MDP, Q-Learning & ADP: Concepts and Algorithms

Chunyue Song
csong(at)zju.edu.cn

College of Control Science and Engineering, Zhejiang University

December 26, 2018

Outline

Exploration - Exploitation

Average Reward: $Q(k) = \frac{1}{n}\sum_{i=1}^{n} v_i$

$Q_n(k) = Q_{n-1}(k) + \frac{1}{n}(v_n - Q_{n-1}(k))$

Exploration - Exploitation

Average Reward: $Q(k) = \frac{1}{n}\sum_{i=1}^{n} v_i$

$Q_n(k) = Q_{n-1}(k) + \frac{1}{n}(v_n - Q_{n-1}(k))$

Greedy Algorithms

Exploration - Exploitation

MDP, Q-Learning and ADP

C. Song

Gamble

MDP

Algorithm

Model-free Learning

Approximate Dynamic Programming

Average Reward: $Q(k) = \frac{1}{n}\sum_{i=1}^{n} v_i$

$Q_n(k) = Q_{n-1}(k) + \frac{1}{n}(v_n - Q_{n-1}(k))$

Greedy Algorithms

1. Input: K, R, T, ϵ.
2. $r = 0$;
3. $\forall i = 1, 2, \cdots, K : Q(i) = 0$, $count(i) = 0$;
4. for $t = 1, 2, \cdots, T$, do
5. if $rand() < \epsilon$ then $k = uniform\{1, K\}$
6. else $k = \arg\max_i Q(i)$
7. end if
8. $v = R(k)$, $r = r + v$
9. $Q(k) = \frac{Q(k)\times count(k)+v}{count(k)+1}$
10. $count(k) = count(k) + 1$
11. end for
12. Output: r.

Exploration - Exploitation

MDP, Q-Learning and ADP

C. Song

Gamble

MDP

Algorithm

Model-free Learning

Approximate Dynamic Programming

Boltzmann Distr.: $P(k) = \frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^{K} e^{\frac{Q(k)}{\tau}}}$

Exploration - Exploitation

MDP, Q-Learning and ADP

C. Song

Gamble

MDP

Algorithm

Model-free Learning

Approximate Dynamic Programming

Boltzmann Distr.: $P(k) = \frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^{K} e^{\frac{Q(k)}{\tau}}}$

Softmax Algorithms

Exploration - Exploitation

MDP, Q-Learning and ADP

C. Song

Gamble

MDP

Algorithm

Model-free Learning

Approximate Dynamic Programming

Boltzmann Distr.: $P(k) = \frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^{K} e^{\frac{Q(k)}{\tau}}}$

Softmax Algorithms

1. Input: K, R, T, τ.
2. $r = 0$;
3. $\forall i = 1, 2, \cdots, K : Q(i) = 0$, $count(i) = 0$;
4. for $t = 1, 2, \cdots, T$, do
5. $k = P(k)$, and $v = R(k)$
6. $r = r + v$
7. $Q(k) = \frac{Q(k)\times count(k)+v}{count(k)+1}$
8. $count(k) = count(k) + 1$
9. end for
10. Output: r.

MDP: Markov Decision Process

MDP Model

- $MDP = \langle X, A, P, R \rangle$.
- X: State Space; A: Action Space;
  $P : X \times A \to X(P^a_{x \to \acute{x}})$; $R : X \times A \to \mathbb{R}^+(R^a_{x \to \acute{x}})$.

MDP: Markov Decision Process

MDP Model

- $MDP = \langle X, A, P, R \rangle$.
- X: State Space; A: Action Space;
  $P : X \times A \to X(P^a_{x \to \acute{x}})$; $R : X \times A \to \mathbb{R}^+(R^a_{x \to \acute{x}})$.

