Simons Institute for the Theory of Computing

Theory of Reinforcement Learning
Aug. 19 – Dec. 18, 2020

Reinforcement Learning
Part 1: Control Systems & RL

Sean Meyn

Cognition & Control in Complex Systems

Department of Electrical and Computer Engineering — University of Florida

Inria International Chair — Inria, Paris

Thanks to to our sponsors: NSF and ARO

Besides NSF, Thanks to ...

Vivek Borkar

Adithya Devraj

Ana Bušić

Prashant Mehta

Eric Moulines

P.E. Caines
Max Huang
Chen & Chen Barooah
Colombino Bernstein
Priouret Dall'Anese
Raman Surana
Ruppert Polyak
Benveniste Metivier
Kushner Yin
Van Roy Tsitsiklis
Konda Bertsekas
Szepesvari Nedic Yu

Many Others

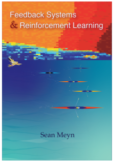
Part 1: Control Fundamentals

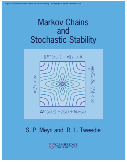
Outline

Resources

Videos from Simons RTDM, 2018

- Feedback Control Theory: Architectures and Tools for Real-Time Decision Making [essential prerequisite]
  https://simons.berkeley.edu/talks/murray-control-1
- Hidden Theory Part I (SA foundations)
  https://www.youtube.com/watch?v=dhEF5pfYmvc
- Hidden Theory Part II (Zap Q-learning)
  https://www.youtube.com/watch?v=Y3w8f1xIb6s





Lecture notes online: Feedback systems and reinforcement learning
simons.berkeley.edu/sites/default/files/docs/16101/monographrlsimonsinstitutebootcampseptember2020.pdf

[1] K. J. Astrom and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, USA, 2008 (recent edition on-line).

[25] D. Huang, W. Chen, P. Mehta, S. Meyn, and A. Surana. Feature selection for neuro-dynamic programming. In F. Lewis, editor, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley, 2011.

[26] A. M. Devraj, A. Bušić, and S. Meyn. Fundamental design principles for reinforcement learning algorithms. In *Handbook on Reinforcement Learning and Control*. Springer, 2020

Apologies

- π will always be an invariant measure 

  ϕ and $\widetilde{\phi}$ will be policies (feedback)

Apologies

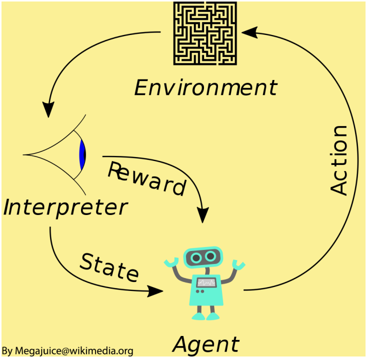
- π will always be an invariant measure 

  ϕ and $\widetilde{\phi}$ will be policies (feedback)
- Control engineers minimize cost $c(x, u)$ (rarely receive rewards)
  x state, u input (action)

Apologies

- π will always be an invariant measure 

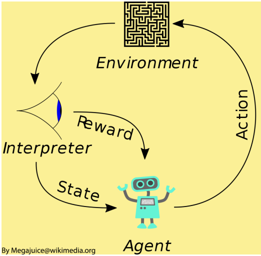
  ϕ and $\widetilde{\phi}$ will be policies (feedback)
- Control engineers minimize cost $c(x,u)$ (rarely receive rewards)
  x state, u input (action)
- I don't mean to offend!
  If I seem critical, it is simply my opinion, and
  - I don't have all the answers
  - My opinion may be stupid


Environment
Interpreter
Reward
State
Action
Agent
By Megajuice@wikimedia.org

Background

Reinforcement Learning

Intelligent actors optimize through interactions with their environment



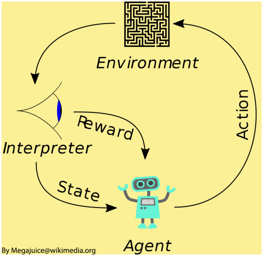
Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward –Wikipedia

Comparison of reinforcement learning algorithms [edit]

Algorithm	Description	Model	Policy	Action Space	State Space	Operator
Monte Carlo	Every visit to Monte Carlo	Model-Free	Off-policy	Discrete	Discrete	Sample-means
Q-learning	State–action–reward–state	Model-Free	Off-policy	Discrete	Discrete	Q-value
SARSA	State–action–reward–state–action	Model-Free	On-policy	Discrete	Discrete	Q-value
Q-learning - Lambda	State–action–reward–state with eligibility traces	Model-Free	Off-policy	Discrete	Discrete	Q-value
SARSA - Lambda	State–action–reward–state–action with eligibility traces	Model-Free	On-policy	Discrete	Discrete	Q-value
DQN	Deep Q Network	Model-Free	Off-policy	Discrete	Continuous	Q-value
DDPG	Deep Deterministic Policy Gradient	Model-Free	Off-policy	Continuous	Continuous	Q-value
A3C	Asynchronous Advantage Actor-Critic Algorithm	Model-Free	On-policy	Continuous	Continuous	Advantage
NAF	Q-Learning with Normalized Advantage Functions	Model-Free	Off-policy	Continuous	Continuous	Advantage
TRPO	Trust Region Policy Optimization	Model-Free	On-policy	Continuous	Continuous	Advantage
PPO	Proximal Policy Optimization	Model-Free	On-policy	Continuous	Continuous	Advantage
TD3	Twin Delayed Deep Deterministic Policy Gradient	Model-Free	Off-policy	Continuous	Continuous	Q-value
SAC	Soft Actor-Critic	Model-Free	Off-policy	Continuous	Continuous	Advantage

Reinforcement Learning

Intelligent actors optimize through interactions with their environment



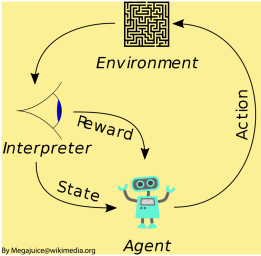
Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward –Wikipedia

Examples:

- Stock trading
- Autonomous cars
- Smart Buildings and Smart Grids

Reinforcement Learning

Intelligent actors optimize through interactions with their environment?



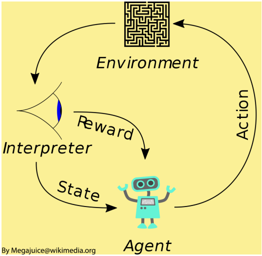
Examples:

- Stock trading
- Autonomous cars
- Smart Buildings and Smart Grids

What are we talking about?

Reinforcement Learning

Intelligent actors optimize through interactions with their environment?



Examples:

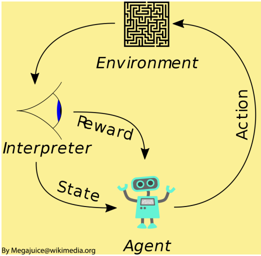
- Stock trading
- Autonomous cars
- Smart Buildings and Smart Grids

What are we talking about?

Learn as we
Trade stocks?
Drive cars?
Manage the grid?

Reinforcement Learning

Intelligent actors optimize through interactions with their environment?

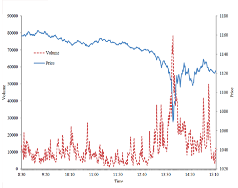


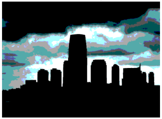
Examples:

- Stock trading
- Autonomous cars
- Smart Buildings and Smart Grids

Learn as we
Trade stocks?
Drive cars? *model free!*
Manage the grid?

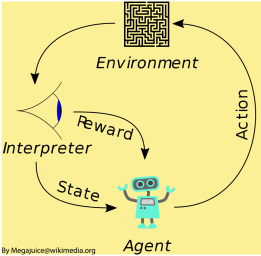
What are we talking about?





Reinforcement Learning

Intelligent actors optimize through interactions with their environment?

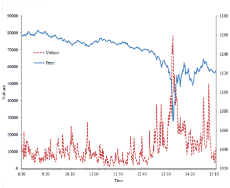


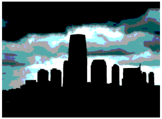
Learn as we
Trade stocks?
Drive cars? *model free!*
Manage the grid?

Examples:

- Stock trading
- Autonomous cars
- Smart Buildings and Smart Grids

What are we talking about?





