Contents

11.5 Estimating a value function

11.5 Estimating a value function

Value functions have appeared in a surprising range of contexts in this book.

(i) The usual home for value functions is within the field of optimization. In the setting of this book, this means MDPs. Chapter 9 provides many examples, following the introduction for the single server queue presented in Chapter 3.

(ii) The stability theory for Markov chains and networks in this book is centered around Condition (V3). This is closely related to Poisson's inequality, which is itself a generalization of the average-cost value function.

(iii) Theorem 8.4.1 contains general conditions ensuring that the h-MaxWeight policy is stabilizing. The essence of the proof is that the function h is an approximation to Poisson's equation under the assumptions of the theorem.

(iv) We have just seen how approximate solutions to Poisson's equation can be used to dramatically accelerate simulation.

TD Learning

Notation: h value function

h^θ approximation

ψ^θ its gradient: $\psi^\theta(x) := \nabla_\theta h^\theta(x)$

L_2 *error*:

$$\mathcal{E}(\theta) = \|h - h^\theta\|_\pi^2 := \mathsf{E}_\pi[|h(X(0)) - h^\theta(X(0))|^2] \qquad (11.61)$$

Goal: Find θ minimizing this error

TD Learning

Notation: h value function

h^θ approximation

ψ^θ its gradient: $\psi^\theta(x) := \nabla_\theta h^\theta(x)$

L_2 *error*:

$$\mathcal{E}(\theta) = \|h - h^\theta\|_\pi^2 := \mathsf{E}_\pi[|h(X(0)) - h^\theta(X(0))|^2] \tag{11.61}$$

Goal: Find θ minimizing this error

Gradient:

$$\nabla_\theta \|h^\theta - h\|_\pi^2 = 2\mathsf{E}_\pi[(h^\theta(X) - h(X))\psi^\theta(X)]$$

Solution for linear parameterization:

$$\theta^* = M_\psi^{-1} b_\psi, \qquad \text{where } M_\psi = \mathsf{E}[\psi(X)\psi(X)^{\mathrm{T}}]$$

$$b_\psi = \mathsf{E}[h(X)\psi(X)]$$

TD Learning for Discounted Cost

Notation: h value function

$$h = R_\gamma c \qquad R_\gamma = \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} P^t$$

Solution for linear parameterization:

$$\theta^* = M_\psi^{-1} b_\psi, \qquad \text{where } M_\psi = \mathsf{E}[\psi(X)\psi(X)^{\intercal}]$$

$$b_\psi = \mathsf{E}[h(X)\psi(X)]$$

TD Learning for Discounted Cost

Notation: $\quad h$ value function

$$h = R_\gamma c \qquad R_\gamma = \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} P^t$$

Hilbert space notation:

$$\begin{aligned} b_\psi &= \mathsf{E}[h(X)\psi(X)] \\ &= \langle R_\gamma c \,, \psi \rangle_\pi \end{aligned}$$

Solution for linear parameterization:

$$\theta^* = M_\psi^{-1} b_\psi, \qquad \text{where } M_\psi = \mathsf{E}[\psi(X)\psi(X)^{\intercal}]$$

$$b_\psi = \mathsf{E}[h(X)\psi(X)]$$

TD Learning for Discounted Cost

Notation: h value function

$$h = R_\gamma c \qquad R_\gamma = \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} P^t$$

Hilbert space notation:

$$b_\psi = \mathsf{E}[h(X)\psi(X)]$$
$$= \langle R_\gamma c \,, \psi \rangle_\pi$$

Adjoint representation:

$$b_\psi = \langle c \,, R_\gamma^\dagger \psi \rangle_\pi$$

Solution for linear parameterization:

$$\theta^* = M_\psi^{-1} b_\psi, \qquad \text{where } M_\psi = \mathsf{E}[\psi(X)\psi(X)^{\mathsf{T}}]$$
$$b_\psi = \mathsf{E}[h(X)\psi(X)]$$

TD Learning for Discounted Cost

Notation: h value function

$$h = R_\gamma c \qquad R_\gamma = \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} P^t$$

Hilbert space notation:

$$b_\psi = \mathsf{E}[h(X)\psi(X)] = \langle R_\gamma c , \psi \rangle_\pi$$

Adjoint representation:

$$b_\psi = \langle c , R_\gamma^\dagger \psi \rangle_\pi$$

Adjoint: Resolvent for *time-reversed process*:

$$R_\gamma^\dagger g(x) = \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} \mathsf{E}[g(X(-t)) \mid X(0) = x], \qquad x \in \mathsf{X}$$

Solution for linear parameterization:

$$\theta^* = M_\psi^{-1} b_\psi, \qquad \text{where } M_\psi = \mathsf{E}[\psi(X)\psi(X)^{\mathsf{T}}]$$

$$b_\psi = \mathsf{E}[h(X)\psi(X)]$$

TD Learning for Discounted Cost

Algorithm:

Elligibility vectors: $$\varphi(k) = \sum_{t=0}^{k} (1+\gamma)^{-t-1} \psi\left(X(k-t)\right)$$

Law of Large Number approximations:

$$b_\psi^n = \frac{1}{n} \sum_{k=1}^{n} \varphi(k) c(X(k))$$

$$M_\psi^n = \frac{1}{n} \sum_{k=1}^{n} \psi(k) \psi(k)^T$$

$$R_\gamma^\dagger g\,(x) = \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} \mathsf{E}[g(X(-t)) \mid X(0) = x], \qquad x \in \mathsf{X}$$

Solution for linear parameterization:

$$\theta^* = M_\psi^{-1} b_\psi, \qquad \text{where } M_\psi = \mathsf{E}[\psi(X)\psi(X)^{\mathrm{T}}]$$

$$\begin{aligned} b_\psi &= \mathsf{E}[h(X)\psi(X)] \\ &= \langle c\,, R_\gamma^\dagger \psi \rangle_\pi \end{aligned}$$

TD Learning for Discounted Cost

Algorithm:

Elligibility vectors: $$\varphi(k) = \sum_{t=0}^{k} (1+\gamma)^{-t-1} \psi\left(X(k-t)\right)$$

Law of Large Number approximations:

$$b_\psi^n = \frac{1}{n} \sum_{k=1}^{n} \varphi(k) c(X(k))$$

$$M_\psi^n = \frac{1}{n} \sum_{k=1}^{n} \psi(k) \psi(k)^T$$

Estimate:

$$\theta(n) = [M_\psi^n]^{-1} b_\psi^n$$

Inverse recursively computed

$$R_\gamma^\dagger g\,(x) = \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} \mathsf{E}[g(X(-t)) \mid X(0) = x], \qquad x \in \mathsf{X}$$

Solution for linear parameterization:

$$\theta^* = M_\psi^{-1} b_\psi, \qquad \text{where } M_\psi = \mathsf{E}[\psi(X)\psi(X)^{\mathrm{T}}]$$

$$b_\psi = \mathsf{E}[h(X)\psi(X)] = \langle c\,, R_\gamma^\dagger \psi \rangle_\pi$$

Approximate Dynamic Programming using Fluid and Diffusion Approximations

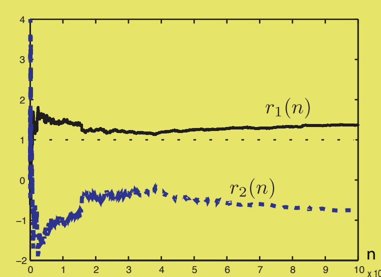
with Applications to Power Management

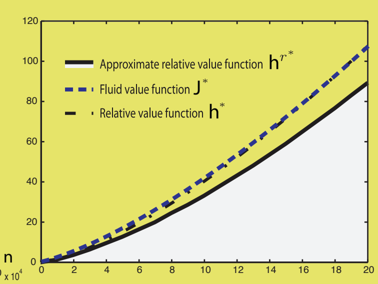
Speaker: Dayu Huang

Wei Chen, Dayu Huang, Ankur A. Kulkarni,[1] Jayakrishnan Unnikrishnan, Quanyan Zhu, Prashant Mehta, Sean Meyn, and Adam Wierman[2]

Coordinated Science Laboratory, UIUC
Dept. of IESE, UIUC [1]
Dept. of CS, California Inst. of Tech. [2]

National Science Foundation (ECS-0523620 and CCF-0830511),
ITMANET DARPA RK 2006-07284, and Microsoft Research





Introduction

MDP model

Control

$$X(t+1) = X(t) + f(X(t), U(t), W(t+1))$$

i.i.d

Cost $\quad c(x,u)$

Minimize average cost $\quad \limsup_{n\to\infty} \frac{1}{n}\sum_{t=0}^{n-1} \mathsf{E}[c(X(t),U(t))]$

Introduction

MDP model

Control

$$X(t+1) = X(t) + f(X(t), U(t), W(t+1))$$

i.i.d

Cost $\quad c(x,u)$

Minimize average cost $\quad \limsup_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \mathsf{E}[c(X(t),U(t))]$

Generator

$$\mathcal{D}_u h\,(x) := \mathsf{E}[h(X(t+1)) - h(X(t)) | X(t) = x, U(t) = u]$$

Introduction

MDP model

Control

$$X(t+1) = X(t) + f(X(t), U(t), W(t+1))$$

i.i.d

Cost $c(x,u)$

Minimize average cost $\limsup_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \mathsf{E}[c(X(t),U(t))]$

Average Cost Optimality Equation (ACOE)

$$\min_u \Big(c(x,u) + \mathcal{D}_u h^*(x) \Big) = \eta^*$$

Generator $\mathcal{D}_u h(x) := \mathsf{E}[h(X(t+1)) - h(X(t)) | X(t) = x, U(t) = u]$

h^* Relative value function

Solve ACOE and Find h^*

TD Learning

$$\min_u \big(c(x,u) + \mathcal{D}_u h^*(x)\big) = \eta^*$$

- The "curse of dimensionality":

  Complexity of solving ACOE grows exponentially with the dimension of the state space.

- Approximate h^* within a finite-dimensional function class

$$h^r = \sum r_i \psi_i$$

- Criterion: minimize the mean-squre error

$$\mathsf{E}_\pi[(h(X(0)) - h^r(X(0)))^2]$$

solved by stochastic approximation algorithms

TD Learning

$$\min_u \big(c(x,u) + \mathcal{D}_u h^*(x)\big) = \eta^*$$

- The "curse of dimensionality":

  Complexity of solving ACOE grows exponentially with the dimension of the state space.

- Approximate h^* within a finite-dimensional function class

$$h^r = \sum r_i \psi_i$$

- Criterion: minimize the mean-squre error

$$\mathsf{E}_\pi[(h(X(0)) - h^r(X(0)))^2]$$

solved by stochastic approximation algorithms

Problem: How to select the basis functions $\{\psi_i, 1 \le i \le d\}$?

key to the success of TD learning

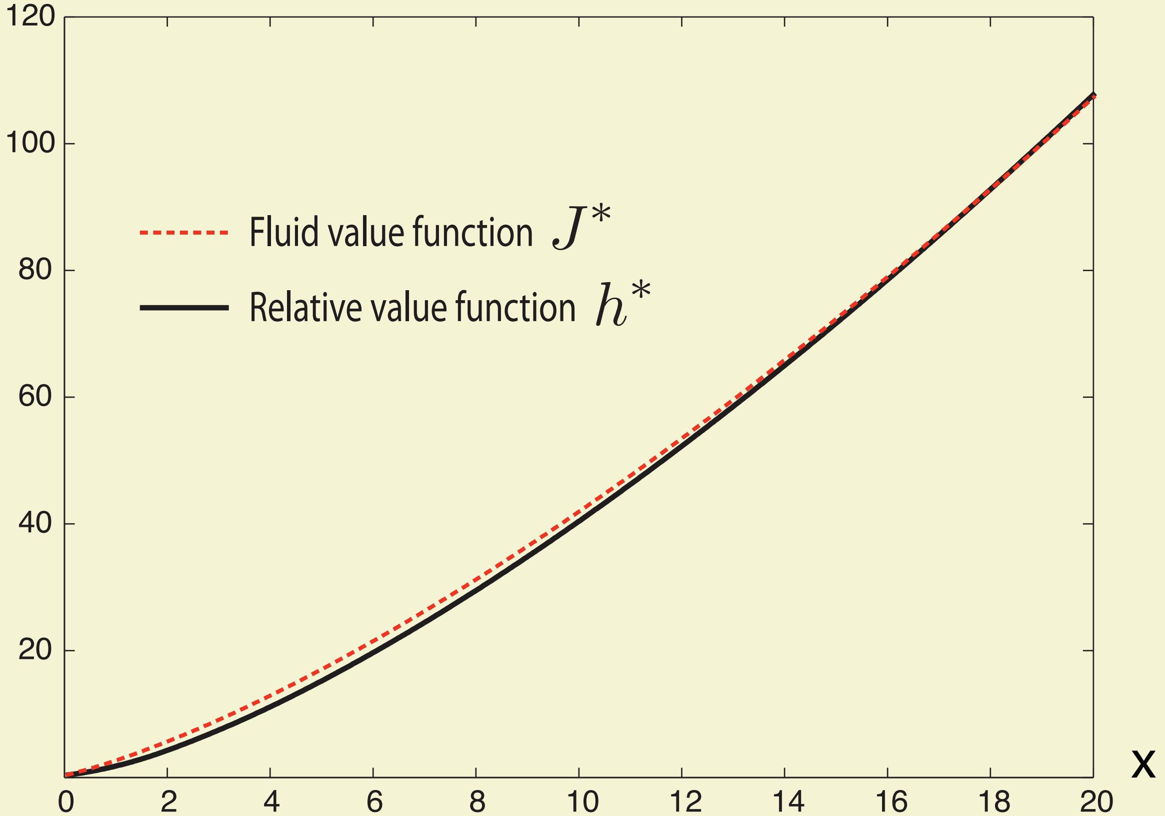
Approach Based on Fluid and Diffusion Models

this talk: fluid model

Value function of the fluid model J^*

Total cost for an associated deterministic model

is a tight approximation to h^*



J^* can be used as a part of the basis $\{\psi_i\}$

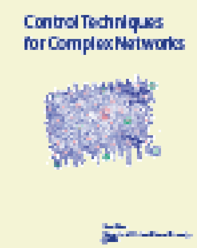
Related Work

Multiclass queueing network

$$\frac{h^*(x)}{J^*(x)} \to 1$$

Meyn 1997, Meyn 1997b

optimal control	Chen and Meyn 1999
simulation	Hendersen et.al. 2003
network scheduling and routing	Veatch 2004 Moallemi, Kumar and Van Roy 2006
	Meyn 2007 Control Techniques for Complex Networks
other approaches	Tsitsiklis and Van Roy 1997 Mannor, Menache and Shimkin 2005



