Reinforcement Learning: Temporal Difference

Andrea Danyluk
March 1, 2017

Announcements

- Programming Assignment 2 due tomorrow at 11pm
  - If working with a partner, send me email with your names and indicate who is turning it in
  - On Friday will make code review sign-up sheet available
- Assignment for Monday
  - Read Holte et al.'s AAAI 2016 paper on bi-directional search
  - Turn in brief reading response (no more than one page, 12pt font, 1.5 spacing) at start of class
- Sample midterm available online
- RL assignment will be available Friday morning
  - Will ask you to confirm partners with me by Monday

Today's Lecture

- Reinforcement Learning
- But a note on Policy Iteration first

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 π
- But this time we don't know P or R
  - Need to try things out in order to learn

Passive RL

- Given:
  - A policy $\pi(s)$ (can begin with a random policy)
  - No knowledge of P(s'|s, a)
  - No knowledge of rewards R(s, a, s')
- Goal: learn state values (or state,action values)
  - But can learn policy with exploring starts and generalized policy iteration
- 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.

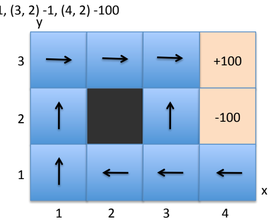
Example: Direct Estimation

Episodes:

(1, 1) -1 , (1, 2) -1 , (1, 2) -1 , (1, 3) -1, (2, 3) -1, (3, 3) -1, (3, 2) -1, (3, 3) -1, (4, 3) +100

(1, 1) -1, (1, 2) -1, (1, 3) -1, (2, 3) -1, (3, 3) -1, (3, 2) -1, (4, 2) -100

$V(s) = E\,[\Sigma\, \gamma^t R(S_{t+1})]$, t from 0 to ∞, $s = S_0$



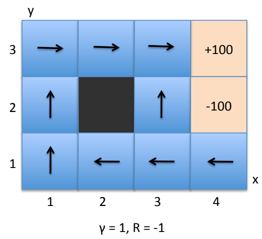
$\gamma = 1$, R = -1

V(2, 3) = (96 + -103) / 2 = -3.5

V(3, 3) = (99 + 97 + -102) / 3 = 31.3

Example: Direct Estimation

$V(s) = E\,[\Sigma\, \gamma^t R(S_{t+1})]$,
t from 0 to ∞
$s = S_0$



Every-visit Monte Carlo method for estimating V^π

Can also have first-visit MC method to do the same.

Note that all learning happens *at the end* of an episode.

Model-Based Learning

- Count outcomes for each s, a
- Normalize to give estimate of P(s' | s, a)
- Discover R(s) [or R(s, a, s')] when exploring
- Solve the MDP with the learned model as if it were correct
  - Use Policy Iteration, for example

Example: Model-Based Learning

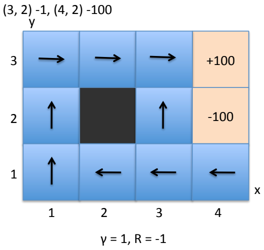
Episodes:

(1, 1) -1 , (1, 2) -1 , (1, 2) -1 , (1, 3) -1, (2, 3) -1, (3, 3) -1, (3, 2) -1, (3, 3) -1, (4, 3) +100

(1, 1) -1, (1, 2) -1, (1, 3) -1, (2, 3) -1, (3, 3) -1, (3, 2) -1, (4, 2) -100

P((1,2) | (1, 1), North) = 1

P((4, 3) | (3, 3), Right) = 1/3
P((3, 2) | (3, 3), Right) = 2/3



Model-Based vs Model-Free Learning

- Want to compute an expectation weighted by P(x):
  $E[f(x)] = \Sigma_x P(x) f(x)$, i.e.,
  $V(s) = E\,[\Sigma\, \gamma^t R(S_{t+1})]$
  $V^\pi(s) = \Sigma_{s'} P(s' \mid s,\pi(s))\cdot[R(s, \pi(s),s') + \gamma\cdot V^\pi(s')]$
- Model-Based: estimate P(x) from samples, and then compute expectation
  $P(x) = num(x)/N$, i.e.,
  the number of times s' is reached from s on action a, divided by the number of times a is applied in s
- Model-Free: estimate expectation directly from samples
  $E[f(x)] = 1/N\, \Sigma_i f(x_i)$, i.e.,
  $V^\pi(s) = 1/N\, \Sigma_{s'} num(s,a,s') \cdot [R(s, \pi(s),s') + \gamma\cdot V^\pi(s')]$, where N is the number of times s is reached, and a is π(s)
- That is, the samples appear with the right frequencies.

Temporal Difference Learning

- Learn from every experience: don't have to wait for an episode to end
  - Update V(s) each time we experience (s, a, s', r')
  - Likely s' will contribute updates more often
- Policy is still fixed
- Moves a state's value toward the value of whatever successor occurs: running average

Temporal Difference Learning

$V^\pi(s) = V^\pi(s) + \alpha(\textit{sample} - V^\pi(s))$

Note: V on right hand side is old value. V on left hand side is new. (Like an assignment statement.)

- Get sample of V(s):
  $\textit{sample} = R(s, \pi(s),s') + \gamma\cdot V^\pi(s')$
- Update V(s):
  $V^\pi(s) = (1-\alpha)\, V^\pi(s) + \alpha(\textit{sample})$

Exponential Moving Average

- $V^{\pi}(s) = (1-\alpha)\ V^{\pi}(s) + \alpha(\textit{sample})$
- Let

  $V_k^{\pi}(s)$ be the kth estimate of $V^{\pi}(s)$

  $V_k^{\pi}(s)$ be the kth sample
- Then

  $V_k^{\pi}(s) = \alpha(V_k^{\pi}(s)) + (1-\alpha)\ V_{k-1}^{\pi}(s)$

  $= \alpha(V_k^{\pi}(s)) + (1-\alpha)\ [\alpha(V_{k-1}^{\pi}(s)) + (1-\alpha)\ V_{k-2}^{\pi}(s)]$

  $= \ldots$
- Since $\alpha < 1$, older estimates get less and less weight as time goes on
- α is called the *learning rate*