Direct Gradient-Based Reinforcement Learning

Jonathan Baxter
Research School of Information Sciences and Engineering
Australian National University
http://csl.anu.edu.au/∼jon

Joint work with Peter Bartlett and Lex Weaver

December 5, 1999

Reinforcement Learning

Models **agent** interacting with its **environment**.

1. Agent receives information about its **state**.
2. Agent chooses **action** or **control** based on state-information.
3. Agent receives a **reward**.
4. State is updated.
5. Goto **??**.

Reinforcement Learning

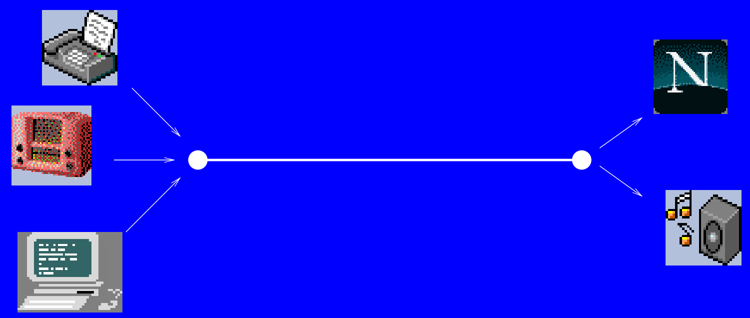
- **Goal:** Adjust agent's behaviour to maximize long-term average reward.
- **Key Assumption:** state transitions are **Markov**.

Chess

- **State:** Board position.
- **Control:** Move pieces.
- **State Transitions:** My move, followed by opponent's move.
- **Reward:** Win, draw, or lose.

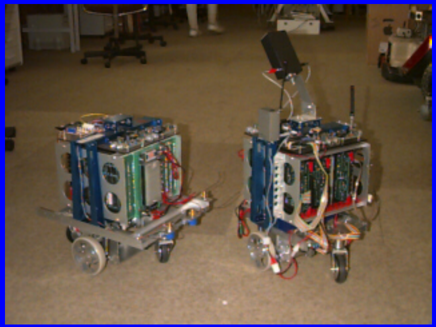
Call Admission Control

Telecomms carrier selling bandwidth: queueing problem.



- **State:** Mix of call types on channel.
- **Control:** Accept calls of certain type.
- **State Transitions:** Calls finish. New calls arrive.
- **Reward:** Revenue from calls accepted.

Cleaning Robot



- **State:** Robot and environment (position, velocity, dust levels, . . .).
- **Control:** Actions available to robot.
- **State Transitions:** depend on dynamics of robot and statistics of environment.
- **Reward:** Pick up rubbish, don't damage the furniture.

Summary

Previous approaches:

- **Dynamic Programming** can find optimal policies in **small** state spaces.
- **Approximate Value-Function** based approaches currently the method of choice in **large** state spaces.
- Numerous practical successes, **BUT**
- Policy performance can **degrade** at each step.

Summary

Alternative Approach:

- Policy parameters $\boldsymbol{\theta} \in \mathbb{R}^{K}$, **Performance:** $\boldsymbol{\eta}(\boldsymbol{\theta})$.
- Compute $\boldsymbol{\nabla\eta}(\boldsymbol{\theta})$ and step uphill (gradient ascent).
- Previous algorithms relied on accurate **reward baseline** or **recurrent states**.

Summary

Our Contribution:

- Approximation $\nabla_\beta \eta(\theta)$ to $\nabla \eta(\theta)$.
- Parameter $\beta \in [0, 1)$ related to **Mixing Time** of problem.
- Algorithm to approximate $\nabla_\beta \eta(\theta)$ via simulation (**POMDPG**)
- Line search in the presence of noise.

Partially Observable Markov Decision Processes (POMDPs)

States: $\mathcal{S} = \{1, 2, \ldots, n\}$ $\quad X_t \in \mathcal{S}$

Observations: $\mathcal{Y} = \{1, 2, \ldots, M\}$ $\quad Y_t \in \mathcal{Y}$

Actions or Controls: $\mathcal{U} = \{1, 2, \ldots, N\}$ $\quad U_t \in \mathcal{U}$

