Reading:
Kaelbling et al. 1996 (see class website)

Reinforcement Learning

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

May 3rd, 2006

Announcements

- Project:
  - Poster session: Friday May 5th 2-5pm, NSH Atrium
    - please arrive a little early to set up
    - posterboards, easels, and pins provided
    - class divided into two shift so you can see other posters
- FCEs!!!!
  - Please, please, please, please, please, please give us your feedback, it helps us improve the class! ☺
    - http://www.cmu.edu/fce

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

$$x_0 \xrightarrow{a_0} x_1 \xrightarrow{a_1} x_2 \; - - - \cdots$$
$$r_0 \qquad r_1 \qquad r_2$$

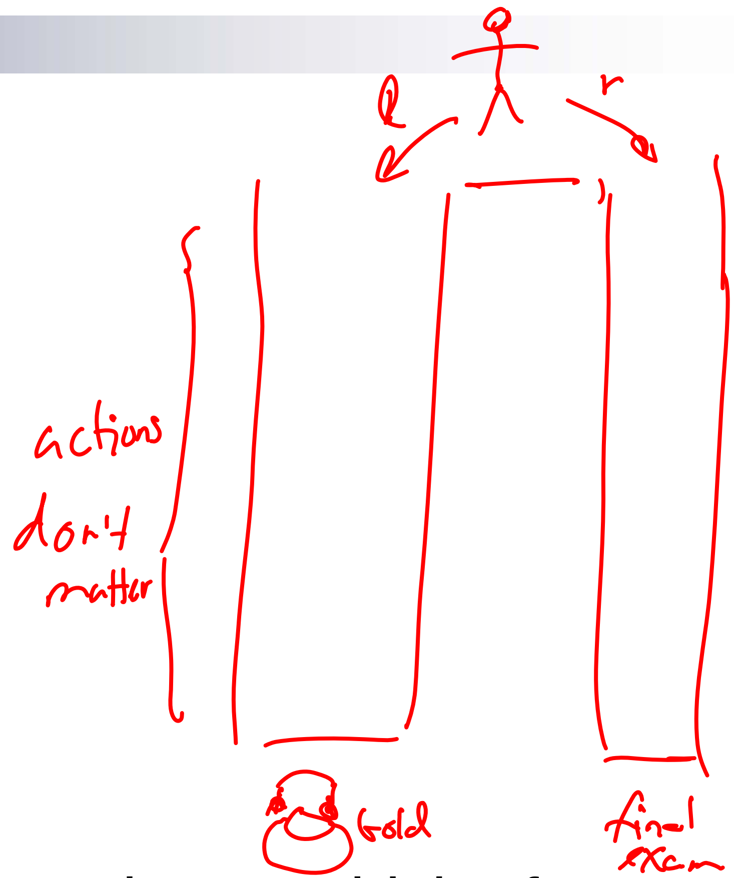
- Interact with world at each time step *t*:
  - world gives state $\mathbf{x}_t$ and reward r_t
  - you give next action $\mathbf{a}_t$

$\langle x_0, r_0, a_0 \rangle$
$\langle x_1, r_1, a_1 \rangle$
$\langle x_2, r_2, a_2 \rangle$
$\vdots$

- **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.

$P(x' | x, a)$ is unknown

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?
  of state space
  - at the risk of wasting my time or collecting a lot of negative reward

Two main reinforcement learning approaches

- Model-based approaches:
  - explore environment $\rightarrow$ learn model ($P(\mathbf{x}'|\mathbf{x},\mathbf{a})$ and $R(\mathbf{x},\mathbf{a})$) (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:

  V^* π^*

  - don't learn a model $\rightarrow$ learn value function or policy directly
  - leads to weaker theoretical results
  - often works well when state space is large

Brafman & Tennenholtz 2002
(see class website)

Rmax – A model-based approach

Given a dataset – learn model

Given data, learn (MDP) Representation:

- Dataset: $x_1 a_1 r_1 \rightarrow x_2 a_2 r_2 \rightarrow x_3 a_3 r_3$

- Learn reward function:
  - R(**x**,**a**) = when I visit x, a at time t, set $R(x,a) = r_t$
- Learn transition model:
  - P(**x**'|**x**,**a**) $= \dfrac{\text{count}(x, a, x')}{\text{count}(x, a)}$

Some challenges in model-based RL 1: Planning with insufficient information

- Model-based approach:
  - estimate $R(\mathbf{x},\mathbf{a})$ & $P(\mathbf{x}'|\mathbf{x},\mathbf{a})$
  - obtain policy by value or policy iteration, or linear programming
  - No credit assignment problem → learning model, planning algorithm takes care of "assigning" credit
- What do you plug in when you don't have enough information about a state?

