MCTS for MDPs

3/7/18

Real-Time Dynamic Programming

Repeat while there's time remaining:

- state ← start state
- repeat until terminal (or depth bound):
  - action ← optimal action in current state
  - V(state) ← R(state) + discount * Q(state, action)
    - Q(state, action) calculated from $V(s')$ for all reachable s'. If s' hasn't been seen before, initialize $V(s') \leftarrow h(s')$.
  - state ← result of taking action

RTDP does rollouts and backprop

Rollouts:

- Repeatedly select actions until terminal.

Backpropagation:

- Update policy/value for visited states/actions.

It’s not doing either of these things particularly well:

- Greedy action selection means no exploration.
- Updating every state means lots of storage.

MCTS is a better version of the same thing!

MCTS Review

- Selection
  - Runs in the already-explored part of the state space.
  - Choose a random action, according to UCB weights.
- Expansion
  - When we first encounter something unexplored.
  - Chose an unexplored action uniformly at random.
- Simulation
  - After we've left the known region.
  - Select actions randomly according to the default policy.
- Backpropagation
  - Update values for states visited in selection/expansion.
  - Average previous values with value on current rollout.

Differences from game-playing MCTS

- Learning state/action values instead of state values.
- The next state is non-deterministic.
- Simulation may never reach a terminal state.
- There is no longer a tree-structure to the states.
- Non-terminal states can have rewards.
- Rewards in the future need to be discounted.

Online vs. Offline Planning

Offline: do a bunch of thinking before you start to figure out a complete plan.

Online: do a little bit of thinking to come up with the next (few) action(s), then do more planning later.

Are RTDP and MCTS online or offline planners?

…or both?

UCB Exploration Policy

Formula from today's reading:

$$Policy(s) = \arg\min_{a \in A(s)} \left\{ \hat{Q}(s,a) - C\sqrt{\frac{\ln(n_s)}{n_{s,a}}} \right\}$$

visits to state s

trials of action a in state s

- We now need to track visits for each state/action.

How does this differ from the UCB formula we saw two weeks ago?

MCTS Backprop for MDPs

- Track all rewards experienced during rollout.
- At the end of a rollout, update Q-values for the states/actions experienced during that rollout.
- Also update visits.

Update: $$Q(s,a) \leftarrow \frac{R_{\geq T} + (n_{s,a} - 1)Q(s,a)}{n_{s,a}}$$

$$R_{\geq T} = \sum_{t=T}^{end} \gamma^{t-T} R(t)$$

Notes on the backpropagation step

$$Q(s,a) \leftarrow \frac{R_{\geq T} + (n_{s,a} - 1)Q(s,a)}{n_{s,a}}$$

$$R_{\geq T} = \sum_{t=T}^{end} \gamma^{t-T} R(t)$$

- This doesn't depend on any other Q-value.
  - ***We're no longer doing dynamic programming***.

- $R_{\geq T}$ can be computed incrementally:

$$R_{\geq T} = \gamma R_{\geq T+1} + R(T)$$

Online MCTS Value Backup

Observe sequence of (state, action) pairs and corresponding rewards.

- Save (state, action, reward) during **selection**/**expansion**
- Save only reward during **simulation**

Want to compute value (on the current rollout) for each (s,a) pair, then average with old values.

$\gamma R_{\geq T+1} + R(T)$

```
states:   [s0, s7, s3, s5]
actions:  [a0, a0, a2, a1]
rewards:  [ 0, -1, +2,  0, 0, +1, -1]
```

γ=.9

Compute values for the current rollout.

When should a rollout end?

A rollout ends if a terminal state is reached.

- Will we always reach a terminal state?
- If not, what can we do about it?

- As t grows, γ^t gets exponentially smaller.
- Eventually γ^t will be small enough that rewards have negligible effect on the start state's values.
- This means we can set a depth limit on our rollouts.

Heuristics

- If we cut off rollouts at some depth, we may not have found any useful rewards.
- If we have a heuristic that helps us estimate future value, we could evaluate it at the end of the rollout.
- We could also change the agents' rewards to give it intermediate goals.
  - This is called reward shaping, and is a topic of active research in reinforcement learning.