BASIC ALGORITHMS

Algorithms for small problems

Diego Klabjan
Professor, Industrial Engineering and Management Sciences

Northwestern | McCORMICK SCHOOL OF ENGINEERING

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

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

  - 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

VALUE ITERATION

Value Iteration

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

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
- $v_1 \rightarrow v_2 \rightarrow \ \dots \rightarrow v_\pi$
- 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

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
- $\boldsymbol{R}^{\boldsymbol{\pi}} = \left(E_{a\sim\pi(a|s)} r(s,a) \right)_s = \left(\sum_{a\in\mathcal{A}} \pi(a|s)\, r(s,a) \right)_s$
- $\boldsymbol{P}^{\boldsymbol{\pi}} = \left(\sum_{a\in\mathcal{A}} \pi(a|s) P_{ss'}^{a} \right)_{s,s'}$

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 Algorithm

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

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

  - Let v^π be the converged function
  - For each possible state s compute

$$\pi'(s) = \max_a r(s,a) + \gamma E_{\bar{s}\sim p(\bar{s}|s,a)} v^\pi(\bar{s}|s)$$

  - Set $\pi = \pi'$

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 $\boldsymbol{v}^{\boldsymbol{\pi}} = (\boldsymbol{I} - \gamma\boldsymbol{P}^{\boldsymbol{\pi}})^{-1}\,\boldsymbol{R}^{\boldsymbol{\pi}}$
- K=0
  - Value iteration

- 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