mettre à jour la moyenne de chaque partition S_i :

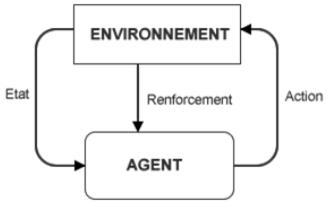
$$\mathbf{m}_i = \frac{1}{|S_i|} \sum_{\mathbf{x}_j \in S_i} \mathbf{x}_j$$

jusqu'à *convergence des $\mathbf{m}_i$ (i.e. aucun changement);*

Apprentissage par renforcement

Apprentissage par renforcement

- Le système apprenant est agent en interaction avec un environnement
  - Le problème est découpé en pas de temps
  - A chaque pas de temps, l'agent
    1. perçoit l'état de l'environnement,
    2. exécute une action dans l'environnement
  - reçoit (éventuellement) un renforcement (une récompense)
- Le but de l'agent est de trouver un comportement qui maximise le renforcement



Apprentissage par renforcement

- Les problèmes résolus par l'agent sont des problèmes de décision séquenteille

Définition

L'apprentissage par renforcement désigne toute méthode adaptative permettant de résoudre un problème de décision séquentielle. (d ?après Sutton et Barto, 1998).

- Le terme "adaptatif" signifie qu'on part d'une solution inefficace, et qu'elle est améliorée progressivement en fonction de l'expérience de l'agent (ou des agents).

→ Apprentisage

Processus de décision de Markov (MDP)

Définition

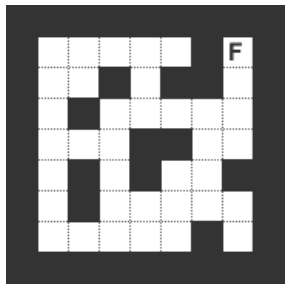
Un processus de décision de Markov (MDP) se définit par un tuple $< S, A, T, R >$:

- S est un ensemble fini d'états possibles.
- A est un ensemble fini d'actions.
- $T : S \times A \rightarrow \Delta(S)$ est une fonction de transition entre états où $\Delta(S)$ est l'ensemble de toutes les distributions de probabilités sur S.
- $R : S \times A \times S \rightarrow \mathbb{R}$ est une fonction de renforcement

- Les processus de décision de Markov forment le cadre formel de l'apprentissage par renforcement

- Ils sont discrets, finis, stochastiques, totalement observables

Exemple



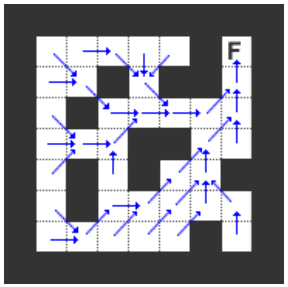
- $S =$ ensemble des cases de la grille
- $A =$ {Haut, Bas, Gauche, Droite, Haut-droite, Haut-gauche, Bas-droite, Bas-gauche}
- $R(s, a) = 1$ si l'action mène l'état F
  $R(s, a) = 0$ sinon
- Les transitions sont déterministes.
- L'état terminal F est un état "puits".

Politique

Définition

Une politique est une fonction qui à un état associe une action :
$\pi : S \to A$

- Une politique définit une stratégie de décision
- Une politique est un plan adapté aux environnements stockastiques



La fonction V : valeurs d'états

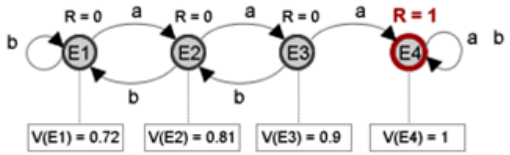
Définition

La fonction $V : S \to \mathbb{R}$, associe à chaque état une valeur définie par l'équation de Bellman pour V :

$$V(s) = \max_{a \in A} \{R(s, a) + \gamma \sum_{s' \in S} T(s, a)(s').V(s')\}$$

avec $\gamma \in [0, 1]$ un paramètre appelé facteur amortissement (discount factor).

