Machine Learning for Systems and Control

5SC28

Lecture 7

dr. ir. Maarten Schoukens & dr. ir. Roland Tóth

Control Systems Group

Department of Electrical Engineering

Eindhoven University of Technology

Academic Year: 2020-2021 (version 1.0)

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Past Lectures

Introduction to Reinforcement Learning

Temporal Difference Learning

Value-based learning: Tabular Q-Learning

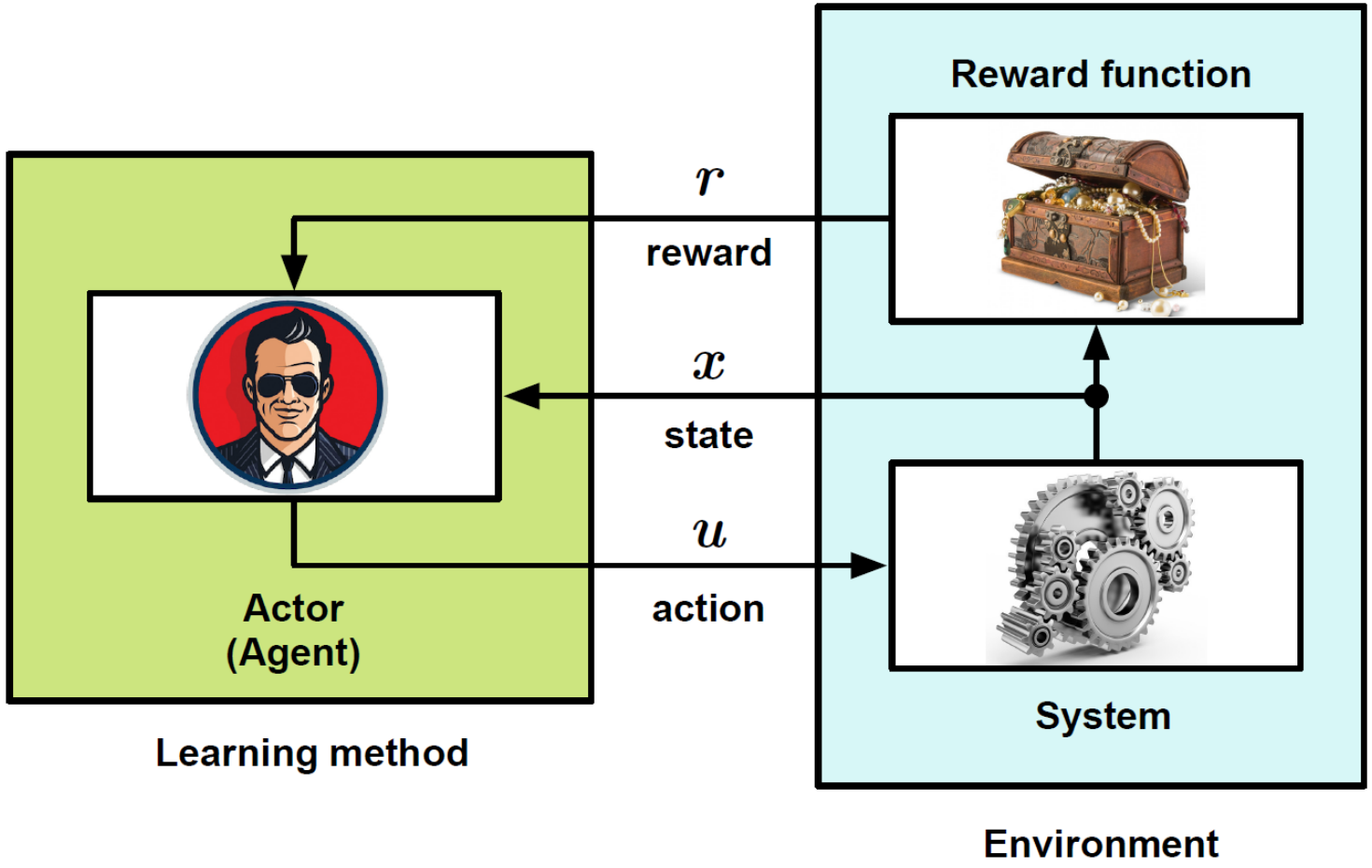
Approximate Q-Learning

DQN

Learning Goals

- **Explain, discuss, compare and interpret** the main techniques and theory for reinforcement learning based control starting from classical Q-learning up to **actor-critic** and model internalization-based methods;
- **Recommend and evaluate** a machine learning method for a real-life application in a systems and **control setting**.
- **Implement, tailor, and apply** the Gaussian process, (deep) neural network and **reinforcement learning techniques** for model and control learning on real-world systems (e.g. on an inverted pendulum laboratory setup).

Action-Value Reinforcement Learning



Agent learns Q_* or Q_π to improve its actions

Action-Value Reinforcement Learning

Concept: Learn the value of actions through $Q_*(x,u)$ or $Q_\pi(x,u)$

Strategy:

Training & Data ➔ θ ➔ Q-function ➔ Policy π

Action-Value Reinforcement Learning

Concept: Learn the value of actions through $Q_*(x,u)$ or $Q_\pi(x,u)$

Strategy:

Training & Data ➔ θ ➔ Q-function ➔ Policy π

Disadvantages: Finite action space to solve $\pi_*(x) = \arg\max_u Q_*(x,u)$

Poor exploration in case of large action space (slow)

Policy Gradient Reinforcement Learning

Concept: Learn the policy that ensures maximum rewards for the actions taken

Strategy: Training & Data → θ → Policy π

Policy Gradient Reinforcement Learning

Concept: Learn the policy that ensures maximum rewards for the actions taken

Strategy: Training & Data ➔ θ ➔ Policy π

Main Advantages: No need to solve $\pi_*(x) = \arg\max_u Q_*(x, u)$

No need to discretize the action space

Can handle stochastic policies

Increased algorithm stability

(Dis)advantage: More complicated algorithm, more options to tune

Policy Gradient Methods

Stochastic Policy Representation

Stochastic policy: Learn a function that gives the probability distribution over actions from the current state:

$$\pi_\theta(x, u) \approx P(u|x)$$

parameters state action

Probabilistic policies can be optimal in a probabilistic or a partially observed environment (e.g. card games)

Stochastic Policy Representation

Stochastic policy: Learn a function that gives the probability distribution over actions from the current state:

$$\pi_\theta(x, u) \approx P(u|x)$$

Discrete actions: softmax policy

$$\pi_\theta(x, u) = \frac{\exp\left(f_\theta(x, u)\right)}{\sum_{u'} \exp\left(f_\theta(x, u')\right)}$$

Continuous actions: Gaussian policy

$$\pi_\theta(x, u) = \frac{1}{\sqrt{2\pi}\sigma_\theta(x)} \exp\left(-\frac{\left(u - f_\theta\left(x\right)\right)^2}{2\sigma_\theta(x)^2}\right)$$

Policy Gradient: Optimization

Objective function: $J(\theta, x_k) = V_\pi(x_k)$

Interpretation: *J* is the expected cumulative reward (via the value function) using the policy π.

Policy Gradient: Optimization

Objective function: $J(\theta, x_k) = V_\pi(x_k)$

Interpretation: *J* is the expected cumulative reward (via the value function) using the policy π.

Solution: Stochastic Gradient **Ascent** (SGA)

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta_k, x_k)$$

Where α is the step size. Replay can be used to assist convergence.

Policy Gradient: Optimization

Objective function: $J(\theta, x_k) = V_\pi(x_k)$

Interpretation: *J* is the expected cumulative reward (via the value

How to calculate the value gradient to arrive to a gradient over the policy?

Solution: Stochastic Gradient Ascent (SGA)

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta_k, x_k)$$

Where α is the step size. Replay can be used to assist convergence.

