BASIC ALGORITHMS

Algorithms for small problems

Diego Klabjan
Professor, Industrial Engineering and Management Sciences

Northwestern | McCORMICK SCHOOL OF ENGINEERING

Outline

- Enumeration
- Value iteration
- Policy iteration
  - Value of policy
- Hybrid

ENUMERATION

Bellman's Optimality Equation

- $V_t(s_t) = \max_{a_t \in \mathcal{A}} [r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} V_{t+1}(s_{t+1}|s_t)]$
- Compute value functions recursively
  - Training (planning)
- Given computed value functions
  - 'Measure' state
  - Solve optimization problem

$$\max_{a \in \mathcal{A}} [r(s, a) + \gamma E_{\bar{s} \sim p(\bar{s}|s, a)} V(\bar{s}|s)]$$

  - Often not many actions – enumerate
  - Often deterministic system – no expectation

Enumeration

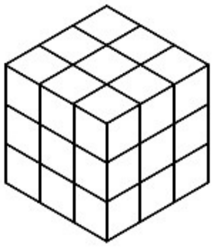
- For *t*=*T* down to 0
  - For each possible state s_t
    - Compute

$$V_t(s_t) = \max_{a_t \in \mathcal{A}} [r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} V_{t+1}(s_{t+1}|s_t)]$$

- Works if
  - Small number of states
  - Small number of actions
  - Somehow cope with expectation
- Three courses of dimensionality

Enumeration Example

- 5 drones in a room of size 30 x 30 x 30 (feet)
  - Discretize by 6 inches
- Guide them to avoid collisions
  - Each one has its own destination
- Number of possible states $(60 \cdot 60 \cdot 60)^5 \approx 4.7 \cdot 10^{26}$
  - Small room, imagine a warehouse
  - Not capturing angle
- Actions
  - $27^3 \approx 20{,}000$
  - Tractable



http://www.m759.net/wordpress/?s=galois+cube

VALUE ITERATION

Enumeration

- Episode length vary or is infinite
  - Enumeration does not work
  - Vast majority of practical problems
- Recursively defined optimality equation

$$V(s) = \max_{a \in \mathcal{A}} [r(s,a) + \gamma E_{\bar{s} \sim p(\bar{s}|s,a)} V(\bar{s}|s)]$$

- Can be shown

$$V_t^T(s_t) = \max_{a_t \in \mathcal{A}} [r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} V_{t+1}^{\mathrm{T}}(s_{t+1}|s_t)]$$

$$V(s) = \lim_{T \to \infty} V_0^T(s)$$

Value Iteration

- $V(s) = \max_{a \in \mathcal{A}} [r(s, a) + \gamma E_{\bar{s} \sim p(\bar{s}|s, a)} V(\bar{s}|s)]$
- Assume right-hand side known
  - Use approximate *V*
  - Can compute left-hand side
    - Gives better approximation of *V*

Value Iteration

- For $k = 0,1,2,\cdots$
  - For each possible state s
    - Compute

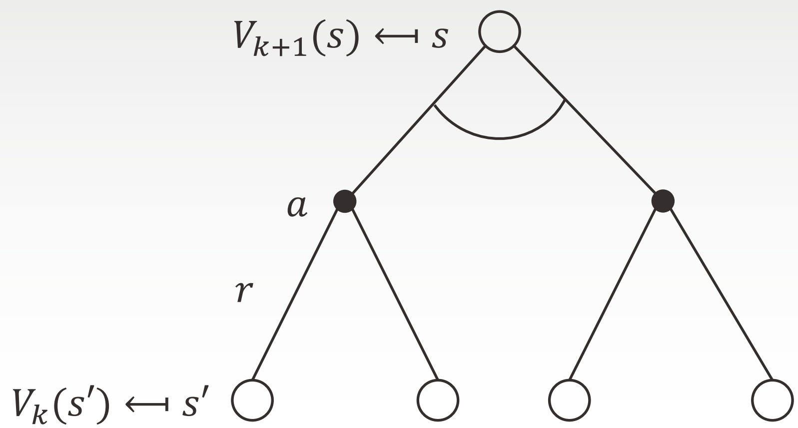
$$V_{k+1}(s) = \max_{a \in \mathcal{A}}[r(s,a) + \gamma E_{\bar{s} \sim p(\bar{s}|s,a)} V_k(\bar{s}|s)]$$

- If discount factor less than 1 and everything is finite
  - Convergence (pointwise) to optimal value function
- Same pitfalls as enumeration
- No explicit policy

Value Function and Policy

- $\pi(s) = \underset{a \in \mathcal{A}}{argmax}[r(s,a) + \gamma E_{\bar{s} \sim p(\bar{s}|s,a)} V(\bar{s}|s)]$
  - If V optimal, π is optimal
- $V(s) = r(s, \pi(s)) + \gamma E_{\bar{s} \sim p(\bar{s}|s,a)} V^{\pi}(\bar{s}|s)$
  - Must know V^{π}
  - If π is optimal, V is optimal