- $V(s) =$ gain immédiat $+$ une proportion γ des gains des états futurs lorsqu'on exécute la *meilleure* des actions.



La fonction Q : valeurs d'actions

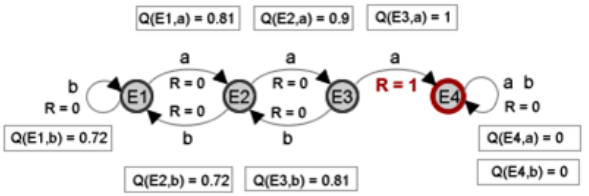
Définition

La fonction $Q : S \times A \to \mathbb{R}$, associe à chaque couple état-action une valeur définie par l'équation de Bellman pour Q :

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

avec $\gamma \in [0, 1]$ un paramètre appelé facteur amortissement (discount factor).

- $Q(s, a) =$ gain immédiat après l'exécution de a $+$ une proportion γ des gains des états futurs



Des valeurs et une politique

- Valeur d'état en fonction des valeurs d'actions :

$$V(s) = \max_{a \in A} Q(s,a)$$

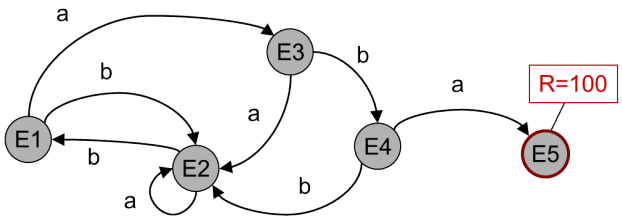
- Valeur d'action en fonction des valeurs d'état :

$$Q(s,a) = \sum_{s' \in S} T(s,a)(s').V(s")$$

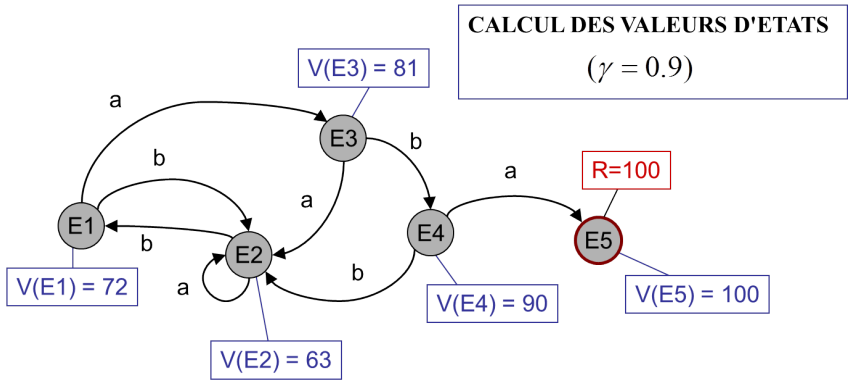
- Calcul d'une politique à partir des fonctions valeurs :

$$\begin{aligned} \pi(s) &= argmax_{a \in A} Q(s,a) \\ &= argmax_{a \in A} \sum_{s' \in S} T(s,a)(s').V(s") \end{aligned}$$

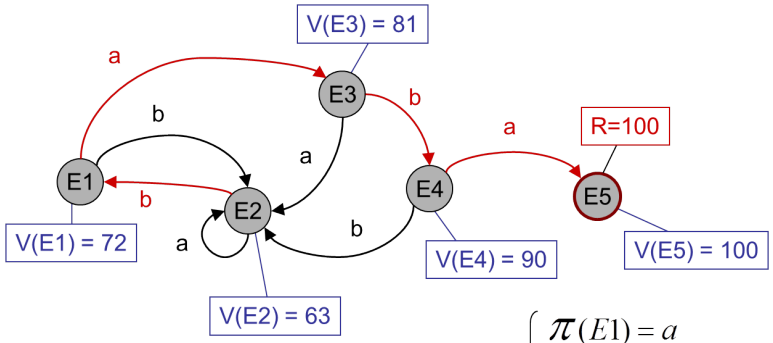
Des valeurs et une politique



Des valeurs et une politique



Des valeurs et une politique



STRATEGIE OPTIMALE

$$\begin{cases} \pi(E1) = a \\ \pi(E2) = b \\ \pi(E3) = b \\ \pi(E4) = a \end{cases}$$

Programmation dynamique

- Considérons un processus de décision de Markov M. Comment faire pour calculer les fonctions V et Q correspondant à M ?