$R(\hat{x},\hat{a})$? if never visit $\hat{x},\hat{a}$

  - don't reward at a particular state
    - plug in smallest reward (R_{min})? never visit $\hat{x},\hat{a}$
    - plug in largest reward (R_{max})? actively try to visit $\hat{x},\hat{a}$,

don't explore good states

but $\hat{x},\hat{a}$ could be bad...

  - don't know a particular transition probability?

$P(x'|\hat{x},\hat{a})$?

Some challenges in model-based RL 2: Exploration-Exploitation tradeoff

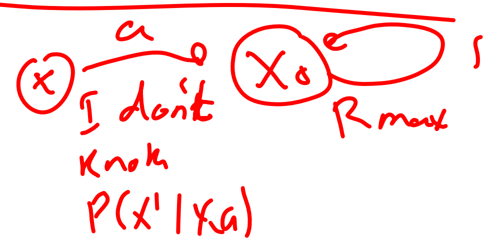
- A state may be very hard to reach
  - waste a lot of time trying to learn rewards and transitions for this state
  - after a much effort, state may be useless

- A strong advantage of a model-based approach:
  - you know which states estimate for rewards and transitions are bad
  - can (try) to plan to reach these states
  - have a good estimate of how long it takes to get there

A surprisingly simple approach for model based RL – The Rmax algorithm [Brafman & Tennenholtz]

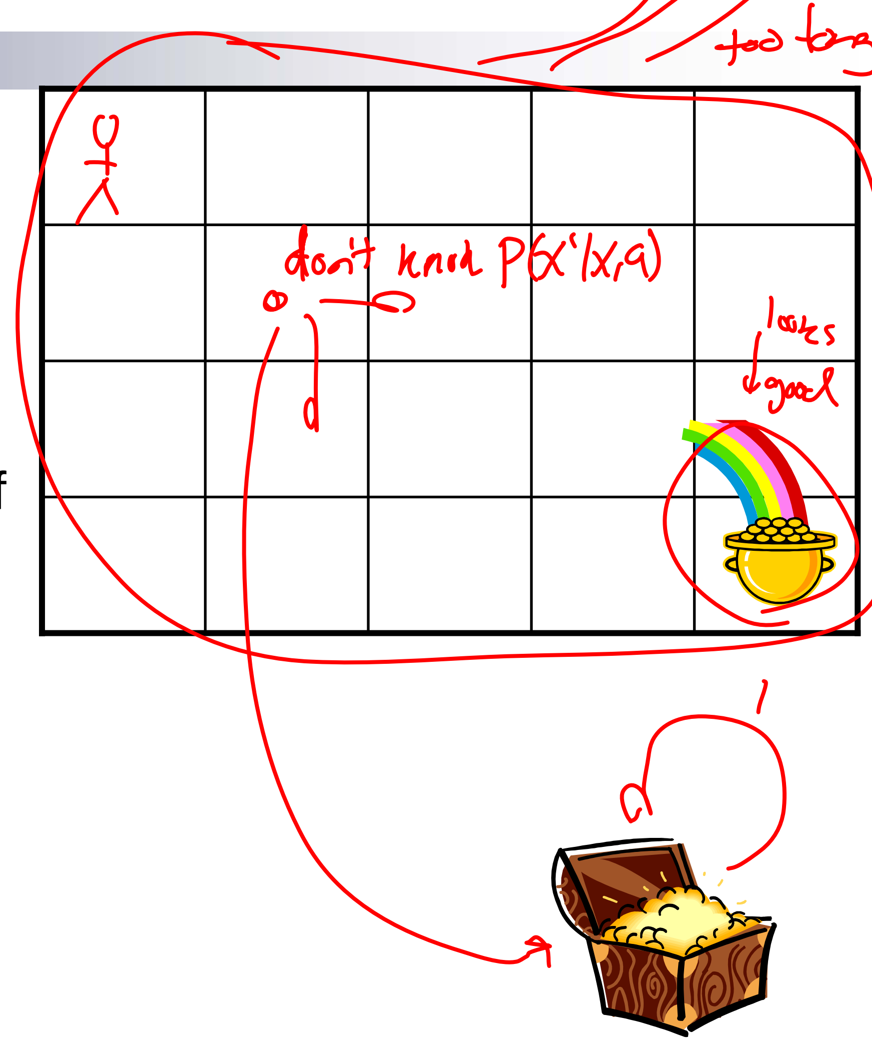
- **Optimism in the face of uncertainty!!!!**
  - heuristic shown to be useful long before theory was done (e.g., Kaelbling '90)
- If you don't know reward for a particular state-action pair, set it to R_{max}!!!

