FoSyMa Cours 5
Apprentissage dans les Systèmes Multiagents

Aurélie Beynier

FoSyMa, Master 1 ANDROIDE

26 février 2019

Plan

Plan

Apprentissage et IA

Définition

L'apprentissage consiste à améliorer ses comportements futurs à partir de ses expériences passées.

L'apprentissage consiste en l'acquisition de nouvelles connaissances, de nouvelles capacités d'actions et de nouvelles capacités cognitives, et en l'intégration des connaissances et compétences acquises dans les futures activités du système conduisant ainsi à améliorer les performances du système.[SW99]

Apprentissage et IA

Problématiques

- Quelle composante doit être apprise ? Fonction état $\rightarrow$ actions, dynamique du monde, fonction d'utilité, fonction action-valeur, buts ?
- Quelles sont les données disponibles pour réaliser l'apprentissage ? Exemples, récompenses ?
- Quelle représentation doit prendre l'information apprise ? Fonction d'utilité, propositions logiques, réseaux bayésiens ?

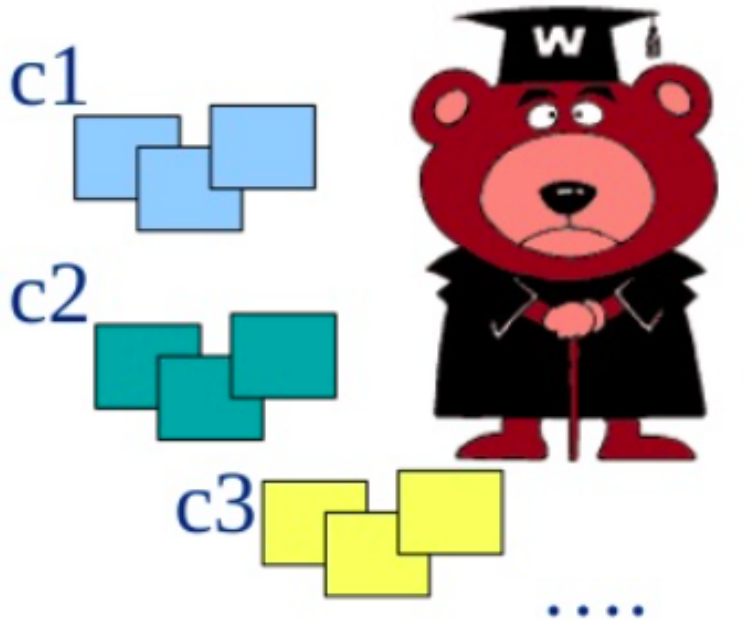
Apprentissage et IA

Données disponibles pour apprendre

- Apprentissage d'une fonction à partir d'exemples d'entrées et de sorties → **apprentissage supervisé**
- Modèles en entrée et aucune information en sortie → **apprentissage non-supervisé**
- Récompense en réponse à un comportement → **apprentissage par renforcement**

