《强化学习与控制》
--
Indirect RL w/ Func Approximation

Shengbo Eben Li
(李升波)

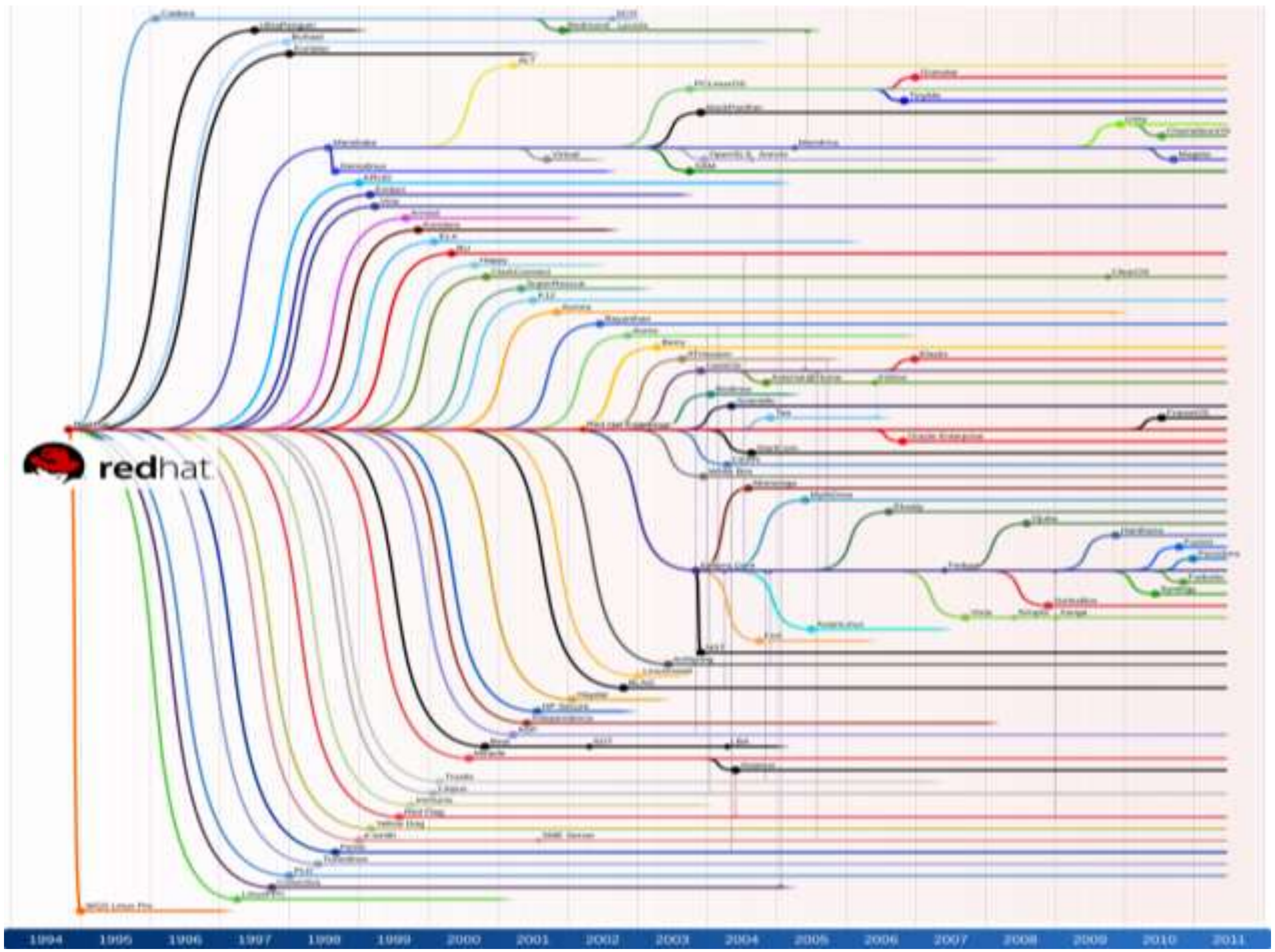
Intelligent Driving Laboratory (iDLab)

Tsinghua University

Idea is cheap. Implement it!

Knowing is not enough, we must apply.
Willing is not enough, we must do.

-- Johann Wolfgang von Goethe (1749 - 1832)



Linus Torvalds

Talk is cheap.
Show me the code!

Outline

1. Motivation from Real Tasks
2. Approximate functions
3. Value approximation
4. Policy approximation
5. Actor-Critic from Indirect RL

Motivation from Real Tasks

- **Large-scale problems**
  - Backgammon: 10^{20} states
  - Chinese Go: 10^{170} states
  - Robot / Autonomous car: continuous state space

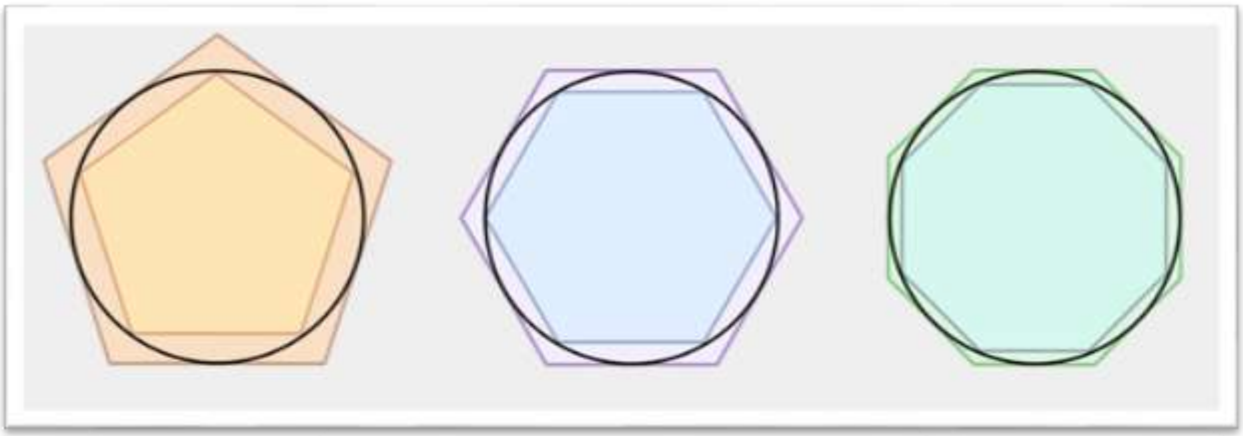


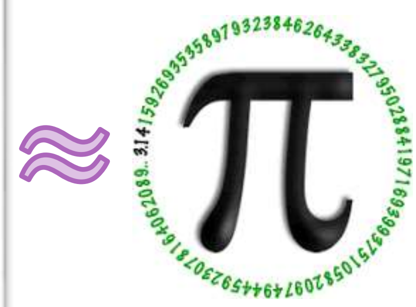

  - Tabular RL suffers from "Curse of Dimensionality"
    - Too many states and actions to store in memory
    - Too slow to learn the value of each state individually

Approximation is Necessary

What is function approximation?

