FIT 3080: Intelligent Systems

Expectimax and Reinforcement Learning

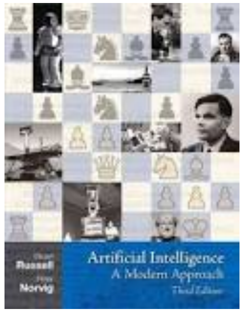
Gholamreza Haffari – Monash University

Many slides over the course adapted from Stuart Russell, Andrew Moore, or Dan Klein

Announcements

- Online Reading:
  - Reinforcement Learning: An Introduction, by Richard Sutton and Andrew Barto, MIT Press
  - Chapter 3 and Chapter 4
  - Accessible from: http://webdocs.cs.ualberta.ca/~sutton/book/ebook/the-book.html
  - Different treatment and notation than the R&N book, beware!
  - Lecture version is the standard for this class

- R&N book:
  - Section 5.5
  - Sections 17.1-3



Outline

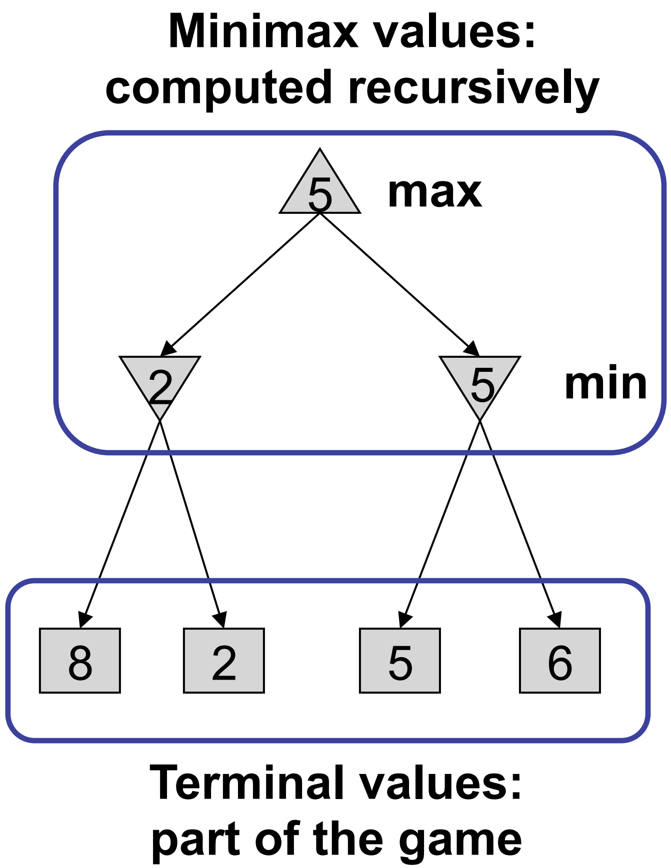
- Expectimax Search
- Reinforcement Learning (RL)
- Passive Learning in RL
  - Model-based
  - Model-free
    - Direct Estimation
    - Temporal Difference
- Active Learning in RL
  - Q-Learning

Outline

- Expectimax Search
- Reinforcement Learning (RL)
- Passive Learning in RL
  - Model-based
  - Model-free
    - Direct Estimation
    - Temporal Difference
- Active Learning in RL
  - Q-Learning

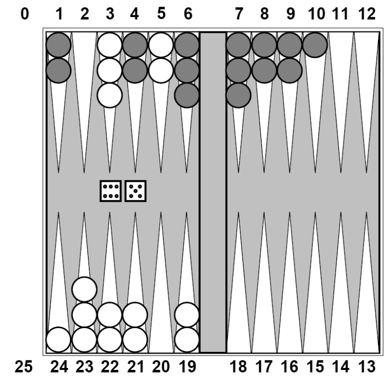
Deterministic Games

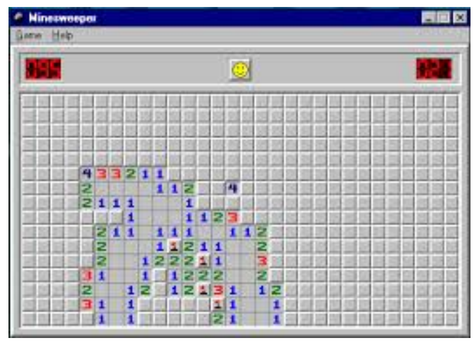
- Deterministic, zero-sum two-player games:
  - Tic-tac-toe, chess, checkers
  - One player maximizes result
  - The other minimizes result
- Minimax search:
  - A state-space search tree
  - Players alternate turns
  - Each node has a minimax value: best achievable utility against a rational adversary



Stochastic/Non-Deterministic Games

- Stochastic games:
  - Backgammon, Solitaire, Minesweeper, …
- Result of an action can be uncertain
  - eg in Backgammon, before rolling the dice, we don't know what's the outcome
- Can we approach it as search in a state space?
  - What's the utility of an action with uncertain outcomes?