RL is an emerging science, evolving alongside decision and control theory

Reinforcement Learning

Dreams of Model Free Control *well before my graduate student days*

"Typical" Adaptive Control System: MIT Rule

(NASA Report by Staff engineers at Edwards AFB, Nov, 1970)

Early Dreams
of
Model Free Control

From a tutorial by Erik Ydstie, CMU

Conclusions after 65 flight tests:

- Nearly invariant response at essentially all conditions
- accurate a priori knowledge of aerodynamic characteristics not needed (model-free)
- aircraft configuration changes compensated for
- redundancy (dual) provided a reliable and fail safe system

Pilot Observations

The true superiority of the X-15 AFCS was that it unburdened the pilot. The airplane was stable at any dynamic pressure and at any angle of attack. The AFCS inspired confidence and allowed the pilot to spend time cross-checking flight instruments, checking subsystems, and "sightseeing."

RL is an emerging science, evolving alongside decision and control theory

Reinforcement Learning

Dreams of Model Free Control *well before my graduate student days*

The flight-test program also disclosed several disadvantages associated with the system, including the following: (1) Commands by the pilot and other spurious inputs caused gain reduction and degraded performance at undesirable times; and (2) supercritical gain operation existed in flight, which, because of mechanical nonlinearities and electrical saturation, resulted in divergent airplane motions.

Early Dreams
of
Model Free Control

Conclusions after 65 flight tests:
- Nearly invariant response at essentially all conditions
- accurate a priori knowledge of aerodynamic characteristics not needed
- aircraft configuration changes compensated for
- redundancy (dual) provided a reliable and fail safe system

Problems: Gain changes due to disturbance inputs
Parameter drift and bursting
Lack of robustness in the presence of constraints (wind-up)

RL is an emerging science, evolving alongside decision and control theory

Reinforcement Learning

Dreams of Model Free Control *well before my graduate student days*

The flight-test program also disclosed several disadvantages associated with the system, including the following: (1) Commands by the pilot and other spurious inputs caused gain reduction and degraded performance at undesirable times; and (2) supercritical gain operation existed in flight, which, because of mechanical nonlinearities and electrical saturation, resulted in divergent airplane motions.

Problems: Gain changes due to disturbance inputs
Parameter drift and bursting
Lack of robustness in the presence of constraints (wind-up)

https://www.nasa.gov/centers/armstrong/news/FactSheets/FS-052-DFRC.html

RL is an emerging science, evolving alongside decision and control theory

What Control Can Offer

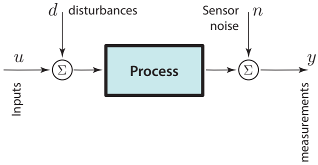
Control theory: goals and architectures

Murray tutorial, 2018

Process: car, plane, pancreas
bike sharing
wall street, semi-conductor manufacturing
power grid, transportation network

Inputs: throttle, wheel position, insulin rate,
truck dispatch, commands to generators and batteries

Measurements: speed and position, insulin, glucose, blood pressure,
camera and driver reports, frequency, phase, voltage



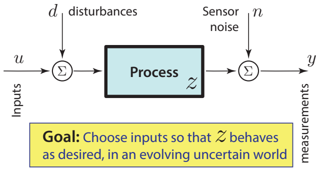
Control theory: goals and architectures

Murray tutorial, 2018

Process: car, plane, pancreas
bike sharing
wall street, semi-conductor manufacturing
power grid, transportation network

Inputs: throttle, wheel position, insulin rate,
truck dispatch, commands to generators and batteries

Measurements: speed and position, insulin, glucose, blood pressure,
camera and driver reports, frequency, phase, voltage



Control System Specifications

Simons Institute, 24 Jan 2018

Richard M. Murray, Caltech CDS

Transient: initial response to input

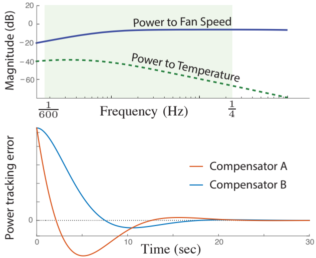
- Step response: rise time, overshoot, settling time, etc

Steady state: response after the transients have died out

- Frequency response: magnitude and phase for sinusoids

Safety: constraints that the system should never violate

Liveness: conditions that system should satisfy repeatedly



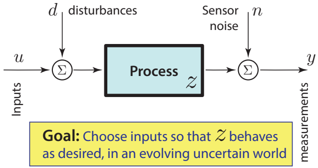
Control theory: goals and architectures

Murray tutorial, 2018

Process: car, plane, pancreas
bike sharing
wall street, semi-conductor manufacturing
power grid, transportation network

Inputs: throttle, wheel position, insulin rate,
truck dispatch, commands to generators and batteries

Measurements: speed and position, insulin, glucose, blood pressure,
camera and driver reports, frequency, phase, voltage

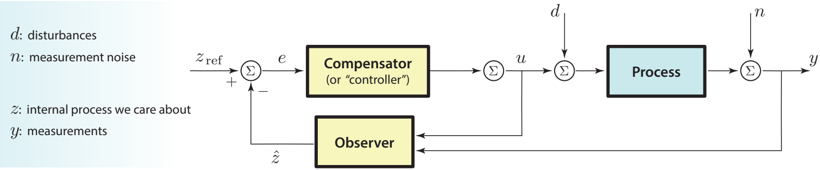


Strategies: Open loop control, assuming perfect model $z = G_{zu}u$
Invert dynamics:

$$u = G_{zu}^{-1} z_{\text{des}}$$

Control theory: goals and architectures

Murray tutorial, 2018



Strategies: Open loop control, assuming perfect model $z = G_{zu}u$

Invert dynamics:

$$u = G_{zu}^{-1} z_{\text{des}}$$

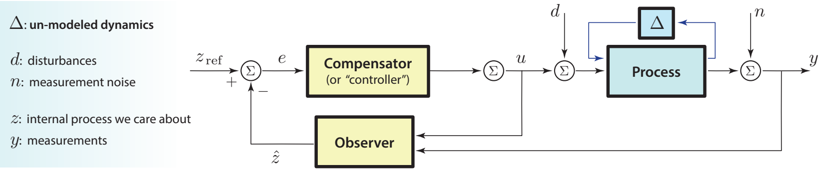
Classical control: Choose $u = H\hat{z}_{\text{des}} + G_c y$ so that

$$u \approx G_{zu}^{-1} z_{\text{des}}$$

Murray [Simons, 2018] called this *purely reactive*

Control theory: goals and architectures

Murray tutorial, 2018



Strategies: Classical control: Choose $u = H\hat{z}_{\text{des}} + G_c y$ so that

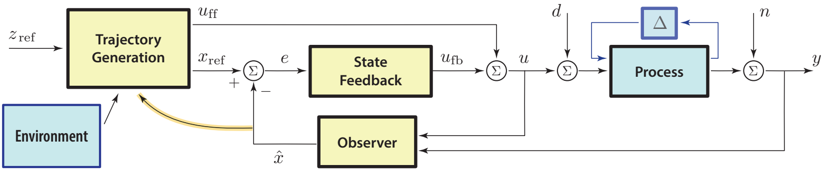
$$u \approx G_{zu}^{-1} z_{\text{des}}$$

Typical control education: design H, G_c and observer so that desired specifications are met, based on

- Process model
- d, n, Δ in some bounded class

Control theory: goals and architectures

Murray tutorial, 2018

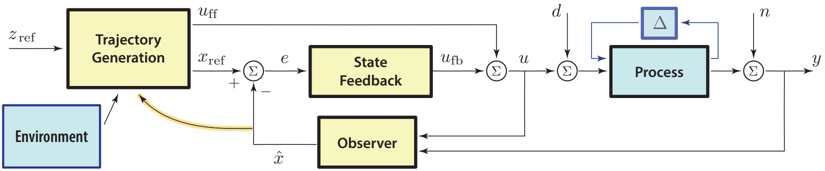


Strategies: Classical control: $u = H\hat{z}_{\text{des}} + G_c y$

Consider the steps taken when you plan to drive across town.
There is a *reactive component*. What else?

Control theory: goals and architectures

Murray tutorial, 2018



Consider the steps taken when you plan to drive across town.
There is a *reactive component*. What else?

