1 Non-deterministic rewards and actions

Also known as a Non-deterministic Markov Decision Process (MDP)

1.1 Value of a policy

The value V_π of a policy is the same as before:

$$V_\pi(s_t) \equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{1}$$

But now just with an expected value:

$$V_\pi(s_t) \equiv E\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}\right] \tag{2}$$

1.2 Q equation

Remember:

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a)) \tag{3}$$

Which now becomes:

$$Q(s,a) \equiv E[r(s,a)] + \gamma E[V^*(\delta(s,a)))] \tag{4}$$

$$Q(s,a) \equiv E[r(s,a)] + \gamma \sum_{s'} P(s'|s,a) V^*(s') \tag{5}$$

And expressing recursively:

$$Q(s,a) = E[r(s,a)] + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a') \tag{6}$$

The key is that this reduces back to original Equation 13.6.

1.3 $\hat{Q}$ estimation

Previous approach was to just assign $\hat{Q}$ values using $\max \hat{Q}$ of other states. This doesn't work, because our reward is non-deterministic. However, we can solve by slowly converging $\hat{Q}$ using an α term.

New update rule:

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_n - 1(s,a) + \alpha_n[(eq13.9)] \tag{7}$$

And one way of assigning α which slowly reduces the more we examine each relationship:

$$\alpha_n = \frac{1}{1 + visits_n(s, a)} \tag{8}$$

This is similar to simulated annealing, but without resetting the temperature.

2 Temporal difference learning

Normal Q learning to this point only has one-step lookahead. TDL offers a multi-step lookahead.

Pros: we get very close to correct Q value. Cons: if any of the actions are chosen suboptimally, we wont have an opportunity to correct them

3 Generalizing from examples

Instead of using large tables, consider training a back-propagation neural network using $\hat{Q}$ as training examples.

Input is (s, a) and output is $\hat{Q}$ estimate, or input (s) and multiple $\hat{Q}$ outputs for each a.

This is a more accurate way of representing problem. In neuroscience they have watched dopamine stimulate backpropagation in a monkey brain to learn distant rewards.

4 Relationship to dynamic programming

Q learning is novel because it doesn't require knowledge of $d(s, a)$ or $r(s, a)$. (Otherwise we might look for inverse of δ?)

Online versus offline training: offline computations are lower-cost, but require knowledge of how to define our $r(s, a)$ rewards.