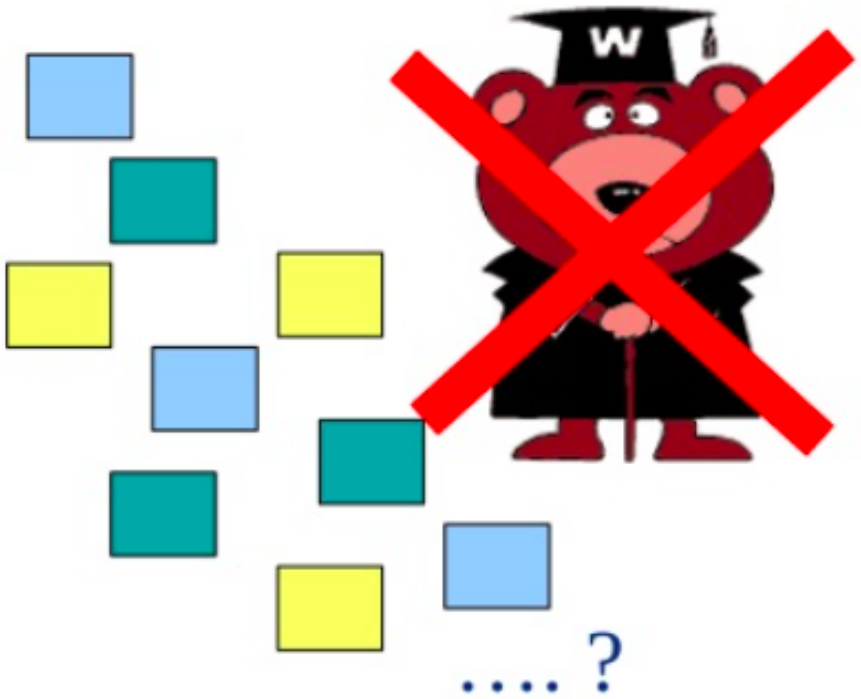
Apprentissage supervisé

Apprentissage guidé par un expert ou un « enseignant »
On connaît la sortie correspondant aux entrées
Objectif : utiliser les connaissances pour résoudre de nouveaux problèmes
Méthodes : arbres de décision, réseaux de neurones,...



Apprentissage non-supervisé

Pas d'expert, pas de connaissance des sorties
Objectif : identifier des groupes ou des modèles
Méthodes : « clustering », analyses en composantes principales,...



Plan

Apprentissage et Agents

Motivations

- Permettre aux agents de faire face à des situations non-prévues : nécessaire dans des environnements dynamiques, incertains, inconnus, non parfaitement modélisables.
- Permettre aux agents d'améliorer leurs comportements au fur et à mesure des expériences

Apprentissage dans un SMA

Problématiques liées au multiagent

- Comment tenir compte des interactions entre agents ?
- Comment mesurer / évaluer l'influence d'un agent dans le résultat de l'action jointe ?
- Comment différencier les changements de l'environnement dus à sa stochasticité et ceux dus aux autres agents ?
- Comment garantir la convergence de l'apprentissage vers le comportement souhaité ?

Apprentissage dans un SMA

Complexité

- Le nombre d'états possible est exponentiel en le nombre d'agents et de caractéristiques de l'environnement
- Le nombre d'actions jointes est exponentiel en le nombre d'agents
- Le nombre d'issues possibles pour une action jointe est exponentiel en le nombre d'agents.