- A version of a piece of information that does not describe it exactly, but is close enough to be used





Benefit of approximation in RL

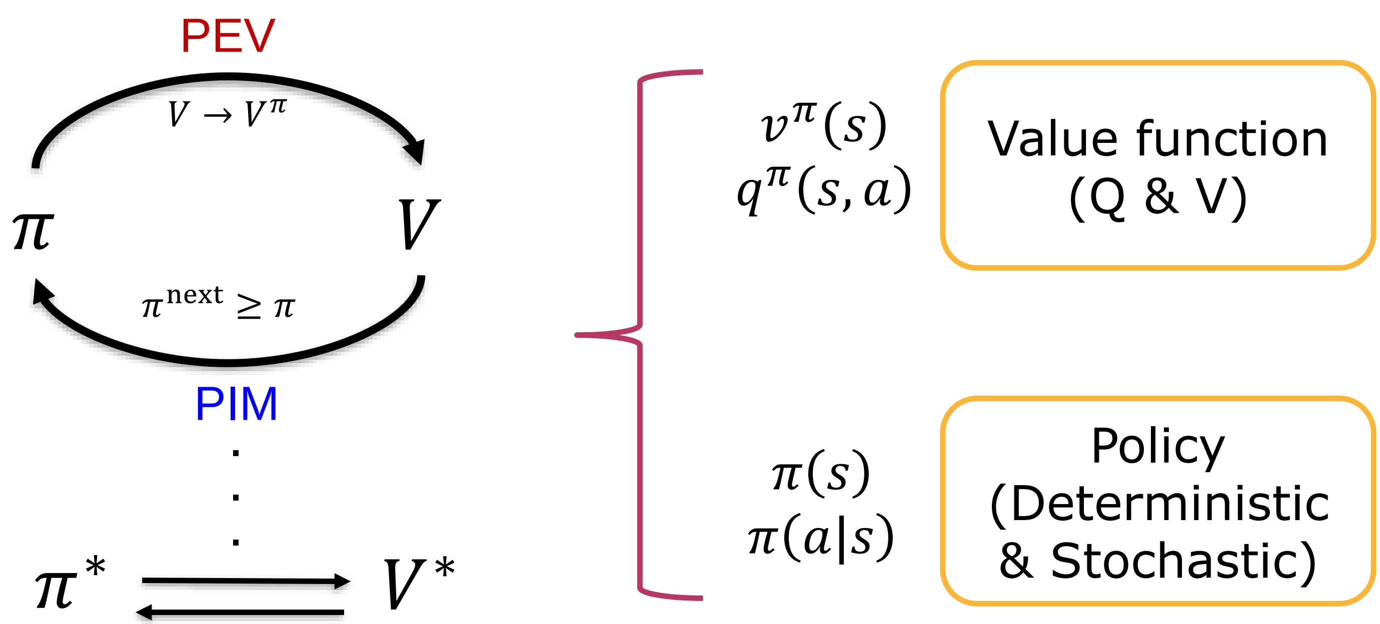
- Get rid of explicitly storing and learning value for every single state
- More compact representation that generalizes across states and actions

Basis of Indirect RLs

- Indirect RL seeks to find the solution of Bellman equation

$$v^*(s) = \max_{\pi}\left\{\mathbb{E}_{\pi}\{r + \gamma v^*(s')\}\right\}$$

- Generalized Policy Iteration (GPI) framework
  - Policy EValuation (PEV) + Policy IMprovement (PIM)



RL with Function Approximation

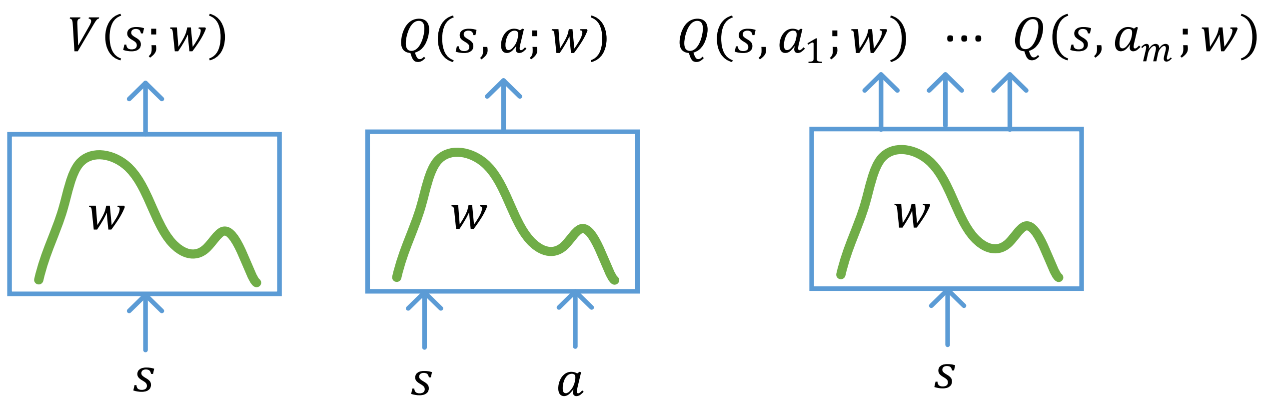
- **(1) Value approximation only**
  - Substitute tabular values by a parameterized function
  - e.g., Deep Q-learning

- **(2) Policy approximation only**
  - Represent policy by a parameterized function
  - e.g., REINFORCE, finite-horizon policy gradient

- **(3) Actor-Critic (AC) architecture**
  - Approximate both value and policy
  - e.g., A3C, DDPG, PPO, TRPO, SAC, DSAC, etc