Value Iteration



$$V_{k+1}(s) = \max_{a \in \mathcal{A}} \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V_k(s') \right)$$

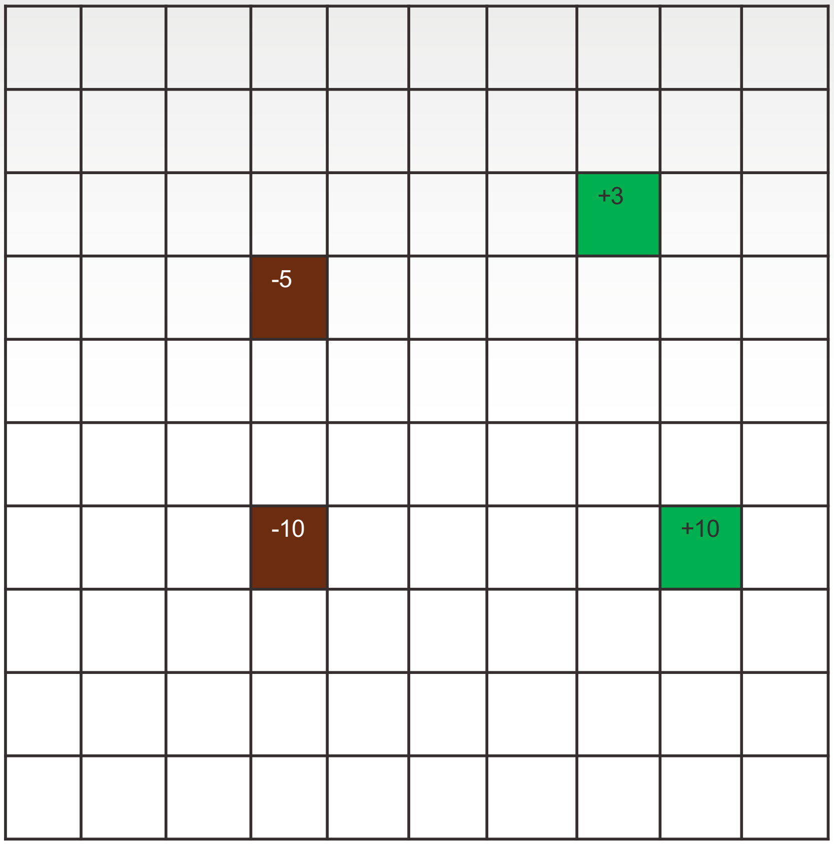
$$R_s^a = r(s,a)$$

$$R^a = \big(r(s,a)\big)_{s \in S}$$

$$\boldsymbol{V}_{k+1} = \max_{a \in \mathcal{A}} \boldsymbol{R}^{\boldsymbol{a}} + \gamma \boldsymbol{P}^{\boldsymbol{a}} \boldsymbol{V}_k$$

Apply for each state

Example



- Actions: up, down, left, or right.
  - 0.7 chance of going one step in the desired direction
  - 0.1 chance of going one step in any of the other three directions
- Bump into the outside wall
  - Penalty of 1 (reward of -1)
  - Agent does not actually move
- There are four rewarding states
  - In each of these states, the agent gets the reward after it carries out an action in that state
    - Not when it enters the state.

https://artint.info/html/ArtInt_224.html#gridworld-ex

Example

- 9 cells around +10
- Discount 0.9
- Start with value function of 0

0	0	-0.1
0	10	-0.1
0	0	-0.1

0	6.3	-0.1
6.3	9.8	6.2
0	6.3	-0.1

4.5	6.2	4.4
6.2	9.7	6.6
4.5	6.1	4.4

Demo of Example

- http://www.cs.ubc.ca/~poole/demos/mdp/vi.html

Example: Shortest Path

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

Other Applications of Value Iteration

- Shortest Path can be solved by value iteration
- Other algorithms
  - Levensthein distance
  - String algorithms
    - String alignment
    - Dynamic time warping
      - Generalization of Levensthein
  - Graphical models
    - Viterbi algorithm

POLICY ITERATION

Evaluating Policy

- Given policy π find V^{π}

$$V^{\pi}(s) = E_{\substack{a\sim\pi(a|s) \\ \bar{s}\sim p(\bar{s}|s,a)}} [r(s,a) + \gamma V^{\pi}(\bar{s}|s)]$$

- Can use similar idea to value iteration
- Given approximate right-hand side
  - Find better left-hand side by using the equation