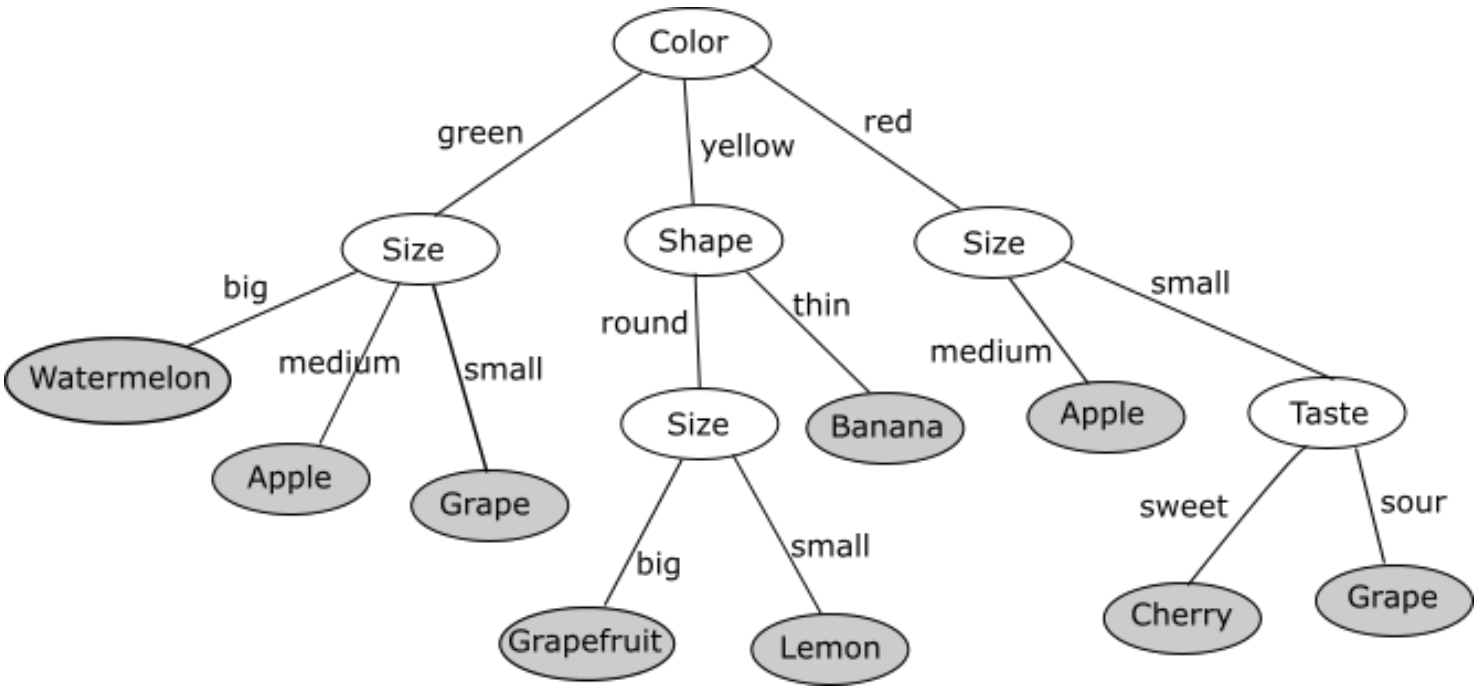
Plan

Apprendre des arbres de décision

- Apprentissage supervisé de classification
- Déterminer la classe d'un objet en fonction de certaines de ses caractéristiques
- Chaque nœud correspond à une caractéristique discriminante et chaque arc vers un fils correspond à une valeur pour cette caractéristique.
- Les feuilles de l'arbre sont des classes.
- Chaque classe peut être associée à une décision : l'agent décide comment agir en fonction des valeurs associées à des attributs.

Exemple d'arbre de décision

- Chaque nœud correspond à un test sur les valeurs d'un attribut.
- Les branches sortant du nœud correspondent aux différentes valeurs de ce test.



Exemples de données pour construire l'arbre

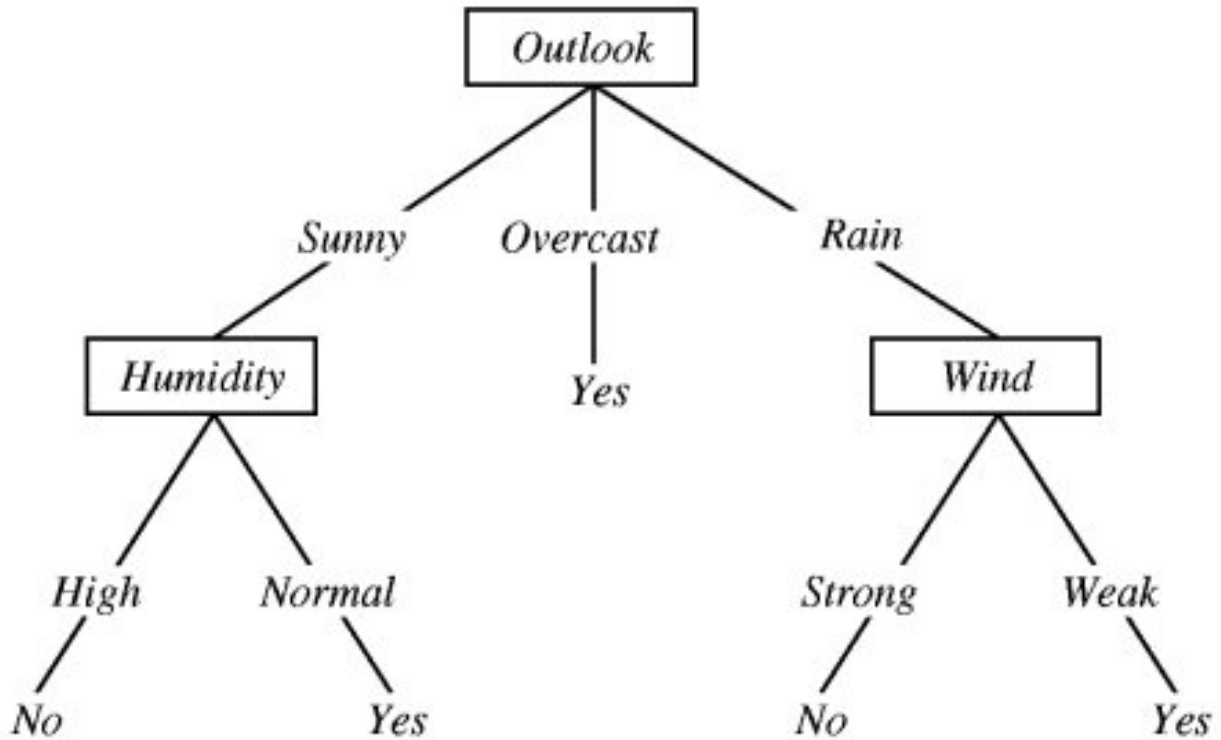
[WFH11]

