To summarize:

Initialize $\Phi_T = -U_T$. $\Psi_T = 0$.

Recursive calculate Φ_t, Ψ_t using Φ_{t+1}, Ψ_{t+1} with D.T.R. Eg "

(for $t = T-1, T-2, \dots, 0$)

Compute L_t using Φ_{t+1}, Ψ_{t+1}.

$\pi_t^*(s_t) = L_t s_t$

$V_t^*(s_t) = s_t^T \Phi_t s_t + \Psi_t$.

(scale polynomial complexity to high dimension)

* apply variation of this to helicopter

$\boxed{\text{Lecture 19}}$

- Debugging R.L. algorithm
- LQR
  - Differential dynamic programming (DDP)
- Kalman filter
- Linear Quadratic Gaussian (LQG)

"Advice for applying Machine Learning" [PDF]

$\max E[R(s_0,a_0)+\cdots+R(s_T,a_T)]$.

DP: $V_T^*(s)=\max_{a_T} R(s_T,a_T)$

$V_t^*(s)=\max_a R(s,a)+\sum_{s'} P_{sa}^{(t)}(s')V_{t+1}^*(s')$

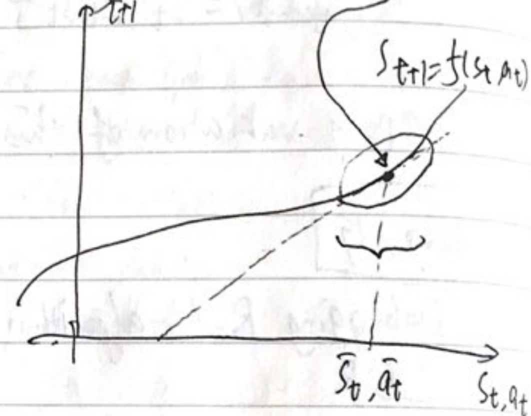
$\pi_t^*(s)=\arg\max_a R(s,a)+\sum_{s'} P_{sa}^{(t)}(s')V_{t+1}^*(s')$

$V_T^*,\ V_{T-1}^*,\ V_{T-2}^*,\ \cdots,\ V_0^*$

$s_{t+1}=f(\bar{s}_t,\bar{a}_t)$
$+(\nabla_s f(\bar{s}_t,\bar{a}_t))^T(s_t-\bar{s}_t)$
$+(\nabla_a f(\bar{s}_t,\bar{a}_t))^T(a_t-\bar{a}_t)$

$\downarrow$

$s_{t+1}=A_t s_t+B_t a_t$



LQR: $s_t\in\mathbb{R}^n$, $a_t\in\mathbb{R}^d$.

$s_{t+1}=A_t s_t+B_t a_t+\underbrace{w_t}_{\text{(ignored)}}$ ← $w_t\sim\mathcal{N}(0,\Sigma_w)$

$R(s_t,a_t)=-(s_t^T U_t s_t+a_t^T V_t a_t)$.

DP: $V_t^*(s_t)=s_t^T\Phi_t s_t+\Psi_t$ ← Do depend on noise, larger noise cause worse value function.

V_T^*: $\Phi_T=-Q_T$ $\Psi_T=0$

$V_t^*\begin{cases}\Phi_t=-A_t(\Phi_{t+1}-\Phi_{t+1}B_t(B_t^T\Phi_{t+1}B_t-V_t)^{-1}B_t\Phi_{t+1})A_t-Q_t\\ \Psi_t=-\mathrm{tr}\,\Sigma_w\Phi_{t+1}+\Psi_{t+1}\ \text{(ignored)}\end{cases}$

$\pi^*(s_t)=L_t s_t$

$L_t=(B_t^T\Phi_{t+1}B_t-V_t)^{-1}B_t^T\Phi_{t+1}A_t$ ← Not depend on Ψ

Special property of LQR systems:
① You can forget Ψ_t's!
② π^*'s don't depend on noise terms w_t's.

Kalman filter use this property.