Observation Process ν: $\Pr(Y_t = y | X_t = i) = \nu_y(i)$

Stochastic Policy μ: $\Pr(U_t = u | Y_t = y) = \mu_u(\theta, y)$

Rewards: $r : \mathcal{S} \longrightarrow \mathbb{R}$

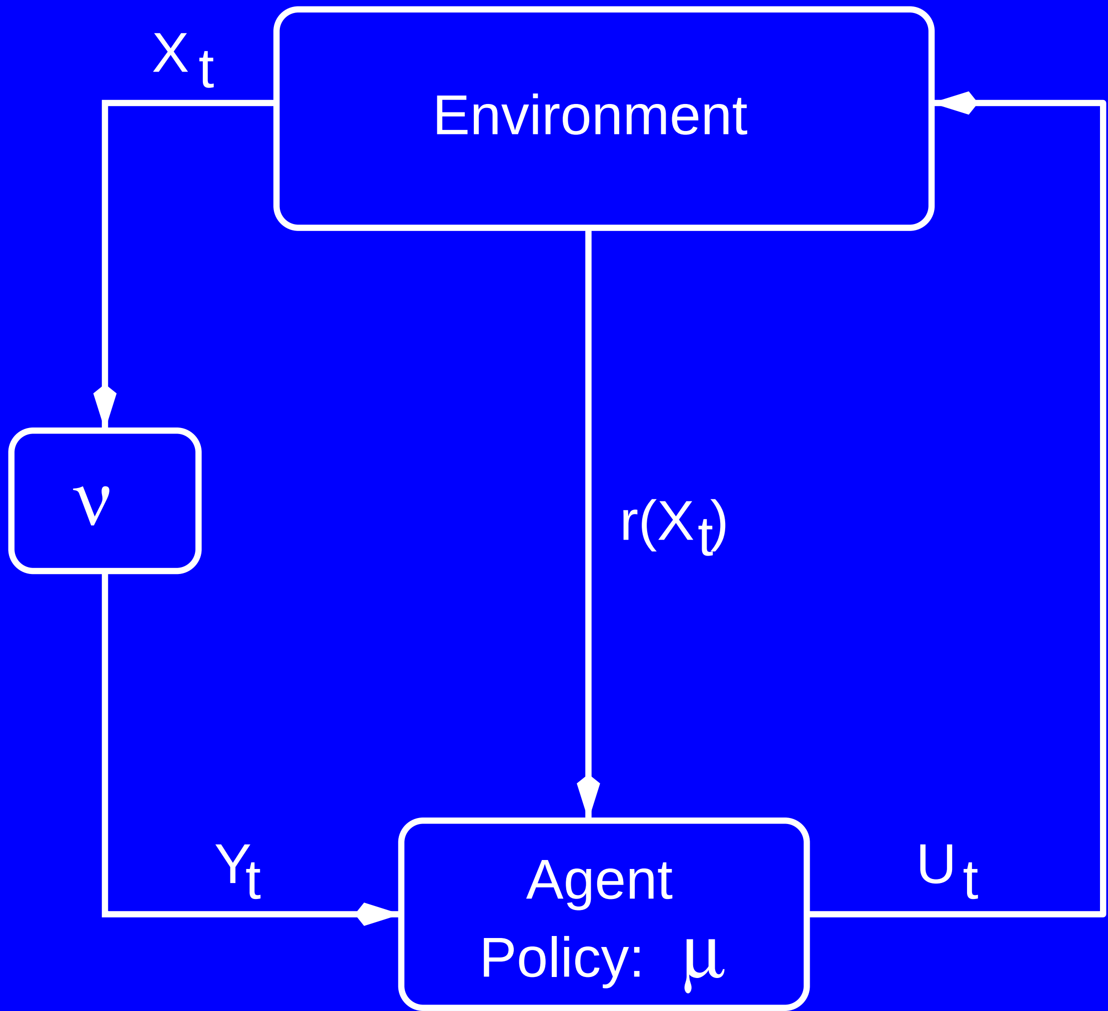
Adjustable parameters: $\theta \in \mathbb{R}^K$

POMDP

Transition Probabilities:

$$\Pr(X_{t+1} = j | X_t = i, U_t = u) = p_{ij}(u)$$

POMDP



The Induced Markov Chain

- **Transition Probabilities:**

$$\begin{aligned} p_{ij}(\theta) &= \Pr\left(X_{t+1} = j \,|\, X_t = i\right) \\ &= \mathbf{E}_{y \sim \nu(X_t)}\, \mathbf{E}_{u \sim \mu(\theta, y)}\, p_{ij}(u) \end{aligned}$$

- **Transition Matrix:**

$$P(\theta) = [p_{ij}(\theta)]$$

Stationary Distributions

$\boldsymbol{q} = [\boldsymbol{q}_1 \cdots \boldsymbol{q}_n]' \in \mathbb{R}^n$ is a distribution over states.

$$\begin{aligned} & \boldsymbol{X}_t \sim \boldsymbol{q} \\ \Rightarrow \quad & \boldsymbol{X}_{t+1} \sim \boldsymbol{q}' \boldsymbol{P}(\boldsymbol{\theta}) \end{aligned}$$

Definition: A probability distribution $\boldsymbol{\pi} \in \mathbb{R}^n$ is a **stationary distribution** of the Markov chain if

$$\boldsymbol{\pi}' \boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{\pi}'.$$

Stationary Distributions

Convenient Assumption: For all values of the parameters θ, there is a **unique** stationary distribution $\pi(\theta)$.

Implies the Markov chain **mixes**:

For all X_0, the distribution of X_t approaches $\pi(\theta)$.

Inconvenient Assumption: Number of states n "essentially infinite".

Meaning: forget about storing a number for each state, or inverting $n \times n$ matrices.

Measuring Performance

- **Average Reward:**

$$\eta(\theta) = \sum_{i=1}^{n} \pi_i(\theta) r(i)$$

- **Goal:** Find θ maximizing $\eta(\theta)$.

Summary

- Partially Observable Markov Decision Processes.
- **Previous approaches: value function methods.**
- Direct gradient ascent
- Approximating the gradient of the average reward.
- Estimating the approximate gradient: POMDPG.
- Line search in the presence of noise.
- Experimental results.