Table 1.2 The weather data.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Exemple d'arbre de décision

Play Tennis ?



Représentation par un arbre de décision

- Soit un ensemble d'exemples associant à une liste d'attributs/valeurs, une classe
- On souhaite trouver un arbre de décision qui permette de représenter cet ensemble d'exemples.
- Approche Triviale :
  - Construire un arbre de décision en construisant un chemin de la racine à une feuille pour chaque exemple.
  - Limitation : revient à représenter un ensemble d'observations mais ne permet pas d'extrapoler à des exemples inconnus
- Dans l'idéal : trouver l'arbre de décision minimal $\rightarrow$ potentiellement insolvable (d'un point de vue complexité algorithmique).

Apprentissage d'un arbre de décision

- Idée : tester les propriétés les plus discriminantes en premier
- Dans le cas d'une classification en 2 classes `Vrai` / `Faux`, le meilleur attribut est celui qui permettrait de séparer tous les éléments de la classe `Vrai` de tous ceux de la classe `Faux`. Un "mauvais" attribut conserve les mêmes proportions d'exemples positifs et négatifs.
- Mesure de la valeur d'un attribut - **Entropie** - : mesurer l'**information** fournie par un attribut $\rightarrow$ théorie de l'information [SW48]

Entropie

- Pour une catégorisation binaire :

$$Entropie(S) = -p_+ log_2(p_+) - p_- log_2(p_-)$$

où S est l'ensemble des exemples, p_+ est la proportion d'exemples positifs et p_- est la proportion d'exemples négatifs.

- Généralisation à n catégories :

$$Entropie(S) = \sum_{i=1}^{n} -p_i log_2(p_i)$$

où p_i est la proportion d'exemples de la catégorie i.

Entropie

Exemples :

- Attribut A permettant de discriminer 15 exemples + de 5 exemples - :

$$Entropie(A) = -15/20 log_2(15/20) - 5/20 log_2(5/20) =$$

$$0,311 + 0,5 = 0,811$$

- Attribut A permettant de discriminer 20 exemples + de 0 exemples - :

$$Entropie(A) = -20/20 log_2(20/20) - 0/20 log_2(0/20) = 0$$

Entropie

Pour décider par quel attribut on va étiqueter un nœud, il faut calculer le gain de chaque nœud :

$$Gain(A_i) = Entropie(A_i) - EP(A_i)$$

où :

- $Entropie(A_i)$ correspond à l'entropie de l'ensemble des exemples du nœud.
- $EP(A_i)$ correspond à l'entropie pondérée du sous-arbre résultant du développement selon A_i.

Entropie pondérée

Si le nœud courant contient n exemples et que l'attribut A_i permet d'obtenir m nœuds alors l'entropie pondérée associée à A_i est définie par :

$$EP(A_i) = \sum_{j=1}^{m} \frac{n_j}{n} Entropie([n_j^+, n_j^-])$$

où $n_j = n_j^+ + n_j^-$

Algorithme générique de construction de l'arbre

Data: Exemples E, ensemble d'attributs A, classe C
initialiser à l'arbre vide ; // la racine est le nœud courant
repeat
  décider si le nœud courant est terminal ;
  if *le nœud est terminal* **then**
    affecter une classe ;
  else
    sélectionner un attribut à tester et créer les sous-arbres ;
  end
until *Jusqu'à obtenir un arbre de décision*;

Algorithme de construction de l'arbre : ID3 [Qui79]

