Reduction in Reinforcement Learning

Policy Search by Dynamic Programming

Drew Bagnell
dbagnell@ri.cmu.edu

http://www.cs.cmu.edu/~dbagnell

Carnegie Mellon Robotics Institute

Joint work with Sham Kakade, Andrew Ng, and Jeff Schneider

Outline

- Introduction to Stochastic Control Problem
- POMDPs and Memoryless Policies
- Policy Search by Dynamic Programming
- Exact (Discrete) PSDP
- PSDP using Classification
- PSDP using Regression
- Refinements and Conclusions

Stochastic Control Problem

- Elements of the control problem
  - Space of paths Ξ (system trajectories)
  - A sequence of controls $< a_t >_{t \in \{0,..,T\}}$
  - A sequence of observations $< o_t >_{t \in \{0,..,T\}}$
  - A *controller*, π that maps $< o_t >$ to a distribution over a_t
  - A probability distribution over paths $P_\pi(\xi)$
  - A reinforcement function $R(\xi)$

Stochastic Control Problem

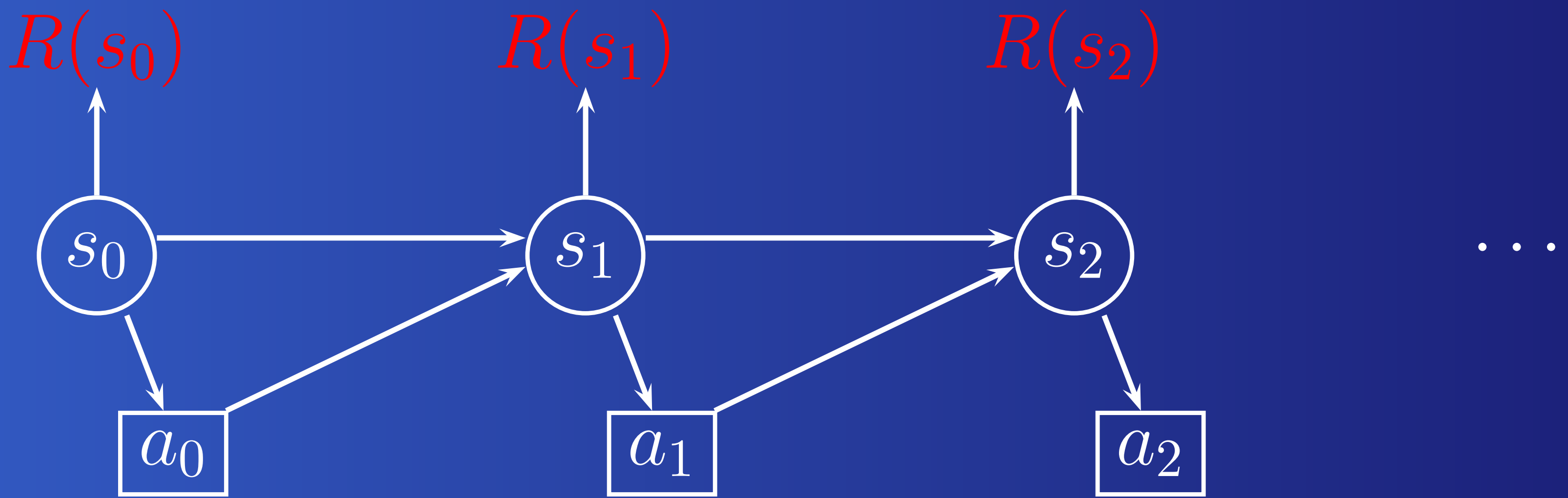
- Goal: Optimize the expectation

$$J_\pi = \sum_{\Xi} P_\pi(\xi) R(\xi)$$

- Special cases: MDPs, POMDPs, state-space systems
- *Reinforcement Learning: Stochastic control with unknown model*