Approximate Value Functions

- **Discount Factor** $\beta \in [0,1)$, **Discounted value** of state i under policy μ:

$$J_\beta^\mu(i) = \mathbf{E}_\mu \left[r(X_0) + \beta r(X_1) + \beta^2 r(X_2) + \cdots \,\middle|\, X_0 = i \right]$$

- **Idea:** Choose restricted class of **value functions** $\tilde{J}(\theta, i)$, $\theta \in \mathbb{R}^K, i \in \mathcal{S}$ (e.g neural network with parameters θ).

Policy Iteration

Iterate:

- Given policy μ, find approximation $\tilde{J}(\theta,\cdot)$ to J_β^μ.
- Many algorithms for finding θ: TD(λ), Q-learning, Bellman residuals, ...
- Simulation and non-simulation based.
- Generate new policy μ' using $\tilde{J}(\theta,\cdot)$:

$$\mu'_{u^*}(\theta,i) = 1 \Leftrightarrow u^* = \operatorname{argmax}_{u\in\mathcal{U}} \sum_{j\in\mathcal{S}} p_{ij}(u)\tilde{J}(\theta,j)$$

Approximate Value Functions

- **The Good:**
  - ⋆ Backgammon (world-champion), chess (International Master), job-shop scheduling, elevator control, . . .
  - ⋆ Notion of “backing-up” state values can be efficient.
- **The Bad:**
  - ⋆ Unless $\left|\tilde{J}(\theta, i) - J_\beta^\mu(i)\right| = 0$ for **all** states i, the new policy μ' can be a lot worse than the old one.
  - ⋆ “Essentially Infinite” state spaces means we are likely to have very bad approximation error for some states.

Summary

- Partially Observable Markov Decision Processes.
- Previous approaches: value function methods.
- **Direct gradient ascent.**
- Approximating the gradient of the average reward.
- Estimating the approximate gradient: POMDPG.
- Line search in the presence of noise.
- Experimental results.

Direct Gradient Ascent

- **Desideratum:** Adjusting the agent's parameters θ should improve its performance.

- Implies. . .

- **Adjust the parameters in the direction of the gradient of the average reward:**

$$\theta := \theta + \gamma \nabla \eta(\theta)$$

Direct Gradient Ascent: Main Results

1. Algorithm to estimate approximate gradient($\nabla_\beta \eta$) from a sample path.

2. Accuracy of approximation depends on parameter of the algorithm (β); bias/variance trade-off.

3. Line search algorithm using only gradient estimates.

Related Work

Machine Learning: Williams' REINFORCE algorithm (1992).

- Gradient ascent algorithm for restricted class of MDPs.
- Requires accurate *reward baseline*, i.i.d. transitions.

Kimura *et. al.* , 1998: extension to infinite horizon.

Discrete Event Systems: Algorithms that rely on recurrent states. MDPs: (Cao and Chen, 1997), POMDPs: (Marbach and Tsitsiklis, 1998).

Control Theory: Direct adaptive control using derivatives (Hjalmarsson, Gunnarsson, Gevers, 1994), (Kammer, Bitmead, Bartlett, 1997), (DeBruyne, Anderson, Gevers, Linard, 1997).

Summary

- Partially Observable Markov Decision Processes.
- Previous approaches: value function methods.
- Direct gradient ascent.
- **Approximating the gradient of the average reward.**
- Estimating the approximate gradient: POMDPG.
- Line search in the presence of noise.
- Experimental results.

Approximating the gradient

Recall: For $\beta \in [0,1)$, **Discounted value** of state i is

$$J_\beta(i) = \mathbf{E}\left[r(X_0) + \beta r(X_1) + \beta^2 r(X_2) + \cdots \,\middle|\, X_0 = i \right].$$

Vector notation: $J_\beta = (J_\beta(1), \ldots, J_\beta(n))$.

Theorem: For all $\beta \in [0,1)$,

$$\nabla\eta(\theta) = \beta\pi'(\theta)\nabla P(\theta)J_\beta + (1-\beta)\nabla\pi'(\theta)J_\beta.$$

$$= \beta\underbrace{\nabla_\beta\eta(\theta)}_{\text{estimate}} + \underbrace{(1-\beta)\nabla\pi'(\theta)J_\beta}_{\to 0 \text{ as } \beta\to 1}.$$

Mixing Times of Markov Chains

- **ℓ_1-distance:** If p, q are distributions on the states,

$$\|p - q\|_1 := \sum_{i=1}^{n} |p(i) - q(i)|$$

- **$d(t)$-distance:** Let $p^t(i)$ be the distribution over states at time t, starting from state i.

$$d(t) := \max_{ij} \|p^t(i) - p^t(j)\|_1$$

- **Unique stationary distribution** $\Rightarrow d(t) \rightarrow 0$.

