Reinforcement Learning & Markov Decision Processes (MDPs)

Machine Learning – CSE546
Sham Kakade
University of Washington

December 1, 2016

Announcements:

- HW4 posted
- Poster Session Thurs, Dec 8

- Today:
  - Review: EM
  - Neural nets and deep learning

Poster Session

- Thursday Dec 8, 9-11:30am
  - Please arrive 20 mins early to set up
- Everyone is expected to attend
- Prepare a poster
  - We provide poster board and pins
  - Both one large poster (recommended) and several pinned pages are OK
- Capture
  - Problem you are solving
  - Data you used
  - ML methodology
  - Results
- ***Prepare a 1-minute speech about your project***
- Two instructors will visit your poster separately
- Project Grading: scope, depth, data

Reinforcement Learning

training by feedback

Learning to act

- Reinforcement learning
- An agent
  - Makes sensor observations
  - Must select action
  - Receives rewards
    - positive for “good” states
    - negative for “bad” states

[Ng et al. ’05]

Markov Decision Process (MDP) Representation

- State space:
  - Joint state **x** of entire system
- Action space:
  - Joint action **a**= {a_1,…, a_n} for all agents
- Reward function:
  - Total reward R(**x**,**a**)
    - sometimes reward can depend on action
- Transition model:
  - Dynamics of the entire system P(**x**’|**x**,**a**)

Discount Factors

People in economics and probabilistic decision-making do this all the time.

The "Discounted sum of future rewards" using discount factor γ" is

(reward now) +

γ (reward in 1 time step) +

γ^2 (reward in 2 time steps) +

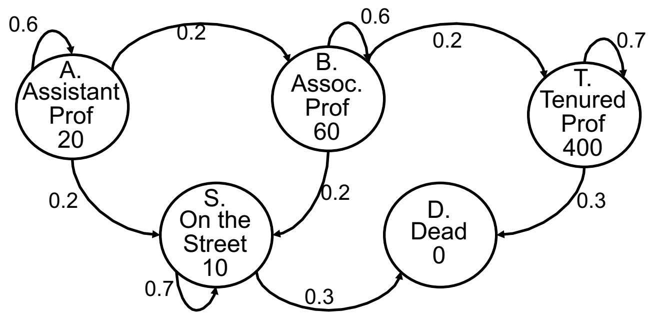
γ^3 (reward in 3 time steps) +

:

: (infinite sum)

The Academic Life

Assume Discount Factor $\gamma = 0.9$



Define:

V_A = Expected discounted future rewards starting in state A

V_B = Expected discounted future rewards starting in state B

V_T = " " " " " " " T

V_S = " " " " " " " S

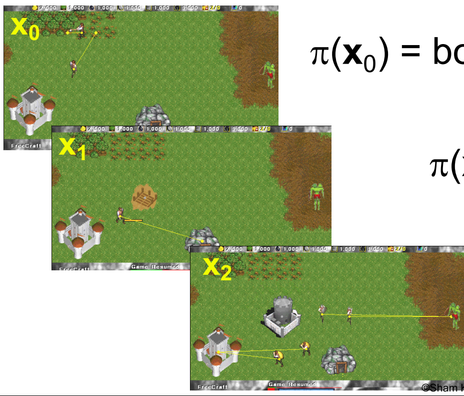
V_D = " " " " " " " D

How do we compute V_A, V_B, V_T, V_S, V_D ?

Policy

Policy: $\pi(\mathbf{x}) = \mathbf{a}$

At state **x**, action **a** for all agents



$\pi(\mathbf{x}_0)$ = both peasants get wood

$\pi(\mathbf{x}_1)$ = one peasant builds barrack, other gets gold

$\pi(\mathbf{x}_2)$ = peasants get gold, footmen attack

©Sham Kakade

9

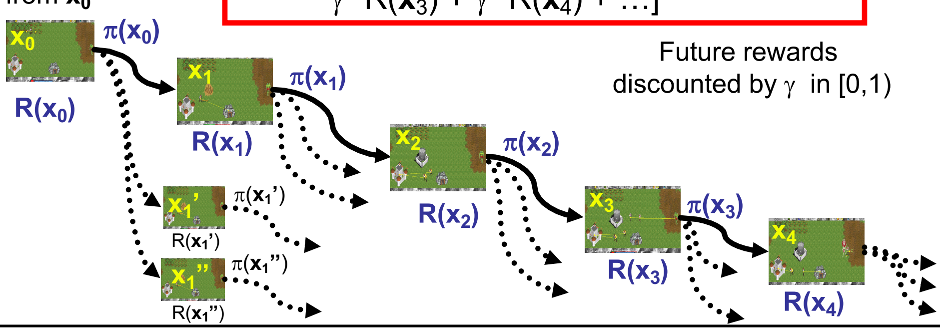
Value of Policy

Value: $V_\pi(\mathbf{x})$

Expected long-term reward starting from **x**

$$V_\pi(\mathbf{x_0}) = \mathbf{E}_\pi[R(\mathbf{x}_0) + \gamma\, R(\mathbf{x}_1) + \gamma^2\, R(\mathbf{x}_2) + \gamma^3\, R(\mathbf{x}_3) + \gamma^4\, R(\mathbf{x}_4) + \ldots]$$