Markov Decision Process

- Formal states that render past and future independent
- $R_{path}(\xi) = \frac{1}{T}\sum_t R(s_t)$



MDP Solution Techniques

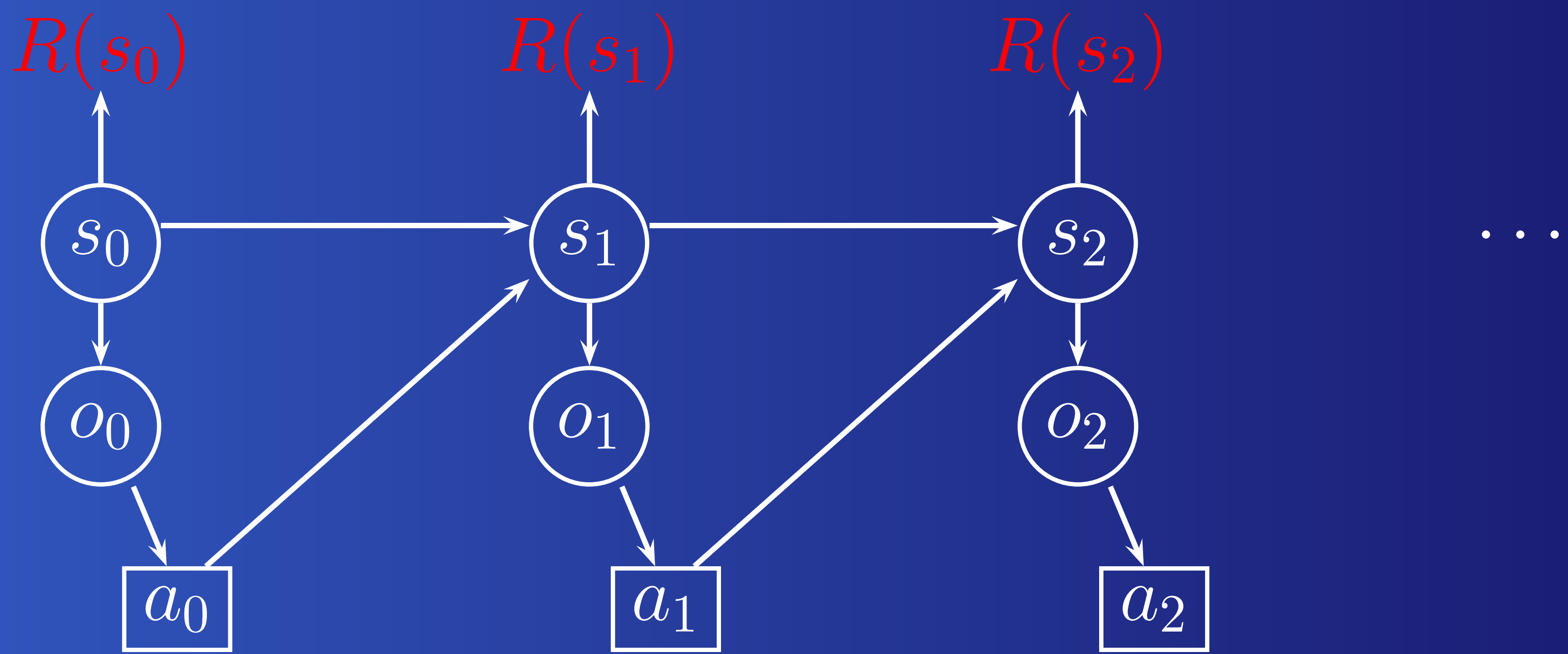
- Standard solution techniques leverage dynamic programming and the *Bellman equations*
- Bellman equations relate the values starting from one state with that from starting from other states
- Optimal solutions are memoryless mappings from states to controls
- Algorithms scale polynomially in the number of states, exponentially in state variables
- Approximate value function techniques can give very impressive results [Tesauro95]

