Markov Decision Processes (MDPs) (cont.)

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University
November 29th, 2007

Markov Decision Process (MDP) Representation

- State space:
  - Joint state \( x \) of entire system

- Action space:
  - Joint action \( a = \langle a_1, \ldots, a_n \rangle \) for all agents

- Reward function:
  - Total reward \( R(x,a) \)
    - sometimes reward can depend on action

- Transition model:
  - Dynamics of the entire system \( P(x'|x,a) \)
Computing the value of a policy

Discounted value of a state:

- value of starting from $x_0$ and continuing with policy $\pi$ from then on

$$V_\pi(x_0) = E_\pi[R(x_0) + \gamma R(x_1) + \gamma^2 R(x_2) + \gamma^3 R(x_3) + \ldots]$$

- A recursion!

$$V_\pi(x_0) = E_\pi[R(x_0) + \gamma E_\pi[R(x_1) + \gamma R(x_2) + \gamma^2 R(x_3) + \ldots)]$$

$$= R(x_0) + \gamma \sum_{x_1} P(x_1|x_0, \pi(x_0)) V_\pi(x_1)$$

Simple approach for computing the value of a policy: Iteratively

Can solve using a simple convergent iterative approach:

(a.k.a. dynamic programming)

- Start with some guess $V_0$

- Iteratively say:
  - $V_{t+1} = R + \gamma P_x V_t$

- Stop when $|V_{t+1} - V_t| \leq \epsilon$

  means that $|V_\pi - V_{t+1}| \leq \epsilon/(1-\gamma)$
But we want to learn a **Policy**

- So far, told you how good a policy is... $V_T(x)$
- 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!

\[
\pi^*(x) = \arg \max_a \mathbb{E}_{x'} R(x) + \gamma \sum_{x'} \mathbb{P}(x' | x, a) V(x')
\]

---

Unrolling the recursion

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

\[
V^*(x_0) = \max_{a_0} R(x_0, a_0) + \gamma E_{a_0} \left[ \max_{a_1} R(x_1) + \gamma^2 E_{a_1} \left[ \max_{a_2} R(x_2) + \ldots \right] \right]
\]
Bellman equation

- Evaluating policy $\pi$:
  \[ V_\pi(x) = R(x) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V_\pi(x') \]

- Computing the optimal value $V^*$ - Bellman equation
  \[ V^*(x) = \max_a R(x, a) + \gamma \sum_x P(x' \mid x, a) V^*(x') \]

Optimal Long-term Plan

**Optimal value function $V^*(x)$**  \[ \rightarrow \]
**Optimal Policy: $\pi^*(x)$**

**Optimal policy:**
\[ \pi^*(x) = \arg \max_a R(x, a) + \gamma \sum_{x'} P(x' \mid x, a) V^*(x') \]
Interesting fact – Unique value

\[ V^*(x) = \max_a R(x, a) + \gamma \sum_{x'} P(x' | x, a)V^*(x') \]

- 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 \( V^*(x) \)

Optimal policy \( \pi^*(x) \)

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

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

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

A simple example

You run a startup company.
In every state you must choose between Saving money or Advertising.

\( \gamma = 0.9 \)
Let’s compute $V_t(x)$ for our example

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

<table>
<thead>
<tr>
<th>t</th>
<th>$V_t(\text{PU})$</th>
<th>$V_t(\text{PF})$</th>
<th>$V_t(\text{RU})$</th>
<th>$V_t(\text{RF})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4.5</td>
<td>14.5</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>2.03</td>
<td>6.53</td>
<td>25.08</td>
<td>18.55</td>
</tr>
<tr>
<td>4</td>
<td>3.852</td>
<td>12.20</td>
<td>29.63</td>
<td>19.26</td>
</tr>
<tr>
<td>5</td>
<td>7.22</td>
<td>15.07</td>
<td>32.00</td>
<td>20.40</td>
</tr>
<tr>
<td>6</td>
<td>10.03</td>
<td>17.65</td>
<td>33.58</td>
<td>22.43</td>
</tr>
</tbody>
</table>
What you need to know

- What’s a Markov decision process
  - state, actions, transitions, rewards
  - a policy
  - value function for a policy
  - computing $V_{\pi}$

- Optimal value function and optimal policy
  - Bellman equation

- Solving Bellman equation
  - with value iteration, (other possibilities: 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](http://www.cs.cmu.edu/~awm/tutorials)
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 $x_t$ and reward $r_t$
  - you give next action $a_t$

- **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
Rmax – A model-based approach

Given a dataset – learn model

Given data, learn (MDP) Representation:

- Dataset:

- Learn reward function:
  - $R(x,a)$

- Learn transition model:
  - $P(x'|x,a)$
Some challenges in model-based RL 1: Planning with insufficient information

- Model-based approach:
  - estimate $R(x,a)$ & $P(x'|x,a)$
  - 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?
  - don’t reward at a particular state
    - plug in smallest reward ($R_{\min}$)?
    - plug in largest reward ($R_{\max}$)?
  - don’t know a particular transition probability?

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 $R_{\text{max}}$!!!