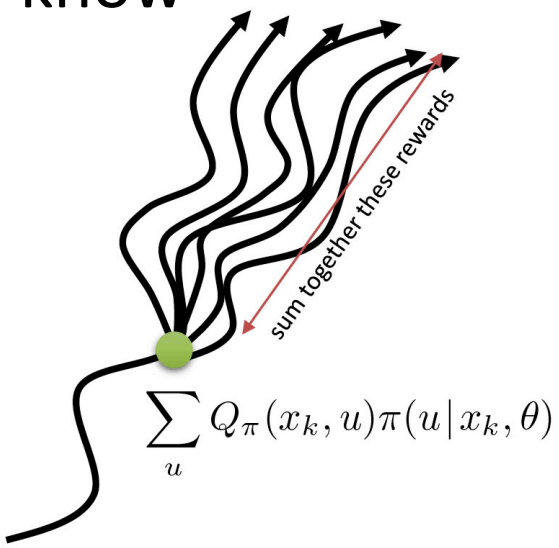
Policy Gradient Theorem

Assumptions:

Assume U is a discrete action space (for simplicity)

Probability that action u is taken for a given x,θ is nonzero over U

We know



$$V_\pi(x_k) = \sum_u Q_\pi(x_k, u)\pi(u|x_k, \theta)$$

i.e., we can characterize the effect of the current action made by the policy over the future based on Q_π

Policy Gradient Theorem

We can show in terms of the policy gradient theorem[1,2]:

$$\nabla_\theta J(\theta, x_k) \propto \sum_u Q_\pi(x_k, u) \nabla_\theta \pi(u|x_k, \theta)$$

Equivalent reformulation:

$$\nabla_\theta J(\theta, x_k) \propto \sum_u Q_\pi(x_k, u) \pi(u|x_k, \theta) \frac{\nabla_\theta \pi(u|x_k, \theta)}{\pi(u|x_k, \theta)}$$

Or:

$$\nabla_\theta J(\theta, x_k) \propto \sum_u Q_\pi(x_k, u) \pi(u|x_k, \theta) \nabla_\theta \ln \pi(u|x_k, \theta)$$

1 R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction.” A Bradford Book, pp. 356, 2008.

2 V. Konda and J. Tsitsiklis, “Actor-Critic Algorithms.” SIAM Journal on Control and Optimization, pp. 1008-1014, 2000.

Policy Gradient Algorithm Backbone

$$\nabla_\theta J(\theta, x_k) \propto \sum_u Q_\pi(x_k, u)\pi(u|x_k, \theta)\nabla_\theta \ln \pi(u|x_k, \theta)$$

Replace *u* with the sample as the current action taken

$$u_k \sim \pi(x_k, \theta)$$

$$\nabla_\theta J(\theta, x_k) \propto \mathbb{E}_\pi \{Q_\pi(x_k, u_k)\nabla_\theta \ln \pi(u_k|x_k, \theta)\}$$

$$\mathbb{E}_\pi\{G_k \mid x_k, u_k\} = Q_\pi(x_k, u_k)$$

$$\nabla_\theta J(\theta, x_k) \propto \mathbb{E}_\pi \{G_k\nabla_\theta \ln \pi(u_k|x_k, \theta)\}$$

Policy Gradient Algorithm Backbone

$$\nabla_\theta J(\theta, x_k) \propto \sum_u Q_\pi(x_k, u)\pi(u|x_k, \theta)\nabla_\theta \ln \pi(u|x_k, \theta)$$

Replace *u* with the sample as the current action taken

$$u_k \sim \pi(x_k, \theta)$$

Implement in REINFORCE algorithm

$$\mathbb{E}_\pi\{G_k \mid x_k, u_k\} = Q_\pi(x_k, u_k)$$

$$\nabla_\theta J(\theta, x_k) \propto \mathbb{E}_\pi \{G_k \nabla_\theta \ln \pi(u_k|x_k, \theta)\}$$

REINFORCE Algorithm

Input: a differentiable policy parameterization $\pi(u|x,\theta)$
Parameters: step size $\alpha \in (0,1]$
Initialize: policy parameters θ

for each episode **do**

Generate an episode $x_0, u_0, r_1, \ldots, x_{T-1}, u_{T-1}, r_T$

repeat for episode step $t = 0, 1, \ldots, T$

$\hat{G}_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$

$\theta \leftarrow \theta + \alpha \gamma^t \hat{G}_t \nabla_\theta \ln \pi(u_t \,|\, x_t, \theta)$

until end episode

end for

REINFORCE Algorithm

Input: a differentiable policy parameterization $\pi(u|x,\theta)$
Parameters: step size $\alpha \in (0,1]$
Initialize: policy parameters θ

for each episode **do** Episodic algorithm! Complete full episode for one parameter update

Generate an episode $x_0, u_0, r_1, \ldots, x_{T-1}, u_{T-1}, r_T$

repeat for episode step $t = 0, 1, \ldots, T$

$\hat{G}_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$ Episodic nature to obtain estimate of G

$\theta \leftarrow \theta + \alpha\gamma^t \hat{G}_t \nabla_\theta \ln \pi(u_t|x_t,\theta)$

until end episode

end for

Types of RL Control

By path to optimal solution

- Off-policy – find optimal policy regardless of the agent's motivation
- On-policy – improve the policy that is used to make decisions

By level of interaction with the process

- Online – learn by interacting with the process
- Offline – data collected in advance (Monte-Carlo methods)

By model knowledge

- Model-free – no p , only transition data (e.g. standard Q-Learning RL)
- Model-based – p is known (e.g. Dynamic Programming)
- Model-learning – estimate p from transition data

REINFORCE + baseline algorithm

$$\theta_{k+1} = \theta_k + \alpha(\hat{G}_k - b(x_k))\nabla_\theta \ln \pi(u_k \,|\, x_k, \theta_k)$$

REINFORCE + baseline algorithm

$$\theta_{k+1} = \theta_k + \alpha(\hat{G}_k - b(x_k))\nabla_\theta \ln \pi(u_k | x_k, \theta_k)$$

Baseline: We can introduce arbitrary baseline *b(x)* without consequence*:

$$\nabla_\theta J(\theta, x_k) = \mathbb{E}_\pi\{(G_k - b(x_k))\nabla_\theta \pi(u_k | x_k, \theta)\}$$

Explanation: baseline *does not introduce bias* in gradient

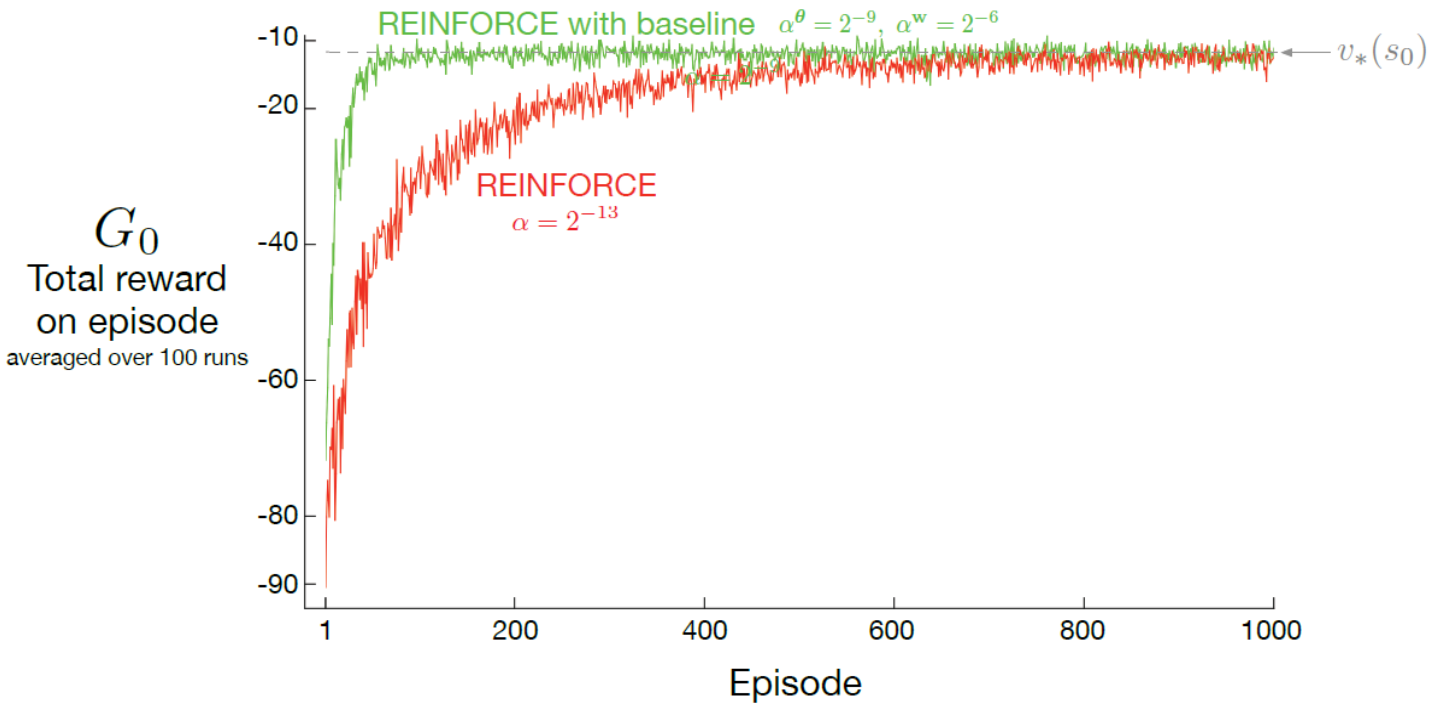
$$\sum_u b(x_k)\nabla_\theta \pi(u | x_k, \theta) = b(x_k)\nabla_\theta \sum_u \pi(u | x_k, \theta) = b(x_k)\nabla_\theta 1 = 0$$

But *reduces variance* of the gradient estimate in practice if

$$G_k \approx b(x_k)$$

* As long as *b* does not depend on *u*

REINFORCE + baseline algorithm



Source: Richard S. Sutton and Andrew G. Barto, Reinforcement Learning An Introduction, second edition, Figure 13.2

Actor-Critic Methods

Why Actor-Critic?