- If you don't know the transition probabilities $P(\mathbf{x'}|\mathbf{x},\mathbf{a})$ from some some state action pair **x,a** assume you go to **a magic, fairytale** new state $\mathbf{x}_0$!!!
  - $R(\mathbf{x}_0,\mathbf{a}) = R_{max}$
  - $P(\mathbf{x}_0|\mathbf{x}_0,\mathbf{a}) = 1$



Understanding R_{max}

- With R_{max} you either:
  - **explore** – visit a state-action pair you don't know much about
    - because it seems to have lots of potential
  - **exploit** – spend all your time on known states
    - even if unknown states were amazingly good, it's not worth it
- Note: you never know if you are exploring or exploiting!!!



Implicit Exploration-Exploitation Lemma

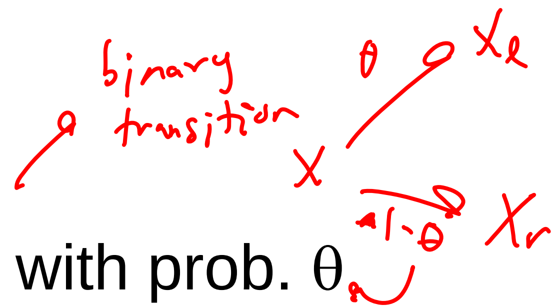
- **Lemma**: every T time steps, either:
  - **Exploits**: achieves near-optimal reward for these T-steps, or
  - **Explores**: with high probability, the agent visits an unknown state-action pair
    - learns a little about an unknown state
  - T is related to *mixing time* of Markov chain defined by MDP
    - time it takes to (approximately) forget where you started

The Rmax algorithm

- **Initialization**:
  - Add state $\mathbf{x}_0$ to MDP
  - $R(\mathbf{x},\mathbf{a}) = R_{max}, \forall \mathbf{x},\mathbf{a}$
  - $P(\mathbf{x_0}|\mathbf{x},\mathbf{a}) = 1, \forall \mathbf{x},\mathbf{a}$
  - all states (except for $\mathbf{x}_0$) are **unknown**
- Repeat
  - obtain policy for current MDP and Execute policy
  - for any visited state-action pair, set reward function to appropriate value
  - if visited some state-action pair **x**,**a** enough times to estimate $P(\mathbf{x'}|\mathbf{x},\mathbf{a})$
    - update transition probs. $P(\mathbf{x'}|\mathbf{x},\mathbf{a})$ for **x**,**a** using MLE
    - recompute policy

Visit enough times to estimate P(**x'**|**x**,**a**)?

- How many times are enough?
  - use Chernoff Bound!