Three Types of Value Approximation

Parameterized value function



- (A) State-value approximation

$$V(s;w) \approx v^{\pi}(s), \forall s \in \mathcal{S}$$

- (B.1) Action-value approximation (continuous/discrete action space)

$$Q(s,a;w) \approx q^{\pi}(s,a), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

- (B.2) Action-value approximation (discrete action space)

Three Types of Policy Approximation

Parameterized policy

- (A) Deterministic policy

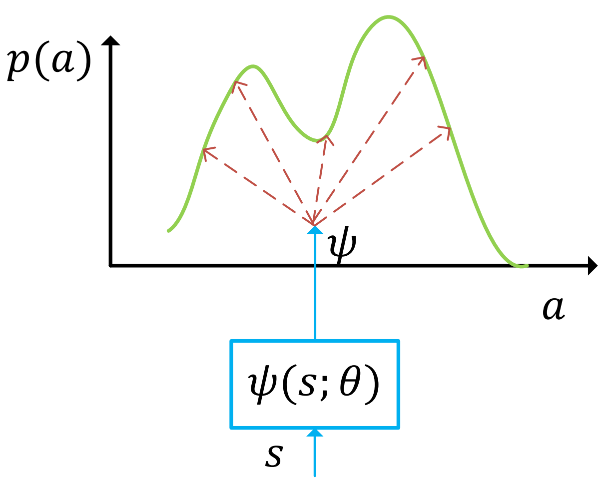
$$a = \pi(s;\theta) \approx \pi(s), \forall s \in \mathcal{S}$$

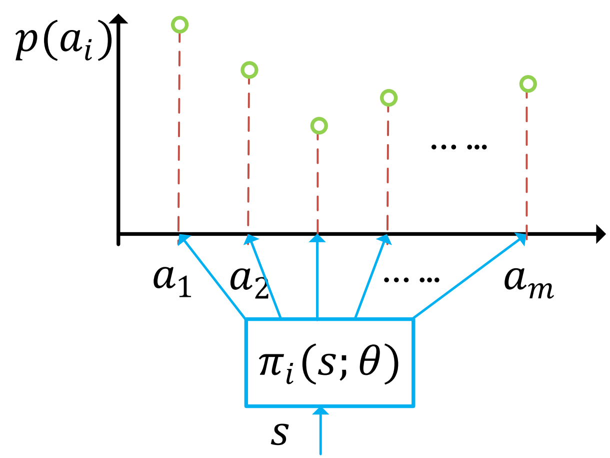
- (B.1) Stochastic policy (continuous action space)

$$p\big(a;\psi(s,\theta)\big) \approx \pi(a|s), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

- (B.2) Stochastic policy (discrete action space)

$$p(a_i|s;\theta) \approx \pi(a_i|s), \forall s \in \mathcal{S}, i = 1, \cdots, m$$



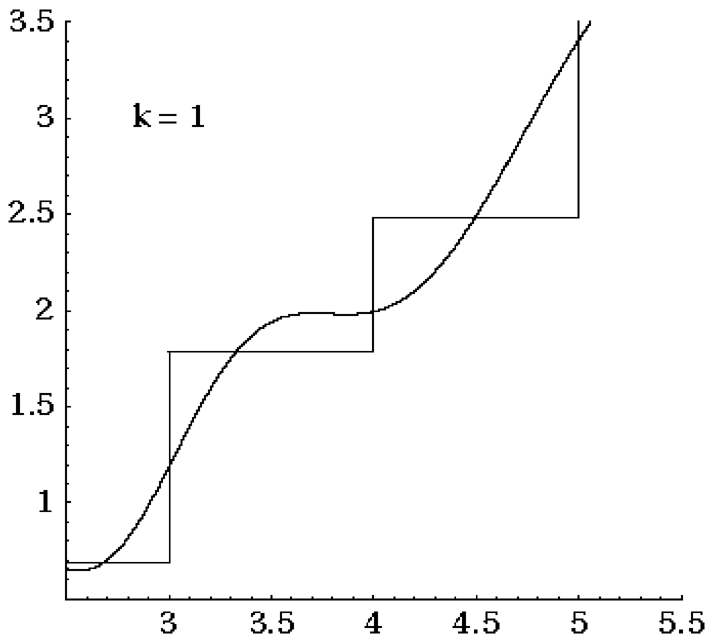


Outline

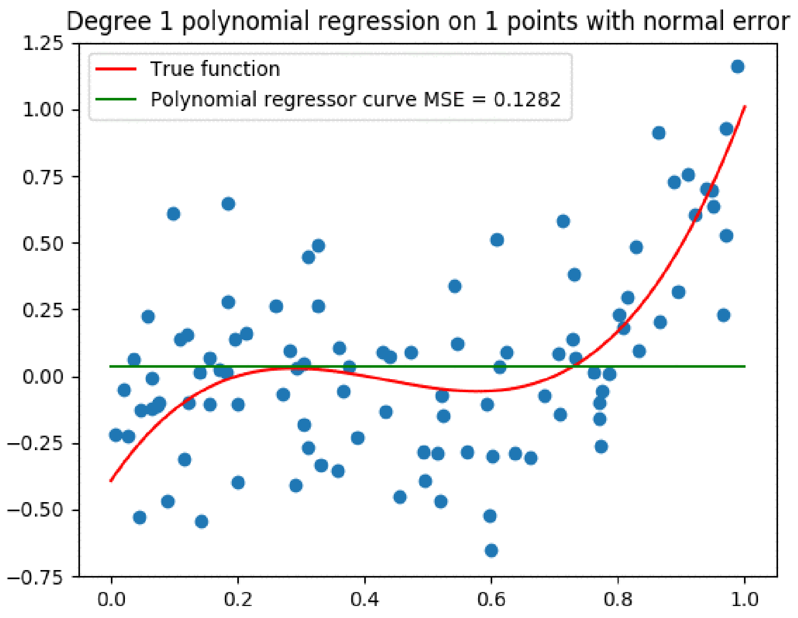
1. Motivation from Real Tasks
2. Approximate functions
3. Value approximation
4. Policy approximation
5. Actor-Critic from Indirect RL

Function Approximation

- **Select a parameterized function that closely matches a target function**
  - Target function is directly known, e.g., Fourier series
  - Only have a set of target points, e.g., Interpolation, Regression



