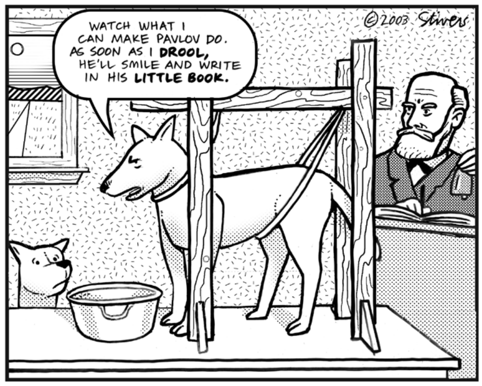
CSE 473

Lecture 18
(Chapter 21)

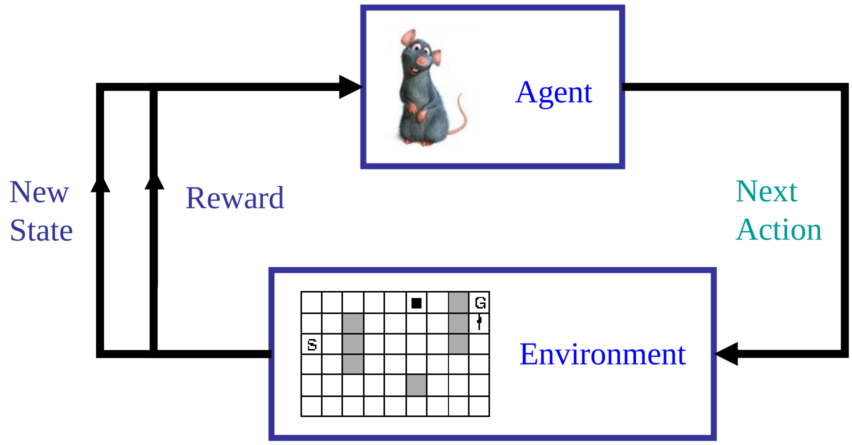
Reinforcement Learning



Today's Outline

- Reinforcement Learning
  - Q-learning
  - Exploration versus Exploitation
  - ε-Greedy Q-learning
  - Feature-based Q-learning

Enter…Reinforcement Learning (RL)



Two main approaches to RL

- Model-based approaches:
  - Explore environment & learn model T=P(**s**'|**s**,**a**) and R(**s**,**a**,**s**')
  - Use model to compute policy MDP-style
  - Works well when state-space is small
- Model-free approach:
  - Don't learn a model
  - Learn value function (Q value) or policy *directly*
  - Works better when state space is large

Algorithms for RL

- We will focus on Q-learning
  - From Q-value iteration to Q-learning
- Approaches for mixing exploration & exploitation
  - ε-greedy method

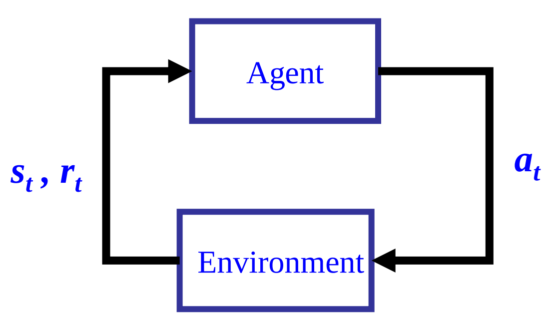
Recall: Q-value iteration

$$Q_{i+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_i(s',a') \right]$$

In RL, we don't have this!

But we get a sample at each time step *t*:

$$(s_t, a_t, r_t, s_{t+1})$$



Q-learning Idea

Instead of expectation under *T*:

$$Q_{i+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_i(s',a') \right]$$

What if we compute a running average of *Q* from all samples received thus far?

$$Q(s,a) \leftarrow \frac{1}{t} \sum_{t \text{ samples}} \left(r + \gamma \max_{a'} Q(s',a') \right)$$

Why does this compute the correct expectation?
Because environment produces samples at the right frequencies!

Running Average

- Running average of t samples of a quantity x:

$$\bar{x}_t = \frac{x_1 + x_2 + \ldots x_{t-1} + x_t}{t}$$

$$= \frac{x_1 + x_2 + \ldots x_{t-1}}{t} \cdot \frac{(t-1)}{(t-1)} + \frac{x_t}{t}$$

$$= \frac{(t-1)}{t} \bar{x}_{t-1} + \frac{1}{t} x_t$$

$$= (1-\alpha)\bar{x}_{t-1} + \alpha\, x_t \quad \text{where } \alpha = 1/t$$

- Running average of Q:

$$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + \alpha (r + \gamma \max_{a'} Q(s',a'))$$

Q-Learning

- Q-Learning = Online ***sample-based*** Q-value iteration. At each time step:
  - Execute action and get new sample (s,a,s’,r)
  - Incorporate new sample into **running average of *Q***:

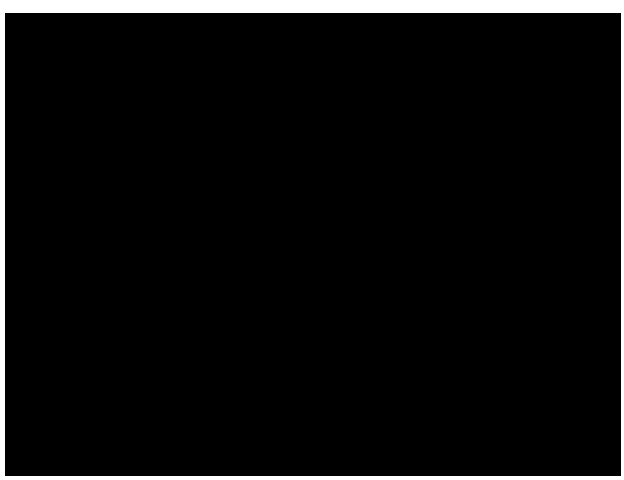
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$$

where α is the learning rate $(0 < \alpha < 1)$.

  - Update policy:

$$\pi(s) = \arg\max_{a} Q(s,a))$$

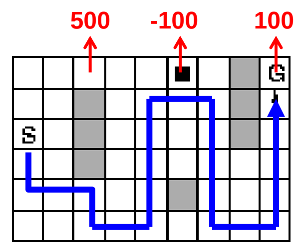
Q-learning example (with manual control)



10

RL agents must tackle an Exploration versus Exploitation tradeoff

- You have explored part of your world and found a reward of 100 – is this the best we can do?
- **Exploitation**: Stick with what you know and accumulate reward
  - RISK: You may be missing out on better rewarding states elsewhere
- **Exploration**: Explore world for states w/ more reward
  - RISK: Wasting time & possibly getting negative reward



ε-Greedy Action Selection for Q-learning

- Balance exploration versus exploitation by allowing *some* random actions
  - Every time step, flip a coin
  - With probability ε, act randomly
  - With probability 1- ε, act according to current policy

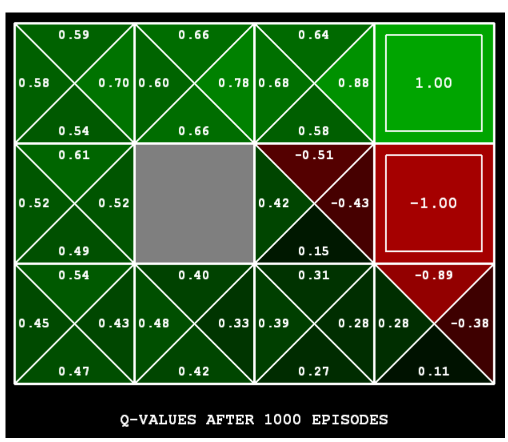
  (ε is a small positive parameter you choose)

- Problems with random actions?
  - Good for exploration but keep thrashing around once learning is done
  - Solution: lower ε over time

ε-Greedy Q-Learning (Movie)

Q-Learning Final Solution

- Q-learning produces table of Q(s,a) values



Q-Learning Properties

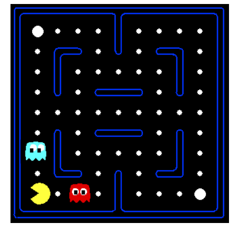
- Amazing result: Q-learning converges to optimal policy
  - If you explore enough and..
  - If you make the learning rate α small enough
  - … but not decrease it too quickly!
  - Q-learning not too sensitive to how you select actions (!)
- Neat property: "off-policy" learning
  - learn optimal policy without following it (doing exploration etc.)

Q-Learning – Small Problem

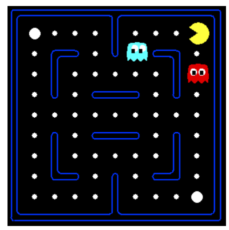
- Doesn't work in the real world
- In realistic situations, we can't possibly learn about every single state!
  - Too many states: Cannot visit them all in training
  - Too many states: Cannot hold all Q-values in memory
- Instead, we need to ***generalize***:
  - Learn about a few states from experience
  - Generalize that experience to new, ***similar*** states
    (Fundamental idea in machine learning)

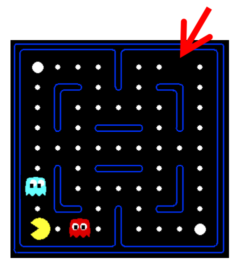
Example: Pacman

- Let's say we discover through experience that this "trapped" state is bad:



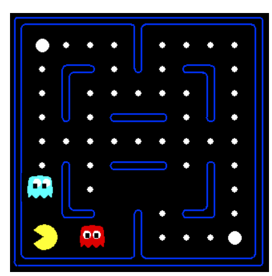
- In naïve Q learning, we know nothing about related states and their Q values:





- Or even this third one!

Feature-Based Representations



- Solution: Describe a state using a vector of features (properties)
  - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
  - Example features:
    - Distance to closest ghost
    - Distance to closest dot
    - Number of ghosts
    - 1 / (dist to dot)^2
    - Is Pacman in a tunnel? (0/1)
    - …… etc.
- Can also describe a Q-state (s, a) with features (e.g. action in a state moves closer to food)

Next Time

- Feature-based Q-learning
- Uncertainty and Probability
- To Do
  - Finish Chapter 21
  - Read Chapter 13
  - Work on Project 3

19