

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

# Reinforcement Learning III: Policy Search and Policy Gradients

Many slides courtesy of  
Dan Klein, Stuart Russell,  
Andrew Moore or  
Alan Fern

CS 5300 / CS 6300  
Artificial Intelligence  
Spring 2009

Hal Daumé III  
hal@cs.utah.edu

www.cs.utah.edu/~hal/courses/2009S\_AI

Slide 1

CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## Announcements

- Project 2 grades out now/soon
- More homework grades out soon
- HW4 due Thursday
- Project 3 is/will be up today (RL)
- HW5 (RL) up today/tomorrow
- Midterm in two weeks

Slide 2

CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## What's Coming Up...

- Today: Policy methods
- 19 Feb: Final day of RL (inverse RL)
- 24 Feb: Robot motion
- 26 Feb: Probability (preparing for 2<sup>nd</sup> half of the course)
- 03 Mar: Midterm (in class, open book, like HW)
- 05 Mar...: Reasoning under uncertainty

Slide 3


CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## Example: Pacman

- Let's say we discover through experience that this state is bad:
- In naïve q learning, we know nothing about this state or its q states:
- Or even this one!



Slide 4

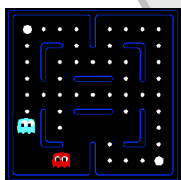
CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## Feature-Based Representations

- Solution: describe a state using a vector of features
- Features are functions from states to real numbers (often 0/1) that capture important properties of the state
- Example features:
  - Distance to closest ghost
  - Distance to closest dot
  - Number of ghosts
  - 1 / (dist to dot)<sup>2</sup>
  - Is Pacman in a tunnel? (0/1)
  - ..... etc.
- Is it the exact state on this slide?
- Can also describe a q-state (s, a) with features (e.g. action moves closer to food)



Slide 5

CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## Linear Feature Functions

- Using a feature representation, we can write a q function (or value function) for any state using a few weights:

- Advantage: our experience is summed up in a few powerful numbers
- Disadvantage: states may share features but be very different in value!

Slide 6

CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## RL via Policy Gradient Search

- So far all of our RL techniques have tried to learn an exact or approximate utility function or Q-function
  - I.e. learn the optimal "value" of being in a state, or taking an action from a state.
- Value functions can often be much more complex to represent than the corresponding policy
  - Do we really care about knowing  $Q(s, \text{left}) = 0.3554$ ,  $Q(s, \text{right}) = 0.533$
  - Or just that "right is better than left in state s"
- Motivates searching directly in a parameterized policy space

Slide 7 CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## Policy Search

- Simplest policy search:
  - Start with an initial linear value function or q-function
  - Nudge each feature weight up and down and see if your policy is better than before
- Problems:
  - How do we tell the policy got better?
  - Need to run many sample episodes!
  - If there are a lot of features, this can be impractical

Slide 8 CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## Aside: Gradient Ascent

- Given a function  $f(\theta_1, \dots, \theta_n)$  of  $n$  real values  $\theta = (\theta_1, \dots, \theta_n)$  suppose we want to maximize  $f$  with respect to  $\theta$
- A common approach to doing this is gradient ascent
- The gradient of  $f$  at point  $\theta$ , denoted by  $\nabla_{\theta} f(\theta)$ , is an  $n$ -dimensional vector that points in the direction where  $f$  increases most steeply at point  $\theta$
- Vector calculus tells us that  $\nabla_{\theta} f(\theta)$  is just a vector of partial derivatives

$$\nabla_{\theta} f(\theta) = \left[ \frac{\partial f(\theta)}{\partial \theta_1}, \dots, \frac{\partial f(\theta)}{\partial \theta_n} \right]$$

where  $\frac{\partial f(\theta)}{\partial \theta_i} = \lim_{\epsilon \rightarrow 0} \frac{f(\theta_1, \dots, \theta_{i-1}, \theta_i + \epsilon, \theta_{i+1}, \dots, \theta_n) - f(\theta)}{\epsilon}$

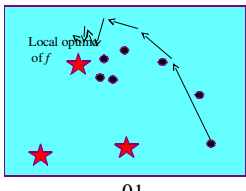
Slide 9 CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## Aside: Gradient Ascent

- Gradient ascent iteratively follows the gradient direction starting at some initial point
  - ▢ Initialize  $\theta$  to a random value
  - ▢ Repeat until stopping condition

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} f(\theta)$$


Slide 10 CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## RL via Policy Gradient Ascent

- This general approach has the following components
  1. Select a space of parameterized policies:
  2. Compute the gradient of the value function of the policy wrt parameters
  3. Move parameters in the direction of the gradient
  4. Repeat these steps until we reach a local maxima
- So we must answer the following questions:
  - ▢ How should we represent parameterized policies?
  - ▢ How can we compute the gradient?