Data: Exemples E, ensemble d'attributs A, classe C
Créer nœud N
if *tous les exemples de E sont de la même classe C* **then**
  Retourner N comme une feuille étiquetée par C ;
end
if *la liste des attributs A est vide* **then**
  Retourner N Comme une feuille étiquetée de la classe majoritaire dans E ;
end
otherwise
  Sélectionner l'attribut a qui classifie le mieux les exemples E ;
  Etiqueter N par l'attribut sélectionné
  for *chaque valeur possible v_i de a* **do**
    Ajouter une nouvelle branche sous N :
    Attacher à N le sous arbre généré par ID3(Exemples(v_i), $A - \{a\}$, C)
  end
end

Algorithme de construction de l'arbre : ID3

- Construction récursive de l'arbre
- Algorithme glouton non-optimal
- Calcule des arbres de profondeur faible mais pas obligatoirement ayant la plus petite profondeur.
- Pas de backtracking pour améliorer l'arbre

Plan

Apprentissage par renforcement

Principe

- Apprentissage par essai-erreur.
- Inspiré de l'apprentissage chez les animaux : par exemple, dressage d'un animal.
- Les actions souhaitées sont récompensées, les actions non-souhaitées sont pénalisées.
- Les récompenses et pénalités sont un ≪ feedback ≫ de l'environnement

Apprentissage par renforcement

[SB98]

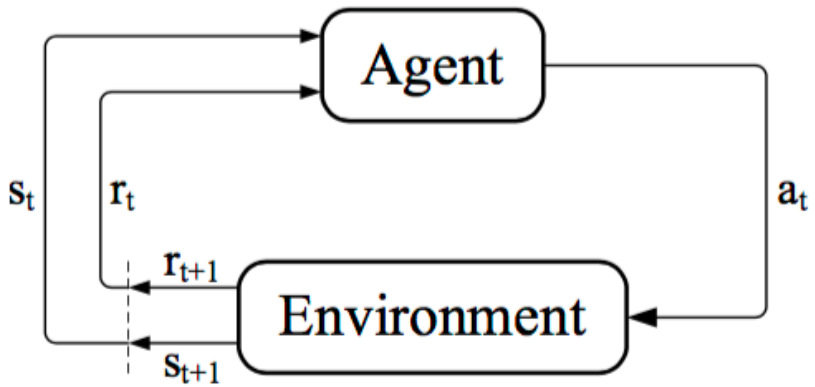


FIGURE: Extrait de Bloembergen et al.

Apprentissage pour 1 agent / 1 état

- Considérons le cas d'un agent devant apprendre quelle action effectuer en 1 état
- Choix entre n actions : « bandits manchots » à n bras
- L'agent souhaite maximiser ses récompenses à long terme en apprenant la fonction de valeur de chaque action (i.e. bras).
- A chaque pas de temps, l'agent sélectionne une action et maintient une estimation de la récompense moyenne de chaque action [RN03] :

$$Q_t(a_i) = \frac{r_1 + r_2 ... + r_k}{k}$$

- Extension au cadre multiagent (interactions limitées) : les agents apprennent de manière indépendante.

Apprentissage de politiques

- Idée : apprendre directement la politique plutôt qu'une fonction de valeur
- Contexte : plusieurs agents jouent de façon répétée le même jeu.
- Chaque agent maintient une distribution de probabilités sur les actions à jouer.
- Cette distribution est mise à jour après chaque exécution d'action.

Apprentissage de politiques

- Soit $p(t) = (p_1(t), p_2(t), ...p_s(t))$ le vecteur de probabilités associant une probabilité de sélection à chacune des s actions d'un agent
- Dans le cas à deux agents, soit r_{ij}^1 la récompense obtenue par le premier agent lorsqu'il joue a_i et que l'autre agent joue a_j.
- Si l'agent sélectionne a_i :

$$p_i(t+1) = r_{ij}^1 + (1 - r_{ij}^1)p_i(t)$$

- et toutes les autres actions $a_{i'} \neq a_i$:

$$p_{i'}(t+1) = (1 - r_{ij}^1)p_{i'}(t)$$

- Solution pouvant être non-optimale

Apprentissage par renforcement

Limitations des deux approches précédentes

- Pas d'états multiples
- Pas de prise en compte des récompenses à long terme : une mauvaise récompense immédiate peut permettre par la suite d'obtenir une très forte récompense