Approximating the gradient

Mixing time: $\tau^* := \min\{t : d(t) \leq e^{-1}\}$

Theorem: For all $\beta \in [0,1)$, $\theta \in \mathbb{R}^k$,

$$\|\nabla\eta(\theta) - \nabla_\beta\eta(\theta)\| \leq \text{constant} \times \tau^*(\theta)(1-\beta).$$

That is, if $1/(1-\beta)$ is large compared with the mixing time $\tau^*(\theta)$, $\nabla_\beta\eta(\theta)$ accurately approximates the gradient direction $\nabla\eta(\theta)$.

Summary

- Partially Observable Markov Decision Processes.
- Previous approaches: value function methods.
- Direct gradient ascent.
- Approximating the gradient of the average reward.
- **Estimating the approximate gradient: POMDPG.**
- Line search in the presence of noise.
- Experimental results.

Estimating $\nabla_\beta \eta(\theta)$: POMDPG

Given: parameterized policies, $\mu_u(\theta, y), \beta \in [0, 1)$:

1. Set $z_0 = \Delta_0 = 0 \in \mathbb{R}^K$.
2. **for** each observation y_t, control u_t, reward $r(i_{t+1})$ **do**
3. Set $z_{t+1} = \beta z_t + \dfrac{\nabla \mu_{u_t}(\theta, y_t)}{\mu_{u_t}(\theta, y_t)}$ (eligibility trace)
4. Set $\Delta_{t+1} = \Delta_t + \frac{1}{t+1}\left[r(i_{t+1}) z_{t+1} - \Delta_t\right]$
5. **end for**

Convergence of POMDPG

Theorem: For all $\beta \in [0,1)$, $\boldsymbol{\theta} \in \mathbb{R}^{\boldsymbol{K}}$,

$$\boldsymbol{\Delta}_t \longrightarrow \boldsymbol{\nabla}_\beta \boldsymbol{\eta}(\boldsymbol{\theta}).$$

Explanation of POMDPG

Algorithm computes:

$$\Delta_T = \frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla \mu_{u_t}}{\mu_{u_t}} \underbrace{\left(r(i_{t+1}) + \beta r(i_{t+2}) + \cdots + \beta^{T-t-1} r(i_T)\right)}_{\text{Estimate of \textit{discounted value} `due to' action } u_t}$$

- $\nabla \mu_{u_t}(\theta, y_t)$ is the direction to increase the probability of the action u_t.
- It is weighted by something involving subsequent rewards, and
- divided by μ_{u_t}: ensures "popular" actions don't dominate

POMDPG: Bias/Variance trade-off

$$\boxed{\mathbf{\Delta}_t \xrightarrow{t\to\infty} \nabla_\beta \boldsymbol{\eta}(\boldsymbol{\theta}) \xrightarrow{\beta\to 1} \nabla \boldsymbol{\eta}(\boldsymbol{\theta})}.$$

- **Bias/Variance Tradeoff:** $\beta \approx 1$ gives:
  - ⋆ Accurate gradient approximation ($\nabla_\beta \boldsymbol{\eta}$ close to $\nabla \boldsymbol{\eta}$), but
  - ⋆ Large variance in estimates $\mathbf{\Delta}_t$ of $\nabla_\beta \boldsymbol{\eta}$ for small t.

POMDPG: Bias/Variance trade-off

$$\boxed{\Delta_t \xrightarrow{t\to\infty} \nabla_\beta \eta(\theta) \xrightarrow{\beta\to 1} \nabla \eta(\theta)}.$$

- **Recall:** $1/(1-\beta) \approx \tau^*(\theta)$ (mixing time).
  - $\star$ **Small mixing time** $\Rightarrow$ **small** β $\Rightarrow$ accurate gradient estimate from **short POMDPG simulation**.
  - $\star$ **Large mixing time** $\Rightarrow$ **large** β $\Rightarrow$ accurate gradient estimate only from **long POMDPG simulation**.
- **Conjecture:** Mixing time is an **intrinsic** constraint on any simulation-based algorithm.

Example: 3-state Markov Chain

Transition Probabilities:

$$P(u_1) = \begin{bmatrix} 0 & 4/5 & 1/5 \\ 4/5 & 0 & 1/5 \\ 0 & 4/5 & 1/5 \end{bmatrix} \qquad P(u_2) = \begin{bmatrix} 0 & 1/5 & 4/5 \\ 1/5 & 0 & 4/5 \\ 0 & 1/5 & 4/5 \end{bmatrix}$$

Observations: $(\phi_1(i), \phi_2(i))$:

State 1: $(2/3, 1/3)$ State 2: $(1/3, 2/3)$ State 3: $(5/18, 5/18)$

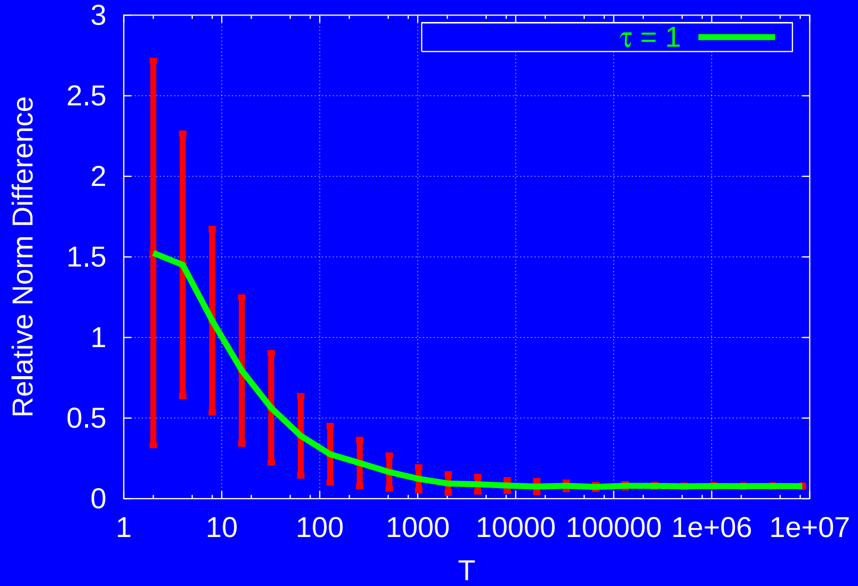
Parameterized Policy: $\theta \in \mathbb{R}^2$

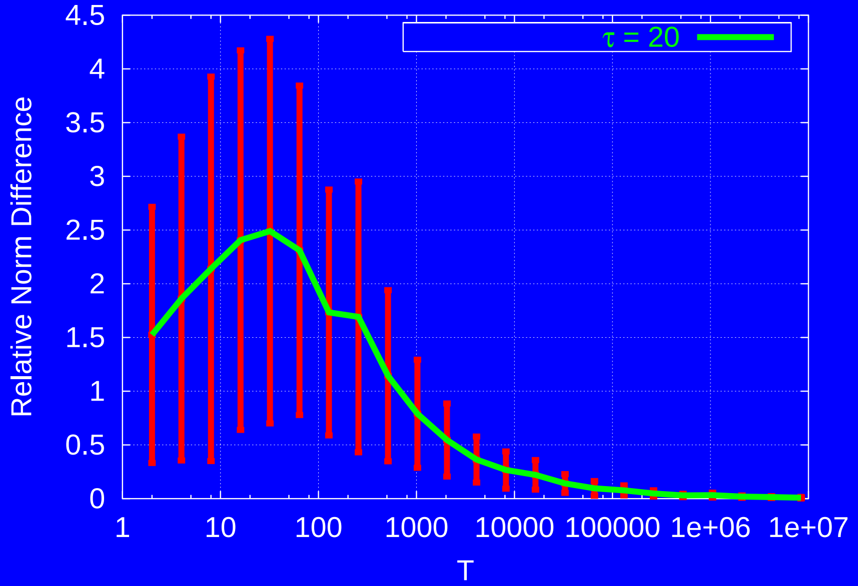
$$\mu_{u_1}(\theta, i) = \frac{e^{(\theta_1\phi_1(i)+\theta_2\phi_2(i))}}{1 + e^{(\theta_1\phi_1(i)+\theta_2\phi_2(i))}} \qquad \mu_{u_2}(\theta, i) = 1 - \mu_{u_1}(\theta, i)$$