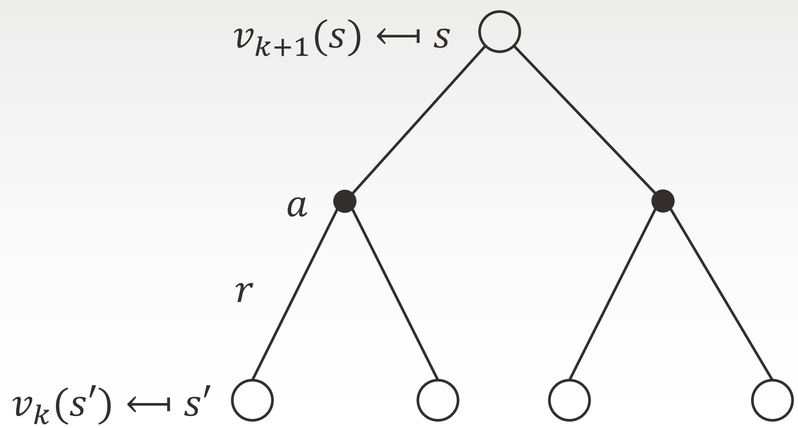
Iterative Policy Evaluation

- Problem
  - Evaluate given policy π
- Solution: iterative application of Bellman expectation equation
- $v_1 \rightarrow v_2 \rightarrow \; \ldots \rightarrow v_\pi$
- For $k = 0,1,2,\cdots$
  - For each possible state s compute

$$v_{k+1}(s) = E_{\substack{a\sim\pi(a|s) \\ \bar{s}\sim p(\bar{s}|s,a)}} [r(s,a) + \gamma v_k(\bar{s}|s)]$$

- Convergence to V^π can be proven

Iterative Policy Evaluation



$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_k(s') \right)$$

$$\pi(a|s) = P[a = \pi(s)]$$

$$\boldsymbol{v}_{k+1} = \boldsymbol{R}^{\boldsymbol{\pi}} + \gamma \boldsymbol{P}^{\boldsymbol{\pi}} \boldsymbol{v}_k$$

Apply for each state

Evaluate Policy

$$\boldsymbol{v}_{k+1} = \boldsymbol{R}^{\boldsymbol{\pi}} + \gamma \boldsymbol{P}^{\boldsymbol{\pi}} \boldsymbol{v}_k$$

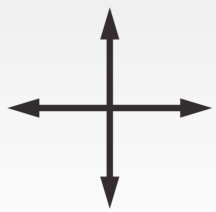
- Limit $k \to \infty$

$$\boldsymbol{v}^{\boldsymbol{\pi}} = \boldsymbol{R}^{\boldsymbol{\pi}} + \gamma \boldsymbol{P}^{\boldsymbol{\pi}} \boldsymbol{v}^{\boldsymbol{\pi}}$$

$$\boldsymbol{v}^{\boldsymbol{\pi}} = (\boldsymbol{I} - \gamma \boldsymbol{P}^{\boldsymbol{\pi}})^{-1} \boldsymbol{R}^{\boldsymbol{\pi}}$$

- Algorithm is a way to compute the inverse
  - Inverse exists if discount less than 1
- $\boldsymbol{R}^{\boldsymbol{\pi}} = \left(E_{a \sim \pi(a|s)} r(s,a) \right)_s = \left(\sum_{a \in \mathcal{A}} \pi(a|s)\, r(s,a) \right)_s$
- $\boldsymbol{P}^{\boldsymbol{\pi}} = \left(\sum_{a \in \mathcal{A}} \pi(a|s) P_{ss'}^{a} \right)_{s,s'}$

Example of Evaluating Random Policy



actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$r = -1$
on all transitions

- Undiscounted episodic MDP ($\gamma = 1$)
- Nonterminal states 1, … ,14
- Shaded squares terminal nodes
- Actions leading out of the grid leave state unchanged
- Reward is -1 until the terminal state is reached
- Agent follows uniform random policy
  - Each action selected with same probability

Example of Evaluating Random Policy

v_k for random policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

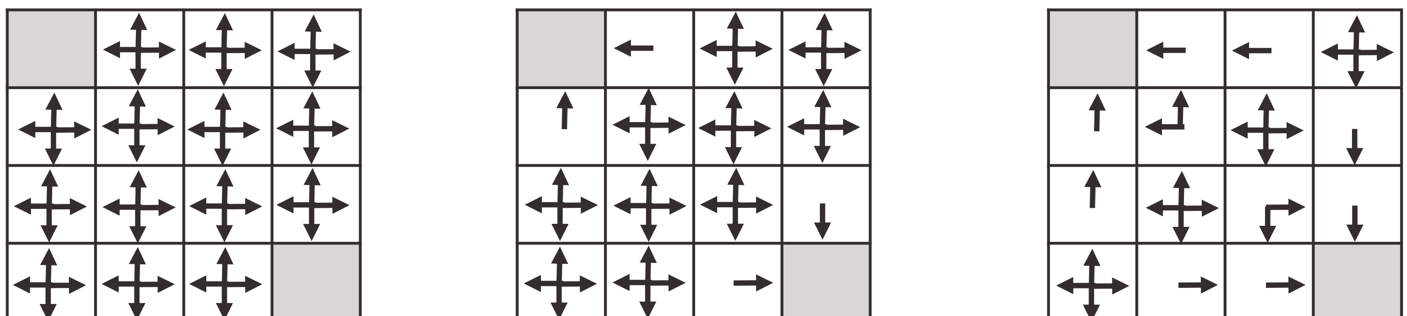
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