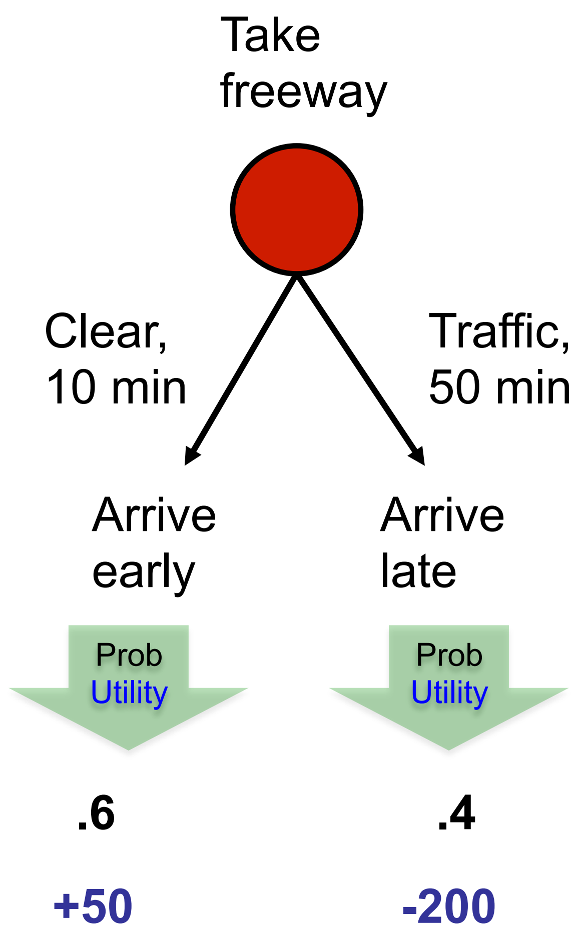


Utility of an Uncertain Action?

- For uncertain actions, consider the expected utility:

  Utility(action) = Σ P(state|action) * Utility(state)

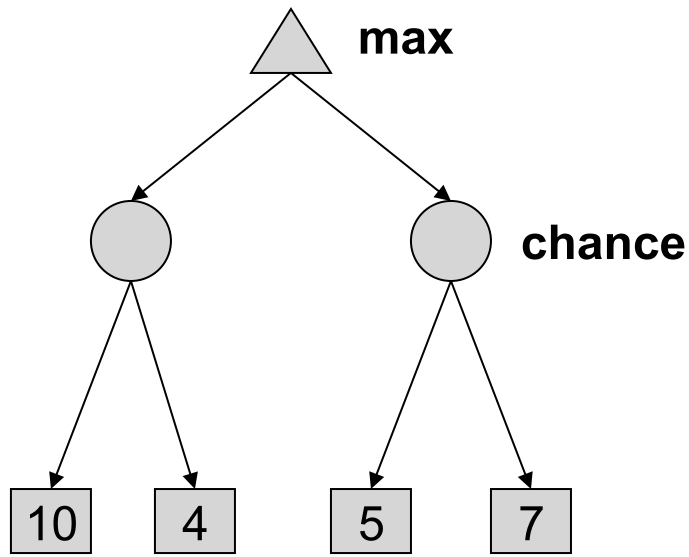
- Example:
  - I want to go from home to the airport
  - I can take the freeway (action)
  - Outcome of take freeway is uncertain:
    - State Arrive Early: (.6, +50)
    - State Arrive Late: (.4, -200)
    - Expected Utility = .6 * 50 + .4 * (-200) = -50



Expectimax Search Trees

(vs mini-max search trees)

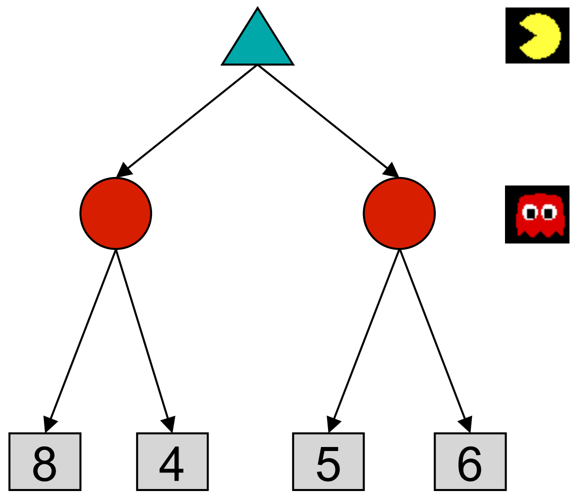
- Can do **expectimax search**
  - Chance nodes, like min nodes, except the outcome is uncertain
  - Calculate expected utilities
  - Max nodes as in minimax search
  - Chance nodes take average (expectation) of value of children

