Lecture 23: Reinforcement Learning II

RL is black-box optimization for stochastic optimal control

Last time Policy Gradient methods

$u = \pi_\alpha(x)$ e.g. $u = -Kx$

α ↑ parameter vector $u = DNN(x)$

$u = \pi_\alpha(y)$ ← outputs

"output feedback"

$\dot{x} = f(x,u)$, $y = h(x,u)$

Trial-and-error

Too many concerns:

1) sample complexity

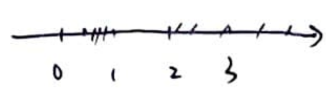
2) Non-convexity

$\dot{x} = Ax + Bu$ $A = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$y = Cx$

$u = -Ky$ $C = [1 \ 1 \ 3]$

stable K



set of good controllers are disconnected

tasks diverse enough that bad local minimum disappeared, domain randomization

Today - Q-Learning

- Actor-Critic

Idea: Learning Value Functions

Policy Gradient $\longleftrightarrow$ Dynamic Programming

inefficient — efficient

evaluate your dynamics exactly once from the terminal back to $t=0$

Learn $f(x,u)$	Learn $\pi(x)$	Learn $\hat{J}(x)$
system i.d. + model-based control	Policy search	cost-to-go

$J^*(x)$, $J^\pi(x)$

$$J(x) = E\left[\sum_{n=0}^{\infty} \gamma^n g(x[n], u[n])\right] \qquad x[n+1] = f(x[n], u[n], w[n])$$

int. horizon

discount factor $0 \le \gamma \le 1$

Learn only $\hat{J}^*(x)$

depends on knowing $f(x,u)$

$$J^*(x) = \min_u g(x,u) + \gamma E\left[\hat{J}^*(f(x,u))\right]$$

$$\pi^*(x) = \arg\min_u [\qquad]$$

Two work-arounds

1) Q-learning $Q(x,u)$ instead of $J(x)$

2) Actor-critic Learn $\pi(x)$ and $J^*(x)$ simultaneously

Q function

Define $Q^\pi(x,u) = E[g(x,u) + J^\pi(f(x,u))]$

$J^*(x) = \min_u Q^\pi(x,u)$

J has dim(x) inputs (scalar ~~input~~ output)

Q has dim(x) + dim(u) inputs (scalar output)

$\pi^*(x) = \arg\min_u Q^*(x,u)$

Q is deep neural network DNN

QT-Opt says runtime optimization of $\hat{Q}^*(x,u)$

$Q^*(x,u)$ can be difficult to model

google arm farm

Actor-Critic

Policy gradient

$[g(\alpha+\beta) - b]\beta$

$$\Delta\alpha = -\eta\,[J(\overset{x}{\alpha}+\beta) - b]\left(\frac{\partial}{\partial\alpha}\log f_{x|\alpha}(x|\alpha)\right)^T$$

one time experiment

Idea: use $J^\pi(\overset{x}{\alpha})$ as the baseline

policy gradient from REINFORCE

Expected value

(better baseline than running an extra trail)

reestimate $\pi(x)$

→ update $J^\pi(x)$

Toddler

feedback, linear comb. nonlinear basis

dead beat controller, return every cycle

batch update does not change mean, but reduce variance

if reset policy, keep value function, converge to policy fast

online learning only