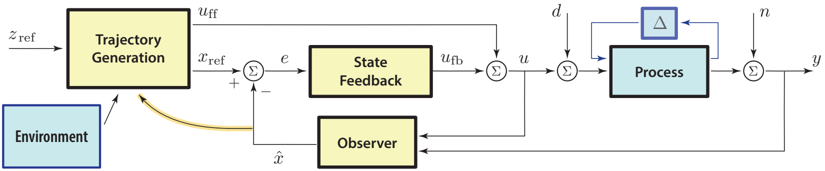
Yellow boxes may be built around optimization (MPC/RHC):

$$J^{\star}(x) = \min_{\boldsymbol{u}} \left\{ \int_0^T c(x_t, u_t)\, dt + V_0(X_T) \right\}$$

Many optimization problems to solve, because there are many objectives

Control theory: goals and architectures

Murray tutorial, 2018



Just as in RL, the definition of *state* depends on goals and observations

Consider the steps taken when you plan to drive across town.
There is a *reactive component*. What else?

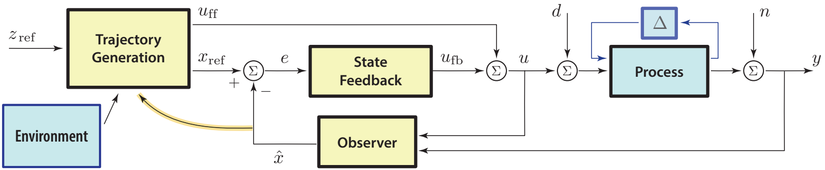
Yellow boxes may be built around optimization (MPC/RHC):

$$J^\star(x) = \min_{\boldsymbol{u}} \Big\{ \int_0^T c(x_t, u_t)\, dt + V_0(X_T) \Big\}$$

Many optimization problems to solve, because there are many objectives

Control theory: goals and architectures

Murray tutorial, 2018



Approximating $J^\star$ and/or $\boldsymbol{u}^\star$ can be addressed using RL

Just as in RL, the definition of *state* depends on goals and observations

Consider the steps taken when you plan to drive across town.
There is a *reactive component*. What else?

Yellow boxes may be built around optimization (MPC/RHC):

$$J^\star(x) = \min_{\boldsymbol{u}} \Big\{ \int_0^T c(x_t, u_t)\, dt + V_0(X_T) \Big\}$$

Many optimization problems to solve, because there are many objectives

An Incomplete History of Adaptive Control

Adaptation

Dynamic process by which the controller adjusts its interaction with a system in order to carry out an objective (or reach a goal) w/o exact knowledge of the system.

Adaptive Control

Adaptation: dynamic process by which the controller adjusts its interaction with a system in order to carry out an objective (or reach a goal) w/o exact knowledge of the system.

Some (incomplete) history:

1950s: gain scheduling
early Model Reference Adaptive Control (MRAC)

1958: R. Kalman — self-tuning controller (regulator) for the linear quadratic problem.

1960s: stability of adaptive controllers
Lyapunov stability
adaptation = learning (Feldbaum, Tsypkin)

1966: Parks — Lyapunov redesign approach to MRAC

1970s: stability analysis (Narendra, Morse, ...)

1980s: limitations (Rohrs et al.: sensitivity to unmodeled dynamics)

1983: Morse's conjecture $\dot{x} = ax + bu$, $a \in \mathbb{R}$, $b \neq 0$

cannot stabilize w/o knowledge of sign(b)

1983: Nussbaum disproves Morse's conjecture
1984: Willems - Byrnes: simplified Nussbaum's construction
exploration vs. exploitation

An Incomplete History of Adaptive Control

Adaptation

Dynamic process by which the controller adjusts its interaction with a system in order to carry out an objective (or reach a goal) w/o exact* knowledge of the system.

*For example: MDP or linear system with bound on dimension

Assumed for *analysis* and not implementation

Adaptive control and RL have nearly identical roots and goals

More history to be found in Lecture 4

Adaptive Control

Adaptation: dynamic process by which the controller adjusts its interaction with a system in order to carry out an objective (or reach a goal) w/o exact knowledge of the system.

Some (incomplete) history:

1950s: gain scheduling
early Model Reference Adaptive Control (MRAC)

1958: R. Kalman — self-tuning controller (regulator) for the linear quadratic problem.

1960s: stability of adaptive controllers
Lyapunov stability
adaptation = learning (Feldbaum, Tsypkin)

1966: Parks — Lyapunov redesign approach to MRAC

1970s: stability analysis (Narendra, Morse, ...)

1980s: limitations (Rohrs et al.: sensitivity to unmodeled dynamics)

1983: Morse's conjecture $\dot{x} = ax + bu$, $a \in \mathbb{R}$, $b \neq 0$

cannot stabilize w/o knowledge of sign(b)

1983: Nussbaum disproves Morse's conjecture
1984: Willems-Byrnes: simplified Nussbaum's construction
exploration vs. exploitation

An Incomplete History of Adaptive Control

Adaptive control and RL have nearly identical roots and goals

Common analytical tool: **ODE Method**
[stochastic approximation of Robbins & Monro]

- [Tsitsiklis, 1994] and [Jaakola, Jordan, and Singh, 1994] [14, 15]

Asynchronous Stochastic Approximation and Q-Learning

JOHN N. TSITSIKLIS jnt@athena.mit.edu
Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139

Editor: Richard Sutton

Abstract. We provide some general results on the convergence of a class of stochastic approximation algorithms and their parallel and asynchronous variants. We then use these results to study the Q-learning algorithm, a reinforcement learning method for solving Markov decision problems, and establish its convergence under conditions more general than previously available.

Keywords: Reinforcement learning, Q-learning, dynamic programming, stochastic approximation

- [Wittenmark, 1975], [Ljung, 1977] [7, 8]

Analysis of Recursive Stochastic Algorithms

LENNART LJUNG, MEMBER, IEEE

II. The Algorithm

A general recursive algorithm can be written

$$x(t) = x(t-1) + \gamma(t) Q\big(t; x(t-1), \varphi(t)\big), \qquad (1)$$

Adaptive Control

Adaptation: dynamic process by which the controller adjusts its interaction with a system in order to carry out an objective (or reach a goal) w/o exact knowledge of the system.

Some (incomplete) history:

1950s: gain scheduling
early Model Reference Adaptive Control (MRAC)

1958: R. Kalman — self-tuning controller (regulator) for the linear quadratic problem.

1960s: stability of adaptive controllers
Lyapunov stability
adaptation = learning (Feldbaum, Tsypkin)

1966: Parks — Lyapunov redesign approach to MRAC

1970s: stability analysis (Narendra, Morse, ...)

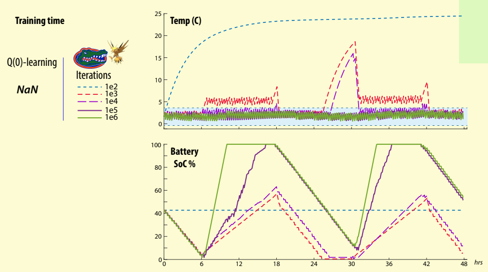
1980s: limitations (Rohrs et al.: sensitivity to unmodeled dynamics)

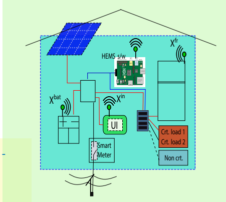
1983: Morse's conjecture $\dot{x} = ax + bu$
$a \in \mathbb{R}$
$b \neq 0$

cannot stabilize w/o knowledge of sign(b)

1983: Nussbaum disproves Morse's conjecture
1984: Willems - Byrnes: simplified Nussbaum's construction
exploration vs. exploitation

See Liberzon's lecture notes: `http://liberzon.csl.illinois.edu/teaching/16ece517notes.pdf`
Recent survey: [Matni et al, 2019] From self-tuning regulators to reinforcement learning and back again.





Optimal Control and RL

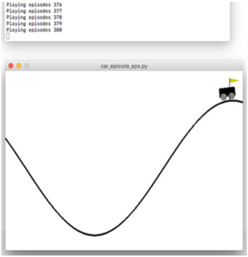
Dynamic Programming and RL

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Example from `gym.openai.com`: get up the hill efficiently

State: X_k denotes position and velocity (why?)