→ La programmation dynamique

- Le domaine de la programmation dynamique propose de nombreux algorithmes d'optimisation adaptés aux MDP
  - Algorithme des valeurs itérées
  - Algorithme des politiques itérées
  - ...

Algorithme des valeurs itérées (Bellman 1957)

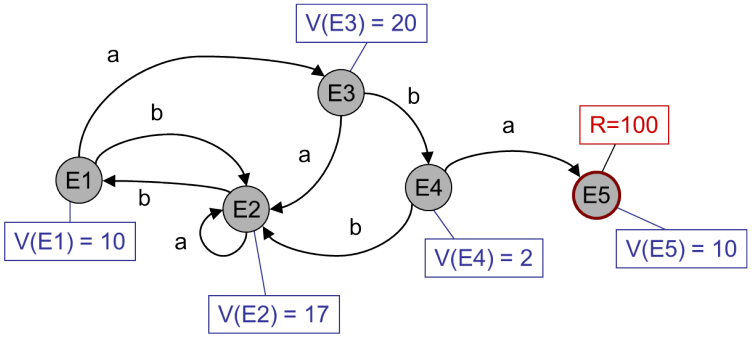
Algorithme (ValueIteration (S, A, T, R, γ))

Soit $V^{(0)}$ *une fonction qui associe des valeurs arbitraires aux états*
Soit $\pi^{(0)}$ *la politique correspondant* $V^{(0)}$
Soit t *un compteur d'itération*
$t \leftarrow 1$
répéter
  pour chaque $s \in S$ **faire**
    $V^{(t)}(s) \leftarrow \{R(s, \pi^{(t-1)}(s)) + \gamma \sum_{s' \in S} T^{(t-1)}(s, \pi^{(t-1)}(s))(s').V(s')\}$
  $t \leftarrow t + 1$
jusqu'à *convergence;*

- Les valeurs d'état sont calculées itérativement :
- Les itérations sont effectuées jusqu'à ce que la variation apportée soit inférieure à un seuil