- **Chernoff Bound**:
  - $X_1,\ldots,X_n$ are i.i.d. Bernoulli trials with prob. θ
  - $P(|1/n \sum_i X_i - \theta| > \varepsilon) \leq \exp\{-2n\varepsilon^2\}$

Putting it all together

- **Theorem**: With prob. at least 1-δ, Rmax will reach a ε-optimal policy in time polynomial in: num. states, num. actions, T, 1/ε, 1/δ

  |X|

  |A|

  - Every T steps:

    Because of Implicit Explore-Exploit-Lemma

    - achieve near optimal reward (great!), or
    - visit an unknown state-action pair $\rightarrow$ num. states and actions is finite, so can't take too long before all states are known

      (almost)

Problems with model-based approach

- If state space is large
  - transition matrix is very large! $|X|^2 \cdot |A|$
  - requires many visits to declare a state as know

Chernoff is loose

- Hard to do "approximate" learning with large state spaces
  - some options exist, though

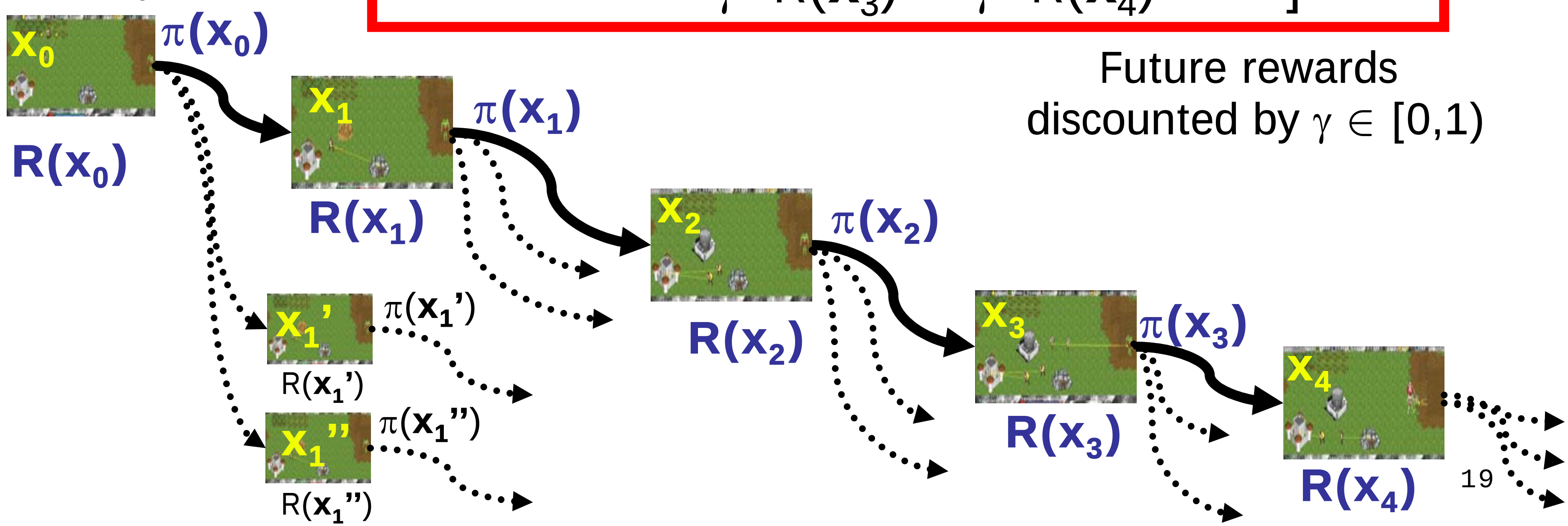
TD-Learning and Q-learning – Model-free approaches

Value of Policy

Value: $V_\pi(\mathbf{x})$ → Expected long-term reward starting from **x**

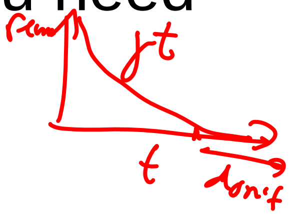
$$V_\pi(\mathbf{x_0}) = \mathbf{E}_\pi[R(\mathbf{x}_0) + \gamma\, R(\mathbf{x}_1) + \gamma^2\, R(\mathbf{x}_2) + \gamma^3\, R(\mathbf{x}_3) + \gamma^4\, R(\mathbf{x}_4) + \cdots]$$