Rewards: $(r(1), r(2), r(3)) = (0, 0, 1)$

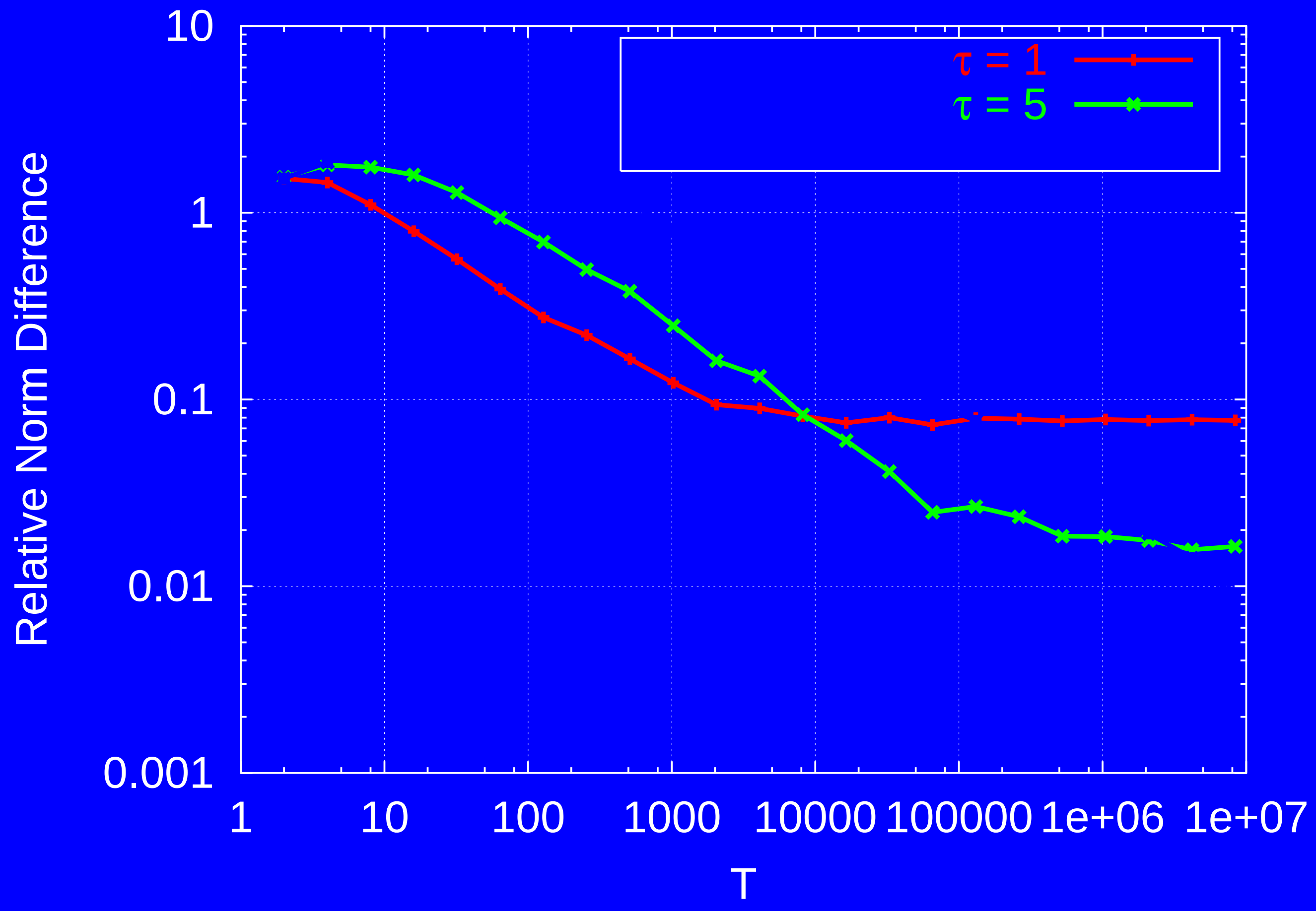
Bias/Variance Trade-off

$$\text{Relative norm difference} = \frac{\|\nabla\eta - \Delta_T\|}{\|\nabla\eta\|}.$$





Bias/Variance Trade-off



Summary

- Partially Observable Markov Decision Processes.
- Previous approaches: value function methods.
- Direct gradient ascent.
- Approximating the gradient of the average reward.
- Estimating the approximate gradient: POMDPG.
- **Line search in the presence of noise.**
- Experimental results.

Line-search in the presence of noise

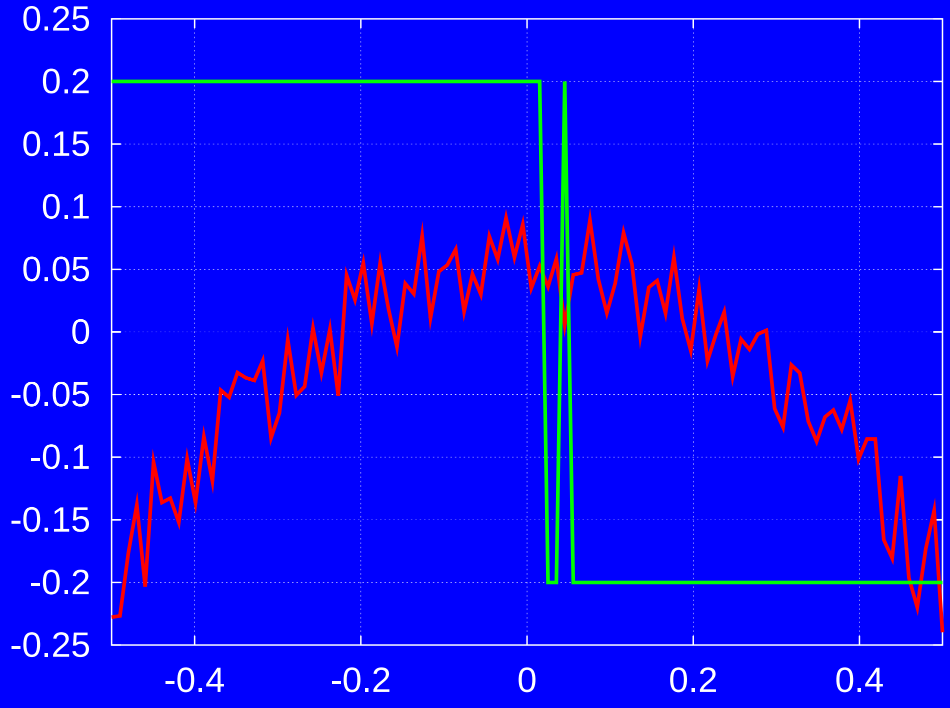
- Want to find maximum of $\eta(\theta)$ in direction $\nabla_\beta \eta(\theta)$.
- Usual method: find 3 points
$\theta_i = \theta + \gamma_i \nabla_\beta \eta(\theta), \quad i = 1, 2, 3,$
with $\gamma_1 < \gamma_2 < \gamma_3$ such that:
$\eta(\theta_2) > \eta(\theta_1), \quad \eta(\theta_2) > \eta(\theta_3)$ and interpolate.
- **Problem:** $\eta(\theta)$ only available by simulation (*e.g.* $\eta_T(\theta)$), so noisy:

$$\lim_{\theta_1 \to \theta_2} \mathrm{var}\left[\mathrm{sign}\left(\eta_T(\theta_2) - \eta_T(\theta_1)\right)\right] = 1$$

Line-search in the presence of noise

- **Solution:** Use gradients to bracket (POMDPG).
$$\nabla_\beta \eta(\theta_1) \cdot \nabla_\beta \eta(\theta) > 0, \quad \nabla_\beta \eta(\theta_2) \cdot \nabla_\beta \eta(\theta) < 0$$

- Variance independent of $||\theta_2 - \theta_1||$.



