Apprentissage par renforcement

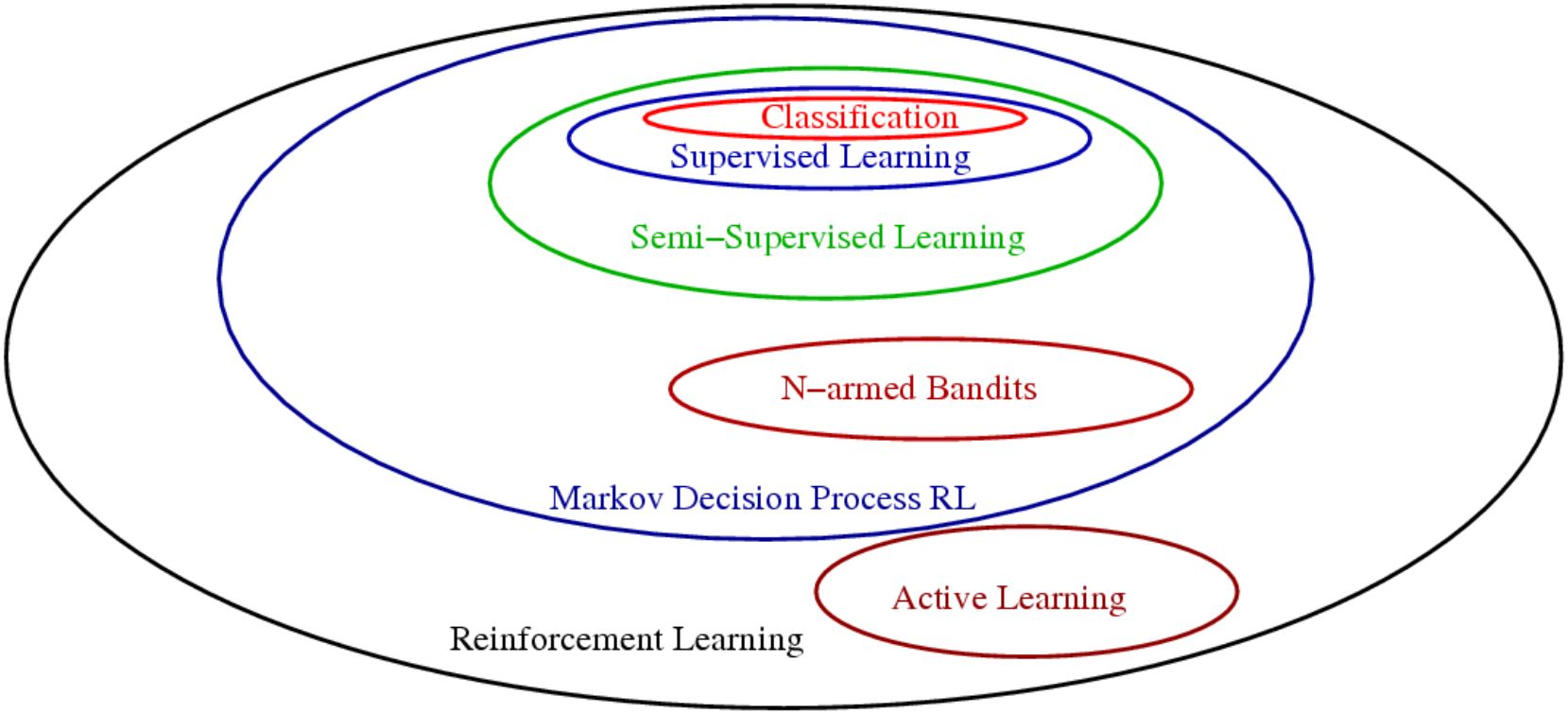
Fabrice CLEROT

TECH/SUSI/TSI

Objectif

- Définir le type de problème abordé
- Fixer quelques éléments de vocabulaire
- Décrire les approches les plus simples
- Faire le lien avec les autres techniques d'apprentissage

