Reinforcement learning

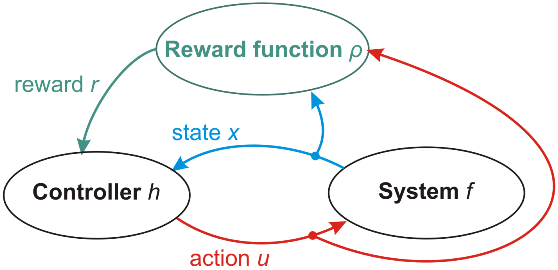
Master CPS, Year 2 Semester 1

Lucian Buşoniu, Florin Gogianu

Part II

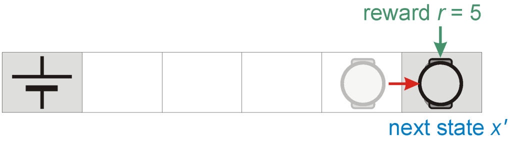
Optimal solution. Dynamic programming

Recap: RL principle



- Interact with system: measure states, apply actions
- Performance feedback in the form of rewards
- Inspired by human and animal learning

Recap: RL elements



- Measure state x
- Apply action u
  per policy $u = h(x)$
- Reach new state x'
  per transition function $x' = f(x, u)$, or $x' \approx \tilde{f}(x, u, \cdot)$
- Receive reward r = quality of the transition
  per reward function $r = \rho(x, u)$, or $r = \tilde{\rho}(x, u, x')$

Part II in plan

- Reinforcement learning problem
- **Optimal solution**
- **Exact dynamic programming**
- Exact reinforcement learning
- Approximation techniques
- Approximate dynamic programming
- Approximate reinforcement learning

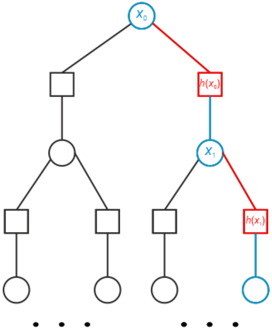
Contents

1. Optimal solution – deterministic case
2. Dynamic programming – deterministic case
3. Analysis of dynamic programming algorithms
4. Optimal solution – stochastic case
5. Dynamic programming – stochastic case

Objective recap and V-function

Find optimal policy h^* that from any x_0 maximizes return

$$V^h(x_0) := \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, h(x_k))$$



V is also called V-function or value function; it is the long-term accumulated reward by following h from x_0

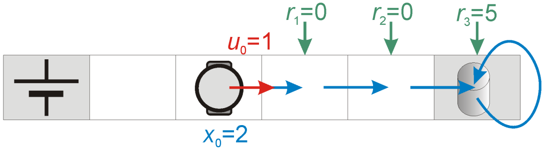
Q-function

Q-function under a policy h
measures the quality of state-action pairs:

$$Q^h(x_0, u_0) := \rho(x_0, u_0) + \gamma V^h(x_1)$$

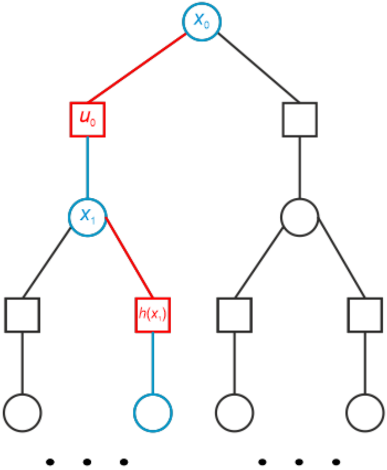
(the return obtained by performing u_0 in x_0 and then following h)

- Note that $V^h(x) = Q^h(x, h(x))$
- Q-function leaves the choice of the first action u_0 open; the rest of the actions are chosen using h:



Q-value illustration

$$Q^h(x_0, u_0) := \rho(x_0, u_0) + \gamma V^h(x_1)$$



V- and Q-functions

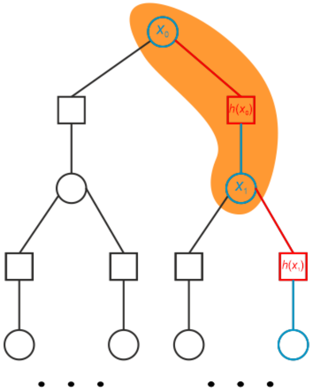
- In this part, we give the math in terms of V-functions first, then Q-functions
- For simplicity, some methods and examples are given for Q-functions only
- Later on, we will prefer Q-functions as they are easier to use for choosing actions

Bellman equation for V^h

- Expand V-function one step forward:

$$V^h(x_0) = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, h(x_k))$$
$$= \rho(x_0, h(x_0)) + \gamma V^h(x_1)$$

Recall: $x_1 = f(x_0, u_0)$



⇒ Bellman equation for V^h

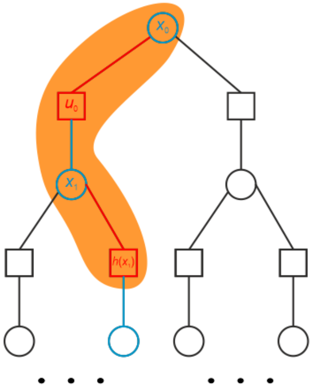
$$V^h(x) = \rho(x, h(x)) + \gamma V^h(f(x, h(x)))$$