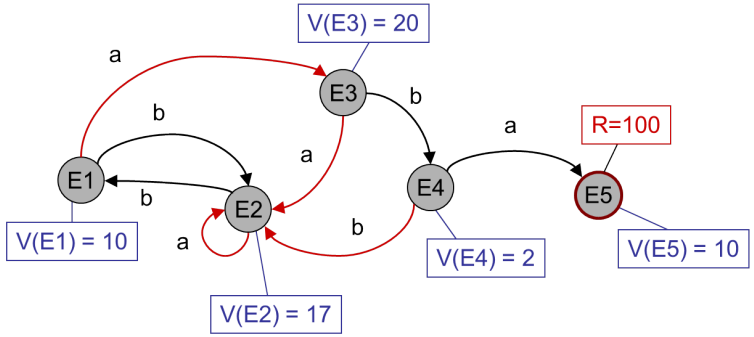
Algorithme des valeurs itérées

Itération 0 : initialisation des valeurs aléatoirement



Algorithme des valeurs itérées

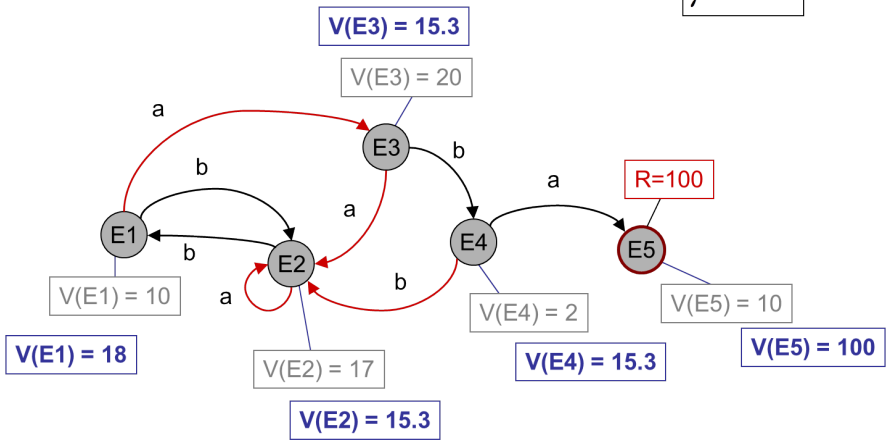
Itération 0 : détermination de la politique