Fourier series



Polynomial regression

Linear Approximation

Approximate as linear combination of features

$$g(\cdot; w) = w^{\mathrm{T}} \cdot \boldsymbol{F}(\boldsymbol{s})$$

Feature = basis function

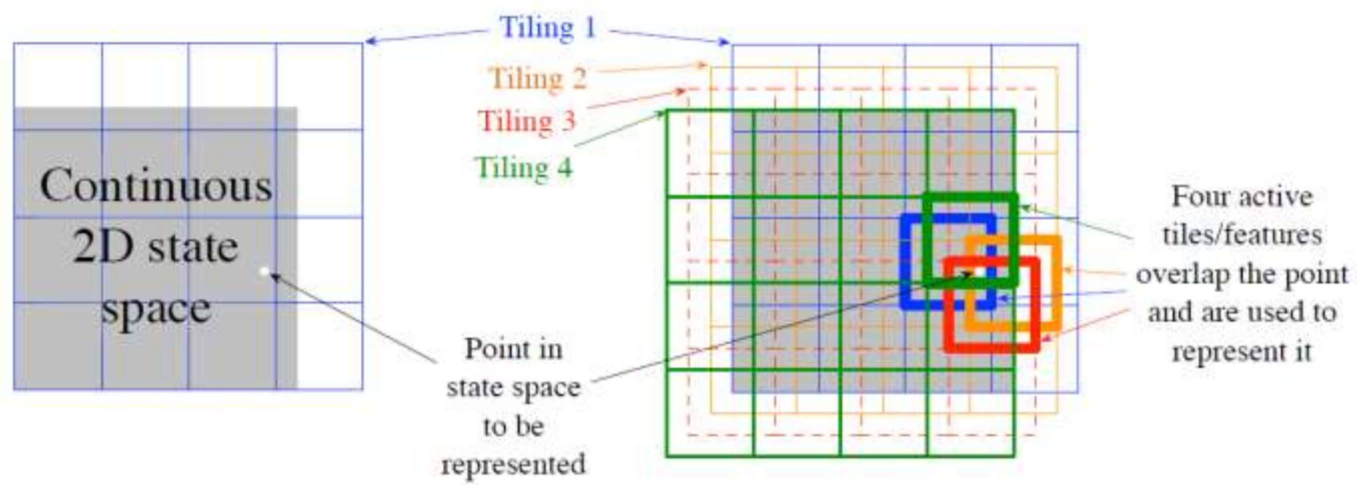
- $w = [w_1, \cdots, w_l]^{\mathrm{T}} \in \mathbb{R}^l$: weights to be learned
- $F(s) = [f_1(s), f_2(s), \cdots, f_l(s)]^{\mathrm{T}} \in \mathbb{R}^l$: features of state s

- Choices of basis function
  - (1) Binary basis function
  - (2) Polynomial basis function
  - (3) Fourier basis function
  - (4) Radial basis function (RBF)

Basis Function of Linear Approximation

(1) Binary basis function

- Suitable for low dimensional space
- Basis function: $F(s) \in \{0,1\}^l$, where l is the dimension of the binary feature
- Example: **Tile coding**
  - Coding a state by a set of overlapping **tilings**
  - Each element of a tiling, called a **tile**



Number of features = 4 tilings x 16 tiles !

Basis Function of Linear Approximation

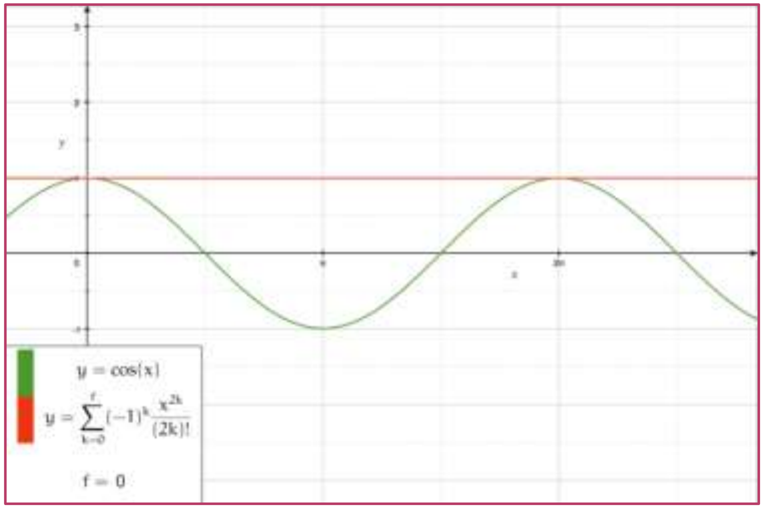
(2) Polynomial basis function

- Suitable for continuous space
- Polynomials with d-order

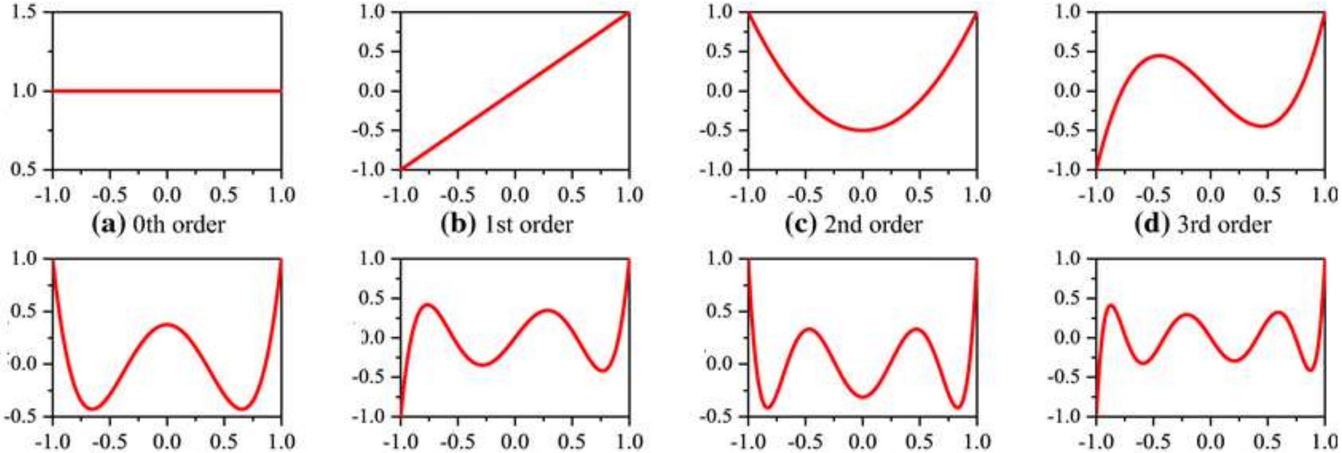
$$s = [s_1, s_2, \dots, s_n]^{\mathrm{T}} \in \mathbb{R}^n$$