Bellman equation for Q^h

- Expand Q-function one step forward:

$$Q^h(x_0, u_0) = \rho(x_0, u_0) + \gamma V^h(x_1)$$
$$= \rho(x_0, u_0) + \gamma[\rho(x_1, h(x_1)) + \gamma V^h(x_2)]$$
$$= \rho(x_0, u_0) + \gamma Q^h(x_1, h(x_1))$$

and since as above $x_1 = f(x_0, u_0)$

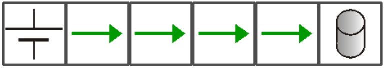


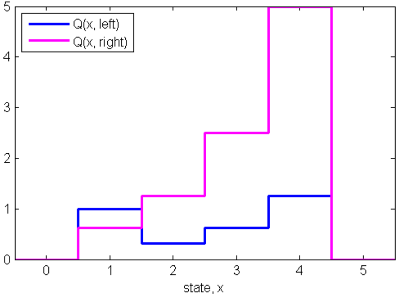
$\Rightarrow$ Bellman equation for Q^h

$$Q^h(x, u) = \rho(x, u) + \gamma Q^h(f(x, u), h(f(x, u)))$$

Cleaning robot: Q-function example

Discount factor $\gamma = 0.5$
Policy $h(x) = 1$, always go right





The optimal solution

- **Optimal V-function and Q-function**:

$$V^* := \max_h V^h \qquad Q^* := \max_h Q^h$$

- Any **greedy policy** in either V^* or Q^*:

$$h^*(x) \in \arg\max_u \underbrace{[\rho(x,u) + \gamma V^*(f(x,u))]}_{Q^*(x,u)}$$

$$h^*(x) \in \arg\max_u Q^*(x,u)$$

  is optimal (achieves maximal returns)

Notes:

- Optimal policy easier to compute from Q^* than from V^*! (without using a model)
- $V^*(x) = \max_u Q^*(x,u)$

Bellman optimality equation for V^*

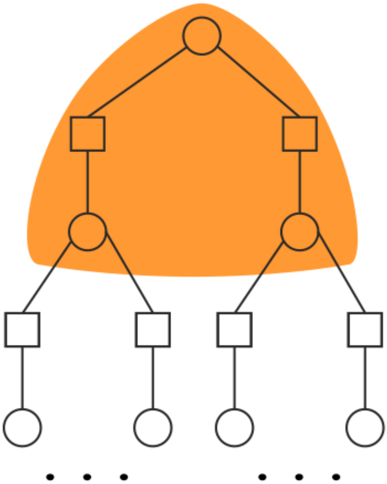
$$
\begin{aligned}
V^*(x_0) &= \max_h V^h(x_0) \\
&= \max_{u_0, u_1, \dots} \left[\rho(x_0, u_0) + \gamma\rho(x_1, u_1) + \gamma^2\rho(x_2, u_2) + \dots\right] \\
&= \max_{u_0} \left[\rho(x_0, u_0) + \gamma \max_{u_1, u_2, \dots} \left[\rho(x_1, u_1) + \gamma\rho(x_2, u_2) + \dots\right]\right] \\
&= \max_{u_0} \left[\rho(x_0, u_0) + \gamma V^*(x_1)\right]
\end{aligned}
$$

and since $x_1 = f(x_0, u_0)$

$\Rightarrow$ **Bellman optimality equation** for V^*

$$
V^*(x) = \max_u \left[\rho(x, u) + \gamma V^*(f(x, u))\right]
$$

Bellman optimality equation for V^*: illustration



Bellman optimality equation for Q^*

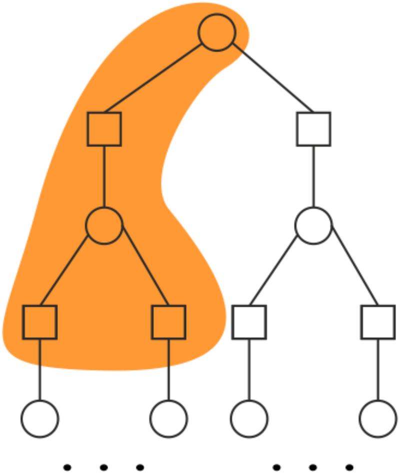
$$
\begin{aligned}
Q^*(x_0, u_0) &= \max_h Q^h(x_0, u_0) \\
&= \max_{u_1, u_2, \dots} \left[\rho(x_0, u_0) + \gamma\rho(x_1, u_1) + \gamma^2\rho(x_2, u_2) + \dots\right] \\
&= \rho(x_0, u_0) + \gamma \max_{u_1, u_2, \dots} \left[\rho(x_1, u_1) + \gamma\rho(x_2, u_2) + \dots\right] \\
&= \rho(x_0, u_0) + \gamma \max_{u_1} \left\{\rho(x_1, u_1) + \gamma \max_{u_2, \dots}[\rho(x_2, u_2) + \dots]\right\} \\
&= \rho(x_0, u_0) + \gamma \max_{u_1} Q^*(x_1, u_1)
\end{aligned}
$$

and since $x_1 = f(x_0, u_0)$

⇒ **Bellman optimality equation** (for Q^*)

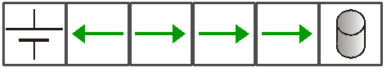
$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u'} Q^*(f(x, u), u')$$

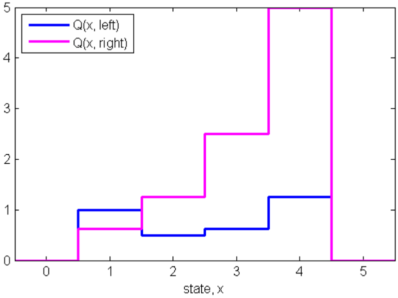
Bellman optimality equation for Q^*: illustration



Cleaning robot: optimal Q-function

Discount factor $\gamma = 0.5$





Checklist

	Bellman eqn-s	value iter.	policy iter.
deterministic	V ☑ / Q ☑	V ☐ / Q ☐	Q ☐
stochastic	V ☐ / Q ☐	V ☐ / Q ☐	Q ☐

Next:

Algorithms to find the optimal solution

In this part: Dynamic programming

1. Value iteration
2. Policy iteration

Dynamic programming in the landscape of algorithms

By model usage:

- Model-based: f, ρ known a priori
- Model-free: f, ρ unknown (reinforcement learning)

By interaction level:

- Offline: algorithm runs in advance
- Online: algorithm runs with the system

Exact vs. approximate:

- Exact: x, u small number of discrete values
- Approximate: x, u continuous (or many discrete values)

Value iteration template

Value iteration

1: find the optimal V-function V^* or Q-function Q^*
2: find h^*, greedy with respect to V^* or Q^*

V-iteration

- Transform the Bellman optimality equation for V^*:

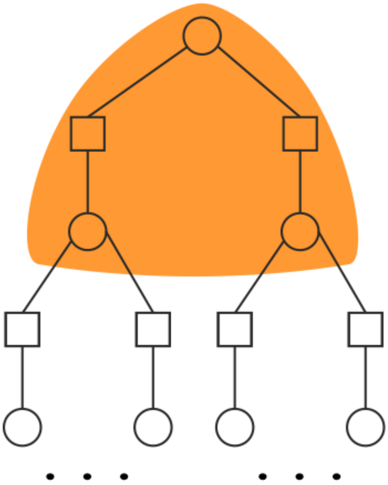
$$V^*(x) = \max_u [\rho(x, u) + \gamma V^*(f(x, u))]$$

into an **iterative update**:

V-iteration

initialize V_0 arbitrarily (e.g. $V_0(x) = 0 \ \forall x$)
repeat at each iteration ℓ
  for all x **do**
    $V_{\ell+1}(x) = \max_u [\rho(x, u) + \gamma V_\ell(f(x, u))]$
  end for
until convergence to V^*
Once V^* available: $h^*(x) = \arg\max_u [\rho(x, u) + \gamma V^*(f(x, u))]$

V-iteration: illustration



Q-iteration

- Transform the Bellman optimality equation for Q^*:

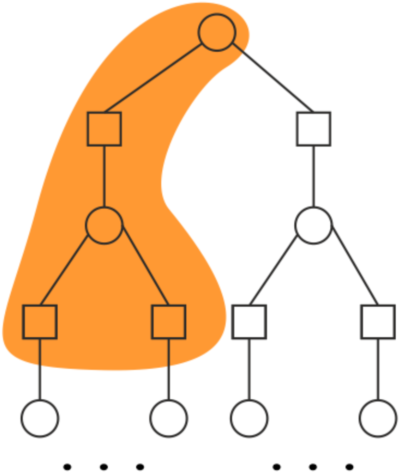
$$Q^*(x,u) = \rho(x,u) + \gamma \max_{u'} Q^*(f(x,u),u')$$

into an **iterative update**:

Q-iteration

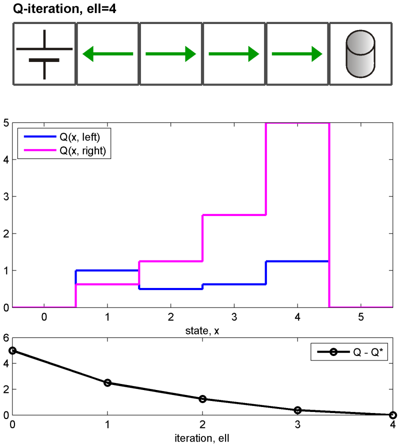
initialize Q_0 arbitrarily (e.g. $Q_0(x,u) = 0 \; \forall x,u$)
repeat at each iteration ℓ
  for all x,u **do**
    $Q_{\ell+1}(x,u) \leftarrow \rho(x,u) + \gamma \max_{u'} Q_\ell(f(x,u),u')$
  end for
until convergence to Q^*
Once Q^* available: $h^*(x) = \arg\max_u Q^*(x,u)$

Q-iteration: illustration



Cleaning robot: Q iteration, demo

Discount factor: $\gamma = 0.5$



Cleaning robot: Q iteration

$$Q_{\ell+1}(x,u) \leftarrow \rho(x,u) + \gamma \max_{u'} Q_\ell(f(x,u),u')$$

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$
Q_0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0
Q_1	0 ; 0	1 ; 0	0 ; 0	0 ; 0	0 ; 5	0 ; 0
Q_2	0 ; 0	1 ; 0	0.5 ; 0	0 ; 2.5	0 ; 5	0 ; 0
Q_3	0 ; 0	1 ; 0.25	0.5 ; 1.25	0.25 ; 2.5	1.25 ; 5	0 ; 0
Q_4	0 ; 0	1 ; 0.625	0.5 ; 1.25	0.625 ; 2.5	1.25 ; 5	0 ; 0
Q_5	0 ; 0	1 ; 0.625	0.5 ; 1.25	0.625 ; 2.5	1.25 ; 5	0 ; 0
h^*	$*$	-1	1	1	1	$*$

$$h^*(x) = \arg\max_u Q^*(x,u)$$

Checklist

	Bellman eqn-s	value iter.	policy iter.
deterministic	V ✓ / Q ✓	V ✓ / Q ✓	Q □
stochastic	V □ / Q □	V □ / Q □	Q □

Policy iteration template

Policy iteration

initialize policy h_0 arbitrarily
repeat at each iteration ℓ
1: policy evaluation: find V^{h_ℓ} or Q^{h_ℓ}
2: policy improvement:
find $h_{\ell+1}(x)$ greedy in V^{h_ℓ} or Q^{h_ℓ}
until convergence to h^*

Policy evaluation with Q-functions

As in Q-iteration:

- Transform the Bellman equation for Q^h:

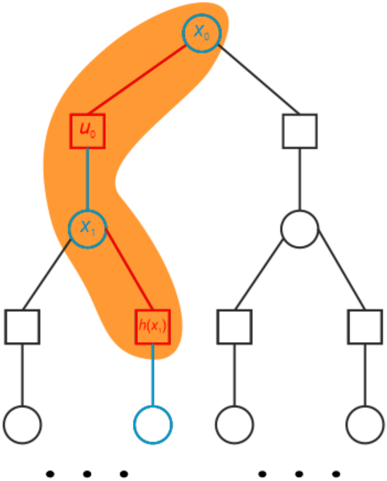
$$Q^h(x,u) = \rho(x,u) + \gamma Q^h(f(x,u), h(f(x,u)))$$

into an iterative update:

Policy evaluation

initialize Q_0 arbitrarily
repeat at each iteration τ
  for all x, u **do**
    $Q_{\tau+1}(x,u) \leftarrow \rho(x,u) + \gamma Q_\tau(f(x,u), h(f(x,u)))$
  end for
until convergence to Q^h

Policy evaluation with Q-functions: illustration



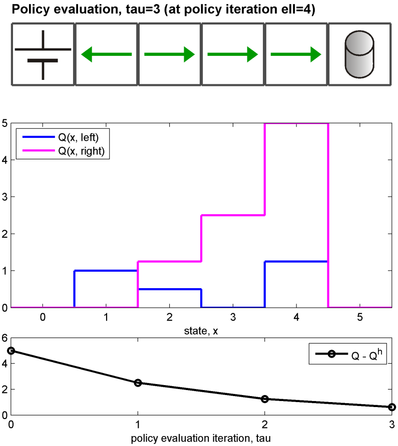
Policy iteration with Q-functions

Policy iteration

initialize policy h_0 arbitrarily
repeat at each iteration ℓ
  1: policy evaluation:
  initialize Q_0 arbitrarily
  repeat at each iteration τ
    for all x, u **do**
      $Q_{\tau+1}(x, u) \leftarrow \rho(x, u) + \gamma Q_\tau(f(x, u), h(f(x, u)))$
    end for
  until convergence to Q^h
  2: policy improvement:
    $h_{\ell+1}(x) \leftarrow \arg\max_u Q^{h_\ell}(x, u)$
until convergence to h^*

Cleaning robot: policy iteration, demo

Initial policy: always go left



Cleaning robot: policy iteration

$$Q_{\tau+1}(x,u) \leftarrow \rho(x,u) + \gamma Q_\tau(f(x,u), h(f(x,u)))$$
$$h_{\ell+1}(x) \leftarrow \arg\max_u Q^{h_\ell}(x,u)$$

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$
h_0	*	-1	-1	-1	-1	*
Q_0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0
Q_1	0 ; 0	1 ; 0	0 ; 0	0 ; 0	0 ; 5	0 ; 0
Q_2	0 ; 0	1 ; 0	0.5 ; 0	0 ; 0	0 ; 5	0 ; 0
Q_3	0 ; 0	1 ; 0.25	0.5 ; 0	0.25 ; 0	0 ; 5	0 ; 0
Q_4	0 ; 0	1 ; 0.25	0.5 ; 0.125	0.25 ; 0	0.125 ; 5	0 ; 0
Q_5	0 ; 0	1 ; 0.25	0.5 ; 0.125	0.25 ; 0.0625	0.125 ; 5	0 ; 0
Q_6	0 ; 0	1 ; 0.25	0.5 ; 0.125	0.25 ; 0.0625	0.125 ; 5	0 ; 0
h_1	*	-1	-1	-1	1	*

...algorithm continues...

Cleaning robot: policy iteration (cont.)

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$
h_1	*	-1	-1	-1	1	*
Q_0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0
...	...	...	...	...	...	...
Q_5	0 ; 0	1 ; 0.25	0.5 ; 0.125	0.25 ; 2.5	0.125 ; 5	0 ; 0
h_2	*	-1	-1	1	1	*
Q_0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0
...	...	...	...	...	...	...
Q_4	0 ; 0	1 ; 0.25	0.5 ; 1.25	0.25 ; 2.5	1.25 ; 5	0 ; 0
h_3	*	-1	1	1	1	*
Q_0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0
...	...	...	...	...	...	...
Q_5	0 ; 0	1 ; 0.625	0.5 ; 1.25	0.625 ; 2.5	1.25 ; 5	0 ; 0
h_4	*	-1	1	1	1	*

Checklist

	Bellman eqn-s	value iter.	policy iter.
deterministic	V ✓ / Q ✓	V ✓ / Q ✓	Q ✓
stochastic	V ☐ / Q ☐	V ☐ / Q ☐	Q ☐

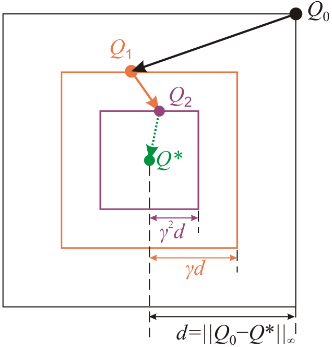
Convergence of value iteration

Stated for Q-functions, but works the same for V-functions:

- Each iteration is a contraction with factor γ:

$$\|Q_{\ell+1} - Q^*\|_\infty \leq \gamma \, \|Q_\ell - Q^*\|_\infty$$

$\Rightarrow$ Q iteration **converges monotonically** to Q^*, **with convergence rate** γ $\Rightarrow \gamma$ helps convergence



Stopping criterion

- Convergence to Q^* guaranteed in the limit, when $\ell \to \infty$
- In practice, the algorithm can be stopped when:

$$\|Q_{\ell+1} - Q_\ell\|_\infty \leq \varepsilon_{\text{qiter}}$$

Convergence of policy iteration

Evaluation component – like value iteration:

- Policy evaluation iterations are contractive with with factor γ

$\Rightarrow$ **converges monotonically** to Q^h, with rate γ

Complete algorithm:

- Policy is either improved or already optimal
- But the maximum number of improvements is finite! ($|U|^{|X|}$)

$\Rightarrow$ **converges** to h^* in a finite number of iterations

Stopping criteria

In practice:

- Policy evaluation can be stopped when:

$$\|Q_{\tau+1} - Q_\tau\| \leq \varepsilon_{\text{peval}}$$

- Overall algorithm can be stopped when:

$$\|h_{\ell+1} - h_\ell\| \leq \varepsilon_{\text{piter}}$$

- Note: $\varepsilon_{\text{piter}}$ can be 0!

