McGill Artificial Intelligence Society

Lecture 8: Reinforcement Learning: AN Optimization Tour

Announcements

Last Lecture of the Bootcamp!

Science Fair Coming Up

Blog Posts, make sure they're ready

Today's Lesson Plan

1. What is Reinforcement Learning
2. Bandit Problems
3. Markov Decision Processes
4. Connection to Supervised Learning
5. Imitation Learning
6. Model-Based Reinforcement Learning
7. Approximate Dynamic Programming
8. Direct Policy Search
    a. Policy Gradient
    b. Pure Random Search
    c. Guided Policy Search

What is Reinforcement Learning

- So far, we've been concerned with supervised learning
  - Given $D = \{X, y\}^N$ find function f such that $P(f(X) = Y < \epsilon) > 1 - \delta$
- Reinforcement Learning concerns itself with a much different paradigm
  - Deals with Autoregressive Time Series Problems
  - Allows us to make long term decisions

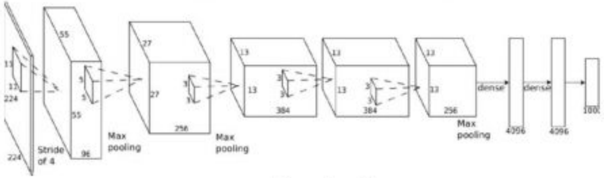
What is Reinforcement Learning

Control Theory 🤔

~~Reinforcement Learning~~ is the study of how to use past data to enhance the future manipulation of a dynamical system

Recht 2018

What is Reinforcement Learning

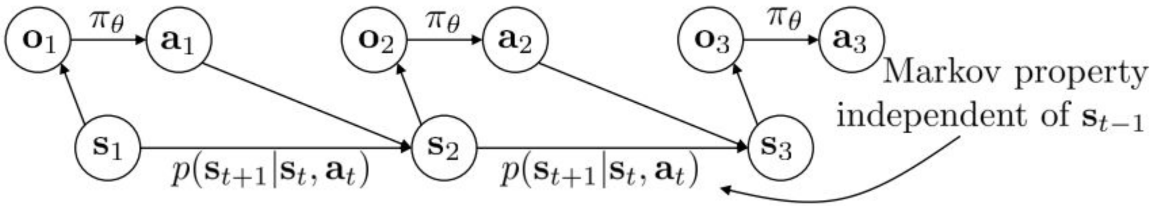


$\mathbf{o}_t$ $\qquad$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\qquad$ $\mathbf{a}_t$

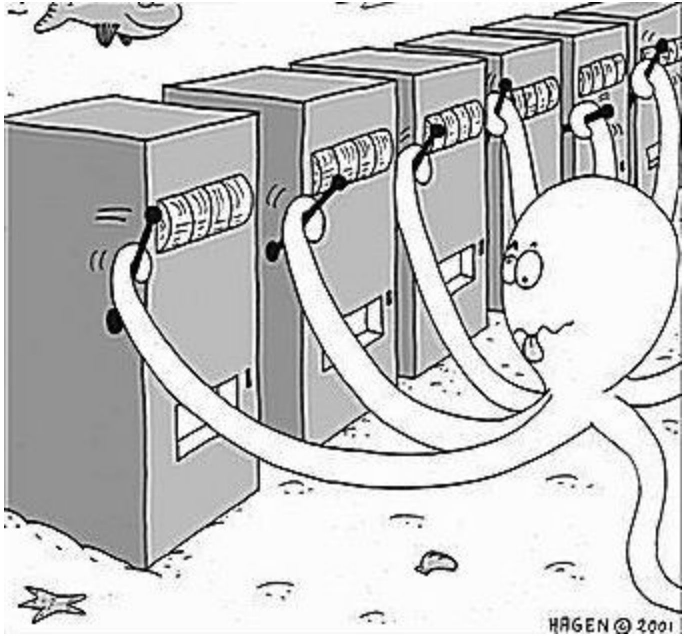
$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)

Levine 2018



Bandit Problems



Barto 2018

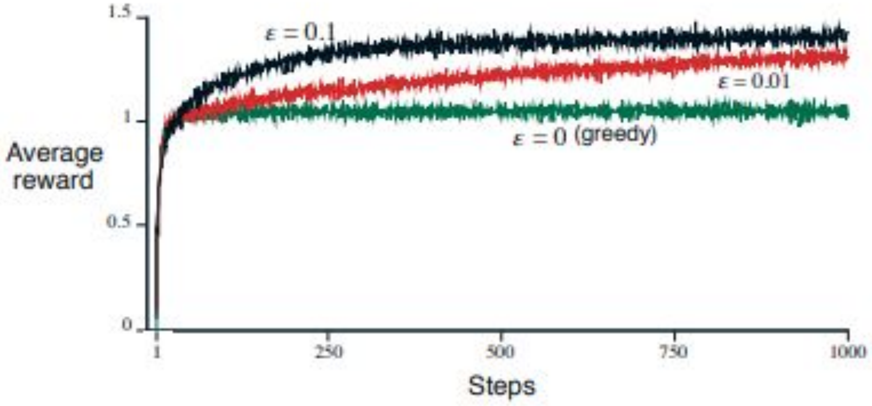
Bandit Problems

- Trying to find $q_*(a) = \mathbb{E}[R_t | A_t = a]$
- Applying $\underset{a}{argmax}\, Q_t(a) = \mathbb{E}[R_t | A_t = a]$ is called a greedy algorithm
- But what if our estimate of Q is wrong?

Barto 2018

Exploration vs. Exploitation

- Exploration is the idea of taking what you consider to be suboptimal actions in order to discover if they're optimal
- ϵ - greedy, with a probability ϵ , you take a suboptimal action



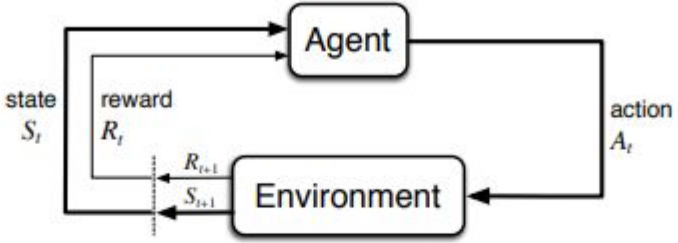
Barto 2018

Further Readings

- Upper Confidence Bounds - http://www.jmlr.org/papers/volume3/auer02a/auer02a.pdf
- Thompson Sampling Tutorial - https://web.stanford.edu/~bvr/pubs/TS_Tutorial.pdf
- E3 algorithm - https://www.cis.upenn.edu/~mkearns/papers/reinforcement.pdf
- The “Bandit” book: https://banditalgs.com/

Markov Decision Processes

- Given, s_t take action a_t, and get tuple (r_t, s_{t+1})
- Goal is to maximize $V(s_0) = \sum_{i=1}^{N} r_t$



Barto 2018

Markov Decision Processes

- Problem with this formulation
- Say you're solving a maze, and by solving the maze, you get a reward = 1, otherwise, you get reward = 0
    - Previous formulation would therefore not care if you completed it in the next step or in 10 years
- More relatably
    - 1 million dollars is worth more to you tomorrow than it is in 10 years
    - But how do we capture this?

