Q-Learning

2/22/17

MDP Examples

MDPs model environments where state transitions are affected both by the agent's action and by external random elements.

- Gridworld
  - Randomness from noisy movement control
- PacMan
  - Randomness from movement of ghosts
- Autonomous vehicle path planning
  - Randomness from controls and dynamic environment
- Stock market investing
  - Randomness from unpredictable price movements

What is value?

The value of a state (or action) is the expected sum of discounted future rewards.

γ = discount

r_t = reward at time t

$$V = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

$$V(s) = R(s) + \gamma \max_a Q(s,a)$$

$$Q(s,a) = \sum_{s'} P(s' \mid s,a) V(s')$$

VI Pseudocode (again)

```
values = {state : R(state) for each state}

until values don't change:
   prev = copy of values

   for each state s:
      initialize best_EV
      for each action:
         EV = 0
         for each next state ns:
            EV += prob * prev[ns]
         best_EV = max(EV, best_EV)

      values[s] = R(s) + gamma*best_EV
```

Optimal Policy from Value Iteration

Once we know values, the optimal policy is easy:

- Greedily maximize value.
  - Pick the action with the highest expected value.
  - We don't need to think about the future, just the value of states that can be reached in one action.

Why does this work?

Why don't we need to consider the future?

The state-values already incorporate the future

- Sum of discounted future rewards.

What if we don’t know the MDP?

- We might not know all the states.
- We might not know the transition probabilities.
- We might not know the rewards.

- The only way to figure it out is to explore.

- We now need two things:
  - A policy to use while exploring.
  - A way to learn expected values without knowing exact transition probabilities.

Known vs. Unknown MDPs

If we know the full MDP:

- All states and actions
- All transition probabilities
- All rewards

Then we can use value iteration to find an optimal policy before we start acting.

If we don't know the MDP:

- Missing states
  - Generally know actions
- Missing transition probabilities
- Missing rewards

Then we need to try out various actions to see what happens. This is called RL: ***Reinforcement Learning***.

Temporal Difference (TD) Learning

Key idea: Update estimates based on experience, using differences in utilities between successive states.

Update rule: $V(s) = \alpha\left[R(s) + \gamma V(s')\right] \; + \; (1-\alpha)V(s)$

Equivalently: $V(s) \mathrel{+}= \; \alpha\left[R(s) + \underbrace{\gamma V(s') - V(s)}_{\text{temporal difference}}\right]$

How the heck does TD learning work?

TD learning maintains no model of the environment.

- It never learns transition probabilities.

Yet TD learning converges to correct value estimates. Why?

Consider how values will be modified...

- when all values are initially 0.
- when s' has a high value.
- when s' has a low value.
- when discount is close to 1.
- when discount is close to 0.
- over many, many runs.

$$V(s) \; += \alpha\Big[R(s) + \gamma V(s') - V(s)\Big]$$

Q-learning

Key idea: TD learning on (state, action) pairs.

- Q(s,a) is the expected value of doing action a in state s.
- Store Q values in a table; update them incrementally.

Update rule:

$$Q(s,a) = \alpha \left[R(s) + \gamma \left[\underbrace{\max_{a'} Q(s',a')}_{\boxed{V(s')}} \right] \right] + (1-\alpha) Q(s,a)$$

Equivalently:

$$Q(s,a) \mathrel{+}= \alpha \left[R(s) + \gamma \left[\max_{a'} Q(s',a') \right] - Q(s,a) \right]$$

Exercise: carry out Q-learning

0 0 0 0	0 0 0 0	0 0 0 0	+1
0 0 0 0		0 0 0 0	-1
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

discount: 0.9 learning rate: 0.2

We've already seen the terminal states.

Use these exploration traces:

(0,0)→(1,0)→(2,0)→(2,1)→(3,1)

(0,0)→(0,1)→(0,2)→(1,2)→(2,2)→(3,2)

(0,0)→(0,1)→(0,2)→(1,2)→(2,2)→(2,1)→(3,1)

(0,0)→(1,0)→(2,0)→(2,1)→(2,2)→(3,2)

(0,0)→(1,0)→(2,0)→(3,0)→(3,1)

(0,0)→(0,1)→(0,2)→(1,2)→(2,2)→(3,2)

$$Q(s,a) \;+=\; \alpha\Big[R(s)+ \gamma\Big[\max_{a'} Q(s',a')\Big] - Q(s,a)\Big]$$

Optimal Policy from Q-Learning

Once we know values, the optimal policy is easy:

- Greedily maximize value.
  - Pick the action with the highest **Q-value**.
  - We don't need to think about the future, just the **Q-value of each action**.

If our value estimates are correct, then this policy is optimal.

Exploration Policy *During* Q-Learning

What policy should we follow while we're learning (before we have good value estimates)?

- We want to explore: try out each action enough times that we have a good estimate of its value.

- We want to exploit: we update other Q-values based on the best action, so we want a good estimate of the value of the best action.

We need a policy that handles this tradeoff.

- One option: ε-greedy