Checklist

	Bellman eqn-s	value iter.	policy iter.
deterministic	V ✓ / Q ✓	V ✓ / Q ✓	Q ✓
stochastic	V □ / Q □	V □ / Q □	Q □

Comparison between DP algorithms

Number of iterations

- value iteration $>$ policy iteration

Complexity

- 1 value iteration $>$ 1 policy evaluation iteration
- many evaluation iterations for each policy improvement

$\Rightarrow$ value iteration **???** policy iteration

Exercises for the deterministic case

1. State the policy evaluation algorithm with V-functions, and then the policy iteration algorithm based on it
2. Apply this policy iteration algorithm (on paper) to the cleaning robot problem
3. Prove the contraction property:

$$\|Q_{\ell+1} - Q^*\|_\infty \leq \gamma \|Q_\ell - Q^*\|_\infty$$

4. Write the desired contraction properties for:
   - V^* and V-iteration
   - Q^h and policy evaluation with Q-functions
   - V^h and policy evaluation with V-functions

   and prove these properties

Stochastic MDP reminder

State space X, action space U stay the same

Changes:

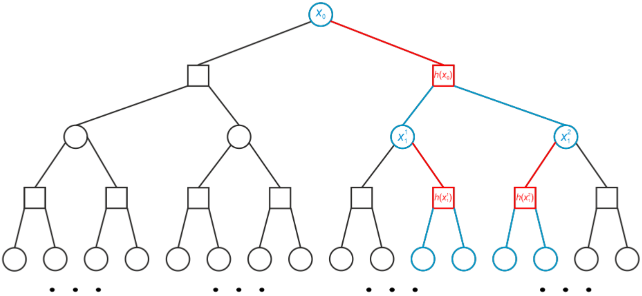
- Transition function $\tilde{f}(x, u, x')$, $\quad \tilde{f} : X \times U \times X \to [0, 1]$
- Reward function $\tilde{\rho}(x, u, x')$, $\quad \tilde{\rho} : X \times U \times X \to \mathbb{R}$

Objective recap and V-function

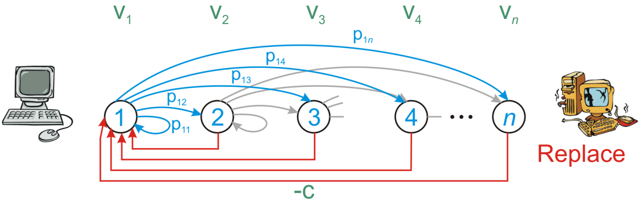
Objective: Find h^* that maximizes **expected** discounted return (V-function):

$$V^h(x_0) = \mathrm{E}_{x_1, x_2, \ldots} \left\{ \sum_{k=0}^{\infty} \gamma^k \tilde{\rho}(x_k, h(x_k), x_{k+1}) \right\}$$

from any x_0



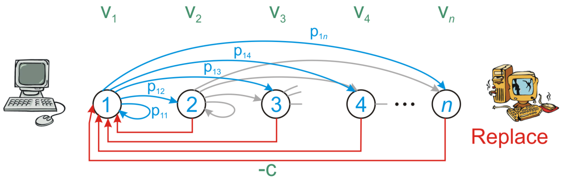
Example: Machine replacement



- Revenue: $v_1 = 1, v_2 = 0.9, ..., v_5 = 0.5$
- Cost of a new machine: $c = 1$
- Wear level increases stochastically:

$$[p_{ij}] = \begin{bmatrix} 0.6 & 0.3 & 0.1 & 0 & 0 \\ 0 & 0.6 & 0.3 & 0.1 & 0 \\ 0 & 0 & 0.6 & 0.3 & 0.1 \\ 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 1.0 \end{bmatrix}$$

Machine replacement: MDP



- Transition function:

$$\tilde{f}(i,u,j) = \begin{cases} p_{ij} & \text{if } u = \text{W and } i \leq j \\ 1 & \text{if } u = \text{R and } j = 1 \\ 0 & \text{otherwise} \end{cases}$$

- Reward function:

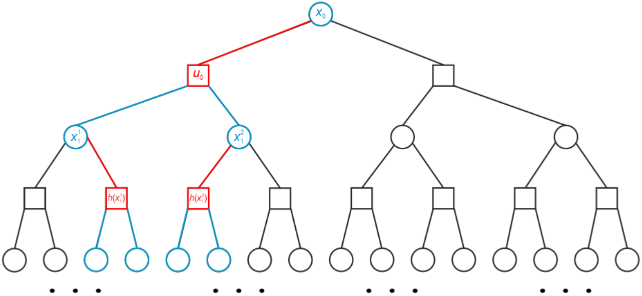
$$\tilde{\rho}(i,u,j) = \begin{cases} v_i & \text{if } u = \text{W} \\ -c + v_1 & \text{if } u = \text{R} \end{cases}$$

Q-function, stochastic

Q-function under a policy h:

$$Q^h(x_0, u_0) = \mathrm{E}_{x_1}\left\{\tilde{\rho}(x_0, u_0, x_1) + \gamma V^h(x_1)\right\}$$

Generalization of deterministic case: **expected** return obtained by performing u_0 in x_0 and then following h



Optimal solutions, stochastic

Most formulas remain unchanged from the deterministic case:

- Optimal V-function and Q-function:

$$V^* := \max_h V^h \qquad Q^* := \max_h Q^h$$

- Greedy policy in Q^*

$$h^*(x) \in \arg\max_u Q^*(x, u)$$

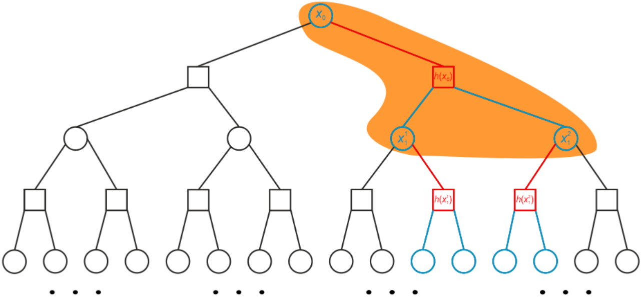
However, the stochastic transitions must be taken into account to find greedy policy in V^*:

$$h^*(x) \in \arg\max_u \mathrm{E}_{x'} \left\{ \tilde{\rho}(x, u, x') + \gamma V^*(x') \right\}$$

$$= \arg\max_u \sum_{x'} \tilde{f}(x, u, x') \left[\tilde{\rho}(x, u, x') + \gamma V^*(x') \right]$$

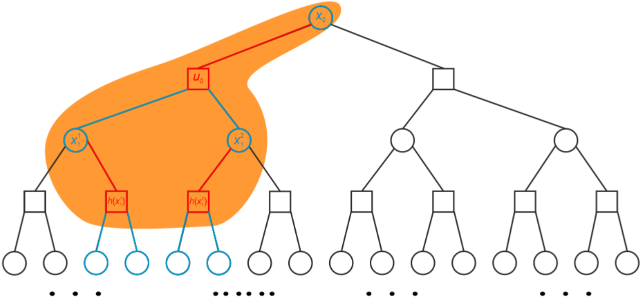
Bellman equation for V^h

$$V^h(x) = \mathrm{E}_{x'}\left\{\tilde{\rho}(x, h(x), x') + \gamma V^h(x')\right\}$$
$$= \sum_{x'} \tilde{f}(x, h(x), x')\left[\tilde{\rho}(x, h(x), x') + \gamma V^h(x')\right]$$



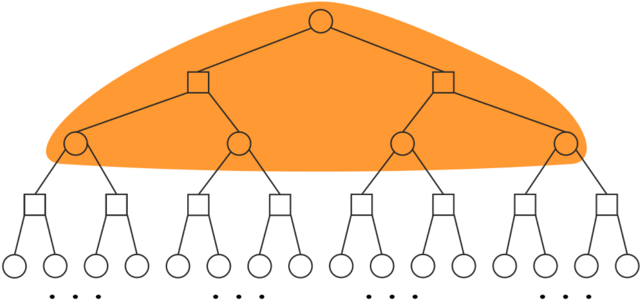
Bellman equation for Q^h

$$
\begin{aligned}
Q^h(x,u) &= \mathrm{E}_{x'}\left\{\tilde{\rho}(x,u,x') + \gamma Q^h(x', h(x'))\right\} \\
&= \sum_{x'} \tilde{f}(x,u,x')\left[\tilde{\rho}(x,u,x') + \gamma Q^h(x', h(x'))\right]
\end{aligned}
$$



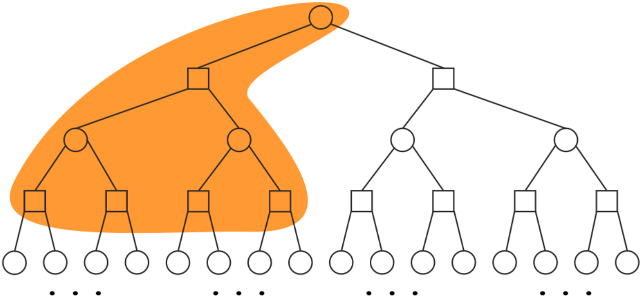
Bellman optimality equation for V^*

$$V^*(x) = \max_u \mathrm{E}_{x'} \left\{ \tilde{\rho}(x,u,x') + \gamma V^*(x') \right\}$$
$$= \max_u \sum_{x'} \tilde{f}(x,u,x') \left[\tilde{\rho}(x,u,x') + \gamma V^*(x') \right]$$



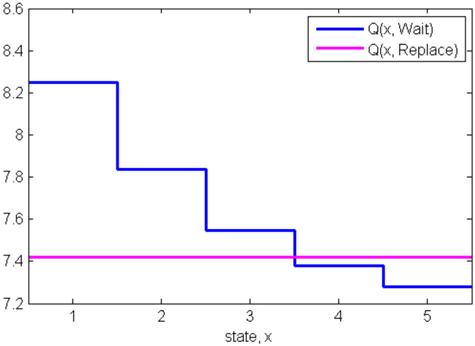
Bellman optimality equation for Q^*

$$
\begin{aligned}
Q^*(x,u) &= \mathrm{E}_{x'}\left\{\tilde{\rho}(x,u,x') + \gamma \max_{u'} Q^*(x',u')\right\} \\
&= \sum_{x'} \tilde{f}(x,u,x')\left[\tilde{\rho}(x,u,x') + \gamma \max_{u'} Q^*(x',u')\right]
\end{aligned}
$$



Machine replacement: Optimal solution

Discount factor $\gamma = 0.9$



Checklist

	Bellman eqn-s	value iter.	policy iter.
deterministic	V ✓ / Q ✓	V ✓ / Q ✓	Q ✓
stochastic	V ✓ / Q ✓	V ☐ / Q ☐	Q ☐

V-iteration, stochastic

- Recall Bellman optimality equation for V^*:

$$V^*(x) = \max_u \sum_{x'} \tilde{f}(x, u, x') \left[\tilde{\rho}(x, u, x') + \gamma V^*(x')\right]$$