Outline

- Introduction to Stochastic Control Problem
- POMDPs and Memoryless Policies
- Policy Search by Dynamic Programming
- Exact (Discrete) PSDP
- PSDP using Regression
- PSDP using Classification
- Refinements and Conclusions

Partial Observability

- Natural extension is to make o_t a random variable

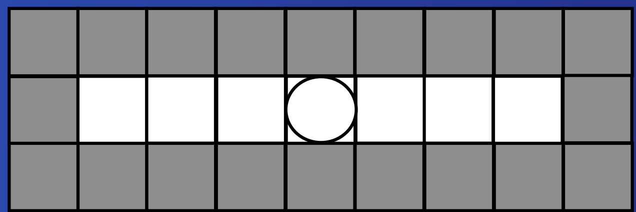


POMDPs

- Elegant structural properties to optimal controllers
  - POMDPs under total reward become belief-state MDPs [Littman96]
- Belief state scales exponentially in number of states (and it's continuous!)
- With rare exceptions (Linear-Quadratic Gaussian) POMDPs are overwhelmingly intractable [Stengel86]
- Policy-search has become a preferred approach to solving large MDPs and POMDPs

Finding memoryless policies

- Consider the problem of finding a policy that maps observations to controls
- In contrast to MDPs, it is NP-hard to find the best such policy in a POMDP
- There may exist no satisficing deterministic stationary policy



Typical approaches

- Ignore it– run an MDP learning algorithm
  - May perform arbitrarily badly
- Learn a stochastic policy that maps observations to distributions
  - Use gradient method to optimize
  - Sample complexity may be exponential
  - Local minima abound

Outline

- Introduction to Stochastic Control Problem
- POMDPs and Memoryless Policies
- Policy Search by Dynamic Programming
- Exact (Discrete) PSDP
- PSDP using Classification
- PSDP using Regression
- Refinements and Conclusions

Rethinking Dynamic Programming

- Bellman's original *Principle of Optimality*

"An optimal policy has the property that whatever the initial state and optimal first decision may be, the remaining decisions constitute an optimal policy with regard to the state resulting from the first decision"

- is a statement about *policies*
- Recourse to functional (Bellman) equations is neither necessary, nor always helpful

Rethinking Dynamic Programming

- Instead, perhaps we can have out cake and eat it too
- We can conceive of a *generalized* principle of optimality:

An optimal $\pi \in \Pi$ has the property that whatever the initial state and optimal first decision may be, the remaining decisions should be optimal with respect to the observations and the optimal distribution over states.

- Suppose instead of backing up value-functions, we backed up *policies*

Policy Search by DP

Algorithm 1 (PSDP) *Given* T, μ_t, *and* Π:

for $t = T-1, T-2, \ldots, 0$

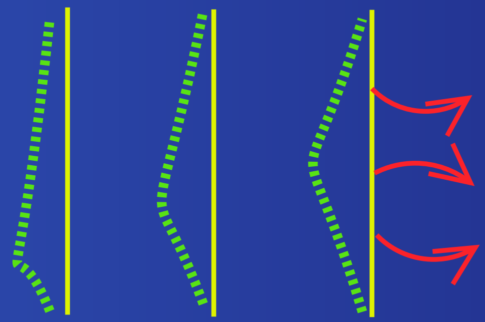
Set $\pi_t = \arg\max_{\pi' \in \Pi} \mathrm{E}_{s \sim \mu_t}[V_{\pi', \pi_{t+1} \ldots, \pi_{T-1}}(s)]$

Policy Search by DP

Algorithm 1 (PSDP) *Given* T, μ_t, *and* Π*:*

for $t = T-1, T-2, \ldots, 0$

Set $\pi_t = \arg\max_{\pi' \in \Pi} \mathrm{E}_{s \sim \mu_t}[V_{\pi', \pi_{t+1} \ldots, \pi_{T-1}}(s)]$

