

Q-Learning Review

- Want to maintain good properties of TD
- Learns good policies and optimal value function, not just the value of a fixed policy
- Simple modification to TD that learns the optimal policy regardless of how you act! (mostly)

Q-learning

• Recall value iteration:

$$V^{i+1}(s) = \max_{a} R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^{i}(s')$$
• Can split this into two functions:

$$Q^{i+1}(s,a) = R(s,a) + \gamma \sum_{s'} P(s' | s,a) V^{i}(s' | s')$$
$$V^{i+1}(s) = \max_{a} Q^{t+1}(s,a)$$

Q-learning

- Store Q values instead of a value function
- Makes selection of best action easy
- Update rule:

 $Q^{temp}(s,a) = r + \gamma \max_{a'} Q^{i}(s',a')$

$$Q^{i+1}(s,a) = (1-\alpha)Q^{i}(s,a) + \alpha Q^{temp}(s,a)$$

Q-learning Properties

- Converges under same conditions as TD
- Still must visit every state infinitely often
- Separates policy you are currently following from value function learning:

 $Q^{temp}(s,a) = r + \gamma \max_{a'} Q^{i}(s',a')$

 $Q^{i+1}(s,a) = (1-\alpha)Q^{i}(s,a) + \alpha Q^{temp}(s,a)$

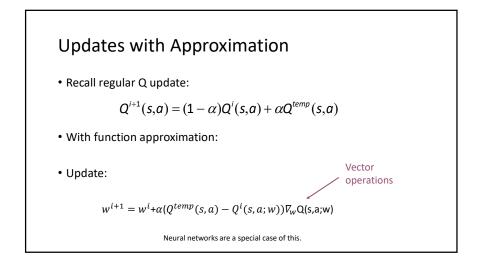
Note: If there is only one action possible in each state, then Q-learning and TD-learning are identical

Value Function Representation

- Fundamental problem remains unsolved:
 - TD/Q learning solves model-learning problem, but
 - Large models still have large value functions
 - Too expensive to store these functions
 - Impossible to visit every state in large models
- Function approximation
 - Use machine learning methods to generalize
 - Avoid the need to visit every state

Function Approximation

- General problem: Learn function f(s)
 - Linear regression
 - Neural networks
 - State aggregation (violates Markov property)
- Idea: Approximate f(s) with $g(s,\theta)$
 - g is some easily computable function of s and $\boldsymbol{\theta}$
 - Try to find $\boldsymbol{\theta}$ that minimizes the error in g



Learning to play Backgammon

- Neurogammon developed in 1989 using supervised learning
 - Trained NN on expert human moves
 - Played at level of intermediate human player
- TD-gammon developed in 1992 using RL
 - Neural network value function approximation
 - TD sufficient (known model)
 - Using raw board positions, learned to play as well as neurogammon
 - Tesauro added carefully selected features to the network
 - Then had it play 1 million games played against self
 - Comparable performance to best human players

RL after TD-gammon

- For 20 years after TD-gammon, many tried to reproduce success of combination of RL with neural networks for other domains
- Often FAILED with bad policies or weights that diverged (went to infinity)
- Community largely retreated into linear value function approximation and focused on techniques for generating and selecting good features
- Deepmind Deep RL result causes seismic shift in community comparable or larger to Tesauro's result

Switch to David Silver's Slides