Algorithme des valeurs itérées

Itération 1 : calcul des valeurs

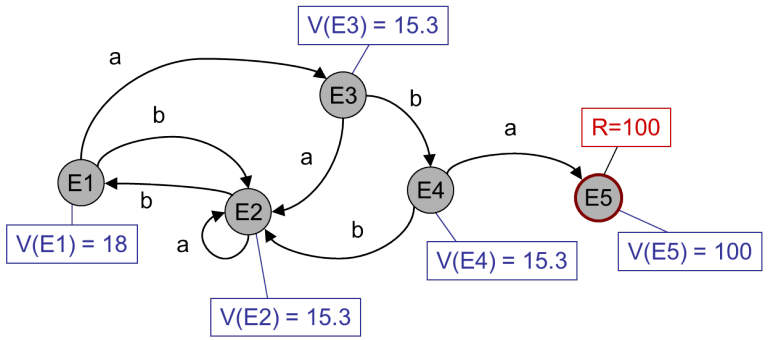
$\gamma = 0.9$



Algorithme des valeurs itérées

Itération 1 : calcul des valeurs

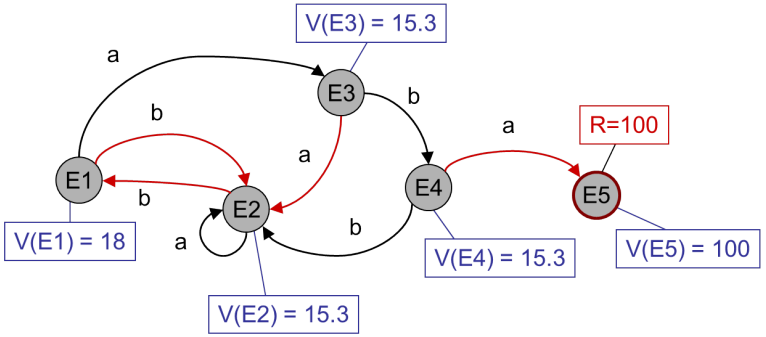
$\gamma = 0.9$



Algorithme des valeurs itérées

Itération 1 : détermination de la politique

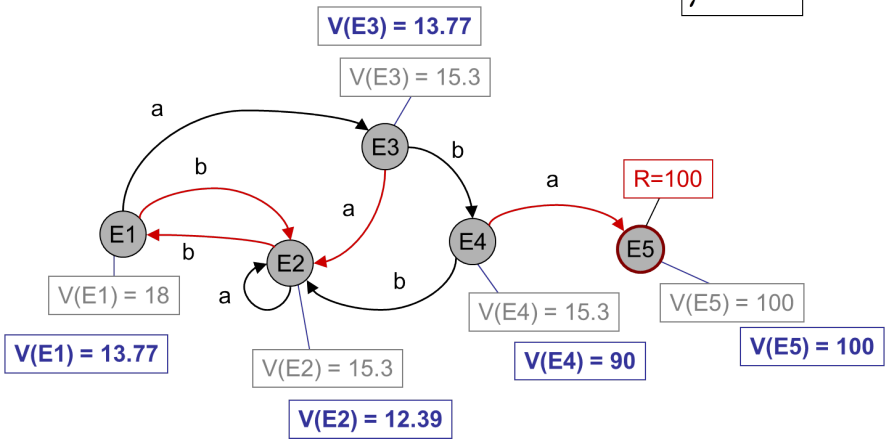
$\gamma = 0.9$



Algorithme des valeurs itérées

Itération 2 : calcul des valeurs

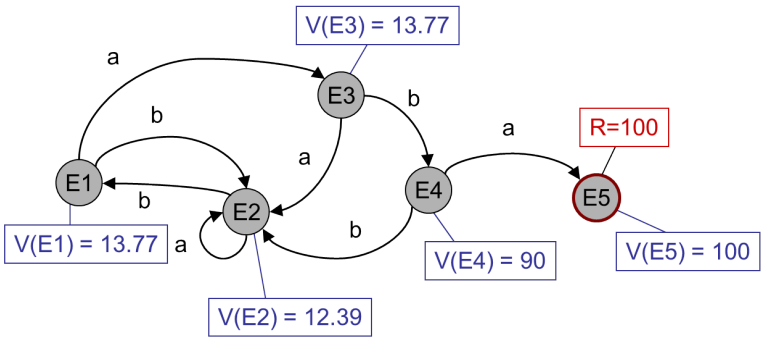
$\gamma = 0.9$



Algorithme des valeurs itérées

Itération 2 : calcul des valeurs

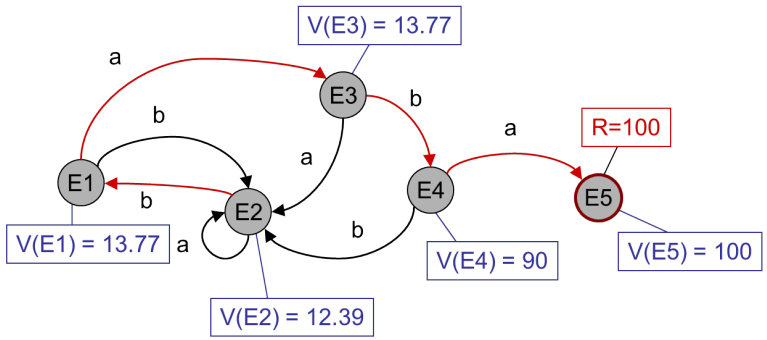
$$\gamma = 0.9$$



Algorithme des valeurs itérées

Itération 2 : détermination de la politique

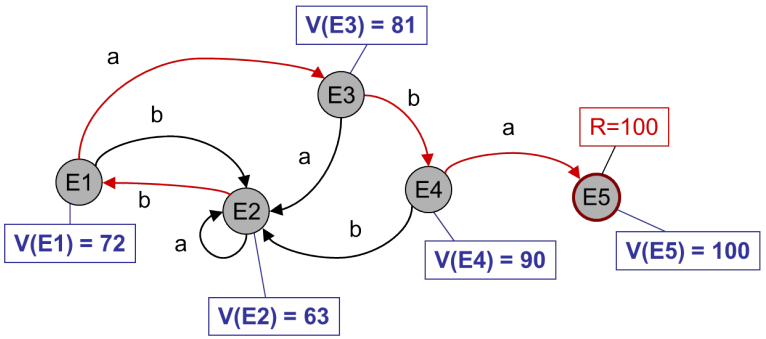
$$\gamma = 0.9$$



Algorithme des valeurs itérées

Itération 5 :

$$\gamma = 0.9$$



Environnement inconnu ?

- La programmation dynamique s'applique à un MDP parfaitement connu
- Que faire si l'environnement est à priori inconnu ?
  Que faire si l'on considère un MDP dont les fonctions T et R inconnus ?

→ Apprentissage par renforcemennt

- L'agent doit explorer l'environnement en plus d'optimiser son comportement
- Approximer V ou Q durant l'interaction avec l'environnement
- C'est un apprentissage online : le système à besoin d'interagir avec l'environnement pendant l'apprentissage.