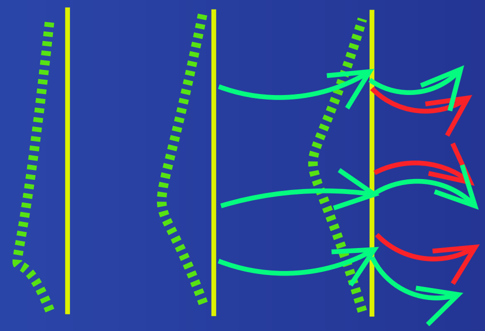


Policy Search by DP

Algorithm 1 (PSDP) *Given T, μ_t, and Π:*

for $t = T-1, T-2, \ldots, 0$

Set $\pi_t = \arg\max_{\pi' \in \Pi} \mathrm{E}_{s \sim \mu_t} [V_{\pi', \pi_{t+1} \ldots, \pi_{T-1}}(s)]$

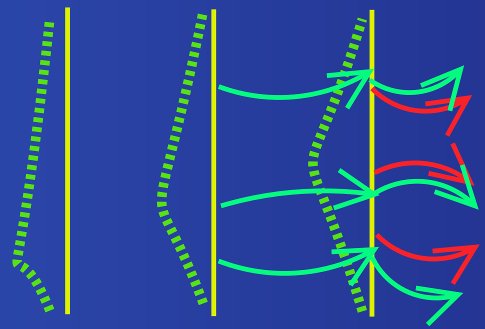


Policy Search by DP

Algorithm 1 (PSDP) *Given* T, μ_t, *and* Π:

for $t = T-1, T-2, \ldots, 0$

Set $\pi_t = \arg\max_{\pi' \in \Pi} \mathrm{E}_{s \sim \mu_t} [V_{\pi', \pi_{t+1} \ldots, \pi_{T-1}}(s)]$



- At each step, we only have to think about the best mapping from O to A
- The distribution μ makes previous decisions irrelevant
- And the *future* decisions are already made (optimally)

How well can we do?

Intuitively, we'd like that PSDP returns a policy that competes favorably with all policies whose future state distributions are close to μ Define variation distance as

$$d_{\mathrm{var}}(\mu, \mu') \equiv \frac{1}{T} \sum_{t=0}^{T-1} \sum_{s \in S} |\mu_t(s) - \mu'_t(s)|$$

Theorem 0 (Performance Guarantee) *Let $\pi = (\pi_0, \ldots, \pi_{T-1})$ be a non-stationary policy returned by an ε-approximate version of PSDP in which, on each step, the policy π_t found comes within ε of maximizing the value. I.e.,*

$$\mathrm{E}_{s \sim \mu_t}[V_{\pi_t, \pi_{t+1} \ldots, \pi_{T-1}}(s)] \geq \max_{\pi' \in \Pi} \mathrm{E}_{s \sim \mu_t}[V_{\pi', \pi_{t+1} \ldots, \pi_{T-1}}(s)] - \varepsilon \,. \qquad (0)$$

Then for all $\pi_{\mathrm{ref}} \in \Pi^T$ we have that

$$V_\pi(s_0) \geq V_{\pi_{\mathrm{ref}}}(s_0) - T\varepsilon - T d_{\mathrm{var}}(\mu, \mu_{\pi_{\mathrm{ref}}}) \,.$$

Outline

- Introduction to Stochastic Control Problem
- POMDPs and Memoryless Policies
- Policy Search by Dynamic Programming
- Exact (Discrete) PSDP
- PSDP using Classification
- PSDP using Regression
- Refinements and Conclusions

