Reinforcement learning: Learning rules

Monday, February 15, 2016 4:26 PM

Descibe chapter 6.1-6.5 in Sutton.
Don't go into on/off-policy stuff, Alex wil do that
Ari will talk about eligibility traces

TD learning

- Combination of MC and dynamic programming ideas
- Focus on policy evaluation: given a policy, what is my expected payoff at each state in the world?
- Monte Carlo: The value of the state is updated using the "actual return" after time t (i.e. the return that was achieved at the end of the episode)

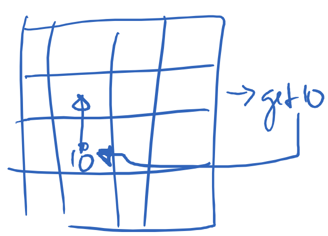
  TD: The value is updated at every time step, using the predicted return based on the stored estimates of the values at other states.

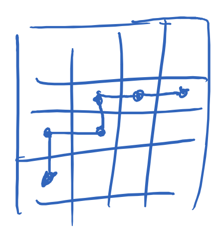
How to find value function? (policy evaluation)

- TD learning procedure:

Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0, \forall s \in \mathcal{S}^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

This updates at every step





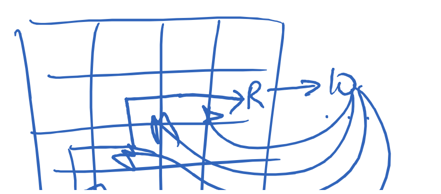
. . .

Compare with MC:

Initialize:
  $\pi \leftarrow$ policy to be evaluated
  $V \leftarrow$ an arbitrary state-value function
  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:
  Generate an episode using π — full episode first
  For each state s appearing in the episode: — then update
    $G \leftarrow$ return following the first occurrence of s
    Append G to $Returns(s)$
    $V(s) \leftarrow$ average($Returns(s)$)

This does one episode and stores the results:



Generate an episode using π
For each state s appearing in the episode:
  $G \leftarrow$ return following the first occurrence of s
  Append G to $Returns(s)$
  $V(s) \leftarrow$ average($Returns(s)$)

then up

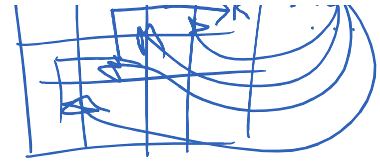


Figure 5.1: The first-visit MC method for estimating v_π.

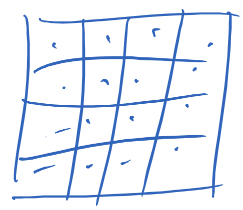
And dynamic programming:

(Big disadvantage here: You need a model of the world (i.e. know all states & transitions))

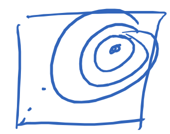
Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Figure 4.1: Iterative policy evaluation.

This evaluates all states at each iteration and stores the results:



→ values propagate back



Brute force

4	4	8	8
4	5	9	10
4	4	8	8
3	3	7	7

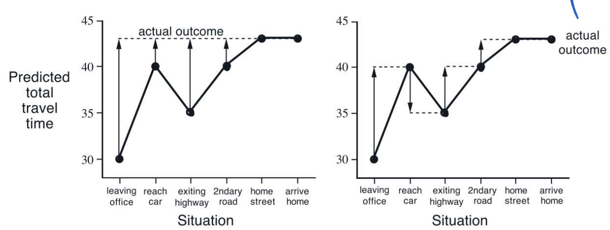


Figure 6.2: Changes recommended in the driving home example by Monte Carlo methods (left) and TD methods (right).

↳ MC and TD differ in when, or how often, you learn, i.e. assimilate experience into previous knowledge.

Driving example: When you get stuck behind a truck (i.e. visit a state that you know already to predict a lower reward) you can update your estimate for the reward at the current state now. You don't have to wait until you know the actual final outcome.

(MC and TD will differ, e.g., at the state "just before reaching home".

In TD, your estimate for that state

will differ, depending on whether you visited the state "getting stuck in traffic" beforehand.

In MC, you wouldn't update any values until you actually reach home.

... not sure if the analogy is perfect, because those two paths should be different states and thus differ even in MC... they just don't get updated as frequently.

Practical Advantages of TD over MC:

- fully "on-line", incremental
- Faster in tasks with long episodes
- makes sense for continuous (no episodes) tasks
- Faster in general (but not provenly so)

Optimality

MC: optimal : min sqr. error for training rewards

TD: optimal : finds ML solution given the Markov Model of the system

What is the difference?

Example:

Imagine you observe the following episodes and have to find the value function:

$A=0 \rightarrow B=0$ $B=1$

$B=1$ $B=1$

$B=1$ $B=1$

$B=1$ $\quad$ $B=1$

$B=1$ $\quad$ $B=1$

$B=1$ $\quad$ $B=0$

Value of state B: $6/8 = 3/4$

Value of State A:

Two views:

1. MC: Seen A once, got 0
   $\longrightarrow A=0$
2. TD: Seen A once, got to B, $B=3/4$ so $A=3/4$.

#1 is optimal in predicting rewards on training set

#2 is optimal under Markov model

$\Longrightarrow$ TD learning has markov baked in

$\Longrightarrow$ For ML, where problems can be specified as markovian, TD is clearly better

How about Neuroscience?

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

space for max (Q)

Figure 6.5: Sarsa: An on-policy TD control algorithm.

$S_t \quad A_t \quad R_{t+1} \quad S_{t+1} \quad A_{t+1}$

Init Q, S, A arbit.

Repeat (each epis.)

  Init S
  Choose A for S using π based on Q
  Repeat (each step in epis.)
    Do A, get R, S'
    Choose A'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S terminal.