Computer Engineering Department

Reinforcement Learning: Model Based RL

Mohammad Hossein Rohban, Ph.D.

Spring 2024

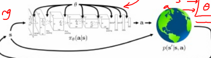
Courtesy: Most of slides are adopted from CS 285 Berkeley.

Overview

- Introduction to model-based reinforcement learning
- What if we know the dynamics? How can we make decisions?
- Stochastic optimization methods
- Monte Carlo tree search (MCTS)
- Trajectory optimization
- Goal: Understand how we can perform planning with known dynamics models in discrete and continuous spaces

Recap: Model-Free RL

Planning



$a \;\; s \rightarrow \boxed{\theta} \rightarrow s'$

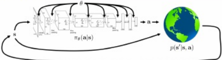
$r(s, a)$

Learning

$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

$$\theta^* = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Recap: Model-Free RL



$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}$$

assume this is unknown
don't even attempt to learn it

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

What if we knew the transition dynamics?

- Often we do know the dynamics
  - Games (e.g., Atari games, chess, Go)
  - Easily modeled systems (e.g., navigating a car)
  - Simulated environments (e.g., simulated robots, video games)
- Often we can learn the dynamics
  - System identification – fit unknown parameters of a known model
  - Learning – fit a general-purpose model to observed transition data

Does knowing the dynamics make things easier?

Often, yes!

Model-based RL

- Model-based reinforcement learning: learn the transition dynamics, then figure out how to choose actions.
- Today: how can we make decisions if we know the dynamics?
  - a. How can we choose actions under perfect knowledge of the system dynamics?
  - b. Optimal control, trajectory optimization, planning

Model Based RL

reliant on the planning

vs.

transition
reward
Model

Model Free RL

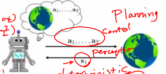
The deterministic case

$P(s_{t+1}|s_t, a_t) = \mathcal{N}(\mu, \sigma^2 I)$

Planning

Pure Planning

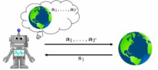
$s_t, a_t \rightarrow \boxed{f} \rightarrow s_{t+1}$



$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg\max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \text{ s.t. } \mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$$

$P(s_{t+1}|s_t, a_t) = \sum_{k=1} \alpha_k \mathcal{N}(\mu_k, \sigma_k^2)$

The stochastic open-loop case



$$p_\theta(\mathbf{s}_1, \ldots, \mathbf{s}_T | \mathbf{a}_1, \ldots, \mathbf{a}_T) = p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg\max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} E\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{a}_1, \ldots, \mathbf{a}_T\right]$$

why is this suboptimal?

The stochastic open-loop case

کری می خواست به عیادت بیماری برود.اندیشید که هنگام احوال پرسی ممکن است صدای اورانشنوم وپاسخی ناشایسته بدهم.ازاین رودرپی چاره برآمدوبالاخره باخود گفت:بهتراست پرسشهارا پیش ازرفتن بسنجم وپاسخ رانیزبرآورد کنم تادچاراشتباه نشوم

بنابراین پرسشهای خودراچنین پیش بینی کرد:

-ابتداازاومی پرسم حالت بهتراست؟ اوخواهد گفت "آری" من درجواب می گویم:خدا را شکر

-بعدازاومی پرسم چه خورده ای؟ لابد نام غذایی راخواهد آورد من می گویم گوارا باد.

-درپایان می پرسم پزشکت کیست؟ نام پزشکی رامی گویدومن پاسخ می دهم:مقدمش مبارک باد.

چون به خانه ی بیمارر‌سید همان گونه که ازپیش آماده شده بودبه احوال پرسی پرداخت:

-کر گفت:"چگونه ای؟"

بیمارگفت: مُردم

کر گفت: خدارا شکر

بیمارازاین سخن بیجا برآشفت.

-بعدازآن پرسید:"چه خورده ای؟"

بیمارگفت: زهر

کر گفت: گواراباد.داروی خوبی است.

بیمار ازاین پاسخ نیزبیشتربه خود پیچید.

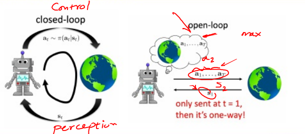
-بعد ازآن کر گفت:" ازطبیبان کیست او   کاوهمی آید به چاره پیش تو؟"

بیمار که آشفتگی وناراحتی اش به نهایت رسیده بود در پاسخ گفت:

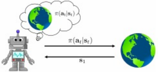
عزرائیل می آید. برو.