greedy policy with respect to v_k



random policy

Example of Evaluating Random Policy

v_k for random policy

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

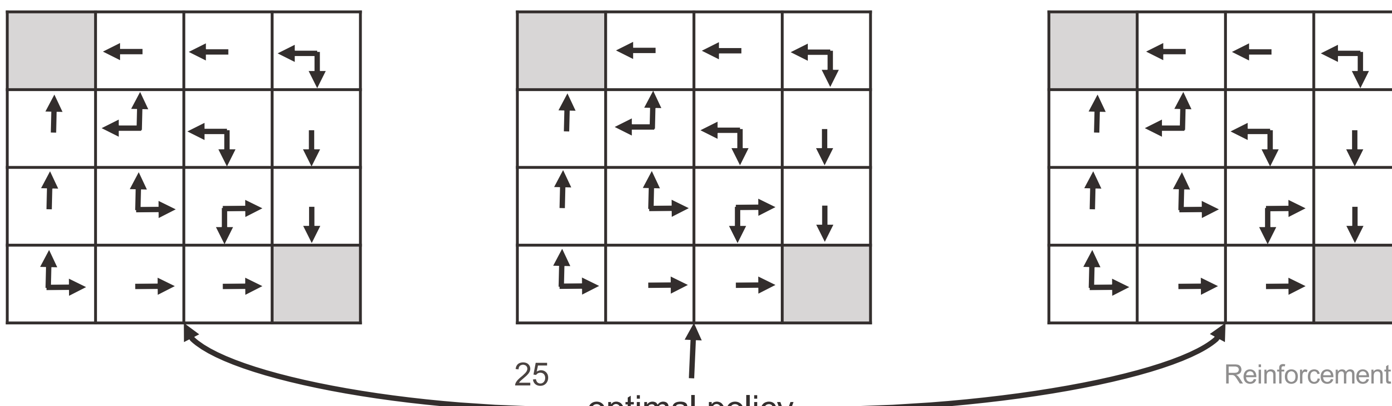
$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

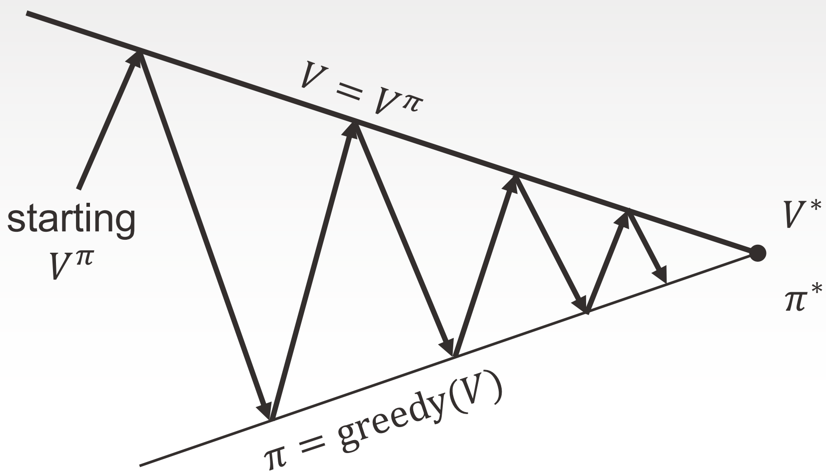
greedy policy with respect to v_k

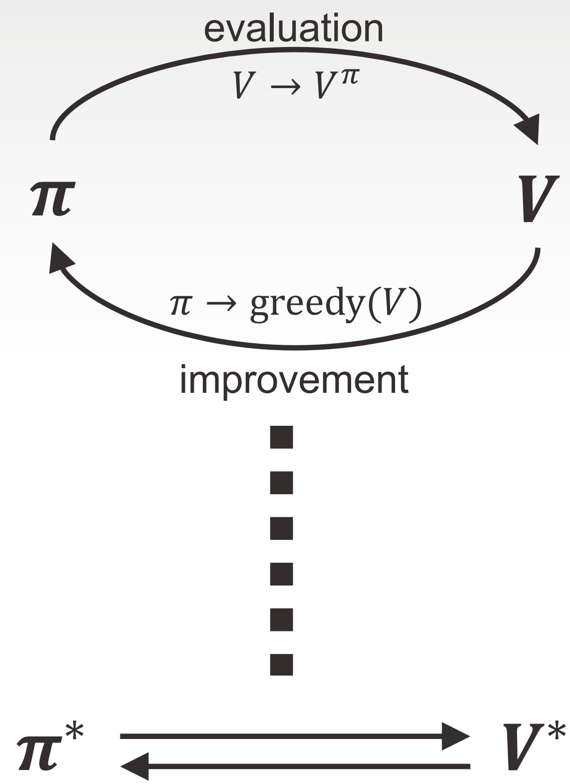


Policy Iteration

- Given a policy π
  - Evaluate policy π
    $$V^{\pi}(s) = \mathbb{E}[r_t + \gamma r_{t+1} + \ \dots | S_t = s]$$
  - Improve the policy by acting greedily with respect to V^{π}
    $$\pi' = \text{greedy}(V^{\pi})$$
- This process of policy iteration always converges to π^* - optimal policy
  - Finite cardinality assumptions