Reinforcement Learning is Always Relevant



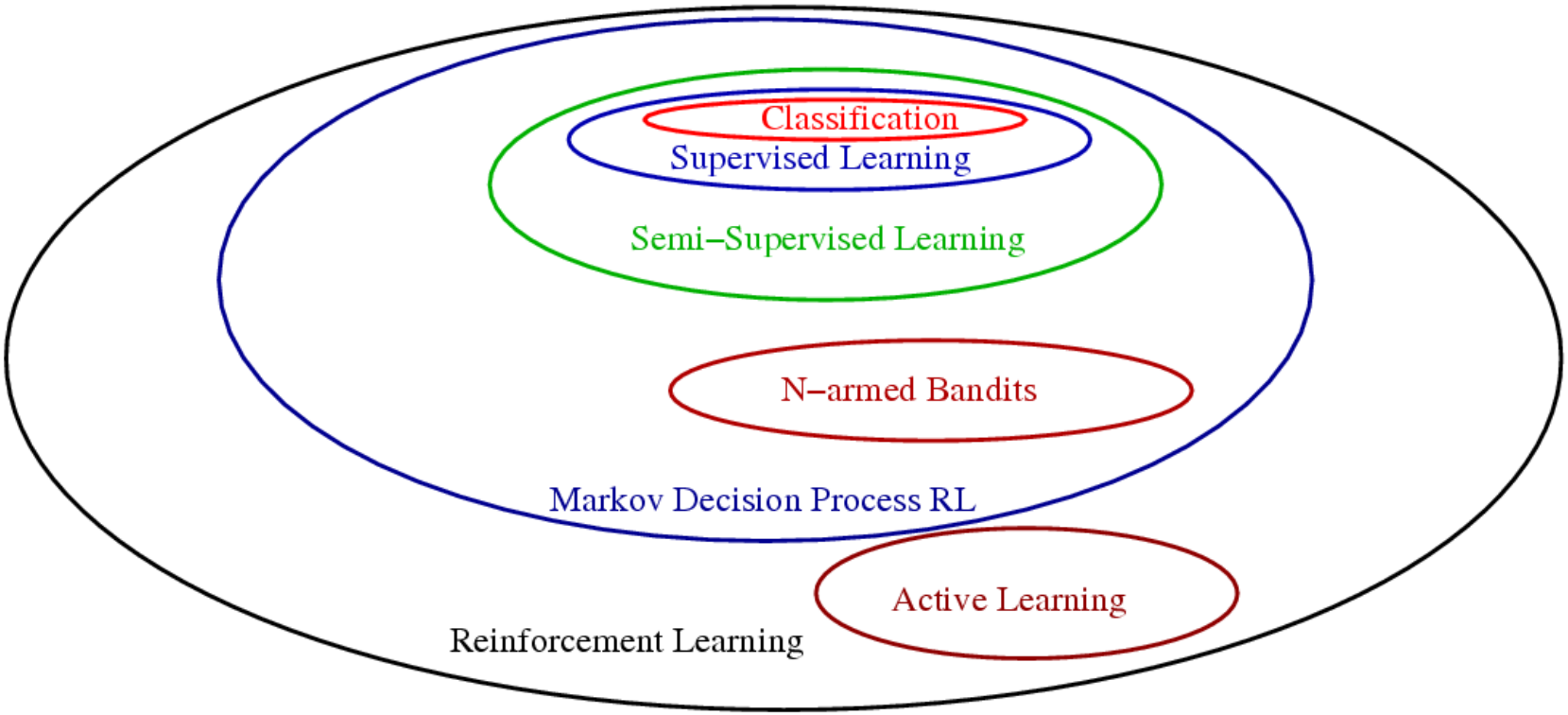
The answer to: "Is this an RL problem?" is always "yes".

The implication: RL theory is broadly applicable.