Differential Dynamic Programming

$S_{t+1} = f(S_t, a_t)$ - simulator : nonlinear . deterministic

have wanted trajectory

apply LQR on:
helicopter.
car.
chemical factory

(1) Come up with nominal trajectory
$\bar{s}_0, \bar{a}_0, \bar{s}_1, \bar{a}_1, \ldots, \bar{s}_T, \bar{a}_T$

(2) linearize f around nominal trajectory
i.e. $S_{t+1} \approx f(\bar{s}_t, \bar{a}_t) + (\nabla_s f(\bar{s}_t, \bar{a}_t))^T (s_t - \bar{s}_t)$
$\qquad + (\nabla_a f(\bar{s}_t, \bar{a}_t))^T (a_t - \bar{a}_t)$

$= A_t s_t + B_t a_t$

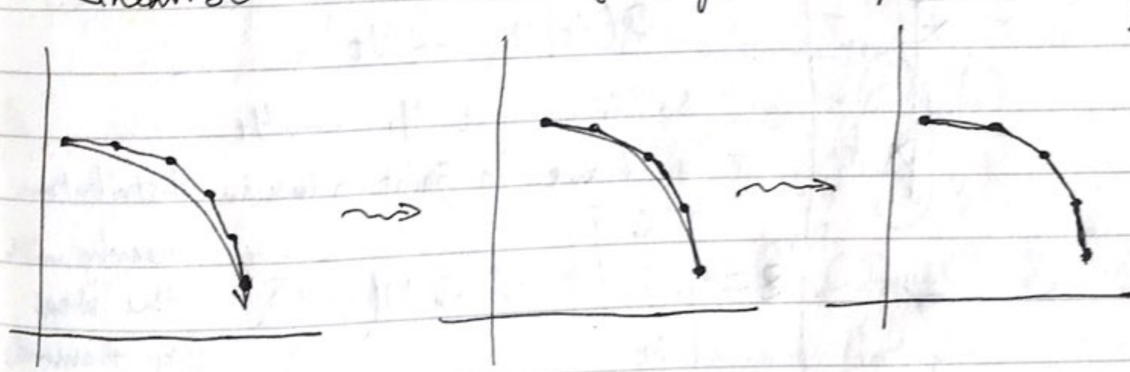
Expect $(s_t, a_t) \approx (\bar{s}_t, \bar{a}_t)$

(3) Use LQR to get π_t

(4) Use simulator to get new nominal trajectory
i.e. $\bar{s}_0$ = initial state
$\bar{a}_t = \pi_t(\bar{s}_t)$
$\bar{s}_{t+1} = f(\bar{s}_t, \bar{a}_t)$

$\bar{s}_0, \bar{a}_0, \bar{s}_1, \bar{a}_1, \ldots, \bar{s}_T, \bar{a}_T$

Linearize around new trajectory and repeat.



DDP: Local optimal search algorithm.

This works well on the Ng's helicopter and works well on many problems.

Kalman Filter & LQG

Assume: know the state of system so far: $\pi^*(s_t) = L_t s_t$.

But when cannot observe the state explicitly in some dynamic systems?

$$s_{t+1} = As_t + w_t \quad \text{(now forget control first)}$$

$$s_t = \begin{pmatrix} x_t \\ \dot{x}_t \\ \theta_t \\ \dot{\theta}_t \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 1 & & \\ & 0.9 & & \\ & & 1 & 1 \\ & & & 0.9 \end{pmatrix}$$

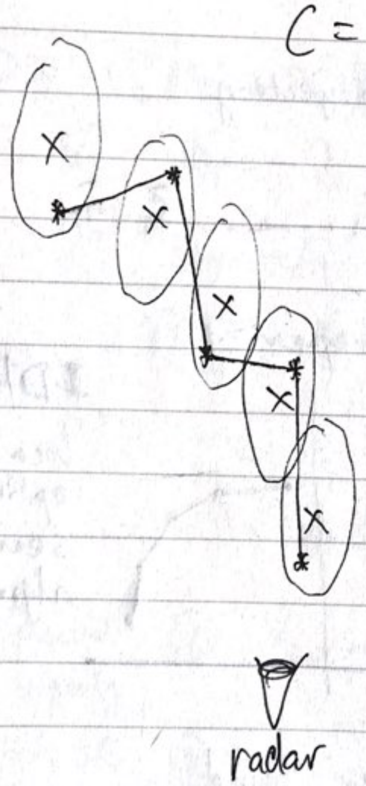
have a simulator and a radar to position the helicopter and estimate its states.