Related Work

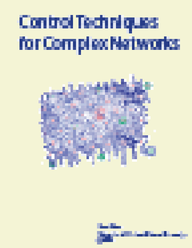
Multiclass queueing network

$$\frac{h^*(x)}{J^*(x)} \to 1$$

Meyn 1997, Meyn 1997b

optimal control	Chen and Meyn 1999
simulation	Hendersen et.al. 2003
network scheduling	Veatch 2004
and routing	Moallemi, Kumar and Van Roy 2006

Meyn 2007 — Control Techniques for Complex Networks



Taylor series approximation — this work

Power Management via Speed Scaling

Bansal, Kimbrel and Pruhs 2007

Wierman, Andrew and Tang 2009

Single processor

job arrivals

$$Q(t+1) = Q(t) - U(t) + A(t+1)$$

processing rate $U(t)$
determined by the current power

Control the processing speed to balance delay and energy costs

$$c(x,u) = x + \beta\mathcal{P}(u)$$

Processor design: polynomial cost $\mathcal{P}(u) \propto u^{\varrho}$

Kaxiras and Martonosi 2008
Wierman, Andrew and Tang 2009

This talk

We also consider $\mathcal{P}(u) \propto e^{\kappa u}$
for wireless communication applications

Fluid Model

MDP $X(t+1) = X(t) + f(X(t), U(t), W(t+1))$

Fluid model:

$$\frac{d}{dt}x(t) = \overline{f}(x(t), u(t))$$

$$\overline{f}(x,u) := \mathsf{E}[f(x,u,W(1))]$$

Total Cost

$$J^*(x) = \inf_{\mathbf{u}} \int_0^{T_0} c(x(t), u(t))dt, x(0) = x.$$

Total Cost Optimality Equation (TCOE) for the fluid model:

$$\min_u \bigl(c(x,u) + \nabla J^*(x) \cdot \overline{f}(x,u)\bigr) = 0$$

Why Fluid Model?

MDP $X(t+1) = X(t) + f(X(t), U(t), W(t+1))$

First order Taylor series approximation

$$\mathcal{D}_u J^*(x) \approx \mathsf{E}_{x,u}\big[\nabla J^*(X(0))(X(1) - X(0))\big]$$
$$= \nabla J^*(x)\overline{f}(x,u)$$

Why Fluid Model?

MDP $X(t+1) = X(t) + f(X(t), U(t), W(t+1))$

First order Taylor series approximation

$$\mathcal{D}_u J^*(x) \approx \mathsf{E}_{x,u}\big[\nabla J^*(X(0))(X(1) - X(0))\big]$$
$$= \nabla J^*(x)\overline{f}(x,u)$$

TCOE

$$\min_u \Big(c(x,u) + \nabla J^*(x) \cdot \overline{f}(x,u)\Big) = 0$$
$$\approx c(x,u) + \mathcal{D}_u J^*(x)$$

ACOE

$$\min_u \Big(c(x,u) + \mathcal{D}_u h^*(x)\Big) = \eta^*$$

J^* almost solves the ACOE

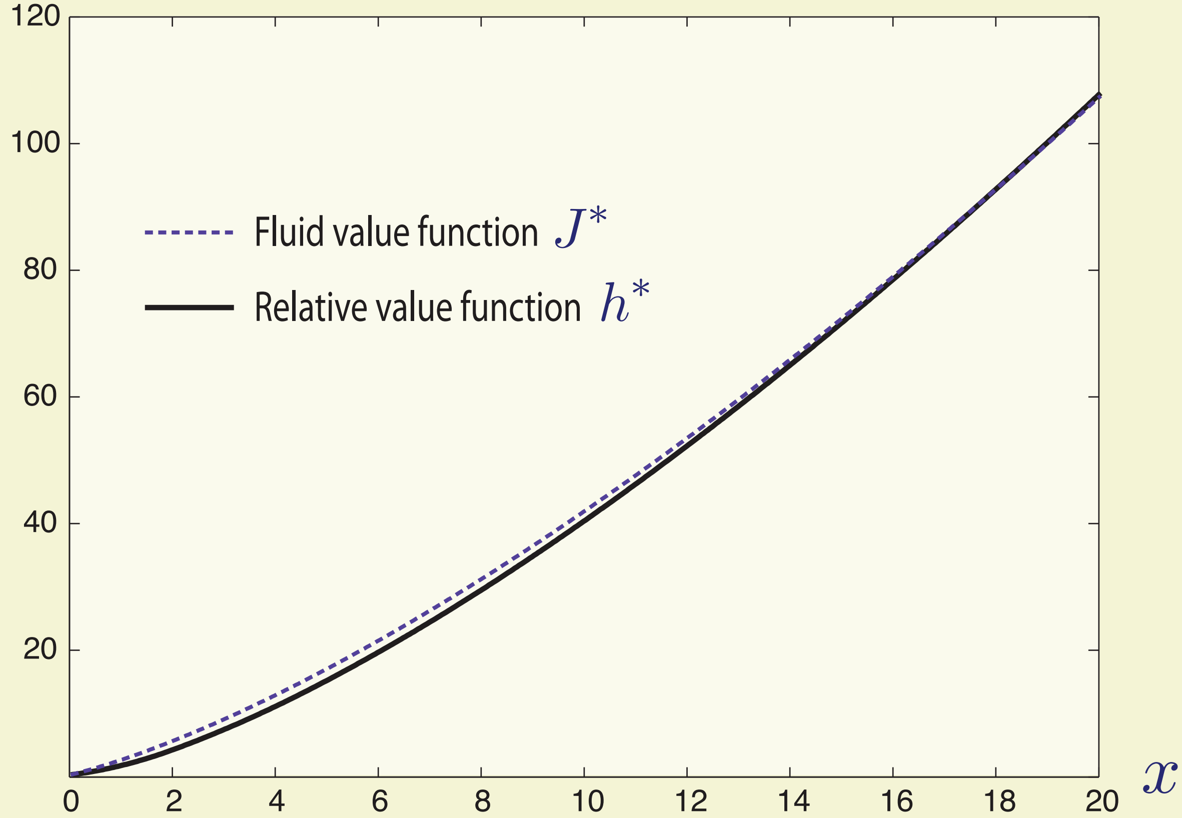
Simple but powerful idea!

Approach Based on Fluid and Diffusion Models

this talk: fluid model

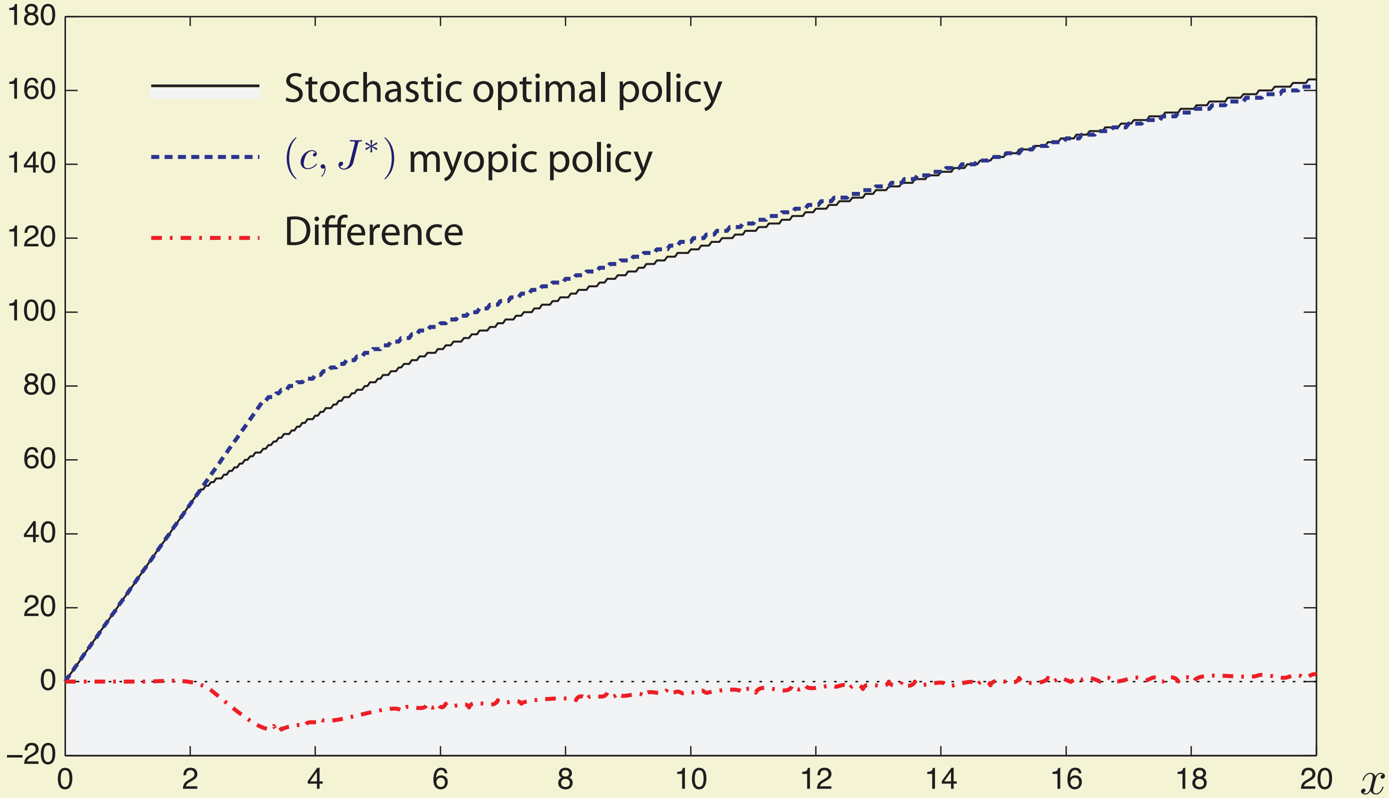
Value function of the fluid model J^* is a tight approximation to h^*

Total cost for an associated deterministic model



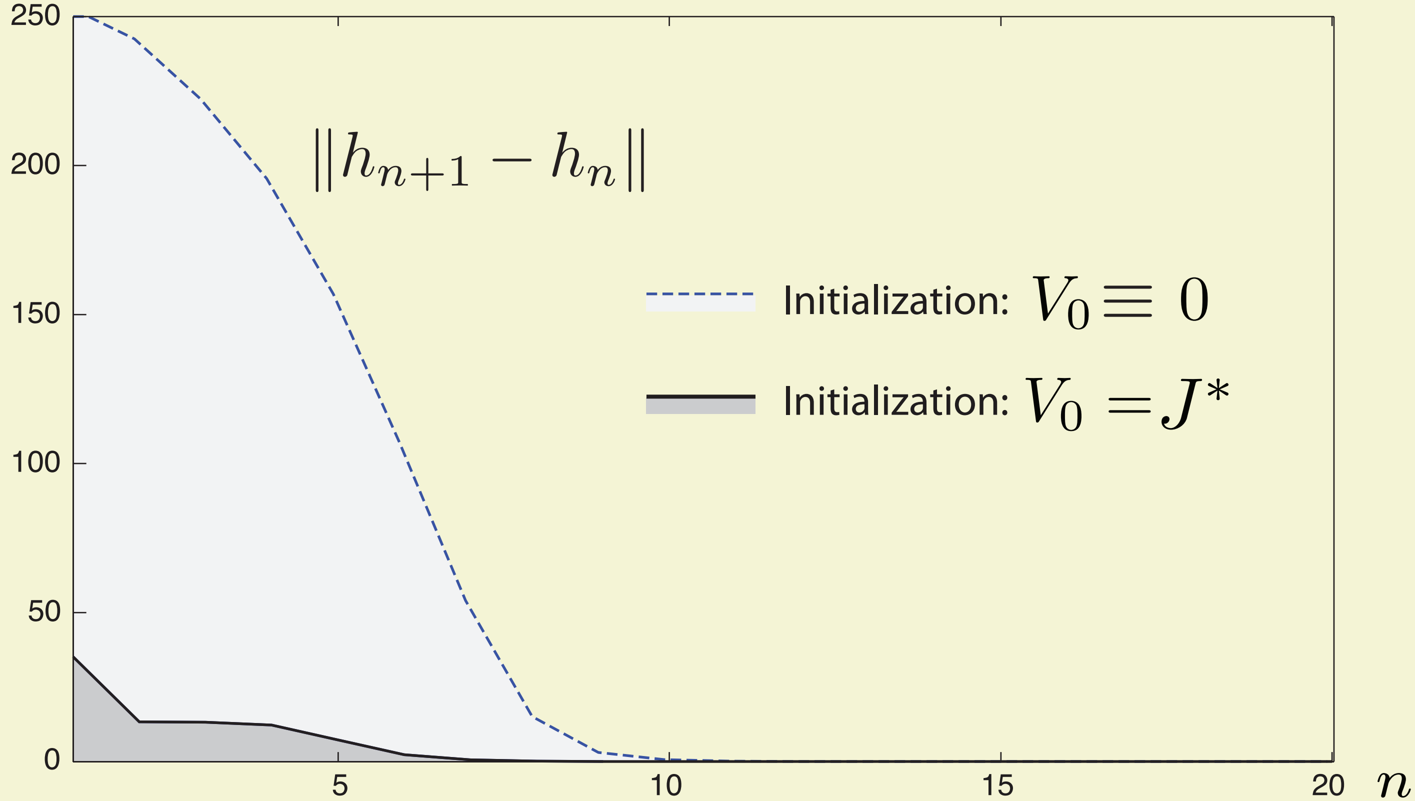
J^* can be used as a part of the basis $\{\psi_i\}$