Future rewards discounted by $\gamma \in [0,1)$



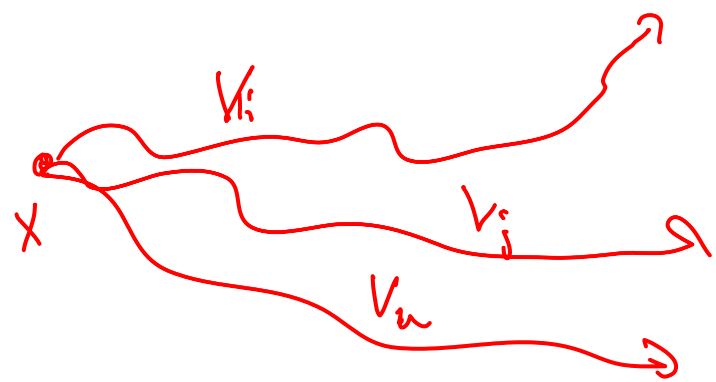
A simple monte-carlo policy evaluation

- Estimate $V_\pi(\mathbf{x})$, start several trajectories from $\mathbf{x} \rightarrow V_\pi(\mathbf{x})$ is average reward from these trajectories
  - Hoeffding's inequality tells you how many you need
  - discounted reward $\rightarrow$ don't have to run each trajectory forever to get reward estimate



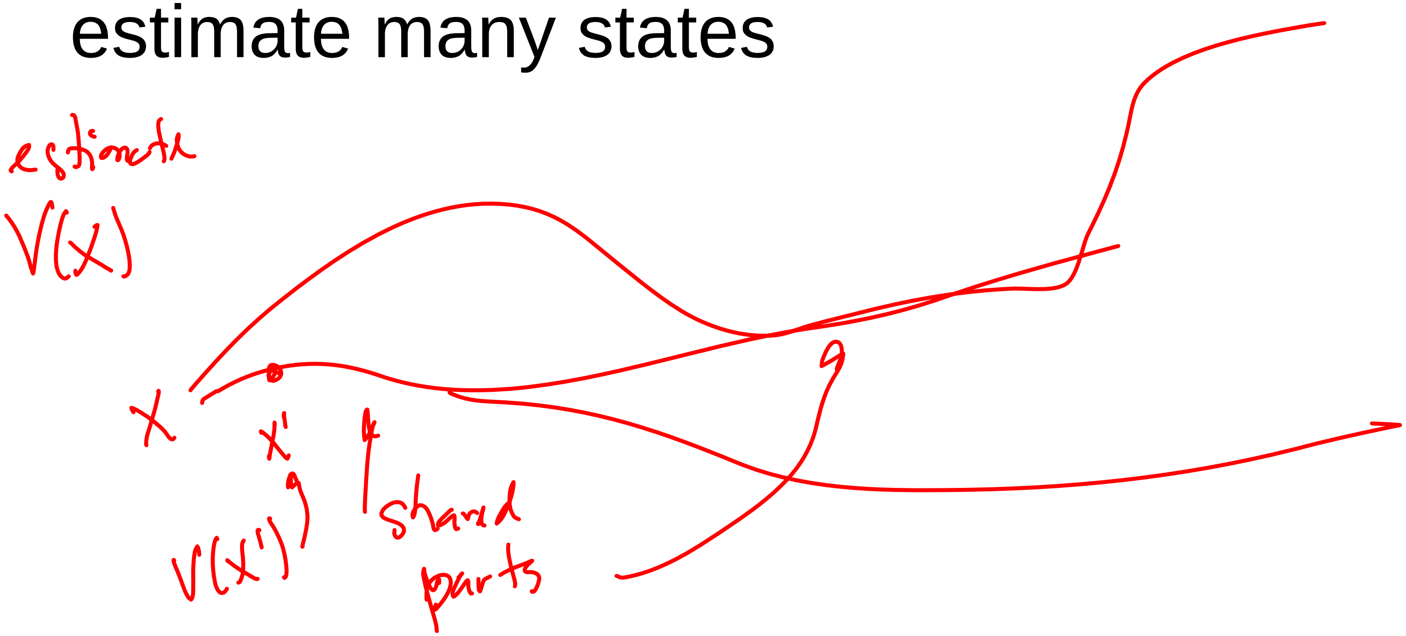
play game from x, following π, K times, each time $V_i = \sum_{t=0}^{T} \gamma^t r_t$

$$V_\pi(x) \approx \frac{1}{K} \sum_{i=1}^{K} V_i$$



Problems with monte-carlo approach

- **Resets**: assumes you can restart process from same state many times
- **Wasteful**: same trajectory can be used to estimate many states