Start from $\mathbf{x_0}$



Future rewards discounted by γ in [0,1)

Computing the value of a policy

$V_\pi(\mathbf{x_0}) = \mathbf{E}_\pi[R(\mathbf{x}_0) + \gamma\, R(\mathbf{x}_1) + \gamma^2\, R(\mathbf{x}_2) + \gamma^3\, R(\mathbf{x}_3) + \gamma^4\, R(\mathbf{x}_4) + \ldots]$

- Discounted value of a state:
  - value of starting from x_0 and continuing with policy π from then on

$$V_\pi(x_0) = E_\pi[R(x_0) + \gamma R(x_1) + \gamma^2 R(x_2) + \gamma^3 R(x_3) + \cdots] = E_\pi[\sum_{t=0}^{\infty} \gamma^t R(x_t)]$$

- A recursion!

Simple approach for computing the value of a policy: Iteratively

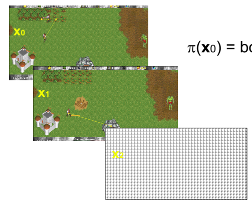
$$V_\pi(x) = R(x) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V_\pi(x')$$

- Can solve using a simple convergent iterative approach: (a.k.a. dynamic programming)
  - Start with some guess V^0
  - Iteratively say:
    - $V_\pi^{t+1}(x) \leftarrow R(x) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V_\pi^t(x')$
  - Stop when $||V_{t+1}-V_t||_\infty < \varepsilon$
    - means that $||V_\pi-V_{t+1}||_\infty < \varepsilon/(1-\gamma)$

But we want to learn a **Policy**

- So far, told you how good a policy is…
- But how can we choose the best policy???
- Suppose there was only one time step:
  - world is about to end!!!
  - select action that maximizes reward!

Policy: $\pi(\mathbf{x}) = \mathbf{a}$ → At state **x**, action **a** for all agents



$\pi(\mathbf{x}_0)$ = both peasants get wood

$\pi(\mathbf{x}_1)$ = one peasant builds barrack, other gets gold

$\pi(\mathbf{x}_2)$ = peasants get gold, footmen attack

Unrolling the recursion

- Choose actions that lead to best value in the long run
  - Optimal value policy achieves optimal value V^*

$$V^*(x_0) \;=\; \max_{a_0} R(x_0, a_0) + \gamma E_{a_0}[\max_{a_1} R(x_1) + \gamma^2 E_{a_1}[\max_{a_2} R(x_2) + \cdots]]$$

Bellman equation

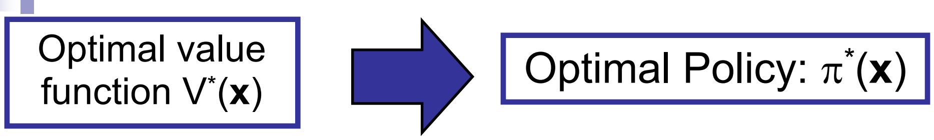
- Evaluating policy π:

$$V_\pi(x) = R(x) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V_\pi(x')$$

- Computing the optimal value V^* - Bellman equation

$$V^*(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) V^*(\mathbf{x}')$$

Optimal Long-term Plan



Optimal policy:

$$\pi^*(\mathbf{x}) = \arg\max_{a} R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) V^*(\mathbf{x}')$$

Interesting fact – Unique value

$$V^*(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V^*(\mathbf{x}')$$

- *Slightly surprising fact*: There is only one V^* that solves Bellman equation!
  - there may be many optimal policies that achieve V^*
- *Surprising fact*: optimal policies are good everywhere!!!

$$V_{\pi^*}(x) \geq V_{\pi}(x), \;\; \forall x, \;\; \forall \pi$$

Solving an MDP

Solve Bellman equation → Optimal value $V^*(\mathbf{x})$ → Optimal policy $\pi^*(\mathbf{x})$

$$V^*(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V^*(\mathbf{x}')$$

Bellman equation is non-linear!!!

Many algorithms solve the Bellman equations:

- Policy iteration [Howard ‘60, Bellman ‘57]
- Value iteration [Bellman ‘57]
- Linear programming [Manne ‘60]
- …

Value iteration (a.k.a. dynamic programming) – the simplest of all

$$V^*(x) \quad = \quad R(x,a) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V^*(x')$$

