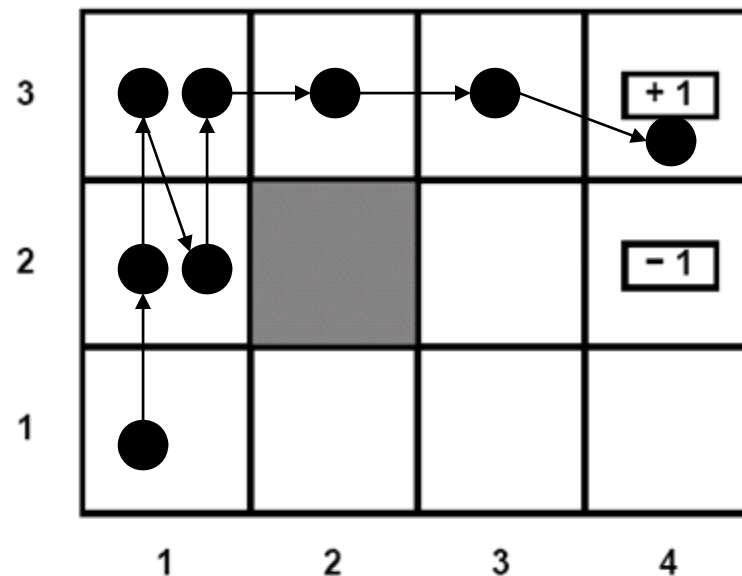# Reinforcement Learning cont.

Dec – 01 – 2008

Refresh Your Memory…

- Last class, we assumed that the agent executes a fixed policy π

- The goal is to evaluate how good π is, based on some sequence of trials performed by the agent

Passive Learning

- Methods:
  - DUE: Directly estimate the utility of the states $U_\pi(s)$ by averaging **_"reward-to-go" from each state_** – slow convergence, not using the Bellman equation constraints
  - ADP: learn the transition model **T** and the reward function **R,** then do **_policy evaluation_** to learn $U_\pi(s)$ – few updates, but each update is expensive (O(n^3))
  - TD learning: maintain a running average of the state utilities by doing **_online mean estimation_** – cheap updates but needs more updates than ADP

Goal of active learning

- Let's suppose we still have access to some sequence of trials performed by the agent
- The goal is to learn an optimal policy
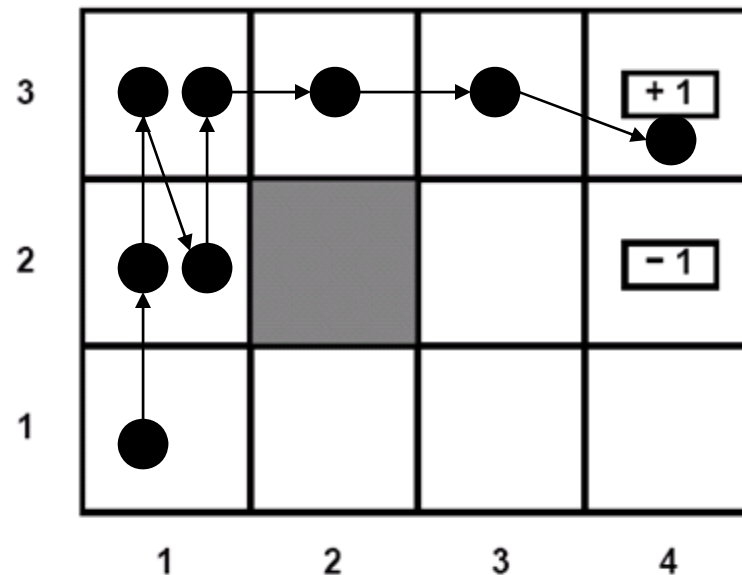
Active Reinforcement Learning Agents

We will describe two types of Active Reinforcement Learning agents:

- Active ADP agent

- Q-learner (based on TD algorithm)

Active ADP Agent
(Model-based)

- Using the data from its trials, the agent learns a transition model **T** and a reward function **R**

- With T(s,a,s') and R(s), it has an estimate of the underlying MDP

- It can compute the optimal policy by solving the Bellman equations using value iteration or policy iteration

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s')U(s')$$

Active ADP Agent

- Now that we've got a policy that is optimal based on our current understanding of the world, what should we do?