$$x_{t+1} = x_t + \dot{x}_t + \text{noise}$$
$$\dot{x}_{t+1} = 0.9\dot{x}_t + \text{noise}$$

Observe

$$y_t = Cs_t + v_t, \quad v_t \sim N(0, \Sigma_v)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad Cs_t = \begin{bmatrix} x \\ y \end{bmatrix}.$$



* observation y_t,

x actual position.

estimate distribution on the state:

Want

$$P(s_t | y_1 \ldots y_t)$$

$s_0, s_1, \ldots, s_t, y_1, \ldots, y_t$ have a joint Gaussian distribution

$$z = \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_t \\ y_1 \\ \vdots \\ y_t \end{bmatrix} \qquad z \sim N(\mu, \Sigma)$$

linearly with time steps (like: thousands)

Compute: $P(s_t | y_1 \ldots y_t)$

Computationally inefficient.

Kalman Filter

Dan's discussion on HMM's

Kalman Filter actually be a HMM.

notation:

$z \to s$

$x \to y$

Dan's $\to$ Today

Today: HMM with continous state rather than discrete states

$P(s_t \mid y_1 \cdots y_t)$

Predict

$P(s_{t+1} \mid y_1 \cdots y_t)$

Update

$P(s_{t+1} \mid y_1 \cdots y_{t+1})$

Predict step:

$s_t \mid y_1 \cdots y_t \sim N(s_{t|t}, \Sigma_{t|t})$

Then

$s_{t+1} \mid y_1 \cdots y_t \sim N(s_{t+1|t}, \Sigma_{t+1|t})$

where

$s_{t+1|t} = A s_{t|t}$

$\Sigma_{t+1|t} = A \Sigma_{t|t} A^T + \Sigma_v$

$s_t, y_t \to$ true states/observations

$s_{t|t}$ $s_{t+1|t}$

$\Sigma_{t|t}$ $\Sigma_{t+1|t}$

$\hookrightarrow$ Computations

Update step:

Get $s_{t+1} \mid y_1 \cdots y_{t+1} \sim N(s_{t+1|t+1}, \Sigma_{t+1|t+1})$

where $s_{t+1|t+1} = s_{t+1|t} + K_{t+1} \cdot (y_{t+1} - C s_{t+1|t})$

$K_{t+1} = \Sigma_{t+1|t} C^T (C \Sigma_{t+1|t} C^T + \Sigma_v)^{-1}$

$\Sigma_{t+1|t+1} = \Sigma_{t+1|t} - \Sigma_{t+1|t} C^T (C \Sigma_{t+1|t} C^T + \Sigma_v)^{-1} C \cdot \Sigma_{t+1|t}$

$s_{t+1|t+1}$ is our "best" estimate for s_{t+1}.

$$Z = \begin{bmatrix} s_0 \\ \vdots \\ s_t \\ y_1 \\ \vdots \\ y_t \end{bmatrix} \qquad Z \sim N(\mu, \Sigma)$$

$$\Sigma \in \mathbb{R}^{t \times t}$$

If Compute marginal dist. : Time complexity: $O(t^3)$

KF: $O(1)$ for every step

$$\begin{array}{ccccc} \cancel{y_1} & & \cancel{y_2} & & y_3 \\ \downarrow & & \downarrow & & \downarrow \\ P(s_1|y_1) & \rightsquigarrow & P(s_2|y_1, y_2) & \rightsquigarrow & P(s_3|y_1, y_2, y_3) \end{array}$$

Putting these together (KF + LQR = LQG).