Barto 2018

Markov Decision Processes

- Instead of using $V(s_0) = \sum_{i=1}^{N} r_t$, we solve the discounted sum of rewards

$$V(s_0) = \sum_{i=1}^{N} \gamma^t r_t$$

- A morbid interpretation of this is that there is a probability gamma at each time step that you will die, and therefore, you could interpret the formulation as
  - With probability gamma, you die
  - With probability 1 - gamma, you get reward r_t
- Full formulation

Barto 2018

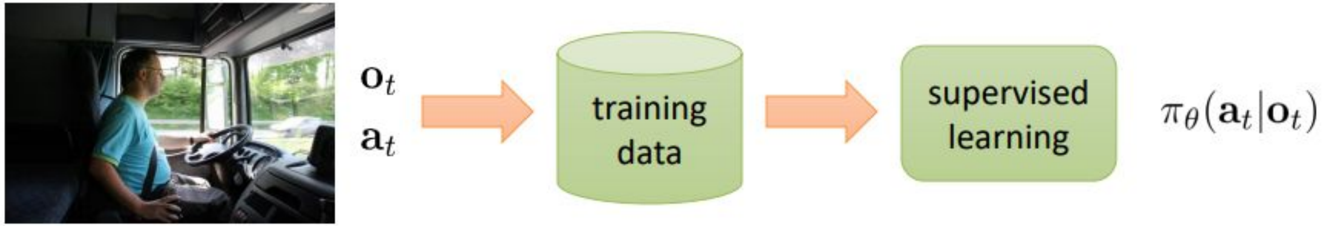
$$\begin{array}{ll} \text{maximize} & \mathbb{E}_{e_t}[\sum_{t=0}^{N} R_t[x_t, u_t]] \\ \text{subject to} & x_{t+1} = f_t(x_t, u_t, e_t) \\ & (x_0 \text{ given}). \end{array}$$

Connection to Supervised Learning

- Instead of x -> y, we now have to consider two variables: action a (or u) and reward r
- Reinforcement Learning is evidently more challenging than supervised learning, but it allows us to work in a much more complex paradigm.
- Difficulties:
  - Non i.i.d data
  - Complex feedback loop
  - Exploration Exploitation Tradeoff
  - Online updates
  - Defining rewards

Recht 2018

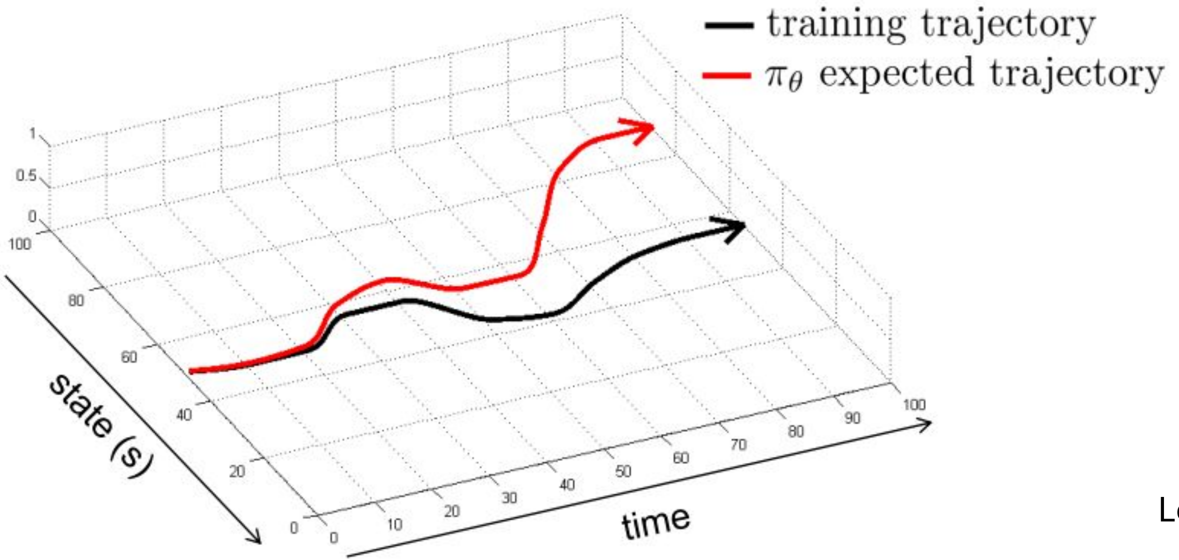
Imitation Learning



behavior cloning

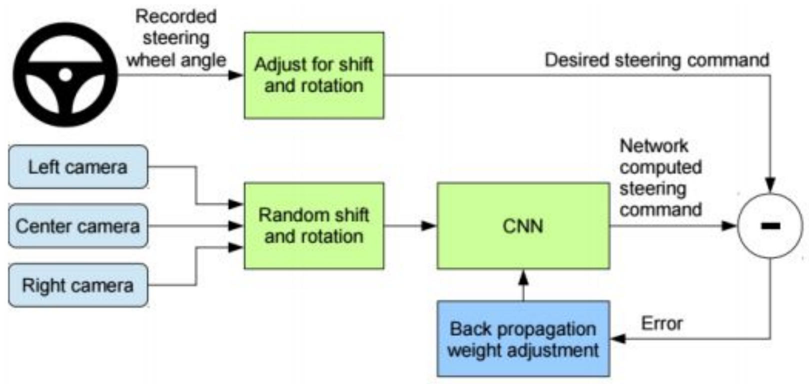
Levine 2018

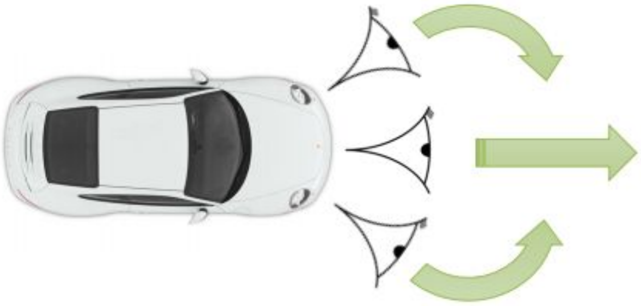
Does It Work? Not really



Levine 2018

Does It Work? Ok…maybe sometimes





Levine 2018

A Slightly Smarter Solution

DAgger: Dataset Aggregation

goal: collect training data from $p_{\pi_\theta}(\mathbf{o}_t)$ instead of $p_{\text{data}}(\mathbf{o}_t)$

how? just run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

but need labels $\mathbf{a}_t$!

