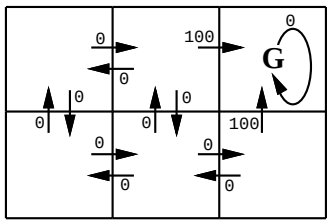
Outline

Elaborations on Q-learning

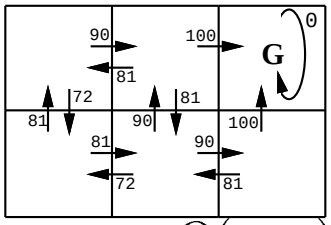
- The convergence theorem
- Generalizing $\hat{Q}$ to unseen states
- Temporal difference learning TD(λ)
- Exploration strategies
- Using prior knowledge
- Perceptual aliasing

©Tom M. Mitchell, 1994

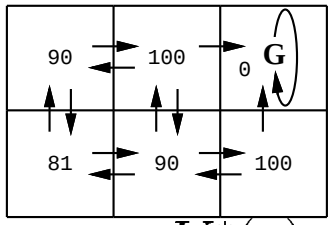
Q-learning



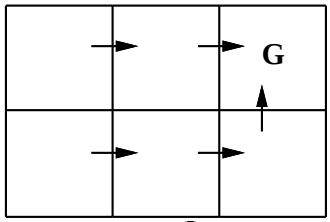
r



$Q(s,a)$



$V^*(s)$



Optimal policy

$V^*(s_t) = \Sigma_{i=0}^{\inf} \gamma^i r_{t+i}$, assuming optimal policy

$Q(s_t, a_t) = \Sigma_{i=0}^{\inf} \gamma^i r_{t+i}$, optimal policy after a_t

$$Q^{Train}(s_t, a_t) \leftarrow r_t + \gamma \max_a \hat{Q}(s_{t+1}, a)$$

©Tom M. Mitchell, 1994

Convergence Theorem

Note we'd like training examples of form

$$< s_t, a_t >, Q(s_t, a_t)$$

but have only a sequence of examples of form

$$< s_t, a_t >, r_t$$

so algorithm estimates $Q(s_t, a_t)$ from r_t, $\hat{Q}(s_t, a_t)$

- Not obvious this should converge
- Theorem shows it does, if update rule for $\hat{Q}$ is a contraction (reduces global error each iteration)
- but uses unrealistic assumptions
  - Visit each $< s, a >$ infinitely often
  - Build Q table of size $|S||A|$

©Tom M. Mitchell, 1994

Convergence of Q-Learning

[Watkins & Dyan, ML Journal, 8(3), May 1992]

If

- s_{t+1} is probabilistic function of s_t, a_t
- r_t is bounded probabilistic function of s_t, a_t
- visit each $< s, a >$ pair infinitely often
- use following algorithm:

1. observe current state s_t
2. select action a_t
3. receive immediate reward r_t
4. adjust $\hat{Q}$ using following rule

$$\hat{Q}_n(s_t, a_t) \leftarrow (1 - \alpha_n)\hat{Q}_{n-1}(s_t, a_t) + \alpha_n[r_t + \max_a \hat{Q}(s_{t+1}, a)]$$

- α_n decays with n, but not too fast

Then

- $\hat{Q}$ converges to Q with probability 1

©Tom M. Mitchell, 1994

Generalizing from Examples

- Algorithm builds table, doesn't generalize to unvisited states
- Impractical for complex problems
- Could use our favorite function approximator (i.e., curve fitting, neural network, ...)
- But convergence theorem no longer applies
- Does it converge in practice??

©Tom M. Mitchell, 1994

Problems with Generalizing from Examples

1. Function approximator might wildly overestimate Q for unseen states
   - Global error in $\hat{Q}$ will no longer decrease each iteration
2. What if we use assume well-behaved approximator?
   - Assume function approximator makes errors, but mean error is zero
   - assume $\hat{Q}_n(s,a)$ is distributed about $Q(s,a)$
   - **but** $\max_a \hat{Q}(s,a)$ won't be distributed about $\max_a Q(s,a)$
   - so neither will $\hat{Q}_{n+1}(s,a)$
3. Can we find function approximators that are contraction operations? [Gordon, in progress]

©Tom M. Mitchell, 1994

Generalization and Q-learning

It (sometimes) works even though unprovable

- Learning Backgammon with neural nets [Tesauro, ML Journal, May 1992]
- Learning "Pacman" with neural nets [Lin, ML Journal, May 1992]
- Learning mobile robot control [Thrun & Mitchell, 1994]

©Tom M. Mitchell, 1994

TD(λ) [Sutton, MLJ, 1988]

Assume agent memorizes entire episode up to absorbing reward state

One way to estimate Q training values:

$$Q_1^{Train}(s_t, a_t) \leftarrow r_t + \gamma \max_a \hat{Q}(s_{t+1}, a)$$

Second way: use observed rewards rather than $\hat{Q}$

$$Q_2^{Train}(s_t, a_t) \leftarrow r_t + \gamma r_{t+1} + ... + \gamma^n r_{t+n}$$

Danger of Q_1^{Train}:

- $\hat{Q}$ is only an approximation to Q

Danger of Q_2^{Train}:

- this episode might not use optimal actions
- observed rewards might be atypical (when non-deterministic)

TD(λ): blend these two methods:

$$Q_\lambda^{Train}(s_t, a_t) \leftarrow r_t + \gamma[\ (1-\lambda) \max_a \hat{Q}(s_t, a_t) + \lambda Q_\lambda^{Train}(s_{t+1}, a_{t+1})]$$

©Tom M. Mitchell, 1994

Exploration Strategies

What's best way to explore the environment?

- *Exploration*? i.e., seek unvisited states/actions
- *Exploitation*? i.e., seek known rewards

Some approaches:

- Completely random search
- Probabilistic, prefering higher $\hat{Q}(s, a)$
  - what if initialize $\hat{Q}$ to +1000 (optimistic)?
  - what if initialize $\hat{Q}$ to -1000 (pessimistic)?
- Based on count of visits
- Based on recency of visits
- Based on error in recent updates

©Tom M. Mitchell, 1994

Using prior knowledge

Learn and use action model ($S \times A \rightarrow S'$)

- If accurate, complete model, can use dynamic programming
- If accurate, incomplete model, simulate *some* actions (DYNA, [Sutton, IWML 1990])
- If inaccurate model, can use EBNN [Mitchell & Thrun, NIPS 1992]

Using previously learned policies as primitive actions [Lin 1992], [Singh 1992]

- First train low level (e.g., wall following), then higher level (i.e., room navigation)

©Tom M. Mitchell, 1994

Perceptual aliasing

Markov process: optimal next action completely determined by current observed state

What if state is only partially observable?

- Could model actions as non-deterministic (but they're not really independent of past states, actions)
- Could try to learn more elaborate state representation that incorporates features of previous states

©Tom M. Mitchell, 1994