Objective Function

- $\begin{cases} \min_\pi \mathbb{E}_\pi[\sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x], \\ \min_\pi \mathbb{E}_\pi[\frac{1}{T} \sum_{t=0}^{T} r_{t+1} | x_0 = x]. \end{cases}$

MDP: Markov Decision Process

MDP Model

- $MDP = \langle X, A, P, R \rangle$.
- X: State Space; A: Action Space;
  $P : X \times A \to X(P^a_{x \to \acute{x}})$; $R : X \times A \to \mathbb{R}^+(R^a_{x \to \acute{x}})$.

Objective Function

- $\begin{cases} \min_\pi \mathbb{E}_\pi[\sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x], \\ \min_\pi \mathbb{E}_\pi[\frac{1}{T} \sum_{t=0}^{T} r_{t+1} | x_0 = x]. \end{cases}$

Policy: π Vs. Action: $a, a = \pi(x)$

Value Function

Model-based Learning

Value Function

Model-based Learning

State Value Function: $V^{\pi}(\cdot)$

Value Function

Model-based Learning

State Value Function: $V^{\pi}(\cdot)$

- $\begin{cases} V^{\pi}_{\gamma}(x) = \mathbb{E}_{\pi}[\sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x], \\ V^{\pi}_{T}(x) = \mathbb{E}_{\pi}[\frac{1}{T} \sum_{t=0}^{T} r_{t+1} | x_0 = x]. \end{cases}$

Value Function

Model-based Learning

State Value Function: $V^{\pi}(\cdot)$

- $\begin{cases} V^{\pi}_{\gamma}(x) = \mathbb{E}_{\pi}[\sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x], \\ V^{\pi}_{T}(x) = \mathbb{E}_{\pi}[\frac{1}{T} \sum_{t=0}^{T} r_{t+1} | x_0 = x]. \end{cases}$

State-action Value Function: $Q^{\pi}(x, a)$

Value Function

Model-based Learning

State Value Function: $V^{\pi}(\cdot)$

- $\begin{cases} V^{\pi}_{\gamma}(x) = \mathbb{E}_{\pi}[\sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x], \\ V^{\pi}_{T}(x) = \mathbb{E}_{\pi}[\frac{1}{T} \sum_{t=0}^{T} r_{t+1} | x_0 = x]. \end{cases}$

State-action Value Function: $Q^{\pi}(x, a)$

- $\begin{cases} Q^{\pi}_{\gamma}(x, a) = \mathbb{E}_{\pi}[\sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x, a_0 = a], \\ Q^{\pi}_{T}(x, a) = \mathbb{E}_{\pi}[\frac{1}{T} \sum_{t=0}^{T} r_{t+1} | x_0 = x, a_0 = a]. \end{cases}$

DP: Dynamic Programming

Bellman equation

- $$\begin{cases} V_\gamma^\pi(x) = \sum_{a\in A}\pi(x,a)\sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(R^a_{x\to\acute{x}}+\gamma V^\pi_\gamma(\acute{x})), \\ V_T^\pi(x) = \sum_{a\in A}\pi(x,a)\sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(\frac{1}{T}R^a_{x\to\acute{x}}+\frac{T-1}{T}V^\pi_{T-1}(\acute{x})). \end{cases}$$

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

DP: Dynamic Programming

Bellman equation

- $$\begin{cases} V_\gamma^\pi(x) = \sum_{a\in A}\pi(x,a)\sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(R^a_{x\to\acute{x}}+\gamma V^\pi_\gamma(\acute{x})), \\ V_T^\pi(x) = \sum_{a\in A}\pi(x,a)\sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(\frac{1}{T}R^a_{x\to\acute{x}}+\frac{T-1}{T}V^\pi_{T-1}(\acute{x})). \end{cases}$$
- $$\begin{cases} Q_\gamma^\pi(x,a) = \sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(R^a_{x\to\acute{x}}+\gamma V^\pi_\gamma(\acute{x})), \\ Q_T^\pi(x,a) = \sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(\frac{1}{T}R^a_{x\to\acute{x}}+\frac{T-1}{T}V^\pi_{T-1}(\acute{x})). \end{cases}$$

