CS 440: Introduction to Artificial Intelligence

Lecture 20, Nov 18, 2015

Matthew Stone

November 20, 2015

Recap—Decision principle

- Agent prefers outcome that maximizes expected utility

Recap—Decision principle

- Agent prefers outcome that maximizes expected utility

Formalism

- Choose a as

$$\operatorname*{argmax}_a EU(a|\mathbf{e})$$

Recap—Methodology

- Build prototype agent
- Build schema of possible designs
- Get experience from agent acting randomly
- Build model from schema plus experience
- Solve model for policy
- Use policy

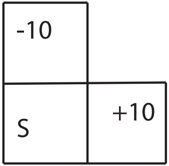
Recap—Efficient Representation

- Have set of *states*
- Have set of *actions*
- Have transition model $P(S_{i+1}|A_i, S_i)$
- Have reward function $R(S_i, A_i, S_{i+1})$
- Utility is sum of rewards, perhaps discounted into the future (1 unit of fun tomorrow is worth γ units of fun now)

Simple Illustration

Robot navigation.

- The robot can try to move in any of the cardinal directions. When the robot tries to move, there is a 40% chance it heads in the direction it wants. There is a 20% chance it heads to the right instead (90 degrees clockwise), a 20% chance it heads to the left (90 degrees counterclockwise), and a 20% chance it stays still.
- Discount factor 0.5



Model is called "Markov Decision Process"

- Describes certain situations well.
- Named for independence assumptions.
- Solution is again a policy:
  Here policy says what action to do in each state

Solving MDPs

- Same idea: Work backwards
- Jointly predict value
  Expected utility of each state
- and optimal policy
  Mapping from state to best action

Demo: `http://www.cs.ubc.ca/~poole/demos/mdp/vi.html`

Technical Concepts

Value of a state S_i—$V(S_i)$

- Expected value over the indefinite future
  starting in state S_i
  and acting optimally

Fixed-point equation

$$V(S_i) = \max_A \sum P(S_{i+1}|A, S_i)(R(S_i, A, S_{i+1}) + \gamma V(S_{i+1}))$$

Special case of definition of expected utility

Value and Q-value

- Value V gives expected outcome for each state.

$$V(S) = \max_A \sum_{S'} P(S'|A,S)(R(S,A,S') + \gamma V(S'))$$

- Q-value Q gives expected outcome for each action in each state

$$Q(S,A) = \sum_{S'} P(S'|A,S)(R(S,A,S') + \gamma V(S'))$$

or

$$Q(S,A) = \sum_{S'} P(S'|A,S)(R(S,A,S') + \gamma \max_{A'} Q(S',A'))$$

Value Iteration

Iteratively approximate the value function

- Compute series of approximations $V^k(S_i)$
- Each approximation looks further into the future

Initial Step

Set $V^0(S_i) = Q^0(S_i, A) = 0$.

- Don't worry about the future at all

Iteration

Update with

$$V^{j+1}(S_i) = \max_A \sum P(S_{i+1}|A, S_i)(R(S_i, A, S_{i+1}) + \gamma V^j(S_{i+1}))$$

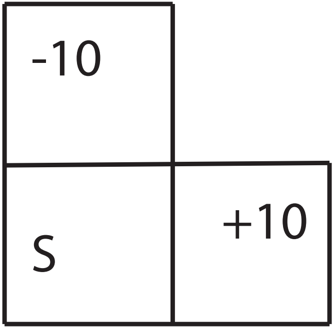
or

$$Q^{j+1}(S_i, A) = \sum P(S_{i+1}|A, S_i)(R(S_i, A, S_{i+1}) + \max_{A'} \gamma Q^j(S_{i+1}, A'))$$

Repeat until changes from V^j to V^{j+1} are small

- small changes are unlikely to lead to changes to the optimal policy

Simple Illustration – Round 1



Go right:

- $Q^1(S, \text{right}) = 0.4{*}10{-}0.2{*}10{+}0.4{*}0 = 2$

Go down:

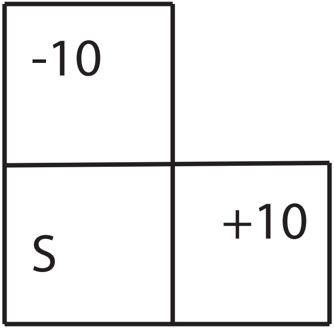
- $Q^1(S, \text{down}) = 0.2 * 10 + 0.8 * 0 = 2$

Go up:

- $Q^1(S, \text{up}) = -.4 * 10 + 0.2 * 10 + 0.4 * 0 = -2$

$V^1(S) = \max_A Q^1(S, A) = 2$

Simple Illustration – Round 2



Go right:

- $Q^2(S, \text{right}) = 0.4 * 10 - 0.2 * 10 + 0.4 * 1 = 2.4$

Go down:

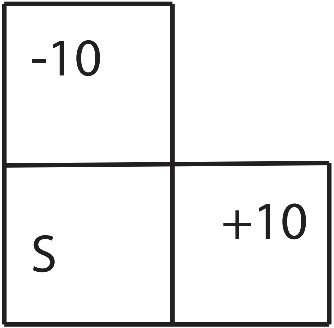
- $Q^2(S, \text{down}) = 0.2 * 10 + 0.8 * 1 = 2.8$

Go up:

- $Q^2(S, \text{up}) = -.4 * 10 + 0.2 * 10 + 0.4 * 1 = -1.6$

$V^2(S) = \max_A Q^2(S, A) = 2.8$

Simple Illustration – Round 3



Go right:

- $Q^3(S, \text{right}) = 0.4 * 10 - 0.2 * 10 + 0.4 * 1.4 = 2.56$

Go down:

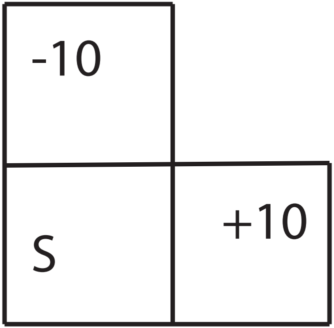
- $Q^3(S, \text{down}) = 0.2 * 10 + 0.8 * 1.4 = 3.12$

Go up:

- $Q^3(S, \text{up}) = -.4 * 10 + 0.2 * 10 + 0.4 * 1.4 = -1.44$

$V^3(S) = \max_A Q^3(S, A) = 3.12$

Simple Illustration – Round 4



Go right:

- $Q^4(S, \text{right}) = 0.4 * 10 - 0.2 * 10 + 0.4 * 1.56 = 2.624$

Go down:

- $Q^4(S, \text{down}) = 0.2 * 10 + 0.8 * 1.56 = 3.248$

Go up:

- $Q^4(S, \text{up}) = -.4 * 10 + 0.2 * 10 + 0.4 * 1.56 = -1.376$

$V^4(S) = \max_A Q^4(S, A) = 3.248$

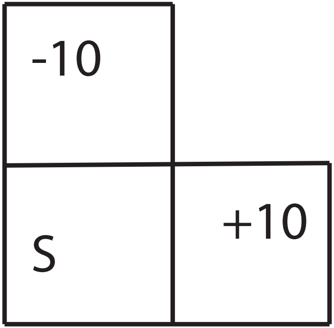
Other Techniques

Can solve for value of a given policy exactly

- Have N unknowns—values of each state
- Have N equations—fixed point equation for each state
- Solve using linear algebra
  (ie. invert a big matrix)

Simple example

Can solve for value of a given policy exactly—take go right:



- $V(S) = 0.2 * 10 + 0.8 * 0.5 * V(S)$
- $V(S) = 2 + 0.4 * V(S)$
- $0.6V(S) = 2$
- $V(S) = 10/3 = 3.333$

Other Techniques

Policy Iteration

- Start with a reasonable initial policy
- Compute the exact values for each state
- Update the policy
  Chose the action with largest expected utility in each state
  Measured using computed values
- Repeat until policy does not change