Policy



The optimal policy compared to the (c, J^*)-myopic policy for the quadratic cost function

Value Iteration



The convergence of value iteration for the quadratic cost function

The error $\|h_{n+1} - h_n\|$ converges to zero *much faster* when the algorithm is initialized using the fluid value function.

(See also [Chen and Meyn 1999])

Approximation of the Cost Function

$$\min_u \big(c(x,u) + \nabla J^*(x) \cdot \overline{f}(x,u) \big) = 0$$
$$\approx c(x,u) + \mathcal{D}_u J^*(x)$$

$$\min_u \big(c(x,u) + \mathcal{D}_u h^*(x) \big) = \eta^*$$

Error Analysis

$$\mathcal{E}(x,u) = c(x,u) + \mathcal{D}_u J^*(x)$$

$$\underline{\mathcal{E}}(x) = \min_{0 \le u \le x} \mathcal{E}(x,u)$$ $\approx$ constant?

$$\min_u \left(c(x,u) + \nabla J^*(x) \cdot \overline{f}(x,u) \right) = 0$$
$$\approx c(x,u) + \mathcal{D}_u J^*(x)$$

Approximation of the Cost Function

$$\min_u \left(c(x,u) + \mathcal{D}_u h^*(x) \right) = \eta^*$$

Error Analysis $\quad \mathcal{E}(x,u) = c(x,u) + \mathcal{D}_u J^*(x)$

$$\underline{\mathcal{E}}(x) = \min_{0 \le u \le x} \mathcal{E}(x,u) \quad \approx \text{constant?}$$

Surrogate cost $\quad c^\circ(x,u) = c(x,u) - \underline{\mathcal{E}}(x) + \eta^\circ$

$$\min_{0 \le u \le x} \{ c^\circ(x,u) - \eta^\circ + \mathcal{D}_u J^*(x) \} = 0$$

Bounds on $\underline{\mathcal{E}}(x)$?

Structure Results on the Fluid Solution

$$\min_u \left(c(x,u) + \nabla J^*(x) \cdot \overline{f}(x,u) \right) = 0$$

$$\approx c(x,u) + \mathcal{D}_u J^*(x)$$

$$Polynomial\ cost \quad c(x,u) = x + \beta([u-\alpha]_+)^{\varrho}$$

$$Exponential\ cost \quad c(x,u) = x + \beta[e^{\kappa u} - e^{\kappa \alpha}]_+$$

Proposition 0.1 *For any of the cost functions defined above, the fluid value function J^* is increasing, convex, and its second derivative $\nabla^2 J^*$ is non-increasing. Moreover, For polynomial cost the value function and optimal policy are given by, respectively,*

$$J^*(x) = x^{\frac{2\varrho-1}{\varrho}} \frac{\varrho}{2\varrho-1} \left(\frac{1}{\beta(\varrho-1)} \right)^{\frac{\varrho-1}{\varrho}}$$

$$\phi^{\mathrm{F}*}(x) = \left(\frac{x}{\beta(\varrho-1)} \right)^{1/\varrho} + \alpha, \qquad x \in \mathbb{R}_+.$$

Lower Bound

$$\min_u \big(c(x,u) + \nabla J^*(x) \cdot \overline{f}(x,u) \big) = 0$$

$$\approx c(x,u) + \mathcal{D}_u J^*(x)$$

$$\mathcal{E}(x,u) = c(x,u) + \mathcal{D}_u J^*(x)$$

Lemma 2 $\mathcal{E}(x,u) \geq 0$ *everywhere, giving* $c \geq c^\circ - \eta^\circ$.

$$\begin{aligned} \mathcal{D}_u J^*(x) &= \mathsf{E}_{x,u}[J^*(Q(1)) - J^*(Q(0))] \\ &\geq \mathsf{E}_{x,u}[\nabla J^*(Q(0)) \cdot ((Q(1)) - Q(0))] \qquad \text{Convexity of } J^* \\ &= \nabla J^*(x) \cdot (-u + \alpha) \end{aligned}$$

Lower Bound

$$\min_u \big(c(x,u) + \nabla J^*(x) \cdot \overline{f}(x,u)\big) = 0$$

$$\approx c(x,u) + \mathcal{D}_u J^*(x)$$

$$\mathcal{E}(x,u) = c(x,u) + \mathcal{D}_u J^*(x)$$

Lemma 2 $\mathcal{E}(x,u) \geq 0$ *everywhere, giving* $c \geq c^\circ - \eta^\circ$.

$$\begin{aligned}
\mathcal{D}_u J^*(x) &= \mathsf{E}_{x,u}[J^*(Q(1)) - J^*(Q(0))] \\
&\geq \mathsf{E}_{x,u}[\nabla J^*(Q(0)) \cdot ((Q(1)) - Q(0))] \qquad \text{Convexity of } J^* \\
&= \nabla J^*(x) \cdot (-u + \alpha)
\end{aligned}$$

$$\begin{aligned}
\mathcal{E}(x,u) &= c(x,u) + \mathcal{D}_u J^*(x) \\
&\geq c(x,u) + \nabla J^*(x) \cdot (-u + \alpha) \\
&\geq 0
\end{aligned}$$

Upper Bound

$$\min_u \big(c(x,u) + \nabla J^*(x) \cdot \overline{f}(x,u) \big) = 0$$

$$\approx c(x,u) + \mathcal{D}_u J^*(x)$$

$$\mathcal{E}(x,u) = c(x,u) + \mathcal{D}_u J^*(x)$$

Lemma 3 *For the polynomial cost with* $\varrho = 2$, $\beta = \frac{1}{2}$, *we have* $\underline{\mathcal{E}}(x) = \mathcal{O}(\sqrt{x})$, *and hence* $c(x,u) \le c^\circ(x,u) + \mathcal{O}(\sqrt{x})$.

$$\begin{aligned}
\mathcal{D}_u J^*(x) &:= \mathsf{E}_{x,u}[J^*(Q(1)) - J^*(Q(0))] \\
&= \nabla J^*(x) \cdot (-u + \alpha) \\
&\qquad + \tfrac{1}{2}\mathsf{E}\left[\nabla^2 J^*(\overline{Q}) \cdot (-u + A(1))^2\right] \qquad x - u + A(1) \le \overline{Q} \le x \\
&\le \nabla J^*(x) \cdot (-u + \alpha) \qquad \textit{second derivative } \nabla^2 J^* \textit{ is non-increasing.} \\
&\qquad + \tfrac{1}{2}\mathsf{E}\left[\nabla^2 J^*(x - u) \cdot (-u + A(1))^2\right]
\end{aligned}$$

Upper Bound

$$\min_u \big(c(x,u) + \nabla J^*(x) \cdot \overline{f}(x,u)\big) = 0$$

$$\approx c(x,u) + \mathcal{D}_u J^*(x)$$

$$\mathcal{E}(x,u) = c(x,u) + \mathcal{D}_u J^*(x)$$

Lemma 3 *For the polynomial cost with* $\varrho = 2$, $\beta = \frac{1}{2}$, *we have* $\underline{\mathcal{E}}(x) = \mathcal{O}(\sqrt{x})$, *and hence* $c(x,u) \le c^\circ(x,u) + \mathcal{O}(\sqrt{x})$.

$$\begin{aligned}
\mathcal{D}_u J^*(x) &:= \mathsf{E}_{x,u}[J^*(Q(1)) - J^*(Q(0))] \\
&= \nabla J^*(x) \cdot (-u + \alpha) \\
&\qquad + \tfrac{1}{2}\mathsf{E}\left[\nabla^2 J^*(\overline{Q}) \cdot (-u + A(1))^2\right] \qquad x - u + A(1) \le \overline{Q} \le x \\
&\le \nabla J^*(x) \cdot (-u + \alpha) \qquad \textit{second derivative } \nabla^2 J^* \textit{ is non-increasing.} \\
&\qquad + \tfrac{1}{2}\mathsf{E}\left[\nabla^2 J^*(x - u) \cdot (-u + A(1))^2\right]
\end{aligned}$$

$$\begin{aligned}
\underline{\mathcal{E}}(x) &\le \mathcal{E}(x, \phi^{\mathrm{F}*}(x)) \\
&\le \tfrac{1}{2}\mathsf{E}\left[\nabla^2 J^*(x - \phi^{\mathrm{F}*}(x)) \cdot (-\phi^{\mathrm{F}*}(x) + A(1))^2\right].
\end{aligned}$$

$$c(x,u) = c^\circ(x,u) + \underline{\mathcal{E}}(x) - \eta^\circ \le c^\circ(x,u) + O(\sqrt{x})$$

Upper Bound

$$\min_u \big(c(x,u) + \nabla J^*(x) \cdot \overline{f}(x,u)\big) = 0$$

$$\approx c(x,u) + \mathcal{D}_u J^*(x)$$

$$\mathcal{E}(x,u) = c(x,u) + \mathcal{D}_u J^*(x)$$

Lemma 3 *For the polynomial cost with* $\varrho = 2$, $\beta = \frac{1}{2}$, *we have* $\underline{\mathcal{E}}(x) = \mathcal{O}(\sqrt{x})$, *and hence* $c(x,u) \le c^\circ(x,u) + \mathcal{O}(\sqrt{x})$.

$$\begin{aligned}
\mathcal{D}_u J^*(x) &:= \mathsf{E}_{x,u}[J^*(Q(1)) - J^*(Q(0))] \\
&= \nabla J^*(x) \cdot (-u + \alpha) \\
&\qquad + \tfrac{1}{2}\mathsf{E}\left[\nabla^2 J^*(\overline{Q}) \cdot (-u + A(1))^2\right] \\
&\le \nabla J^*(x) \cdot (-u + \alpha) \\
&\qquad + \tfrac{1}{2}\mathsf{E}\left[\nabla^2 J^*(x-u) \cdot (-u + A(1))^2\right]
\end{aligned}$$