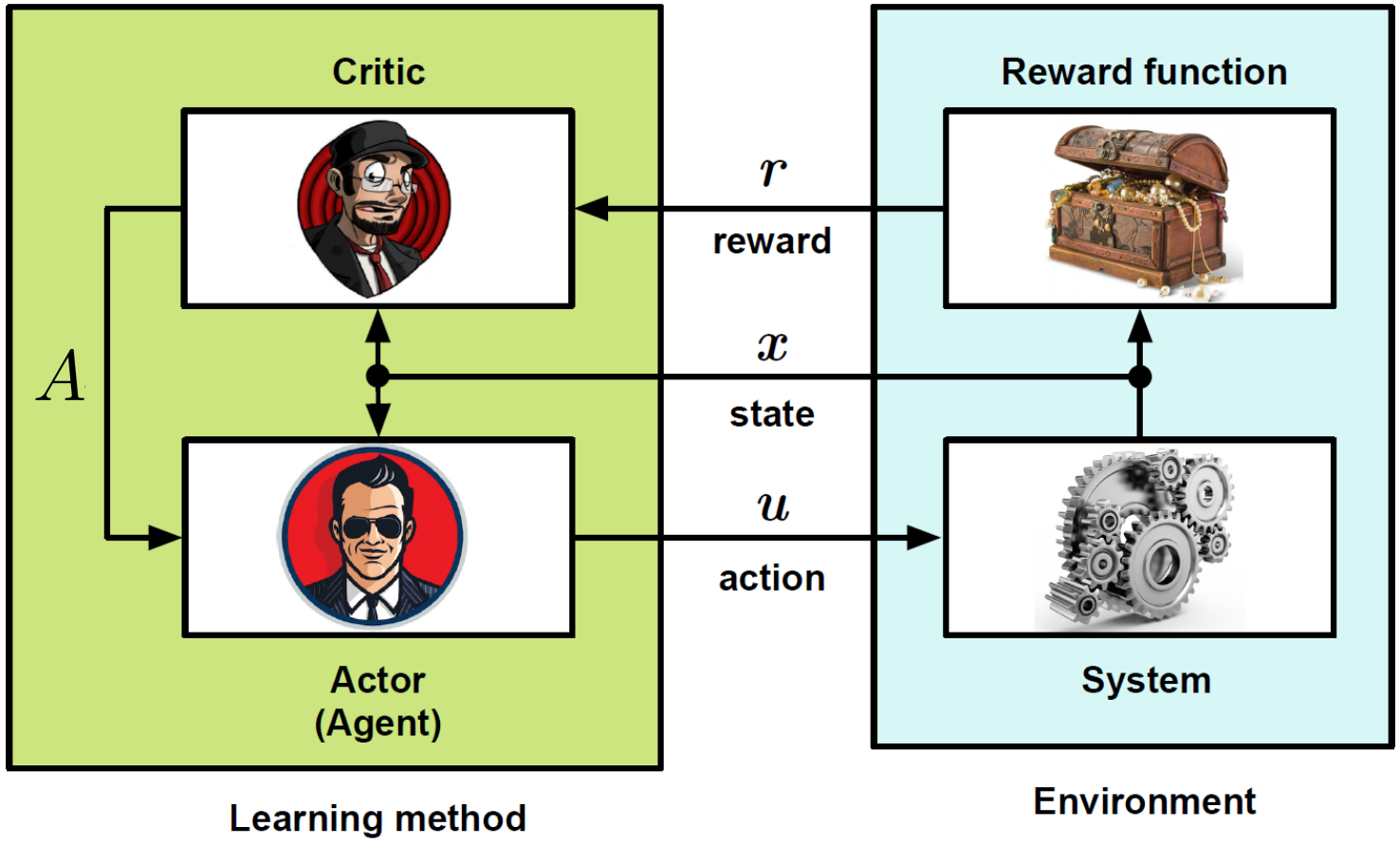
Problem: REINFORCE uses episode learning to estimate G_k

Idea: Introduce value function (cfr. Lecture 5 and 6)

$$\mathbb{E}_\pi\{G_k \mid x_k\} = \mathbb{E}_\pi\{r_k + V_\pi(x_{k+1})\} = Q_\pi(x_k, u_k)$$

Estimate using temporal difference approach (Lecture 6)

Action-Critic Reinforcement Learning



Strategy: the **critic** aims to learn V_π while the **actor** tries to improve π

Actor-Critic Overview

Explicitly separated value function and policy

Critic = state value function $V_\pi(x)$

Actor = control policy $\pi(x)$

Problem setting:

Continuous synchronous learning

Continuous action and state space

Deterministic state transition and reward function

Actor-Critic Parameterization

Critic: $\hat{V}_\pi(x, \eta) = \Psi(x)\eta, \quad \Psi : X \to \mathbb{R}^{n_\eta}$

basis functions/features — parameter vector

Actor: $\hat{\pi}_{\mathrm{d}}(x, \theta) = \Phi(x)\theta, \quad \Phi : X \to \mathbb{R}^{n_\theta}$

Alternatively, they can be described via GP, DNN, etc.

Don’t forget: *probabilistic policies* (softmax, Gaussian, ε-greedy)

On-Policy Advantage Based Critic

Objective Critic: learn $V_\pi(x)$ for the current policy π

Step 1: Use the Bellman equation for $V_\pi(x_k)$

$$V_\pi(x_k) = \mathbb{E}_\pi\Big\{\rho(x_k, \pi(x_k, \theta_k)) + \gamma V_\pi(x_{k+1}) \mid x_k\Big\}$$

$$= \mathbb{E}_\pi\Big\{\rho(x_k, u_k) + \gamma V_\pi(x_{k+1}) \mid x_k\Big\}$$

$$\approx r_{k+1} + \gamma V_\pi(x_{k+1})$$

Sample

On-Policy Advantage Based Critic

Objective Critic: learn $V_\pi(x)$ for the current policy π

Step 1: Use the Bellman equation for $V_\pi(x_k)$

$$V_\pi(x_k) = \mathbb{E}_\pi\left\{\rho(x_k, \pi(x_k, \theta_k)) + \gamma V_\pi(x_{k+1}) \mid x_k\right\} \approx r_{k+1} + \gamma V_\pi(x_{k+1})$$

Step 2: Compute *advantage* using temporal difference

$$A_k = Q_\pi(x_k, u_k) - V_\pi(x_k)$$

use the transition sample

$$A_k = \mathbb{E}_\pi\left\{r_{k+1} + \gamma V_\pi(x_{k+1}) \middle| x_k, u_k\right\} - V_\pi(x_k)$$

$$A_k = r_{k+1} + \gamma V_\pi(x_{k+1}) - V_\pi(x_k)$$

$$\hat{A}_k = r_{k+1} + \gamma \hat{V}_\pi(x_{k+1}, \eta_k) - \hat{V}_\pi(x_k, \eta_k)$$

Bootstrap

On-Policy Advantage Based Critic

Step 3: Minimize prediction error of the current estimate $\hat{V}_\pi$

cost: $L_{VF} = \sum_{x \in \hat{X}} (V_\pi(x) - \hat{V}_\pi(x))^2$

set of observations (current sample, replay, entire batch, …)

On-Policy Advantage Based Critic

Step 3: Minimize prediction error of the current estimate $\hat{V}_\pi$

$$\text{cost: } L_{VF} = \sum_{x \in \hat{X}} \left(V_\pi(x) - \hat{V}_\pi(x)\right)^2$$

set of observations (current sample, replay, entire batch, …)

Take $\hat{X} = \{x_k\}$, SGD gives:

$$\eta_{k+1} = \eta_k + \alpha_c \hat{A}_k \nabla_\eta \hat{V}_\pi(x_k, \eta_k)$$

learning rate

On-Policy Advantage Based Critic

Step 3: Minimize prediction error of the current estimate $\hat{V}_\pi$

$$\text{cost: } L_{VF} = \sum_{x \in \hat{X}} \left(V_\pi(x) - \hat{V}_\pi(x)\right)^2$$

set of observations (current sample, replay, entire batch, …)

Take $\hat{X} = \{x_k\}$, SGD gives:

$$\eta_{k+1} = \eta_k + \alpha_c \hat{A}_k \nabla_\eta \hat{V}_\pi(x_k, \eta_k)$$

learning rate

See also the prediction problem in Lecture 6!

