Value and Policy Iteration

Andrea Danyluk
February 27, 2017

Announcements

• Programming Assignment 2 in progress
• On Wednesday will announce an article to read for Monday

Today’s Lecture

• Quick review of Value Iteration
• Policy Iteration

Stochastic Gridworld

Policies, not Plans

Value Iteration

• Will calculate successive estimates $V_k^*$ of $V^*$
• Start with $V_0^*(s) = 0$ for all $s$
• Given $V_i^*$, calculate the values for all states for depth $i+1$
  \[ V_{i+1}^*(s) = \max_a \sum P(s' | s,a) \cdot [R(s') + \gamma \cdot V_i^*(s')] \]
• Throw out old vector $V_i^*$
• Repeat until convergence
• Called value update or Bellman update

[Adapted from CS 188 Berkeley]
Value Iteration Demos

• All rewards are 1
• The value of a state is either the value itself or the value + the penalty if you got there by running into a wall (so in this case we aim to minimize expected “reward”)
• PJOG = how badly you go off course
  – 0 means your action does what you intended
  – 0.3 means 70% of the time your action does what’s intended; splits the 30% evenly among the remaining options
• Discount rate (gamma) is always 1

Things to notice in the demos

• Value approximations get refined toward optimal values
• Information propagates outward from the terminal states until all states have correct information
• The policy may converge long before the values do

Value Iteration Demos

• All base rewards are 1
• The reward at a state is either the reward itself or the reward + the penalty if you got there by running into a wall (so in this case we aim to minimize expected value)
• PJOG = how badly you go off course
  – 0 means your action does what you intended
  – 0.3 means 70% of the time your action does what’s intended; splits the 30% evenly among the remaining options
• Discount rate (gamma) is always 1

The Bellman Equation: a closer look

\[ V^*(s) = \max_a \sum P(s' | s, a) \left[ R(s, a, s') + \gamma V^*(s') \right] \]

Reconciling the formulations in the two texts:

Sutton and Barto:
\[ V^*(s) = \max_a \sum P(s' | s, a) \left[ R(s, a, s') + \gamma V^*(s') \right] \]
We’ve been taking the reward of the transition to be the reward of the state we would enter upon transition

Russell and Norvig:
\[ V^*(s) = R(s) + \max_a \sum P(s' | s, a) \left[ \gamma V^*(s') \right] \]
A common formulation: take the reward of the transition to be the one of the state you’re in

Values (Utilities) for Fixed Policies

• How do we compute the utility of state under a fixed (not necessarily optimal) policy?
\[ V^\pi(s) = \sum P(s' | s, \pi(s)) \left[ R(s, \pi(s), s') + \gamma V^\pi(s') \right] \]
where the sum is over all \( s' \)
This is the expected total discounted reward starting in \( s \) and following the policy

Policy Evaluation

• Can calculate the V's for a fixed policy just as we calculated \( V^* \) earlier
• Set values to 0 initially
• Perform recursive update
\[ V_{\pi}^i(s) = \sum P(s' | s, \pi(s)) \left[ R(s, \pi(s), s') + \gamma V_{\pi}^{i-1}(s') \right] \]
where the sum is over all \( s' \)
Note: No “max” here. So this is just a set of linear equations that can be solved without recursive update.
Policy Iteration

Repeat
• Step 1: Policy evaluation
  – Calculate utilities for fixed (probably suboptimal) policy until convergence (in practice, a reasonable approximation is good enough)
• Step 2: Policy improvement
  – Update policy using one-step lookahead
Until policy converges

Reinforcement Learning

• Assume an MDP
  – $S$: a set of states
  – $A$: a set of actions
  – $P(s' | s, a)$: the probability of ending up in state $s'$, given that the agent is in state $s$ and takes action $a$
  – $R(s)$: or $R(s, a, s')$: a reward function
  – Want to find a policy $\pi$
• But this time we don’t know $P$ or $R$
  – Need to try things out in order to learn

Reinforcement Learning

Assume an MDP
– $S$: a set of states
– $A$: a set of actions
– $P(s' | s, a)$: the probability of ending up in state $s'$, given that the agent is in state $s$ and takes action $a$
– $R(s)$: or $R(s, a, s')$: a reward function
– Want to find a policy $\pi$

Passive RL
• Given:
  – A policy $\pi(s)$
  – No knowledge of $P(s' | s, a)$
  – No knowledge of rewards $R(s, a, s')$
• Goal: learn state values (not policy yet...)
  – Recall policy evaluation!
• Passive in the sense that there’s no choice about what actions to take
  – Need to execute the policy to learn from experience
  – Not offline planning. Actually take actions to learn.

Example: Direct Estimation

Episodes:

\begin{align*}
(1, 1) & -1, (1, 2) & -1, (1, 3) & -1, (2, 2) & -1, (2, 3) & -1, (3, 2) & -1, (3, 3) & -1, (4, 3) & +100 \\
(1, 1) & -1, (1, 2) & -1, (1, 3) & -1, (2, 2) & -1, (2, 3) & -1, (3, 2) & -1, (3, 3) & -1, (4, 2) & -100 \\
\end{align*}

$V(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(S_{t+1}) \right]$, $s = S_0$

$V(2, 3) = \frac{(96 - 103)}{2} = -3.5$
$V(3, 3) = \frac{(99 + 97 - 102)}{3} = 31.3$