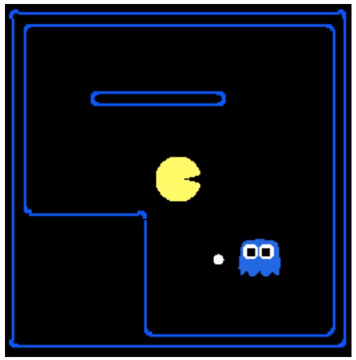


- More formally, we have seen how to formalize the underlying problem as a **Markov Decision Process**

Expectimax Pseudocode

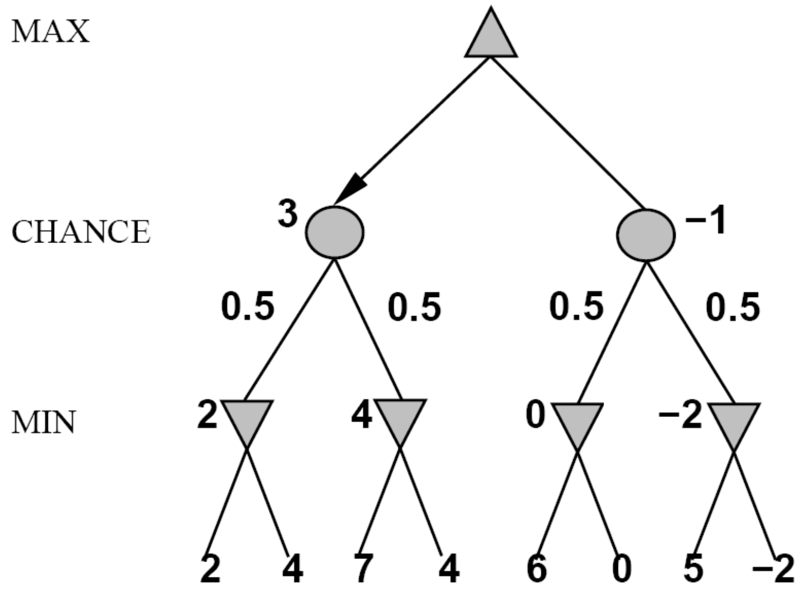
```
def value(s)
    if s is a max node return maxValue(s)
    if s is an exp node return expValue(s)
    if s is a terminal node return evaluation(s)

def maxValue(s)
    values = [value(s’) for s’ in successors(s)]
    return max(values)

def expValue(s)
    values = [value(s’) for s’ in successors(s)]
    weights = [probability(s, s’) for s’ in successors(s)]
    return expectation(values, weights)
```





Mixed Layer Types

- E.g. Backgammon
- Expectiminimax
  - Environment is an extra player that moves after each agent
  - Chance nodes take expectations, otherwise like minimax



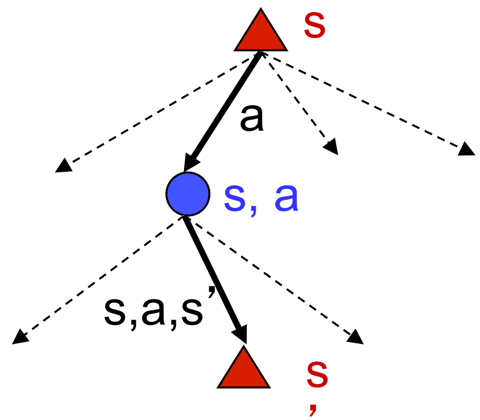
if *state* is a MAX node **then**
    return the highest EXPECTIMINIMAX-VALUE of SUCCESSORS(*state*)
if *state* is a MIN node **then**
    return the lowest EXPECTIMINIMAX-VALUE of SUCCESSORS(*state*)
if *state* is a chance node **then**
    return average of EXPECTIMINIMAX-VALUE of SUCCESSORS(*state*)

Outline

- Expectimax Search
- Reinforcement Learning (RL)
- Passive Learning in RL
  - Model-based
  - Model-free
    - Direct Estimation
    - Temporal Difference
- Active Learning in RL
  - Q-Learning