Apprentissage par renforcement

Algorithme du Q-learning

- Apprentissage par renforcement dans le cadre des Processus Décisionnels de Markov
- Permet de tenir compte de l'incertitude sur les résultats des actions, de modéliser plusieurs états possibles et considérer les effets des actions à long-terme.
- Ne nécessite pas la connaissance des fonctions de transition et de récompense
- Apprentissage d'une table des Q-valeurs associant une valeur à chaque action

Apprentissage par renforcement

Rappel sur les MDPs (cf. cours précédents) :

- Un MDP est défini par un tuple $\langle S, A, T, R \rangle$
  - S : l'ensemble des états s du système
  - A : l'ensemble des actions a de l'agent
  - T : la fonction de transition modélisant l'incertitude, $T(s'|s, a)$ est la probabilité de passer d'un état s à un état s' en effectuant l'action a
  - R : la fonction de récompense modélisant les objectifs de l'agent, $R(s, a)$ est la récompense obtenue par l'agent lorsque l'action a est exécutée partir de l'état s.

Apprentissage par renforcement

Dans le cadre d'un MDP, la valeur d'une politique π est donnée par l'équation de Bellman :

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') \times V^{\pi}(s') \qquad (1)$$

où $\gamma \in [0, 1[$ est un facteur d'actualisation
La politique optimale π^* est la politique ayant la valeur maximale :

$$V^{\pi^*}(s) = max_{\pi} V^{\pi}(s) \quad \forall s \in S \qquad (2)$$

Apprentissage par renforcement

- Lorsque les fonctions de transition et de récompense ne sont pas connues, l'agent doit utiliser ses interactions avec l'environnement afin d'apprendre la meilleure politique en-ligne.
- Pas d'apprentissage du modèle, on apprend directement la politique
- L'agent maintient une table de valeur associant à chaque couple (état, action) une valeur réelle. $\rightarrow$ table des Q-valeurs
- Cette table est mise à jour à chaque instant t à partir de la récompense (ou pénalité) obtenue par l'agent par différence temporelle
- Différents algorithmes : TD learning, SARSA, Q-learning [SB98]

Apprentissage par renforcement

- L'algorithme Q-learning procède à la mise à jour suivante :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t [r_{t+1} + \gamma max_a (Q(s_{t+1}, a) - Q_t(s_t, a_t)] \quad (3)$$

  où α désigne le taux d'apprentissage ($\alpha \in [0, 1]$) et α_t converge vers 0 au fur et à mesure de l'apprentissage
- Pour réaliser la mise à jour, il est nécessaire d'avoir connaissance du tuple $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$

Apprentissage par renforcement

Q-learning :

Initialiser Q_0
for t=0 **to** T-1 **do**
$s_t \leftarrow$ Choisir état
$a_t \leftarrow$ Choisir action
$(s_{t+1}, r_{t+1}) \leftarrow$ Simuler action a_t à partir de s_t
Mettre à jour $Q(s_t, a_t)$
end for

Apprentissage par renforcement

Dans un environnement stochastique, il est nécessaire d'essayer plusieurs fois chaque action à partir de chaque état afin d'avoir une bonne estimation de l'utilité espérée de chaque action $\rightarrow$ nécessité de répéter l'apprentissage
Problématique : comment choisir l'action à exécuter dans un état donné ?

- **Exploitation** : consiste à exploiter les connaissance acquises à partir des expériences passées et à choisir l'action qui s'est avérée la plus bénéfique par le passé (étant donné l'état courant) $\rightarrow$ utilisation de la table des Q-valeurs
- **Exploration** : consiste à essayer de nouvelles actions afin de chercher pour une action plus performante que celles déjà expérimentées $\rightarrow$ choix aléatoire

Apprentissage par renforcement

Comment choisir de façon efficace entre exploitation et exploration ?