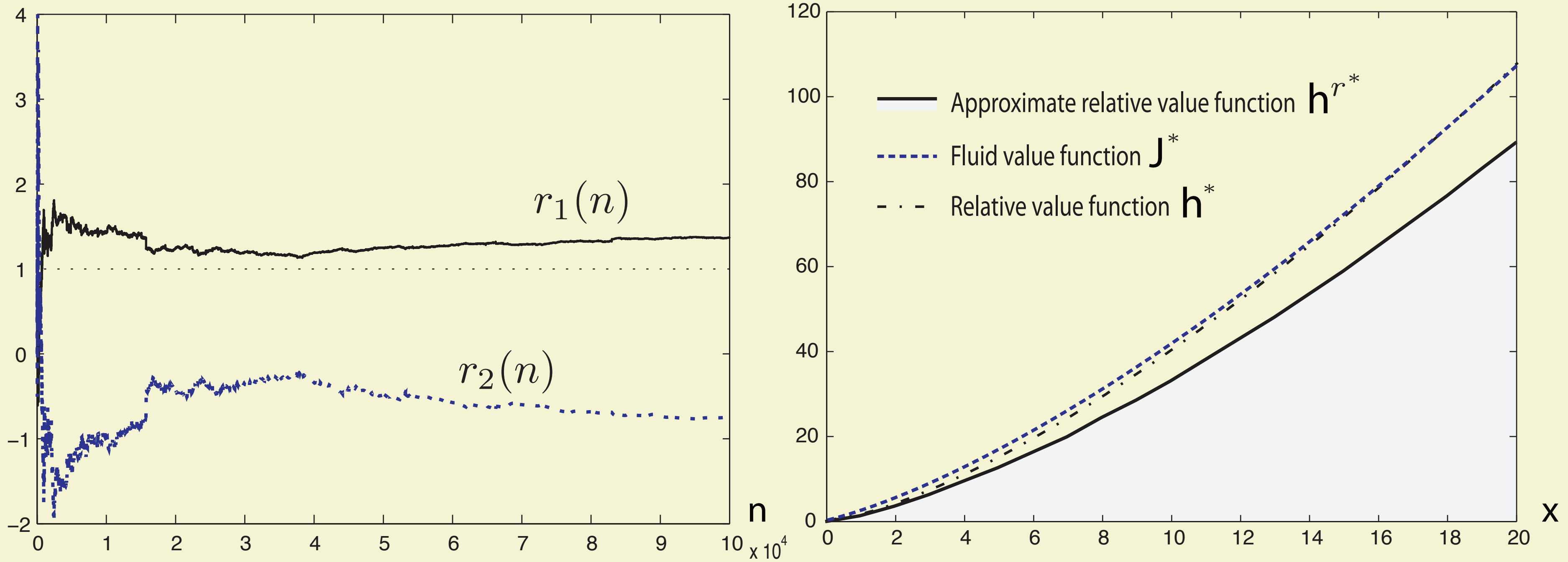
$$x - u + A(1) \le \overline{Q} \le x$$

second derivative $\nabla^2 J^*$ *is non-increasing.*

$$c^\circ(x,u) - \eta^\circ \le c(x,u) \le c^\circ(x,u) + O(\sqrt{x})$$

TD Learning Experiment

Basis functions: $\psi_1(x) = J^*(x), \quad \psi_2(x) = x$

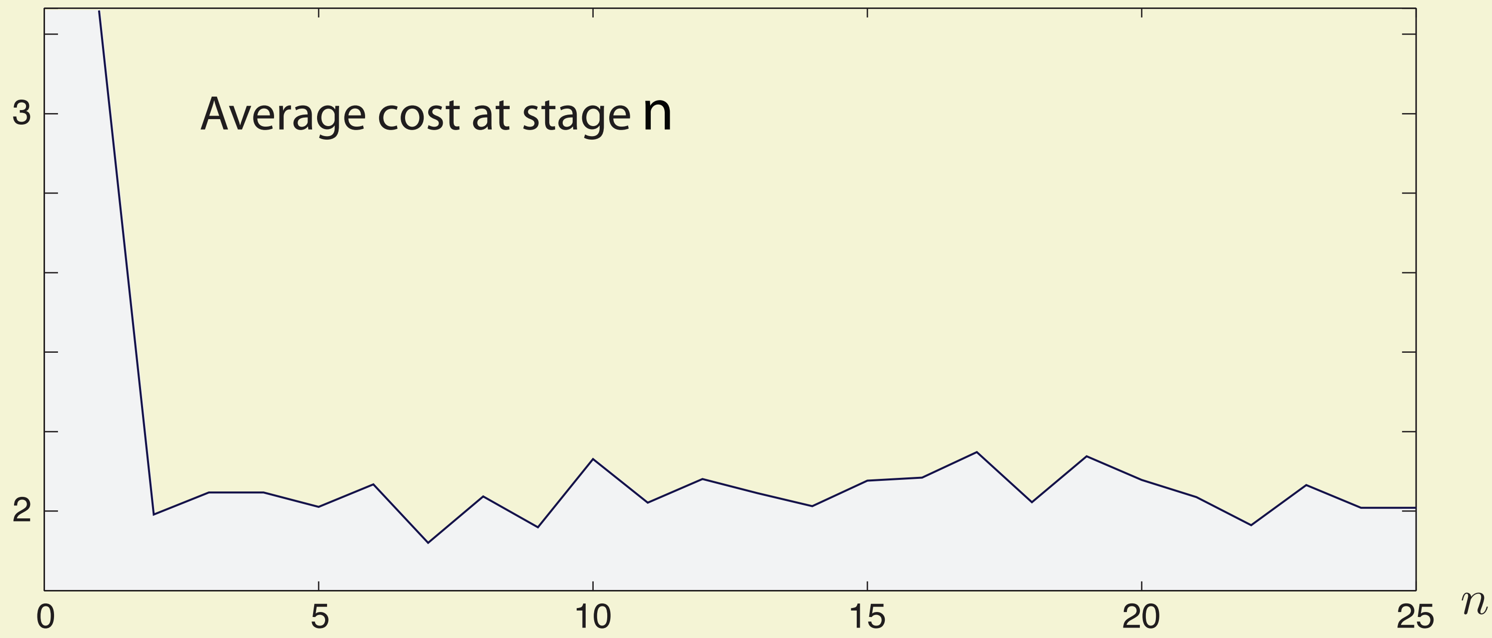


Estimates of Coefficients for the case of quadratic cost

TD Learning with Policy Improvement

(i) Given the policy ϕ^k, find the approximate solution h^k_{TD} to Poisson's equation $\mathcal{D}_{\phi^k} h^k_{\text{TD}} \approx h^k - c_k + \eta_k$, where $c_k(x) = c(x, \phi^k(x))$, and η_k is the average cost.

(ii) Update the policy via $\phi^{k+1}(x) \in \arg\min_u \{c(x,u) + \mathcal{D}_u h^k_{\text{TD}}(x)\}$.



Simulation result for TDPIA with the quadratic cost function, and basis $\{\psi_1, \psi_2\} \equiv \{J^*, x\}$.

Nearly optimal after just a few iterations

Conclusions

- The fluid value function can be used as a part of the basis for TD-learning.

- Motivated by analysis using Taylor series expansion:

  The fluid value function almost solves ACOE. In particular, it solves the ACOE for a slightly different cost function; and the error term can be estimated.

- TD learning with policy improvement gives a near optimal policy in a few iterations, as shown by experiments.

- Application in power management for processors.

Q-Learning
and Pontryagin's Minimum Principle

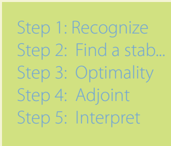
Sean Meyn

Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois

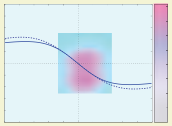
Joint work with Prashant Mehta

Research support: NSF: ECS-0523620
AFOSR: FA9550-09-1-0190

Outline



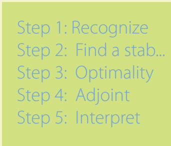
Q-learning for nonlinear state space models



Example: Local approximation

Example: Decentralized control

Outline



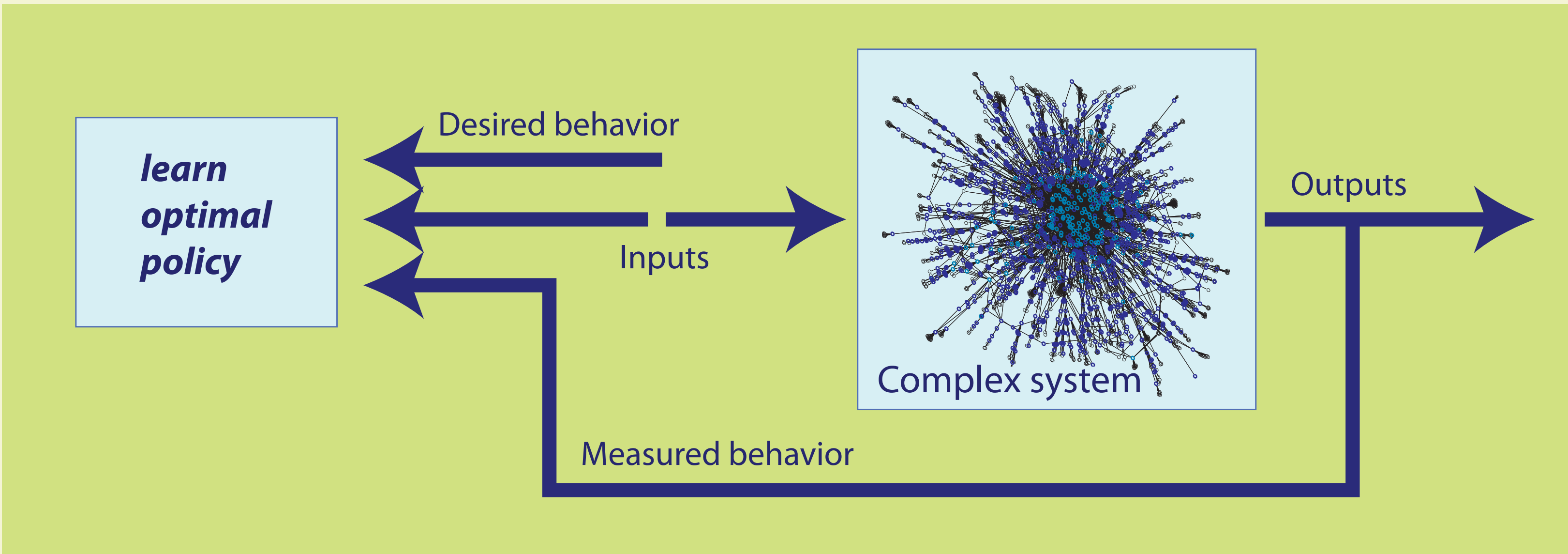
Q-learning for nonlinear state space models

Example: Local approximation

Example: Decentralized control

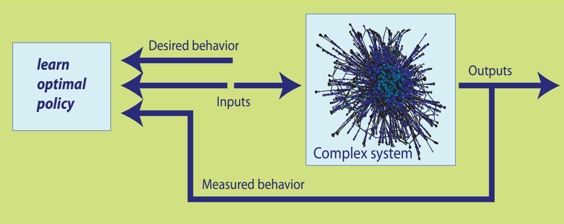
What is Q learning?

Identify optimal policy based on observations:



Watkin's 1992 formulation applied to finite state space MDPs

What is Q learning?



Watkin's 1992 formulation applied to finite state space MDPs

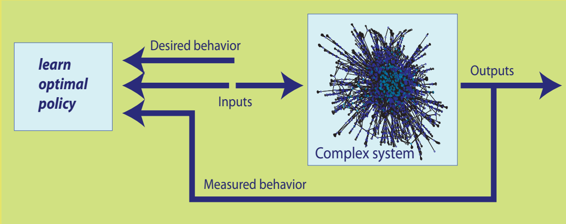
Watkins and P. Dayan, 1992

Goal: Find the best approximation to dynamic programming equations over a parameterized class, based on observations using a non-optimal policy.

Watkin's algorithm known to be effective only for

- Finite state-action space
- Complete parametric family

What is Q learning?



Watkin's 1992 formulation applied to finite state space MDPs

Watkins and P. Dayan, 1992

Goal: Find the best approximation to dynamic programming equations over a parameterized class, based on observations using a non-optimal policy.

Watkin's algorithm known to be effective only for
 Finite state-action space
 Complete parametric family

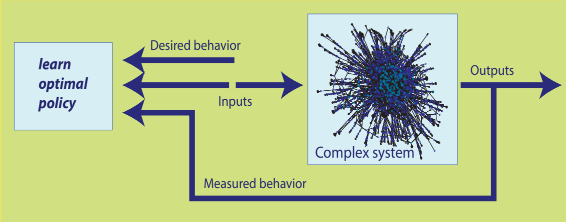
Extensions: when cost depends on control,
 but dynamics are oblivious

Duff, 1995
Tsitsiklis and Van Roy, 1999
Yu and Bertsekas, 2007

Approach: Similar to *differential dynamic programming*

Differential dynamic programming
D. H. Jacobson and D. Q. Mayne
American Elsevier Pub. Co. 1970

What is Q learning?



Watkin's 1992 formulation applied to finite state space MDPs

This lecture:

Deterministic formulation: Nonlinear system on Euclidean space,

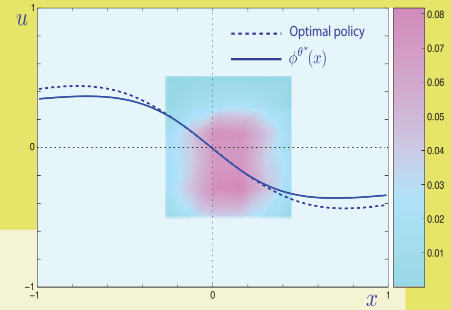
$$\frac{d}{dt}x(t) = f(x(t), u(t)), \qquad t \geq 0$$

Infinite-horizon discounted cost criterion,

$$J^*(x) = \inf \int_0^\infty e^{-\gamma s} c(x(s), u(s))\, ds, \qquad x(0) = x$$

with c a non-negative cost function.

What is Q learning?



Deterministic formulation: Nonlinear system on Euclidean space,

$$\frac{d}{dt}x(t) = f(x(t), u(t)), \qquad t \geq 0$$

Infinite-horizon discounted cost criterion,

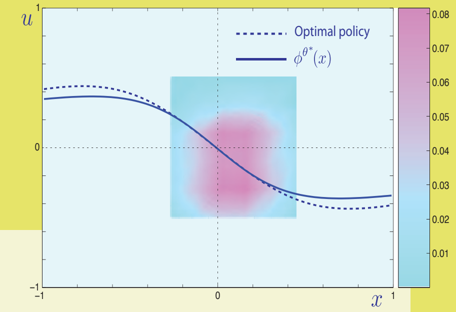
$$J^*(x) = \inf \int_0^\infty e^{-\gamma s} c(x(s), u(s))\, ds, \qquad x(0) = x$$

with c a non-negative cost function.