Apprentissage par renforcement

- Faut-il modéliser explicitement le MDP sous-jacent de l'environnement ?
- Méthodes d'apprentissage indirect
  - Modélisation explicite du MDP
  - Approximation de T et de R
  - Calcul de V et/ou Q à partir de cette approximation
  - Ex : Certainty equivalence, Dyna, Prioritized Sweeping, ...
- Méthodes d'apprentissage direct
  - Pas de modélisation du MDP
  - Approximation de Q directement pendant l'interaction
  - Ex : Q-learning, Sarsa, Systèmes de Classeurs, ...

Algorithme Dyna (Sutton 1991)

- Méthode d'apprentissage par renforcement indirect
- Principe de l'algorithme :
  - Initialisation :
    - La fonction T est initialisée avec des probablilités aléatoires
    - La fonction R retourne une valeur nulle
  - A chaque iteration, l'agent intéragit avec l'environnement :
    - L'agent est dans un état s
    - L'agent choisit une action a et l'exéute dans l'environnement
    - L'agent perçoit l'état résultant t et un renforcement r

  $\rightarrow$ A chaque iteration, l'apprentissage s'effectue à partir du tuple $< s, a, t, r >$
  - Les fonctions T et R sont mises à jour à partir de $< s, a, t, r >$
  - La valeur $Q(s, a)$ est mise à jour à partir de T et R
  - K couples état-action sont choisis aléatoirement et leurs valeurs Q sont mises à jour
  - Les iterations sont répétées jusqu'à *convergence* des fonctions et des valeurs

Algorithme Q-learning (Watkins 1992)

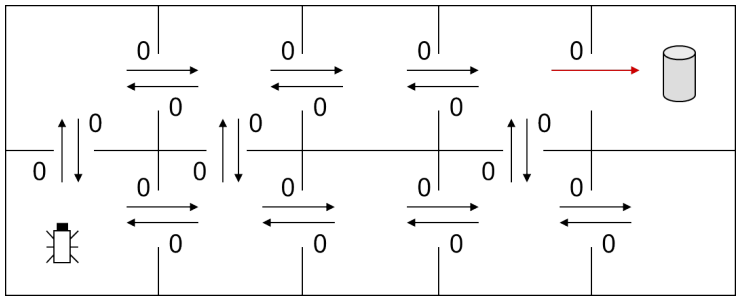
- Méthode d'apprentissage par renforcement direct
  - Les fonctions T et R ne sont pas calculées
- Principe de l'algorithme :
  - Initialisation :
    - La fonction Q retourne une valeur nulle
  - A chaque iteration, l'agent intéragit avec l'environnement :
    - L'agent est dans un état s
    - L'agent choisit une action a et l'exéute dans l'environnement
    - L'agent perçoit l'état résultant t et un renforcement r
  - A chaque iteration, la fonction Q est mise à jour à partir de $< s, a, t, r >$

    $$Q(s,a) \leftarrow Q(s,a) + \alpha\Big[r + \gamma \max_{b \in A} Q(t,b) - Q(s,a)\Big]$$

    où $\alpha \in [0, 1]$ est un taux d'apprentissage
  - Les iterations sont répétées jusqu'à *convergence* des valeurs

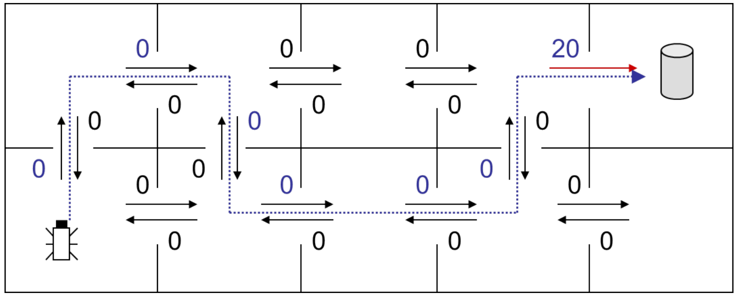
Q-learning

- Recompense de 100 à l'objectif (flèche rouge)
- Facteur d'amortissement : $\gamma = 0.9$
- Taux d'apprentissage : $\alpha = 0.2$