- Exploiter pendant N_1 étapes puis explorer pendant N_2 étapes
- A chaque étape, exploiter avec une probabilité de $1 - \epsilon$ ou explorer avec une probabilité de ϵ avec $\epsilon \in [1, 0]$
- Calculer la probabilité de chaque action en utilisant uen distribution de Boltzman :

$$P_T(a) = \frac{exp(-\frac{Q_n(s_n,a)}{T}}{\sum_{a'} exp(-\frac{Q_n(s_n,a')}{T}} \quad (4)$$

Apprentissage par renforcement

Dans un cadre mono-agent, il est prouvé que l'apprentissage par renforcement converge vers la politique optimale si :

- un nombre suffisant d'étapes de mise à jour sont réalisées,
- toutes les actions peuvent être sélectionnées avec une probabilité supérieure à 0,
- le taux d'apprentissage est décroissant et tend vers 0.

Une hypothèse sous-jacente au modèle est que l'environnement évolue de façon markovienne : les probabilité de transition vers le prochain état ne dépendent que de l'état et d'action courante.

Apprentissage par renforcement multiagent

- L'application de l'apprentissage par renforcement au cadre multiagent (MARL) soulève de nombreuses difficultés.
- Comme les agents interagissent, la récompense obtenue par un agent dépend des actions des autres agents. Ces agents apprennent eux aussi, leurs comportements ne sont donc pas stationnaires. $\Rightarrow$ l'hypothèse de Markov n'est plus vérifiée $\Rightarrow$ l'optimalité n'est plus garantie.
- 2 approches principales :
  - apprentissage indépendant : chaque agent n'observe que ses propres actions
  - apprentissage joint : chaque agent observe les actions de tous les agents

Plan

Intelligence Collective

Principes

- Technique d'apprentissage bio-inspirée (fourmis, abeilles, oiseaux,...)
- Un grand ensemble d'agents aux capacités cognitives limitées (agents réactifs)
- A partir des interactions locales entre ces agents, des comportements collectifs de plus haut niveau peuvent émerger.