Exact PSDP

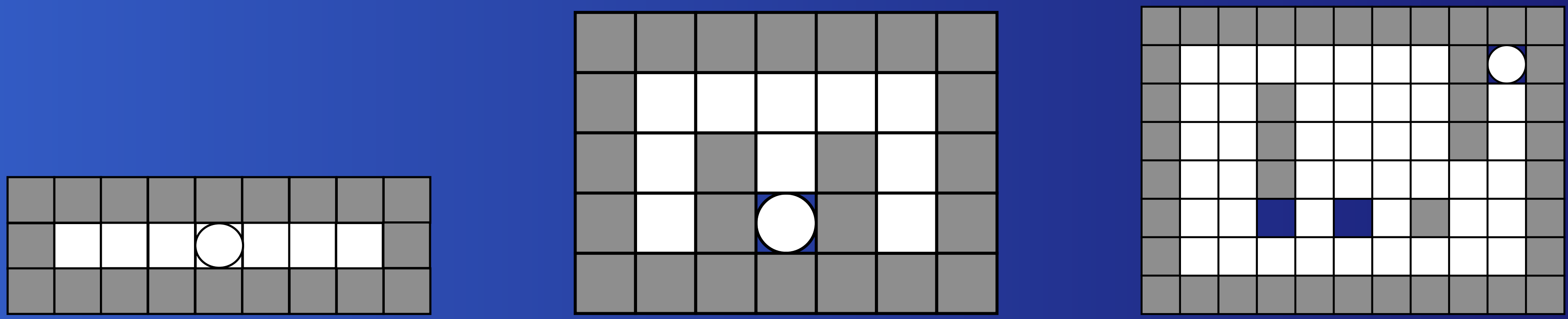
Under exact PSDP (discrete observations), the policy update is as follows:

$$\pi_t(o) = \arg\max_a \mathrm{E}_{s\sim\mu_t}[p(o|s)V_{a,\pi_{t+1}\dots,\pi_{T-1}}(s)] \tag{0}$$

Proposition 0 (PSDP complexity) *For any POMDP, exact PSDP ($\varepsilon = 0$) runs in time polynomial in the size of the state and observation spaces and in the horizon time T.*

Intuitively, distribution u specifies how to trade-off different state-action pairs that share an observation.

Mazes



(a) Hallway (b) McCallum's Maze (c) Sutton

	μ uniform	μ iterated	Optimal SD	Optimal
Hallway	21	21	∞	18
McCallum	55	48	∞	39
Sutton	412	412	416	≥ 408

Memoryless policy classes

- Four natural classes:stationary deterministic (*SD*), stationary stochastic (*SS*), non-stationary deterministic (*ND*) and non-stationary stochastic (*NS*)

Memoryless policy classes

- Four natural classes:stationary deterministic (*SD*), stationary stochastic (*SS*), non-stationary deterministic (*ND*) and non-stationary stochastic (*NS*)

Proposition 1 (Policy ordering) *For any POMDP,*

$$opt(SD) \leq opt(SS) \leq opt(ND) = opt(NS)$$

Memoryless policy classes

- Four natural classes:stationary deterministic (*SD*), stationary stochastic (*SS*), non-stationary deterministic (*ND*) and non-stationary stochastic (*NS*)

Proposition 1 (Policy ordering) *For any POMDP,*

$$\mathit{opt}(SD) \leq \mathit{opt}(SS) \leq \mathit{opt}(ND) = \mathit{opt}(NS)$$

- Unfortunately NP-hard to find optimal policies
- PSDP offers well-founded, tractable alternative to search heuristics

Outline

- Introduction to Stochastic Control Problem
- POMDPs and Memoryless Policies
- Policy Search by Dynamic Programming
- Exact (Discrete) PSDP
- PSDP using Classification
- PSDP using Regression
- Refinements and Conclusions

Weighted classification

- Consider discrete (two!) action POMDPs
- Suppose the maximization $\arg\max_{\pi' \in \Pi} \mathrm{E}_{s \sim \mu_t}[V_{\pi', \pi_{t+1} \ldots, \pi_{T-1}}(s)]$ can be closely approximated by a linear policy
- This algorithm turns the maximization into a classification problem:

Algorithm 1 (Linear maximization) *Given $m1$ and $m2$:*

for $i = 1$ to m_1

Sample $s^{(i)} \sim \mu_t$.