کر گفت: پایش بس مبارک.شاد شو!

open-loop vs. closed-loop case



The stochastic open-loop case



$$p(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T) = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

$$\pi = \arg\max_{\pi} E_{\tau \sim p(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

form of π?

neural net global

time-varying linear

$\mathbf{K}_t \mathbf{s}_t + \mathbf{k}_t$ local

Stochastic optimization

abstract away optimal control/planning:

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg\max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \underbrace{J(\mathbf{a}_1, \ldots, \mathbf{a}_T)}_{\text{don't care what this is}} \qquad \mathbf{A} = \arg\max_{\mathbf{A}} J(\mathbf{A})$$

simplest method: guess & check "random shooting method"

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)
2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

Cross-entropy Method (CEM)

$(a_1 \ldots a_T)$

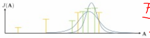
$\frac{1}{Z} e^{+J(A)}$

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)
2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

can we do better?

$(a_1 \ldots a_T)$

$P(A)$



$P(A)$

cross-entropy method with continuous-valued inputs:

$P(A)$

1. sample $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from $p(\mathbf{A})$
2. evaluate $J(\mathbf{A}_1), \ldots, J(\mathbf{A}_N)$
3. pick the *elites* $\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_M}$ with the highest value, where $M < N$
4. refit $p(\mathbf{A})$ to the elites $\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_M}$

$f(s_{i-1}, a_{i-1})$

$\rightarrow J = \sum r(s_i, a_i)$

$$\begin{bmatrix} J(A_1) \\ \vdots \\ J(A_N) \end{bmatrix} \xrightarrow{} CE\left(Z \begin{bmatrix} e^{J(A_1)} \\ \vdots \\ e^{J(A_N)} \end{bmatrix} \right),$$

$$P(A)) = \overset{\text{avg}}{\sum} \overline{P(A_i)} \quad \left(e^{J(A_i)}\right)$$

$$= \sum e^{J(A_i)} \log P(A_i)$$

$$\log P(A_i) \propto e^{J(A_i)}$$

$A_1 \cdots A_N \longrightarrow J(A_1) \cdots J(A_N)$

$$\longrightarrow \left[\frac{e^{J(A_1)}}{\sum_i e^{J(A_i)}}, \ldots, \frac{e^{J(A_N)}}{\sum_i e^{J(A_i)}} \right]$$

$\xrightarrow[\text{Sample } N]{} \longrightarrow \cdots$

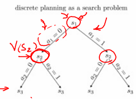
Pros and Cons

- Pros
  - Could be very fast (Parallelizable)
  - Extremely simple
- Cons
  - Very harsh dimensionality limit
  - Only open-loop planning

Discrete Case: Monte Carlo Tree Search

s_t

a_t

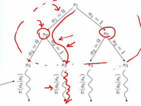


Discrete Case: Monte Carlo Tree Search

$V(s) \leftarrow \left(\frac{N}{N+1} V(s) + \frac{\hat{V}(s)}{N+1} \right)$

s_t a_t

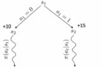
how to approximate value without full tree?



Discrete Case: Monte Carlo Tree Search

s_t a_t

can't search all paths – where to search first?



intuition: choose nodes with best reward, but also prefer rarely visited nodes

Discrete Case: Monte Carlo Tree Search

generic MCTS sketch

1. find a leaf s_l using TreePolicy(s_1)
2. evaluate the leaf using DefaultPolicy(s_l)
3. update all values in tree between s_1 and s_l

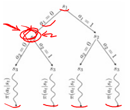
take best action from s_1

UCT TreePolicy(s_t)

if s_t not fully expanded, choose new a_t

else choose child with best Score(s_{t+1})

$$\text{Score}(s_t) = \frac{Q(s_t)}{N(s_t)} + 2C\sqrt{\frac{2\ln N(s_{t-1})}{N(s_t)}}$$



$N(s_2)$

Discrete Case: Monte Carlo Tree Search

Algorithm 7 (Monte-Carlo Tree Search)

Input : MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s, a, s') \rangle$, base Q-function Q, time limit T
Output : updated Q-function Q

while $currentTime < T$ **do**
  $selected\_node \leftarrow \text{Select}(s_0)$
  $child \leftarrow \text{Expand}(selected\_node)$ -- expand and choose a child to simulate
  $G \leftarrow \text{Simulate}(child)$ -- simulate from $child$
  $\text{Backpropagate}(selected\_node, child, Q, G)$
return Q

Discrete Case: Monte Carlo Tree Search

Algorithm 8 (Function – Select($s : S$))