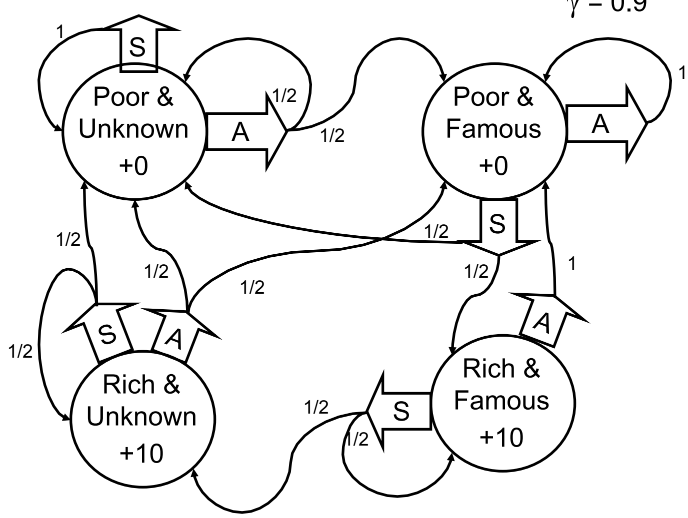
- Start with some guess V^0
- Iteratively say:
  - $V^{t+1}(x) \quad \leftarrow \quad \max_a R(x,a) + \gamma \sum_{x'} P(x' \mid x, a) V^t(x')$
- Stop when $||V_{t+1}-V_t||_\infty < \varepsilon$
  - means that $||V^*-V_{t+1}||_\infty < \varepsilon/(1-\gamma)$

A simple example

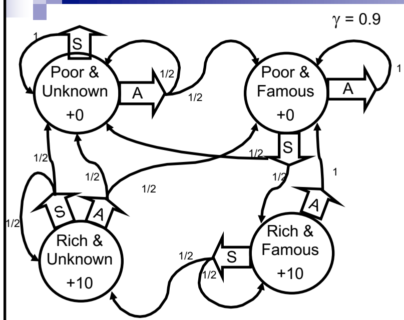
$\gamma = 0.9$

You run a startup company.

In every state you must choose between Saving money or Advertising.



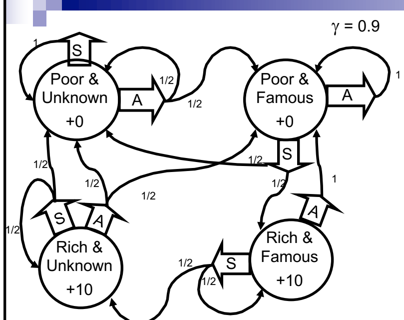
Let's compute $V_t(x)$ for our example



t	V^t(PU)	V^t(PF)	V^t(RU)	V^t(RF)
1				
2				
3				
4				
5				
6				

$$V^{t+1}(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V^{t}(\mathbf{x}')$$

Let's compute $V_t(x)$ for our example



t	V^t(PU)	V^t(PF)	V^t(RU)	V^t(RF)
1	0	0	10	10
2	0	4.5	14.5	19
3	2.03	9.46	17.44	25.08
4	5.17	13.61	20.17	29.13
5	8.45	16.91	22.88	32.19
6	11.41	19.62	25.43	34.78
∞	31.59	38.60	44.02	54.02

$$V^{t+1}(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V^{t}(\mathbf{x}')$$

What you need to know

- What's a Markov decision process
  - state, actions, transitions, rewards
  - a policy
  - value function for a policy
    - computing V_π
- Optimal value function and optimal policy
  - Bellman equation
- Solving Bellman equation
  - with value iteration, policy iteration and linear programming

Reinforcement Learning

Machine Learning – CSE546
Sham Kakade
University of Washington

December 1, 2016

The Reinforcement Learning task

World: You are in state 34.
Your immediate reward is 3. You have possible 3 actions.

Robot: I'll take action 2.
World: You are in state 77.
Your immediate reward is -7. You have possible 2 actions.

Robot: I'll take action 1.
World: You're in state 34 (again).
Your immediate reward is 3. You have possible 3 actions.

Formalizing the (online) reinforcement learning problem

- Given a set of states **X** and actions **A**
  - in some versions of the problem size of **X** and **A** unknown
- Interact with world at each time step *t*:
  - world gives state $\mathbf{x}_t$ and reward r_t
  - you give next action $\mathbf{a}_t$
- **Goal**: (quickly) learn policy that (approximately) maximizes long-term expected discounted reward

The "Credit Assignment" Problem

I'm in state 43,	reward = 0,	action = 2
" " " 39,	" = 0,	" = 4
" " " 22,	" = 0,	" = 1
" " " 21,	" = 0,	" = 1
" " " 21,	" = 0,	" = 1
" " " 13,	" = 0,	" = 2
" " " 54,	" = 0,	" = 2
" " " 26,	" = 100,	

Yippee! I got to a state with a big reward! But which of my actions along the way actually helped me get there??

This is the Credit Assignment problem.

Exploration-Exploitation tradeoff

- You have visited part of the state space and found a reward of 100
  - is this the best I can hope for???
- **Exploitation**: should I stick with what I know and find a good policy w.r.t. this knowledge?
  - at the risk of missing out on some large reward somewhere
- **Exploration**: should I look for a region with more reward?
  - at the risk of wasting my time or collecting a lot of negative reward

Two main reinforcement learning approaches

- Model-based approaches:
  - explore environment, then learn model (P(**x**'|**x**,**a**) and R(**x**,**a**)) (almost) everywhere
  - use model to plan policy, MDP-style
  - approach leads to strongest theoretical results
  - works quite well in practice when state space is manageable
- Model-free approach:
  - don't learn a model, learn value function or policy directly
  - leads to weaker theoretical results
  - often works well when state space is large