MDPs cont, Lecture 25

Nov 24 2008

Policy iteration comments

- Each step of policy iteration is guaranteed to strictly improve the policy at some state when improvement is possible
- Converge to optimal policy
- Gives exact value of optimal policy

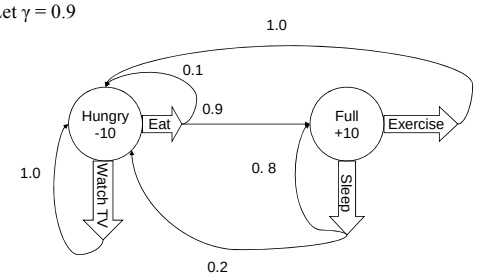
Review

- Critical components of MDPs
  - State space and action space; Transition model; Reward function
- Value iteration
  - U(S): the expected sum of maximum rewards achievable starting at a particular state
  - Bellman equation: $U(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s')$
  - Bellman iteration: $U_{i+1}(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U_i(s')$
  - Optimal policy:

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')U^*(s')$$

Policy Iteration Example

Do one iteration of policy iteration on the MDP below. Assume an initial policy of π_1(Hungry) = Eat and π_1(Full) = Sleep.
Let $\gamma = 0.9$



Review: Policy Iteration

- Start with a randomly chosen initial policy π_0
- Iterate until no change in utilities:
  1. Policy evaluation: given a policy π_i, calculate the utility $U_i(s)$ of every state s using policy π_i by solving the system of equations:

$$U_\pi(s) = R(s) + \beta \sum_{s'} T(s,\pi(s),s') \cdot U_\pi(s')$$

  2. Policy improvement: calculate the new policy π_{i+1} using one-step look-ahead based on $U_i(s)$:

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s,a,s')U_i(s')$$

Policy Iteration Example

Policy Evaluation Phase

Use initial policy for Hungry: π_1(Hungry) = Eat

U_1(Hungry) = -10 + (0.9)[(0.1)U_1(Hungry)+(0.9)U_1(Full)]
$\Rightarrow U_1$(Hungry) = -10 + (0.09)U_1(Hungry)+(0.81)U_1(Full)
$\Rightarrow$(0.91)U_1(Hungry)-(0.81)U_1(Full) = -10

Use initial policy for Full: π_1(Full) = Sleep.
U_1(Full) = 10 + (0.9)[(0.8)U_1(Full) + (0.2)U_1(Hungry)]
$\Rightarrow U_1$(Full) = 10 + (0.72)U_1(Full) + (0.18)U_1(Hungry)]
$\Rightarrow$(0.28)U_1(Full) - (0.18)U_1(Hungry) = 10

Policy Iteration Example

$(0.91)U_1(\text{Hungry})-(0.81)U_1(\text{Full}) = -10$ ....(Equation 1)
$(0.28)U_1(\text{Full}) - (0.18)U_1(\text{Hungry})=10$ ...(Equation 2)

Solve for $U_1(\text{Hungry})$ and $U_1(\text{Full})$

From Equation 1:

$(0.91)U_1(\text{Hungry}) = -10+(0.81)U_1(\text{Full})$

$\Rightarrow U_1(\text{Hungry}) = (-10/0.91)+(0.81/0.91)U_1(\text{Full})$

$\Rightarrow U_1(\text{Hungry})=-10.9+(0.89)U_1(\text{Full})$

Policy Iteration Example

$$\pi_2(\text{Full})$$

$$= \underset{\{\text{Exercise,Sleep}\}}{\text{argmax}} \begin{cases} T(\text{Full},\text{Exercise},\text{Hungry})U_1(\text{Hungry}) & [\text{Exercise}] \\ T(\text{Full},\text{Sleep},\text{Full})U_1(\text{Full}) + & \\ \quad T(\text{Full},\text{Sleep},\text{Hungry})U_1(\text{Hungry}) & [\text{Sleep}] \end{cases}$$

$$= \underset{\{\text{Exercise,Sleep}\}}{\text{argmax}} \begin{cases} (1.0)U_1(\text{Hungry}) & [\text{Exercise}] \\ (0.8)U_1(\text{Full}) + (0.2)U_1(\text{Hungry}) & [\text{Sleep}] \end{cases}$$

$$= \underset{\{\text{Exercise,Sleep}\}}{\text{argmax}} \begin{cases} (1.0)(48.7) & [\text{Exercise}] \\ (0.8)(67) + (0.2)(48.7) & [\text{Sleep}] \end{cases}$$

$$= \underset{\{\text{Exercise,Sleep}\}}{\text{argmax}} \begin{cases} 48.7 & [\text{Exercise}] \\ 63.34 & [\text{Sleep}] \end{cases}$$

$$= \text{Sleep}$$

Policy Iteration Example

$(0.91)U_1(\text{Hungry})-(0.81)U_1(\text{Full}) = -10$ ....(Equation 1)
$(0.28)U_1(\text{Full}) - (0.18)U_1(\text{Hungry})=10$ ...(Equation 2)