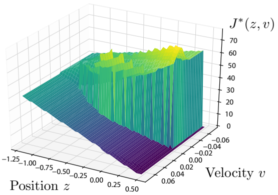
Input (or *action*): U_k is force



Dynamic Programming and RL

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Example from `gym.openai.com`: get up the hill efficiently



State: X_k denotes position and velocity
Input (or *action*): U_k is force

Value function: $$J^\star(x) = \min_{\textit{actions}} \sum_{k=0}^{\tau} c(X_k, U_k)$$

c : cost function

τ : time to reach the hill top

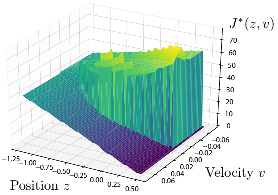
Similar to a control favorite: *Swinging up a pendulum by energy control* [2]

Dynamic Programming and RL

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Example from `gym.openai.com`: get up the hill efficiently

State: X_k denotes position and velocity
Input (or *action*): U_k is force



Value function: $$J^\star(x) = \min_{\textit{actions}} \sum_{k=0}^{\tau} c(X_k, U_k)$$

c : cost function

τ : time to reach the hill top

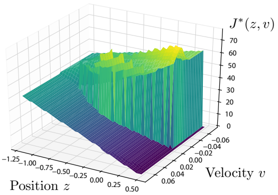
DP eqn: $$J^\star(X_0) = \min_{U_0} \{c(X_0, U_0) + J^\star(X_1)\}$$

Recall Bellman or Bellman-Ford

Dynamic Programming and RL

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Example from `gym.openai.com`: get up the hill efficiently



State: X_k denotes position and velocity

Input (or *action*): U_k is force

Value function: $$J^\star(x) = \min_{\textit{actions}} \sum_{k=0}^{\tau} c(X_k, U_k)$$

c : cost function

τ : time to reach the hill top

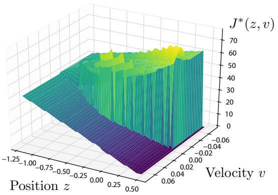
DP eqn: $$J^\star(X_0) = \min_{U_0}\{\underbrace{c(X_0,U_0) + J^\star(X_1)}_{Q^\star(X_0,U_0)}\}$$

Q-learning is all about approximating $Q^\star$

Dynamic Programming and RL

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Example from `gym.openai.com`: get up the hill efficiently



State: X_k denotes position and velocity

Input (or *action*): U_k is force

Value function: $$J^\star(x) = \min_{\textit{actions}} \sum_{k=0}^{\tau} c(X_k, U_k)$$

c : cost function

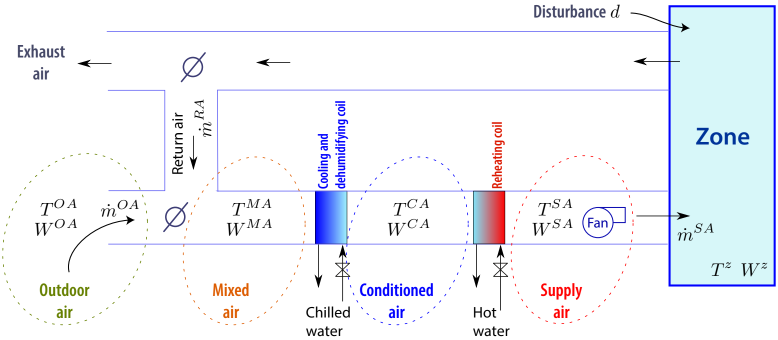
τ : time to reach the hill top

DP eqn: $$J^\star(X_0) = \min_{U_0} \{ \underbrace{c(X_0, U_0) + J^\star(X_1)}_{Q^\star(X_0, U_0)} \}$$

Q-learning is all about approximating $Q^\star$

If we know $Q^\star$, we obtain $U_k^* = \phi^*(X_k)$

Control Design for Heating and Ventilation



Eight dimensional state space and four dimensional input space

Joint work with N. S. Raman, P. Barooah @ UF MAE,   A. Devraj @ Stanford

See final page of references, and bibliography of [94]

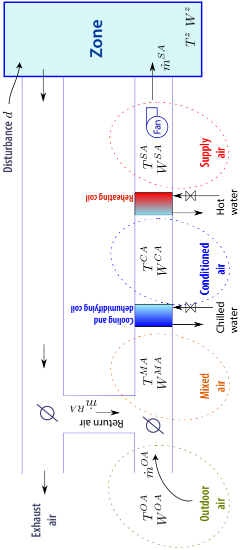
Control Design for Heating and Ventilation

Input: $U_k \stackrel{\text{def}}{=} [m_{sa}(k), r_{oa}(k), T_{ca}(k), T_{sa}(k)]^T$

1. Supply air flow rate (m_{sa})
2. Outdoor air ratio (r_{oa})
3. Conditioned air temperature (T_{ca})
4. Supply air temperature (T_{sa})

State: $X_k \stackrel{\text{def}}{=} [T_z(k), W_z(k), T_{oa}(k), W_{oa}(k), U(k-1)]^T$

1. Zone air temperature (T_z)
2. Zone air humidity ratio (W_z)
3. Outdoor air temperature (T_{oa})
4. Outdoor air humidity ratio (W_{oa})
5. Control inputs from the previous time step
6. ... forecast of occupancy, weather, ... Exercise: make a list of useful data



Control Design for Heating and Ventilation

Input: $U_k \stackrel{\text{def}}{=} [m_{sa}(k), r_{oa}(k), T_{ca}(k), T_{sa}(k)]^T$

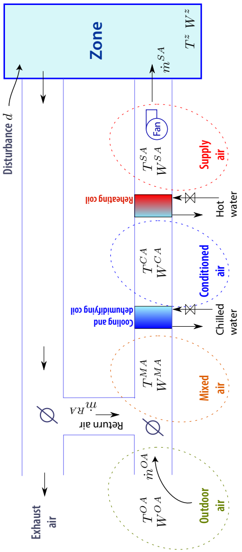
State:
$X_k \stackrel{\text{def}}{=} [T_z(k), W_z(k), T_{oa}(k), W_{oa}(k), U(k-1)]^T$

Quadratic basis: $+$ Zap Q-learning

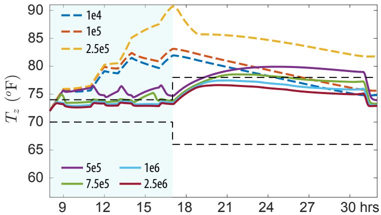
$$Q^\theta(x,u) = (x,u)^\intercal M_\theta (x,u) + (x,u)^\intercal L_\theta + k_\theta$$

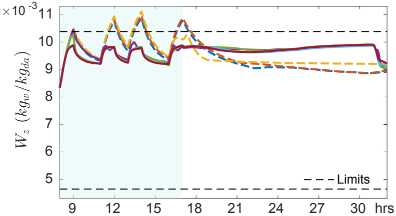
$$= \sum_i \theta_i \psi_i(x,u)$$

Initial results are great ...



Close Loop Response: Temperature and humidity evolution

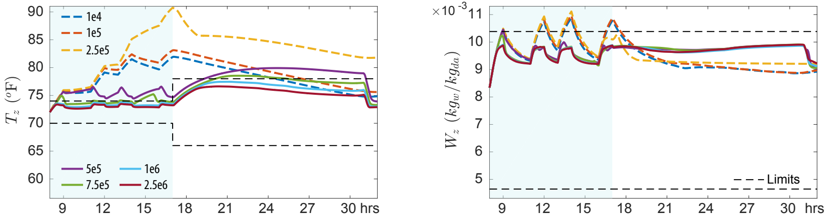




- Goal: Maintain temperature / humidity, *and* minimize energy consumption
- Inputs: Air-flow rate, out-door air ratio, conditioned air temperature, supply air temperature
- Approach: Find θ^* with quadratic basis:

$$Q^\theta(x,u) = (x,u)^T M_\theta (x,u) + (x,u)^T L_\theta + k_\theta$$

Close Loop Response: Temperature and humidity evolution

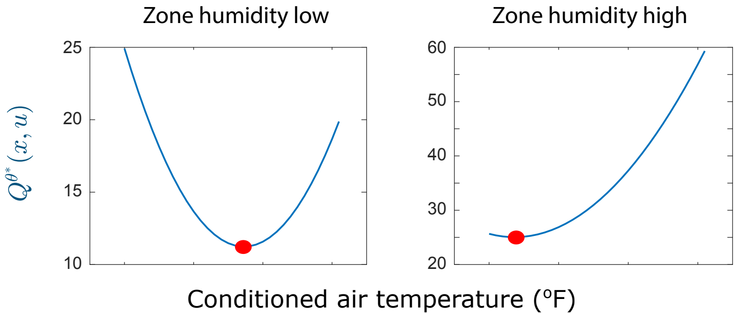


- Goal: Maintain temperature / humidity, *and* minimize energy consumption
- Inputs: Air-flow rate, out-door air ratio, conditioned air temperature, supply air temperature
- Approach: Find θ^* with quadratic basis:

$$Q^\theta(x,u) = (x,u)^\intercal M_\theta(x,u) + (x,u)^\intercal L_\theta + k_\theta$$