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Levine 2018

Why We Might Fail To Fit

1. Non-Markovian behavior
2. Multimodal behavior

$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$$

behavior depends only on current observation

$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_1, ..., \mathbf{o}_t)$$

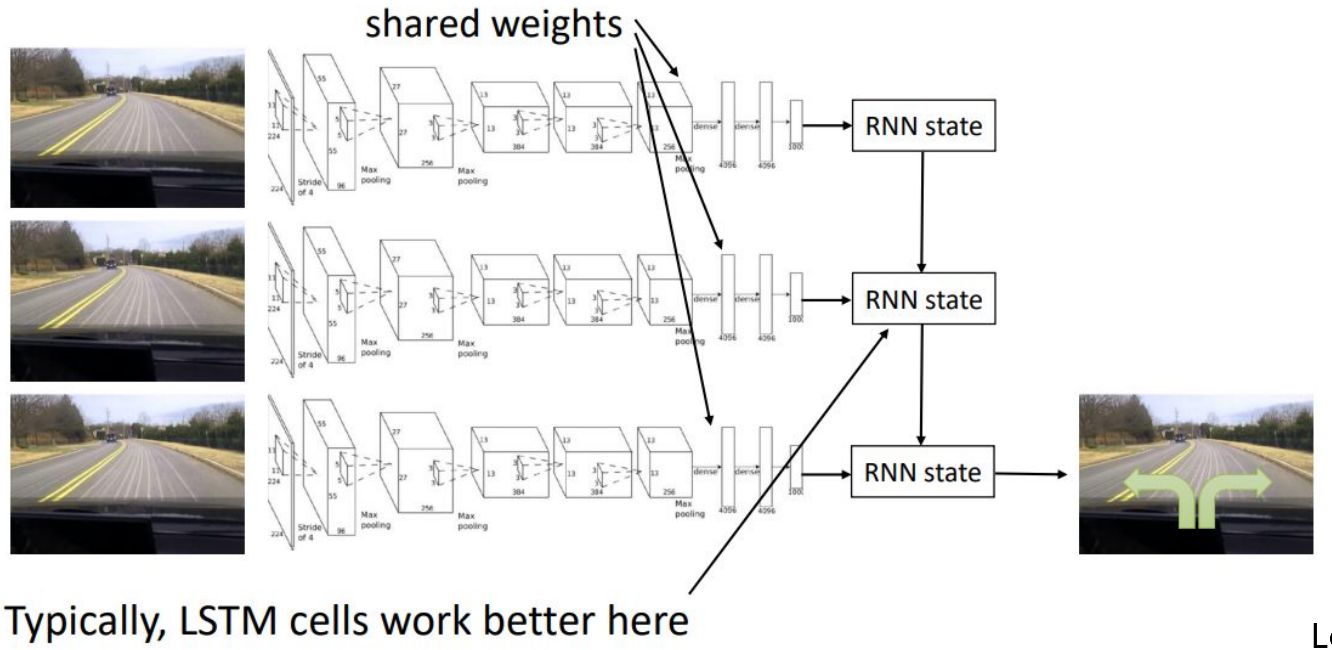
behavior depends on all past observations

Levine 2018

If we see the same thing twice, we do the same thing twice, regardless of what happened before

Often very unnatural for human demonstrators

How We Could Make Use Of Full History

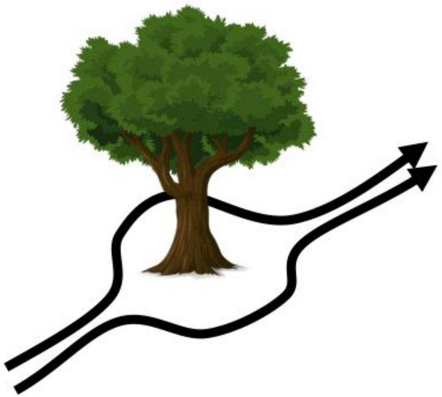


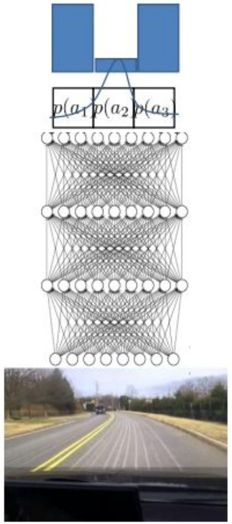
Levine 2018

How We Could Make Use Of Full History

Why might we fail to fit the expert?

1. Non-Markovian behavior
2. Multimodal behavior



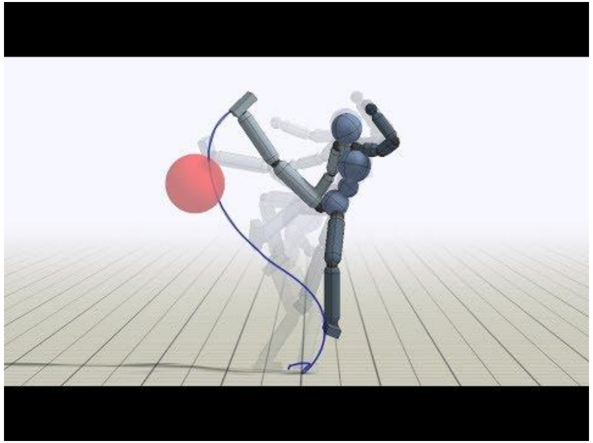


1. Output mixture of Gaussians
2. Latent variable models
3. Autoregressive discretization

Levine 2018

Imitation Learning: Further Reading

- Generative Adversarial Imitation Learning (GAIL): https://arxiv.org/abs/1606.03476
- Learning Robust Rewards with Adversarial Inverse Reinforcement Learning: https://arxiv.org/abs/1710.11248