$$s_{t+1} = A s_t + B a_t + w_t, \quad w_t \sim N(0, \Sigma_w)$$

$$y_t = C s_t + v_t, \quad v_t \sim N(0, \Sigma_v)$$

Use KF to estimate state

$$s_{0|0} = s_0 \quad \Sigma_{0|0} = 0, \text{ for } s_0 \sim N(s_{0|0}, \Sigma_{0|0}).$$

Predict $\begin{cases} s_{t+1|t} = A s_{t|t} + B a_t \\ \Sigma_{t+1|t} = A \Sigma_{t|t} A^T + \Sigma_v \end{cases}$

Compute L_t's using LQR (Assuming observed states)

$$a_t = L_t s_t \rightarrow a_t = L_t \cdot s_{t|t}$$

" $s_t = s_{t|t}$ + noise " This is optimal. Due to seperate principle.

Only hold true for spec. case like LQG

(estimate state and directly plug-in estimation into controller)

For many other systems (nonlinear or other changes in LQG) this is NOT hold true. i.e. not optimal if you do this.

Lecture 20 (The last lecture of CS229)

- POMDPs (Partially observed MDPs)
- Policy search
- Reinforce algorithm
- Pegasus algorithm
- Conclusion

$$\begin{cases} s_{t+1} = A s_t + B a_t + w_t \\ y_t = C s_t + v_t \end{cases}$$ ← observation

Actions $a_t = L_t s_t$

Compute $s_{t|t}$ (estimate for s_t)

Kalman filter: $s_t | y_1 \dots y_t \sim N(s_{t|t}, \Sigma_{t|t})$.

Actions: $a_t = L_t s_{t|t}$

---

* Find the optimal policy for POMDP is NP-hard.

POMDP: $(S, A, Y, \{P_{sa}\}, \{O_s\}, T, R)$

Y - set of possible observations

O_s - observation distributions.

At each step, observe $y_t \sim O_{s_t}$ (if in state s_t)

Policy Search (Direct policy search)

One of most effective algo's for Full-Observed MDPs and Partial-Observed MDPs (POMDPs).

Define a set Π of policies, Search for a goal $\pi \in \Pi$

(c.f. Define a set $\mathcal{H}$ of hypothesis, search for a good $h \in \mathcal{H}$)

New definition: A stochastic policy is a function.

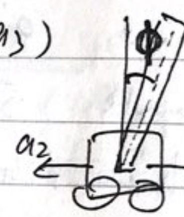
$\pi : S \times A \mapsto \mathbb{R}$ when $\pi(s,a)$ is probability of taking action "a" in state s.

$\left(\sum_a \pi(s,a) = 1. \quad \pi(s,a) \geq 0 \right)$

---

execute π: In state s. take action $a_1 : a_2 : a_3$

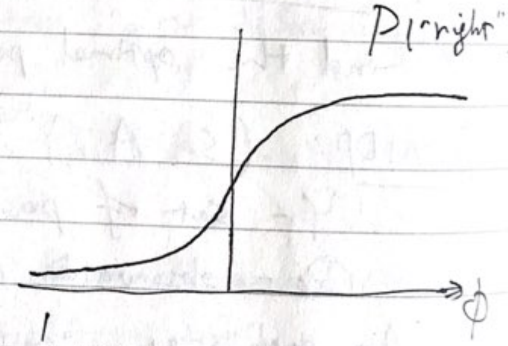
$= \pi(s,a_1) : \pi(s,a_2) : \pi(s,a_3)$



$$\pi_\theta(s,a_1) = \frac{1}{1+e^{-\theta^T s}}$$

$$\pi_\theta(s,a_2) = 1 - \frac{1}{1+e^{-\theta^T s}}$$

E.g. $S = \begin{bmatrix} 1 \\ x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} \quad \theta = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$



$$P(\text{"a=right"}) = \frac{1}{1+e^{-\theta^T s}} = \frac{1}{1+e^{-\phi}}$$

Goal: $\max_\theta E\left[R(s_0,a_0) + \cdots + R(s_T,a_T) \mid \pi_\theta, s_0 \right]$

$\theta_1, \ldots, \theta_d$: $\pi_\theta(s,a_i) = \dfrac{e^{\theta_i^T s}}{\sum_{j=1}^{d} e^{\theta_j^T s}}$ (softmax)

Reinforce Algorithm (isn't exactly the reinforce algorithm, ~~but~~ as originally presented by Ron Williams, but it captures its essence).