Once we know θ^*, we define $U_k = \phi^{\theta^*}(X_k) = \arg\min_u Q^{\theta^*}(X_k, u)$

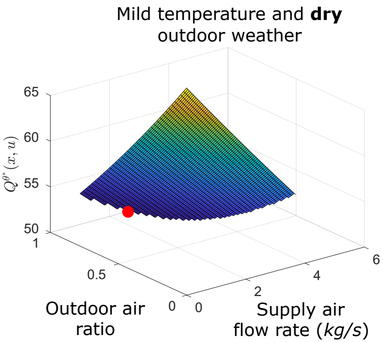
Algorithm learns: Cooling reduces humidity



Q-learning solution: zone is humid $\Longrightarrow$ conditioned air temperature reduced

Once we know θ^*, we define $U_k = \phi^{\theta^*}(X_k)$

Algorithm learns: Humid air can be expensive



Q-learning solution: humid exterior $\Longrightarrow$ outdoor air in-flow rate reduced

Once we know θ^*, we define $U_k = \phi^{\theta^*}(X_k)$

From DP to Q-learning

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Value function: $$J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} c(X_k, U_k)\,, \quad X_0 = x \in \mathsf{X}$$

DP eqn: $$J^\star(X_0) = \min_{U_0} \{ \underbrace{c(X_0, U_0) + J^\star(X_1)}_{Q^\star(X_0, U_0)} \}$$

From DP to Q-learning

$X_{k+1} = \mathrm{F}(X_k, U_k) \qquad J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} c(X_k, U_k)$

DP eqn: $J^\star(X_0) = \min_{U_0} \{ \underbrace{c(X_0, U_0) + J^\star(X_1)}_{Q^\star(X_0, U_0)} \}$

Magic: Denote $\underline{Q}^\star(x) = \min_u Q^\star(x, u) = J^\star(x)$

$\Longrightarrow$ Fixed point equation for Q-function

$$Q^\star(x, u) = c(x, u) + J^\star(\mathrm{F}(x, u))$$

From DP to Q-learning

$X_{k+1} = \mathrm{F}(X_k, U_k) \qquad J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} c(X_k, U_k)$

DP eqn: $\quad J^\star(X_0) = \min_{U_0} \{ \underbrace{c(X_0, U_0) + J^\star(X_1)}_{Q^\star(X_0, U_0)} \}$

Magic: Denote $\underline{Q}^\star(x) = \min_u Q^\star(x, u) = J^\star(x)$

$\Longrightarrow$ Fixed point equation for Q-function

$$Q^\star(x, u) = c(x, u) + J^\star(\mathrm{F}(x, u)) = c(x, u) + \underline{Q}^\star(\mathrm{F}(x, u))$$

From DP to Q-learning

$X_{k+1} = \mathrm{F}(X_k, U_k) \qquad J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} c(X_k, U_k)$

DP eqn: $\displaystyle J^\star(X_0) = \min_{U_0}\{\underbrace{c(X_0, U_0) + J^\star(X_1)}_{Q^\star(X_0,U_0)}\}$

Magic: Denote $\underline{Q}^\star(x) = \min_u Q^\star(x,u) = J^\star(x)$

$\Longrightarrow$ Fixed point equation for Q-function

$$Q^\star(x,u) = c(x,u) + J^\star(\mathrm{F}(x,u)) = c(x,u) + \underline{Q}^\star(\mathrm{F}(x,u))$$

Choose approximation among $\{Q^\theta(x,u) : \theta \in \mathbb{R}^d\}$

Bellman error: $\mathcal{E}^\theta(x,u) = -Q^\theta(x,u) + c(x,u) + \underline{Q}^\theta(\mathrm{F}(x,u))$

For example, θ_i is a "weight" in a neural network or

$$Q^\theta(x,u) = \sum_i \theta_i \psi_i(x,u)$$

From DP to Q-learning

$X_{k+1} = \mathrm{F}(X_k, U_k) \qquad J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} c(X_k, U_k)$

DP eqn: $$J^\star(X_0) = \min_{U_0}\{\underbrace{c(X_0,U_0) + J^\star(X_1)}_{Q^\star(X_0,U_0)}\}$$

Magic: Denote $\underline{Q}^\star(x) = \min_u Q^\star(x,u) = J^\star(x)$

$\Longrightarrow$ Fixed point equation for Q-function

$$Q^\star(x,u) = c(x,u) + J^\star(\mathrm{F}(x,u)) = c(x,u) + \underline{Q}^\star(\mathrm{F}(x,u))$$

Choose approximation among $\{Q^\theta(x,u) : \theta \in \mathbb{R}^d\}$

Bellman error: $\mathcal{E}^\theta(x,u) = -Q^\theta(x,u) + c(x,u) + \underline{Q}^\theta(\mathrm{F}(x,u))$

Model Free Error Representation:

$$\mathcal{E}^\theta(X_k,U_k) = -Q^\theta(X_k,U_k) + c(X_k,U_k) + \underline{Q}^\theta(X_{k+1})$$

From DP to Q-learning

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Model Free Error Representation:

$$\mathcal{E}^\theta(X_k, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

Goal: find θ^* such that $\mathcal{E}^{\theta^*}(X_k, U_k) \approx 0$

From DP to Q-learning

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Model Free Error Representation:

$$\mathcal{E}^\theta(X_k, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

Optimization Criterion:

$$L(\theta) \stackrel{\text{def}}{=} \mathsf{E}_\infty[\mathcal{E}^\theta(X, U)^2] = \lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{T-1} \mathcal{E}^\theta(X_k, U_k)^2$$

assuming this exists for each θ

From DP to Q-learning

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Model Free Error Representation:

$$\mathcal{E}^\theta(X_k, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

Optimization Criterion:

$$L(\theta) \stackrel{\text{def}}{=} \mathsf{E}_\infty[\mathcal{E}^\theta(X, U)^2] = \lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{T-1} \mathcal{E}^\theta(X_k, U_k)^2$$

Input: stable feedback + mixture of sinusoids, $U_k = \phi(X_k) + \xi_k$

Just one option

From DP to Q-learning

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Model Free Error Representation:

$$\mathcal{E}^\theta(X_k, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

Optimization Criterion:

$$L(\theta) \stackrel{\text{def}}{=} \mathsf{E}_\infty[\mathcal{E}^\theta(X, U)^2] = \lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{T-1} \mathcal{E}^\theta(X_k, U_k)^2$$

Input: stable feedback + mixture of sinusoids, $U_k = \phi(X_k) + \xi_k$

Find zeros of $\bar{f}(\theta) = -\nabla_\theta L(\theta)$

From DP to Q-learning

$X_{k+1} = \mathrm{F}(X_k, U_k)$

Model Free Error Representation:

$$\mathcal{E}^\theta(X_k, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

Optimization Criterion:

$$L(\theta) \stackrel{\text{def}}{=} \mathsf{E}_\infty[\mathcal{E}^\theta(X,U)^2] = \lim_{T\to\infty} \frac{1}{T}\sum_{k=0}^{T-1} \mathcal{E}^\theta(X_k, U_k)^2$$

Input: stable feedback + mixture of sinusoids, $U_k = \phi(X_k) + \xi_k$

Find zeros of $\bar{f}(\theta) = -\nabla_\theta L(\theta)$

Algorithm design:

Step 1: consider an ODE: $\frac{d}{dt}\theta_t = a_t \bar{f}(\theta_t)$ *stable?*

Step 2: translate

$$\theta_{n+1} = \theta_n - \alpha_{n+1}\nabla_\theta\{\mathcal{E}^\theta(X_n, U_n)^2\}\Big|_{\theta=\theta_n}$$

Q-learning for Markovian Models

$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$

Galerkin relaxation (or projected DP equation)

With the introduction of (i.i.d.) noise:

$$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$$

a controlled Markovian model.

