Programmation dynamique

Présentation et exploitation

dans l'apprentissage par renforcement

T. AL-ANI

A²SI-ESIEE-PARIS

La programmation dynamique = diviser pour régner.

- Partition du problème en sous-problèmes, résolution Récursive de chacun et enfin combiner les solutions pour résoudre le problème initial.

Planification d'un algorithme de programmation dynamique:

1- caractériser la structure d'une solution optimale.

2- définir récursivement la valeur d'une solution optimale.

3- calcul de la valeur d'une solution optimale en remontant progressivement à l'énoncé du problème initial.

4- construire une solution optimale pour les informations calculées.

Programmation dynamique et reinforcement learning

- L'objet de la DP et du RL est l'utilisation des Value Functions pour structurer et organiser la recherche des meilleures politiques.

- On peut trouver les politiques optimales si on a trouvé les meilleures Value Functions:V*(s)

  Q*(s),qui satisfassent aux équations d'optimisation de Bellman.

Policy Evaluation

- Elle consiste en l'évaluation de la fonction de valeur d'état (State Value Function) V^{π} pour une politique arbitraire donnée.

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} [R^{a}_{ss'} + \gamma V(s')] \text{ pour tout s de S}$$

V^{π}

- Existence et unicité sont garanties par un gamma inférieur à 1, la convergence aussi.
- Les méthodes itératives conviennent parfaitement.

$$V_0 \rightarrow V_1 \rightarrow \ldots \rightarrow V_k \rightarrow \ldots \rightarrow V^{\pi}$$

- Un sweep est une une opération de retour appliquée à chaque état.

Evaluation itérative de la politique (Iterative Policy Evaluation)

Input π, the policy to be evaluated
Initialize $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
$\Delta \leftarrow 0$
For each $s \in \mathcal{S}$:
$v \leftarrow V(s)$
$V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
$\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx V^\pi$

Adapté de (R. S. Sutton and A. G. Barto:
Reinforcement Learning: An Introduction http://www-anw.cs.umass.edu/~rich/book/the-book.html) par Tarik AL ANI, A²SI-ESIEE-PARIS

Policy improvement theorem et Policy Iteration

- On dispose de la Value Function V^{π} pour une politique donnée.
- On voudrait savoir si on ne pourrait pas choisir une action $a \neq \pi(s)$ pour l'état s.

Si $Q^{\pi}(s,a) \geq V^{\pi}(s)$ alors sélection de a en s et après reprise de l'utilisation de la politique π. Et seulement dans ce cas.

- On effectue l'opération à chaque fois qu'on a l'état s.
- On étend le procédé sur tous les états et toutes les actions.

Ainsi on obtient une politique π ' plus "greedy"

$$\pi'(s) = \arg\max_{a} Q^{\pi}(s,a)$$

$$\pi'(s) \equiv \arg\max_{a} \sum_{s'} P^{a}_{ss'}[R^{a}_{ss'} + \gamma V^{\pi}(s')]$$

- Possibilité d'améliorer davantage la politique pi', jusqu'à arriver à la politique optimale π^*.

$$\pi_0 \rightarrow V^{\pi} \rightarrow \pi_1 \rightarrow V^{\pi} \rightarrow \ldots \rightarrow \pi^*$$

- Un MDP fini admet un nombre fini de politique , il converge vers une politique optimale et une fonction de valeur optimale.

Itération de la politique

1. **Initialization**

   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

   Repeat

   $\Delta \leftarrow 0$

   For each $s \in \mathcal{S}$:

   $v \leftarrow V(s)$

   $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right]$

   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement

   policy-stable $\leftarrow$ true

   For each $s \in \mathcal{S}$:

   $b \leftarrow \pi(s)$

   $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s') \right]$

   If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ false

   If *policy-stable*, then stop; else go to 2

Adapté de (R. S. Sutton and, Reinforcement Learning: An Introduction http://www-anw.cs.umass.edu/~rich/book/the-book.html) par Tarik AL ANI, A²SI-ESIEE-PARIS

Itération de la valeur (Value Iteration)

- L'itération complète de l'évaluation de la politique (**full policy evaluation backup**)

$$V_{k+1} \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma V_k(s')]$$

- L'itération complète de la valeur (**full value iteration backup**)

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma V_k(s')]$$

Itération de la Valeur Cont.

Initialize V **arbitrarily, e.g.,** $V(s) = 0$**, for all** $s \in \mathcal{S}^+$

Repeat

$\Delta \leftarrow 0$

For each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that

$\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Une autre version de l'amélioration de la politique en utilisant les Q(s,a)

1. Initialization
   $\pi \leftarrow$ an arbitrary deterministic policy
   $Q \leftarrow$ an arbitrary function: $\mathcal{S} \times \mathcal{A}(s) \rightarrow \Re$
   $\theta \leftarrow$ small positive number

2. Policy Evaluation
   Repeat
      $\Delta \leftarrow 0$
      For each $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$:
         $q \leftarrow Q(s,a)$
         $Q(s,a) \leftarrow \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma Q(s', \pi(s'))]$
         $\Delta \leftarrow \max(\Delta, |q - Q(s,a)|)$
   until $\Delta < \theta$

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      $b \leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a Q(s,a)$
      If $b \neq \pi(s)$ then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop; else go to 2

Evaluation de la programmation dynamique

- Assez efficace pour les problèmes de MDP.
- Grande Complexité et temps de calcul.
- Nécessité d'un modèle d'espace parfait.
- Exige un grand espace mémoire.
- Elles convergent, en pratique, plus vite que le prédit le modèle théorique.
- La DP asynchrone est largement utilisée pour des problèmes avec un grand nombre d'états.
- La DP est préférable à la programmation linéaire.