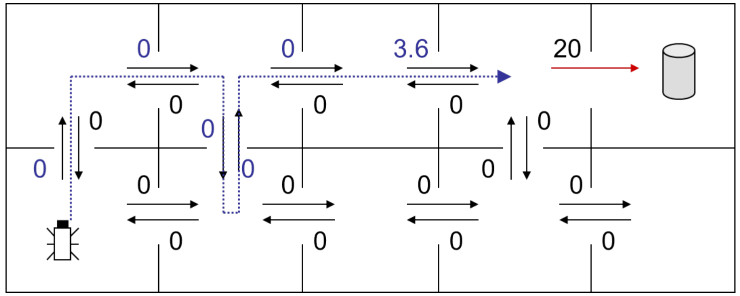
Q-learning

- Recompense de 100 à l'objectif (flèche rouge)
- Facteur d'amortissement : $\gamma = 0.9$
- Taux d'apprentissage : $\alpha = 0.2$



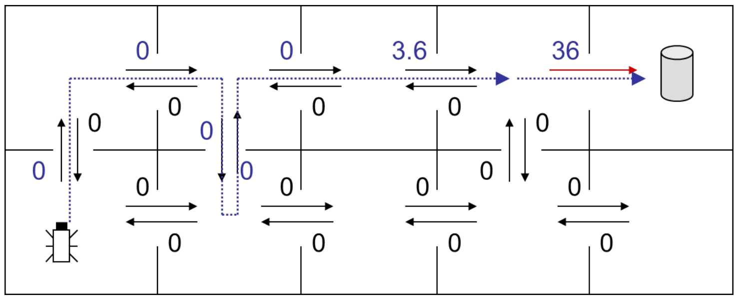
Q-learning

- Recompense de 100 à l'objectif (flèche rouge)
- Facteur d'amortissement : $\gamma = 0.9$
- Taux d'apprentissage : $\alpha = 0.2$



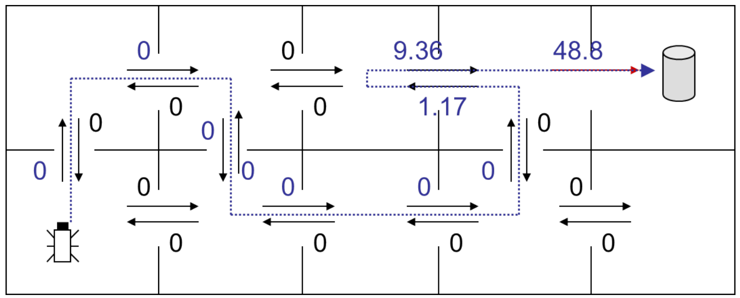
Q-learning

- Recompense de 100 à l'objectif (flèche rouge)
- Facteur d'amortissement : $\gamma = 0.9$
- Taux d'apprentissage : $\alpha = 0.2$



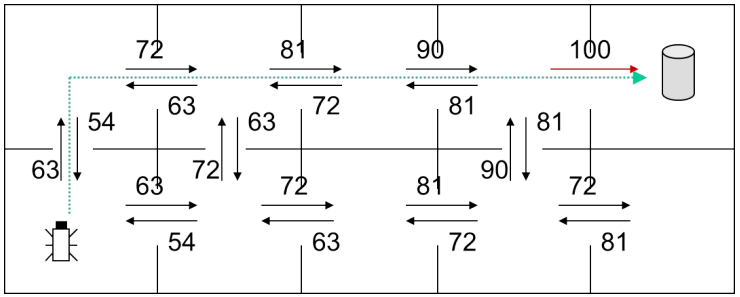
Q-learning

- Recompense de 100 à l'objectif (flèche rouge)
- Facteur d'amortissement : $\gamma = 0.9$
- Taux d'apprentissage : $\alpha = 0.2$



Q-learning

- Recompense de 100 à l'objectif (flèche rouge)
- Facteur d'amortissement : $\gamma = 0.9$
- Taux d'apprentissage : $\alpha = 0.2$