- Greedy agent: an agent that executes the optimal policy for the learned model at each time step

- Let's see what happens in the maze world

The Greedy Agent
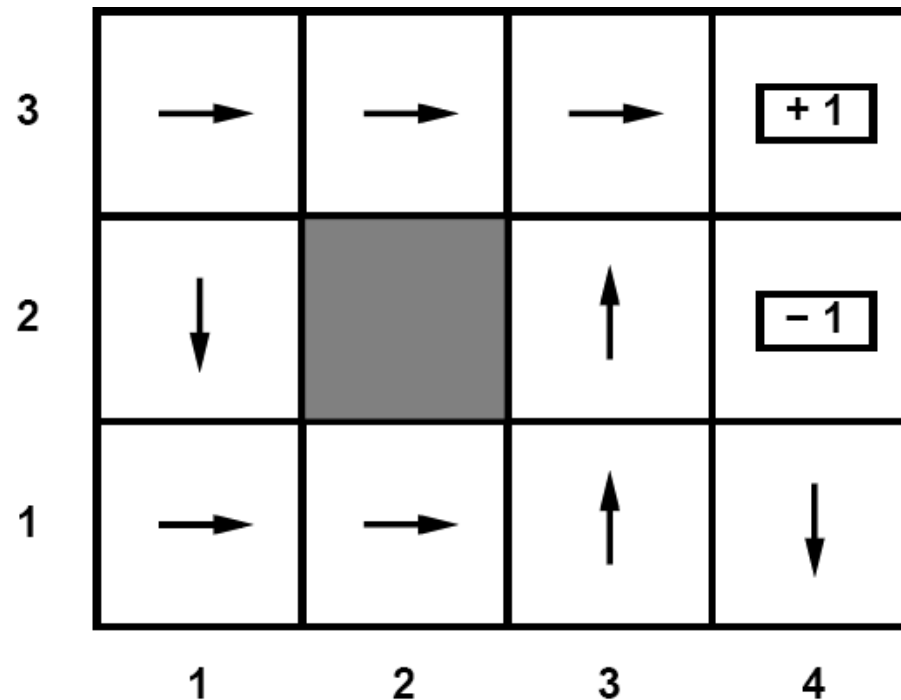
The agent finds the lower route to get to the goal state but never finds the optimal upper route. The agent is stubborn and doesn't change so it doesn't learn the true utilities or the true optimal policy

What happened?

- How can choosing an optimal action lead to suboptimal results?

- The learned model is not the same as the true environment

- In fact, the set of trials observed by the agent was insufficient to build a good model of the environment

How can we address this issue?

    We need more training experience …

Exploitation vs Exploration

- Actions are always taken for one of the two following purposes:
  - Exploitation: Execute the current optimal policy to get high payoff
  - Exploration: Try new sequences of (possibly random) actions to improve the agent's knowledge of the environment even though current model doesn't believe they have high payoff
- Pure exploitation: gets stuck in a rut
- Pure exploration: not much use if you don't put that knowledge into practice

Optimal Exploration Strategy?

- What is the optimal exploration strategy?
  - Greedy?
  - Random?
  - Mixed? (Sometimes use greedy sometimes use random)
- It turns out that the optimal exploration strategy has been studied in-depth in the N-armed bandit problem

N-armed Bandits

- We have N slot machines, each can yield $1 with some probability (different for each machine)
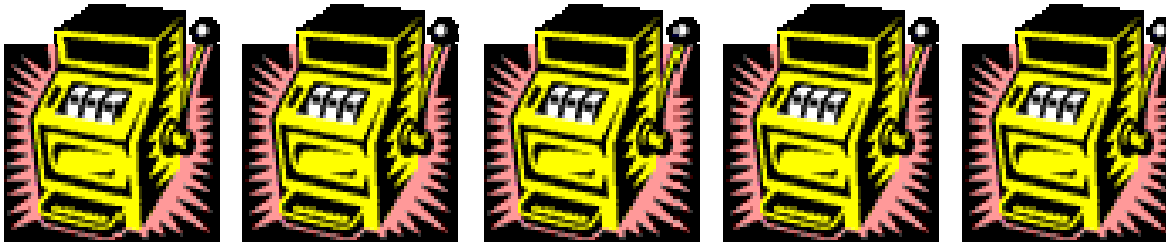
- What order should we try the machines?
  - Stay with the machine with the highest observed probability so far?
  - Random?
  - Something else?
- Bottom line:
  - It's not obvious
  - In fact, an exact solution is usually intractable