Slide 11 CS 5300: RL III

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

## Parameterized Policies

- One example of a space of parametric policies is:
 
$$\pi_{\theta}(s) = \arg \max_a \hat{Q}_{\theta}(s, a)$$
- where  $\hat{Q}_{\theta}(s, a)$  may be a linear function, e.g.
 
$$\hat{Q}_{\theta}(s, a) = \theta_0 + \theta_1 f_1(s, a) + \theta_2 f_2(s, a) + \dots + \theta_n f_n(s, a)$$
- The goal is to learn parameters  $\theta$  that give a good policy
- Note that it is not important that  $\hat{Q}_{\theta}(s, a)$  be close to the actual Q-function
  - Rather we only require  $\hat{Q}_{\theta}(s, a)$  is good at ranking actions in order of goodness

Slide 12 CS 5300: RL III

UNIVERSITY OF UTAH

## Policy Gradient Ascent

Hal Daumé III (hal@cs.utah.edu)

- Let  $\rho(\theta)$  be the expected value of policy  $\pi_\theta$ .
  - $\rho(\theta)$  is just the expected discounted total reward for a trajectory of  $\pi_\theta$ .
  - For simplicity assume each trajectory starts at a single initial state.
- Our objective is to find a  $\theta$  that maximizes  $\rho(\theta)$
- Policy gradient ascent tells us to iteratively update parameters via:  $\theta \leftarrow \theta + \alpha \nabla_\theta \rho(\theta)$
- Problem:**  $\rho(\theta)$  is generally very complex and it is rare that we can compute a closed form for the gradient of  $\rho(\theta)$ .
- We will instead estimate the gradient based on experience

Slide 13 CS 5300: RL III

UNIVERSITY OF UTAH

## Gradient Estimation

Hal Daumé III (hal@cs.utah.edu)

- Concern:** Computing or estimating the gradient of discontinuous functions can be problematic.
- For our example parametric policy
 
$$\pi_\theta(s) = \arg \max_a \hat{Q}_\theta(s, a)$$
 is  $\rho(\theta)$  continuous?
  - No.
    - There are values of  $\theta$  where arbitrarily small changes, cause the policy to change.
    - Since different policies can have different values this means that changing  $\theta$  can cause discontinuous jump of  $\rho(\theta)$ .

Slide 14 CS 5300: RL III

UNIVERSITY OF UTAH

## Example: Discontinuous $\rho(\theta)$

Hal Daumé III (hal@cs.utah.edu)

$$\pi_\theta(s) = \arg \max_a \hat{Q}_\theta(s, a) = \theta_1 f_1(s, a)$$

- Consider a problem with initial state  $s$  and two actions  $a1$  and  $a2$ 
  - $a1$  leads to a very large terminal reward  $R1$
  - $a2$  leads to a very small terminal reward  $R2$
- Fixing  $\theta_2$  to a constant we can plot the ranking assigned to each action by  $Q$  and the corresponding value  $\rho(\theta)$

Slide 15 CS 5300: RL III

UNIVERSITY OF UTAH

## Probabilistic Policies

Hal Daumé III (hal@cs.utah.edu)

- We would like to avoid policies that drastically change with small parameter changes, leading to discontinuities
- A **probabilistic policy**  $\pi_\theta$  takes a state as input and returns a distribution over actions
  - Given a state  $s$   $\pi_\theta(s, a)$  returns the probability that  $\pi_\theta$  selects action  $a$  in  $s$
- Note that  $\rho(\theta)$  is still well defined for probabilistic policies
  - Now uncertainty of trajectories comes from environment and policy
  - Importantly if  $\pi_\theta(s, a)$  is continuous relative to changing  $\theta$  then  $\rho(\theta)$  is also continuous relative to changing  $\theta$
- A common form for probabilistic policies is the **softmax function** or **Boltzmann exploration function**

$$\pi_\theta(s, a) = \Pr(a | s) = \frac{\exp(\hat{Q}_\theta(s, a))}{\sum_{a' \in \mathcal{A}} \exp(\hat{Q}_\theta(s, a'))}$$

Slide 16 CS 5300: RL III

UNIVERSITY OF UTAH

## Empirical Gradient Estimation

Hal Daumé III (hal@cs.utah.edu)