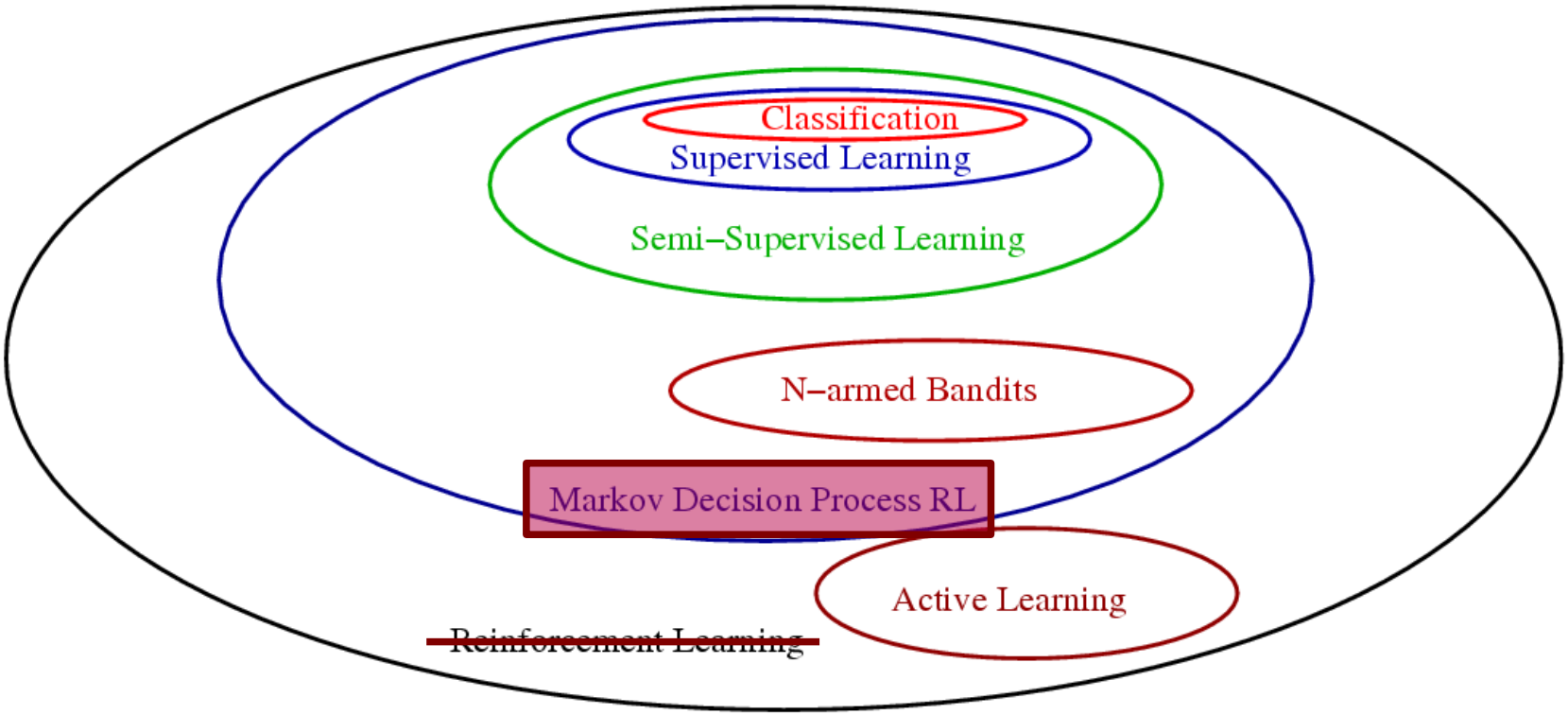
The other implication: RL theory is often only weakly relevant. (breadth+relevance=hard.)

Understanding a problem as an RL problem is the *beginning* to solving it. Whenever possible, you want to understand how the problem is special.

Reinforcement Learning is Always Relevant



Reinforcement Learning is Always Relevant



Plan

- Cadre formel de l'apprentissage par renforcement
- Solution formelle : programmation dynamique
- Solution pratique : apprentissage par différence temporelle (« TD learning »)
- Solution praticable : modéliser pour généraliser
- Extensions diverses

Plan

- Cadre formel de l'apprentissage par renforcement
- Solution formelle : programmation dynamique
- Solution pratique : apprentissage par différence temporelle (« TD learning »)
- Solution praticable : modéliser pour généraliser
- Extensions diverses

Cadre formel

- Interaction entre un agent et son environnement
- Optimisation de la récompense à long terme
- Processus de décision Markovien

Cadre formel : interaction entre un agent et son environnement

- L'agent observe l'univers :
  - Capteur, notion d'état
- L'agent effectue une action :
  - Effecteur, modification de l'état de l'univers
  - Au départ, ces modifications sont inconnues de l'agent

Cadre formel : optimisation de la récompense à long terme

- Après chaque action, l'agent reçoit une récompense immédiate
  - La récompense peut être positive, négative ou nulle
  - La récompense peut être tirée d'une distribution
  - Au départ, ces récompenses sont inconnues de l'agent
- Le but de l'agent est de maximiser sa récompense sur le long terme

Cadre formel : optimisation de la récompense à long terme

- « long terme » ?
  - Équilibre entre les récompenses immédiates (minimes mais sûres) et les récompenses à venir (importantes mais incertaines)
  - Formalisation par un paramètre $\gamma < 1$ de « discount » sur l'avenir
- Autres formalisations possibles
  - Récompense totale sur un horizon T fixe
  - Récompense totale sur toute une trajectoire (si MDP « absorbant »)

Cadre formel : agent « explorateur »

- On se place dans une perspective d'exploration
  - L'agent n'a pas de moyen de « revenir en arrière » une fois une action effectuée (modification irréversible de l'environnement)
  - L'agent ne possède pas de modèle génératif de l'environnement (pas de simulation « exacte » possible)
- On verra que c'est une contrainte forte sur le mécanisme d'apprentissage

Cadre formel : agent « explorateur »

- L'agent peut parfois disposer d'un modèle génératif « exact » de son environnement (contrôle d'un dispositif physique)
- Ne pas disposer d'un modèle génératif « exact » ne signifie pas que l'agent ne peut pas simuler
  - Il faut qu'il construise un modèle génératif de son environnement et simule relativement à ce modèle
  - Par exemple, un (robot) joueur d'échecs construit forcément un tel modèle: « qu'est-ce que je jouerais à la place de mon adversaire ? »