On-Policy Advantage Based Critic

$$\eta_{k+1} = \eta_k + \alpha_c \hat{A}_k \nabla_\eta \hat{V}_\pi(x_k, \eta_k)$$

$\hat{A}_k > 0 \qquad r_{k+1} + \gamma \hat{V}_\pi(x_{k+1}, \eta_k) > \hat{V}_\pi(x_k, \eta_k)$

Old estimate is too low → increase it (too negative criticism in the past)

$\hat{A}_k < 0 \qquad r_{k+1} + \gamma \hat{V}_\pi(x_{k+1}, \eta_k) < \hat{V}_\pi(x_k, \eta_k)$

Old estimate is too high → decrease it (too enthusiastic opinion in the past)

On-Policy Advantage Based Actor

Remember REINFORCE + baseline:

$$\theta_{k+1} = \theta_k + \alpha(\hat{G}_k - b(x_k))\nabla_\theta \ln \pi(u_k | x_k, \theta_k)$$

Step 1 Use Bellman equation + transition sample + critic-based estimate of V_π

$$\mathbb{E}_\pi\{G_k \mid x_k\} = \mathbb{E}_\pi\{r_{k+1} + V_\pi(x_{k+1})\} \Rightarrow \hat{G}_k = r_{k+1} + \hat{V}_\pi(x_{k+1})$$

Step 2 Use the baseline $b(x_k) = \hat{V}_\pi(x_k)$

$$\hat{G}_k - b(x_k) = r_{k+1} + \hat{V}_\pi(x_{k+1}) - \hat{V}_\pi(x_k) = \hat{A}_k$$

Advantage

On-Policy Advantage Based Actor

Step 3 Achieve maximization of $J(\theta, x_k)$ via SGA

$$\theta_{k+1} = \theta_k + \alpha_a \hat{A}_k \nabla_\theta \ln \pi(u_k \,|\, x_k, \theta_k)$$

learning rate

Probabilistic policy

Remember: $$J(\theta, x_k) = V_\pi(x_k)$$

On-Policy Advantage Based Actor

Step 4 Gradient calculation $\nabla_\theta \ln \pi(u_k | x_k, \theta_k)$

Example: choose Gaussian policy $\pi_\theta(x,u) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(u - f_\theta(x))^2}{2\sigma^2}\right)$

$$\nabla_\theta \ln \pi(u_k | x_k, \theta) = \nabla_\theta \ln \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(u_k - f_\theta(x_k))^2}{2\sigma^2}\right)$$

$$= \nabla_\theta \frac{(u_k - f_\theta(x_k))^2}{-2\sigma^2}$$

$$= -\frac{1}{\sigma^2}(u_k - f_\theta(x_k))\nabla_\theta f_\theta(x_k)$$

On-Policy Advantage Based Actor

$$\theta_{k+1} = \theta_k + \alpha_a \hat{A}_k \nabla_\theta \ln \pi(u_k \,|\, x_k, \theta_k)$$

$$\hat{A}_k > 0 \qquad r_{k+1} + \gamma \hat{V}_\pi(x_{k+1}, \eta_k) > \hat{V}_\pi(x_k, \eta_k)$$

Action u_k has unforeseen advantage (exploit it more in the future)

$$\hat{A}_k < 0 \qquad r_{k+1} + \gamma \hat{V}_\pi(x_{k+1}, \eta_k) < \hat{V}_\pi(x_k, \eta_k)$$

Action u_k has unforeseen disadvantage (avoid it in the future)

Advantage Actor-Critic (A2C) Algorithm

Input: a differentiable parameterization for $\pi(u|x,\theta)$ and $V_\pi(x,\eta)$
Parameters: step size $\alpha_a, \alpha_c \in (0,1]$
Initialize: actor and critic parameters θ, η

for each episode **do**
  initialise x_0
  repeat for each step k
    obtain u_k based on policy π
    apply u_k, measure x_{k+1}, recive r_{k+1}
    $\hat{A}_k = r_{k+1} + \gamma \hat{V}_\pi(x_{k+1},\eta) - \hat{V}_\pi(x_k,\eta)$ — **advantage**
    $\eta \leftarrow \eta + \alpha_c \hat{A}_k \nabla_\eta \hat{V}_\pi(x_k,\eta)$ — **critic update - minimization**
    $\theta \leftarrow \theta + \alpha_a \hat{A}_k \nabla_\theta \ln \pi(u_k|x_k,\theta)$ — **actor update - maximization**
  until terminal state
end for

Types of RL Control

By path to optimal solution

- Off-policy – find optimal policy regardless of the agent's motivation
- On-policy – improve the policy that is used to make decisions

By level of interaction with the process

- Online – learn by interacting with the process
- Offline – data collected in advance (Monte-Carlo methods)

By model knowledge

- Model-free – no p , only transition data (e.g. standard Q-Learning RL)
- Model-based – p is known (e.g. Dynamic Programming)
- Model-learning – estimate p from transition data

Actor-Critic Properties

Proof of convergence under various assumptions

Improved stability

Probabilistic in nature

On-policy and off-policy versions

Tabular and continuous formulation

Many tricks to improve performance (reward normalization, …)

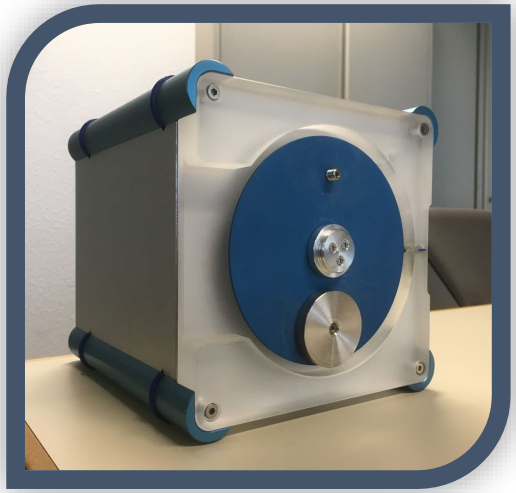
Add entropy term in the cost to improve exploration

Updates can be synchronous, asynchronous or episodic

Many other versions (A3C, PPO, …)

Examples

Unbalanced Disc



- State: $x = \begin{pmatrix} \alpha & \dot{\alpha} \end{pmatrix}$ (angle and velocity)
- Input: $u = \text{voltage}$
- Reward: $r_{k+1} = \rho(x_k, u_k) = -x_k^\top \begin{bmatrix} 5 & 0 \\ 0 & 0.1 \end{bmatrix} x_k - u_k^\top u_k$
- Discount: $\gamma = 0.98$

- Objective: stabilize top position (swing up)
- Insufficient actuation (needs to swing back & forth)

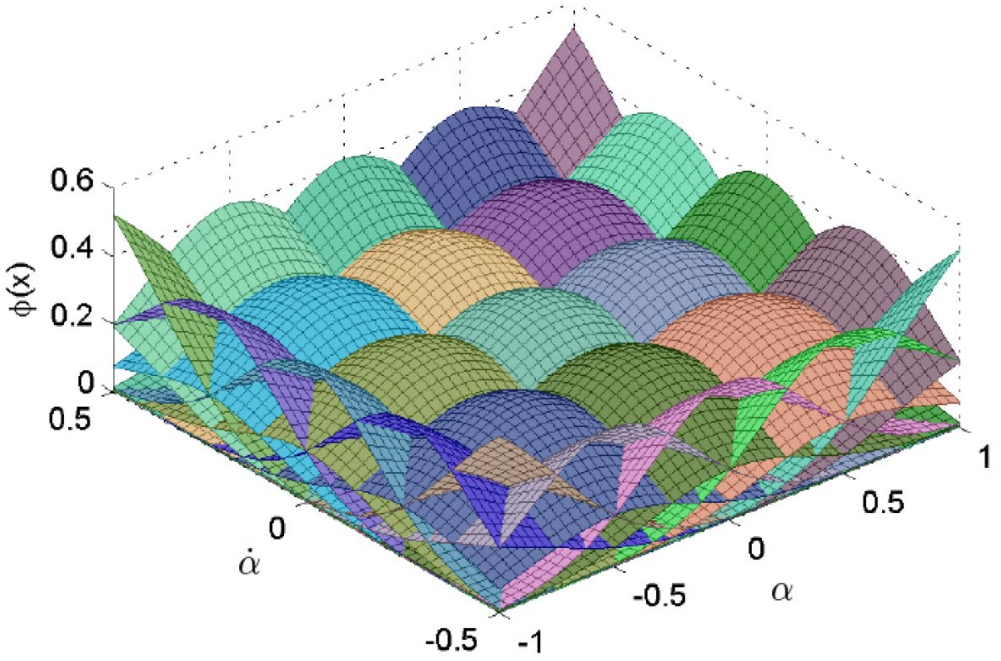
Unbalanced Disc

Parameterization

deterministic part only

Critic: $\hat{V}_\pi(x,\eta) = \Psi(x)\eta$ Actor: $\hat{\pi}_{\mathrm{d}}(x,\theta) = \Phi(x)\theta$