Q-learning for Markovian Models

$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$

Galerkin relaxation (or projected DP equation)

With the introduction of (i.i.d.) noise:

$$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$$

a controlled Markovian model.

The DP equation is nearly identical, but gradient descent fails

Q-learning for Markovian Models

$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$

Galerkin relaxation (or projected DP equation)

With the introduction of (i.i.d.) noise:

$$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$$

The DP equation is nearly identical, but gradient descent fails

Q-learning

Find zeros of $\bar{f}(\theta) = \mathsf{E}_\infty[\zeta_k \mathcal{E}^\theta(X_k, X_{k+1}, U_k)]$, $\quad \theta \in \mathbb{R}^d$

$$\mathcal{E}^\theta(X_k, X_{k+1}, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

Q-learning for Markovian Models

$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$

Galerkin relaxation (or projected DP equation)

With the introduction of (i.i.d.) noise:

$$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$$

The DP equation is nearly identical, but gradient descent fails

Q-learning

Find zeros of $\bar{f}(\theta) = \mathsf{E}_\infty[\zeta_k \mathcal{E}^\theta(X_k, X_{k+1}, U_k)], \quad \theta \in \mathbb{R}^d$

$$\mathcal{E}^\theta(X_k, X_{k+1}, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

$\zeta_k \in \mathbb{R}^d$: "eligibility vector"

Q-learning for Markovian Models

$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$

Galerkin relaxation (or projected DP equation)

With the introduction of (i.i.d.) noise:

$$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$$

The DP equation is nearly identical, but gradient descent fails

Q-learning

Find zeros of $\bar{f}(\theta) = \mathsf{E}_\infty[\zeta_k \mathcal{E}^\theta(X_k, X_{k+1}, U_k)], \quad \theta \in \mathbb{R}^d$

$$\mathcal{E}^\theta(X_k, X_{k+1}, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

$\zeta_k \in \mathbb{R}^d$: "eligibility vector"

$$\bar{f}(\theta^*) = 0$$

Q-learning for Markovian Models

$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$

Galerkin relaxation (or projected DP equation)

With the introduction of (i.i.d.) noise:

$$X_{k+1} = \mathrm{F}(X_k, U_k, W_{k+1})$$

The DP equation is nearly identical, but gradient descent fails

Q-learning

Find zeros of $\bar{f}(\theta) = \mathsf{E}_\infty[\zeta_k \mathcal{E}^\theta(X_k, X_{k+1}, U_k)], \quad \theta \in \mathbb{R}^d$

$$\mathcal{E}^\theta(X_k, X_{k+1}, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

$\zeta_k \in \mathbb{R}^d$: "eligibility vector"

Design principle unchanged:

Step 1: consider an ODE: $\frac{d}{dt}\theta_t = -G_t \bar{f}(\theta_t)$ (matrix gain part of design)

Step 2: translate to a discrete time algorithm based on measurements.

Where to go from here?

Control Theory Offers Useful Tricks and Lessons

Summary

Aspects of control philosophy we have covered:

- Every control problem is multi-objective. We want to minimize fuel, get to our destination on time, minimize risk, ...
- Design is hierarchical, both in time *and* space (an approach to distributed control)
- If you have a model, use it! But recognize that no model is perfect.
- As every MLer knows: test in many non-ideal scenarios.

Control Theory Offers Useful Tricks and Lessons

Summary

Aspects of control philosophy we have covered:

- Every control problem is multi-objective. We want to minimize fuel, get to our destination on time, minimize risk, ...
- Design is hierarchical, both in time *and* space (an approach to distributed control)
- If you have a model, use it! But recognize that no model is perfect.
- As every MLer knows: test in many non-ideal scenarios.

Other tricks from the trade:

- Controlled Lyapunov functions
- Model reduction techniques many built on theory of singular perturbations.
  - Fluid models for networks, and their workload relaxations [CTCN]
  - Feature selection for neuro-dynamic programming, 2011 [25]

Control Theory Offers Useful Tricks and Lessons

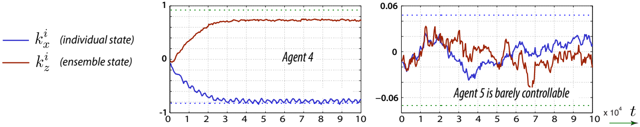
Summary

Aspects of control philosophy we have covered:

- Every control problem is multi-objective. We want to minimize fuel, get to our destination on time, minimize risk, ...
- Design is hierarchical, both in time *and* space (an approach to distributed control)
- If you have a model, use it! But recognize that no model is perfect.
- As every MLer knows: test in many non-ideal scenarios.

Other tricks from the trade:

- Controlled Lyapunov functions
- Model reduction techniques many built on theory of singular perturbations.
- Mean field game approximations for multi-agent systems



Control & RL to Come

- Every Optimization Problem Is a Quadratic Program Chapters 3 & 5

The complex nonlinear *Bellman equation* has been a road block in Q-learning

Estimating the Q-function should be easy: it is the solution to an LP or QP

Control & RL to Come

- Every Optimization Problem Is a Quadratic Program Chapters 3 & 5
- The ODE Method Chapter 4 [Basics of Algorithm Design and Analysis]

Don't start with an algorithm!

Analysis of Recursive Stochastic Algorithms

LENNART LJUNG, MEMBER, IEEE

II. THE ALGORITHM

A general recursive algorithm can be written

$$x(t)=x(t-1)+\gamma(t)Q(t;x(t-1),\varphi(t)), \qquad (1)$$

I see a noisy Euler approximation of the ODE:

$$\frac{d}{dt}x_t = q(t, x_t)$$

ODE Method: design the vector field q first, then translate to create an algorithm

Approximate policy iteration is a simple application

Control & RL to Come

- Every Optimization Problem Is a Quadratic Program Chapters 3 & 5
- The ODE Method Chapter 4
- Gradient Free Optimization and Policy Gradient RL Chapter 4

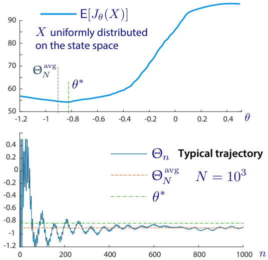
Quasi-stochastic approximation (QSA) recipe:

$$\frac{d}{dt}\bar{\Theta}_t = a_t \bar{f}(\bar{\Theta}_t) \qquad \Longleftarrow \textit{Design for your goals}$$

$$\frac{d}{dt}\Theta_t = a_t f(\Theta_t, \xi_t) \qquad \Longleftarrow \textit{QSA (cts time is simplest)}$$

$$\theta_{n+1} = \theta_n + a_{n+1} f(\theta_n, \xi_{n+1}) \qquad \Longleftarrow \textit{Euler/Runge-Kutta}$$

https://en.wikipedia.org/wiki/Runge-Kutta_methods



qPG for Mountain Car: objective function, and typical behavior of estimates

Control & RL to Come

- Every Optimization Problem Is a Quadratic Program Chapters 3 & 5
- The ODE Method Chapter 4
- Gradient Free Optimization and Policy Gradient RL Chapter 4

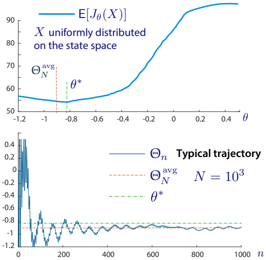
Quasi-stochastic approximation (QSA) recipe:

$$\frac{d}{dt}\bar{\Theta}_t = a_t \bar{f}(\bar{\Theta}_t) \qquad \Longleftarrow \textit{Design for your goals}$$

$$\frac{d}{dt}\Theta_t = a_t f(\Theta_t, \xi_t) \qquad \Longleftarrow \textit{QSA (cts time is simplest)}$$

$$\theta_{n+1} = \theta_n + a_{n+1} f(\theta_n, \xi_{n+1}) \qquad \Longleftarrow \textit{Euler/Runge-Kutta}$$

https://en.wikipedia.org/wiki/Runge-Kutta_methods



qPG for Mountain Car: objective function, and typical behavior of estimates

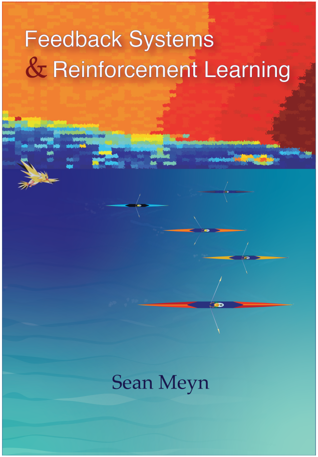
QSA much more reliable than stochastic methods. Lots of theory to explain why

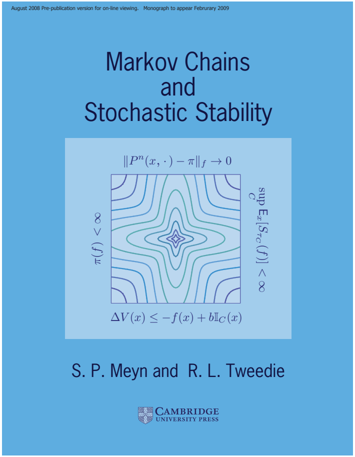
Control & RL to Come

- Every Optimization Problem Is a Quadratic Program Chapters 3 & 5
- The ODE Method Chapter 4
- Gradient Free Optimization and Policy Gradient RL Chapter 4
- Real-Time System Optimization with Applications to Power Systems

"We examine the problem of real-time optimization of networked systems and develop online algorithms that steer the system towards the optimal system trajectory..."