Differential generator: For any smooth function h,

$$\mathcal{D}_u h\,(x) := (\nabla h\,(x))^{\mathsf{T}} f\,(x, u)$$

What is Q learning?



Deterministic formulation: Nonlinear system on Euclidean space,

$$\frac{d}{dt}x(t) = f(x(t), u(t)), \qquad t \geq 0$$

Infinite-horizon discounted cost criterion,

$$J^*(x) = \inf \int_0^\infty e^{-\gamma s} c(x(s), u(s))\, ds, \qquad x(0) = x$$

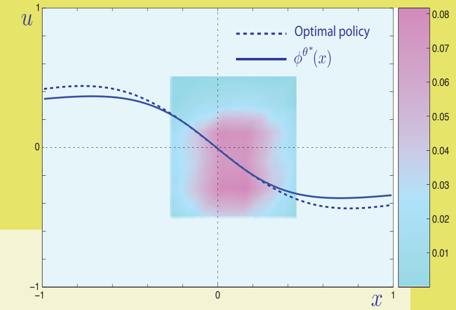
with c a non-negative cost function.

Differential generator: For any smooth function h,

$$\mathcal{D}_u h\,(x) := (\nabla h\,(x))^{\intercal} f(x, u)$$

HJB equation: $$\min_u \Big(c(x, u) + \mathcal{D}_u J^*\,(x) \Big) = \gamma J^*(x)$$

What is Q learning?



Deterministic formulation: Nonlinear system on Euclidean space,

$$\frac{d}{dt}x(t) = f(x(t), u(t)), \qquad t \geq 0$$

Infinite-horizon discounted cost criterion,

$$J^*(x) = \inf \int_0^\infty e^{-\gamma s} c(x(s), u(s))\, ds, \qquad x(0) = x$$

with c a non-negative cost function.

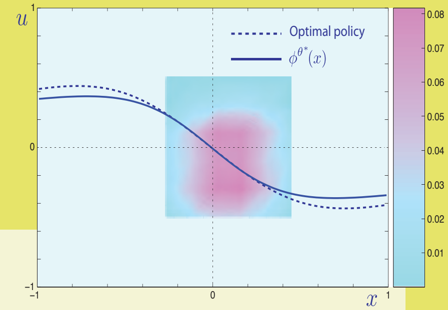
Differential generator: For any smooth function h,

$$\mathcal{D}_u h(x) := (\nabla h(x))^T f(x, u)$$

HJB equation: $$\min_u \Big(c(x, u) + \mathcal{D}_u J^*(x) \Big) = \gamma J^*(x)$$

The *Q-function* of Q-learning is this function of two variables

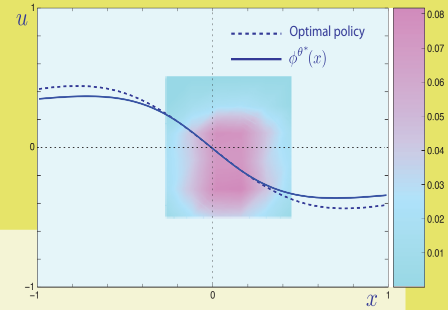
Q learning - Steps towards an algorithm



Sequence of five steps:

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - Steps towards an algorithm



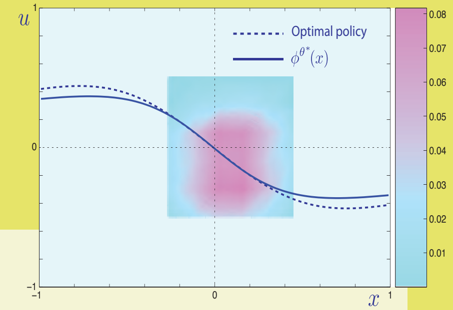
Sequence of five steps:

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Goal - seek the best approximation,
within a parameterized class

$$H^{\theta}(x,u) = \theta^{T}\psi(x,u), \qquad \theta \in \mathbb{R}^d$$

Q learning - Steps towards an algorithm



Step 1: Recognize fixed point equation for the Q-function

Q-function: $$H^*(x,u) \;=\; c(x,u) + \mathcal{D}_u J^*(x)$$

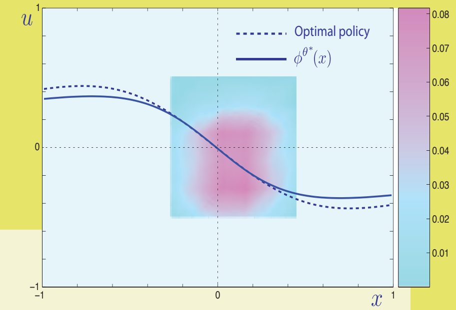
Its minimum: $$\underline{H}^*(x) := \min_{u \in \mathsf{U}} H^*(x,u) = \gamma J^*(x)$$

Fixed point equation:

$$\boxed{\mathcal{D}_u \underline{H}^*(x) = -\gamma\big(c(x,u) - H^*(x,u)\big)}$$

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - Steps towards an algorithm



Step 1: Recognize fixed point equation for the Q-function

Q-function: $$H^*(x,u) = c(x,u) + \mathcal{D}_u J^*(x)$$

Its minimum: $$\underline{H}^*(x) := \min_{u \in \mathsf{U}} H^*(x,u) = \gamma J^*(x)$$

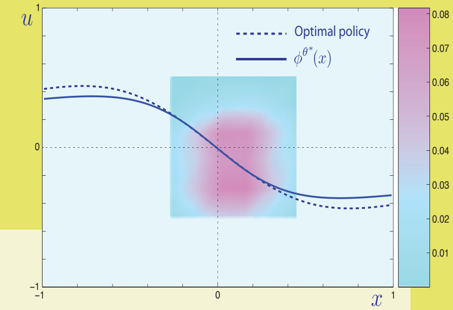
Fixed point equation:

$$\mathcal{D}_u \underline{H}^*(x) = -\gamma\big(c(x,u) - H^*(x,u)\big)$$

Key observation for learning: For any input-output pair,

$$\mathcal{D}_u \underline{H}^*(x) = \frac{d}{dt}\underline{H}^*(x(t))\Big|_{\substack{x=x(t)\\ u=u(t)}}$$

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!



Q learning - LQR example

Linear model and quadratic cost,

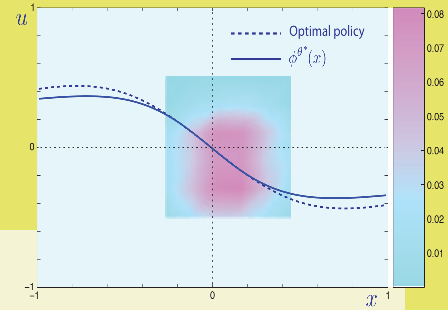
Cost: $$c(x,u) \;=\; \tfrac{1}{2}x^T Q x \;+\; \tfrac{1}{2}u^T R u$$

Q-function: $$H^*(x) = c(x,u) + (Ax + Bu)^T P^* x$$

Solves Riccatti eqn

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - LQR example



Linear model and quadratic cost,

Cost:

$$c(x,u) \;=\; \tfrac{1}{2}x^T Q x \;+\; \tfrac{1}{2}u^T R u$$

Q-function:

$$H^*(x) = c(x,u) + (Ax + Bu)^T P^* x$$

Solves Riccati eqn

Q-function approx:

$$H^\theta(x,u) = c(x,u) + \tfrac{1}{2}\sum_{i=1}^{d_x} \theta_i^x x^T E^i x + \sum_{j=1}^{d_{xu}} \theta_j^x x^T F^i u$$

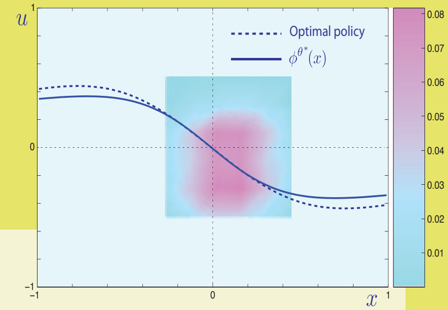
Minimum:

$$\underline{H}^\theta(x) = \tfrac{1}{2}x^T \left(Q + E^\theta - F^{\theta^T} R^{-1} F^\theta\right) x$$

Minimizer:

$$u^\theta(x) = \phi^\theta(x) = -R^{-1} F^\theta x$$

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!



Q learning - Steps towards an algorithm

Step 2: Stationary policy that is ergodic?

Assume the LLN holds for continuous functions

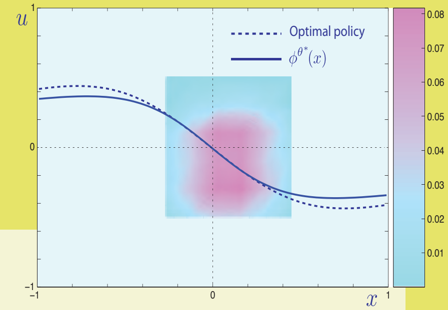
$$F \colon \mathbb{R}^{\ell} \times \mathbb{R}^{\ell_u} \to \mathbb{R}$$

As $T \to \infty$,

$$\frac{1}{T}\int_0^T F(x(t), u(t))\, dt \longrightarrow \int_{\mathsf{X}\times\mathsf{U}} F(x, u)\, \varpi(dx, du)$$

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - Steps towards an algorithm



Step 2: Stationary policy that is ergodic?

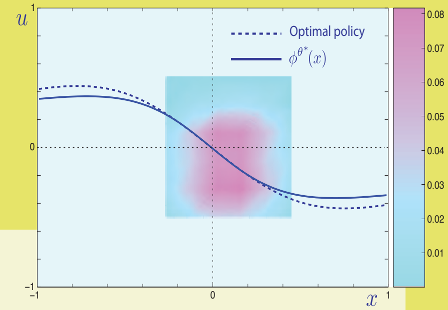
Suppose for example the input is scalar, and the system is *stable*

[Bounded-input/Bounded-state]

Can try a linear combination of sinusouids

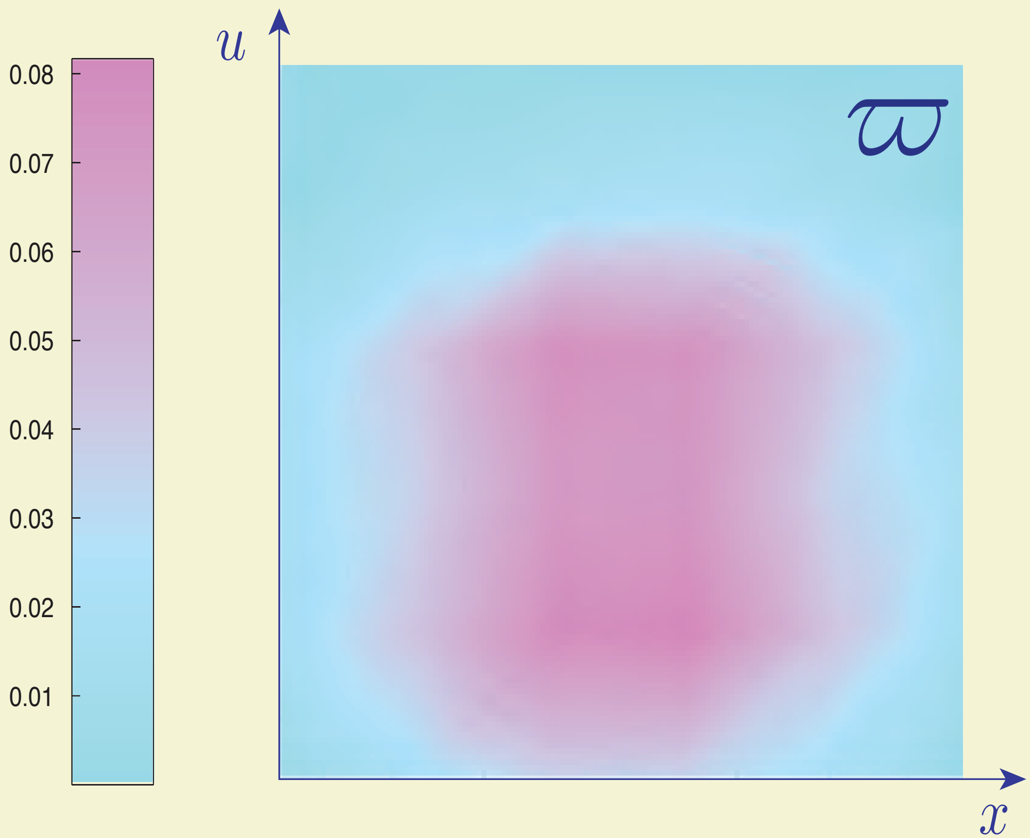
Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - Steps towards an algorithm



Step 2: Stationary policy that is ergodic?