Recap: MDPs

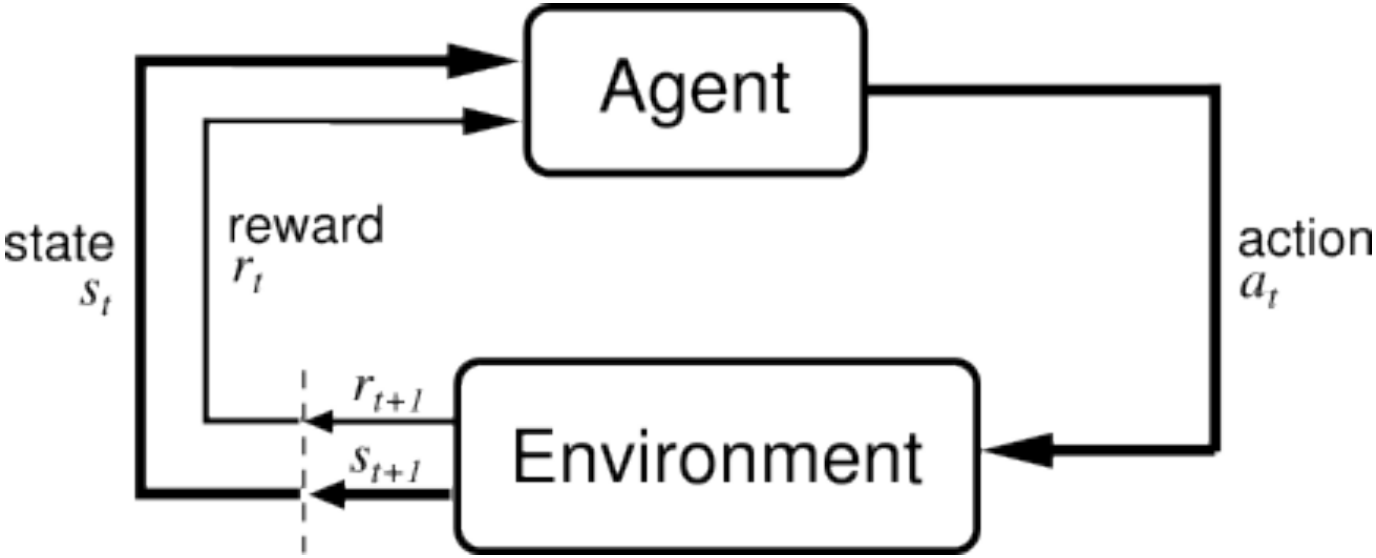
- Markov decision processes:
  - States S
  - Actions A
  - Transitions P(s'|s,a) (or T(s,a,s'))
  - Rewards R(s,a,s') (and discount γ)
  - Start state s_0 (or distribution P_0)



- Quantities:
  - Policy = map of states to actions
  - Utility = sum of discounted rewards
  - Values = expected future utility from a state
  - Q-Values: expected future utility from a q-state

Reinforcement Learning

- Basic idea:
  - Receive feedback in the form of rewards
  - Agent's utility is defined by the reward function
  - Must learn to act so as to maximize expected rewards



Reinforcement Learning

- Reinforcement learning:
  - Still assume an MDP:
    - A set of states $s \in S$
    - A set of actions (per state) A
    - A model T(s,a,s')
    - A reward function R(s,a,s')
  - Still looking for a policy $\pi(s)$
  - New twist: don't know T or R
    - I.e. don't know which states are good or what the actions do
    - Must actually try actions and states out to learn

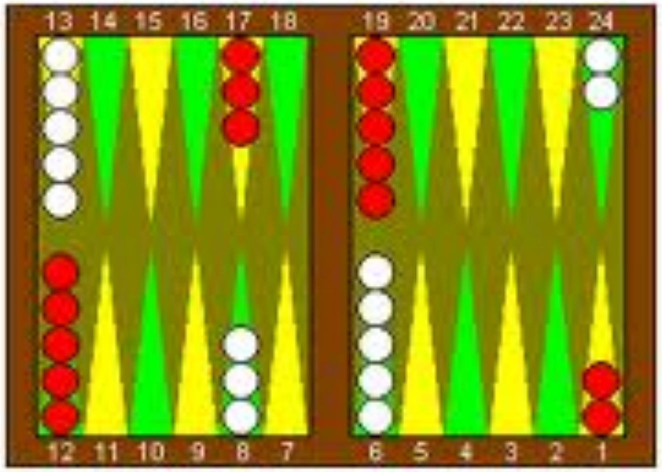
Example: Animal Learning

- RL studied experimentally for more than 60 years in psychology
  - Rewards: food, pain, hunger, drugs, etc.
  - Mechanisms and sophistications debated