Use m_2 Monte Carlo samples to estimate $V_{a_1, \pi_{t+1}, \ldots, \pi_{T-1}}(s^{(i)})$ and $V_{a_2, \pi_{t+1}, \ldots, \pi_{T-1}}(s^{(i)})$. Call the resulting estimates q_1 and q_2.

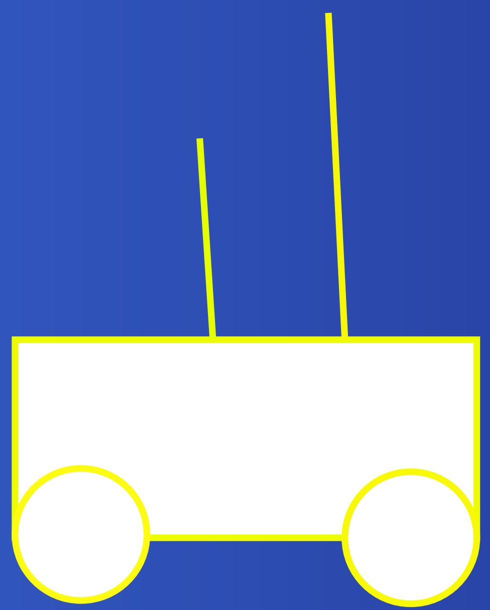
Let $y^{(i)} = 1\{q_1 > q_2\}$, and $w^{(i)} = |q_1 - q_2|$.

Find $\theta = \arg\min_\theta \sum_{i=1}^{m_1} w^{(i)} 1\{1\{\theta^T \phi(s^{(i)}) \geq 0\} \neq y^{(i)}\}$.

Output π_θ.

Weighted classification

- The weighted 0-1 loss problem is NP-hard, but approximable [Amaldi98]
- Can take variational approach and pick convex bound on loss
- Logistic regression $-\ell(\theta) = -\sum_i w^{(i)} \log p(y^{(i)}|s^{(i)}, \theta)$ where $p(y=1|s,\theta) = 1/(1+\exp(-\theta^T s))$



Applied to double cart-pole

Outline

- Introduction to Stochastic Control Problem
- POMDPs and Memoryless Policies
- Policy Search by Dynamic Programming
- Exact (Discrete) PSDP
- PSDP using Classification
- PSDP using Regression
- Refinements and Conclusions

l_1 Regression

- PSDP can also be efficiently implemented if we can efficiently find action-value function $\tilde{V}_{a,\pi_{t+1}\ldots,\pi_{T-1}}(s)$, i.e., if at each timestep

$$\epsilon \geq \mathrm{E}_{s\sim\mu_t}\left[\max_{a\in A}|\tilde{V}_{a,\pi_{t+1}\ldots,\pi_{T-1}}(s) - V_{a,\pi_{t+1}\ldots,\pi_{T-1}}(s)|\right].$$

- Policy acts greedily with respect to V
- Easy to check that we differ from optimal PSDP solution by $2T\epsilon$

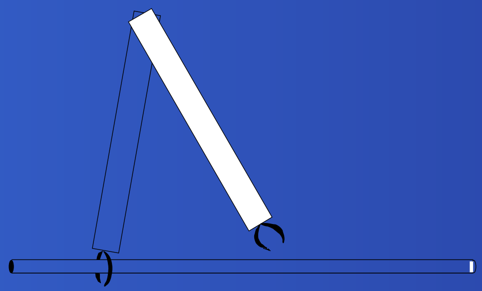
l_1 Regression

- PSDP can also be efficiently implemented if we can efficiently find action-value function $\tilde{V}_{a,\pi_{t+1}\dots,\pi_{T-1}}(s)$, i.e., if at each timestep

$$\epsilon \geq \mathrm{E}_{s\sim\mu_t}\left[\max_{a\in A}|\tilde{V}_{a,\pi_{t+1}\dots,\pi_{T-1}}(s) - V_{a,\pi_{t+1}\dots,\pi_{T-1}}(s)|\right].$$