Input : state s
Output : unexpanded state s

while s is fully expanded **do**
  Select action a to apply in s using a multi-armed bandit algorithm
  Choose one outcome s' according to $P_a(s' \mid s)$
  $s \leftarrow s'$
return s

Discrete Case: Monte Carlo Tree Search

Algorithm 9 (Function – $\mathrm{Expand}(s : S)$)

Input : state s
Output : expanded states$'$

if s is fully expanded **then**
  Randomly select action a to apply in s
  Expand one outcome s' according to $P_a(s' \mid s)$ and observe reward r
return s'

Discrete Case: Monte Carlo Tree Search

Algorithm 10 (Function – Backpropagation$(s:S; a:A; Q: S\times A \to \mathbb{R}; G:\mathbb{R})$)

Input : state-action pair (s,a), Q-function Q, rewards G
Output : none

do

$N(s,a) \leftarrow N(s,a)+1$

$G \leftarrow r + \gamma G$

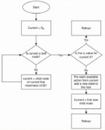
$Q(s,a) \leftarrow Q(s,a) + \frac{1}{N(s,a)}\left[G - Q(s,a)\right]$

$s \leftarrow$ parent of s

$a \leftarrow$ parent action of s

while $s \neq s_0$

Discrete Case: Monte Carlo Tree Search



Additional reading

- Browne, Powley, Whitehouse, Lucas, Cowling, Rohlfshagen, Tavener, Perez, Samothrakis, Colton. (2012). A Survey of Monte Carlo Tree Search Methods.
  - Survey of MCTS methods and basic summary.

Trajectory Optimization: Can we use derivatives?

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

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

usual story: differentiate via backpropagation and optimize!

need $\frac{df}{d\mathbf{x}_t}, \frac{df}{d\mathbf{u}_t}, \frac{dc}{d\mathbf{x}_t}, \frac{dc}{d\mathbf{u}_t}$

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

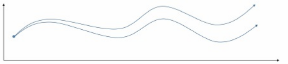
$\mathbf{x}_t$ – state
$\mathbf{u}_t$ – action

in practice, it really helps to use a 2nd order method!

Shooting methods vs collocation

shooting method: optimize over actions only

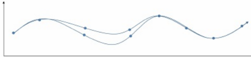
$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} c(\mathbf{x}_1, \mathbf{u}_1) + c(f(\mathbf{x}_1, \mathbf{u}_1), \mathbf{u}_2) + \cdots + c(f(f(\ldots)\ldots), \mathbf{u}_T)$$



Shooting methods vs collocation

collocation method: optimize over actions and states, with constraints

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



Linear case: LQR

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

$$f(\mathbf{x}_t,\mathbf{u}_t) = \mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t$$

linear

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

quadratic

Linear case: LQR

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} c(\mathbf{x}_1,\mathbf{u}_1) + c(f(\mathbf{x}_1,\mathbf{u}_1),\mathbf{u}_2) + \cdots + \underbrace{c(\overbrace{f(f(\ldots)\ldots)}^{\mathbf{x}_T \text{ (unknown)}},\mathbf{u}_T)}_{\text{only term that depends on } \mathbf{u}_T}$$

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

$$f(\mathbf{x}_t,\mathbf{u}_t) = \mathbf{F}_t\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix} + \mathbf{f}_t$$

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

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

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

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

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

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

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

Linear Case: LQR

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

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

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

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

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

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

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

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

Linear Case: LQR

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

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

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

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

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

Linear Case: LQR

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

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

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

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

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

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

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

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

Linear Case: LQR

Backward recursion

for $t = T$ to 1:

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

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

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

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

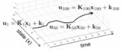
$$\mathbf{K}_t = -\mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t}^{-1} \mathbf{Q}_{\mathbf{u}_t,\mathbf{x}_t}$$

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

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

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

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



we know $\mathbf{x}_1$!

Forward recursion

for $t = 1$ to T:

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

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

LQR for Stochastic Dynamics

$$f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t$$

$$\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{u}_t)$$

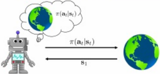
$$p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}\left(\mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t, \Sigma_t \right)$$

Solution: choose actions according to $\mathbf{u}_t = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$

$\mathbf{x}_t \sim p(\mathbf{x}_t)$, no longer deterministic, but $p(\mathbf{x}_t)$ is Gaussian

no change to algorithm! can ignore Σ_t due to symmetry of Gaussians (checking this is left as an exercise; hint: the expectation of a quadratic under a Gaussian has an analytic solution)

The stochastic closed-loop case



form of π?

$$p(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T) = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

time-varying linear

$\mathbf{K}_t \mathbf{s}_t + \mathbf{k}_t$

$$\pi = \arg\max_{\pi} E_{\tau \sim p(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Nonlinear case: DDP/iterative LQR

Linear-quadratic assumptions:

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

Can we *approximate* a nonlinear system as a linear-quadratic system?