Model-Based RL

- Fit a Predictive Model and use Dynamic Programming to plan for prescribed control problem
  - This estimated model is known as the nominal model
  - Control design is nominal controller
- Estimation of a dynamical system is called system identification
- Suppose we want to build a predictor of x_{t+1} from past history data
  - Simple strategy is to inject random probing sequence {u}^t and see how the system responds. Up to stochastic noise, we should get

$$x_{t+1} \approx \varphi(x_t, u_t) \,,$$

Recht 2018

Model-Based RL

- In such a case, we are focused on the following objective function

$$\text{minimize}_{\varphi} \quad \sum_{t=0}^{N-1} ||x_{t+1} - \varphi(x_t, u_t)||^2 \,.$$

- So the optimal control problem becomes

$$\begin{array}{ll} \text{maximize} & \mathbb{E}_{\omega_t}[\sum_{t=0}^{N} R(x_t, u_t)] \\ \text{subject to} & x_{t+1} = \hat{\varphi}(x_t, u_t) + \omega_t, \;\; u_t = \pi_t(\tau_t) \,. \end{array}$$

Recht 2018

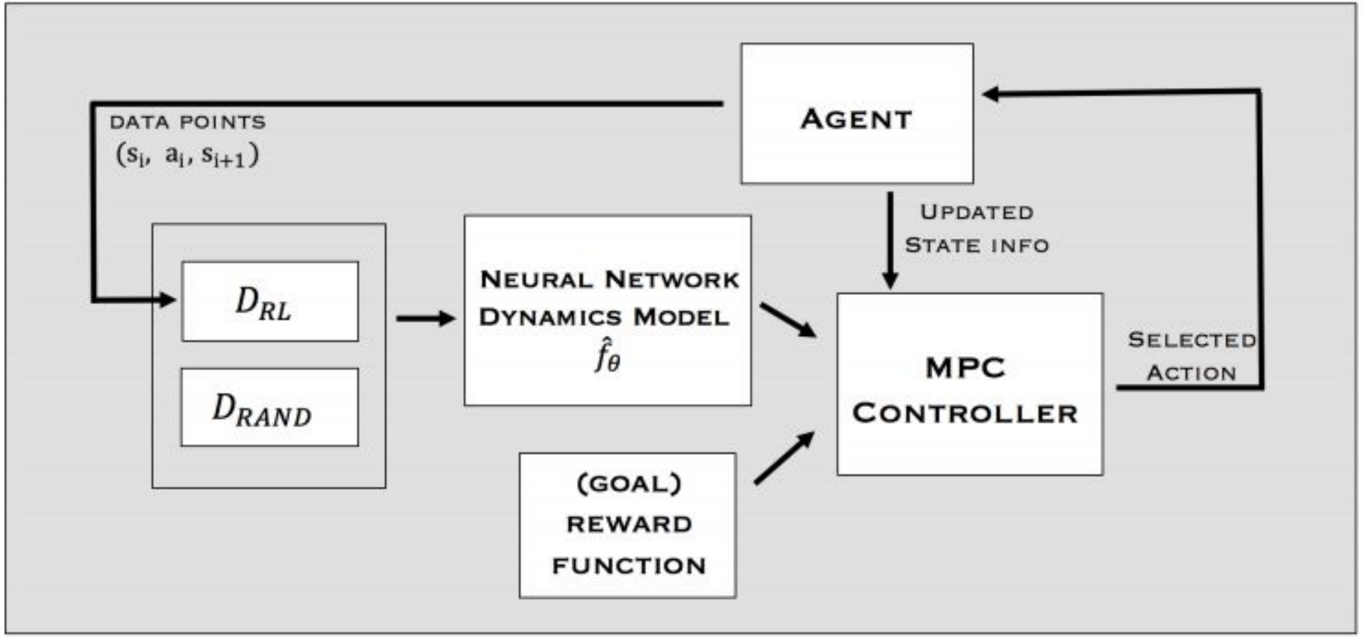
Model-Based RL

Nagabandi 2017

Algorithm 1 Model-based Reinforcement Learning

1: gather dataset $\mathcal{D}_{\text{RAND}}$ of random trajectories
2: initialize empty dataset $\mathcal{D}_{\text{RL}}$, and randomly initialize $\hat{f}_\theta$
3: **for** iter=1 **to** max_iter **do**
4:     train $\hat{f}_\theta(\mathbf{s}, \mathbf{a})$ by performing gradient descent on Eqn. 2, using $\mathcal{D}_{\text{RAND}}$ and $\mathcal{D}_{\text{RL}}$
5:     **for** $t = 1$ **to** T **do**
6:         get agent's current state $\mathbf{s}_t$
7:         use $\hat{f}_\theta$ to estimate optimal action sequence $\mathbf{A}_t^{(H)}$ (Eqn. 4)
8:         execute first action $\mathbf{a}_t$ from selected action sequence $\mathbf{A}_t^{(H)}$
9:         add $(\mathbf{s}_t, \mathbf{a}_t)$ to $\mathcal{D}_{\text{RL}}$
10:     **end for**
11: **end for**

Model-Based RL



Nagabandi 2017

Case Study: Linear Quadratic Regulator

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f(\mathbf{x}_t, \mathbf{u}_t), \Sigma)$$

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t$$

$$\mathbf{A}_t = \frac{df}{d\mathbf{x}_t} \qquad \mathbf{B}_t = \frac{df}{d\mathbf{u}_t}$$

$Q(\mathbf{x}_t, \mathbf{u}_t)$ is the cost to go: total cost we get after taking an action

$$Q(\mathbf{x}_t, \mathbf{u}_t) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{Q}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{q}_t$$

Case Study: Linear Quadratic Regulator

$$\begin{array}{ll} \text{minimize} & \mathbb{E}_{e_t}\left[\frac{1}{2}\sum_{t=0}^{N} x_t^T Q x_t + u_t^T R u_t + \frac{1}{2} x_{N+1}^T S x_{N+1}\right], \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t + e_t, \; u_t = \pi_t(\tau_t) \\ & (x_0 \text{ given}). \end{array}$$

$$u_t = -Kx_t \qquad K = (R + B^T M B)^{-1} B^T M A$$

$$M = Q + A^T M A - (A^T M B)(R + B^T M B)^{-1}(B^T M A)$$

But we don’t actually know the model!

Case Study: Linear Quadratic Regulator

$$\text{minimize}_{A,B} \quad \sum_{t=0}^{N-1} ||x_{t+1} - Ax_t - Bu_t||^2 \,.$$

1. Use supervised learning to learn a coarse model of the dynamical system to be controlled. I will refer to the system estimate as the *nominal system*.

2. Using either prior knowledge or statistical tools like the bootstrap, build probabilistic guarantees about the distance between the nominal system and the true, unknown dynamics.