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

DP: Dynamic Programming

Bellman equation

- $$\begin{cases} V_\gamma^\pi(x) = \sum_{a\in A}\pi(x,a)\sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(R^a_{x\to\acute{x}}+\gamma V^\pi_\gamma(\acute{x})), \\ V_T^\pi(x) = \sum_{a\in A}\pi(x,a)\sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(\frac{1}{T}R^a_{x\to\acute{x}}+\frac{T-1}{T}V^\pi_{T-1}(\acute{x})). \end{cases}$$
- $$\begin{cases} Q_\gamma^\pi(x,a) = \sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(R^a_{x\to\acute{x}}+\gamma V^\pi_\gamma(\acute{x})), \\ Q_T^\pi(x,a) = \sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(\frac{1}{T}R^a_{x\to\acute{x}}+\frac{T-1}{T}V^\pi_{T-1}(\acute{x})). \end{cases}$$

Policy: $a = \pi(x)$

- Deterministic: $\pi : X \mapsto A, a = f(x)$.

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

DP: Dynamic Programming

Bellman equation

- $$\begin{cases} V_\gamma^\pi(x) = \sum_{a\in A}\pi(x,a)\sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(R^a_{x\to\acute{x}}+\gamma V^\pi_\gamma(\acute{x})), \\ V_T^\pi(x) = \sum_{a\in A}\pi(x,a)\sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(\frac{1}{T}R^a_{x\to\acute{x}}+\frac{T-1}{T}V^\pi_{T-1}(\acute{x})). \end{cases}$$
- $$\begin{cases} Q_\gamma^\pi(x,a) = \sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(R^a_{x\to\acute{x}}+\gamma V^\pi_\gamma(\acute{x})), \\ Q_T^\pi(x,a) = \sum_{\acute{x}\in X}P^a_{x\to\acute{x}}(\frac{1}{T}R^a_{x\to\acute{x}}+\frac{T-1}{T}V^\pi_{T-1}(\acute{x})). \end{cases}$$

Policy: $a = \pi(x)$

- Deterministic: $\pi : X \mapsto A, a = f(x)$.
- Probabilistic: $\pi : X \times A \mapsto \mathbb{R}$, the probability of choosing action a on state x, $\sum_a \pi(x,a) = 1$.

Policy Evaluation

Policy Evaluation

Policy Evaluation

Policy Evaluation

1. Input: $MDP = \langle X, A, P, R \rangle$; the policy π; the horizon T.
2. $\forall x \in X : V(x) = 0$;
3. for $t = 1, 2, \cdots,$ do
4. $\forall x \in X$:
   $V'(x) = \sum_{a \in A} \pi(x, a) \sum_{\acute{x} \in X} P^a_{x \to \acute{x}} (\frac{1}{t} R^a_{x \to \acute{x}} + \frac{t-1}{t} V(\acute{x}))$;
5. if $t = T + 1$ then
6. break
7. else $V = V'$
8. end if
9. end for
10. Output: $V(x)$.

Optimal Policy

$\pi^* = \arg\max_\pi V^\pi(x), \forall x \in X, V^*(x) = V^{\pi^*}(x)$

Optimal Policy

$\pi^* = \arg\max_\pi V^\pi(x), \forall x \in X, V^*(x) = V^{\pi^*}(x)$

Bellman Equation

- $$\begin{cases} V^*_\gamma(x) = \max_{a \in A} \sum_{\acute{x} \in X} P^a_{x \to \acute{x}} (R^a_{x \to \acute{x}} + \gamma V^*_\gamma(\acute{x})), \\ V^*_T(x) = \max_{a \in A} \sum_{\acute{x} \in X} P^a_{x \to \acute{x}} (\frac{1}{T} R^a_{x \to \acute{x}} + \frac{T-1}{T} V^*_{T-1}(\acute{x})). \end{cases}$$