$$f_i(s) = \prod_{j=1}^{n} s_j^{c_{i,j}}$$

$$\sum_j c_{i,j} \le d, c_{i,j} \in \{0,1,\dots,n\}$$



- Legendre polynomial basis



(a) 0th order (b) 1st order (c) 2nd order (d) 3rd order

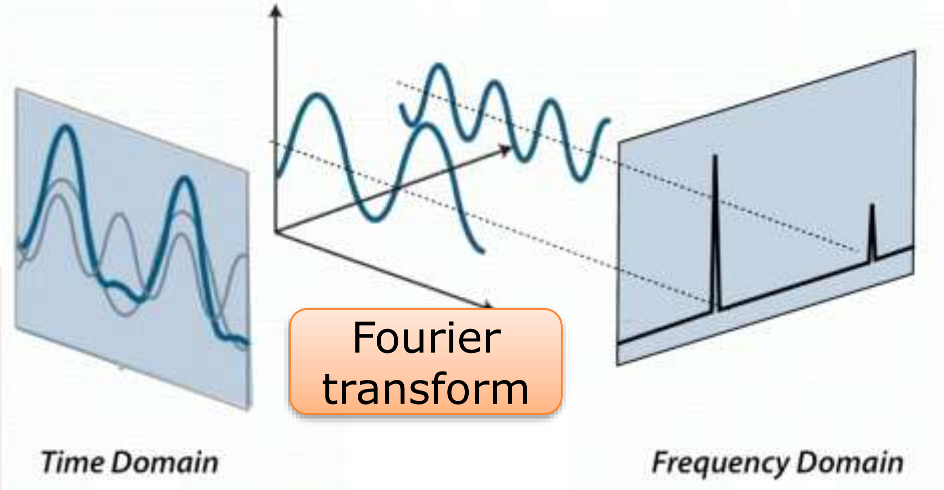
(e) 4th order (f) 5th order (g) 6th order (h) 7th order

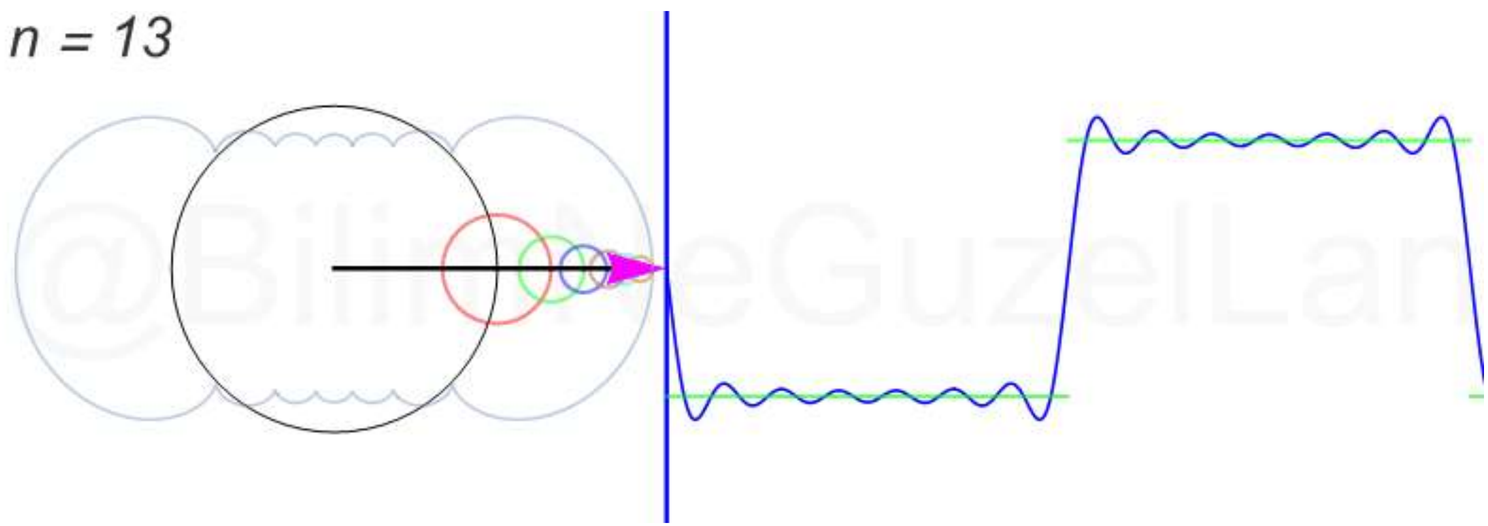
Basis Function of Linear Approximation

(3) Fourier basis function

- Fourier transform
- d-order Fourier cosine approximation

$$f_i(s) = \cos\left(\pi c_i^{\mathrm{T}} s\right)$$
$$c_i = [c_{i,1}, c_{i,2}, \dots, c_{i,n}]^{\mathrm{T}},$$
$$c_{i,j} \in \{0,1,\dots,d\},$$
$$s = [s_1, s_2, \dots, s_n]^{\mathrm{T}}, 0 \le s_j \le 1$$





(4) Radial basis functions (RBF)

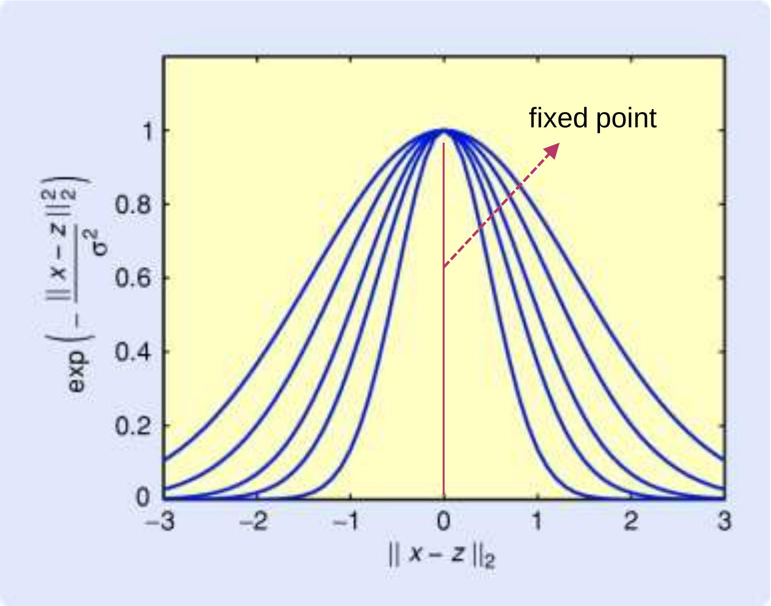
- Michael Powell in 1977
- A real-valued function whose output depends only on the distance between the input and some fixed points.