Andrey Bernstein [96, 97, 98]
(National Renewable Energy Laboratory)


Feedback Systems
& Reinforcement Learning
Sean Meyn


August 2008 Pre-publication version for on-line viewing. Monograph to appear Februrary 2009
Markov Chains
and
Stochastic Stability
$\|P^n(x,\cdot)-\pi\|_f \to 0$
$\pi(f) < \infty$
$\sup_C \mathrm{E}_x[S_{\tau_C}(f)] < \infty$
$\Delta V(x) \le -f(x) + b\mathbb{I}_C(x)$
S. P. Meyn and R. L. Tweedie
CAMBRIDGE
UNIVERSITY PRESS

References

Control Background I

[1] K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, USA, 2008 (recent edition on-line).

[2] K. J. Åström and K. Furuta. Swinging up a pendulum by energy control. *Automatica*, 36(2):287 – 295, 2000.

[3] K. J. Astrom and B. Wittenmark. *Adaptive Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

[4] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos. *Nonlinear and adaptive control design*. John Wiley & Sons, Inc., 1995.

[5] K. J. Åström. Theory and applications of adaptive control—a survey. *Automatica*, 19(5):471–486, 1983.

[6] K. J. Åström. Adaptive control around 1960. *IEEE Control Systems Magazine*, 16(3):44–49, 1996.

[7] B. Wittenmark. Stochastic adaptive control methods: a survey. *International Journal of Control*, 21(5):705–730, 1975.

[8] L. Ljung. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, 22(4):551–575, 1977.

Control Background II

[9] N. Matni, A. Proutiere, A. Rantzer, and S. Tu. From self-tuning regulators to reinforcement learning and back again. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 3724–3740, 2019.

RL Background I

[10] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press. On-line edition at `http://www.cs.ualberta.ca/~sutton/book/the-book.html`, Cambridge, MA, 2nd edition, 2018.

[11] C. Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

[12] R. S. Sutton.*Learning to predict by the methods of temporal differences. Mach. Learn.*, 3(1):9–44, 1988.

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

[14] J. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202, 1994.

[15] T. Jaakola, M. Jordan, and S. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201, 1994.

[16] J. N. Tsitsiklis and B. Van Roy. *An analysis of temporal-difference learning with function approximation. IEEE Trans. Automat. Control*, 42(5):674–690, 1997.

[17] 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.

RL Background II

[18] D. Choi and B. Van Roy. *A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning. Discrete Event Dynamic Systems: Theory and Applications*, 16(2):207–239, 2006.

[19] S. J. Bradtke and A. G. Barto. *Linear least-squares algorithms for temporal difference learning. Mach. Learn.*, 22(1-3):33–57, 1996.

[20] J. A. Boyan. *Technical update: Least-squares temporal difference learning. Mach. Learn.*, 49(2-3):233–246, 2002.

[21] A. Nedic and D. Bertsekas. *Least squares policy evaluation algorithms with linear function approximation. Discrete Event Dyn. Systems: Theory and Appl.*, 13(1-2):79–110, 2003.

[22] C. Szepesvári. *The asymptotic convergence-rate of Q-learning.* In *Proceedings of the 10th Internat. Conf. on Neural Info. Proc. Systems*, 1064–1070. MIT Press, 1997.

[23] E. Even-Dar and Y. Mansour. *Learning rates for Q-learning. Journal of Machine Learning Research*, 5(Dec):1–25, 2003.

[24] M. G. Azar, R. Munos, M. Ghavamzadeh, and H. Kappen. *Speedy Q-learning.* In *Advances in Neural Information Processing Systems*, 2011.

RL Background III

[25] D. Huang, W. Chen, P. Mehta, S. Meyn, and A. Surana. *Feature selection for neuro-dynamic programming.* In F. Lewis, editor, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley, 2011.

[26] A. M. Devraj, A. Bušić, and S. Meyn. Fundamental design principles for reinforcement learning algorithms. In *Handbook on Reinforcement Learning and Control*. Springer, 2020.

[27] S. P. Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, 2007. *See last chapter on simulation and average-cost TD learning*

DQN:

[28] M. Riedmiller. *Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method.* In J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, editors, *Machine Learning: ECML 2005*, pages 317–328, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[29] S. Lange, T. Gabel, and M. Riedmiller. *Batch reinforcement learning.* In *Reinforcement learning*, pages 45–73. Springer, 2012.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. *Playing Atari with deep reinforcement learning. ArXiv*, abs/1312.5602, 2013.

RL Background IV

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. *Human-level control through deep reinforcement learning.* *Nature*, 518:529–533, 2015.

Actor Critic / Policy Gradient

[32] P. J. Schweitzer. Perturbation theory and finite Markov chains. *J. Appl. Prob.*, 5:401–403, 1968.

[33] C. D. Meyer, Jr. The role of the group generalized inverse in the theory of finite Markov chains. *SIAM Review*, 17(3):443–464, 1975.

[34] P. W. Glynn. Stochastic approximation for Monte Carlo optimization. In *Proceedings of the 18th conference on Winter simulation*, pages 356–365, 1986.

[35] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[36] T. Jaakkola, S. P. Singh, and M. I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in neural information processing systems*, pages 345–352, 1995.

RL Background V

[37] X.-R. Cao and H.-F. Chen. Perturbation realization, potentials, and sensitivity analysis of Markov processes. *IEEE Transactions on Automatic Control*, 42(10):1382–1393, Oct 1997.

[38] P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of Markov reward processes. *IEEE Trans. Automat. Control*, 46(2):191–209, 2001.

[39] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[40] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[41] P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of Markov reward processes. *IEEE Trans. Automat. Control*, 46(2):191–209, 2001.

[42] S. M. Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

RL Background VI

[43] H. Mania, A. Guy, and B. Recht. Simple random search provides a competitive approach to reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1800–1809, 2018.

MDPs, LPs and Convex Q:

[44] A. S. Manne. Linear programming and sequential decisions. *Management Sci.*, 6(3):259–267, 1960.

[45] C. Derman. *Finite State Markovian Decision Processes*, volume 67 of *Mathematics in Science and Engineering*. Academic Press, Inc., 1970.

[46] V. S. Borkar. Convex analytic methods in Markov decision processes. In *Handbook of Markov decision processes*, volume 40 of *Internat. Ser. Oper. Res. Management Sci.*, pages 347–375. Kluwer Acad. Publ., Boston, MA, 2002.

[47] D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Res.*, 51(6):850–865, 2003.

[48] D. P. de Farias and B. Van Roy. *A cost-shaping linear program for average-cost approximate dynamic programming with performance guarantees.* *Math. Oper. Res.*, 31(3):597–620, 2006.

RL Background VII

[49] P. G. Mehta and S. P. Meyn. *Q-learning and Pontryagin's minimum principle.* In *Proc. of the IEEE Conf. on Dec. and Control*, pages 3598–3605, Dec. 2009.

[50] P. G. Mehta and S. P. Meyn. *Convex Q-learning, part 1: Deterministic optimal control.* *ArXiv e-prints:2008.03559*, 2020.

Gator Nation:

[51] A. M. Devraj and S. P. Meyn. *Fastest convergence for Q-learning.* *ArXiv* , July 2017 (extended version of NIPS 2017).