Reusing trajectories

discounted expected Value of next state

- Value determination:

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

- Expressed as an expectation over next states:

$$V_\pi(x) \;=\; R(x) + \gamma E\left[V_\pi(x') \mid x, a = \pi(x)\right]$$

- Initialize value function (zeros, at random,…) V_0
- Idea 1: Observe a transition: $\mathbf{x}_t \rightarrow \mathbf{x}_{t+1}, \mathbf{r}_{t+1}$, approximate expec. with single sample:

$$V(x_t) = R(x_t) + \gamma \cdot V(x_{t+1})$$

$V(x_t)$

$x_t \rightarrow x_{t+1}$

  - unbiased!!
  - but a very bad estimate!!!

high variance!!
one sample

Simple fix: Temporal Difference (TD) Learning [Sutton '84]

1 3 2 7 -5 ....

exponentially decaying Moving average: $\bar{X}_t = (1-\alpha)\bar{X}_{t-1} + \alpha \cdot x_t$

$$V_\pi(x) \;=\; R(x) + \gamma E\left[V_\pi(x') \mid x, a = \pi(x)\right]$$

- Idea 2: Observe a transition: $\mathbf{x_t} \rightarrow \mathbf{x_{t+1}}, \mathbf{r_{t+1}}$, approximate expectation by mixture of new sample with old estimate:

$$V_{t+1}(x_t) = \underbrace{\alpha\left(r_t + \gamma V_t(x_{t+1})\right)}_{\text{a little of new}} + \underbrace{(1-\alpha)\, V_t(x_t)}_{\text{a lot of the old estimate}}$$

$V_t \leftarrow$ estimate of value function at time t

  - $\alpha>0$ is learning rate

α is a parameter of the algorithm

trajectory

update at x_1

$x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow x_3$ . . . . .

update V at state x_0

TD converges (can take a long time!!!)

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

V $V_\pi(x)$

as $t \to \infty$ to value of policy π

- **Theorem**: TD converges in the limit (with prob. 1), if:
  - every state is visited infinitely often
  - Learning rate decays just so:
    - $\sum_{i=1}^{\infty} \alpha_i = \infty$
    - $\sum_{i=1}^{\infty} \alpha_i^2 < \infty$

$\alpha_i = \frac{1}{i}$

Using TD for Control

- TD converges to value of current policy π_t

run TD for T steps

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

- Policy improvement:

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

- TD for control:
  - run T steps of TD
  - compute a policy improvement step

Problems with TD

- How can we do the policy improvement step if we don't have the model?

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

don't know

- TD is an **on-policy** approach: execute policy π_t trying to learn V_t
  - must visit all states infinitely often
  - What if policy doesn't visit some states???

Another model-free RL approach: Q-learning [Watkins & Dayan '92]

- Simple modification to TD

- Learns optimal value function (and policy), not just value of fixed policy

- Solution (almost) independent of policy you execute!

Recall Value Iteration

- Value iteration: $V_{t+1}(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' | \mathbf{x},\mathbf{a}) V_t(\mathbf{x}')$

- Or: $Q_{t+1}(\mathbf{x},\mathbf{a}) = R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' | \mathbf{x},\mathbf{a}) V_t(\mathbf{x}')$

  $V_{t+1}(\mathbf{x}) = \max_{\mathbf{a}} Q_{t+1}(\mathbf{x},\mathbf{a})$

  $V_t(x') = \max_{a'} Q_t(x', a')$

let's cont

- Writing in terms of Q-function:

$$Q_{t+1}(\mathbf{x},\mathbf{a}) = R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' | \mathbf{x},\mathbf{a}) \max_{\mathbf{a}'} Q_t(\mathbf{x}',\mathbf{a}')$$

Q-learning

$$Q_{t+1}(\mathbf{x},\mathbf{a}) = R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}'|\mathbf{x},\mathbf{a}) \max_{\mathbf{a}'} Q_t(\mathbf{x}',\mathbf{a}')$$