Michael Powell
(1936-2015)
Cambridge University

- Gaussian RBF

$$f_i(s) = \exp\left(-\frac{\|s-\mu_i\|^2}{2\sigma_i^2}\right)$$

$$s = [s_1, s_2, \cdots, s_n]^{\mathrm{T}} \in \mathbb{R}^n$$

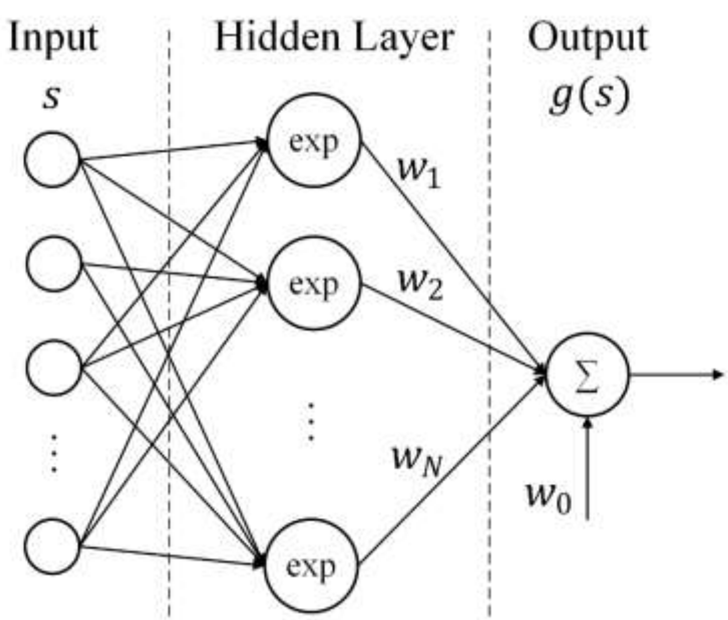


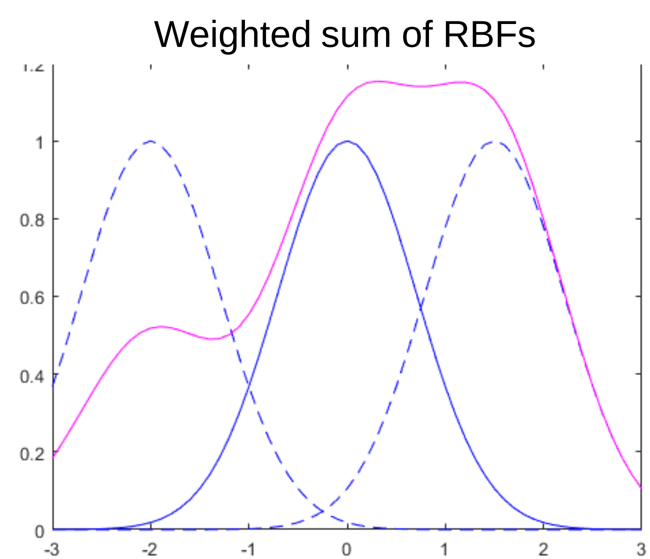
(5) Radial basis function network (RBFN)

- Broomhead & Lowe in 1988
- Three-layer RBF network

$$g(s) = w_0 + \sum_{i=1}^{N} w_i \exp\left(-\frac{\|s-\mu_i\|^2}{2\sigma_i^2}\right)$$

  - A linear input layer, a hidden layer with nonlinear RBF activation function and a linear output layer





Outline

1. Motivation from Real Tasks
2. Approximate functions
3. Value approximation
4. Policy approximation
5. Actor-Critic from Indirect RL

Basis of Value Approximation

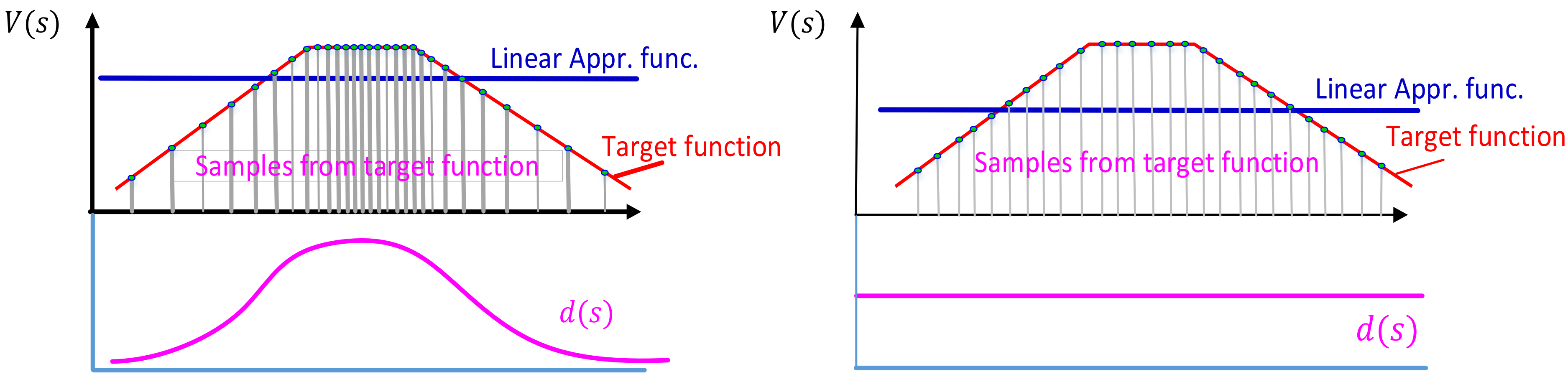
Value approximation problem

- Minimize the difference between target function and approximate function under a certain measure

$$V(s;w) \approx v^{\pi}(s), \forall s \in \mathcal{S}$$

$$\min_{w} J(w) = \mathbb{E}_{s\sim d(s)}\{\phi(v^{\pi}(s), V(s;w))\} = \sum_{s} d(s)\phi(v^{\pi}(s), V(s;w))$$