Le dilemme exploration-exploitation

- Comment choisir l'action à exécuter à chaque itération ?
- Si l'agent choisit toujours l'action qui maximise Q :
  - Il va avoir tendance à toujours prendre le même chemin
  - Il n'explorera pas les autres possibilités qui sont peut-être meilleures
- Pour apprendre, il est nécessaire d'explorer l'environnement
  - Il faut tester les actions dans les différents états
  - Il faut utiliiser une autre politique que celle issue de Q
- Mais à quel moment faut-il cesser d'explorer ?

→ Dilemme entre explorer l'environnement et exploiter ses connaissances

Méthodes de choix de l'action

- Méthode gloutonne (greedy)
  - L'action sélectionnée est toujours celle de plus forte valeur

    $$action = argmax_{a \in A} Q(s, a)$$
  - Méthodes d'exploitation "pure"
- Méthode ϵ-gloutonne (ϵ-greedy)
  - Avec une probabilité ϵ, l'action est sélectionnée aléatoirement
  - Sinon l'action de valeur maximale est sélectionnée
- Méthode Softmax
  - La probabilité de sélection d'une action est proportionnelle à sa valeur
  - Distribution de Boltzmann :

    $$p(a|s) = \frac{e^{\frac{1}{T}Q(s,a)}}{\sum_b e^{\frac{1}{T}Q(s,b)}}$$

Approximation et généralisation

- Les algorithmes classiques stocker les valeurs d'états et d'actions dans une table
- Cette méthode peut fonctionner avec 10 000 états, mais pas pour des problèmes plus complexes
  - Le backgammon contient 10^{50} états
  - Les échecs contiennent 10^{120} états
  - Il serait absurde de vouloir visiter tous ces états pour pouvoir apprendre à jouer.

→ Utiliser une fonction d'approximation des fonctions valeurs

- Elle permet de calculer une estimation des valeurs
- Par exemple : appliquer une régression sur les valeurs avec un réseau de neurones multi-couches

  → Problème d'apprentissage supervisé

Apprentissage par renforcement : extensions

- Environnements continus
  - Nombre d'états infinis
- Gestion du temps
  - Calcul de politique en temps réel
  - Pris en compte de la durée des actions
- Environnement partiellement observables
  - Processus de décision de Markov *partiellement observables*
- Dilemme exploration-exploitation
  - Limiter la complexité en échantillons (*sample complexity*)
- Hiérarchies de comportements
  - Modèles d'*options* au lieu de modèles d'actions

Conclusion

Quelques applications notables

- TD-Gammon (Esauro, 1992) : joueur de Backgammon
  - Apprentisage direct approximé par un perceptron multi-couches
  - L'algorithme a appris en jouant 1 500 000 parties contre lui-même

  → Programme champion du monde
- Contrôle d'ascenseurs (Crites & Barto, 1996)
  - Plusieurs agents apprenants
  - Etats continus
  - Gestion du temps

  → Surpasse les meilleurs algorithmes de gestion des acsenseurs
- Contrôle d'un hélicopère (Ng, 2000)
  - Apprendre à effectuer la manoeuvre très diffcile "nose-in-circle"
  - PEGASUS policy search algorithm

  → Performances meilleures que les pilotes experts

Conclusion

- L'apprentissage automatique regroupe les techniques permettant à une machine d'adapter et d'améliorer ses performances par l'expérience
- Trois types d'apprentissage tous utiles en robotique
  - Apprentissage supervisé
    - Reconnaissance de formes, des couleurs, des sons
    - Modélisation et prédiction des comportements des autres
    - ...
  - Apprentissage non-supervisé
    - Découverte d'information dans les observations
    - Structuration des connaissances, généralisation
    - ...
  - Apprentissage par renforcement
    - Apprentissage de mouvements
    - Apprentissage de comportements
    - Amélioration de stratégies de décision
    - Adaptation au dynamisme environnemental (évènements imprévus)
    - ...