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

$$c(\mathbf{x}_t, \mathbf{u}_t) \approx c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}^T \nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

Nonlinear case: DDP/iterative LQR

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

$$c(\mathbf{x}_t, \mathbf{u}_t) \approx c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}^T \nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

$$\bar{f}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = \underbrace{\mathbf{F}_t}_{\nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix} \qquad \bar{c}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix}^T \underbrace{\mathbf{C}_t}_{\nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix}^T \underbrace{\mathbf{c}_t}_{\nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)}$$

$$\delta\mathbf{x}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$$

$$\delta\mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t$$

Now we can run LQR with dynamics $\bar{f}$, cost $\bar{c}$, state $\delta\mathbf{x}_t$, and action $\delta\mathbf{u}_t$

Nonlinear case: DDP/iterative LQR

Iterative LQR (simplified pseudocode)

until convergence:

$\mathbf{F}_t = \nabla_{\mathbf{x}_t,\mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$

$\mathbf{c}_t = \nabla_{\mathbf{x}_t,\mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$

$\mathbf{C}_t = \nabla^2_{\mathbf{x}_t,\mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$

Run LQR backward pass on state $\delta\mathbf{x}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$ and action $\delta\mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t$

Run forward pass with real nonlinear dynamics and $\mathbf{u}_t = \mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t) + \mathbf{k}_t + \hat{\mathbf{u}}_t$

Update $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{u}}_t$ based on states and actions in forward pass

Nonlinear case: DDP/iterative LQR

Why does this work?

Compare to Newton's method for computing $\min_{\mathbf{x}} g(\mathbf{x})$:

until convergence:

$$\mathbf{g} = \nabla_{\mathbf{x}} g(\hat{\mathbf{x}})$$

$$\mathbf{H} = \nabla^2_{\mathbf{x}} g(\hat{\mathbf{x}})$$

$$\hat{\mathbf{x}} \leftarrow \arg\min_{\mathbf{x}} \frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) + \mathbf{g}^T(\mathbf{x} - \hat{\mathbf{x}})$$

Iterative LQR (iLQR) is the same idea: locally approximate a complex nonlinear function via Taylor expansion

In fact, iLQR is an approximation of Newton's method for solving

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

Nonlinear case: DDP/iterative LQR

In fact, iLQR is an approximation of Newton's method for solving

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

To get Newton's method, need to use *second order* dynamics approximation:

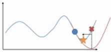
$$f(\mathbf{x}_t,\mathbf{u}_t) \approx f(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t,\mathbf{u}_t} f(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t)\begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix} + \frac{1}{2}\left(\nabla^2_{\mathbf{x}_t,\mathbf{u}_t} f(\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t)\cdot\begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix}\right)\begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix}$$

differential dynamic programming (DDP)

Nonlinear case: DDP/iterative LQR

$$\hat{\mathbf{x}} \leftarrow \arg\min_{\mathbf{x}} \frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})^T\mathbf{H}(\mathbf{x}-\hat{\mathbf{x}}) + \mathbf{g}^T(\mathbf{x}-\hat{\mathbf{x}})$$

why is this a bad idea?



until convergence:

$\mathbf{F}_t = \nabla_{\mathbf{x}_t,\mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$

$\mathbf{c}_t = \nabla_{\mathbf{x}_t,\mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$

$\mathbf{C}_t = \nabla^2_{\mathbf{x}_t,\mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$

Run LQR backward pass on state $\delta\mathbf{x}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$ and action $\delta\mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t$

Run forward pass with $\mathbf{u}_t = \mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t) + \alpha\mathbf{k}_t + \hat{\mathbf{u}}_t$

Update $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{u}}_t$ based on states and actions in forward pass

Additional Reading

- Mayne, Jacobson. (1970). Differential dynamic programming.
  - Original differential dynamic programming algorithm.
- Tassa, Erez, Todorov. (2012). Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization.
  - Practical guide for implementing non-linear iterative LQR.
- Levine, Abbeel. (2014). Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics.
  - Probabilistic formulation and trust region alternative to deterministic line search.