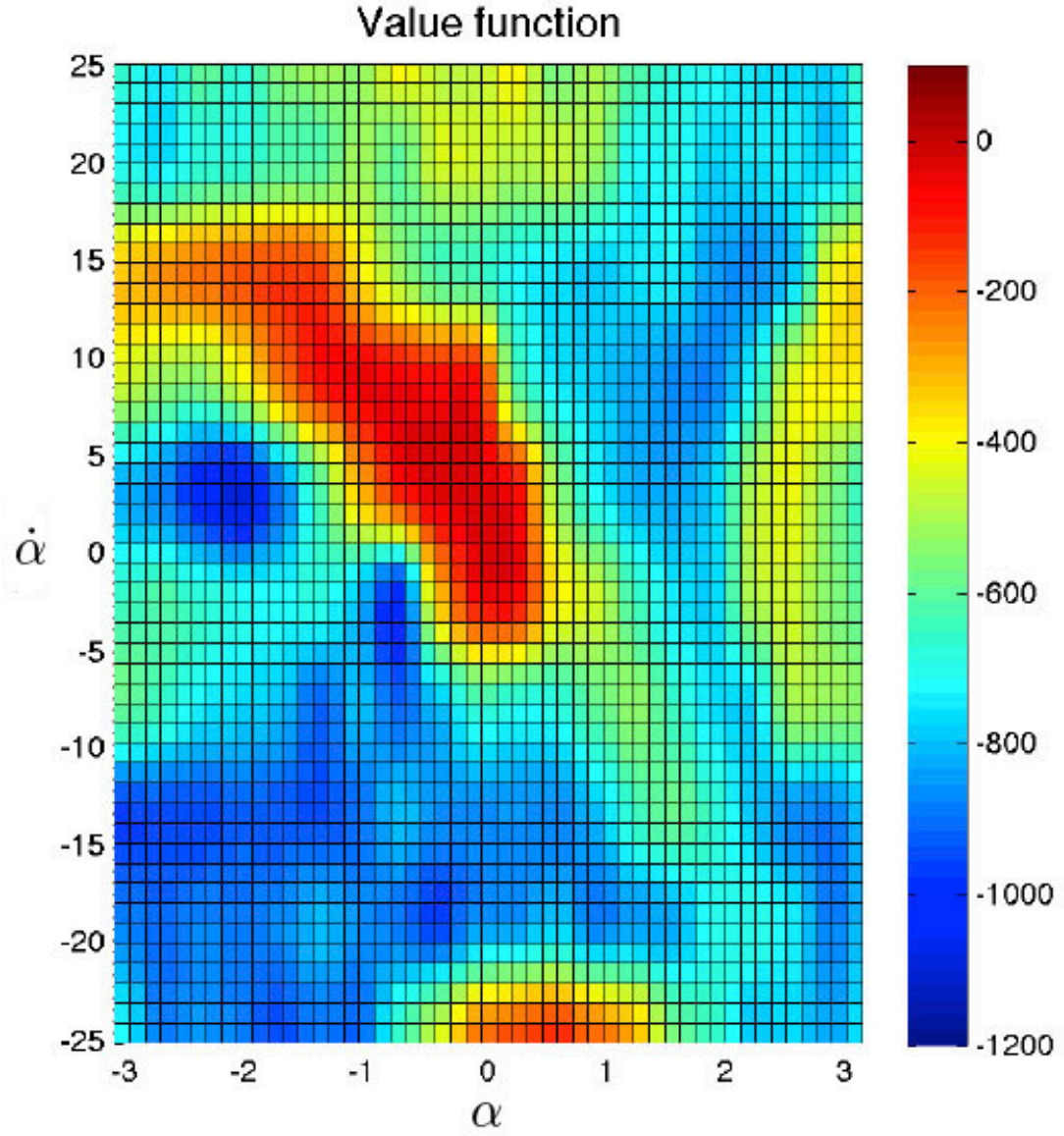
Basis functions are given by normalized RBF with equidistant gridding

