Markov Decision Processes (MDPs)

Machine Learning – CSE446

Carlos Guestrin
University of Washington

June 3, 2013

Reinforcement Learning

training by feedback

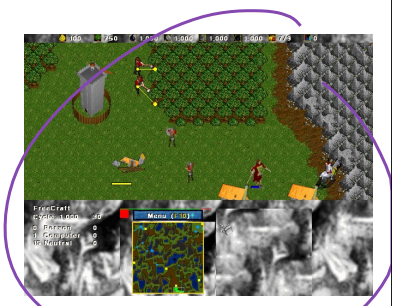
Learning to act

- 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**)



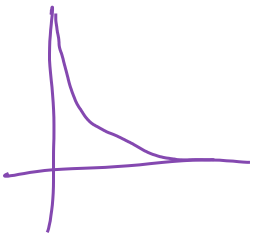
Discount Factors

$\gamma \in [0,1)$

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

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

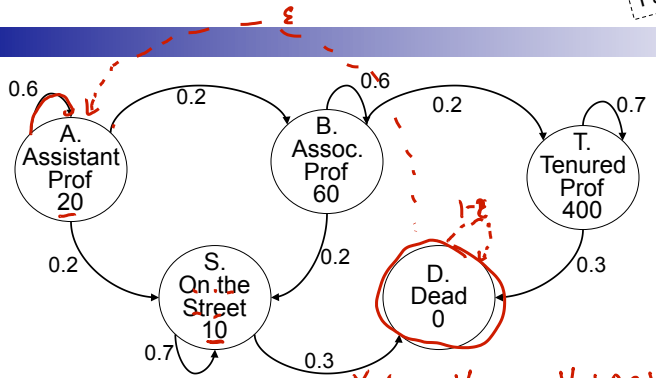
(reward now) +
γ (reward in 1 time step) +
γ^2 (reward in 2 time steps) +
γ^3 (reward in 3 time steps) +
:
: (infinite sum)



The Academic Life

n States

Assume Discount Factor $\gamma = 0.9$



Define:

V_A = Expected discounted future rewards starting in state A $= 20 + \gamma(0.6 V_A + 0.2 V_B + 0.2 V_S)$

V_B = Expected discounted future rewards starting in state B $= 60 + \gamma(0.6 V_B + 0.2 V_S + 0.2 V_T)$

V_T = " " " " " " " T

V_S = " " " " " " " S

V_D = " " " " " " " D

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

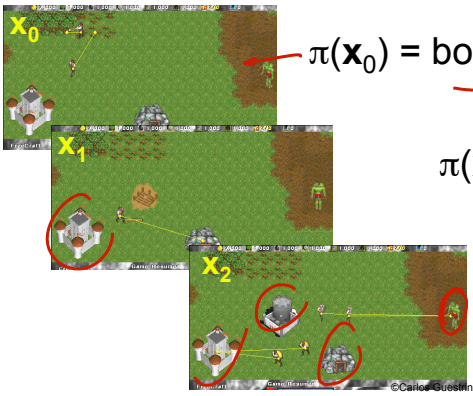
7 eqns, n unknowns, linear

linear system of eqns: matrix inversion

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

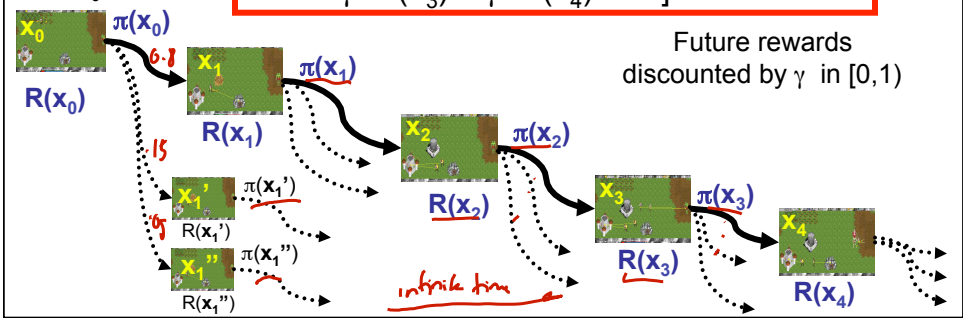
Value of Policy

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

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

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

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!

$$V_\pi(x_0) = E_\pi[\sum_{t=0}^{\infty} \gamma^t R(x_t)] = E_\pi[R(x_0) + \gamma \sum_{t=1}^{\infty} \gamma^{t-1} R(x_t)]$$
$$= R(x_0) + \gamma\, E_\pi[R(x_1) + \sum_{t=2}^{\infty} \gamma^{t-1} R(x_t)]$$
$$= R(x_0) + \gamma \sum_{x'} P(x' \mid x_0, \pi(x_0))\, V_\pi(x')$$

$V_\pi(x_1)$

linear: n equations, n unknowns

matrix inversion is your friend!!

