TD Learning – SARSA

* Experience world through episodes
  + s, a, r, s’, a’, s’’, a’’, a’’, r’’, s’’’ …
* Update estimates at each transition (s, a, r, s’, a’)
  + target = R(s, a, s’) + /gamQ/pi(s’, a’)
  + Q/pi(s, a) <- (1 - /alph)Qhat/pi(s, a) + /alph target
* Over time, updates mimic bellman updates

Q-Value:

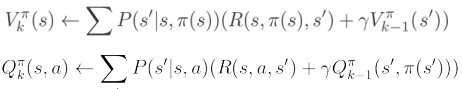
Q/pi(s, a) is expected utility starting in s, taking action a, and then following policy /pi

* 

Monte Carlo Evaluation

* Act according to /pi and collect data: S0, a0, r1, s1, a1, r2,….,sT
* Q/pi(s, a) = average of Gt where st = s, at = a

Recall: policy evaluation?

* 

Big Idea: Learn from every experience

* Update Q/pi each time we experience a transition (s, a, r, s’, a’)
* Keep a running average between current estimate and new experiences
* 
* 

Above is known as SARSA

* s, a, r, s’, a’
* We can use TD-learning to estimate V/pi based on (s, a, r, s’) transitions
* 

# Active Reinforcement Learning

Full reinforcement learning

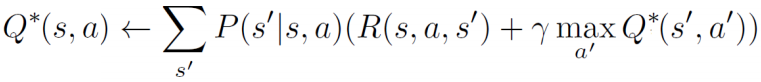
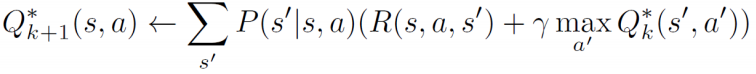
* Don’t know transitions P(s’ | s, a)
* Don’t know rewards R(s, a, s’)
* You choose actions
* Goal: Learn optimal policy / values

Learner makes choices

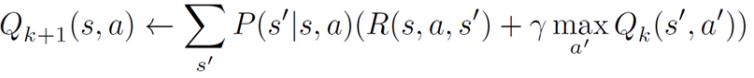
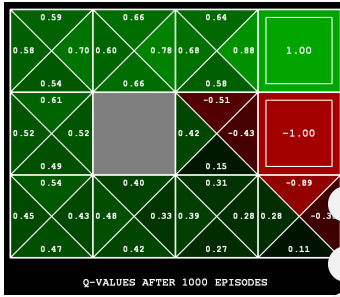
Tradeoff: Exploration vs exploitation

Not offline. Actually take actions in the world

Recap: Q-Values

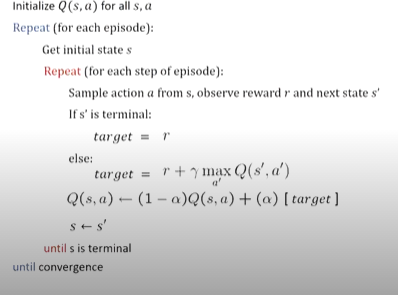
* Q\*(s, a) = expected utility starting in s, taking action a and (thereafter) acting optimally
* Bellman equation:
  + 
* Q-Value Iteration
  + 
  + 

Tabular Q-Learning

* Q-Learning: Sample-based Q-value iteration
* 

Learn Q-Values as you go

* Receive a sample (s, a, s’, r)
* Consider previous estimate: Q-hat(s, a)
* Consider new estimate:
  + 
* Incorporate new estimate into a running average
  + 

Properties

* Converges to optimal policy even if acting suboptimally
* Called off-policy learning
  + On policy: Learn values of policy used to generate data
  + Off: Learn value of another policy
* Caveats
  + Have to explore enogh
  + Have to eventually make learning rate small enough
    1. But not decrease it too early
  + Basically, in the limit, doesn’t matter how you select actions

Exploration vs Exploitation

* All states and actions should be visited infinitely often

How to sample actions:

* Choose random actions?
  + Given infinite time will get most knowledge, but probability of finding optimal Q state low
* Choose action greedily (highest q-value)
  + Insufficient exploration of state action space

Epsilon-greedily

* With (small) probability /eps, act randomly
* With (large) probability 1-/eps, act on current policy
* How to set /eps:
  + Use fixed /eps
  + Start with large, decrease over time to a small positive number (eg 0.1)