Solve for $U_1(\text{Hungry})$ and $U_1(\text{Full})$

Substitute $U_1(\text{Hungry})=-10.9+(0.89)U_1(\text{Full})$ into Equation 2

$(0.28)U_1(\text{Full}) - (0.18)[-10.9+(0.89)U_1(\text{Full})]=10$

$\Rightarrow (0.28)U_1(\text{Full}) + 1.96-(0.16)U_1(\text{Full})=10$

$\Rightarrow (0.12)U_1(\text{Full})=8.04$

$\Rightarrow U_1(\text{Full})=67$

$\Rightarrow U_1(\text{Hungry})=-10.9+(0.89)(67)=-10.9+59.63=48.7$

Policy Iteration Example

- $\pi_2(\text{Hungry}) = \text{Eat}$
- $\pi_2(\text{Full}) = \text{Sleep}$

Policy Iteration Example

$$\pi_2(\text{Hungry})$$

$$= \underset{\{\text{Eat,WatchTV}\}}{\text{argmax}} \begin{cases} T(\text{Hungry},\text{Eat},\text{Full})U_1(\text{Full}) + & \\ \quad T(\text{Hungry},\text{Eat},\text{Hungry})U_1(\text{Hungry}) & [\text{Eat}] \\ T(\text{Hungry},\text{WatchTV},\text{Hungry})U_1(\text{Hungry}) & [\text{WatchTV}] \end{cases}$$

$$= \underset{\{\text{Eat,WatchTV}\}}{\text{argmax}} \begin{cases} (0.9)\text{U1(Full)} + (0.1)\text{U1(Hungry)} & [\text{Eat}] \\ (1.0)\text{U1(Hungry)} & [\text{WatchTV}] \end{cases}$$

$$= \underset{\{\text{Eat,WatchTV}\}}{\text{argmax}} \begin{cases} (0.9)(67) + (0.1)(48.7) & [\text{Eat}] \\ (1.0)(48.7) & [\text{WatchTV}] \end{cases}$$

$$= \underset{\{\text{Eat,WatchTV}\}}{\text{argmax}} \begin{cases} 65.2 & [\text{Eat}] \\ 48.7 & [\text{Watch}] \end{cases}$$

$$= \text{Eat}$$

So far ....

- Given an MDP model we know how to find optimal policies
  - Value Iteration or Policy Iteration
- But what if we don't have any form of the model of the world (e.g., T, and R)
  - Like when we were babies . . .
  - All we can do is wander around the world observing what happens, getting rewarded and punished
  - This is what reinforcement learning about

Why not supervised learning

In supervised learning, we had a teacher providing us with training examples with class labels

Has Fever	Has Cough	Has Breathing Problems	Ate Chicken Recently	Has Asian Bird Flu
true	true	true	false	false
true	true	true	true	true
false	false	false	true	false

The agent figures out how to predict the class label given the features.

Reinforcement/Reward

- The key to this trial-and-error approach is having some sort of feedback about what is good and what is bad
- We call this feedback reward or reinforcement
- In some environment, rewards are frequent
  - Ping-pong: each point scored
  - Learning to crawl: forward motion
- In other environments, reward is delayed
  - Chess: reward only happens at the end of the game

Can We Use Supervised Learning?

- Now imagine a complex task such as learning to play a board game
- Suppose we took a supervised learning approach to learning an evaluation function
- For every possible position of your pieces, you need a teacher to provide an accurate and consistent evaluation of that position
  - This is not feasible

Importance of Credit Assignment

Trial and Error

- A better approach: imagine we don't have a teacher
- Instead, the agent gets to experiment in its environment
- The agent tries out actions and discovers by itself which actions lead to a win or loss
- The agent can learn an evaluation function that can estimate the probability of winning from any given position

Reinforcement

- This is very similar to what happens in nature with animals and humans
- Positive reinforcement:

  Happiness, Pleasure, Food
- Negative reinforcement:

  Pain, Hunger, Lonelinesss

What happens if we get agents to learn in this way? This leads us to the world of Reinforcement Learning

Reinforcement Learning in a nutshell

Imagine playing a new game whose rules you don't know; after a hundred or so moves, your opponent announces, "You lose".

-Russell and Norvig
Introduction to Artificial Intelligence

Reinforcement Learning

- Agent placed in an environment and must learn to behave optimally in it
- Assume that the world behaves like an MDP, except:
  - Agent can act but does not know the transition model
  - Agent observes its current state its reward but doesn't know the reward function
- Goal: learn an optimal policy

Factors that Make RL Difficult

- Actions have non-deterministic effects
  - which are initially unknown and must be learned
- Rewards / punishments can be infrequent
  - Often at the end of long sequences of actions
  - How do we determine what action(s) were really responsible for reward or punishment? (credit assignment problem)
  - World is large and complex