3. Solve a *robust optimization* problem that optimizes control of the nominal system while penalizing signals with respect to the estimated uncertainty, ensuring stable, robust execution.

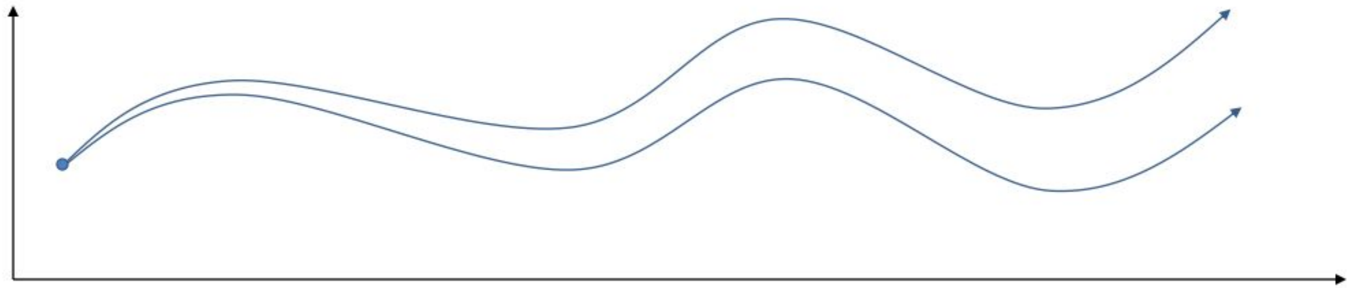
$$\frac{\hat{J} - J_\star}{J_\star} = \tilde{O}\left(\sqrt{\frac{d+p}{T}}\right) \,.$$

State dimension d, control dimension p, trajectory T

Guided Policy Search

shooting method: optimize over actions only

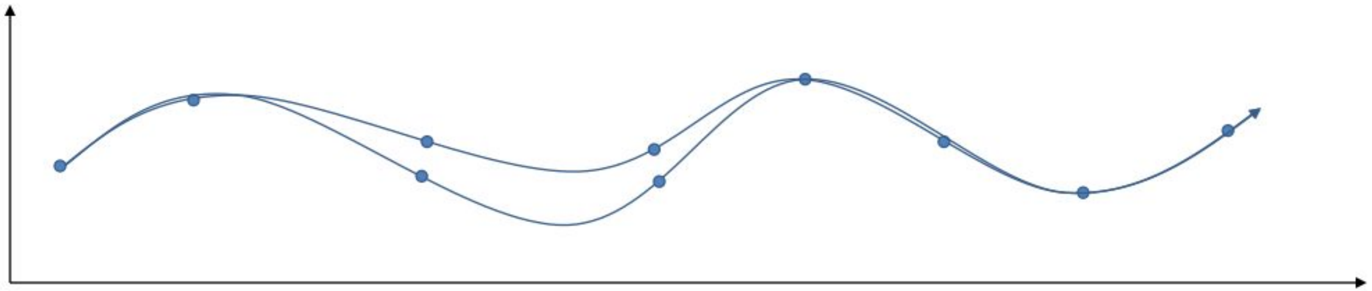
$$\min_{\mathbf{u}_1,\dots,\mathbf{u}_T} c(\mathbf{x}_1,\mathbf{u}_1) + c(f(\mathbf{x}_1,\mathbf{u}_1),\mathbf{u}_2) + \cdots + c(f(f(\dots)\dots),\mathbf{u}_T)$$



Known Model Methods

collocation method: optimize over actions and states, with constraints

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T,\mathbf{x}_1,\ldots,\mathbf{x}_T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \text{ s.t. } \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$



KMM: Start with Linear Dynamics

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} c(\mathbf{x}_1, \mathbf{u}_1) + c(f(\mathbf{x}_1, \mathbf{u}_1), \mathbf{u}_2) + \cdots + c(f(f(\ldots)\ldots), \mathbf{u}_T)$$

$$\underset{\text{linear}}{\underline{f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t}} \qquad \underset{\text{quadratic}}{\underline{c(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{C}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{c}_t}}$$

KMM: Start with Linear Dynamics

Base case: solve for $\mathbf{u}_T$ *only*

$$\mathbf{C}_T = \begin{bmatrix} \mathbf{C}_{\mathbf{x}_T,\mathbf{x}_T} & \mathbf{C}_{\mathbf{x}_T,\mathbf{u}_T} \\ \mathbf{C}_{\mathbf{u}_T,\mathbf{x}_T} & \mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T} \end{bmatrix}$$

$$\mathbf{c}_T = \begin{bmatrix} \mathbf{c}_{\mathbf{x}_T} \\ \mathbf{c}_{\mathbf{u}_T} \end{bmatrix}$$

$$Q(\mathbf{x}_T, \mathbf{u}_T) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_T \\ \mathbf{u}_T \end{bmatrix}^T \mathbf{C}_T \begin{bmatrix} \mathbf{x}_T \\ \mathbf{u}_T \end{bmatrix} + \begin{bmatrix} \mathbf{x}_T \\ \mathbf{u}_T \end{bmatrix}^T \mathbf{c}_T$$

$$\nabla_{\mathbf{u}_T} Q(\mathbf{x}_T, \mathbf{u}_T) = \mathbf{C}_{\mathbf{u}_T,\mathbf{x}_T}\mathbf{x}_T + \mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}\mathbf{u}_T + \mathbf{c}_{\mathbf{u}_T}^T = 0$$

$$\mathbf{K}_T = -\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}^{-1}\mathbf{C}_{\mathbf{u}_T,\mathbf{x}_T}$$

$$\mathbf{u}_T = -\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}^{-1}(\mathbf{C}_{\mathbf{u}_T,\mathbf{x}_T}\mathbf{x}_T + \mathbf{c}_{\mathbf{u}_T}) \qquad \mathbf{u}_T = \mathbf{K}_T\mathbf{x}_T + \mathbf{k}_T \qquad \mathbf{k}_T = -\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}^{-1}\mathbf{c}_{\mathbf{u}_T}$$

KMM: Start with Linear Dynamics

$$\mathbf{u}_T = \mathbf{K}_T\mathbf{x}_T + \mathbf{k}_T \qquad \mathbf{K}_T = -\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}^{-1}\mathbf{C}_{\mathbf{u}_T,\mathbf{x}_T} \qquad \mathbf{k}_T = -\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}^{-1}\mathbf{c}_{\mathbf{u}_T}$$

$$Q(\mathbf{x}_T,\mathbf{u}_T) = \text{const} + \frac{1}{2}\begin{bmatrix}\mathbf{x}_T\\ \mathbf{u}_T\end{bmatrix}^T \mathbf{C}_T \begin{bmatrix}\mathbf{x}_T\\ \mathbf{u}_T\end{bmatrix} + \begin{bmatrix}\mathbf{x}_T\\ \mathbf{u}_T\end{bmatrix}^T \mathbf{c}_T$$