[52] A. M. Devraj. *Reinforcement Learning Design with Optimal Learning Rate*. PhD thesis, University of Florida, 2019.

[53] A. M. Devraj and S. P. Meyn. Q-learning with Uniformly Bounded Variance: Large Discounting is Not a Barrier to Fast Learning. *arXiv e-prints 2002.10301, and to appear AISTATS*, Feb. 2020.

[54] A. M. Devraj, A. Bušić, and S. Meyn. On matrix momentum stochastic approximation and applications to Q-learning. In *Allerton Conference on Communication, Control, and Computing*, pages 749–756, Sep 2019.

Stochastic Miscellanea I

[55] S. Asmussen and P. W. Glynn. *Stochastic Simulation: Algorithms and Analysis*, volume 57 of *Stochastic Modelling and Applied Probability*. Springer-Verlag, New York, 2007.

[56] P. W. Glynn and S. P. Meyn. *A Liapounov bound for solutions of the Poisson equation. Ann. Probab.*, 24(2):916–931, 1996.

[57] S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability*. Cambridge University Press, Cambridge, second edition, 2009. Published in the Cambridge Mathematical Library.

[58] R. Douc, E. Moulines, P. Priouret, and P. Soulier. *Markov Chains*. Springer, 2018.

Stochastic Approximation I

[59] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Hindustan Book Agency and Cambridge University Press, Delhi, India & Cambridge, UK, 2008.

[60] A. Benveniste, M. Métivier, and P. Priouret. *Adaptive algorithms and stochastic approximations*, volume 22 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, 1990. Translated from the French by Stephen S. Wilson.

[61] 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.

[62] M. Benaïm. Dynamics of stochastic approximation algorithms. In *Séminaire de Probabilités, XXXIII*, pages 1–68. Springer, Berlin, 1999.

[63] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23(3):462–466, 09 1952.

[64] D. Ruppert. *A Newton-Raphson version of the multivariate Robbins-Monro procedure. The Annals of Statistics*, 13(1):236–245, 1985.

[65] D. Ruppert. *Efficient estimators from a slowly convergent Robbins-Monro processes.* Technical Report Tech. Rept. No. 781, Cornell University, School of Operations Research and Industrial Engineering, Ithaca, NY, 1988.

Stochastic Approximation II

[66] B. T. Polyak. *A new method of stochastic approximation type. Avtomatika i telemekhanika, 98–107, 1990 (in Russian). Translated in Automat. Remote Control, 51 1991*.

[67] B. T. Polyak and A. B. Juditsky. *Acceleration of stochastic approximation by averaging. SIAM J. Control Optim.*, 30(4):838–855, 1992.

[68] V. R. Konda and J. N. Tsitsiklis. *Convergence rate of linear two-time-scale stochastic approximation. Ann. Appl. Probab.*, 14(2):796–819, 2004.

[69] E. Moulines and F. R. Bach. *Non-asymptotic analysis of stochastic approximation algorithms for machine learning.* In *Advances in Neural Information Processing Systems 24*, 451–459. Curran Associates, Inc., 2011.

[70] S. Chen, A. M. Devraj, A. Bušić, and S. Meyn. Explicit Mean-Square Error Bounds for Monte-Carlo and Linear Stochastic Approximation. *arXiv e-prints*, 2002.02584, Feb. 2020.

[71] W. Mou, C. Junchi Li, M. J. Wainwright, P. L. Bartlett, and M. I. Jordan. On Linear Stochastic Approximation: Fine-grained Polyak-Ruppert and Non-Asymptotic Concentration. *arXiv e-prints*, page arXiv:2004.04719, Apr. 2020.

Optimization and ODEs I

[72] W. Su, S. Boyd, and E. Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *Advances in neural information processing systems*, pages 2510–2518, 2014.

[73] B. Shi, S. S. Du, W. Su, and M. I. Jordan. Acceleration via symplectic discretization of high-resolution differential equations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5744–5752. Curran Associates, Inc., 2019.

[74] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[75] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, 1983.

QSA and Extremum Seeking Control I

[76] S. Chen, A. Bernstein, A. Devraj, and S. Meyn. Accelerating optimization and reinforcement learning with quasi-stochastic approximation. *arXiv:In preparation*, 2020.

[77] B. Lapeybe, G. Pages, and K. Sab. Sequences with low discrepancy generalisation and application to Robbins-Monro algorithm. *Statistics*, 21(2):251–272, 1990.

[78] S. Laruelle and G. Pagès. Stochastic approximation with averaging innovation applied to finance. *Monte Carlo Methods and Applications*, 18(1):1–51, 2012.

[79] S. Shirodkar and S. Meyn. Quasi stochastic approximation. In *Proc. of the 2011 American Control Conference (ACC)*, pages 2429–2435, July 2011.

[80] A. Bernstein, Y. Chen, M. Colombino, E. Dall'Anese, P. Mehta, and S. Meyn. Optimal rate of convergence for quasi-stochastic approximation. *arXiv:1903.07228*, 2019.

[81] A. Bernstein, Y. Chen, M. Colombino, E. Dall'Anese, P. Mehta, and S. Meyn. Quasi-stochastic approximation and off-policy reinforcement learning. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 5244–5251, Mar 2019.

[82] Y. Chen, A. Bernstein, A. Devraj, and S. Meyn. Model-Free Primal-Dual Methods for Network Optimization with Application to Real-Time Optimal Power Flow. In *Proc. of the American Control Conf.*, pages 3140–3147, Sept. 2019.

QSA and Extremum Seeking Control II

[83] S. Bhatnagar and V. S. Borkar. Multiscale chaotic spsa and smoothed functional algorithms for simulation optimization. *Simulation*, 79(10):568–580, 2003.

[84] S. Bhatnagar, M. C. Fu, S. I. Marcus, and I.-J. Wang. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 13(2):180–209, 2003.

[85] M. Le Blanc. Sur l'electrification des chemins de fer au moyen de courants alternatifs de frequence elevee [On the electrification of railways by means of alternating currents of high frequency]. *Revue Generale de l'Electricite*, 12(8):275–277, 1922.

[86] Y. Tan, W. H. Moase, C. Manzie, D. Nešić, and I. M. Y. Mareels. Extremum seeking from 1922 to 2010. In *Proceedings of the 29th Chinese Control Conference*, pages 14–26, July 2010.

[87] P. F. Blackman. Extremum-seeking regulators. In *An Exposition of Adaptive Control*. Macmillan, 1962.

[88] J. Sternby. Adaptive control of extremum systems. In H. Unbehauen, editor, *Methods and Applications in Adaptive Control*, pages 151–160, Berlin, Heidelberg, 1980. Springer Berlin Heidelberg.

QSA and Extremum Seeking Control III

[89] J. Sternby. Extremum control systems–an area for adaptive control? In *Joint Automatic Control Conference*, number 17, page 8, 1980.

[90] K. B. Ariyur and M. Krstić. *Real Time Optimization by Extremum Seeking Control*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[91] M. Krstić and H.-H. Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36(4):595 – 601, 2000.

[92] S. Liu and M. Krstic. Introduction to extremum seeking. In *Stochastic Averaging and Stochastic Extremum Seeking*, Communications and Control Engineering. Springer, London, 2012.

[93] O. Trollberg and E. W. Jacobsen. On the convergence rate of extremum seeking control. In *European Control Conference (ECC)*, pages 2115–2120. 2014.

Selected Applications I

[94] N. S. Raman, A. M. Devraj, P. Barooah, and S. P. Meyn. *Reinforcement learning for control of building HVAC systems.* In *American Control Conference*, July 2020.

[95] K. Mason and S. Grijalva. A review of reinforcement learning for autonomous building energy management. *arXiv.org*, 2019. arXiv:1903.05196.

News from Andrey@NREL:

[96] A. Bernstein and E. Dall'Anese. Real-time feedback-based optimization of distribution grids: A unified approach. *IEEE Transactions on Control of Network Systems*, 6(3):1197–1209, 2019.

[97] A. Bernstein, E. Dall'Anese, and A. Simonetto. Online primal-dual methods with measurement feedback for time-varying convex optimization. *IEEE Transactions on Signal Processing*, 67(8):1978–1991, 2019.

[98] Y. Chen, A. Bernstein, A. Devraj, and S. Meyn. Model-free primal-dual methods for network optimization with application to real-time optimal power flow. In *2020 American Control Conference (ACC)*, pages 3140–3147, 2020.