Markov Decision Processes (MDPs)

Machine Learning – 10701/15781

Carlos Guestrin
Carnegie Mellon University

April 30th, 2007

Thus far this semester

- Regression:
- Classification:
- Density estimation:

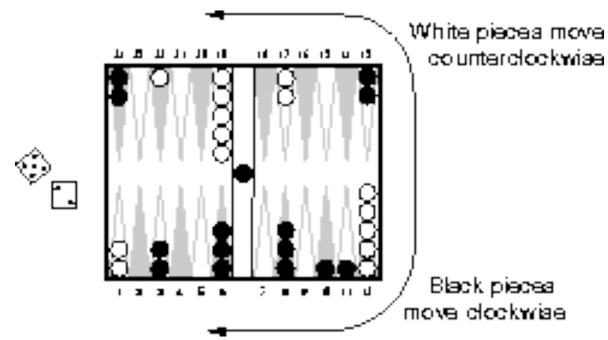
Learning to act

[Ng et al. '05]

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

Learning to play backgammon [Tesauro '95]

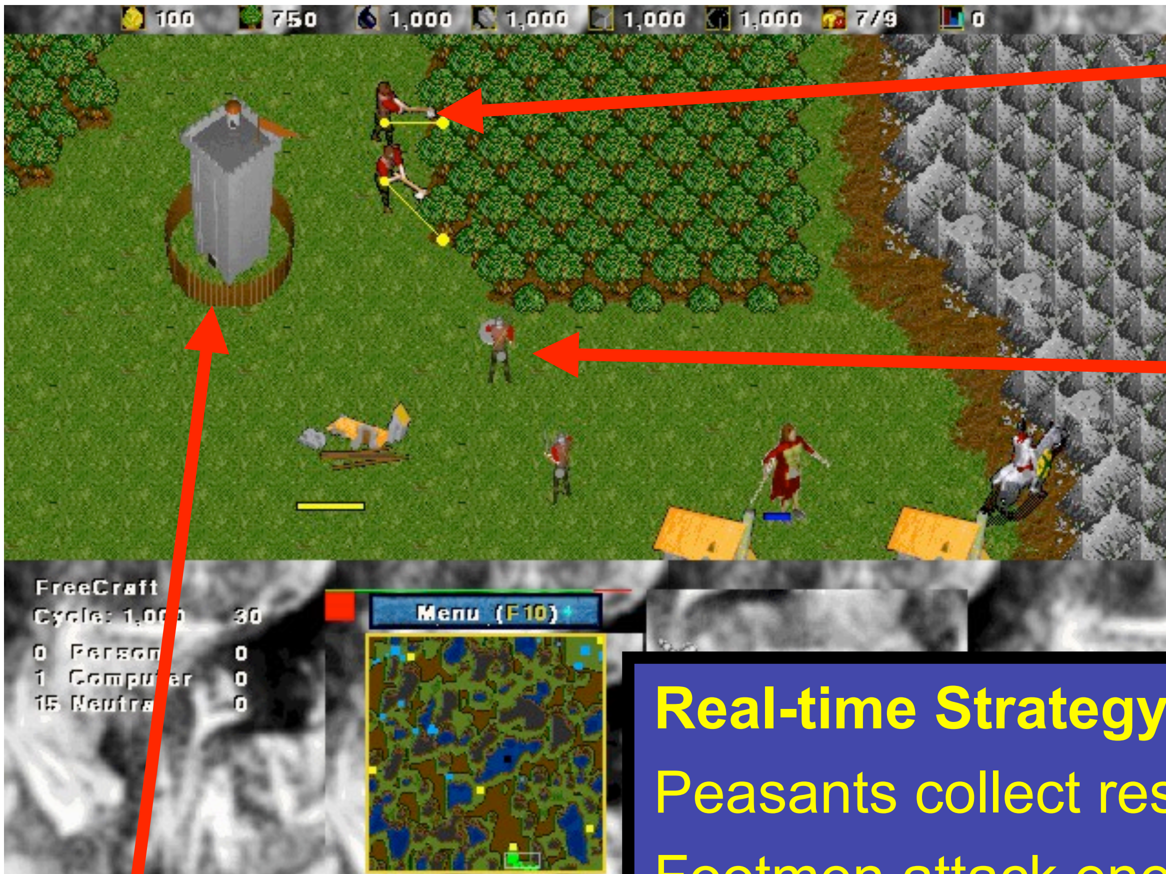
- Combines reinforcement learning with neural networks
- Played 300,000 games against itself
- Achieved grandmaster level!