Since $\mathbf{u}_T$ is fully determined by $\mathbf{x}_T$, we can eliminate it via substitution!

$$V(\mathbf{x}_T) = \text{const} + \frac{1}{2}\begin{bmatrix}\mathbf{x}_T\\ \mathbf{K}_T\mathbf{x}_T + \mathbf{k}_T\end{bmatrix}^T \mathbf{C}_T \begin{bmatrix}\mathbf{x}_T\\ \mathbf{K}_T\mathbf{x}_T + \mathbf{k}_T\end{bmatrix} + \begin{bmatrix}\mathbf{x}_T\\ \mathbf{K}_T\mathbf{x}_T + \mathbf{k}_T\end{bmatrix}^T \mathbf{c}_T$$

$$\begin{aligned} V(\mathbf{x}_T) =& \frac{1}{2}\mathbf{x}_T^T\mathbf{C}_{\mathbf{x}_T,\mathbf{x}_T}\mathbf{x}_T + \frac{1}{2}\mathbf{x}_T^T\mathbf{C}_{\mathbf{x}_T,\mathbf{u}_T}\mathbf{K}_T\mathbf{x}_T + \frac{1}{2}\mathbf{x}_T^T\mathbf{K}_T^T\mathbf{C}_{\mathbf{u}_T,\mathbf{x}_T}\mathbf{x}_T + \frac{1}{2}\mathbf{x}_T^T\mathbf{K}_T^T\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}\mathbf{K}_T\mathbf{x}_T + \\ & \mathbf{x}_T^T\mathbf{K}_T^T\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}\mathbf{k}_T + \frac{1}{2}\mathbf{x}_T^T\mathbf{C}_{\mathbf{x}_T,\mathbf{u}_T}\mathbf{k}_T + \mathbf{x}_T^T\mathbf{c}_{\mathbf{x}_T} + \mathbf{x}_T^T\mathbf{K}_T^T\mathbf{c}_{\mathbf{u}_T} + \text{const} \end{aligned}$$

$$V(\mathbf{x}_T) = \text{const} + \frac{1}{2}\mathbf{x}_T^T\mathbf{V}_T\mathbf{x}_T + \mathbf{x}_T^T\mathbf{v}_T \qquad \mathbf{V}_T = \mathbf{C}_{\mathbf{x}_T,\mathbf{x}_T} + \mathbf{C}_{\mathbf{x}_T,\mathbf{u}_T}\mathbf{K}_T + \mathbf{K}_T^T\mathbf{C}_{\mathbf{u}_T,\mathbf{x}_T} + \mathbf{K}_T^T\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}\mathbf{K}_T$$

$$\mathbf{v}_T = \mathbf{c}_{\mathbf{x}_T} + \mathbf{C}_{\mathbf{x}_T,\mathbf{u}_T}\mathbf{k}_T + \mathbf{K}_T^T\mathbf{C}_{\mathbf{u}_T} + \mathbf{K}_T^T\mathbf{C}_{\mathbf{u}_T,\mathbf{u}_T}\mathbf{k}_T$$

KMM: Start with Linear Dynamics

Solve for $\mathbf{u}_{T-1}$ in terms of $\mathbf{x}_{T-1}$

$\mathbf{u}_{T-1}$ affects $\mathbf{x}_T$!

$$f(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) = \mathbf{x}_T = \mathbf{F}_{T-1} \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix} + \mathbf{f}_{T-1}$$

$$Q(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix}^T \mathbf{C}_{T-1} \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix} + \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix}^T \mathbf{c}_{T-1} + \underbrace{V(f(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}))}_{V(\mathbf{x}_T) = \text{const} + \frac{1}{2}\mathbf{x}_T^T \mathbf{V}_T \mathbf{x}_T + \mathbf{x}_T^T \mathbf{v}_T}$$

$$V(\mathbf{x}_T) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix}^T \underbrace{\mathbf{F}_{T-1}^T \mathbf{V}_T \mathbf{F}_{T-1}}_{\text{quadratic}} \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix} + \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix}^T \underbrace{\mathbf{F}_{T-1}^T \mathbf{V}_T \mathbf{f}_{T-1}}_{\text{linear}} + \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix}^T \underbrace{\mathbf{F}_{T-1}^T \mathbf{v}_T}_{\text{linear}}$$

KMM: Start with Linear Dynamics

Solve for $\mathbf{u}_{T-1}$ in terms of $\mathbf{x}_{T-1}$

$\mathbf{u}_{T-1}$ affects $\mathbf{x}_T$!

$$f(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) = \mathbf{x}_T = \mathbf{F}_{T-1} \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix} + \mathbf{f}_{T-1}$$

$$Q(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix}^T \mathbf{C}_{T-1} \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix} + \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix}^T \mathbf{c}_{T-1} + \underbrace{V(f(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}))}_{V(\mathbf{x}_T) = \text{const} + \frac{1}{2}\mathbf{x}_T^T \mathbf{V}_T \mathbf{x}_T + \mathbf{x}_T^T \mathbf{v}_T}$$

$$V(\mathbf{x}_T) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix}^T \underbrace{\mathbf{F}_{T-1}^T \mathbf{V}_T \mathbf{F}_{T-1}}_{\text{quadratic}} \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix} + \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix}^T \underbrace{\mathbf{F}_{T-1}^T \mathbf{V}_T \mathbf{f}_{T-1}}_{\text{linear}} + \begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{u}_{T-1} \end{bmatrix}^T \underbrace{\mathbf{F}_{T-1}^T \mathbf{v}_T}_{\text{linear}}$$

