Today

Modified Policy Iteration

→ Uses ideas from value iteration and policy iteration

Review: Value Iteration

Value iteration exploits the contraction mapping property to solve infinite horizon MDPs

Value Iteration:

1. Select some intial vector v^0 *and set* $n = 0$.
2. *Apply* $v^{n+1} = Lv^n$
3. *If* $||v^{n+1} - v^n|| < \epsilon(1 - \lambda)/2\lambda$ *go to step 4. Otherwise n=n+1 and return to step 2.*
4. $d_\epsilon(s) \in argmax_{a \in A} \{\mathrm{r(s,a)} + \lambda \sum_{j \in S} p(j|s,a)v^n(j)\}, for\ all\ s \in S.$

Review: Policy Iteration

MICHIGAN ENGINEERING
UNIVERSITY of MICHIGAN ■ COLLEGE of ENGINEERING

Algorithm (Policy Iteration):

1. Set $n = 0$ and select an arbitrary decision rule $d_0 \in D$.

2. ***Policy Evaluation***: *Obtain* v^n *by solving* $(I - \lambda P_{d_n})v = r_{d_n}$

3. ***Policy Improvement***: *Choose* d_{n+1} *to satisfy*:

$$d_{n+1} \in argmax_{d \in D}\{r_d + \lambda P_d v^n\},$$

   setting $d_{n+1} = d_n$ if possible.

4. If $d_{n+1} = d_n$, stop and set $d^* = d_n$. Otherwise $n = n + 1$ and return to step 2.

Modified Policy Iteration

The policy evaluation step of the policy iteration algorithm requires solution of the linear system:

$$\left(\mathrm{I} - \lambda P_{d_n}\right) v = r_{d_n}$$

Repeated solution for large M can be computationally prohibitive

Modified policy iteration avoids this by using **inexact solutions** at the evaluation step

Modified Policy Iteration

Basic Idea:

- Instead of solving $\left(\mathrm{I} - \lambda P_{d_n}\right)v = r_{d_n}$ exactly to evaluate the policy d_n at each iteration, use **value iteration** to find an approximate solution.
- ϵ defines a limit to how accurate the final solution needs to be
- m_n defines how many iterations of value iteration to perform at iteration n

Modified Policy Iteration

Algorithm (Modified Policy Iteration):

1. Set $n = 0$, select arbitrary v^0, and specify $\epsilon > 0$ and a sequence of nonegative integers $\{m_n\}$.
2. Policy Improvement: Choose d_{n+1} to satisfy:

$$d_{n+1} \in argmax_{d \in D}\{r_d + \lambda P_d v^n\},$$

   setting $d_{n+1} == d_n$ if possible.

3. Partial Policy Evaluation:

   a. Set $k = 0$ and $v_n^0 = \max_{d \in D}\{r_d + \lambda P_d v^n\}$.

   b. If $||v_n^k - v^n|| < \epsilon(1-\lambda)/2\lambda$, go to step 4. Otherwise go to (c).

   c. If $k == m_n$, go to (e). Otherwise, compute v_n^k by

$$v_n^{k+1} = L_{d_{n+1}} v_n^k$$

   d. Set $k = k + 1$ and return to (c)

   e. Set $v^{n+1} = v_n^{m_n}$, set n=n+1, and go to Step 2

4. Set $d_\epsilon = d_{n+1}$ and stop

Apply value iteration for m_n iterations

***See example 6.5.1 on p. 187 of Puterman

Modified Policy Iteration

There are many ways to choose sequence $\{\mathrm{m_n}\}$

- Fixed for all iterations ($\mathrm{m_n} = m$)
- Chosen according to a pattern (e.g. $\mathrm{m_n}$ increasing in n)
- Selected adaptively; for example, requiring $||\mathrm{v_n^k} - v^n||<$ ϵ_n where ϵ_n is either fixed or variable