Policy Iteration





Policy Evaluation Estimate v_π

Iterative policy evaluation

Policy Improvement Generate $\pi' \geq \pi$

Greedy policy improvement

Policy Improvement

- Consider a deterministic policy, $a = \pi(s)$
- We can *improve* the policy by acting greedily

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q^{\pi}(s, a)$$

- Improves the value from any state s over one step

$$Q^{\pi}\big(s, \pi'(s)\big) = \max_{a \in \mathcal{A}} Q^{\pi}(s, a) \geq Q^{\pi}\big(s, \pi(s)\big) = V^{\pi}(s)$$

- Improves the value function, $V^{\pi'}(s) \geq V^{\pi}(s)$

$$\begin{aligned}
V^{\pi}(s) \leq Q^{\pi}\big(s, \pi'(s)\big) &= \mathbb{E}_{\pi'}[r_t + \gamma V^{\pi}(S_{t+1}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}\big[r_t + \gamma Q^{\pi}\big(S_{t+1}, \pi'(S_{t+1})\big) \big| S_t = s\big] \\
&\leq \mathbb{E}_{\pi'}[r_t + \gamma r_{t+2} + \gamma^2 Q^{\pi}(S_{t+2}, \pi'(S_{t+2})) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[r_t + \gamma r_{t+2} + \ \ldots \ | S_t = s] = V^{\pi'}(s)
\end{aligned}$$

Policy Improvement

- If improvements stop

$$Q^{\pi}\big(s, \pi'(s)\big) = \max_{a \in \mathcal{A}} Q^{\pi}(s, a) = Q^{\pi}\big(s, \pi(s)\big) = V^{\pi}(s)$$

- Bellman optimality equation has been satisfied

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

- Conclusion $V^{\pi}(s) = V^{*}(s)$ for all $s \in \mathcal{S}$
- π is an optimal policy

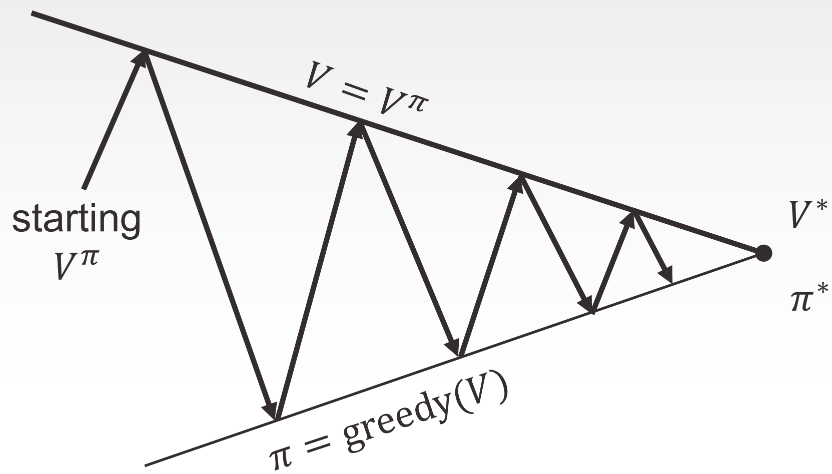
Modified Policy Iteration

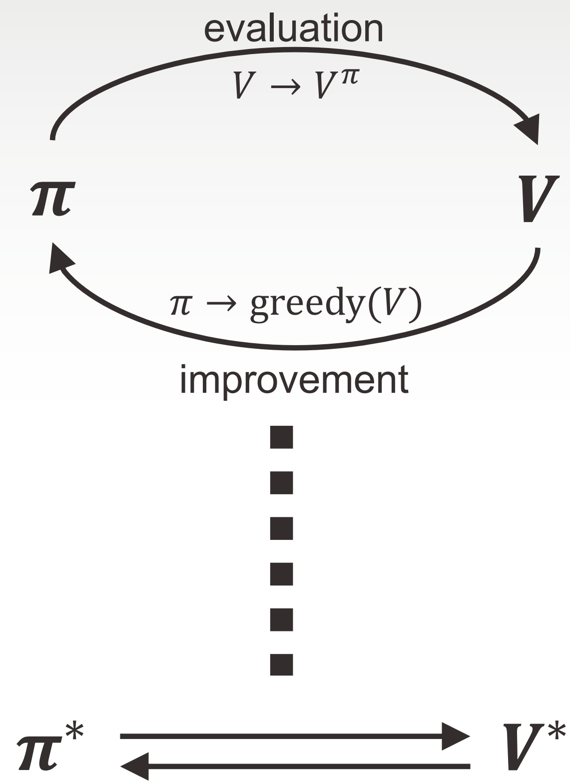
- Stopping criteria
  - ϵ-convergence of value function
  - Stop after K iterations evaluating the policy
- In the example, $K = 3$ was sufficient to achieve optimal policy