Cadre formel : agent « explorateur »

- Cadre réaliste: l'agent doit agir « sous pression » de son environnement
  - La simulation se fait sous contrainte de temps

- Optimiser le « crédit temps » entre deux actions doit être un objectif de toutes les techniques d'apprentissage par renforcement

Cadre formel : processus de décision Markovien (MDP)

- MDP = {états, actions, transitions, récompenses}
- État x(t)
- Action a(t)
- Récompense associée à l'action a dans l'état x, r(x,a);
  - la distribution de la récompense pour un couple (x, a) est fixe et ne dépend que de ce couple (Markov 1)
  - on notera r(t)=r(x(t), a(t)) pour la récompense obtenue à l'instant t

Cadre formel : processus de décision Markovien (MDP)

- MDP = {états, actions, transitions, récompenses}
- Transitions: changement d'état de l'univers suite à une action de l'agent
  - Probabilité que l'univers passe dans l'état y quand l'action a est effectuée dans l'état x

$$P_{x,y}(a)$$

- Ne dépend que de (x, a) (Markov 2)
- La récompense r(x,a) est indépendante de l'état y

Cadre formel : processus de décision Markovien (MDP)

- MDP = {états, actions, transitions, récompenses}
- Objectif, à partir de l'état x(0), trouver la suite des actions a(0), a(1), ... qui maximise

$$\left\langle \sum_{t=0}^{\infty} \gamma^t r(x(t), a(t)) \right\rangle_{x,r}$$

Cadre formel : processus de décision Markovien (MDP)

- MDP = {états, actions, transitions, récompenses}

- Processus en temps discret
- Les états et les actions sont en nombre fini
- Les distributions des transitions et des récompenses sont inconnues au départ

Plan

- Cadre formel de l'apprentissage par renforcement
- Solution formelle : programmation dynamique
- Solution pratique : apprentissage par différence temporelle (« TD learning »)
- Solution praticable : modéliser pour généraliser
- Extensions diverses

Solution formelle : programmation dynamique

- Deux classes de méthodes
  - Itération sur les politiques (« policy iteration »)
  - Itération sur les valeurs (« value iteration »)

Solution formelle : programmation dynamique, itération sur les politiques

- Politique déterministe:
  - Fonction de l'espace des états dans l'espace des actions
    - « Dans l'état x, faire a »
  - $a=\pi(x)$
- Un processus de décision Markovien admet une solution sous forme de politique déterministe
  - même quand les transitions et les récompenses sont stochastiques

Solution formelle : programmation dynamique, itération sur les politiques

- Deux opérations fondamentales sur une politique
  - Évaluation
  - Amélioration
- L'optimisation se fait en évaluant d'abord une politique puis en l'améliorant, puis en évaluant la politique résultante etc ...

Solution formelle : programmation dynamique, itération sur les politiques

- L'évaluation d'une politique π consiste à calculer la récompense à long terme obtenue en appliquant la politique π à partir de tout état x(0)

$$V^{\pi}(x(0)) = \left\langle \sum_{t=0}^{\infty} \gamma^t r(t) \right\rangle_{x,r} = \left\langle \sum_{t=0}^{\infty} \gamma^t r(x(t), \pi(x(t))) \right\rangle_{x,r}$$

Solution formelle : programmation dynamique, itération sur les politiques

$$V^{\pi}(x(0)) = \left\langle \sum_{t=0}^{\infty} \gamma^t r(t) \right\rangle_{x,r} = \hat{r}(x, \pi(x)) + \left\langle \sum_{t=0}^{\infty} \gamma^t r(t+1) \right\rangle_{x,r}$$

- Equation de Bellman

$$V^{\pi}(x(0)) = \hat{r}(x, \pi(x)) + \gamma \sum_{y} P_{x,y}(\pi(x)) V^{\pi}(y)$$

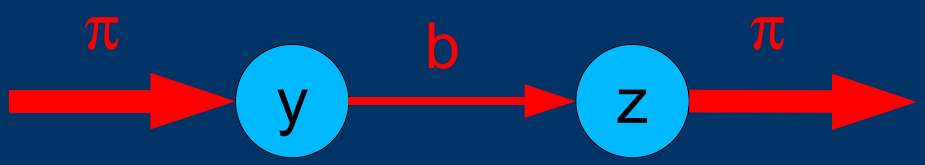
- Si on connaît les récompenses moyennes et les transitions, évaluer une politique revient à résoudre un système linéaire
- Résolution par itérations successives; converge « assez vite »