Suppose for example the input is scalar, and the system is *stable*
[Bounded-input/Bounded-state]

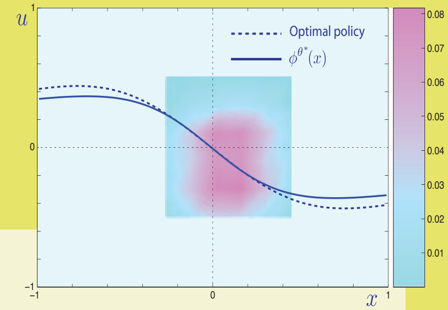


Can try a linear combination of sinusoids

$$u(t) = A(\sin(t) + \sin(\pi t) + \sin(et))$$

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - Steps towards an algorithm



Step 3: Bellman error

$$\mathcal{L}^\theta(x,u) \quad := \quad \mathcal{D}_u \underline{H}^\theta\,(x) + \gamma(c - H^\theta)\,, \qquad \theta \in \mathbb{R}^d$$

Based on observations, minimize the mean-square Bellman error:

$$\mathcal{E}_{\mathrm{Bell}}(\theta) \quad := \quad \int [\mathcal{L}^\theta]^2\, \varpi(dx,du) \;:=\; \langle \mathcal{L}^\theta,\, \mathcal{L}^\theta\, \rangle_\varpi$$

First order condition for optimality: $\quad \langle \mathcal{L}^\theta, \mathcal{D}_u \underline{\psi}_i^\theta - \gamma \psi_i^\theta \rangle_\varpi = 0$

with $\underline{\psi}_i^\theta(x) = \psi_i^\theta(x, \phi^\theta(x))$,

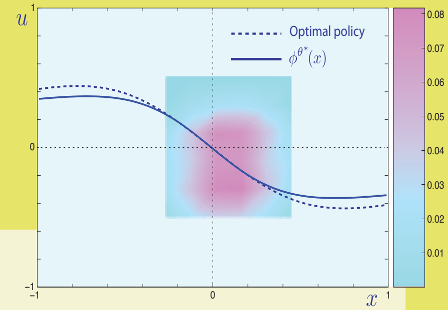
$1 \le i \le d$

$$\mathcal{D}_u \underline{H}^\theta\,(x) = \frac{d}{dt}\underline{H}^\theta(x(t))\Big|_{\substack{x=x(t)\\ u=u(t)}}$$

$$\mathcal{D}_u \underline{\psi}_i^\theta\,(x) = \frac{d}{dt}\underline{\psi}_i^\theta(x(t))\Big|_{\substack{x=x(t)\\ u=u(t)}}$$

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - Convex Reformulation



Step 3: Bellman error

$$\mathcal{L}^\theta(x,u) \quad := \quad \mathcal{D}_u \underline{H}^\theta(x) + \gamma(c - H^\theta), \qquad \theta \in \mathbb{R}^d$$

Based on observations, minimize the mean-square Bellman error:

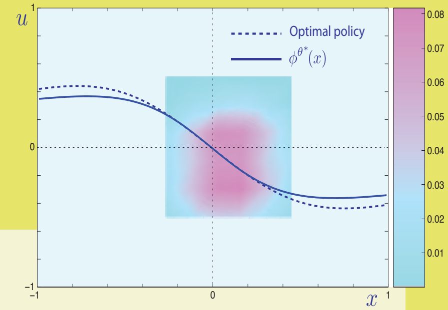
$$\mathcal{E}_{\mathrm{Bell}}(\theta) \quad := \quad \int [\mathcal{L}^\theta]^2 \, \varpi(dx, du) \; := \; \langle \mathcal{L}^\theta, \, \mathcal{L}^\theta \rangle_\varpi$$

$$\mathcal{L}^\theta(x,u) \quad := \quad \mathcal{D}_u G^\theta(x) + \gamma(c - H^\theta), \qquad \theta \in \mathbb{R}^d$$

$$G^\theta(x) \le H^\theta(x,u), \qquad \text{all } x, \, u$$

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - LQR example



Linear model and quadratic cost,

Cost: $$c(x,u) \;=\; \tfrac{1}{2}x^T Q x \;+\; \tfrac{1}{2}u^T R u$$

Q-function: $$H^*(x) = c(x,u) + (Ax + Bu)^T P^* x$$

Solves Riccati eqn

Q-function approx:

$$H^\theta(x,u) = c(x,u) + \frac{1}{2}\sum_{i=1}^{d_x} \theta_i^x x^T E^i x + \sum_{j=1}^{d_{xu}} \theta_j^x x^T F^i u$$

Approximation to minimum

$$G^\theta(x) = \tfrac{1}{2} x^T G^\theta x$$

Minimizer:

$$u^\theta(x) = \phi^\theta(x) = -R^{-1} F^\theta x$$

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

$$\cancel{\mathcal{D}_u \underline{H}^\theta(x) = \tfrac{d}{dt}\underline{H}^\theta(x(t))}$$

$$\cancel{\mathcal{D}_u \underline{\psi}_i^\theta(x) = \tfrac{d}{dt}\underline{\psi}_i^\theta(x(t))}$$

Q learning - Steps towards an algorithm

Step 4: Causal smoothing to avoid differentiation

For any function of two variables, $g : \mathbb{R}^\ell \times \mathbb{R}^{\ell_w} \to \mathbb{R}$
Resolvent gives a new function,

$$R_\beta g\,(x, w) = \int_0^\infty e^{-\beta t} g(x(t), \xi(t))\, dt$$

Skip to examples

Q learning - Steps towards an algorithm

$$\cancel{\begin{aligned}\mathcal{D}_u \underline{H}^\theta(x) &= \tfrac{d}{dt}\underline{H}^\theta(x(t)) \\ \mathcal{D}_u \underline{\psi}_i^\theta(x) &= \tfrac{d}{dt}\underline{\psi}_i^\theta(x(t))\end{aligned}}$$

Step 4: Causal smoothing to avoid differentiation

For any function of two variables, $g : \mathbb{R}^\ell \times \mathbb{R}^{\ell_w} \to \mathbb{R}$

Resolvent gives a new function,

$$R_\beta g\,(x, w) = \int_0^\infty e^{-\beta t} g(x(t), \xi(t))\, dt \;, \qquad \beta > 0$$

controlled using the nominal policy

$$u(t) = \phi(x(t), \xi(t)), \qquad t \geq 0$$

stabilizing & ergodic

Q learning - Steps towards an algorithm

$\cancel{\mathcal{D}_u \underline{H}^\theta(x) = \frac{d}{dt}\underline{H}^\theta(x(t))}$

$\cancel{\mathcal{D}_u \underline{\psi}_i^\theta(x) = \frac{d}{dt}\underline{\psi}_i^\theta(x(t))}$

Step 4: Causal smoothing to avoid differentiation

For any function of two variables, $g : \mathbb{R}^\ell \times \mathbb{R}^{\ell_w} \to \mathbb{R}$
Resolvent gives a new function,

$$R_\beta g\,(x,w) = \int_0^\infty e^{-\beta t} g(x(t),\xi(t))\,dt \quad , \qquad \beta > 0$$

Resolvent equation:

$$R_\beta \mathcal{D} = \beta R_\beta - I$$

$$\mathcal{D}_u \underline{H}^\theta(x) = \tfrac{d}{dt}\underline{H}^\theta(x(t))$$

$$\mathcal{D}_u \underline{\psi}_i^\theta(x) = \tfrac{d}{dt}\underline{\psi}_i^\theta(x(t))$$

Q learning - Steps towards an algorithm

Step 4: Causal smoothing to avoid differentiation

For any function of two variables, $g : \mathbb{R}^\ell \times \mathbb{R}^{\ell_w} \to \mathbb{R}$

Resolvent gives a new function,

$$R_\beta g\,(x,w) = \int_0^\infty e^{-\beta t} g(x(t),\xi(t))\,dt \ , \qquad \beta > 0$$

Resolvent equation:

$$R_\beta \mathcal{D} = \beta R_\beta - I$$

Smoothed Bellman error:

$$\begin{aligned} \mathcal{L}^{\theta,\beta} &= R_\beta \mathcal{L}^\theta \\ &= [\beta R_\beta - I]\underline{H}^\theta + \gamma R_\beta (c - H^\theta) \end{aligned}$$

$$\xcancel{\mathcal{D}_u \underline{H}^\theta(x) = \tfrac{d}{dt}\underline{H}^\theta(x(t))}$$

$$\xcancel{\mathcal{D}_u \underline{\psi}_i^\theta(x) = \tfrac{d}{dt}\underline{\psi}_i^\theta(x(t))}$$

Q learning - Steps towards an algorithm

Smoothed Bellman error:

$$\mathcal{E}_\beta(\theta) \;:=\; \tfrac{1}{2}\|\mathcal{L}^{\theta,\beta}\|_\varpi^2$$

$$\nabla\mathcal{E}_\beta(\theta) = \langle \mathcal{L}^{\theta,\beta}, \nabla_\theta \mathcal{L}^{\theta,\beta}\rangle_\varpi$$

$=$ *zero* at an optimum

Step 4: Causal smoothing to avoid differentiation

$$\cancel{\mathcal{D}_u \underline{H}^\theta(x) = \tfrac{d}{dt}\underline{H}^\theta(x(t))}$$

$$\cancel{\mathcal{D}_u \underline{\psi}_i^\theta(x) = \tfrac{d}{dt}\underline{\psi}_i^\theta(x(t))}$$

Q learning - Steps towards an algorithm

Smoothed Bellman error:

$$\mathcal{E}_\beta(\theta) \ := \ \tfrac{1}{2}\|\mathcal{L}^{\theta,\beta}\|_\varpi^2$$

$$\nabla \mathcal{E}_\beta(\theta) = \langle \mathcal{L}^{\theta,\beta}, \nabla_\theta \mathcal{L}^{\theta,\beta}\rangle_\varpi$$

$=$ *zero* at an optimum

Involves terms of the form $\langle R_\beta g, R_\beta h\rangle$

Step 4: Causal smoothing to avoid differentiation

Q learning - Steps towards an algorithm

$$\cancel{\begin{aligned}\mathcal{D}_u\underline{H}^\theta(x) &= \tfrac{d}{dt}\underline{H}^\theta(x(t)) \\ \mathcal{D}_u\underline{\psi}_i^\theta(x) &= \tfrac{d}{dt}\underline{\psi}_i^\theta(x(t))\end{aligned}}$$

Smoothed Bellman error: $\mathcal{E}_\beta(\theta) := \frac{1}{2}\|\mathcal{L}^{\theta,\beta}\|_\varpi^2$

$$\nabla\mathcal{E}_\beta(\theta) = \langle \mathcal{L}^{\theta,\beta}, \nabla_\theta \mathcal{L}^{\theta,\beta}\rangle_\varpi$$

Adjoint operation:

$$R_\beta^\dagger R_\beta = \frac{1}{2\beta}\left(R_\beta^\dagger + R_\beta\right)$$

$$\langle R_\beta g, R_\beta h\rangle = \frac{1}{2\beta}\left(\langle g, R_\beta^\dagger h\rangle + \langle h, R_\beta^\dagger g\rangle\right)$$

Step 4: Causal smoothing to avoid differentiation

Q learning - Steps towards an algorithm

$$\cancel{\mathcal{D}_u \underline{H}^\theta(x) = \tfrac{d}{dt}\underline{H}^\theta(x(t))}$$

$$\cancel{\mathcal{D}_u \underline{\psi}_i^\theta(x) = \tfrac{d}{dt}\underline{\psi}_i^\theta(x(t))}$$

Smoothed Bellman error: $\mathcal{E}_\beta(\theta) := \frac{1}{2}\|\mathcal{L}^{\theta,\beta}\|_\varpi^2$

$$\nabla \mathcal{E}_\beta(\theta) = \langle \mathcal{L}^{\theta,\beta}, \nabla_\theta \mathcal{L}^{\theta,\beta}\rangle_\varpi$$

Adjoint operation:

$$R_\beta^\dagger R_\beta = \frac{1}{2\beta}\,(R_\beta^\dagger + R_\beta)$$

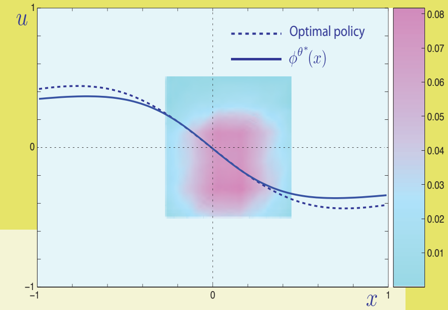
$$\langle R_\beta g, R_\beta h\rangle = \frac{1}{2\beta}\big(\langle g, R_\beta^\dagger h\rangle + \langle h, R_\beta^\dagger g\rangle\big)$$