Assume s_0 is some fixed initial state,

$$\max E[R(s_0,a_0) + \cdots + R(s_T,a_T)]$$

$$= \sum_{\substack{s_0 a_0 \cdots \\ s_T a_T}} P(s_0 a_0 s_1 a_1 \ldots s_T a_T)\left[R(s_0,a_0) + \cdots + R(s_T,a_T)\right].$$

$$= \sum_{\substack{s_0 a_0 \cdots \\ s_T a_T}} P(s_0)\cdot \pi_\theta(s_0,a_0) P_{s_0 a_0}(s_1)\pi_\theta(s_1,a_1) \cdots\cdots \pi_\theta(s_T,a_T)$$

$$* \underbrace{\left[R(s_0,a_0) + \cdots + R(s_T,a_T)\right]}_{\text{Payoff}}.$$

Loop: {

Sample $s_0, a_0, s_1, a_1, \ldots, s_T, a_T$

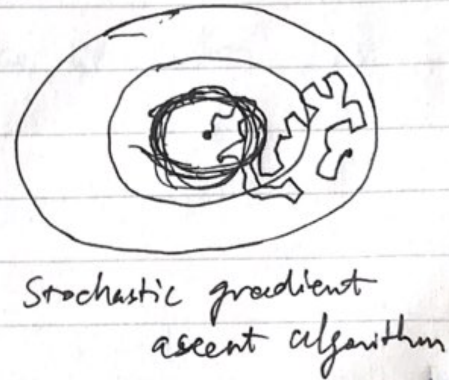
Compute payoff $= R(s_0,a_0) + \cdots + R(s_T,a_T)$

Update:

$$\theta := \theta + \alpha\left[\frac{\nabla_\theta \pi_\theta(s_0,a_0)}{\pi_\theta(s_0,a_0)} + \cdots + \frac{\nabla_\theta \pi_\theta(s_T,a_T)}{\pi_\theta(s_T,a_T)}\right]\cdot \text{payoff}$$

}

$$E\left[\left(\frac{\nabla_\theta \pi_\theta(s_0,a_0)}{\pi_\theta(s_0,a_0)} + \cdots + \frac{\nabla_\theta \pi_\theta(s_T,a_T)}{\pi_\theta(s_T,a_T)}\right)\cdot \text{payoff}\right].$$



converge slowly ... $\rightsquigarrow$ Stochastic gradient ascent algorithm

$\nabla_\theta E[\text{payoff}]$

$$\frac{d}{d\theta} f(\theta) g(\theta) h(\theta) = f'(\theta) g(\theta) h(\theta) + f(\theta) g'(\theta) h(\theta) + f(\theta) g(\theta) h'(\theta)$$

$$= \sum_{\substack{s_0 a_0 \cdots \\ s_T a_T}} \Big[P(s_0) \big(\nabla_\theta \pi_\theta(s_0,a_0)\big) P_{s_0 a_0}(s_1) \pi_\theta(s_1,a_1) \cdots \pi_\theta(s_T,a_T)$$

$$+ P(s_0) \pi_\theta(s_0,a_0) P_{s_0 a_0}(s_1) \big(\nabla_\theta \pi_\theta(s_1,a_1)\big) \cdots \pi_\theta(s_T,a_T)$$

$$+ \cdots$$

$$+ P(s_0) \pi_\theta(s_0,a_0) P_{s_0 a_0}(s_1) \pi_\theta(s_1,a_1) \cdots \big(\nabla_\theta \pi_\theta(s_T,a_T)\big) \Big]$$