Solution formelle : programmation dynamique, itération sur les politiques

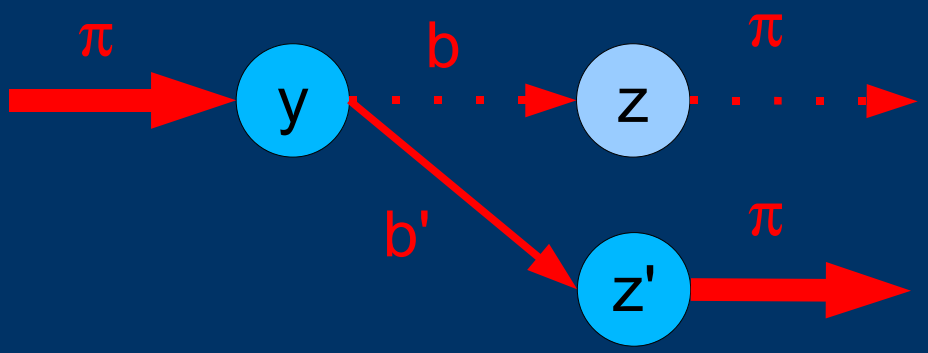
- L'amélioration d'une politique π consiste à construire à partir de π une politique π' qui soit au moins aussi bonne que π

Solution formelle : programmation dynamique, itération sur les politiques

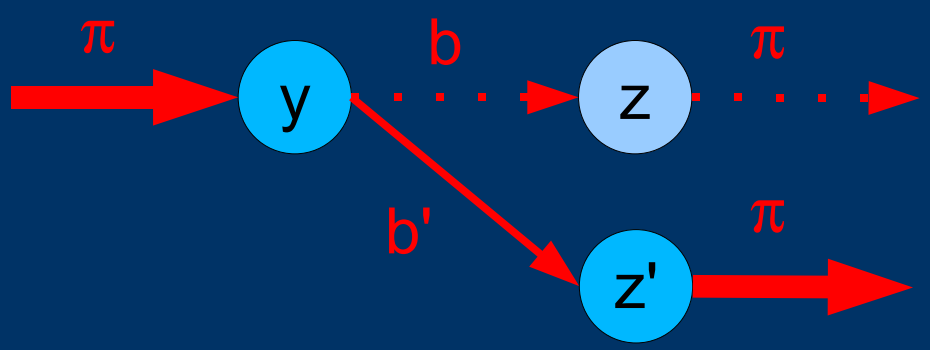
- Une trajectoire selon la politique π (b=π(y)):



- Amélioration possible ?



Solution formelle : programmation dynamique, itération sur les politiques



- Valeur de l'action a en y sous π :

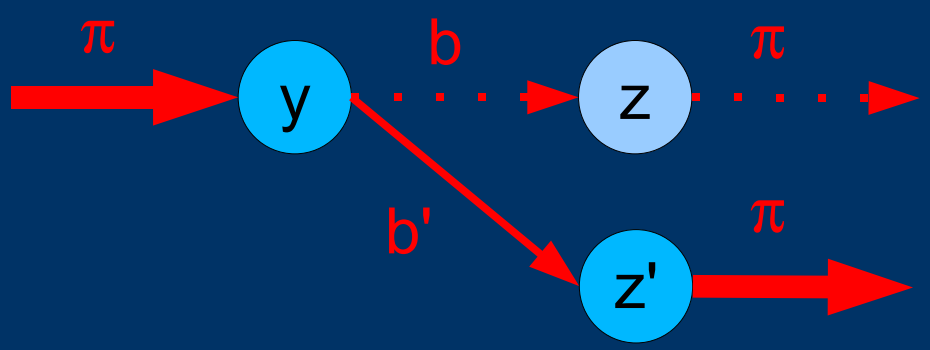
$$Q^{\pi}(y,a)=\hat{r}(y,a)+\gamma\sum_{t}P_{y,t}(a)V^{\pi}(t)$$ (« action value »)

- On retient la meilleure action pour π'

$$\pi'(y)=argmax_{a}\{Q^{\pi}(y,a)\} \quad et \quad \pi'(x)=\pi(x) \quad pour \quad x\neq y$$

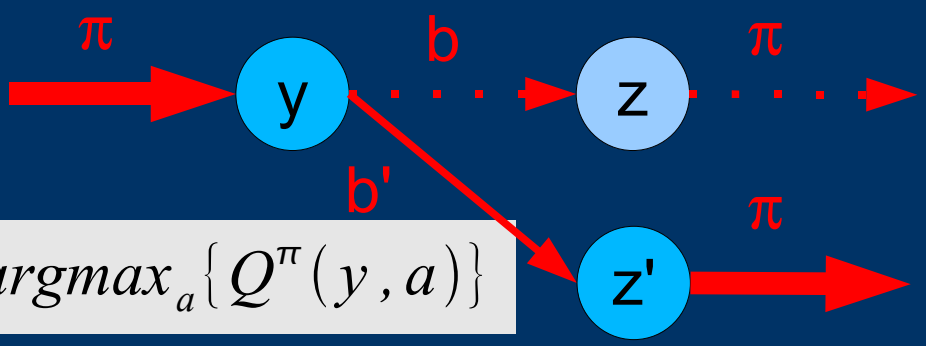
- Evaluation de π' : amélioration de π, $V^{\pi'}(y)\geq V^{\pi}(y)$