- Our first approach to estimating  $\nabla_\theta \rho(\theta)$  is to simply compute empirical gradient estimates
- Recall that  $\theta = (\theta_1, \dots, \theta_n)$  and  $\nabla_\theta \rho(\theta) = \left[ \frac{\partial \rho(\theta)}{\partial \theta_1}, \dots, \frac{\partial \rho(\theta)}{\partial \theta_n} \right]$
- so we can compute the gradient by empirically estimating each partial derivative
 
$$\frac{\partial \rho(\theta)}{\partial \theta_i} = \lim_{\epsilon \rightarrow 0} \frac{\rho(\theta_1, \dots, \theta_{i-1}, \theta_i + \epsilon, \theta_{i+1}, \dots, \theta_n) - \rho(\theta)}{\epsilon}$$
- So for small  $\epsilon$  we can estimate the partial derivatives by
 
$$\frac{\rho(\theta_1, \dots, \theta_{i-1}, \theta_i + \epsilon, \theta_{i+1}, \dots, \theta_n) - \rho(\theta)}{\epsilon}$$
- This requires estimating  $n+1$  values:
 
$$\rho(\theta), \{ \rho(\theta_1, \dots, \theta_{i-1}, \theta_i + \epsilon, \theta_{i+1}, \dots, \theta_n) \mid i = 1, \dots, n \}$$

Slide 17 CS 5300: RL III

UNIVERSITY OF UTAH

## Empirical Gradient Estimation

Hal Daumé III (hal@cs.utah.edu)

- How do we estimate the quantities
 
$$\rho(\theta), \{ \rho(\theta_1, \dots, \theta_{i-1}, \theta_i + \epsilon, \theta_{i+1}, \dots, \theta_n) \mid i = 1, \dots, n \}$$
- For each set of parameters, simply execute the policy for  $N$  trials/episodes and average the values achieved across the trials
- This requires a total of  $N(n+1)$  episodes to get gradient estimate
  - For stochastic environments and policies the value of  $N$  must be relatively large to get good estimates of the true value
  - Often we want to use a relatively large number of parameters
  - Often it is expensive to run episodes of the policy
- So while this can work well in many situations, it is often not a practical approach computationally

Slide 18 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Gradient Estimation: Single Step Problems

- For stochastic policies it is possible to estimate  $\nabla_{\theta} \rho(\theta)$  directly from trajectories of just the current policy  $\pi_{\theta}$ 
  - Idea: take advantage of the fact that we know the functional form of the policy
- First consider the simplified case where all trials have length 1
  - For simplicity assume each trajectory starts at a single initial state and reward only depends on action choice
  - $\rho(\theta)$  is just the expected reward of action selected by  $\pi_{\theta}$ .

$$\rho(\theta) = \sum_a \pi_{\theta}(s_0, a) R(a)$$

where  $s_0$  is the initial state and  $R(a)$  is reward of action  $a$

- The gradient of this becomes

$$\nabla_{\theta} \rho(\theta) = \nabla_{\theta} \sum_a \pi_{\theta}(s_0, a) R(a) = \sum_a (\nabla_{\theta} \pi_{\theta}(s_0, a)) R(a)$$

- How can we estimate this by just observing the execution of  $\pi_{\theta}$ ?

Slide 19 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Gradient Estimation: Single Step Problems

- Rewriting  $\nabla_{\theta} \rho(\theta) = \sum_a (\nabla_{\theta} \pi_{\theta}(s_0, a)) R(a)$ 

$$= \sum_a \pi_{\theta}(s_0, a) \frac{(\nabla_{\theta} \pi_{\theta}(s_0, a))}{\pi_{\theta}(s_0, a)} R(a)$$

$$= \sum_a \pi_{\theta}(s_0, a) \underbrace{\nabla_{\theta} \log(\pi_{\theta}(s_0, a))}_{\text{can get closed form } g(s_0, a)} R(a)$$
- The gradient is just the expected value of  $g(s_0, a)R(a)$  over execution trials of  $\pi_{\theta}$ 
  - Can estimate by executing  $\pi_{\theta}$  for  $N$  trials and

$$\nabla_{\theta} \rho(\theta) \approx \frac{1}{N} \sum_{j=1}^N g(s_0, a_j) R(a_j)$$

$a_j$  is action selected by policy on  $j$ 'th episode

- Only requires executing  $\pi_{\theta}$  for a number of trials that need not depend on the number of parameters

Slide 20 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Gradient Estimation: General Case

- So for the case of a length 1 trajectories we got:
$$\nabla_{\theta} \rho(\theta) \approx \frac{1}{N} \sum_{j=1}^N g(s_0, a_j) R(a)$$
- For the general case where trajectories have length greater than one and reward depends on state we can do some work and get:
$$\nabla_{\theta} \rho(\theta) \approx \frac{1}{N} \sum_{j=1}^N \sum_{t=1}^{T_j} g(s_{j,t}, a_{j,t}) R_j(s_{j,t})$$

length of trial  $j$

Observed total reward in trial  $j$  from step  $t$  to end
- $s_{j,t}$  is  $t$ 'th state of  $j$ 'th episode,  $a_{j,t}$  is  $t$ 'th action of episode  $j$
- The derivation of this is straightforward but messy.