- State distribution $d(s)$ in the expectation really matters



On-policy Approximation

On-policy objective function

- One common solution is to minimize mean squared error (MSE) under the **stationary state distribution (SSD)** of target policy

$$\min_w J(w) = \mathbb{E}_{s\sim d_\pi}\left\{\left(v^\pi(s) - V(s;w)\right)^2\right\}$$

- Its gradient (called value gradient) is

$$\nabla_w J(w) \propto -\mathbb{E}_{s\sim d_\pi}\left\{\left(v^\pi(s) - V(s;w)\right)\frac{\partial V(s;w)}{\partial w}\right\}$$

- **Questions to calculate value gradients**
  - (1) True value of $v^\pi(s)$ is usually unknown
  - (2) How to handle variable's randomness

Q1: Replace true value with its estimate

Substitute true value with value estimate

$$v^{\pi}(s) \cong \mathbb{E}_{\pi}\{R_t | s\}$$

$$R_t = \begin{cases} G_{t:T} & \text{for} \quad \text{MC} \\ r + \gamma V(s'; w) & \text{for} \quad \text{TD}(0) \end{cases}$$

- For MC

$$\nabla J_{\text{MC}}(w) = -\mathbb{E}_{s \sim d_{\pi}}\left\{\left(\mathbb{E}_{\pi}\{G_{t:T}\} - V(s; w)\right)\frac{\partial V(s; w)}{\partial w}\right\}$$

- For TD

$$\nabla J_{\text{TD}}(w) = -\mathbb{E}_{s \sim d_{\pi}}\left\{\left(\mathbb{E}_{a \sim \pi, s' \sim \mathcal{P}}\{r + \gamma V(s'; w)\} - V(s; w)\right)\frac{\partial V(s; w)}{\partial w}\right\}$$

  - It is also called "semi-gradient"

Q2 (A): Stochastic Gradient Descent

(A) Stochastic gradient descent (SGD)

- Minimize when randomness is present

$$\min_x J(x) = \mathbb{E}_\zeta\{l(x,\zeta)\},$$ where ζ is a randomness

- Step 1: Gradient descent algorithm

$$x \leftarrow x - \alpha \nabla_x J(x)$$

$\nabla_x J(x) = \mathbb{E}_\zeta\{\nabla_x l(x,\zeta)\}$ is called "expected gradient"

- Step 2: Estimation of expected gradient

$$\mathbb{E}_\zeta\{\nabla_x l(x,\zeta)\} \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_x l(x_i,\zeta_i)$$ ⇨ Average-based estimation

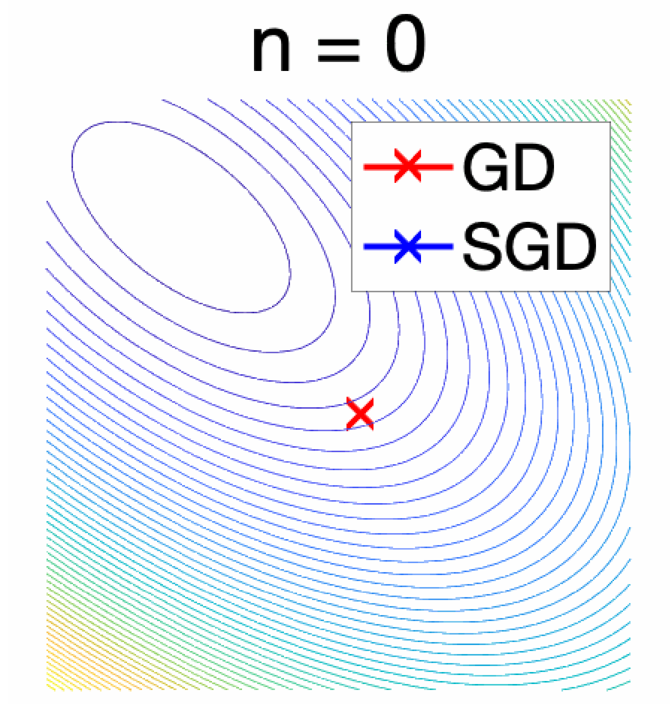
Q2 (A): Stochastic Gradient Descent

(A) Stochastic gradient descent (SGD)

- (1) Use all data (Batch GD)

$$x \leftarrow x - \alpha \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \nabla_x J(x_i, \zeta_i)$$

  - Computationally expensive
  - Delayed update after collecting all data



- (2) Use only one sample (SGD)

$$x \leftarrow x - \alpha \nabla_x J(x_i, \zeta_i)$$

  - Simple and fast convergence
  - Possibly escape from local minimum
  - Randomly choose a small subsets with size ≪ $|\mathcal{D}|$ (Mini-batch GD)

Q2 (B): Least Squares Estimation

Batch Least Squares

- Formulate function approximation as regression problem

$$\min_w J(w) = \sum_{\mathcal{D}} \left(R_t - w^{\mathrm{T}} \cdot F(s_t)\right)^2,$$

subject to $\mathcal{D} = \{(s_t, R_t)\}, t \in \{1{:}T\}$

$$R_t = \begin{cases} G_{t:T} & \text{for} \quad \text{MC} \\ r + \gamma w^{\mathrm{T}} F(s_{t+1}) & \text{for} \quad \text{TD}(0). \end{cases}$$

$\nabla_w J(w) = 0$

- The solution is a fixed point in value function space

$$w = \left(\sum_{t=1}^{T} F(s_t) F(s_t)^{\mathrm{T}}\right)^{-1} \sum_{t=1}^{T} F(s_t) R_t$$