- Observe a transition: $\mathbf{x_t, a_t} \rightarrow \mathbf{x_{t+1}, r_{t+1}}$, approximate expectation by mixture of new sample with old estimate:
  - transition now from state-action pair to next state and reward

$$Q_{t+1}(x_t,a_t) = \underbrace{(1-\alpha)\,Q_t(x_t,a_t)}_{\text{a lot of the old}} + \underbrace{\alpha\left[r_{t+1} + \gamma \max_{a'} Q_t(x_{t+1},a')\right]}_{\text{a little of the new}}$$

  - α>0 is learning rate

Q-learning convergence

Q-function

- Under same conditions as TD, Q-learning converges to optimal value function Q^*

$\pi^*(x) = \arg\max_a Q^*(x,a)$

Off policy method:

- Can run any policy, as long as policy visits every state-action pair infinitely often
- Typical policies (non of these address Exploration-Exploitation tradeoff) directly
  - ε-greedy:
    - with prob. (1-ε) take greedy action: $\mathbf{a}_t = \arg\max_{\mathbf{a}} Q_t(\mathbf{x}, \mathbf{a})$
    - with prob. ε take an action at (uniformly) random
  - Boltzmann (softmax) policy: randomized max which ~~aren~~ actions with high value with high prob.
    - $P(\mathbf{a}_t \mid \mathbf{x}) \propto \exp\left\{\dfrac{Q_t(\mathbf{x},\mathbf{a})}{K}\right\}$
    - K – “temperature” parameter, K→0, as t→∞

The **curse of dimensionality**: A significant challenge in MDPs and RL

- MDPs and RL are polynomial in number of states and actions

- Consider a game with n units (e.g., peasants, footmen, etc.)
  - How many states? K^n
  - How many actions? m^n

K locations
m actions per player

- **Complexity is exponential in the number of variables used to define state!!!**

Addressing the curse!

- Some solutions for the curse of dimensionality:
  - **Learning the value function**: mapping from state-action pairs to values (real numbers) $Q: X \times A \to \mathbb{R}$
    - **A regression problem!**
  - **Learning a policy**: mapping from states to actions $\pi: X \to A \in \{1, \dots, k\}$
    - **A classification problem!**

- Use many of the ideas you learned this semester:
  - linear regression, SVMs, decision trees, neural networks, Bayes nets, etc.!!!

What you need to know about RL

- A model-based approach:
  - address exploration-exploitation tradeoff and credit assignment problem
  - the R-max algorithm
- A model-free approach:
  - never needs to learn transition model and reward function
  - TD-learning
  - Q-learning

Big Picture

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

May 3rd, 2006

What you have learned this semester

- Learning is function approximation
- Point estimation
- Regression
- Discriminative v. Generative learning
- Naïve Bayes
- Logistic regression
- Bias-Variance tradeoff
- Neural nets
- Decision trees
- Cross validation
- Boosting
- Instance-based learning
- SVMs
- Kernel trick
- PAC learning
- VC dimension
- Margin bounds
- Bayes nets
  - representation, inference, parameter and structure learning
- HMMs
  - representation, inference, learning
- K-means
- EM
- Semi-supervised learning
- Feature selection, dimensionality reduction, PCA
- MDPs
- Reinforcement learning

BIG PICTURE

- Improving the performance at some task though experience!!! ☺
  - before you start any learning task, remember the fundamental questions:

What is the learning problem?

From what experience?

What model?

What loss function are you optimizing?

With what optimization algorithm?

Which learning algorithm?

With what guarantees?

How will you evaluate it?

What next?

AI seminar ~aiseminar/

- Machine Learning Lunch talks: http://www.cs.cmu.edu/~learning/
- Journal:
  - JMLR – Journal of Machine Learning Research (free, on the web)
- Conferences:
  - ICML: International Conference on Machine Learning
  - NIPS: Neural Information Processing Systems
  - COLT: Computational Learning Theory
  - UAI: Uncertainty in AI
  - Also AAAI, IJCAI and others
- Some MLD courses:
  - 10-708 Probabilistic Graphical Models (Fall)
  - 10-705 Intermediate Statistics (Fall)
  - 10-702 Statistical Foundations of Machine Learning (Spring)