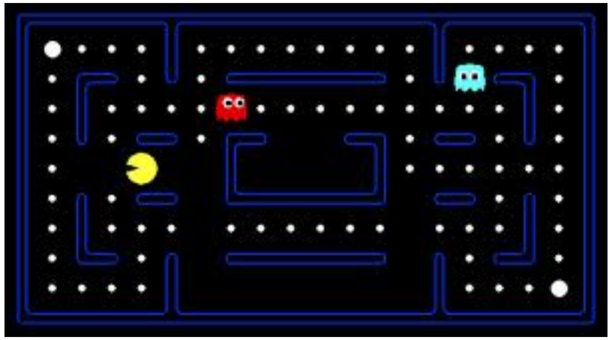
- Example: foraging
  - Bees learn near-optimal foraging plan in field of artificial flowers with controlled nectar supplies
  - Bees have a direct neural connection from nectar intake measurement to motor planning area

Example: Backgammon

- Reward only for win / loss in terminal states, zero otherwise
- TD-Gammon learns a function approximation to V(s) using a neural network
- Combined with depth 3 search, one of the top 3 players in the world



- You could imagine training Pacman this way …
- But it's tricky!



Key Ideas for Learning

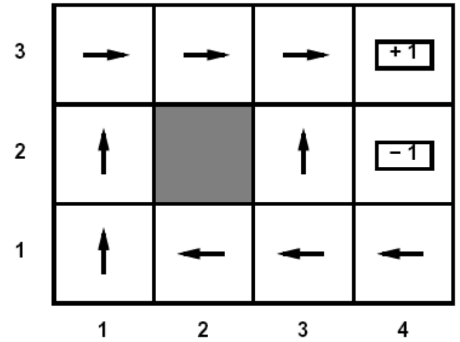
- Online vs. Batch
  - Learn while exploring the world, or learn from fixed batch of data
- Active vs. Passive
  - Does the learner actively choose actions to gather experience? Or, is a fixed policy provided?
- Model learning vs. Model free
  - Do we estimate T(s,a,s’) and R(s,a,s’) , or just learn values/policy directly

Outline

- Expectimax Search
- Reinforcement Learning (RL)
- Passive Learning in RL
  - Model-based
  - Model-free
    - Direct Estimation
    - Temporal Difference
- Active Learning in RL
  - Q-Learning

Passive Learning

- Simplified task
  - You don't know the transitions T(s,a,s')
  - You don't know the rewards R(s,a,s')
  - You are given a policy $\pi(s)$
  - Goal: learn the state values
  - … what value iteration did!



- In this case:
  - Learner has no choice about what actions to take
  - Just execute the policy and learn from experience
  - We'll get to the active case soon

Detour: Sampling Expectations

- What is the average height of people in Monash?
- Method: measure their heights, add them up, and divide by N

Detour: Sampling Expectations

- Want to compute an expectation weighted by P(x):

$$E[f(x)] = \sum_x P(x) f(x)$$

- Model-based: estimate P(x) from samples, compute expectation

$$x_i \sim P(x)$$

$$\hat{P}(x) = \text{count}(x)/k$$

$$E[f(x)] \approx \sum_x \hat{P}(x) f(x)$$

- Model-free: estimate expectation directly from samples

$$x_i \sim P(x)$$

$$E[f(x)] \approx \frac{1}{k} \sum_i f(x_i)$$

- Why does this work? Because samples appear with the right frequencies!

Model-based Learning

- Idea:
  - Learn the model empirically (rather than the “values”)
  - Solve the MDP as if the learned model were correct
  - Better than direct estimation?

- Empirical model learning:
  - Count outcomes for each (s,a)
  - Normalize to give estimate of T(s,a,s’)
  - Discover R(s,a,s’) the first time we experience (s,a,s’)

Example: Model-Based Learning

- Episodes:

(1,1) up -1

(1,2) up -1

(1,2) up -1

(1,3) right -1

(2,3) right -1

(3,3) right -1

(3,2) up -1

(3,3) right -1

(4,3) exit +100

(done)