- As in deterministic case, transform it into iterative update:

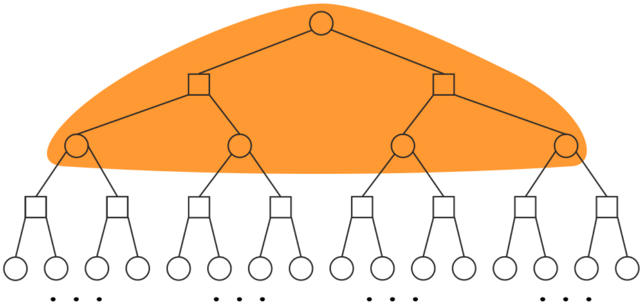
V-iteration, stochastic

initialize V_0 arbitrarily
repeat at each iteration ℓ
  for all x **do**

$$V_{\ell+1}(x) = \max_u \sum_{x'} \tilde{f}(x, u, x') \left[\tilde{\rho}(x, u, x') + \gamma V_\ell(x')\right]$$

  end for
until convergence to V^*
$h^*(x) = \arg\max_u \sum_{x'} \tilde{f}(x, u, x') \left[\tilde{\rho}(x, u, x') + \gamma V^*(x')\right]$

V-iteration, stochastic: illustration



Q-iteration, stochastic

- Recall Bellman optimality equation for Q^*:

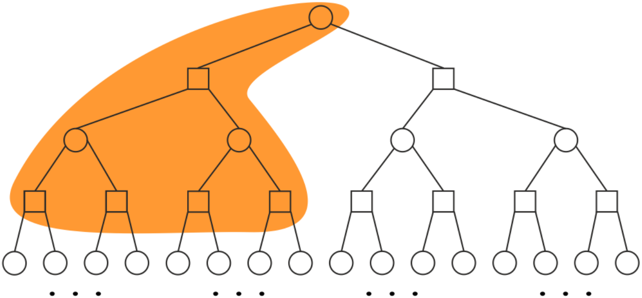
$$Q^*(x,u) = \sum_{x'} \tilde{f}(x,u,x') \left[\tilde{\rho}(x,u,x') + \gamma \max_{u'} Q^*(x',u') \right]$$

- As in deterministic case, transform it into iterative update:

Q-iteration, stochastic

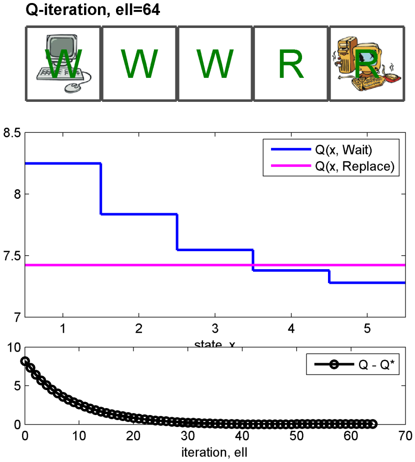
initialize Q_0 arbitrarily
repeat at each iteration ℓ
  for all x, u **do**
    $Q_{\ell+1}(x,u) \leftarrow \sum_{x'} \tilde{f}(x,u,x') \big[\tilde{\rho}(x,u,x') + \gamma \max_{u'} Q_\ell(x',u') \big]$
  end for
until convergence to Q^*
$h^*(x) = \arg\max_u Q^*(x,u)$

Q-iteration, stochastic: illustration

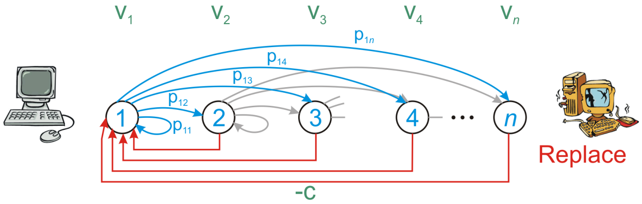


Machine replacement: Q-iteration, demo

Discount factor: $\gamma = 0.9$



Example: Machine replacement



- Revenue: $v_1 = 1, v_2 = 0.9, \ldots, v_5 = 0.5$
- Cost of a new machine: $c = 1$
- Wear level increases stochastically:

$$[p_{ij}] = \begin{bmatrix} 0.6 & 0.3 & 0.1 & 0 & 0 \\ 0 & 0.6 & 0.3 & 0.1 & 0 \\ 0 & 0 & 0.6 & 0.3 & 0.1 \\ 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 1.0 \end{bmatrix}$$

Machine replacement: Q-iteration

$$Q_{\ell+1}(x,u) \leftarrow \sum_{x'} \tilde{f}(x,u,x')[\tilde{\rho}(x,u,x') + \gamma \max_{u'} Q_\ell(x',u')]$$

	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$
Q_0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0
Q_1	1 ; 0	0.9 ; 0	0.8 ; 0	0.7 ; 0	0.6 ; 0
Q_2	1.86 ; 0.9	1.67 ; 0.9	1.48 ; 0.9	1.3 ; 0.9	1.14 ; 0.9
Q_3	2.58 ; 1.67	2.31 ; 1.67	2.05 ; 1.67	1.83 ; 1.67	1.63 ; 1.67
Q_4	3.2 ; 2.33	2.87 ; 2.33	2.55 ; 2.33	2.3 ; 2.33	2.1 ; 2.33
...	...	...	...	...	...
Q_{64}	8.25 ; 7.42	7.84 ; 7.42	7.55 ; 7.42	7.38 ; 7.42	7.28 ; 7.42
Q_{65}	8.25 ; 7.42	7.84 ; 7.42	7.55 ; 7.42	7.38 ; 7.42	7.28 ; 7.42
h^*	W	W	W	R	R

$$h^*(x) = \arg\max_u Q^*(x,u)$$

Checklist

	Bellman eqn-s	value iter.	policy iter.
deterministic	V ✓ / Q ✓	V ✓ / Q ✓	Q ✓
stochastic	V ✓ / Q ✓	V ✓ / Q ✓	Q ☐