Simple approach for computing the value of a policy: Iteratively

recursion from previous slide

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

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

estimate from previous iteration

  - Stop when $||V_{t+1} - V_t||_\infty < \varepsilon$
    - means that $||V_\pi - V_{t+1}||_\infty < \varepsilon/(1-\gamma)$

estimate is close to true value

But we want to learn a **Policy**

- So far, told you how good a policy is…
- But how can we choose the best 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

- Suppose there was only one time step:
  - world is about to end!!!
  - select action that maximizes reward!

$V^{t+1}(x) = 0 \ \forall x$

$V(x) = \max_a R(x,a)$

Unrolling the recursion

n states

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

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

$V^*(x_1)$

$$V^*(x_0) = \max_{a_0} R(x_0,a_0) + \gamma E_{a_0}[V^*(x_1)]$$

$$= \max_{a_0} R(x_0,a_0) + \gamma \sum_{x'} P(x'|x_0,a_0)\, V^*(x')$$

↑ not linear

n unknowns $V^*(x)$

n equations but not linear,

matrix inversion is not your friend !!

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

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

if I know V^*:

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

↑ act optimally

Optimal Long-term Plan

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

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

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

- *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!!!

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

Solving an MDP

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

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

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

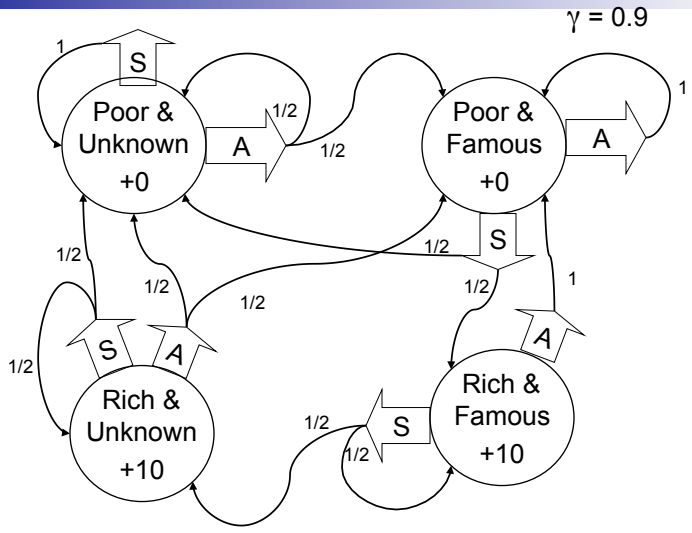
$$V^*(x) = R(x,a) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V^*(x')$$

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

A simple example

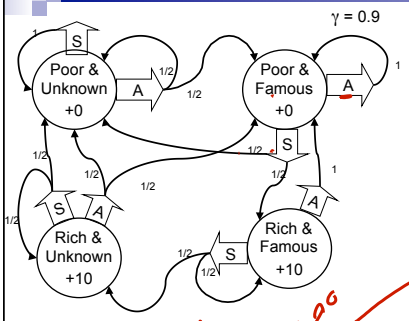
You run a startup company.

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



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

$R(x,a) = R(x)$

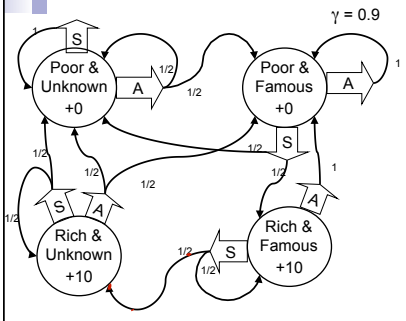


t	V^t(PU)	V^t(PF)	V^t(RU)	V^t(RF)
1	0	0	10	10
2		4.5		
3				
4				
5				
6				

$V^2(PF) = \max \begin{cases} A: 0 + \gamma V^1(PF) = 0 \\ S: 0 + \gamma\left(\frac{1}{2}V^1(PU) + \frac{1}{2}V^1(RF)\right) = 0.9\left(\frac{1}{2}\cdot 0 + \frac{1}{2}10\right) = 4.5 \end{cases}$

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

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



t	V^t(PU)	V^t(PF)	V^t(RF)	V^t(RU)
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
⋮				

Here to be RF

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

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

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 – CSE446

Carlos Guestrin
University of Washington

June 5, 2013

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