Adjoint realization: *time-reversal*

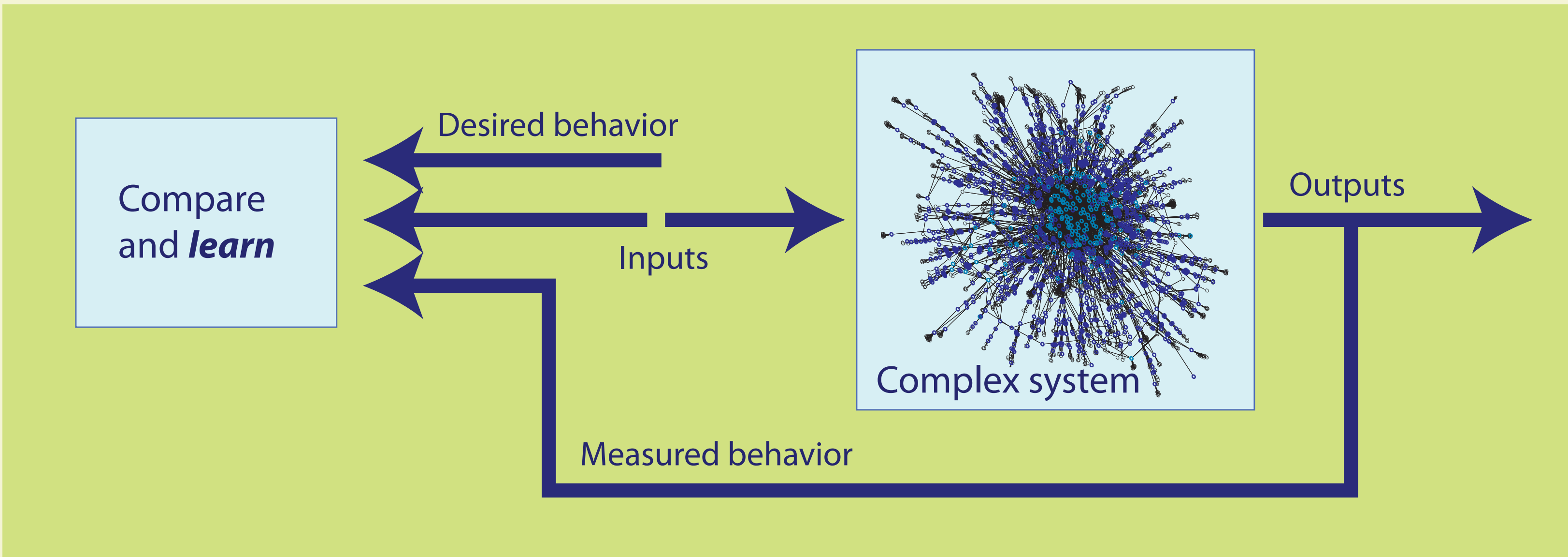
$$R_\beta^\dagger g\,(x,w) = \int_0^\infty e^{-\beta t}\mathsf{E}_{x,\,w}[g(x^\circ(-t),\xi^\circ(-t))]\,dt$$

expectation conditional on $x^\circ(0) = x,\ \xi^\circ(0) = w.$

Q learning - Steps towards an algorithm



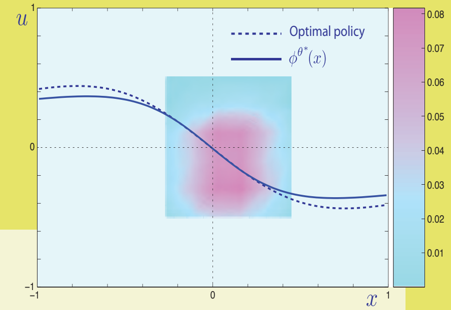
After Step 5: Not quite adaptive control:



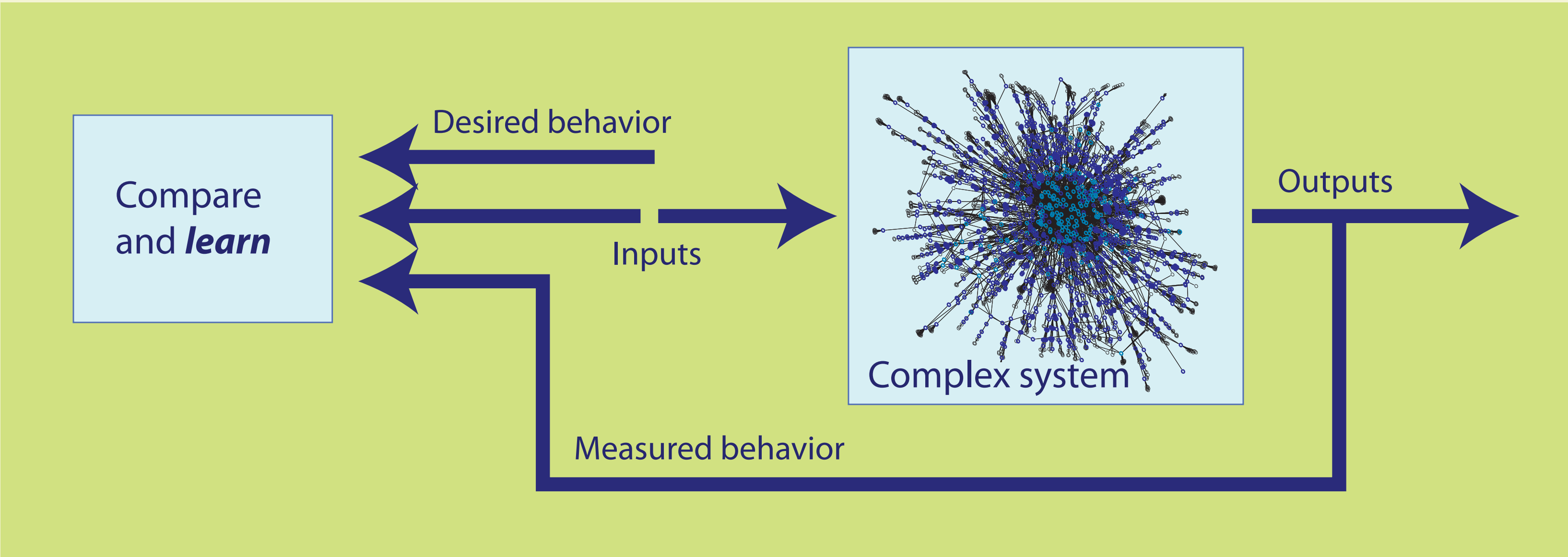
Ergodic input applied

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - Steps towards an algorithm



After Step 5: Not quite adaptive control:

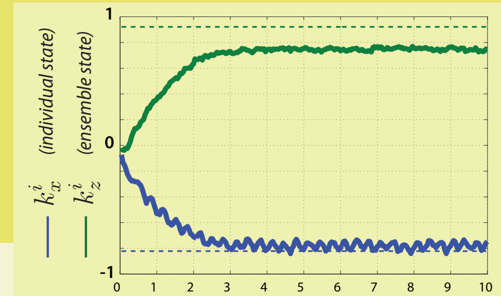


Ergodic input applied

Based on observations minimize the mean-square Bellman error:

$$\begin{aligned} \mathcal{E}_{\text{Bell}}(\theta) \quad &:= \quad \int \big[\mathcal{L}^{\theta}\big]^2 \varpi(dx, du) \\ \mathcal{L}^{\theta}(x,u) \quad &:= \quad \mathcal{D}_u \underline{H}^{\theta}(x) + \gamma(c - H^{\theta}), \qquad \theta \in \mathbb{R}^d \end{aligned}$$

Deterministic Stochastic Approximation



Gradient descent:

$$\frac{d}{dt}\theta = -\varepsilon\langle \mathcal{L}^\theta, \mathcal{D}_u \nabla_\theta \underline{H}^\theta - \gamma \nabla_\theta H^\theta \rangle_\varpi$$

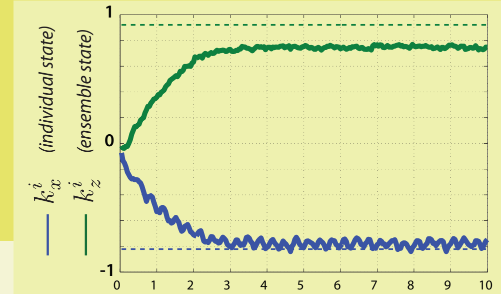
Converges* to the minimizer of the mean-square Bellman error:

$$\begin{aligned} \mathcal{E}_{\mathrm{Bell}}(\theta) \quad &:= \quad \int \big[\mathcal{L}^\theta\big]^2 \varpi(dx, du) \\ \mathcal{L}^\theta(x,u) \quad &:= \quad \mathcal{D}_u \underline{H}^\theta\,(x) + \gamma(c - H^\theta) \end{aligned}$$

$$\frac{d}{dt} h(x(t))\Big|_{\substack{x=x(t)\\ w=\xi(t)}} = \mathcal{D}_u h\,(x)$$

* Convergence observed in experiments!
For a convex re-formulation of
the problem, see Mehta & Meyn 2009

Deterministic Stochastic Approximation



Stochastic Approximation

$$\frac{d}{dt}\theta = -\varepsilon_t \mathcal{L}_t^\theta \Big(\frac{d}{dt}\nabla_\theta \underline{H}^\theta\,(x^\circ(t)) - \gamma \nabla_\theta H^\theta(x^\circ(t), u^\circ(t)) \Big)$$

$$\mathcal{L}_t^\theta := \frac{d}{dt}\underline{H}^\theta\,(x^\circ(t)) + \gamma\big(c(x^\circ(t), u^\circ(t)) - H^\theta(x^\circ(t), u^\circ(t))\big)$$

Gradient descent:

$$\frac{d}{dt}\theta = -\varepsilon \langle \mathcal{L}^\theta, \mathcal{D}_u \nabla_\theta \underline{H}^\theta - \gamma \nabla_\theta H^\theta \rangle_\varpi$$

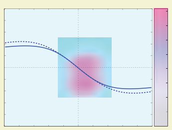
Mean-square Bellman error:

$$\begin{aligned} \mathcal{E}_{\mathrm{Bell}}(\theta) &:= \int [\mathcal{L}^\theta]^2\, \varpi(dx, du) \\ \mathcal{L}^\theta(x,u) &:= \mathcal{D}_u \underline{H}^\theta\,(x) + \gamma(c - H^\theta) \end{aligned}$$

$$\frac{d}{dt}h(x(t))\Big|_{\substack{x=x(t)\\ w=\xi(t)}} = \mathcal{D}_u h\,(x)$$

Outline

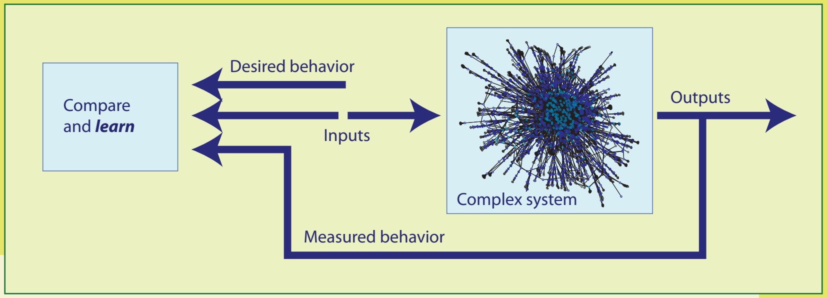
Q-learning for nonlinear state space models



Example: Local approximation

Example: Decentralized control

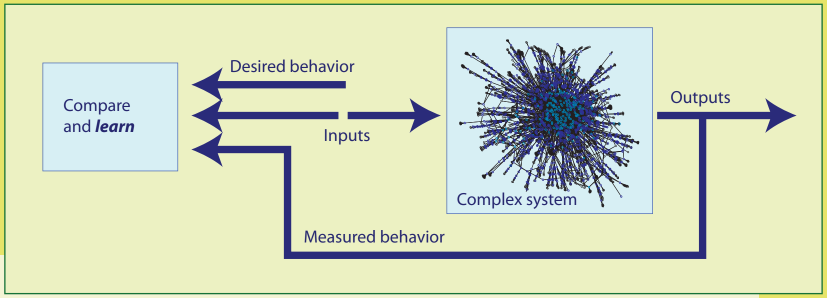
Q learning - Local Learning



Cubic nonlinearity:

$$\frac{d}{dt}x = -x^3 + u, \qquad c(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2$$

Q learning - Local Learning

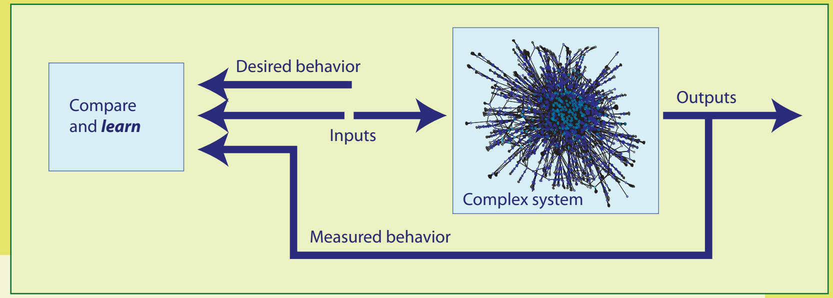


Cubic nonlinearity: $\frac{d}{dt}x = -x^3 + u, \qquad c(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2$

HJB:

$$\min_u \Big(\tfrac{1}{2}x^2 + \tfrac{1}{2}u^2 + (-x^3 + u)\nabla J^*(x) \Big) = \gamma J^*(x)$$

Q learning - Local Learning



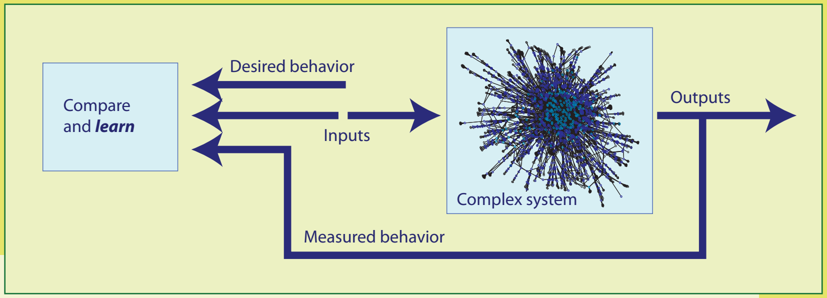
Cubic nonlinearity: $\frac{d}{dt}x = -x^3 + u, \qquad c(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2$