Roadmap to learning about reinforcement learning

- When we learned about Bayes nets:
  - First talked about formal framework:
    - representation
    - inference
  - Then learning for BNs
- For reinforcement learning:
  - Formal framework
    - Markov decision processes
  - Then learning

FreeCraft



peasant

footman

building

Real-time Strategy Game

Peasants collect resources and build

Footmen attack enemies

Buildings train peasants and footmen

States and actions

- State space:
  - Joint state **x** of entire system
- Action space:
  - Joint action **a**= $\{a_1,\ldots, a_n\}$ for all agents

States change over time

- Like an HMM, state changes over time
- Next state depends on current state and action selected
  - e.g., action="build castle" likely to lead to a state where you have a castle
- Transition model:
  - Dynamics of the entire system P(**x**'|**x**,**a**)

Some states and actions are better than others

- Each state **x** is associated with a reward
  - positive reward for successful attack
  - negative for loss

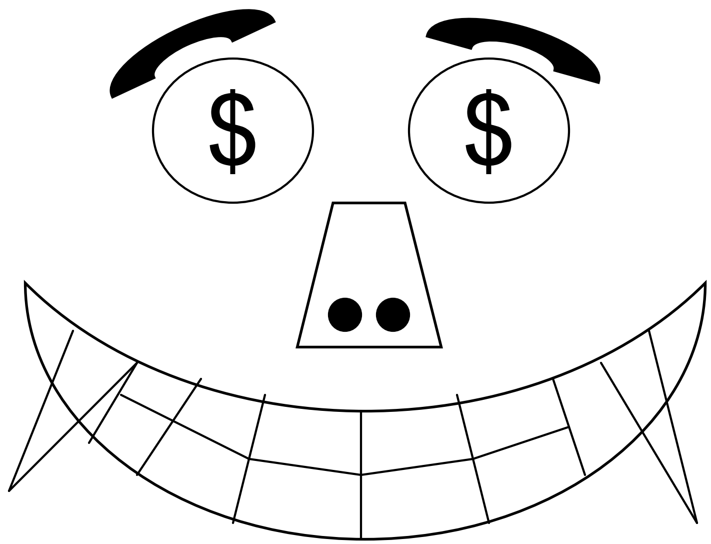
- Reward function:
  - Total reward R(**x**)

Discounted Rewards

An assistant professor gets paid, say, 20K per year.

How much, in total, will the A.P. earn in their life?

20 + 20 + 20 + 20 + 20 + … = Infinity



What's wrong with this argument?

Discounted Rewards

"A reward (payment) in the future is not worth quite as much as a reward now."

- Because of chance of obliteration
- Because of inflation

Example:

Being promised \$10,000 next year is worth only 90% as much as receiving \$10,000 right now.

Assuming payment *n* years in future is worth only $(0.9)^n$ of payment now, what is the AP's Future Discounted Sum of Rewards ?