Generalized Policy Iteration

- Loop
  - For $k = 0,1,2,\cdots,\mathrm{K}$ // K iterations to evaluate the policy
    - For each possible state s compute
      $$v_{k+1}(s) = E_{\substack{a\sim\pi(a|s)\\ \bar{s}\sim p(\bar{s}|s,a)}}\left[r(s,a) + \gamma v_k(\bar{s}|s)\right]$$
  - Improve the policy by acting greedily with respect to v_{K+1}
    $$\pi' = \mathrm{greedy}(v_{K+1})$$
- The inner loop approximately computes the inverse of the matrix in $\boldsymbol{v}^{\boldsymbol{\pi}} = (\boldsymbol{I} - \gamma\boldsymbol{P}^{\boldsymbol{\pi}})^{-\mathbf{1}}\,\boldsymbol{R}^{\boldsymbol{\pi}}$
- K=0
  - Value iteration

Generalized Policy Iteration





Policy Evaluation Estimate V^{π}
Any policy evaluation algorithm
Policy Improvement Generate $\pi' \geq \pi$
Any policy improvement algorithm

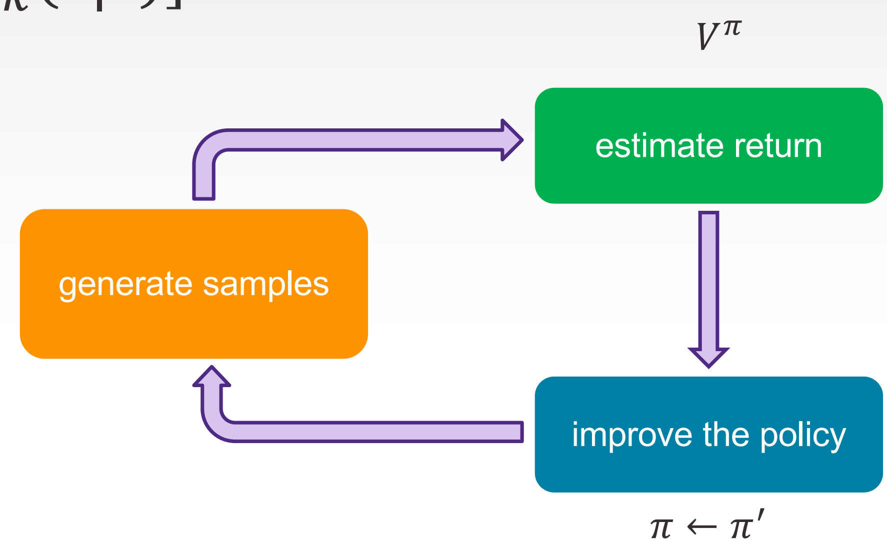
- Value iteration
  - Per iteration time low
  - Needs more iterations
- Policy iteration
  - Per iteration time high
    - Controlled by *K*
  - Needs fewer iterations
  - More flexible

Weaker convergence assumptions for policy iteration

Trade-off

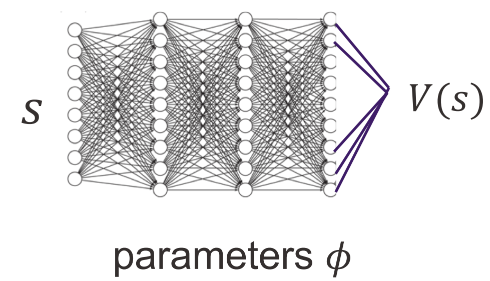
Expectation

- $v_{k+1}(s) = E_{\substack{a \sim \pi(a|s) \\ \bar{s} \sim p(\bar{s}|s,a)}} \left[r(s,a) + \gamma v_k(\bar{s}|s) \right]$
- Number of actions large
  - Sample actions
- For every state s
  - For $n = 1,2,\cdots,N$
    - Sample $a_n \sim \pi(a_n|s)$
    - Sample $s_n' \sim p(s_n'|s, a_n)$
  - $v_{k+1}(s) = \frac{1}{N} \sum_{n=1}^{N} [r(s, a_n) + \gamma v_k(s_n')]$
- In practice *N=1*



Large State Space and Value Iteration

- Tabular value function
  - One big array with one entry per state
  - Not scalable
- Approximate value function
  - Neural network function $V: \mathcal{S} \rightarrow \mathbb{R}$
  - Can be any parametric function
  - No softmax
  - Output a single value



Functional Approximation of V

- Updating the value function approximation
- At state *s*
  - Have reward plus future based on approximate value function
    - Based on optimal action
  - Have value of approximate value function
- Match them
  - L2 loss

fitted value iteration algorithm:

1. set $y_i \leftarrow \max_{a_i}(r(s_i, a_i) + \gamma E[V_\phi(s_i')])$
2. set $\phi \leftarrow \arg\min_\phi \frac{1}{2}\sum_i \|V_\phi(s_i) - y_i\|^2$

Putting it all Together

- For $k = 1,2,\cdots$
  - For $n = 0,1,2,\cdots,N$
    - Sample s_n
  - For $n = 0,1,2,\cdots,N$
    - Compute $y_n = \max_a\left(r(s_n, a) + \gamma E\left[V_\phi(s_n'|s_n)\right]\right)$
  - Set $\phi \leftarrow \operatorname{argmin}_\phi \frac{1}{2}\sum_n \left\|V_\phi(s_n) - y_n\right\|^2$
- Last step
  - Standard gradient optimization by back propagation
- Drawback
  - No policy improvement

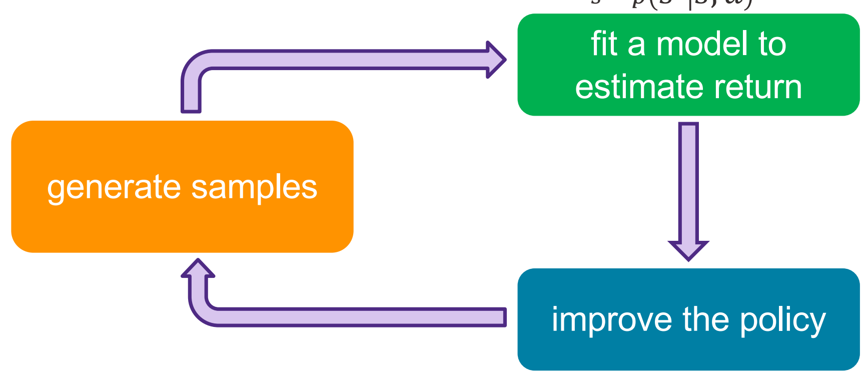
s_n

$$\underbrace{r(s_n, a) + \quad \gamma E\left[V_\phi(s_n'|s_n)\right]}_{V_\phi(s_n)}$$

Revised

- Value function as neural network
- Keep track of Q-factor at samples generated
- Compute $Q^\pi(s,a)$ at generated samples
  - Samples generated based on incumbent policy
  - Use relationship between *Q* and *V*
- Update policy in greedy manner from *Q*

$$Q^\pi(s,a) \leftarrow r(s,a) + \gamma E_{s' \sim p(s'|s,a)}[V^\pi(s')]$$



$$V^\pi(s) \leftarrow \max_a Q^\pi(s,a)$$

Algorithm

- For $k = 1,2,\cdots$
  - For $n = 0,1,2,\cdots,N$
    - Sample $s_n, a_n \sim \pi(a_n|s_n)$
  - For $n = 0,1,2,\cdots,N$
    - $Q^\pi(s_n, a_n) \leftarrow r(s_n, a_n) + \gamma E_{s' \sim p(s'|s_n, a_n)}[V_\phi(s')]$
    - Compute $y_n = \max_a (r(s_n, a) + \gamma E_{s' \sim p(s'|s_n, a_n)}[V_\phi(s')])$
  - Set $\phi \leftarrow \operatorname{argmin}_\phi \frac{1}{2} \sum_n \left\| V_\phi(s_n) - y_n \right\|^2$
  - $\pi(s) \leftarrow \operatorname{argmax}_a Q^\pi(s, a)$

Issues

- How to generate samples from state space?
  - Uniformly at random – unclear for large state space
- Episodes present
  - Sample from episodes
  - Never sample a new state
    - Yes as next state
- Why not parametric policy?
  - Update of parameters not clear
  - Needs further tricks

Unknown Transition Function

- Observe only episodes – imitation learning
- Functional approximation of transition function not possible
  - Output state

fitted value iteration algorithm:

1. set $y_i \leftarrow \underline{\max_{a_i}(r(s_i, a_i) + \gamma E[V_\phi(s_i')])}$
2. set $\phi \leftarrow \arg\min_\phi \frac{1}{2}\sum_i \|V_\phi(s_i) - y_i\|^2$

Need to know outcomes for different actions!

policy iteration algorithm

1. evaluate $Q^\pi(s, a)$
2. set $\pi \leftarrow \pi'$

$$\pi'(a_t|s_t) = \begin{cases} 1 & \text{if } a_t = \arg\max_{a_t} Q^\pi(s_t, a_t) \\ 0 & \text{otherwise} \end{cases}$$

policy evaluation:

$$\cancel{V^\pi(s) \leftarrow r(s, \pi(s)) + \gamma E_{s' \sim p(s'|s, \pi(s))}[V^\pi(s')]}$$

$$Q^\pi(s, a) \leftarrow r(s, a) + \gamma E_{s' \sim p(s'|s, \pi(s))}[Q^\pi(s', \pi(s'))]$$

Functional Approximation of Q

policy iteration:

1. evaluate $V^\pi(s)$
2. set $\pi \leftarrow \pi'$

fitted value iteration algorithm:

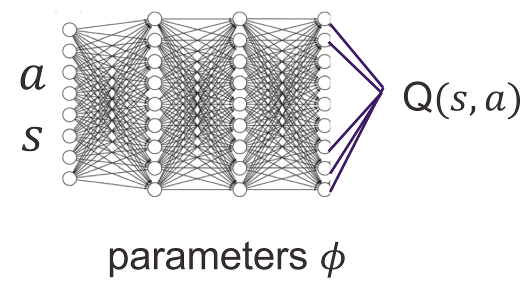
1. set $y_i \leftarrow \max_{a_i}(r(s_i, a_i) + \gamma E[V_\phi(s_i')])$
2. set $\phi \leftarrow \arg\min_\phi \frac{1}{2}\sum_i \|V_\phi(s_i) - y_i\|^2$

fitted Q iteration algorithm:

1. set $y_i \leftarrow r(s_i, a_i) + \gamma E[V_\phi(s_i')]$
2. set $\phi \leftarrow \arg\min_\phi \frac{1}{2}\sum_i \|Q_\phi(s_i, a_i) - y_i\|^2$

approximate $E[V(s_i')] \approx \max_{a'} Q_\phi(s_i', a_i')$

Doesn't require simulation of actions!



Value Iteration with Fitted Q-factor

- Observed data are trajectories
  - Formally, $\mathrm{U} = \{(s_i, a_i, s_i', r_i) | i \in N\}$
- Loop
  - Sample $S \subseteq U$
  - For $i \in S$
    - $y_i \leftarrow r_i + \gamma \max_{a_i'} Q_\phi(s_i', a_i')$
  - Set $\phi \leftarrow \operatorname{argmin}_\phi \frac{1}{2} \sum_{i \in S} \left\| Q_\phi(s_i, a_i) - y_i \right\|^2$
- Only remaining drawback how to compute max

Analysis

full fitted Q-iteration algorithm:

1. dataset $\{(s_i, a_i, s_i', r_i)\}$
2. set $y_i \leftarrow r(s_i, a_i) + \gamma \max_{a_i'} Q_\phi(s_i', a_i')$ this max improves the policy (tabular case)
3. set $\phi \leftarrow \text{argmin}_\phi \frac{1}{2}\sum_i \left\| Q_\phi(s_i, a_i) - y_i \right\|^2$

error $\mathcal{E}$

$$\mathcal{E} = \frac{1}{2} E_{(s,a)\sim\beta}\left[Q_\phi(s,a) - \left[r(s,a) + \gamma \max_{a'} Q_\phi(s', a')\right]\right]$$

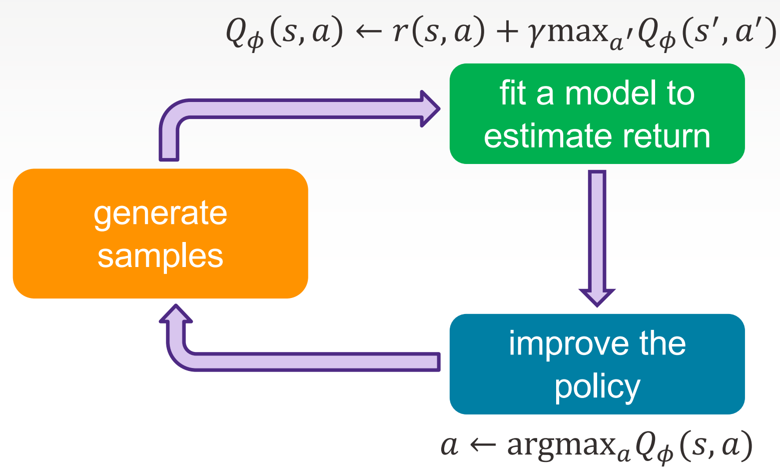
- If $\mathcal{E} = 0$, then $Q_\phi(s,a) = r(s,a) + \gamma \max_{a'} Q_\phi(s', a')$
  - This is an *optimal* Q-function, corresponding to optimal policy π^*:

$$\pi^*(a_t|s_t) = \begin{cases} 1 & \text{if } a_t = \text{argmax}_{a_t} Q_\phi(s_t, a_t) \\ 0 & \text{otherwise} \end{cases}$$

Online Version

full fitted Q-iteration algorithm:

1. collect dataset $\{(s_i, a_i, s_i', r_i)\}$ using some policy
2. set $y_i \leftarrow r(s_i, a_i) + \gamma \max_{a_i'} Q_\phi(s_i', a_i')$
3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \left\| Q_\phi(s_i, a_i) - y_i \right\|^2$



online Q iteration algorithm:

1. take some action a_i and observe (s_i, a_i, s_i', r_i)
2. $y_i = r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s_i', a_i')$
3. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s_i, a_i)(Q_\phi(s_i, a_i) - y_i)$