Solution formelle : programmation dynamique, itération sur les politiques



- Remarque : l'évaluation de π' est nécessaire
- La politique $\pi \rightarrow y \rightarrow z' \rightarrow \pi$ n'est pas forcément stationnaire
  - Après z', on pourrait repasser par y en suivant π et dans ce cas, on fera la transition $y \rightarrow z$

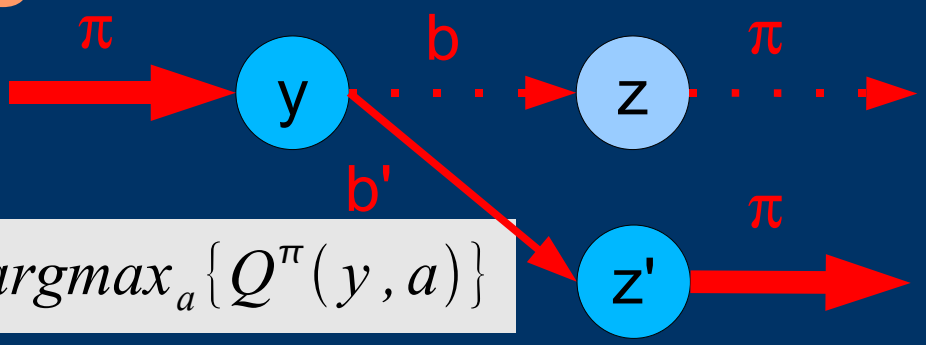
Solution formelle : programmation dynamique, itération sur les politiques



$$b' = \pi'(y) = argmax_a \{ Q^{\pi}(y, a) \}$$

- Par améliorations/évaluations successives, on finit par atteindre une politique optimale, π*
  - V*(x), Q*(x, a), valeurs associées à π*
  - Il faut calculer $b' = \pi'(y) = argmax_a \{ Q^{\pi}(y, a) \}$

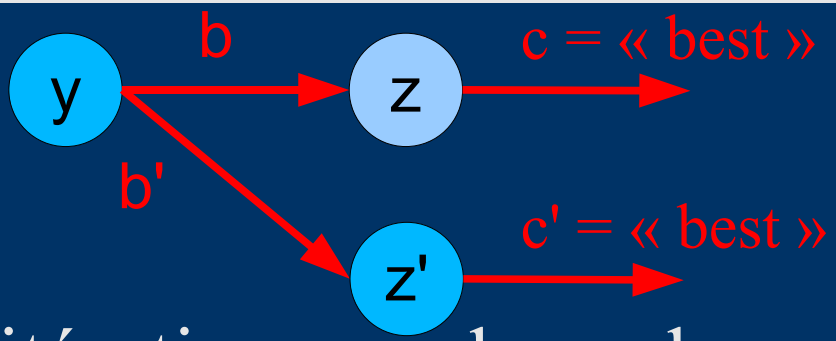
Solution formelle : programmation dynamique, itération sur les valeurs



$$b' = \pi'(y) = argmax_a \{Q^{\pi}(y,a)\}$$

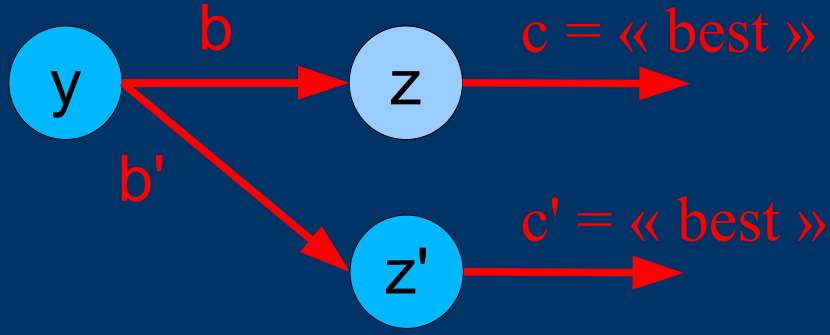
- Améliorer sans évaluer les politiques ?

$$Q'(y,a) = \hat{r}(y,a) + \gamma \sum_t P_{y,t}(a)\, max_c \{Q(t,c)\}$$



- Converge par itération vers les valeurs optimales Q*(y, a)