Example: Call Admission Control

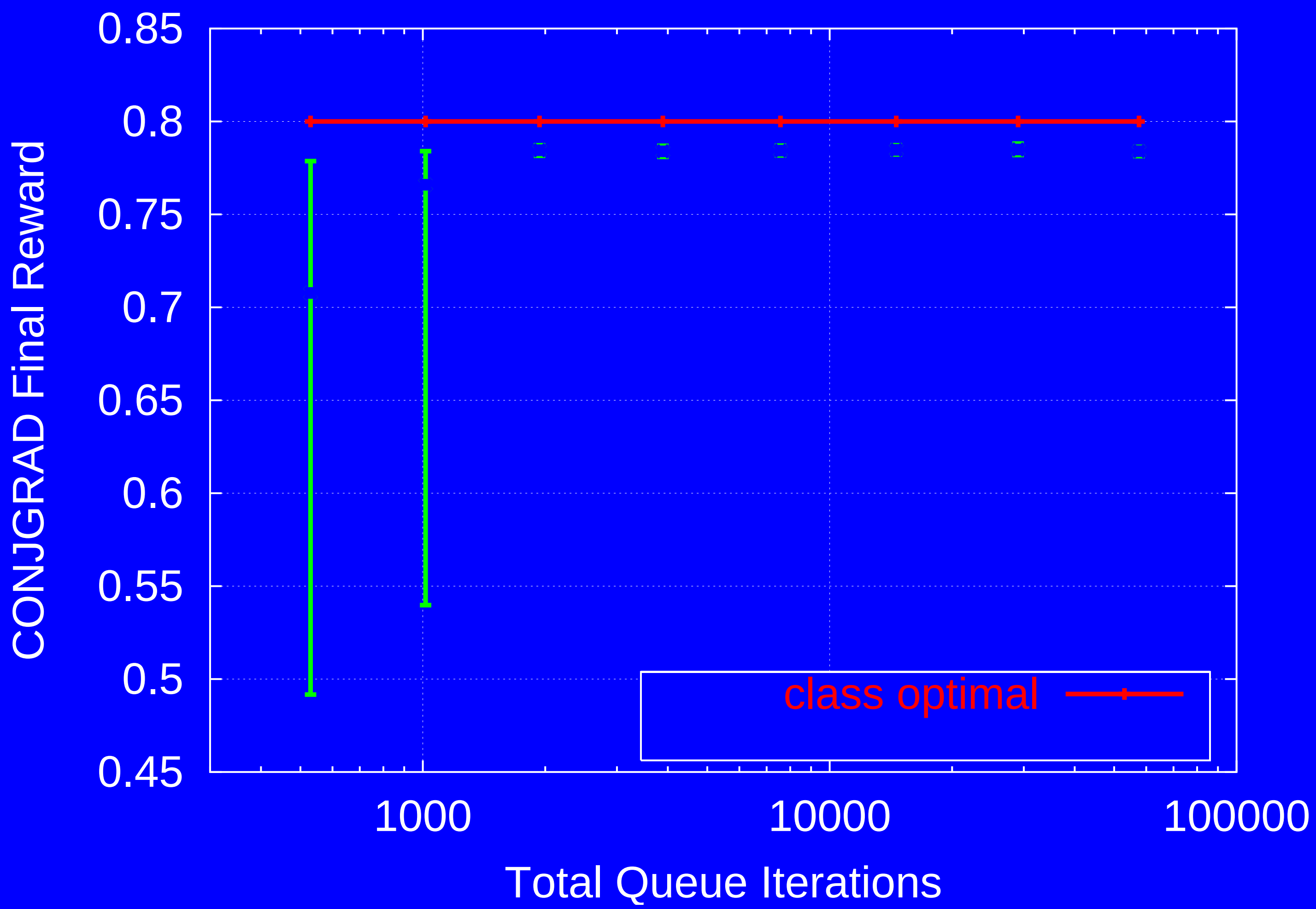
Telecommunications carrier selling bandwidth: queueing problem.
From (Marbach and Tsitsiklis, 1998).

- Three call types, with differing arrival rates (Poisson), bandwidth requirements, rewards, holding times (exponentia

- State $=$ observation $=$ mix of calls.

- Policy $=$ (squashed) linear controller.