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) +

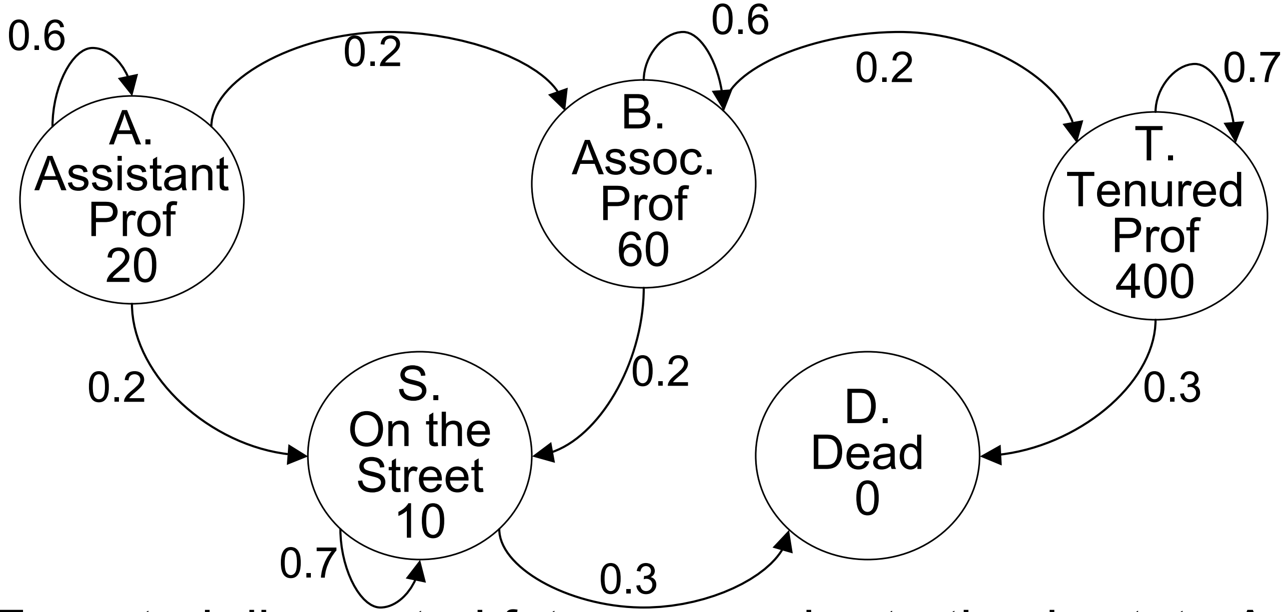
γ^3 (reward in 3 time steps) +

:

: (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

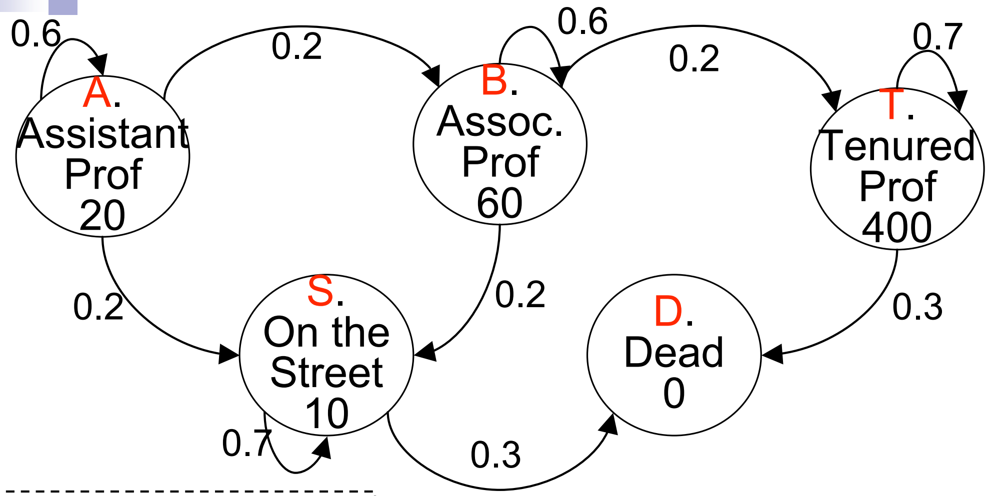
V_T = " " " " " " " T

V_S = " " " " " " " S

V_D = " " " " " " " D

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

Computing the Future Rewards of an Academic



Assume Discount Factor $\gamma = 0.9$

Joint Decision Space

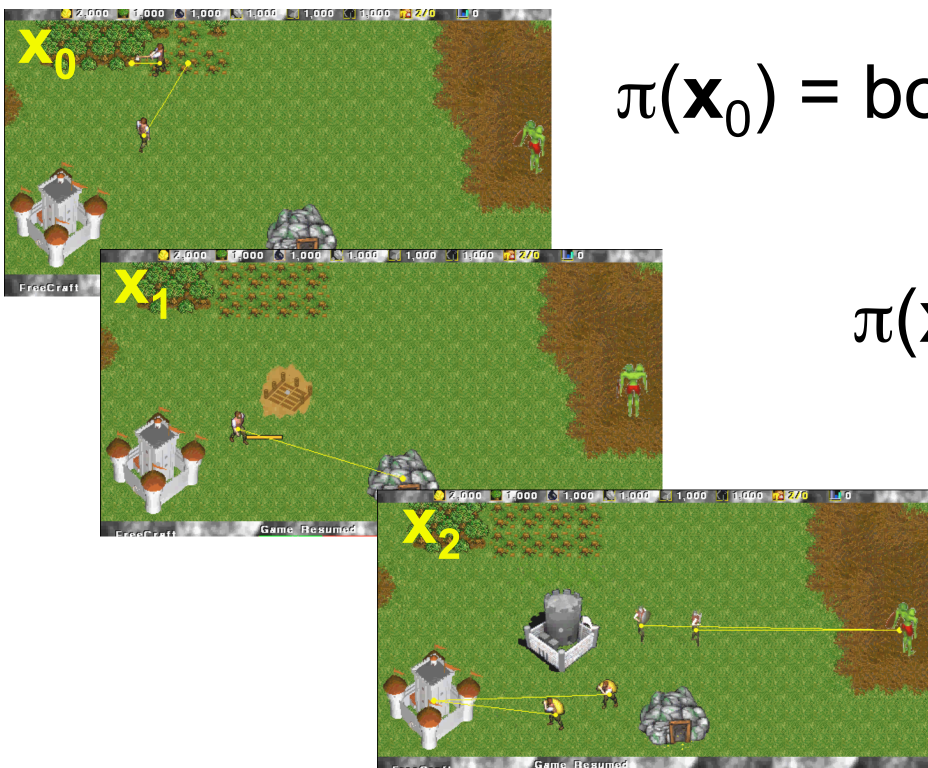
Markov Decision Process (MDP) Representation:

- 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**)