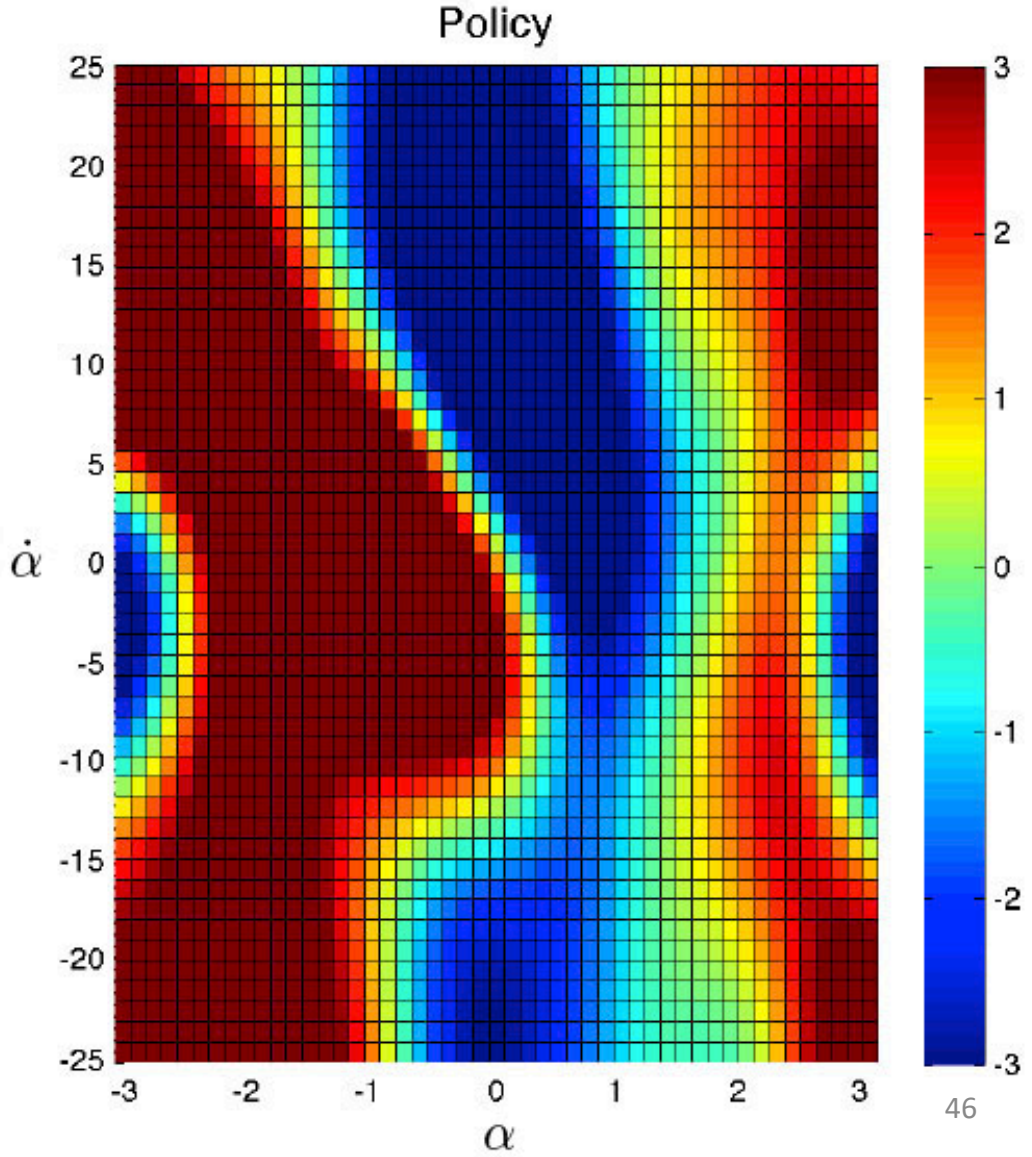


$$\phi_i(x) = \frac{\phi_i(x)}{\sum_{j=1}^{n_\theta} \phi_i(x)}$$

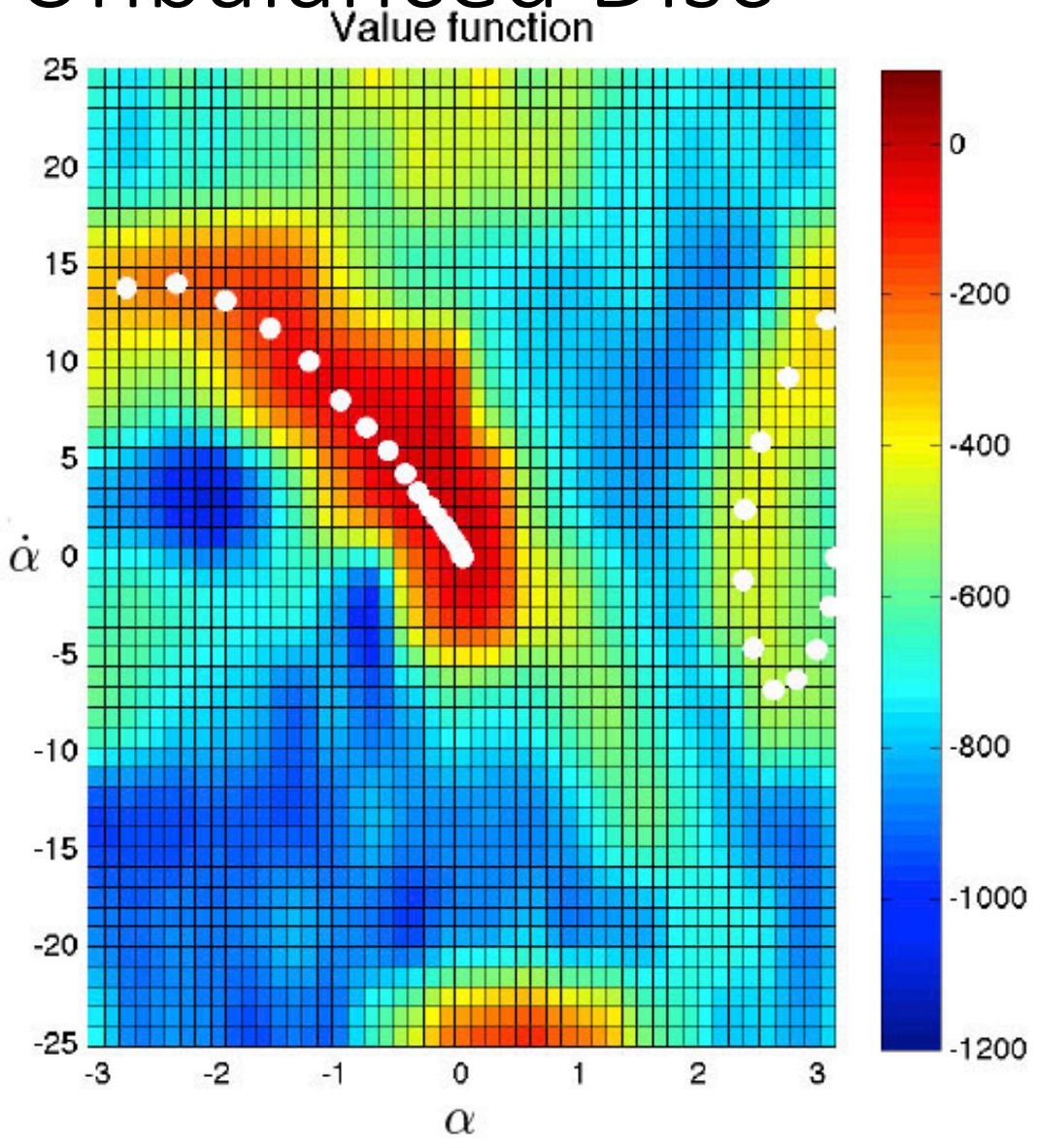
$$\phi_i(x) = \exp\left(-\frac{\|x - c_i\|_2^2}{2\sigma^2}\right)$$