KMM: Start with Linear Dynamics

$$Q(\mathbf{x}_{T-1},\mathbf{u}_{T-1}) = \text{const} + \frac{1}{2}\begin{bmatrix}\mathbf{x}_{T-1}\\ \mathbf{u}_{T-1}\end{bmatrix}^T \mathbf{C}_{T-1}\begin{bmatrix}\mathbf{x}_{T-1}\\ \mathbf{u}_{T-1}\end{bmatrix} + \begin{bmatrix}\mathbf{x}_{T-1}\\ \mathbf{u}_{T-1}\end{bmatrix}^T \mathbf{c}_{T-1} + V(f(\mathbf{x}_{T-1},\mathbf{u}_{T-1}))$$

$$V(\mathbf{x}_T) = \text{const} + \frac{1}{2}\begin{bmatrix}\mathbf{x}_{T-1}\\ \mathbf{u}_{T-1}\end{bmatrix}^T \underbrace{\mathbf{F}_{T-1}^T\mathbf{V}_T\mathbf{F}_{T-1}}_{\text{quadratic}}\begin{bmatrix}\mathbf{x}_{T-1}\\ \mathbf{u}_{T-1}\end{bmatrix} + \begin{bmatrix}\mathbf{x}_{T-1}\\ \mathbf{u}_{T-1}\end{bmatrix}^T \underbrace{\mathbf{F}_{T-1}^T\mathbf{V}_T\mathbf{f}_{T-1}}_{\text{linear}} + \begin{bmatrix}\mathbf{x}_{T-1}\\ \mathbf{u}_{T-1}\end{bmatrix}^T \underbrace{\mathbf{F}_{T-1}^T\mathbf{v}_T}_{\text{linear}}$$

$$Q(\mathbf{x}_{T-1},\mathbf{u}_{T-1}) = \text{const} + \frac{1}{2}\begin{bmatrix}\mathbf{x}_{T-1}\\ \mathbf{u}_{T-1}\end{bmatrix}^T \mathbf{Q}_{T-1}\begin{bmatrix}\mathbf{x}_{T-1}\\ \mathbf{u}_{T-1}\end{bmatrix} + \begin{bmatrix}\mathbf{x}_{T-1}\\ \mathbf{u}_{T-1}\end{bmatrix}^T \mathbf{q}_{T-1}$$

$$\mathbf{Q}_{T-1} = \mathbf{C}_{T-1} + \mathbf{F}_{T-1}^T\mathbf{V}_T\mathbf{F}_{T-1}$$

$$\mathbf{q}_{T-1} = \mathbf{c}_{T-1} + \mathbf{F}_{T-1}^T\mathbf{V}_T\mathbf{f}_{T-1} + \mathbf{F}_{T-1}^T\mathbf{v}_T$$

$$\nabla_{\mathbf{u}_{T-1}} Q(\mathbf{x}_{T-1},\mathbf{u}_{T-1}) = \mathbf{Q}_{\mathbf{u}_{T-1},\mathbf{x}_{T-1}}\mathbf{x}_{T-1} + \mathbf{Q}_{\mathbf{u}_{T-1},\mathbf{u}_{T-1}}\mathbf{u}_{T-1} + \mathbf{q}_{\mathbf{u}_{T-1}}^T = 0$$

$$\mathbf{u}_{T-1} = \mathbf{K}_{T-1}\mathbf{x}_{T-1} + \mathbf{k}_{T-1} \qquad \mathbf{K}_{T-1} = -\mathbf{Q}_{\mathbf{u}_{T-1},\mathbf{u}_{T-1}}^{-1}\mathbf{Q}_{\mathbf{u}_{T-1},\mathbf{x}_{T-1}} \qquad \mathbf{k}_{T-1} = -\mathbf{Q}_{\mathbf{u}_{T-1},\mathbf{u}_{T-1}}^{-1}\mathbf{q}_{\mathbf{u}_{T-1}}$$

KMM: Start with Linear Dynamics

Backward recursion

for $t = T$ to 1:

$$\mathbf{Q}_t = \mathbf{C}_t + \mathbf{F}_t^T \mathbf{V}_{t+1} \mathbf{F}_t$$

$$\mathbf{q}_t = \mathbf{c}_t + \mathbf{F}_t^T \mathbf{V}_{t+1} \mathbf{f}_t + \mathbf{F}_t^T \mathbf{v}_{t+1}$$

$$Q(\mathbf{x}_t, \mathbf{u}_t) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{Q}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{q}_t$$

$$\mathbf{u}_t \leftarrow \arg\min_{\mathbf{u}_t} Q(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$$

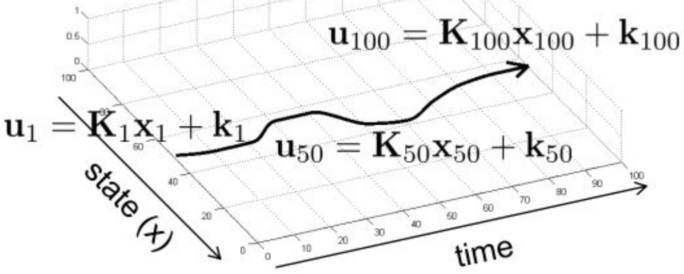
$$\mathbf{K}_t = -\mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t}^{-1} \mathbf{Q}_{\mathbf{u}_t,\mathbf{x}_t}$$

$$\mathbf{k}_t = -\mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t}^{-1} \mathbf{q}_{\mathbf{u}_t}$$

$$\mathbf{V}_t = \mathbf{Q}_{\mathbf{x}_t,\mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t,\mathbf{u}_t} \mathbf{K}_t + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t,\mathbf{x}_t} + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t} \mathbf{K}_t$$

$$\mathbf{v}_t = \mathbf{q}_{\mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t,\mathbf{u}_t} \mathbf{k}_t + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t} + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t} \mathbf{k}_t$$

$$V(\mathbf{x}_t) = \text{const} + \frac{1}{2} \mathbf{x}_t^T \mathbf{V}_t \mathbf{x}_t + \mathbf{x}_t^T \mathbf{v}_t$$



we know $\mathbf{x}_1$!

Forward recursion

for $t = 1$ to T:

$$\mathbf{u}_t = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$$

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$$

Model-Based: Further Reading

- PILCO: http://mlg.eng.cam.ac.uk/pub/pdf/DeiRas11.pdf
- DeepPilco: http://mlg.eng.cam.ac.uk/yarin/PDFs/DeepPILCO.pdf
- SOLAR: https://arxiv.org/pdf/1808.09105.pdf
- Gaussian Processes for RL: https://papers.nips.cc/paper/2420-gaussian-processes-in-reinforcement-learning.pdf
- LQR Sample Complexity: https://arxiv.org/abs/1710.01688
- Multi-step model-based: https://arxiv.org/pdf/1811.00128.pdf
- Theoretical guarantees for model-based RL: https://openreview.net/pdf?id=BJe1E2R5KX
- Guided Policy Search: https://graphics.stanford.edu/projects/gpspaper/gps_full.pdf

Approximate Dynamic Programming

- Directly approximates optimal control costs and solves this using techniques from dynamic programming

$$Q(x, u) = \max \left\{ \mathbb{E}_{e_t} \left[\sum_{t=0}^{N} R(x_t, u_t) \right] \; : \; x_{t+1} = f(x_t, u_t, e_t),\ (x_0, u_0) = (x, u) \right\}$$