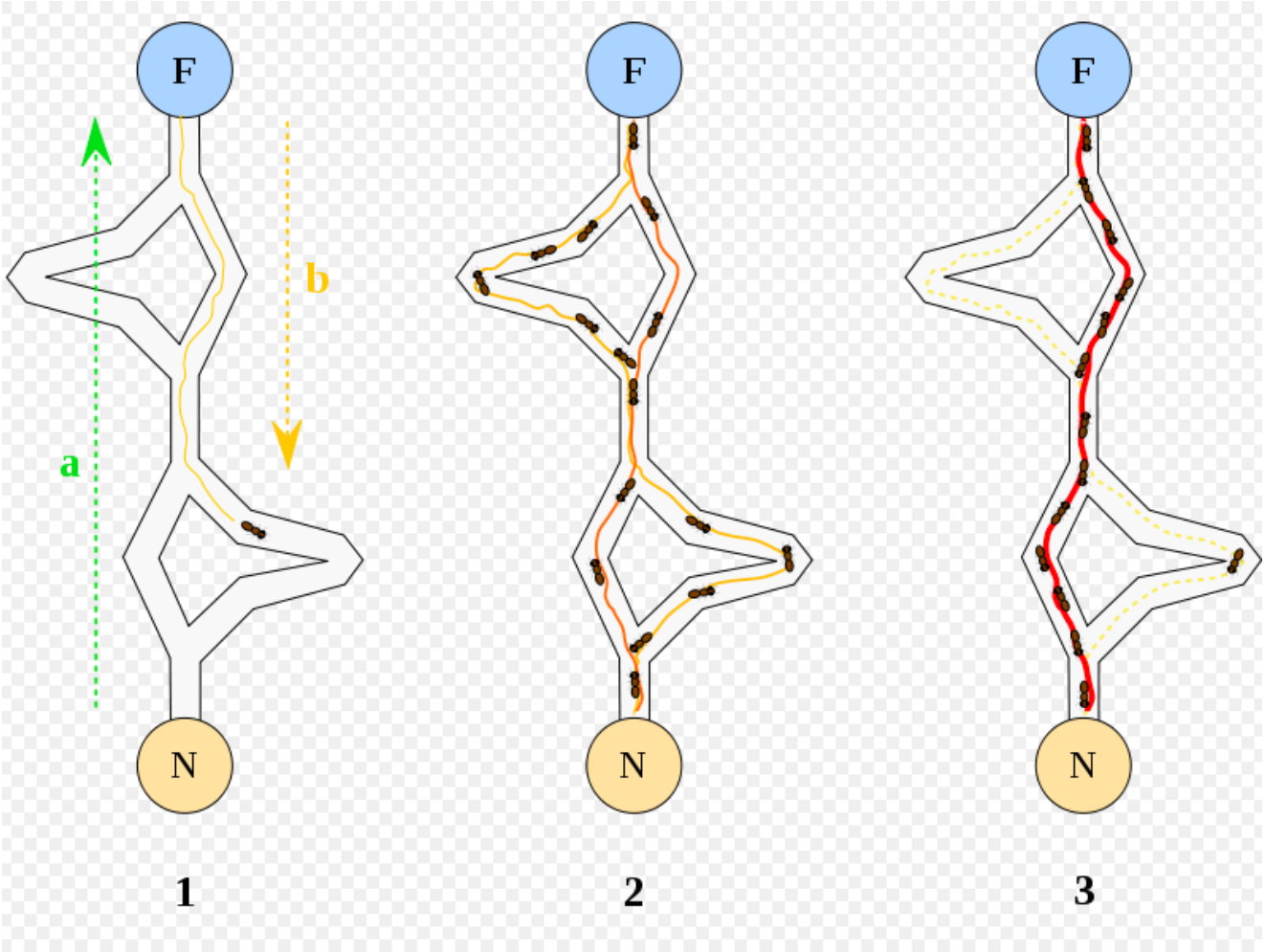
Colonies de fourmis

- Résolution de problème d'optimisation (planification de chemins par exemple).
- Approche basée sur les phéromones (modification de l'environnement, communication indirecte).
- Chaque fourmi dépose des phéromones sur le chemin qu'elle emprunte.
- Les phéromones s'évaporent au fil du temps.
- La probabilité d'emprunter un chemin est proportionnelle à la quantité de phéromones sur ce chemin.

Colonies de fourmis

- Au départ, les fourmis choisissent un chemin aléatoirement.
- Quand une fourmi découvre de la nourriture, elle retourne au nid.
- Dans un même intervalle de temps, plus de fourmis passent par le chemin le plus court et renforcent donc la trace de phéromone sur ce chemin.
- Les fourmis finissent par toutes emprunter le chemin le plus court.

Colonies de fourmis



Colonies de fourmis

- Résolution du problème de la collecte : cf Netlogo
- De nombreux problèmes d'optimisation comme le voyageur de commerce, le sac à dos, ...

Plan

Algorithmes évolutionnistes

Principe :

- Générer une population d'individus (par exemple, représentation de stratégies par un arbre de décision ou un réseau de neurones)
- Traduction du génotype en phénotype décrivant l'agent
- Évaluation du comportement de l'agent
- Sélection des individus : les individus les plus performants sont conservés ou la probabilité de conserver un individu dépend de ses performances
- Génération de nouveaux individus par croisement ou mutation

Plus de détails : [Wei99], [Don02] et dans l'UE IAR en M2 !

Références I


Stéphane Doncieux, *Algorithmes évolutionnistes : de l'optimisation de paramètres à la conception complète d'un système de contrôle*, Proceedings of Journées MicroDrones. CD-ROM ENSICA/SupAero, 2002.


M. L Puterman, *Markov decision processes : discrete stochastic dynamic programming*, Wiley-Interscience, New York, 2005.


J.R. Quinlan, *Discovering rules by induction from large collections of examples*, Expert Systems in the Microelectronic age (1979), 168–201.


S. Russell and P. Norvig, *Artificial intelligence : A modern approach*, Prentice Hall Series, 2003.


R.S. Sutton and A.G. Barto, *Reinforcement learning : An introduction*, MIT press, Cambrige, MA (1998).


C. Shannon and W. Weaver, *The mathematical theory of communication*, University of Illinois Press, 1948.

SCIENCES SORBONNE UNIVERSITÉ

Références II


Sandip Sen and Gerhard Weiss, *Multiagent systems*, MIT Press, Cambridge, MA, USA, 1999, pp. 259–298.


G Weiss, *Multiagent systems a modern approach to distributed artificial intelligence*, MIT Press, 1999.


Ian H. Witten, Eibe Frank, and Mark A. Hall, *Data mining : Practical machine learning tools and techniques*, 3rd ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.