Slide 21 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Gradient Estimation: General Case

- How can we interpret this gradient expression:
$$g(s, a) = \nabla_{\theta} \log(\pi_{\theta}(s, a))$$

Direction to move parameters in order to increase the probability that policy selects  $a_j$  in state  $s_j$
- $$\nabla_{\theta} \rho(\theta) \approx \frac{1}{N} \sum_{j=1}^N \sum_{t=1}^{T_j} g(s_{j,t}, a_{j,t}) R_j(s_{j,t})$$

Total reward observed after taking  $a_{j,t}$  in state  $s_{j,t}$
- So the overall gradient is a reward weighted combination of individual gradient directions
- Intuitively this increases probability of actions in states that were followed by large rewards and decreases the probability of actions in states that were followed by negative rewards

Slide 22 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Basic Policy Gradient Algorithm

- Repeat until stopping condition
  - Execute  $\pi_{\theta}$  for  $N$  trajectories while storing the state, action, reward sequences
  - $\nabla_{\theta} \leftarrow \frac{1}{N} \sum_{j=1}^N \sum_{t=1}^{T_j} g(s_{j,t}, a_{j,t}) R_j(s_{j,t})$
  - $\theta \leftarrow \theta + \alpha \nabla_{\theta}$
- One disadvantage of this approach is the small number of updates per amount of experience
  - Also requires a notion of trajectory rather than an infinite sequence of experience
- Online policy gradient algorithms perform updates after each step in environment (often learn faster)

Slide 23 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Online Policy Gradient

Repeat forever

- Observe state  $s$
- Draw action  $a$  according to distribution  $\pi_{\theta}(s)$
- Execute  $a$  and observe reward  $r$
- $e \leftarrow \beta e + \nabla_{\theta} \log \Pr(a | s, \theta)$  ;; discounted sum of  
;; gradient directions
- $\theta \leftarrow \theta + \alpha \cdot r \cdot e$

- Performs policy update at each time step and executes indefinitely
  - This is the OLPOMDP algorithm [Baxter & Bartlett, 2000]

Slide 24 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Gradient Estimation: General Case

- Both algorithms require computation of  $g(s, a) = \nabla_{\theta} \log(\pi_{\theta}(s, a))$
- For the Boltzmann distribution with linear approximation we have:
 
$$\pi_{\theta}(s, a) = \frac{\exp(\hat{Q}_{\theta}(s, a))}{\sum_{a' \in A} \exp(\hat{Q}_{\theta}(s, a'))}$$
 where
 
$$\hat{Q}_{\theta}(s, a) = \theta_0 + \theta_1 f_1(s, a) + \theta_2 f_2(s, a) + \dots + \theta_n f_n(s, a)$$
- Here the partial derivatives needed for  $g(s, a)$  are:
 
$$\frac{\partial \log(\pi_{\theta}(s, a))}{\partial \theta_i} = f_i(s, a) - \sum_{a'} \pi_{\theta}(s, a') f_i(s, a')$$

Slide 25 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Controlling Helicopters

- Policy gradient techniques have been used to create controllers for difficult helicopter maneuvers

Slide 26 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Quadruped Locomotion

- Optimize gait of 4-legged robots over rough terrain

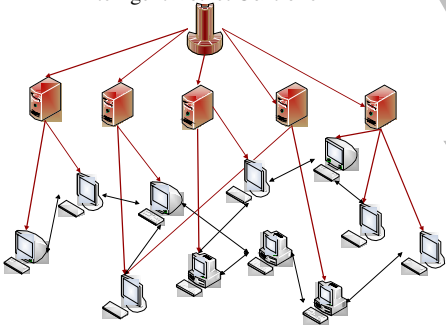


Slide 27 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Proactive Security

Intelligent Botnet Controller



- Used OLPOMDP to proactively discover maximally damaging botnet attacks in peer-to-peer networks [Dejmal & Fern, 2008]

Slide 28 CS 5300: RL III

UNIVERSITY OF UTAH  
Hal Daumé III (hal@cs.utah.edu)

## Policy Gradient Recap

- When policies have much simpler representations than the corresponding value functions, direct search in policy space can be a good idea
  - Allows us to design complex parametric controllers and optimize details of parameter settings
- For baseline algorithm the gradient estimates are unbiased (i.e. they will converge to the right value) but have high variance
  - Can require a large N to get reliable estimates
  - OLPOMDP offers can trade-off bias and variance via the discount parameter [Baxter & Bartlett, 2000]
- Can be prone to finding local maxima
  - Many ways of dealing with this, e.g. random restarts.

Slide 29 CS 5300: RL III