Policy iteration, stochastic

Algorithm template remains unchanged

Policy iteration

initialize policy h_0 arbitrarily
repeat at each iteration ℓ
1: policy evaluation: find V^{h_ℓ} or Q^{h_ℓ}
2: policy improvement:
find $h_{\ell+1}(x)$ greedy in V^{h_ℓ} or Q^{h_ℓ}
until convergence to h^*

Policy evaluation with Q-functions, stochastic

- Bellman equation for Q^h in the stochastic case:

$$Q^h(x,u) = \sum_{x'} \tilde{f}(x,u,x') \left[\tilde{\rho}(x,u,x') + \gamma Q^h(x', h(x')) \right]$$

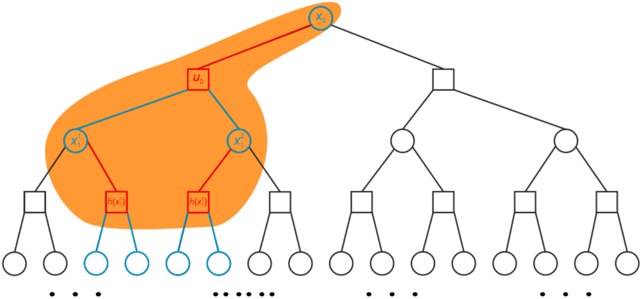
Policy evaluation, stochastic

initialize Q_0 arbitrarily
repeat at each iteration τ
  for all x, u **do**

$$Q_{\tau+1}(x,u) \leftarrow \sum_{x'} \tilde{f}(x,u,x') [\tilde{\rho}(x,u,x') + \gamma Q_\tau(x', h(x'))]$$

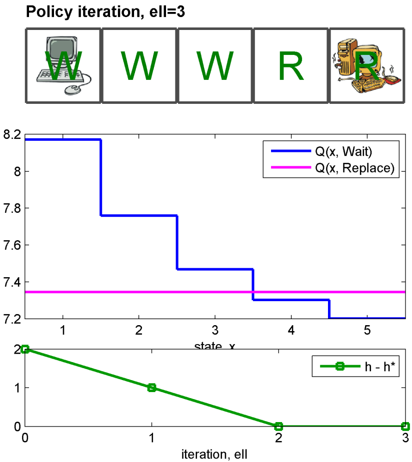
  end for
until convergence to Q^h

Illustration



Machine replacement: policy iteration, demo

Discount factor: $\gamma = 0.9$, $\varepsilon_{\text{peval}} = 0.01$



Machine replacement: policy iteration

$$Q_{\tau+1}(x,u) \leftarrow \sum_{x'} \tilde{f}(x,u,x')[\tilde{\rho}(x,u,x') + \gamma Q_\tau(x', h(x'))]$$

$$h_{\ell+1}(x) \leftarrow \arg\max_u Q^{h_\ell}(x,u)$$

	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$
h_0	W	W	W	W	W
Q_0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0
Q_1	1 ; 0	0.9 ; 0	0.8 ; 0	0.7 ; 0	0.6 ; 0
Q_2	1.86 ; 0.9	1.67 ; 0.9	1.48 ; 0.9	1.3 ; 0.9	1.14 ; 0.9
Q_3	2.58 ; 1.67	2.31 ; 1.67	2.05 ; 1.67	1.83 ; 1.67	1.63 ; 1.67
...	...	...	...	...	...
Q_{39}	7.51 ; 6.75	6.95 ; 6.75	6.49 ; 6.75	6.17 ; 6.75	5.9 ; 6.75
Q_{40}	7.52 ; 6.75	6.96 ; 6.75	6.5 ; 6.75	6.18 ; 6.75	5.91 ; 6.75
h_1	W	W	R	R	R

...algorithm continues...

Machine replacement: policy iteration (cont.)

	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$
h_1	W	W	R	R	R
Q_0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0
...	...	...	...	...	...
Q_{43}	8.01 ; 7.2	7.57 ; 7.2	7.27 ; 7.2	7.17 ; 7.2	7.07 ; 7.2
h_2	W	W	W	R	R
Q_0	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0
...	...	...	...	...	...
Q_{43}	8.17 ; 7.35	7.76 ; 7.35	7.47 ; 7.35	7.3 ; 7.35	7.2 ; 7.35
h_3	W	W	W	R	R

Checklist

	Bellman eqn-s	value iter.	policy iter.
deterministic	V ✓ / Q ✓	V ✓ / Q ✓	Q ✓
stochastic	V ✓ / Q ✓	V ✓ / Q ✓	Q ✓

Analysis

All deterministic-case results remain true in the stochastic case:

- Value iteration converges monotonically to V^* or Q^*, with rate γ
- Policy iteration converges to h^* in finitely many iterations (and policy evaluation converges to V^h or Q^h with rate γ)

- Number of value iterations $>$ policy iterations
- Complexity of 1 value iteration $>$ 1 policy evaluation iteration

$\Rightarrow$ Value iteration **???** policy iteration

Exercises for the stochastic case

1. Write V^* as a function of Q^*, and V^h as a function of Q^h
2. State the policy evaluation algorithm with V-functions, and then the policy iteration algorithm based on it
3. Write the desired contraction properties for:
   - V^* and V-iteration
   - Q^* and Q-iteration
   - Q^h and policy evaluation with Q-functions
   - V^h and policy evaluation with V-functions

   and prove these properties

Hint: Make sure you have solved the corresponding deterministic-case exercises before attempting exercises 2-3 above.

Key terms in this part

- dynamic programming, DP
- Q-function, V-function
- Bellman equation
- value iteration
- Q-iteration
- policy iteration
- policy evaluation
- policy improvement