Optimal Policy

$\pi^* = \arg\max_\pi V^\pi(x), \forall x \in X, V^*(x) = V^{\pi^*}(x)$

Bellman Equation

- $\begin{cases} V^*_\gamma(x) = \max_{a\in A}\sum_{\acute{x}\in X} P^a_{x\to\acute{x}}(R^a_{x\to\acute{x}} + \gamma V^*_\gamma(\acute{x})), \\ V^*_T(x) = \max_{a\in A}\sum_{\acute{x}\in X} P^a_{x\to\acute{x}}(\frac{1}{T}R^a_{x\to\acute{x}} + \frac{T-1}{T}V^*_{T-1}(\acute{x})). \end{cases}$

$V^*(x) = \max_{a\in A} Q^{\pi^*}(x, a)$

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

Optimal Policy

$\pi^* = \arg\max_\pi V^\pi(x), \forall x \in X, V^*(x) = V^{\pi^*}(x)$

Bellman Equation

- $\begin{cases} V^*_\gamma(x) = \max_{a\in A}\sum_{\acute{x}\in X} P^a_{x\to\acute{x}}(R^a_{x\to\acute{x}} + \gamma V^*_\gamma(\acute{x})), \\ V^*_T(x) = \max_{a\in A}\sum_{\acute{x}\in X} P^a_{x\to\acute{x}}(\frac{1}{T}R^a_{x\to\acute{x}} + \frac{T-1}{T}V^*_{T-1}(\acute{x})). \end{cases}$

$V^*(x) = \max_{a\in A} Q^{\pi^*}(x, a)$

- $\begin{cases} Q^*_\gamma(x, a) = \sum_{\acute{x}\in X} P^a_{x\to\acute{x}}(R^a_{x\to\acute{x}} + \gamma \max_{a'} Q^*_\gamma(\acute{x}, a')), \\ Q^*_T(x, a) = \sum_{\acute{x}\in X} P^a_{x\to\acute{x}}(\frac{1}{T}R^a_{x\to\acute{x}} + \frac{T-1}{T} \max_{a'} Q^*_{T-1}(\acute{x}, a')). \end{cases}$

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

Policy Iteration

- $V^\pi(x) \leq Q^\pi(x, \pi'(x)) = \sum_{x'\in X} P^{\pi'(x)}_{x\to x'}(R^{\pi'(x)}_{x\to x'} + \gamma V^\pi(x'))$
$\leq \sum_{x'\in X} P^{\pi'(x)}_{x\to x'}(R^{\pi'(x)}_{x\to x'} + \gamma Q^\pi(x', \pi'(x'))) = \cdots = V^{\pi'}(x).$

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

Policy Iteration

- $V^\pi(x) \leq Q^\pi(x, \pi'(x)) = \sum_{x'\in X} P^{\pi'(x)}_{x\to x'}(R^{\pi'(x)}_{x\to x'} + \gamma V^\pi(x'))$
$\leq \sum_{x'\in X} P^{\pi'(x)}_{x\to x'}(R^{\pi'(x)}_{x\to x'} + \gamma Q^\pi(x', \pi'(x'))) = \cdots = V^{\pi'}(x).$

$\pi^*(x) = \arg\max_{a\in A} Q^\pi(x, a)$

Policy Iteration

1. Input: $MDP = \langle X, A, P, R \rangle$ and the horizon T.
2. $\forall x \in X: V(x) = 0, \pi(x, a) = \frac{1}{|A(x)|}$;
3. Loop
4. for $t = 1, 2, \cdots,$ do
5. $\forall x \in X$:
   $V'(x) = \sum_{a \in A} \pi(x, a) \sum_{\acute{x} \in X} P^a_{x \to \acute{x}} (\frac{1}{t} R^a_{x \to \acute{x}} + \frac{t-1}{t} V(\acute{x}))$;
6. if $t = T + 1$ then
7. break; else $V = V'$
8. end if
9. end for
10. $\forall x \in X: \pi' = \arg\max_{a \in A} Q(x, a)$
11. if $\forall x \in X: \pi'(x) = \pi(x)$ then
12. break; else $\pi = \pi'$
13. end if
14. End loop
15. Output: π^*.