HJB: $\min_u \Big(\frac{1}{2}x^2 + \frac{1}{2}u^2 + (-x^3 + u)\nabla J^*(x) \Big) = \gamma J^*(x)$

Basis:

$$H^\theta(x,u) = c(x,u) + \theta^x x^2 + \theta^{xu} \frac{x}{1+2x^2} u$$

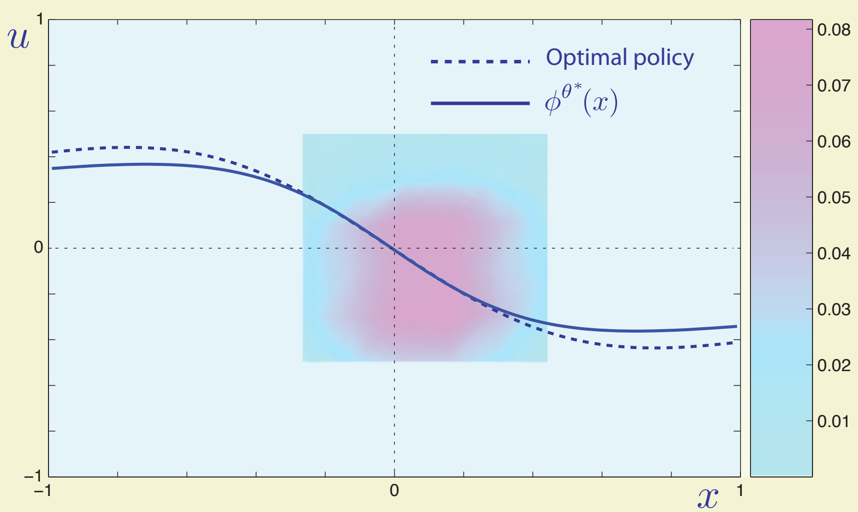
Q learning - Local Learning



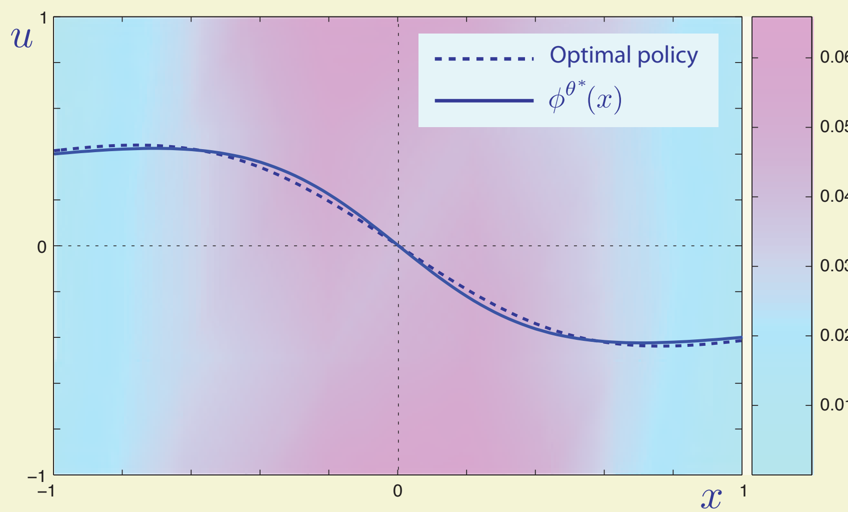
Cubic nonlinearity: $\frac{d}{dt}x = -x^3 + u, \qquad c(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2$

HJB: $\min_u \left(\frac{1}{2}x^2 + \frac{1}{2}u^2 + (-x^3 + u)\nabla J^*(x) \right) = \gamma J^*(x)$

Basis: $H^\theta(x,u) = c(x,u) + \theta^x x^2 + \theta^{xu} \frac{x}{1+2x^2} u$



Low amplitude input



High amplitude input

$$u(t) = A(\sin(t) + \sin(\pi t) + \sin(et))$$

Outline

Q-learning for nonlinear state space models

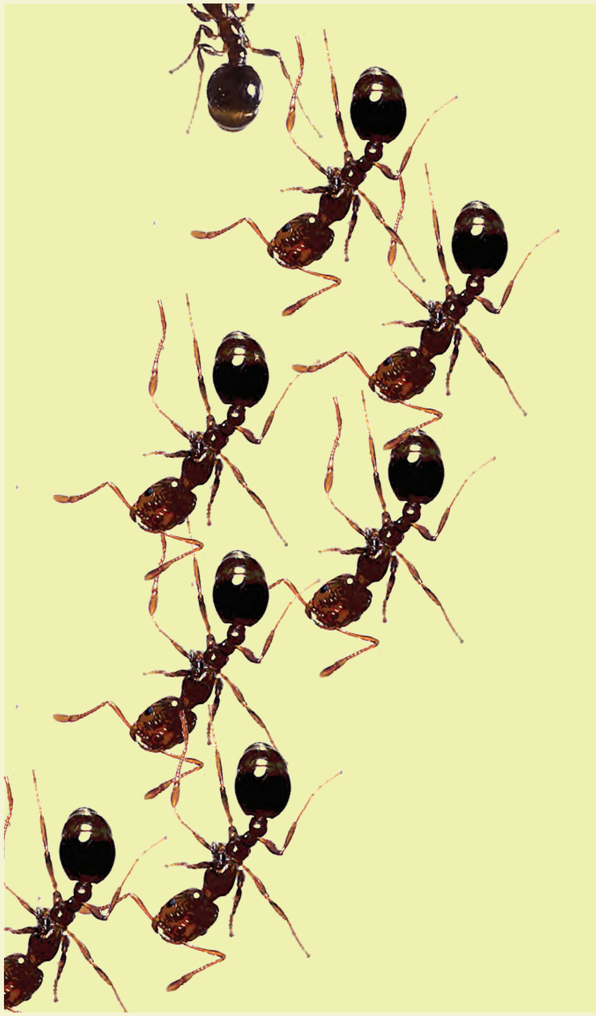
Example: Local approximation

Example: Decentralized control

Multi-agent model

M. Huang, P. E. Caines, and R. P. Malhame. Large-population cost-coupled LQG problems with nonuniform agents: Individual-mass behavior and decentralized ε-Nash equilibria. *IEEE Trans. Auto. Control*, 52(9):1560–1571, 2007.

Huang et. al. Local optimization for global coordination



Multi-agent model

Model: Linear autonomous models - global cost objective

HJB: Individual state + global average

Basis: Consistent with low dimensional LQG model

Results from five agent model:

Multi-agent model

Model: Linear autonomous models - global cost objective

HJB: Individual state + global average

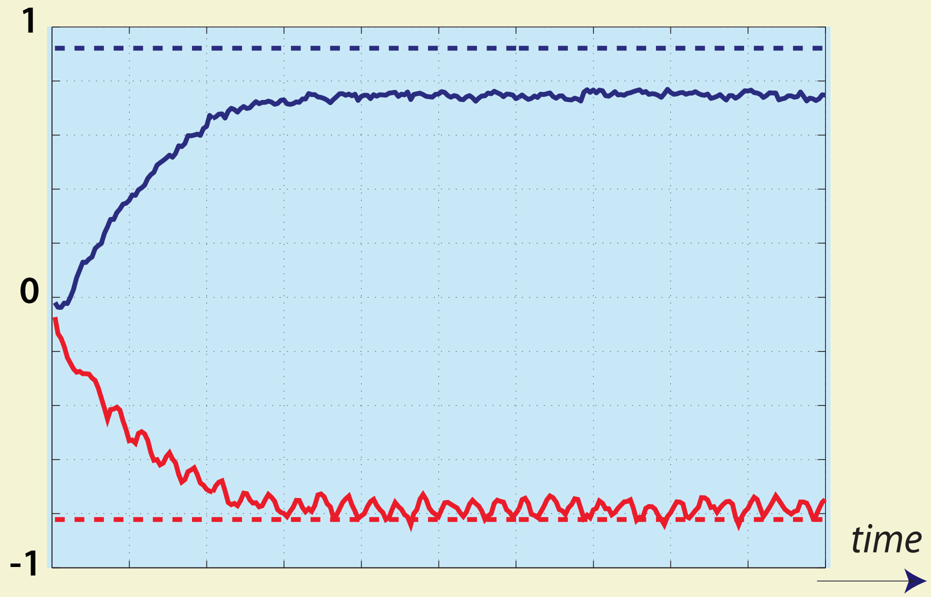
Basis: Consistent with low dimensional LQG model

Results from five agent model:

Estimated state feedback gains

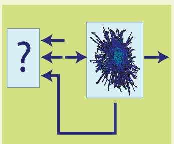
k_x^i *(individual state)*

k_z^i *(ensemble state)*



Gains for agent 4: Q-learning sample paths and gains predicted from ∞-agent limit

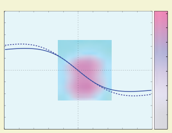
Outline



Coarse models - what to do with them?

Step 1: Recognize
Step 2: Find a stab...
Step 3: Optimality
Step 4: Adjoint
Step 5: Interpret

Q-learning for nonlinear state space models



Example: Local approximation

Example: Decentralized control

... Conclusions

Conclusions

Coarse models give tremendous insight

They are also tremendously useful
for design in approximate dynamic programming algorithms

Conclusions

Coarse models give tremendous insight

They are also tremendously useful
for design in approximate dynamic programming algorithms

Q-learning is as fundamental as the Riccati equation - this
should be included in our first-year graduate control courses

Conclusions

Coarse models give tremendous insight

They are also tremendously useful
for design in approximate dynamic programming algorithms

Q-learning is as fundamental as the Riccati equation - this
should be included in our first-year graduate control courses

Current research: Algorithm analysis and improvements
Applications in biology and economics
Analysis of game-theoretic issues
in coupled systems

References

- W. Chen, D. Huang, A. Kulkarni, J. Unnikrishnan, Q. Zhu, P. Mehta, S. Meyn, and A. Wierman. Approximate dynamic programming using fluid and diffusion approximations with applications to power management. Accepted for inclusion in the 48th IEEE Conference on Decision and Control, December 16-18 2009.

- P. Mehta and S. Meyn. Q-learning and Pontryagin's Minimum Principle. To appear in Proceedings of the 48th IEEE Conference on Decision and Control, December 16-18 2009.

- R.-R. Chen and S. P. Meyn. Value iteration and optimization of multiclass queueing networks. *Queueing Syst. Theory Appl.*, 32(1-3):65–97, 1999.

- S. G. Henderson, S. P. Meyn, and V. B. Tadić. Performance evaluation and policy selection in multiclass networks. *Discrete Event Dynamic Systems: Theory and Applications*, 13(1-2):149–189, 2003. Special issue on learning, optimization and decision making (invited).

- S. P. Meyn. The policy iteration algorithm for average reward Markov decision processes with general state space. *IEEE Trans. Automat. Control*, 42(12):1663–1680, 1997.

- S. P. Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, Cambridge, 2007.

- C. Moallemi, S. Kumar, and B. Van Roy. Approximate and data-driven dynamic programming for queueing networks. Preprint available at http://moallemi.com/ciamac/research-interests.php, 2008.

References

[1] D. H. Jacobson. Differential dynamic programming methods for determining optimal control of non-linear systems. PhD thesis, Univ. of London, 1967

[2] D. H. Jacobson and D. Q. Mayne. Differential dynamic programming. American Elsevier Pub. Co., New York, NY, 1970.

[3] C. J. C. H. Watkins. Learning from Delayed Rewards. PhD thesis, King's College, Cambridge, UK, 1989.

[4] C. J. C. H. Watkins and P. Dayan. Q-learning. Machine Learning, 8(3-4):279–292, 1992.

[5] J. N. Tsitsiklis and B. Van Roy. Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. IEEE Trans. Automat. Control, 44(10):1840–1851, 1999.

[6] V. S. Borkar and S. P. Meyn. The ODE method for convergence of stochastic approximation and reinforcement learning. SIAM J. Control Optim., 38(2):447–469, 2000.

[7] H. Yu and D. P. Bertsekas. Q-learning algorithms for optimal stopping based on least squares. In Proc. European Control Conference (ECC), July 2007.

[8] C. Moallemi, S. Kumar, and B. Van Roy. Approximate and data-driven dynamic programming for queueing networks. Preprint available at http://moallemi.com/ciamac/research-interests.php, 2008.

[9] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf. Brief paper: Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. Automatica, 43(3):473–481, 2007.

[10] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. Lewis. Adaptive optimal control for continuous-time linear systems based on policy iteration. Automatica, 45(2):477 – 484, 2009.

[11] S. P. Meyn. Control Techniques for Complex Networks. Cambridge University Press, Cambridge, 2007.

[12] S. G. Henderson, S. P. Meyn, and V. B. Tadi?. Performance evaluation and policy selection in multiclass networks. Discrete Event Dynamic Systems: Theory and Applications, 13(1-2):149–189, 2003. Special issue on learning, optimization and decision making (invited).

[13] W. Chen, D. Huang, A. Kulkarni, J. Unnikrishnan, Q. Zhu, P. Mehta, S. Meyn, and A. Wierman. Approximate dynamic programming using fluid and diffusion approximations with applications to power management. 48th IEEE Conference on Decision and Control, December 16-18 2009.