Solution formelle : programmation dynamique, itération sur les valeurs



$$Q'(y,a)=\hat{r}(y,a)+\gamma\sum_t P_{y,t}(a)\,max_c\{Q(t,c)\}$$

- Grand espace de représentation (Etats x Actions)
- Il faut calculer $argmax_c\{Q(t,c)\}$ pour tous les successeurs possibles

Solution formelle : programmation dynamique

- Itération sur les politiques vs. Itération sur les valeurs :
  - Itération sur les politiques manipule des trajectoires (objets « étendus »)
  - explosion de l'espace de représentation dans le cas de l'itération sur les valeurs

Solution formelle : programmation dynamique

- Solutions qui reposent sur l'acquisition en cours d'apprentissage (essais multiples avec possibilité de revenir au point de départ) des récompenses moyennes et des transitions
  - Comment apprendre efficacement quand on se situe dans un cadre d'exploration (pas de retour possible) ?

Plan

- Cadre formel de l'apprentissage par renforcement
- Solution formelle : programmation dynamique
- Solution pratique : apprentissage par différence temporelle (« TD learning »)
- Solution praticable : modéliser pour généraliser
- Extensions diverses

Solution pratique : Temporal Difference learning (TD learning)

- Deux ingrédients majeurs :
  - Estimer les moyennes à partir d'échantillons aléatoires
  - « bootstrapper » en employant les estimations courantes (incorrectes) dans les équations de Bellman pour approximer la solution
- S 'applique aux deux variantes de programmation dynamique:
  - Itération sur les politiques : « actor-critic learning »
  - Itération sur les valeurs : « Q-learning »

Solution pratique : actor-critic learning

- Intérêt historique
- Pas le temps d'en parler ...
- Pas de garantie de convergence vers π^* si on ne fait pas converger l'évaluation avant de passer à l'amélioration mais « marche en pratique »
- Regain d'intérêt récemment (dans le cas où on possède un modèle génératif du système et de fortes capacités de simulation)

Solution pratique : Q-learning

- Itération sur les valeurs (MDP, rappel) :

$$Q'(x(t),a) = \hat{r}(x(t),a) + \gamma \sum_y P_{x(t),y}(a) max_b \{Q(y,b)\}$$

- 2 principes d'estimation pour l'apprentissage par les différences temporelles:
  - Échantillonnage :
    - Suivre une trajectoire au lieu d'estimer la somme
      - Une réalisation de la récompense de l'action optimale
      - La valeur de l'état suivant obtenu selon cette réalisation de la politique optimale
  - Bootstrapping :
    - Se contenter de la meilleure politique « disponible » pour définir la trajectoire

Solution pratique : Q-learning

$$Q'(x(t),a) = \hat{r}(x(t),a) + \gamma \sum_y P_{x(t),y}(a) max_b \{Q(y,b)\}$$

- Échantillonnage :
  - Suivre une trajectoire au lieu d'estimer la somme
    - Une réalisation de la récompense

      $$\hat{r}(x(t),a) \rightarrow r(x(t),a)$$
    - La valeur de l'état suivant x(t+1) obtenu selon cette réalisation de la politique optimale

      $$\sum_y P_{x(t),y}(a) max_b \{Q(y,b)\} \rightarrow V^{\hat{\pi}}(x(t+1))$$

Solution pratique : Q-learning

$$Q'(x(t),a) = \hat{r}(x(t),a) + \gamma \sum_y P_{x(t),y}(a) max_b \{Q(y,b)\}$$

- Bootstrapping :
  - Se contenter de la meilleure politique « disponible » pour définir la trajectoire

    $$V^{\hat{\pi}}(x(t+1)) \rightarrow V^{\tilde{\pi}}(x(t+1))$$

  - Avec $\tilde{\pi}(x(t)) = max_b Q(x(t),b)$ définissant l'action optimale en x(t)

Q-learning, récapitulation

- Itération sur les valeurs (MDP, rappel) :

$$Q'(x(t),a) = \hat{r}(x(t),a) + \gamma \sum_y P_{x(t),y}(a) \max_b \{Q(y,b)\}$$

- Estimation/bootstrap sur le terme de droite :

$$r(x(t),a) + \gamma \max_b \{Q(x(t+1),b)\}$$

- Mise à jour :

$$Q(x(t),a(t)) \rightarrow Q(x(t),a(t)) + \epsilon [r(x(t),a(t)) + \gamma \max_b \{Q(x(t+1),b)\} - Q(x(t),a(t))]$$

- À tout instant, $\tilde{\pi}(x) = \operatorname{argmax}_a \{Q(x,a)\}$ définit une politique
- Garanties théoriques de convergence vers la politique optimale

Q-learning, deux interprétations

$$Q(x(t),a(t)) \rightarrow Q(x(t),a(t)) + \epsilon[r(x(t),a(t)) + \gamma \, max_b\{Q(x(t+1),b)\} - Q(x(t),a(t))]$$