GLIE

- Fortunately it is possible to come up with a **reasonable** exploration method that eventually leads to optimal behavior by the agent
- Any such exploration method needs to be Greedy in the Limit of Infinite Exploration (GLIE)
- Properties:
  - Must try each action in each state an unbounded number of times so that it doesn't miss any optimal actions
  - Must eventually become greedy

Examples of GLIE schemes

- ε-greedy:
  - Choose optimal action with probability (1-ε)
  - Choose a random action with probability ε/(number of actions-1)

- Active ε-greedy ADP agent
  1. Start with initial T and R learned from the original sequence of trials
  2. Compute the utilities of states U(s) using value iteration
  3. Take action use the ε-greedy exploitation-exploration strategy
  4. Update T and R, go to 2

Another approach

- Favor actions the agent has not tried very often, avoid actions believed to be of low utility

- We can achieve this by altering value iteration to use $U^+(s)$, which is an optimistic estimate of the utility of the state s (using an ***exploration function***)

Exploration Function

Originally: $\boxed{U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')}$

$$U^+(s) = R(s) + \gamma \max_a f\left(\sum_{s'} T(s, a, s') U^+(s'), N(a, s) \right)$$

N(a,s) = number of times action a tried in state s
U^+(s) = optimistic estimate of the utility of state s

Exploration function f(u,n):
$$f(u, n) = \begin{cases} R^+ \text{ if } n < N_e \\ u \text{ otherwise} \end{cases}$$

- Trades off greedy (preference for high utilities u) against curiosity (preference for low values of n – the number of times a state-action pair has been tried)

- R+ is an optimistic estimate of the best possible reward obtainable in any state

- If a hasn't been tried enough in s, you assume it will somehow lead to *gold* – optimistic

- N_e is a limit on the number of tries for a state-action pair

Using Exploration Functions

1. Start with initial T and R learned from the original sequence of trials
2. Perform value iteration to compute U+ using the exploration function
3. Take the greedy action
4. Update estimated model and goto 2

Q-learning

Previously, we needed to store utility values for a state ie.

- $U(s)$ = utility of state s = expected sum of future rewards

Now, we will store Q-values, which are defined as:

- $Q(a,s)$ = value of taking action a at state s = expected maximum sum of future discounted rewards after taking action a at state s

Q-learning

- Now, instead of storing a table of U(s) values, we store a table of Q(a,s) values

- Note the relationship:

$$U(s) = \max_a Q(a,s)$$

- Note that if you estimate Q(a,s) for all a and s, we can simply choose the action that maximize Q, without using the model

Q-learning

At equilibrium when the Q-values are correct, we can write the constraint equation:

$$Q(a,s) = R(s) + \gamma \sum_{s'} T(s,a,s')U(s')$$

Reward at state s

Expected value for action-state pair (a, s)

Expected value averaged over all possible states s' that can be reached from s after executing action a

Q-learning

At equilibrium when the Q-values are correct, we can write the constraint equation:

$$Q(a,s) = R(s) + \gamma \sum_{s'} T(s,a,s') U(s')$$

$$Q(a,s) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(a',s')$$

Note we still need to learn T(s,a,s')!

Can we learn it in a model free way?

Q-learning

At equilibrium when the Q-values are correct, we can write the constraint equation:

$$Q(a,s) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(a',s')$$
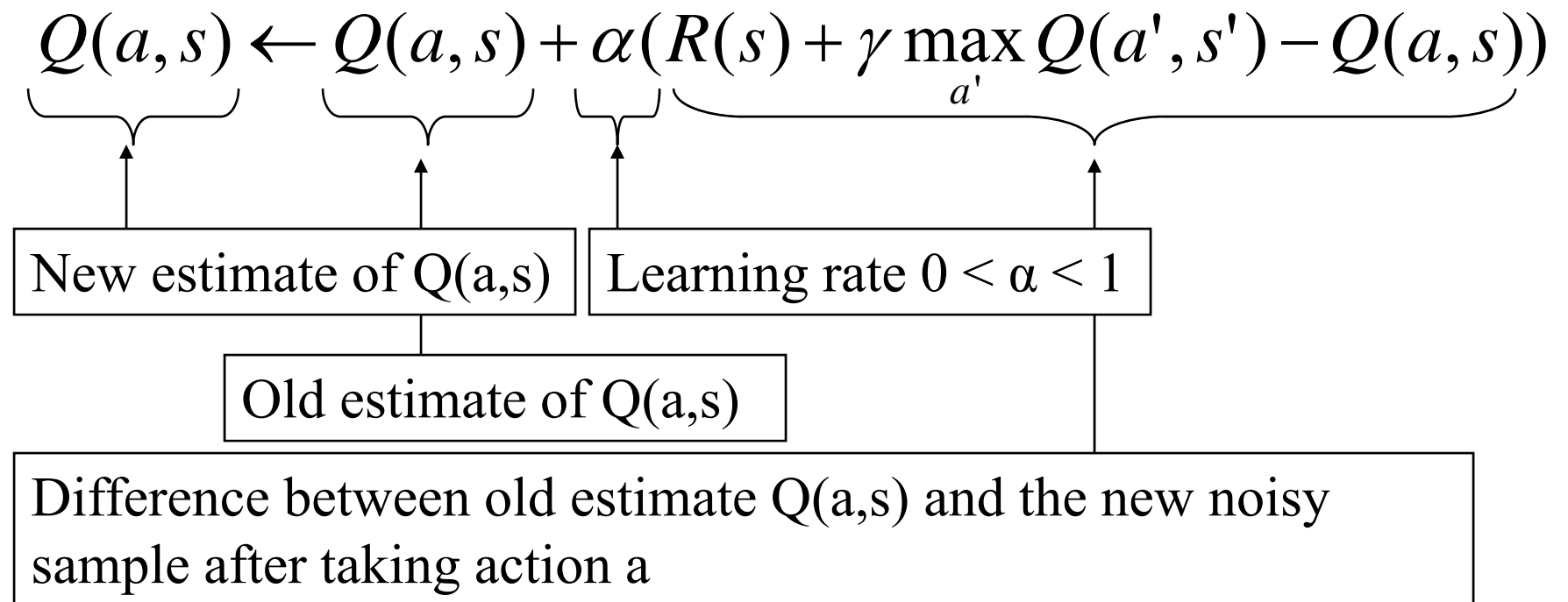
Reward at state s

Best expected value for action-state pair (a, s)

Best value averaged over all possible states s' that can be reached from s after executing action a

Best value at the next state = max over all actions in state s'

Q-learning Without a Model

- We can use a temporal differencing approach which is model-free
- After moving from state s to state s' using action a:

$$Q(a,s) \leftarrow Q(a,s) + \alpha(R(s) + \gamma \max_{a'} Q(a',s') - Q(a,s))$$

New estimate of Q(a,s)

Learning rate $0 < \alpha < 1$

Old estimate of Q(a,s)

Difference between old estimate Q(a,s) and the new noisy sample after taking action a

Q-learning: Estimating the Policy

Q-Update: After moving from state s to state s' using action a:

$$Q(a,s) \leftarrow Q(a,s) + \alpha(R(s) + \gamma \max_{a'} Q(a',s') - Q(a,s))$$

Policy estimation:

$$\pi(s) = \max_a Q(a,s)$$

Note that T(s,a,s') does not appear anywhere! This is a model-free learning algorithm

Q-learning Convergence

- Guaranteed to converge to an optimal policy [Watkins]

- Very general procedure (because it's model free)

- Converges slower than ADP agent (because it is completely model free and it doesn't enforce consistency among values through the model)

Q-learning: Exploration Strategies

- How to choose the next action while we're learning?
    - Random
    - Greedy
    - ε-Greedy
    - Boltzmann: Choose the next action with probability: (T is a temperature parameter that is decayed over time)

$$e^{\frac{Q(a,s)}{T}}$$

Model-based/Model-free

- Two broad categories of reinforcement learning algorithms:

  1. Model-based eg. ADP
  2. Model-free eg. TD, Q-learning

- Which is better?

  – Model-baesed approach is a knowledge-based approach (ie. model represents known aspects of the environment)

  – Book claims that as environment becomes more complex, a knowledge-based approach is better

What You Should Know

- Exploration vs exploitation
- GLIE schemes
- Difference between model-free and model-based methods
- Q-learning