(1,1) up -1

(1,2) up -1

(1,3) right -1

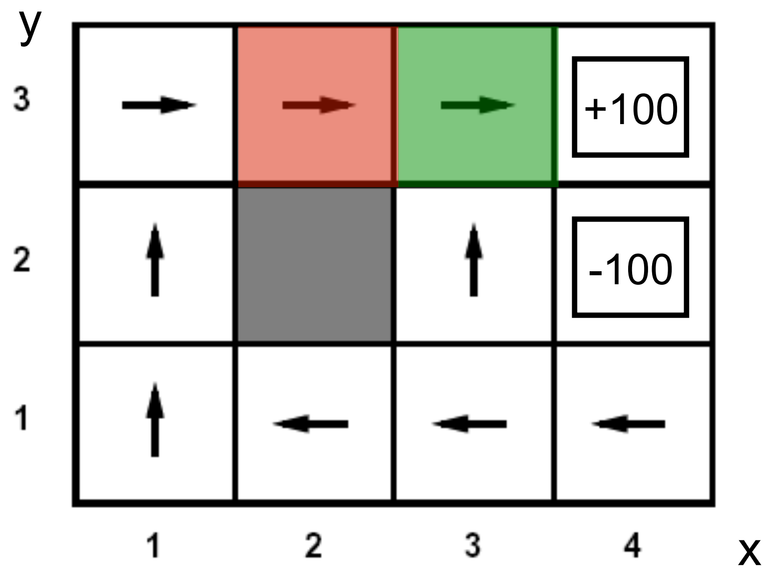
(2,3) right -1

(3,3) right -1

(3,2) up -1

(4,2) exit -100

(done)



$\gamma = 1$

T(<3,3>, right, <4,3>) = 1 / 3

T(<2,3>, right, <3,3>) = 2 / 2

Outline

- Expectimax Search
- Reinforcement Learning (RL)
- Passive Learning in RL
  - Model-based
  - Model-free
    - Direct Estimation
    - Temporal Difference
- Active Learning in RL
  - Q-Learning

Model-free Learning

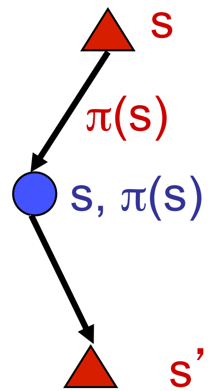
$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

- Big idea: Why bother learning T?

- Question: How can we compute V if we don't know T?

  - Use direct estimation to sample complete trials
  - Compute "values" for each trial based on the sequence of rewards
  - Average "values" across trials at the end

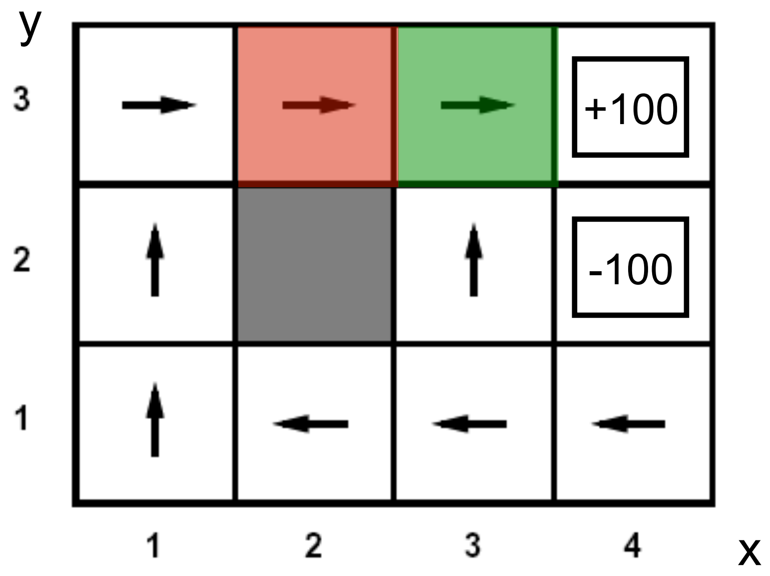
  - i.e. **sampling!**



Simple Case: Direct Estimation

- Episodes:

(1,1) up -1	(1,1) up -1
(1,2) up -1	(1,2) up -1
(1,2) up -1	(1,3) right -1
(1,3) right -1	(2,3) right -1
(2,3) right -1	(3,3) right -1
(3,3) right -1	(3,2) up -1
(3,2) up -1	(4,2) exit -100
(3,3) right -1	(done)
(4,3) exit +100	
(done)	



