Reinforcement Learning I: Temporal Differences

Hal Daumé III

Computer Science
University of Maryland

me@hal3.name

CS 421: Introduction to Artificial Intelligence

23 Feb 2012

Many slides courtesy of Dan Klein, Stuart Russell, or Andrew Moore
Announcements

➢ None...
Survey Results

- Pace:
- Cvg:
- HW:
- P1:
- P2:
Reinforcement Learning

- Reinforcement learning:
  - Still have an MDP:
    - A set of states $s \in S$
    - A set of actions (per state) $A$
    - A model $T(s,a,s')$
    - A reward function $R(s,a,s')$
  - Still looking for a policy $\pi(s)$

- New twist: don’t know $T$ or $R$
  - I.e. don’t know which states are good or what the actions do
  - Must actually try actions and states out to learn
Example: Animal Learning

- RL studied experimentally for more than 60 years in psychology
  - Rewards: food, pain, hunger, drugs, etc.
  - Mechanisms and sophistication debated

- Example: foraging
  - Bees learn near-optimal foraging plan in field of artificial flowers with controlled nectar supplies
  - Bees have a direct neural connection from nectar intake measurement to motor planning area
Example: Backgammon

- Reward only for win / loss in terminal states, zero otherwise
- TD-Gammon learns a function approximation to $V(s)$ using a neural network
- Combined with depth 3 search, one of the top 3 players in the world
- You could imagine training Pacman this way…
- … but it’s tricky!
Passive Learning

- **Simplified task**
  - You don’t know the transitions $T(s,a,s')$
  - You don’t know the rewards $R(s,a,s')$
  - You are given a policy $\pi(s)$
  - **Goal: learn the state values** (and maybe the model)

- **In this case:**
  - No choice about what actions to take
  - Just execute the policy and learn from experience
  - We’ll get to the general case soon
Example: Direct Estimation

 Episodes:

- (1,1) up -1
- (1,2) up -1
- (1,2) up -1
- (1,3) right -1
- (2,3) right -1
- (2,3) right -1
- (3,3) right -1
- (3,2) up -1
- (3,3) right -1
- (4,3) exit +100

\[
\begin{align*}
U(1,1) & \sim \frac{92 + -106}{2} = -7 \\
U(3,3) & \sim \frac{99 + 97 + -102}{3} = 31.3
\end{align*}
\]
Model-Based Learning

➢ In general, want to learn the optimal policy, not evaluate a fixed policy

➢ Idea: adaptive dynamic programming
  ➢ Learn an initial model of the environment:
  ➢ Solve for the optimal policy for this model (value or policy iteration)
  ➢ Refine model through experience and repeat
  ➢ Crucial: we have to make sure we actually learn about all of the model
Model-Based Learning

- Idea:
  - Learn the model empirically (rather than values)
  - Solve the MDP as if the learned model were correct

- Empirical model learning
  - Simplest case:
    - Count outcomes for each s,a
    - Normalize to give estimate of $T(s,a,s')$
    - Discover $R(s,a,s')$ the first time we experience $(s,a,s')$

- More complex learners are possible (e.g. if we know that all squares have related action outcomes, e.g. “stationary noise”)

Example: Model-Based Learning

- **Episodes:**
  - (1,1) up -1
  - (1,2) up -1
  - (1,2) up -1
  - (1,3) right -1
  - (2,3) right -1
  - (3,3) right -1
  - (3,3) right -1
  - (3,2) up -1
  - (4,2) exit -100
  - (3,3) right -1
  - (done)

\[
T(<3,3>, \text{right}, <4,3>) = 1 / 3
\]

\[
T(<2,3>, \text{right}, <3,3>) = 2 / 2
\]
Imagine we find the lower path to the good exit first.

Some states will never be visited following this policy from (1,1).

We’ll keep re-using this policy because following it never collects the regions of the model we need to learn the optimal policy.
What Went Wrong?

- Problem with following optimal policy for current model:
  - Never learn about better regions of the space if current policy neglects them

- Fundamental tradeoff: exploration vs. exploitation
  - Exploration: must take actions with suboptimal estimates to discover new rewards and increase eventual utility
  - Exploitation: once the true optimal policy is learned, exploration reduces utility
  - Systems must explore in the beginning and exploit in the limit
Model-Free Learning

- Big idea: why bother learning T?
  - Update V each time we experience a transition
  - Frequent outcomes will contribute more updates (over time)
- Temporal difference learning (TD)
  - Policy still fixed!
  - Move values toward value of whatever successor occurs

\[
V^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, a, s') + \gamma V^\pi(s')] 
\]

\[
sample = R(s, a, s') + \gamma V^\pi(s')
\]

\[
V^\pi(s) \leftarrow V^\pi(s) + \alpha(ssample - V^\pi(s))
\]
Example: Passive TD

\[
V^\pi(s) \leftarrow V^\pi(s) + \alpha \left[ R(s, a, s') + \gamma V^\pi(s') - V^\pi(s) \right]
\]

(1,1) up -1
(1,2) up -1
(1,2) up -1
(1,3) right -1
(2,3) right -1
(2,3) right -1
(3,3) right -1
(3,2) up -1
(3,3) right -1
(4,3) exit +100

Take \( \gamma = 1, \alpha = 0.5 \)
 TD value leaning is model-free for policy evaluation

然而，如果我们想将我们的价值估计转换成策略，我们就完了：

- Idea: learn Q-values directly
- Makes action selection model-free too!

\[
\pi(s) = \arg \max_a Q^*(s, a)
\]

\[
Q^*(s, a) = \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V^*(s') \right]
\]