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

Value Iteration

1. Input: $MDP = \langle X, A, P, R \rangle$, θ and the horizon T.
2. $\forall x \in X: V(x) = 0$;
3. for $t = 1, 2, \cdots,$ do
4. $\forall x \in X$: $V'(x) = \max_{a \in A} \sum_{\acute{x} \in X} P^a_{x \to \acute{x}} (\frac{1}{t} R^a_{x \to \acute{x}} + \frac{t-1}{t} V(\acute{x}))$;
5. if $\max_{x \in X} |V(x) - V'(x)| < \theta$ then
6. break; else $V = V'$
7. end if
8. end for
9. Output: $\pi^*(x) = \arg\max_{a \in A} Q(x, a)$

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

Remark

For every state x, we will have a optimal action $a = f^*(x)$!

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

Remark

For every state x, we will have a optimal action $a = f^*(x)$!

There no the generalization ability problem!

Monte Carlo Simulation

MDP, Q-Learning and ADP

C. Song

Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

No R, No P.

Monte Carlo Simulation

MDP, Q-Learning and ADP

C. Song

Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

No R, No P.

Sampling to get R

Monte Carlo Simulation

MDP, Q-Learning and ADP

C. Song

Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

No R, No P.

Sampling to get R

We start from x_0, implement one policy π, and get the following trajectory

$\langle (x_0, a_0, r_1); (x_1, a_1, r_2); \cdots, (x_{T-1}, a_{T-1}, r_T), x_T \rangle$

On-policy Learning

MDP, Q-Learning and ADP

C. Song

Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

1. Input: Environment: E, Action Space: A, Initial state: x_0 and the horizon T.
2. $Q(x, a) = 0, count(x, a) = 0, \pi(x, a) = \frac{1}{|A(x)|}$;
3. for $s = 1, 2, \cdots,$ do
4. In the environment E and implement π to produce $\langle (x_0, a_0, r_1); (x_1, a_1, r_2); \cdots, (x_{T-1}, a_{T-1}, r_T), x_T \rangle$;
5. for $t = 0, 1, \cdots, T-1$ do
6. $R = \frac{1}{T-t}\sum_{i=t+1}^{T} r_i$, $Q(x_t, a_t) = \frac{Q(x_t,a_t)\times count(x_t,a_t)+R}{count(x_t,a_t)+1}$;
7. $count(x_t, a_t) = count(x_t, a_t) + 1$;
8. end for
9. For all detected state x: $\pi(x) = \begin{cases} \arg\max_{a'} Q(x, a'), Pr(1-\epsilon) \\ a \in A, Pr(\epsilon) \end{cases}$
10. end for
11. Output: π^*

Off-policy Learning

1. Input: Environment: E, Action Space: A, Initial state: x_0 and the horizon T.
2. $Q(x,a)=0, count(x,a)=0, \pi(x,a)=\frac{1}{|A(x)|}$;
3. for $s=1,2,\cdots,$ do
4. In the environment E and implement $\pi(Pr(\epsilon))$ to produce $\langle(x_0,a_0,r_1);(x_1,a_1,r_2);\cdots,(x_{T-1},a_{T-1},r_T),x_T\rangle$;
5. $p_i = \begin{cases} 1-\epsilon+\epsilon/|A|, a_i=\pi(x_i) \\ \epsilon/|A|, a_i \neq \pi(x_i), \end{cases}$
6. for $t=0,1,\cdots,T-1$ do
7. $R=\frac{1}{T-t}(\sum_{i=t+1}^{T} r_i)\prod_{i=t+1}^{T-1}\frac{\mathbb{I}(a_i=pi(x_i))}{p_i}$, $Q(x_t,a_t)=\frac{Q(x_t,a_t)\times count(x_t,a_t)+R}{count(x_t,a_t)+1}$;
8. $count(x_t,a_t)=count(x_t,a_t)+1$;
9. end for
10. $\pi(x)=\arg\max_{a'} Q(x,a')$
11. end for
12. Output: π^*

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

Remark

DP evaluate Q at each step!

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

Remark

DP evaluate Q at each step!

Monte Carlo evaluate Q after each sampling according to each trajectory!