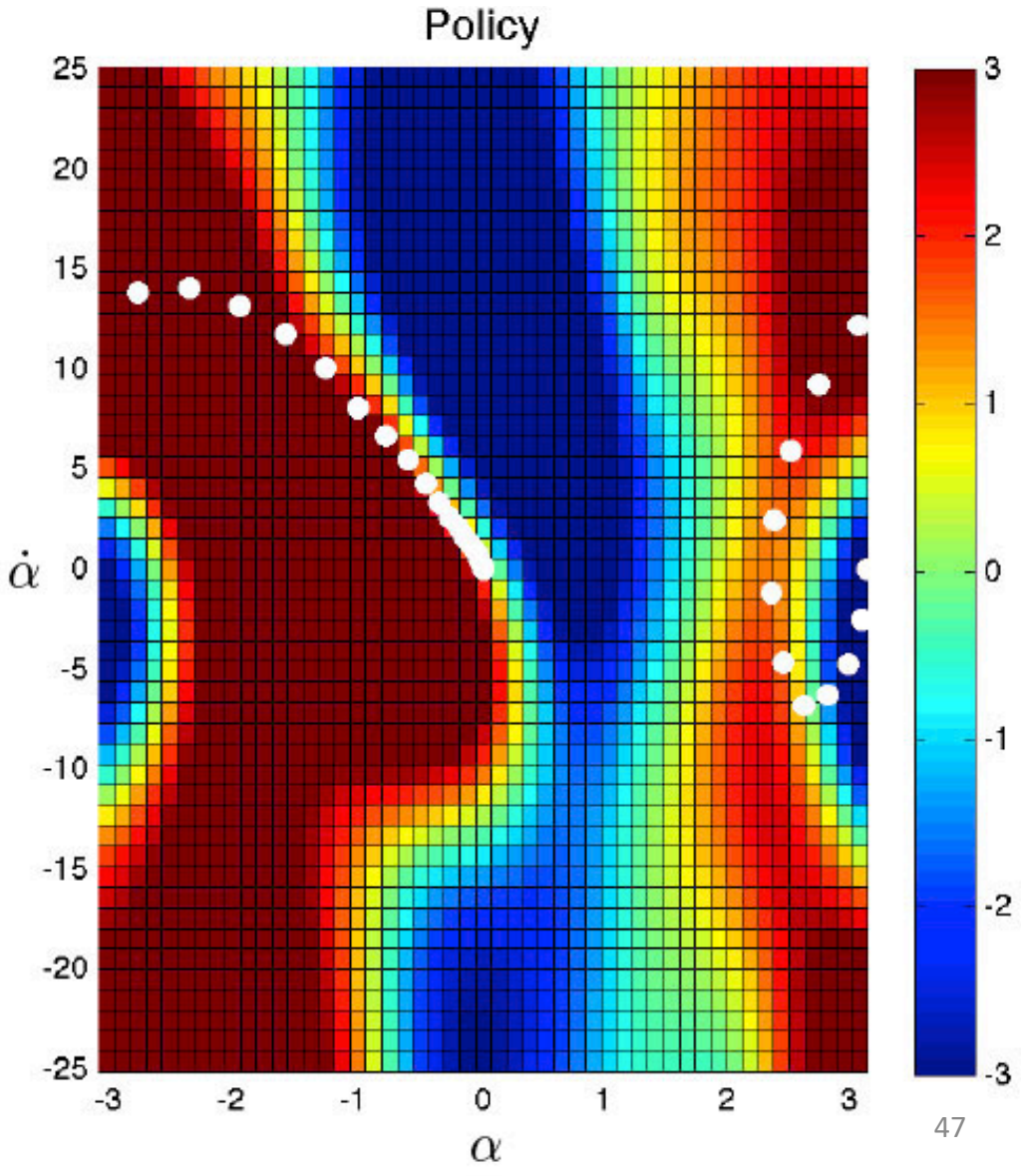
Unbalanced Disc



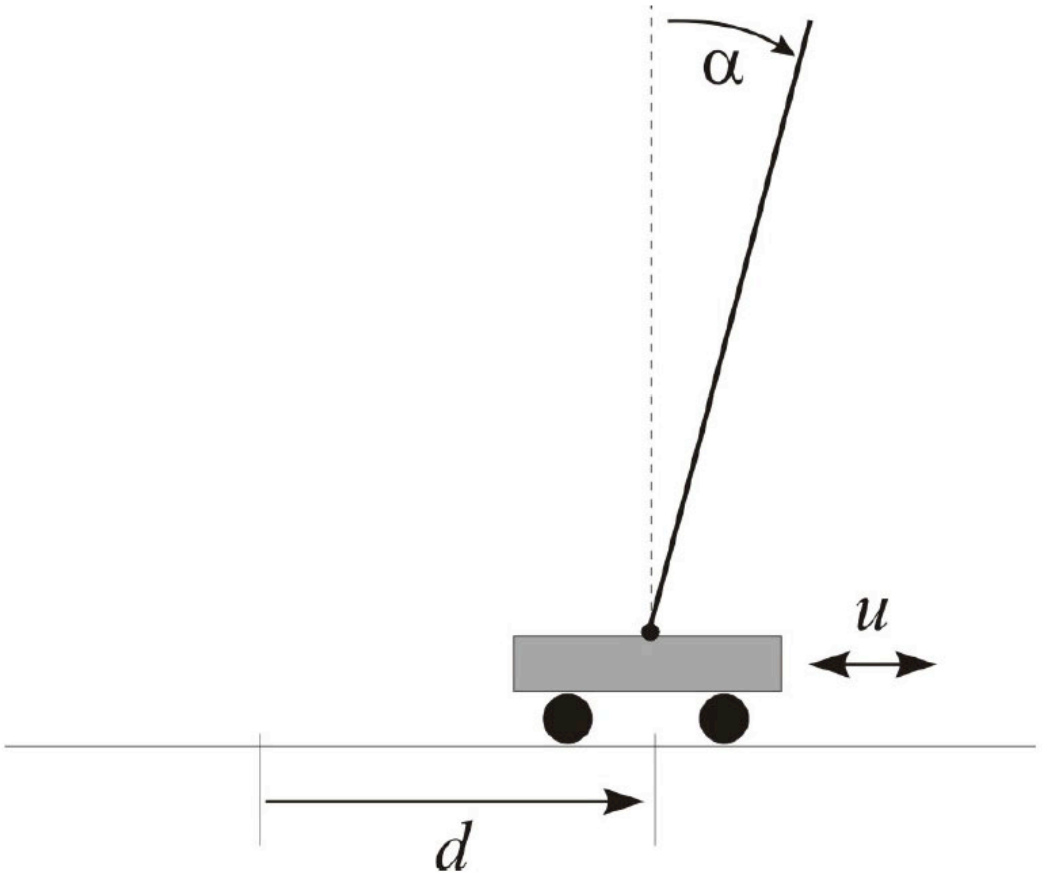


Unbalanced Disc

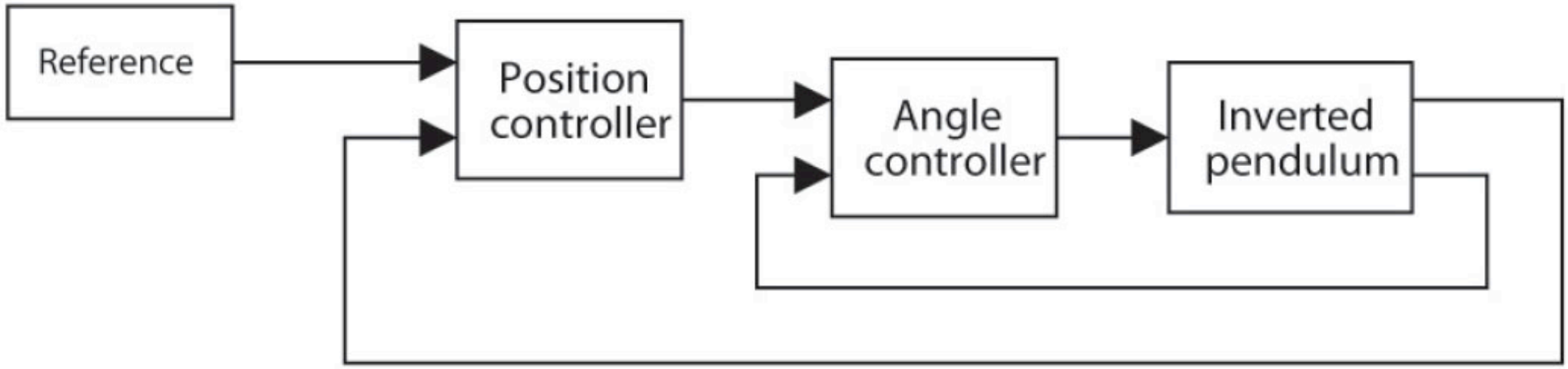




Cart with Pendulum

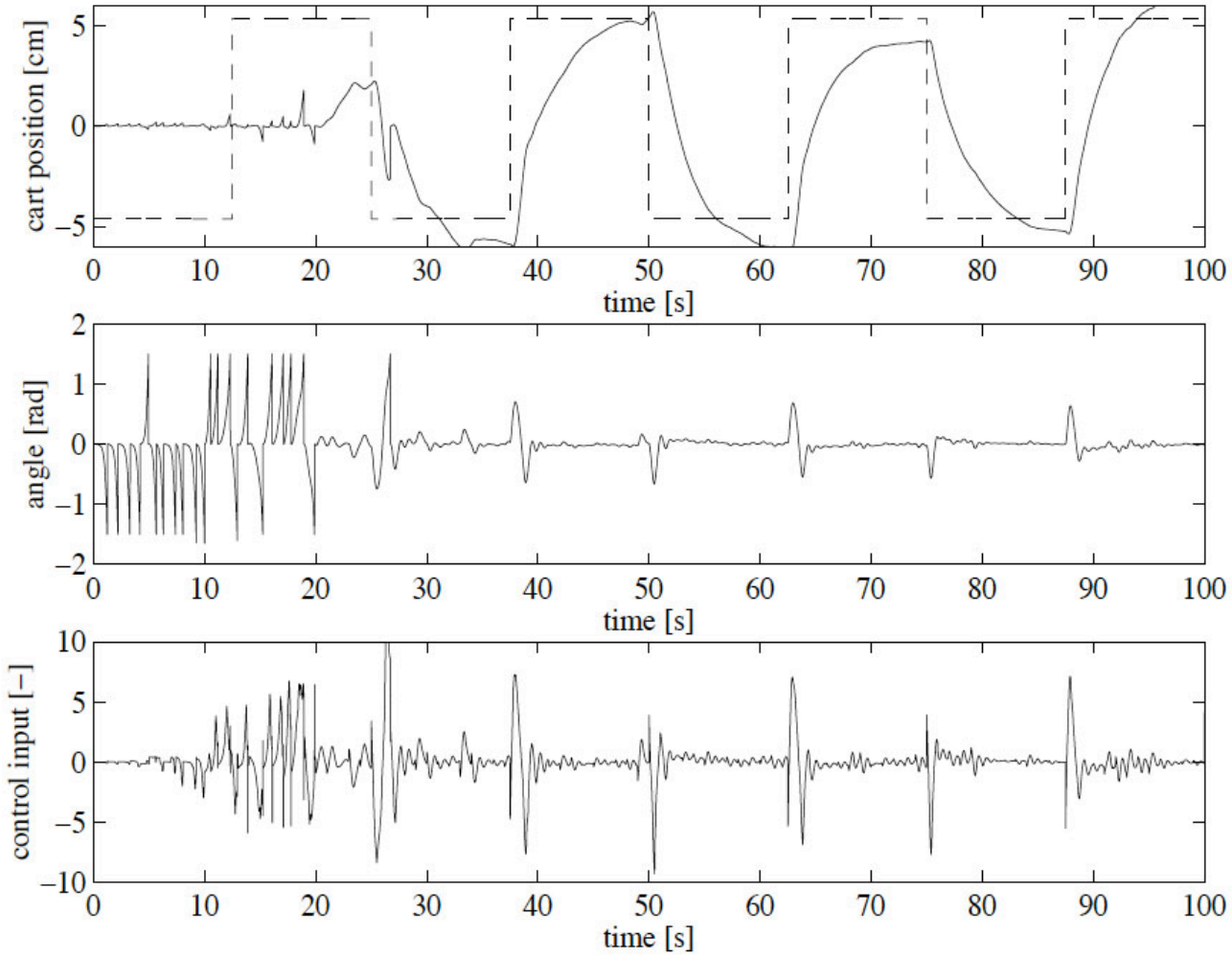


Cart with Pendulum



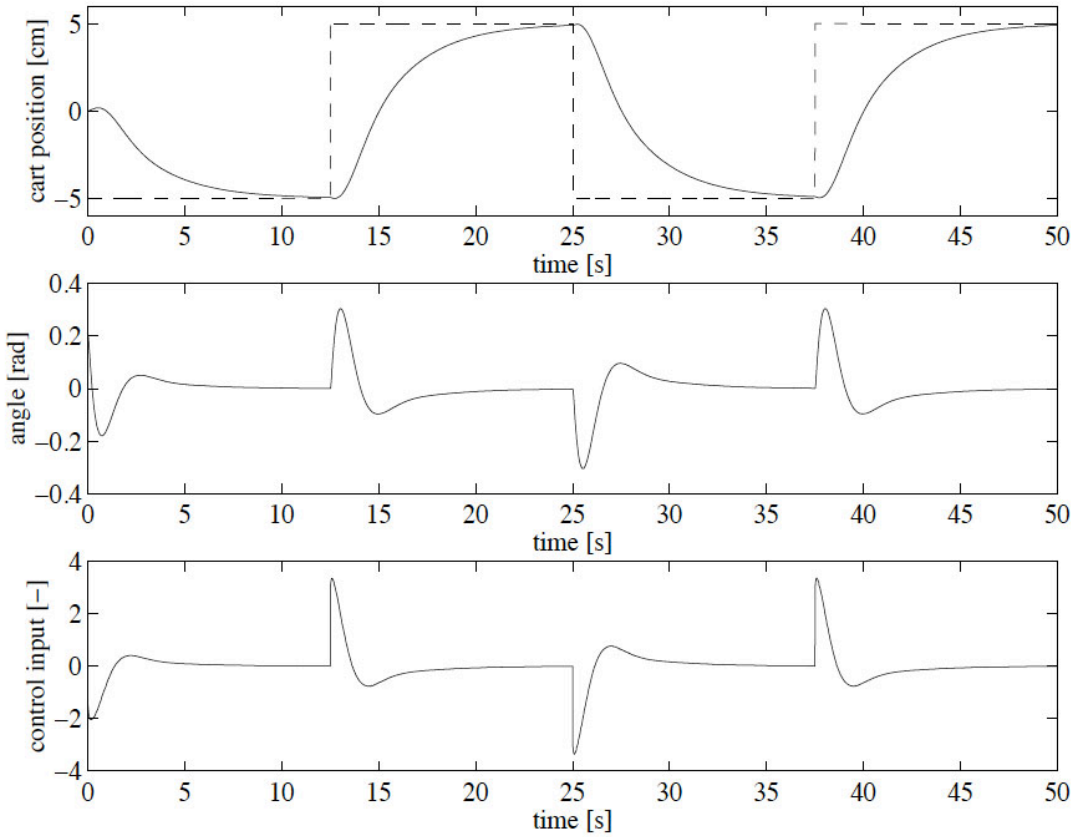
Hierarchical control loop. The position controller is fixed to be a pre-tuned PD controller. The objective is to learn the angle controller in closed loop.