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

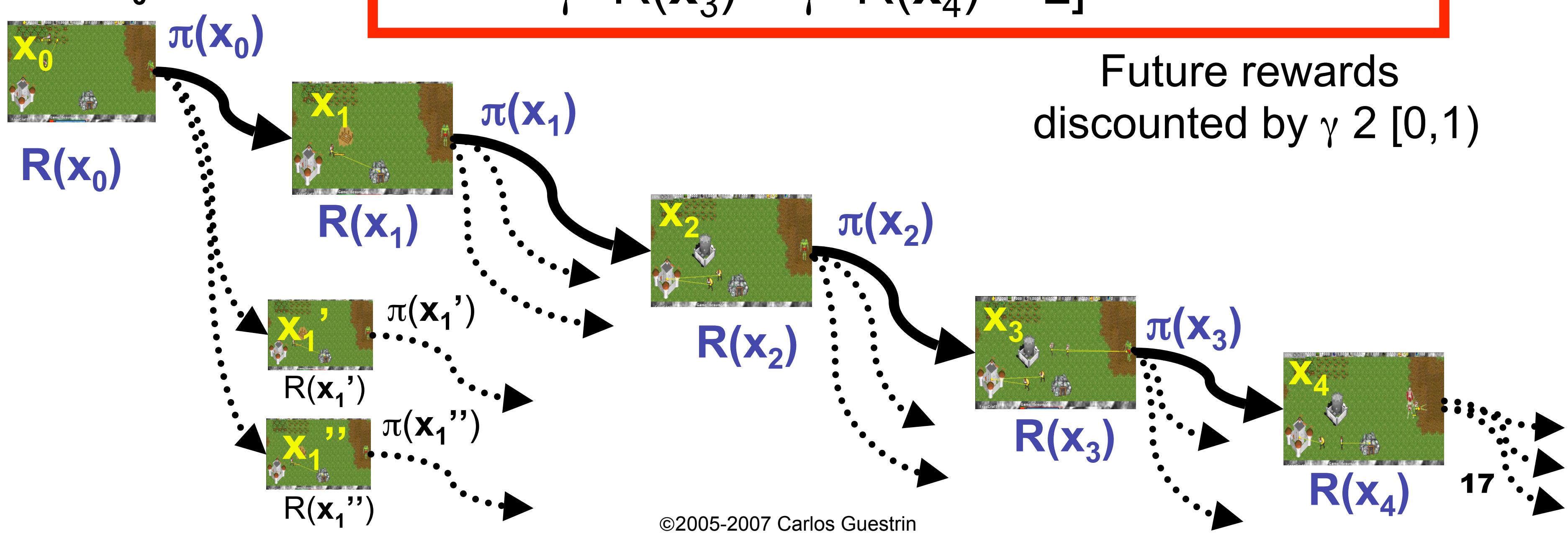
Value of Policy

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

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

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

Start from $\mathbf{x_0}$



Future rewards discounted by γ 2 [0,1)

Computing the value of a policy

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

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

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

- A recursion!

Computing the value of a policy 1 – the matrix inversion approach

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

- Solve by simple matrix inversion:

Computing the value of a policy 2 – iteratively

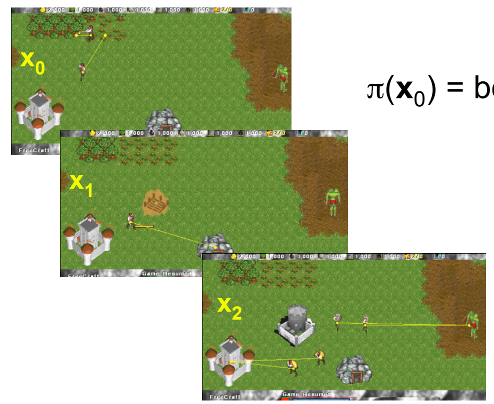
$$V_\pi(x) \;=\; R(x) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V_\pi(x')$$

- If you have 1000,000 states, inverting a 1000,000x1000,000 matrix is hard!
- Can solve using a simple convergent iterative approach: (a.k.a. dynamic programming)
  - Start with some guess V_0
  - Iteratively say:
    - $V_{t+1} = R + \gamma\, P_\pi\, V_t$
  - Stop when $||V_{t+1}\text{-}V_t||_1 \cdot \varepsilon$
    - means that $||V_\pi\text{-}V_{t+1}||_1 \cdot \varepsilon/(1\text{-}\gamma)$

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

Another recursion!

- Two time steps: address tradeoff
  - good reward now
  - better reward in the future

Unrolling the recursion

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

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

Bellman equation [Bellman 60]

- 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 value function $V^*(\mathbf{x})$ → Optimal Policy: $\pi^*(\mathbf{x})$

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

Optimal policy:

$$\pi^*(\mathbf{x}) = \arg\max_{\mathbf{a}} Q^*(\mathbf{x},\mathbf{a})$$

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

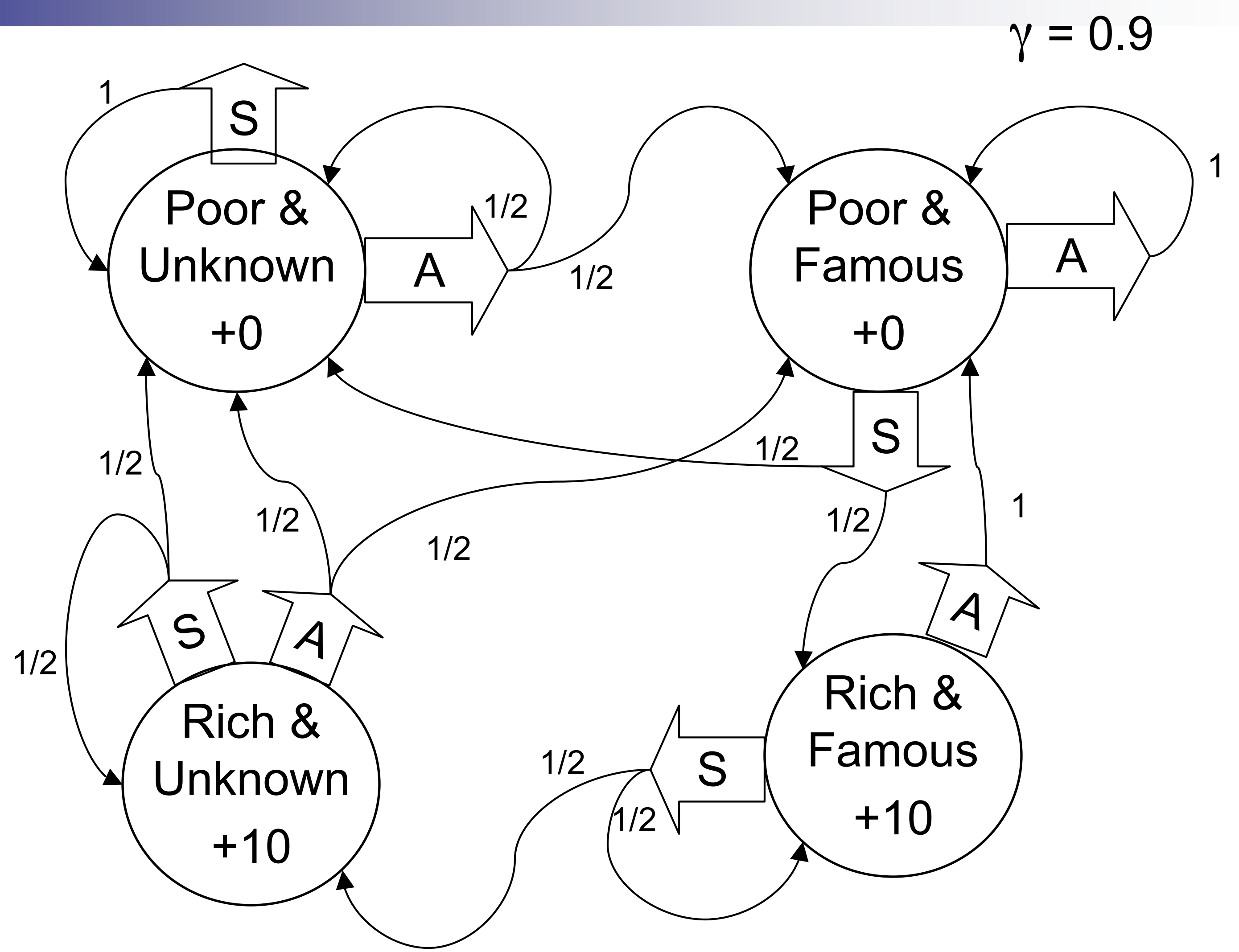
$$V^*(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V^*(\mathbf{x}')$$

- Start with some guess V_0
- Iteratively say:
  - $$V_{t+1}(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V_t(\mathbf{x}')$$
- Stop when $||V_{t+1}-V_t||_1 \cdot \varepsilon$
  - means that $||V^*-V_{t+1}||_1 \cdot \varepsilon/(1-\gamma)$