MDP, Q-Learning and ADP
C. Song
Gamble
MDP
Algorithm
Model-free Learning
Approximate Dynamic Programming

Remark

DP evaluate Q at each step!

Monte Carlo evaluate Q after each sampling according to each trajectory!

- $Q^{\pi}_{t+1}(x,a)=Q^{\pi}_{t}(x,a)+\frac{1}{t+1}(R_{t+1}-Q^{\pi}_{t}(x,a))=Q^{\pi}_{t}(x,a)+\alpha(R_{t+1}-Q^{\pi}_{t}(x,a))$
- $Q^{\pi}(x,a)=\sum_{x'\in X}P^{a}_{x\to x'}(R^{a}_{x\to x'}+\gamma V^{\pi}(x'))=\sum_{x'\in X}P^{a}_{x\to x'}(R^{a}_{x\to x'}+\gamma\sum_{a'\in A}\pi(x',a')Q^{\pi}(x',a'))$
- $Q^{\pi}_{t+1}(x,a)=Q^{\pi}_{t}(x,a)+\alpha(R^{a}_{x\to x'}+\gamma Q^{\pi}_{t}(x',a')-Q^{\pi}_{t}(x,a))$

Sarsa

1. Input: Environment: E, Action Space: A, Initial state: x_0, discounted factor γ and step α.
2. $Q(x,a) = 0, \pi(x,a) = \frac{1}{|A(x)|}$;
3. $x = x_0, a = \pi(x)$;
4. for $t = 1, 2, \cdots$, do
5. r, x' = the state when implement a;
6. $a' = \pi^{\epsilon}(x')$;
7. $Q(x,a) = Q(x,a) + \alpha(r + \gamma Q(x',a') - Q(x,a))$;
8. $\pi(x) = \arg\max_{a''} Q(x,a'')$;
9. $x = x', a = a'$;
10. end for
11. Output: π^*

Q-Learning

1. Input: Environment: E, Action Space: A, Initial state: x_0, discounted factor γ and step α.
2. $Q(x,a) = 0, \pi(x,a) = \frac{1}{|A(x)|}$;
3. $x = x_0$;
4. for $t = 1, 2, \cdots$, do
5. r, x' = the state when implement $a = \pi^{\epsilon}(x)$;
6. $a' = \pi(x')$;
7. $Q(x,a) = Q(x,a) + \alpha(r + \gamma Q(x',a') - Q(x,a))$;
8. $\pi(x) = \arg\max_{a''} Q(x,a'')$;
9. $x = x'$;
10. end for
11. Output: π^*

Value Function Approximation

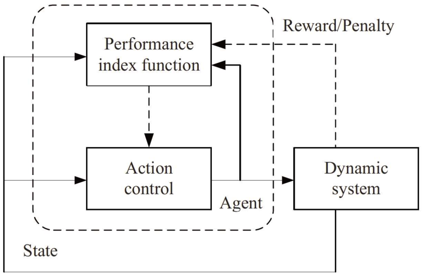
Assumption: Linear Function: $V_\theta(x) = \theta^T x$,

Value Function Approximation

Assumption: Linear Function: $V_\theta(x) = \theta^T x$,

- $E_\theta = \mathbb{E}_{x\sim\pi}[(V^\pi(x) - V_\theta(x))^2]$;
- $-\frac{\partial E_\theta}{\partial \theta} = \mathbb{E}_{x\sim\pi}[2(V^\pi(x) - V_\theta(x))x]$,
- $\theta = \theta + \alpha(V^\pi(x) - V_\theta(x))x =$ $\theta + \alpha(r + \gamma V_\theta(x') - V_\theta(x))x = \theta + \alpha(r + \gamma V_\theta^T x' - \theta^T x)x$,

Approximate Dynamic Programming



Acknowlegement

Thanks for You Attention!

http://person.zju.edu.cn/ChunyueSong

Acknowlegement

Reference

- Zhihua Zhou, Machine Learning, Tsinghua University Press, Bejing, China.