γ = 1, R = -1

V(2,3) ~ (96 + -103) / 2 = -3.5

V(3,3) ~ (99 + 97 + -102) / 3 = 31.3

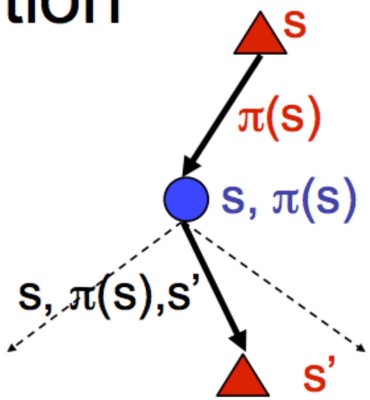
Outline

- Expectimax Search
- Reinforcement Learning (RL)
- Passive Learning in RL
  - Model-based
  - Model-free
    - Direct Estimation
    - Temporal Difference
- Active Learning in RL
  - Q-Learning

Towards Better Model-free Learning

Review: Model-Based Policy Evaluation

- Simplified Bellman updates to calculate V for a fixed policy:
  - New V is expected one-step-look-ahead using current V
  - Unfortunately, need T and R



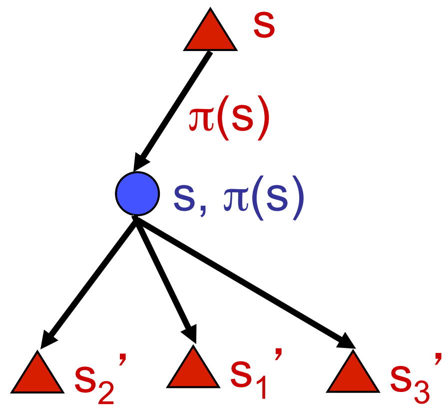
$$V_0^\pi(s) = 0$$

$$V_{i+1}^\pi(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_i^\pi(s')]$$

Sample-Based Policy Evaluation?

$$V_{i+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_i^{\pi}(s')]$$

- Who needs T and R? Approximate the expectation with samples (drawn from T!)



$$sample_1 = R(s,\pi(s),s'_1) + \gamma V_i^{\pi}(s'_1)$$

$$sample_2 = R(s,\pi(s),s'_2) + \gamma V_i^{\pi}(s'_2)$$

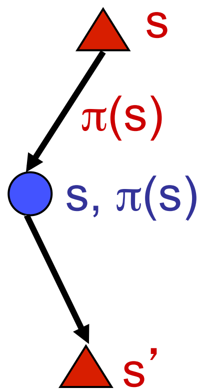
$$\ldots$$

$$sample_k = R(s,\pi(s),s'_k) + \gamma V_i^{\pi}(s'_k)$$

$$V_{i+1}^{\pi}(s) \leftarrow \frac{1}{k}\sum_i sample_i$$

Model-Difference Learning

- **Big idea:** learn from every experience!
  - Update V(s) each time we experience (s,a,s',r)
  - Likely s' will contribute updates more often



- Temporal difference learning
  - Policy still fixed!
  - Move values toward value of whatever successor occurs!

Sample of V(s): $$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

Update to V(s): $$V^{\pi}(s) \leftarrow (1 - \alpha)V^{\pi}(s) + (\alpha)sample$$

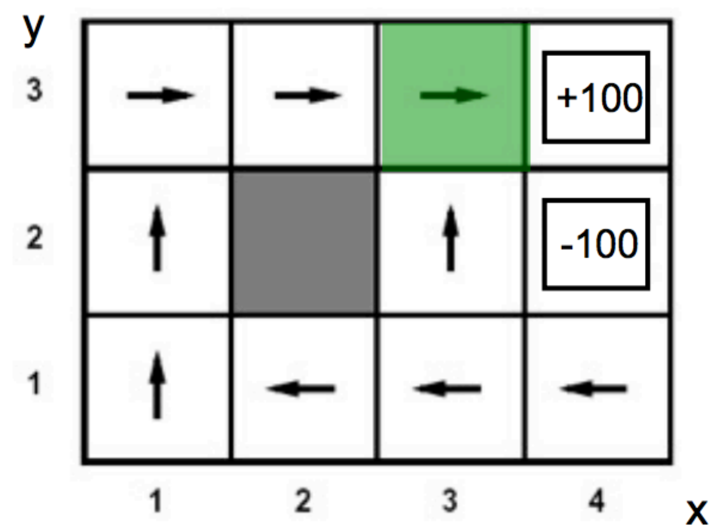
Same update: $$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$$

Example: TD Policy Evaluation

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

(1,1) up -1
(1,2) up -1
(1,2) up -1
(1,3) right -1
(2,3) right -1
(3,3) right -1
(3,2) up -1
(3,3) right -1
(4,3) exit +100
(done)

(1,1) up -1
(1,2) up -1
(1,3) right -1
(2,3) right -1
(3,3) right -1
(3,2) up -1
(4,2) exit -100
(done)



Updates for V(<3,3>):