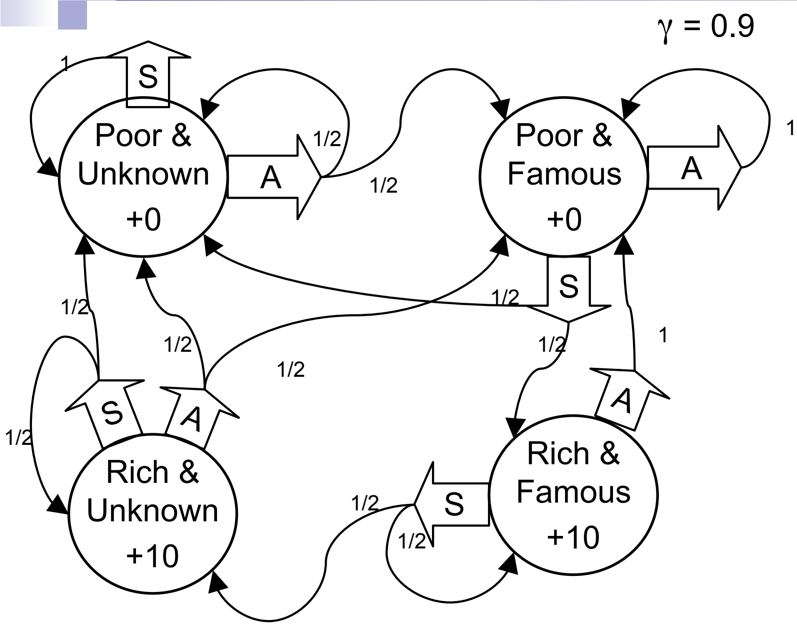
A simple example

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

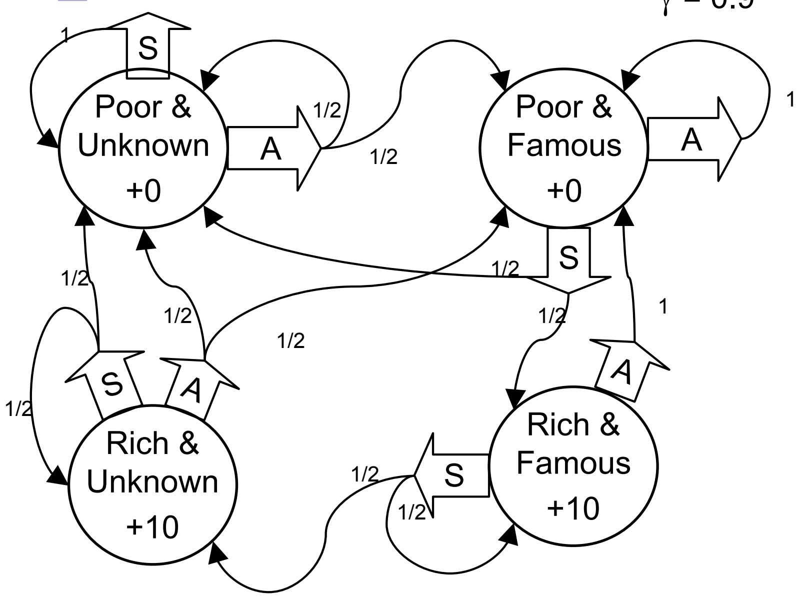


t	V_t(PU)	V_t(PF)	V_t(RU)	V_t(RF)
1				
2				
3				
4				
5				
6				

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

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

$\gamma = 0.9$



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	6.53	25.08	18.55
4	3.852	12.20	29.63	19.26
5	7.22	15.07	32.00	20.40
6	10.03	17.65	33.58	22.43

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

Policy iteration – Another approach for computing π^*

- Start with some guess for a policy π_0
- Iteratively say:
  - evaluate policy:

$$V_t(\mathbf{x}) = R(\mathbf{x}, \mathbf{a} = \pi_t(\mathbf{x})) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a} = \pi_t(\mathbf{x})) V_t(\mathbf{x}')$$

  - improve policy:

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

- Stop when
  - policy stops changing
    - usually happens in about 10 iterations
  - or $||V_{t+1}-V_t||_1 \cdot \varepsilon$
    - means that $||V^*-V_{t+1}||_1 \cdot \varepsilon/(1-\gamma)$

Policy Iteration & Value Iteration: Which is best ???

It depends.

Lots of actions? Choose Policy Iteration

Already got a fair policy? Policy Iteration

Few actions, acyclic? Value Iteration

Best of Both Worlds:

Modified Policy Iteration [Puterman]

…a simple mix of value iteration and policy iteration

3rd Approach

Linear Programming

LP Solution to MDP

[Manne '60]

Value computed by linear programming:

$$\text{minimize}: \sum_{\mathbf{x}} V(\mathbf{x})$$

$$\text{subject to}: \begin{cases} V(\mathbf{x}) \geq R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) V(\mathbf{x}') \\ \forall \mathbf{x}, \mathbf{a} \end{cases}$$

- One variable $V(\mathbf{x})$ for each state
- One constraint for each state **x** and action **a**
- **Polynomial time solution**

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

Acknowledgment

- This lecture contains some material from Andrew Moore's excellent collection of ML tutorials:
  - http://www.cs.cmu.edu/~awm/tutorials

Reinforcement Learning

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

April 30th, 2007

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 ! 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