- Policy acts greedily with respect to V
- Easy to check that we differ from optimal PSDP solution by $2T\epsilon$
- Important: Error here is in terms of *average* over state-space *not* worst-case
- Value-iteration algorithms amplify errors by pushing more probability through where errors are
- PSDP doesn't; rollouts keep it honest

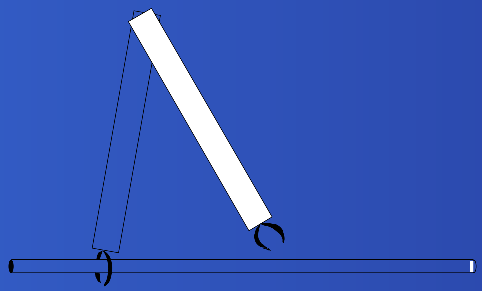
Brachiating robot



PSDP is in spirit related to DDP [Atkeson02]

- Trajectories in DDP serve as the analog of μ
- Central difference is value function backups instead of policy backups
- Use their planar biped walking robot simulator
- Robot has a 5-d (essentially) state-space, control is hip-torque

Brachiating robot



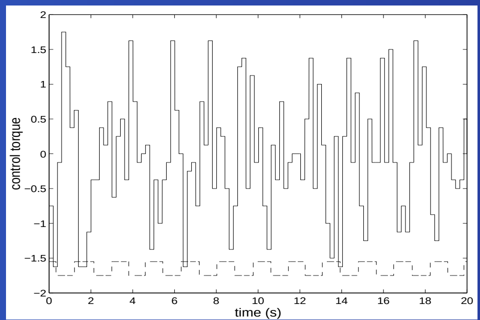
PSDP is in spirit related to DDP [Atkeson02]

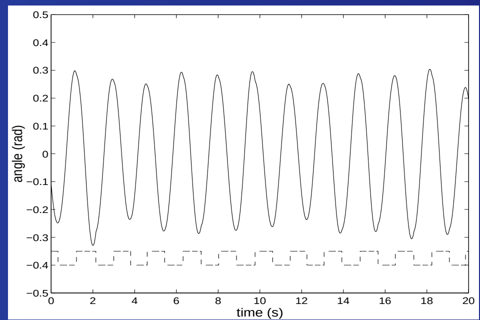
Advantages of PSDP

- Able to handle partial observability, discontinuous cost functions
- Removed state-variables as input to regression one-by-one
- Succeeds with just one-bit: which foot is down (nearly open-loop)

Advantages of PSDP

- Able to handle partial observability, discontinuous cost functions
- Removed state-variables as input to regression one-by-one
- Succeeds with just one-bit: which foot is down (nearly open-loop)





(Left) Control signal from open-loop learned controller.
(Right) Resulting angle of one leg.

Outline

- Introduction to Stochastic Control Problem
- POMDPs and Memoryless Policies
- Policy Search by Dynamic Programming
- Exact (Discrete) PSDP
- PSDP using Classification
- PSDP using Regression
- Refinements and Conclusions

Iterative μ Refinement

- Natural outer-loop to algorithm
- After backwards sweep computing policy
- apply forward sweep computing resulting $\mu(x, t)$
- Improves result in a number of cases– sometimes dramatically
- For $\epsilon = 0$ it follows that performance never decreases
- Seeking a local maximum in the policy space
- Completes analogy with DDP technique

Future and Related Work

- Clearly closely related to MDP policy search technique of [Fern03] [Lagoudakis03] [Langford03] [Kakade03]
- Next steps include more serious problems
- As well as problems with approximate filters/belief states