- La correction vient de la différence entre ce qu'on pensait avoir (Q(x(t), a(t)) et ce qu'on a effectivement obtenu en effectuant l'action a en x
- Temporal Difference

$$Q(x(t),a(t)) \rightarrow (1-\epsilon)Q(x(t),a(t)) + \epsilon[r(x(t),a(t)) + \gamma \, max_b\{Q(x(t+1),b)\}]$$

- Estimation stochastique, style nuées dynamiques

Q-learning, récapitulation

- Garanties théoriques de convergence de π vers π* ?
  - Chaque état est visité infiniment souvent
  - Si on emploie un taux d'apprentissage ε variable au cours du temps
    - $\sum_{t=1}^{\infty} \epsilon_t > \infty$
    - $\sum_{t=1}^{\infty} \epsilon_t^2 < \infty$
- Théorique ... surtout à cause du premier point: dilemme exploration-exploitation

Dilemme exporation-exploitation

- Garanties de convergence seulement si l'ensemble des états sont explorés
- Explorer les états inconnus est
  - Coûteux en temps
  - « Risqué » (dans le cadre du robot « explorateur »)
  - Difficile (comment garantir qu'on a bien exploré tous les états ?)

Dilemme exporation-exploitation

- Traitement essentiellement heuristique ...
  - A chaque étape, choisir une solution non optimale avec une faible probabilité (ε-greedy, « Boltzmann », etc)
- ... jusqu'à récemment, algorithme E^3
  - Garanties sur l'exploration ...
  - ... avec des bornes peu encourageantes (convergence très lente mais pourrait se comporter mieux en pratique)

Récapitulation

- Programmation dynamique:
  - Suppose la possibilité de simuler chaque transition jusqu'à obtenir une estimation robuste des distributions des transitions et des récompenses
  - Simuler ou réaliser de très nombreuses expériences (et n'apprendre qu'à l'issue de chaque trajectoire complète)
- TD learning
  - Compatible avec la contrainte d'exploration:
    - Repose seulement sur la connaissance acquise au cours des actions précédentes
    - Pas de simulation du tout
  - Garanties théoriques dans le cas du Q-learning (si exploration)
  - Convergence extrêmement lente dans certains cas

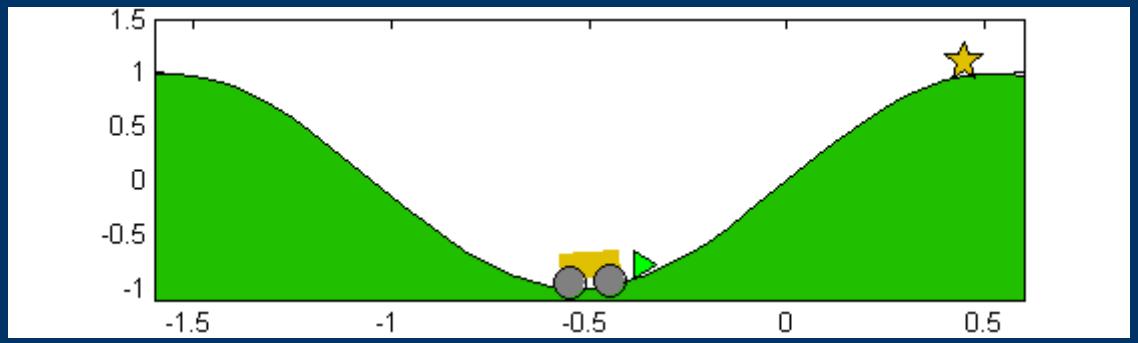
Récapitulation

- Rappel des hypothèses :
  - Actions discrètes
  - États discrets
  - Univers totalement observable
  - Markov
  - Temps discret
- {Actions discrètes, Etats discrets} sont très contraignantes
  - Beaucoup de travaux pour les relâcher: remplacer les tables d'états par des modèles
- {Univers totalement observable, Markov} sont parfois irréalistes
  - Partially Observable MDP, Semi Definite MDP
- {Temps Discret} clairement irréaliste mais difficile à affaiblir

Plan

- Cadre formel de l'apprentissage par renforcement
- Solution formelle : programmation dynamique
- Solution « pratique » : apprentissage par différence temporelle (« TD learning »)
- ~~Solution praticable : modéliser pour généraliser~~
- ~~Extensions diverses~~

Un petit exemple jouet: « mountain car »



Le véhicule n'est pas assez puissant pour monter la pente ...

« Mountain Car »

- Un robot conduit un véhicule
- Objectif: sortir le plus vite possible !

- Etat = (position, vitesse), discrétisation 20x10
- 3 actions possibles: marche avant, point mort, marche arrière

- Récompense: -1 tant qu'il n'est pas sorti, 100 sinon