Cart with Pendulum: Learning

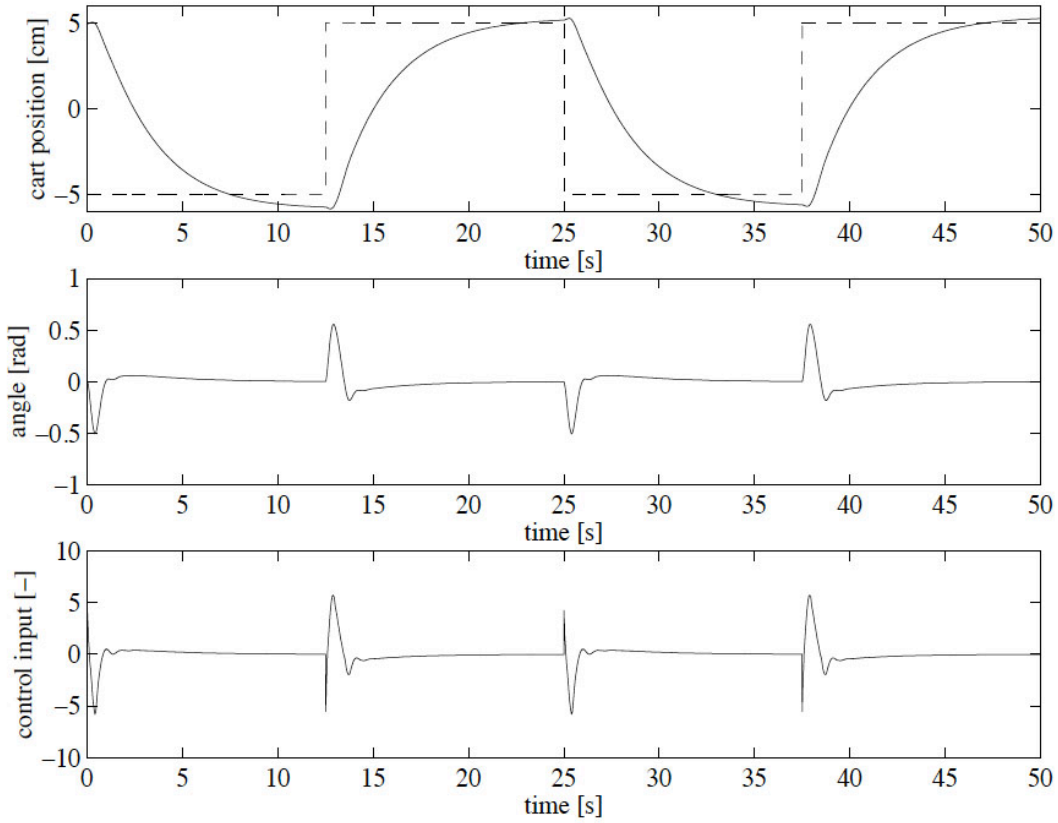


Performance during learning

Cart with Pendulum: RL vs Classical

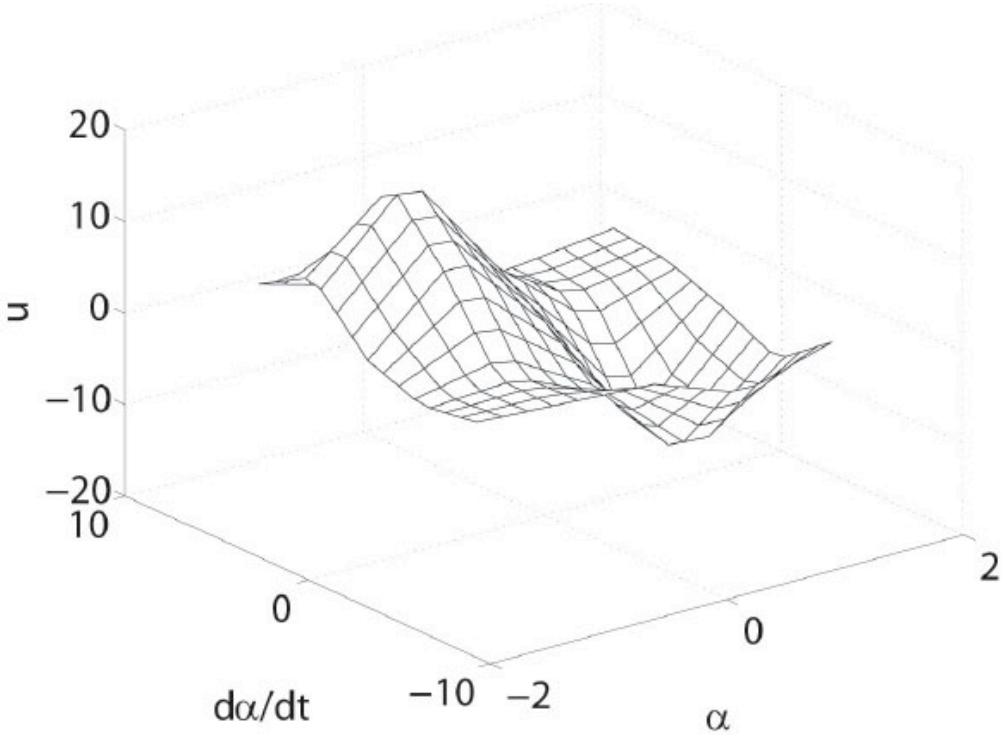


Classical Control Design

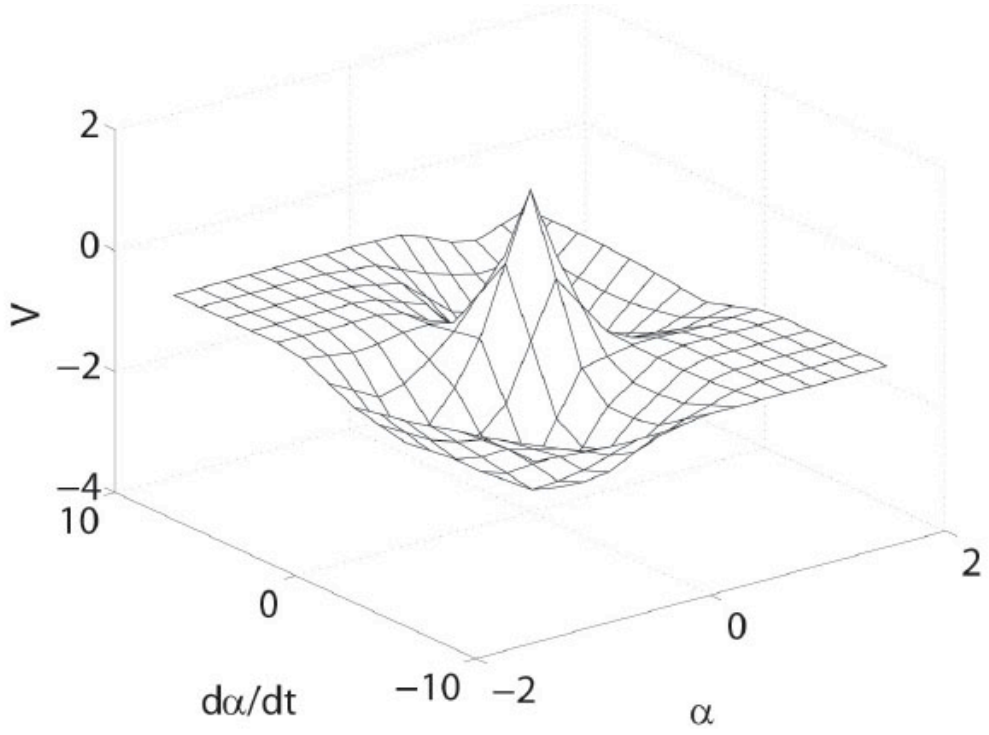


Reinforcement Learning

Cart with Pendulum: Actor and Critic



Actor



Critic

Overview

Introduced policy gradient reinforcement learning

REINFORCE

Actor-Critic (A2C)

Can work probabilistic, handle continuous state *and* action space

Fundamental dilemma remains:

- Efficiency of DP: Models make it possible to plan and synthetize policy, otherwise we only rely on experience. Experiments are costly and risky.
- Efficiency of RL: Experiments allow to explore and improve exploitation on the long run. Models are inherently uncertain.
- How to have a working marriage of DP and RL?