V(<3,3>) = 0.5*0 + 0.5*[-1 + 1*0] = -0.5

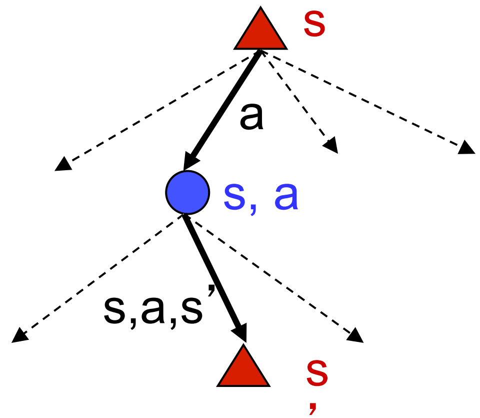
V(<3,3>) = 0.5*-0.5 + 0.5*[-1+1*100] = 49.475

V(<3,3>) = 0.5*49.475 + 0.5*[-1 + 1*-0.75]

Take γ = 1, α = 0.5, V_0(<4,3>)=100, V_0(<4,2>)=-100, V_0 = 0 otherwise

Problems with TD Value Learning

- TD value leaning is a model-free way to do policy evaluation
- However, if we want to turn values into a (new) policy, we're sunk:



$$\pi(s) = \arg\max_a Q^*(s,a)$$

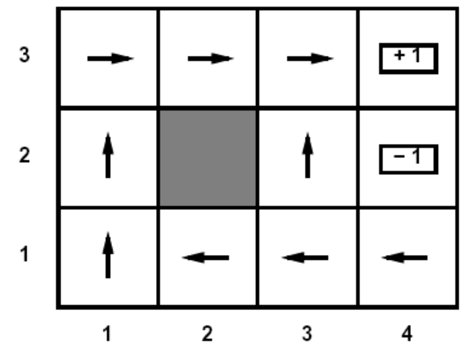
$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

- **Idea**: learn Q-values directly
- Makes action selection model-free too!

Outline

- Expectimax Search
- Reinforcement Learning (RL)
- Passive Learning in RL
  - Model-based
  - Model-free
    - Direct Estimation
    - Temporal Difference
- Active Learning in RL
  - Q-Learning

Active Learning



- Full reinforcement learning
  - You don’t know the transitions T(s,a,s’)
  - You don’t know the rewards R(s,a,s’)
  - You can choose any actions you like
  - Goal: learn the optimal policy
  - … what value iteration did!
- In this case:
  - Learner makes choices!
  - Fundamental tradeoff: exploration vs. exploitation
  - This is NOT offline planning! You actually take actions in the world and find out what happens…

Q-Learning Update

- Q-Learning: sample-based Q-value iteration

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right]$$

- Learn Q*(s,a) values
  - Receive a sample (s,a,s' ,r)
  - Consider your old estimate: $Q(s,a)$
  - Consider your new sample estimate:

$$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

  - Incorporate the new estimate into a running average:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\left[sample\right]$$

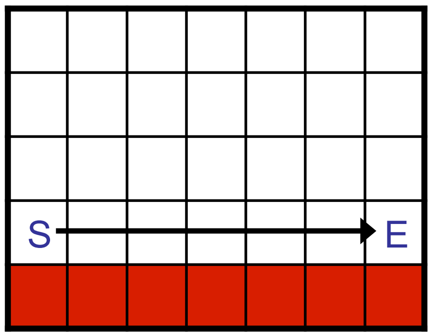
Exploration / Exploitation

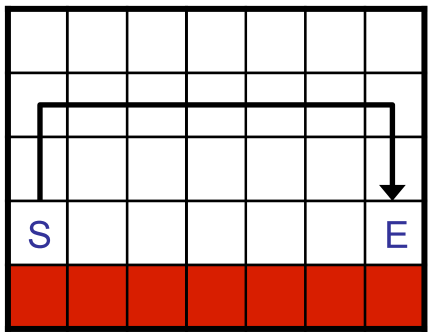
- Several schemes for forcing exploration
  - Simplest: random actions (ε greedy)
    - Every time step, flip a coin
    - With probability ε, act randomly
    - With probability $1-\varepsilon$, act according to current policy
  - Problems with random actions?
    - You do explore the space, but keep thrashing around once learning is done
    - One solution: lower ε over time
    - Another solution: exploration functions

Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy
  - If you explore enough
  - If you make the learning rate small enough
  - … but not decrease it too quickly!
  - Basically doesn't matter how you select actions (!)

- Neat property: off-policy learning
  - learn optimal policy without following it (some caveats)





RL for Helicopter Controller