$$* \text{payoff}$$

$\boxed{\text{4 steps derivation}}$

$$= \sum_{\substack{s_0 a_0 \cdots \\ s_T a_T}} P(s_0) \pi_\theta(s_0,a_0) P_{s_0 a_0}(s_1) \pi_\theta(s_1,a_1) \cdots \pi_\theta(s_T,a_T) \cdot \left[\frac{\nabla_\theta \pi_\theta(s_0,a_0)}{\pi_\theta(s_0,a_0)} + \frac{\nabla_\theta \pi_\theta(s_1,a_1)}{\pi_\theta(s_1,a_1)} + \cdots + \frac{\nabla_\theta \pi_\theta(s_T,a_T)}{\pi_\theta(s_T,a_T)} \right] \cdot \text{payoff}$$

$$= \sum_{\substack{s_0 a_0 \cdots \\ s_T a_T}} P(s_0 a_0 s_1 a_1 \cdots s_T a_T) \cdot \left[\frac{\nabla_\theta \pi_\theta(s_0,a_0)}{\pi_\theta(s_0,a_0)} + \cdots + \frac{\nabla_\theta \pi_\theta(s_T,a_T)}{\pi_\theta(s_T,a_T)} \right] \cdot \text{payoff}$$

$$= E\left[\left(\frac{\nabla_\theta \pi_\theta(s_0,a_0)}{\pi_\theta(s_0,a_0)} + \cdots + \frac{\nabla_\theta \pi_\theta(s_T,a_T)}{\pi_\theta(s_T,a_T)} \right) \cdot \text{payoff} \right].$$

① policy search algo.'s are esp. effective when you can choose a simple policy π: For the proble if exist a simple function (LP | LogitR) that maps features of states to the action).

↳ a proper policy class Π. eg: Inv. P. ~~Flying a helicopter~~

eg. Low Level control/reflexes:

Flying a helicopter. driving a car.

② If the problem requires long multi-step reasoning (eg: game of chess), high-level decision (less instinctual) making. Use value function approximation approaches instead

Have approximation $\hat{s}$ of s. (Could be $\hat{s} = S_{t|t}$ from KF).

$$\pi_\theta(s, a) = \frac{1}{1+e^{-\theta^T \hat{s}}}$$

(can use policy search algo.'s on POMDP)

(often reasonably effective to POMDPs).

* Reinforce algo. often works well, but is often ~~extremely~~ slow. (because of noise)

such as 1-100 million iterations.

* So: Sample $s_0, a_0, s_1, a_1, \ldots, s_T, a_T$.

10 million times: always on a simulator, not a physical device like robot.

Pegasus policy search

[Policy search: Pegasus [PPT]]

Actually use on Ng's autonomous helicopter flight for many years.

Pegasus: Policy Evaluation of Gradient And Search Using Scenarios.

Scenarios. Fixed random numbers (random seq's). generated from random number generator.

* Main idea: Evaluate many times/scenarios on policy, (for opt. a deterministic function). Then average.

∴ * Scale well even to fairly large problems. (high-dim. statespace)

* Key in RL: Sequential decision making (consider long-term consequences).

Other e.g.:

medical decision making: seq. of treatments

[Queues] { bank : multiple queues (wait time loss)
assembly line : objects in queues

financial decision making: sell off stocks.

[OR problems] factory automation: opt. throughput / cost.

R.L. Applications

Little Dog robot: by Ziko Coulter (TA) & Peter Abiel (PhD).

(run similar to: approx. value func.)

Legged wheeled robot: Wheeled robots: very fuel-efficient (cars, trucks)

by Lockheed Martin Cooperation.

Helicopter: by Peter Abiel & Adam Coates.

Machine Learning widely used in:

Industry management.
Optimize computer architecture.
Network security.
Robotics
Computer vision
Computational biology.
Aerospace
Natural Language Understanding (NLP)..
....

Choose AI classes

Stanford has one of the best & broadest sets of AI classes.
Learn more about AI, other fields which often apply learning algorithms to problems.

- CS221. Overview of AI (by Ng)
- ☆ CS228. Probabilistic models in AI (by Daphne Koller).
  closest in spirit to 229.
  HIGHLY RECOMMENDED (so as to Ng's PhDs).
- EE364. Convex Optimization. (by Stephen Boyd).
- CS294. Project Course. (by Ng)

[END]