- If you don’t know the transition probabilities $P(x'|x,a)$ from some state action pair $x,a$ assume you go to a magic, fairytale new state $x_0$!!!
  - $R(x_0,a) = R_{\text{max}}$
  - $P(x_0|x_0,a) = 1$

Understanding $R_{\text{max}}$

- With $R_{\text{max}}$ 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 $x_0$ to MDP
- $R(x,a) = R_{max}, \forall x,a$
- $P(x_0|x,a) = 1, \forall x,a$
- all states (except for $x_0$) 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 $P(x'|x,a)$
  - update transition probs. $P(x'|x,a)$ 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:**
  - $X_1,...,X_n$ are i.i.d. Bernoulli trials with prob. $\theta$
  - $P(|1/n \sum X_i - \theta| > \varepsilon) \leq \exp\{-2n\varepsilon^2\}$

---

Putting it all together

- **Theorem:** With prob. at least $1-\delta$, $R_{\text{max}}$ will reach an $\varepsilon$-optimal policy in time polynomial in: num. states, num. actions, $T$, $1/\varepsilon$, $1/\delta$

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

- University Course Assessments
  - Please, please, please, please, please, please, please, please, please, please, please, please, please, please...

- Project:
  - Poster session: Tomorrow 2-4:45pm, NSH Atrium
    - Please arrive at least 15 minutes early to set up
  - Paper: Friday December 14th by 2pm
    - Electronic submission by email to instructors list
    - Maximum of 8 pages, NIPS format
    - No late days allowed

TD-Learning and Q-learning – Model-free approaches
Value of Policy

Value: $V_\pi(x)$
Expected long-term reward starting from $x$

Value: $V_\pi(x) = E_\pi[R(x_0) + \gamma R(x_1) + \gamma^2 R(x_2) + \gamma^3 R(x_3) + \gamma^4 R(x_4) + L]$

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

A simple monte-carlo policy evaluation

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

- **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, …)
- Idea 1: Observe a transition: \( x_t \mid x_{t+1}, r_{t+1} \). Approximate expec. with single sample:
  - unbiased!!
  - but a very bad estimate!!
Simple fix: Temporal Difference (TD) Learning [Sutton ’84]

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

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

  \[ \alpha > 0 \text{ is learning rate} \]

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

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

- **Theorem**: TD converges in the limit (with prob. 1), if:
  - every state is visited infinitely often
  - Learning rate decays just so:
    - \( \sum_{t=1}^{\infty} \alpha_t = 1 \)
    - \( \sum_{t=1}^{\infty} \alpha_t^2 < 1 \)
Another model-free RL approach: 
Q-learning [Watkins & Dayan '92]

- TD is just for one policy…
  - How do we find the optimal policy?

Q-learning:
- 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: 
  \[ V_{t+1}(x) = \max_a R(x, a) + \gamma \sum_{x'} P(x'| x, a) V_t(x') \]

- Or: 
  \[ Q_{t+1}(x, a) = R(x, a) + \gamma \sum_{x'} P(x'| x, a) V_t(x') \]
  \[ V_{t+1}(x) = \max_a Q_{t+1}(x, a) \]

- Writing in terms of Q-function:
  \[ Q_{t+1}(x, a) = R(x, a) + \gamma \sum_{x'} P(x'| x, a) \max_{a'} Q_t(x', a') \]
Q-learning

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

- Observe a transition: \( x_t, a_t \rightarrow x_{t+1}, r_{t+1} \). Approximate expectation by mixture of new sample with old estimate:
  - Transition now from state-action pair to next state and reward
  - \( \alpha > 0 \) is learning rate

Q-learning convergence

- Under same conditions as TD, Q-learning converges to optimal value function \( Q^* \)
- Can run any policy, as long as policy visits every state-action pair infinitely often
- Typical policies (non of these address Exploration-Exploitation tradeoff)
  - \( \varepsilon \)-greedy:
    - \( a_t = \text{arg max}_{a} Q_t(x, a) \)
    - With prob. \((1-\varepsilon)\) take greedy action:
    - With prob. \(\varepsilon\) take an action at (uniformly) random
  - Boltzmann (softmax) policy:
    - \( P(a_t | x) \propto \exp \left\{ \frac{Q_t(x, a)}{K} \right\} \)
    - \( K \) – “temperature” parameter, \( K \to 0 \), as \( t \to 1 \)
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?
  - How many actions?

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

Closing....
What you have learned this semester

- Learning is function approximation
- Point estimation
- Regression
- Discriminative v. Generative learning
- Naive Bayes
- Logistic regression
- Bias-Variance tradeoff
- Neural nets
- Decision trees
- Cross validation
- Boosting
- Instance-based learning
- SVMs
- Kernel trick
- PAC learning
- VC dimension
- Mistake bounds
- Bayes nets representation, inference, parameter and structure learning
- HMMs representation, inference, learning
- K-means
- EM
- 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:

<table>
<thead>
<tr>
<th>What is the learning problem?</th>
<th>From what experience?</th>
<th>What model?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>What loss function are you optimizing?</th>
<th>With what optimization algorithm?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Which learning algorithm?</th>
<th>With what guarantees?</th>
<th>How will you evaluate it?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What next?

- Intelligence Seminars: http://www.cs.cmu.edu/~iseminar/

- 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
  - AIStats: intersection of Statistics and AI
  - Also AAAI, IJCAI and others

- Some MLD courses:
  - 10-708 Probabilistic Graphical Models (Fall)
  - 10-705 Intermediate Statistics (Fall)
  - 11-762 Language and Statistics II (Fall)
  - 10-702 Statistical Foundations of Machine Learning (Spring)
  - 10-707 Optimization (Spring)
  - ...