- Optimal policy therefore is

$$u_k = \arg\max\nolimits_u Q_k(x_k, u)$$

Approximate Dynamic Programming

$$\mathcal{Q}_\gamma(x_k, u_k) \approx R(x_k, u_k) + \gamma \max_{u'} \mathcal{Q}_\gamma(x_{k+1}, u') \,.$$

With Update Rule

$$Q_\gamma^{(\text{new})}(x_k, u_k) = (1-\eta)Q_\gamma^{(\text{old})}(x_k, u_k) + \eta\left(R(x_k, u_k) + \gamma \max_{u'} Q_\gamma^{(\text{old})}(x_{k+1}, u')\right)$$

Or alternatively, in the value iteration case

$$V(x) = \max_u \mathcal{Q}(x, u) \,.$$

Approximate Dynamic Programming

online Q iteration algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
2. $\mathbf{y}_i = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i)$

these are correlated!

isn't this just gradient descent? that converges, right?

Q-learning is *not* gradient descent!

$$\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)])$$

no gradient through target value

Approximate Dynamic Programming

full Q-learning with replay buffer:

1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, add it to $\mathcal{B}$

$K\times$

2. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$
3. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)])$

Stabilize regression with

full fitted Q-iteration algorithm:

1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy

$K\times$

2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\underline{\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2}$

perfectly well-defined, stable regression

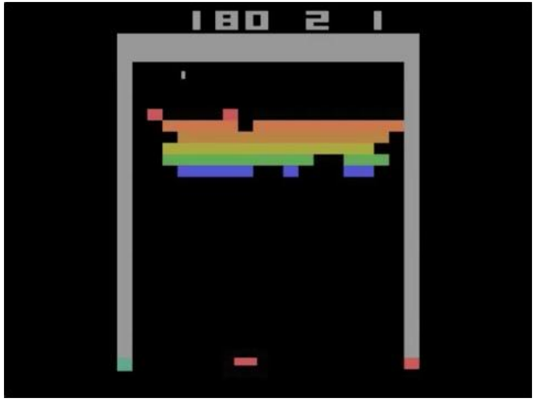
DQN (Mnih et. al 2015)

Q-learning with replay buffer and target network:

1. save target network parameters: $\phi' \leftarrow \phi$
2. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, add it to $\mathcal{B}$
3. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$
4. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}'_i, \mathbf{a}'_i)])$

$N\times$

$K\times$



Further Reading

- DQN: https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
- Double DQN: https://arxiv.org/abs/1509.06461
- Dueling DQN: https://arxiv.org/abs/1511.06581
- DDPG: https://arxiv.org/pdf/1509.02971.pdf
- Is Q-learning Provably Efficient: https://arxiv.org/pdf/1807.03765.pdf

Direct Policy Search

- The idea is, use as little information as possible, still find a policy.

$$\text{maximize}_{\vartheta} \quad \mathbb{E}_{p(z;\vartheta)}[R(z)] .$$

- So we try to maximize a cost function

$$J(\vartheta) := \mathbb{E}_{p(z;\vartheta)}[R(z)]$$

- And with a touch of nice math, we get

$$\begin{aligned}
\nabla_{\vartheta} J(\vartheta) &= \int R(z) \nabla_{\vartheta} p(z;\vartheta) dz \\
&= \int R(z) \left(\frac{\nabla_{\vartheta} p(z;\vartheta)}{p(z;\vartheta)} \right) p(z;\vartheta) dz \\
&= \int \left(R(z) \nabla_{\vartheta} \log p(z;\vartheta) \right) p(z;\vartheta) dz \\
&= \mathbb{E}_{p(z;\vartheta)} \left[R(z) \nabla_{\vartheta} \log p(z;\vartheta) \right] .
\end{aligned}$$

Direct Policy Search

- The idea is, use as little information as possible, still find a policy.

$$\text{maximize}_{\vartheta} \quad \mathbb{E}_{p(z;\vartheta)}[R(z)] \, .$$

- So we try to maximize a cost function

$$J(\vartheta) \; := \; \mathbb{E}_{p(z;\vartheta)}[R(z)]$$

- And with a touch of nice math, we get

$$\begin{aligned}
\nabla_{\vartheta} J(\vartheta) &= \int R(z) \nabla_{\vartheta} p(z;\vartheta) dz \\
&= \int R(z) \left(\frac{\nabla_{\vartheta} p(z;\vartheta)}{p(z;\vartheta)} \right) p(z;\vartheta) dz \\
&= \int \left(R(z) \nabla_{\vartheta} \log p(z;\vartheta) \right) p(z;\vartheta) dz \\
&= \mathbb{E}_{p(z;\vartheta)} \left[R(z) \nabla_{\vartheta} \log p(z;\vartheta) \right] \, .
\end{aligned}$$

Direct Policy Search

Algorithm 1 REINFORCE

1: **Hyperparameters:** step-sizes $\alpha_j > 0$.
2: **Initialize:** ϑ_0 and $k = 0$.
3: **while** ending condition not satisfied **do**
4: Sample $z_k \sim p(z; \vartheta_k)$.
5: Set $\vartheta_{k+1} = \vartheta_k + \alpha_k R(z_k) \nabla_\vartheta \log p(z_k; \vartheta_k)$.
6: $k \leftarrow k + 1$
7: **end while**

Pure Random Search

- All we do is perturb decision variable z, and see what happens
- Sample from some distribution

$$p(z;\vartheta) = p_0(z-\vartheta)$$

- With a gradient estimation

$$g_\sigma(\vartheta) = \frac{R(\vartheta+\sigma\epsilon) - R(\vartheta-\sigma\epsilon)}{2\sigma}\epsilon\,.$$

- And averaging

$$g_\sigma^{(m)}(\vartheta) = \frac{1}{m}\sum_{i=1}^{m}\frac{R(\vartheta+\sigma\epsilon_i) - R(\vartheta-\sigma\epsilon_i)}{2\sigma}\epsilon_i\,.$$

Policy Gradients: Further Reading

- Random Search is competitive in RL: https://arxiv.org/abs/1803.07055
- TRPO: https://arxiv.org/abs/1502.05477
- PPO: https://arxiv.org/pdf/1707.06347.pdf

Further Reading

- RL Book: http://incompleteideas.net/book/bookdraft2017nov5.pdf
- Neuro-dynamic programming: http://athenasc.com/ndpbook.html

Thanks!

Congratulations, you've made it to the end :)