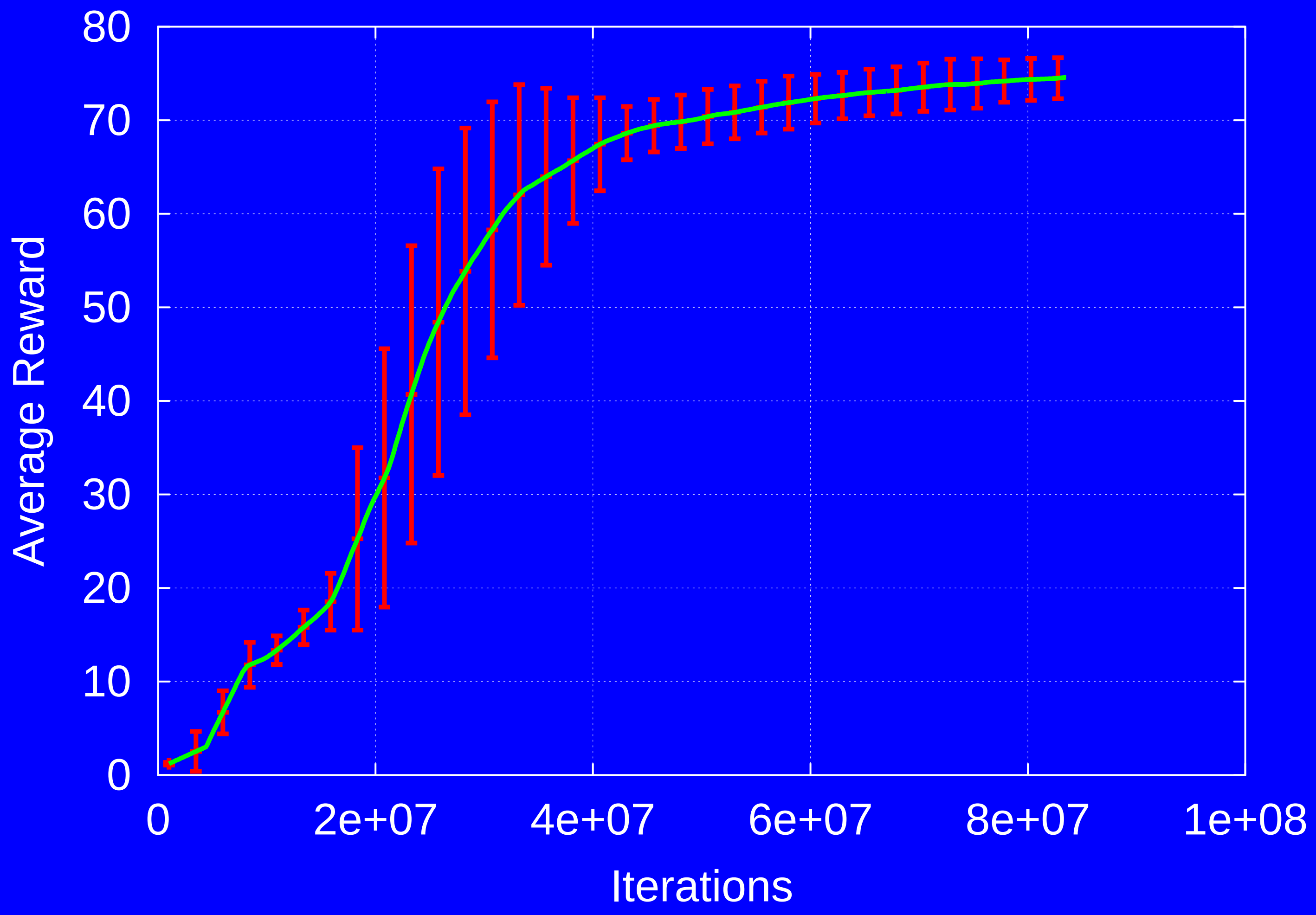
Direct Reinforcement Learning: Call Admission Control



Direct Reinforcement Learning: Puck World

- Puck moving around mountainous terrain.
- Aim is to get out of a valley and on to a plateau
- reward = 0 everywhere except plateau (=100)
- Observation $=$ relative location, absolute location, velocity.
- Neural-Network Controller
- Insufficient thrust to climb directly out of valley, must learn to “oscillate”.

Direct Reinforcement Learning: Puck World



Direct Reinforcement Learning

- **Philosophy:**
  - ⋆ Adjusting policy should improve performance.
  - ⋆ View average reward as function of policy parameters: $\eta(\theta)$.
  - ⋆ For suitably smooth policies: $\nabla\eta(\theta)$ exists.
  - ⋆ Compute $\nabla\eta(\theta)$ and step uphill.

Direct Reinforcement Learning

- **Main results:**
  - ⋆ Approximation $\nabla_\beta \eta(\theta)$ to $\nabla \eta(\theta)$.
  - ⋆ Algorithm to accurately estimate $\nabla_\beta \eta$ from a single sample path (POMDPG).
  - ⋆ Accuracy of approximation depends on parameter of the algorithm ($\beta \in [0,1)$); bias/variance trade-off.
  - ⋆ $1/(1-\beta)$ relates to mixing time of underlying Markov chain.
  - ⋆ Line search using only gradient estimates.
  - ⋆ Many successful applications.

Advertisement

- Papers available from http://csl.anu.edu.au.
- Two research positions available in the Machine Learning Group at the Australian National University.