Batch LS

Q2 (B): Least Squares Estimation

Incremental Least Squares

- For n features, inverse computation is $O(n^3)$ in batch LS

$$D_t^{-1} = \left(\sum_{t=1}^{T} F(s_t)F(s_t)^{\mathrm{T}}\right)^{-1}$$

- Apply Shermann-Morrison formula to inverse computation

$$\left(M + UV^{\mathrm{T}}\right)^{-1} = M^{-1} - \frac{M^{-1}UV^{\mathrm{T}}M^{-1}}{1 + V^{\mathrm{T}}M^{-1}U}$$

$$D_t^{-1} = \Big[\underbrace{D_{t-1}}_{M} + \underbrace{F(s_t)}_{U}\underbrace{F(s_t)}_{V}{}^{\mathrm{T}}\Big]^{-1} = D_{t-1}^{-1} - \frac{D_{t-1}^{-1}F(s_t)F(s_t)^{\mathrm{T}}D_{t-1}^{-1}}{1 + F(s_t)^{\mathrm{T}}D_{t-1}^{-1}F(s_t)^{\mathrm{T}}}$$

- Incremental LS is $O(n^2)$ by Shermann-Morrison formula

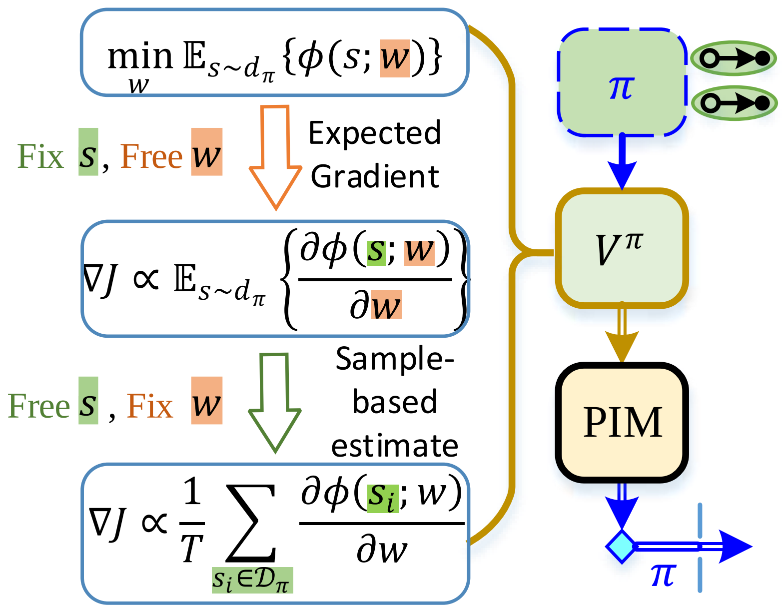
SGD vs LS

SGD	LS
Linear/nonlinear	**Only linear**
Iteratively solve with randomly shuffled samples	Directly solve with batch data
Computationally expensive	Computationally cheap
High variance	Low variance
Oscillation around minimum	Stable

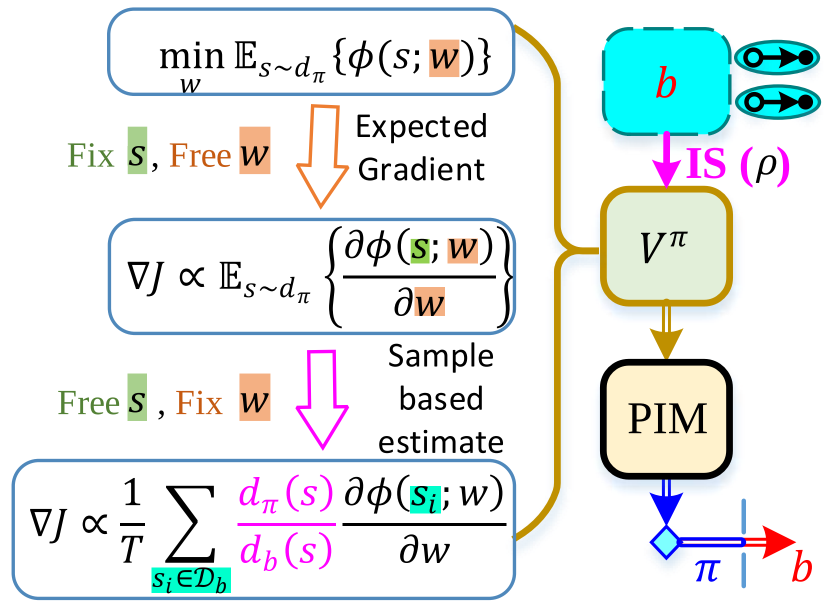
Off-policy value approximation

- Use data from behavior policy b to evaluate target policy π
- One solution is to use the same PEV criterion of on-policy approximation

$$\nabla_w J(w) \propto -\mathbb{E}_{s\sim d_b}\left\{\frac{d_\pi(s)}{d_b(s)}\big(v^\pi(s)-V(s;w)\big)\nabla V(s;w)\right\}$$



On-policy approximation



Off-policy approximation

On-policy vs Off-policy

Off-policy value approximation

- $d_\pi(s)/d_b(s)$ is computationally intractable because we have no access to the SSDs of both target policy and behavior policy
- An alternative is to change PEV criterion to be weighted by the behavior state distribution

$$\min_w J(w) = \mathbb{E}_{s\sim \boldsymbol{d_b}}\left\{\left(v^\pi(s) - V(s;w)\right)^2\right\}$$

$$\nabla_w J(w) \propto -\mathbb{E}_{s\sim \boldsymbol{d_b}}\left\{\left(v^\pi(s) - V(s;w)\right)\frac{\partial V(s;w)}{\partial w}\right\}$$

$$v^\pi(s) = \mathbb{E}_{a\sim b, s'\sim \mathcal{P}}\left\{\rho_{t:t}\left(r + \gamma V(s';w)\right)|s\right\}$$

Take one-step TD as the target

$$\nabla_w J(w) = -\mathbb{E}_b\left\{\left(\boldsymbol{\rho_{t:t}}\left(\boldsymbol{r} + \boldsymbol{\gamma V(s';w)}\right) - V(s;w)\right)\frac{\partial V(s;w)}{\partial w}\right\}$$

Variance reduction in off-policy TD

$$\nabla_w J(w) \propto -\mathbb{E}_b\left\{\boldsymbol{\rho_{t:t}}\left(\boldsymbol{r}+\boldsymbol{\gamma}\boldsymbol{V}(\boldsymbol{s}';\boldsymbol{w})-\underline{\boldsymbol{V}(\boldsymbol{s};\boldsymbol{w})}\right)\frac{\partial V(s;w)}{\partial w}\right\}$$

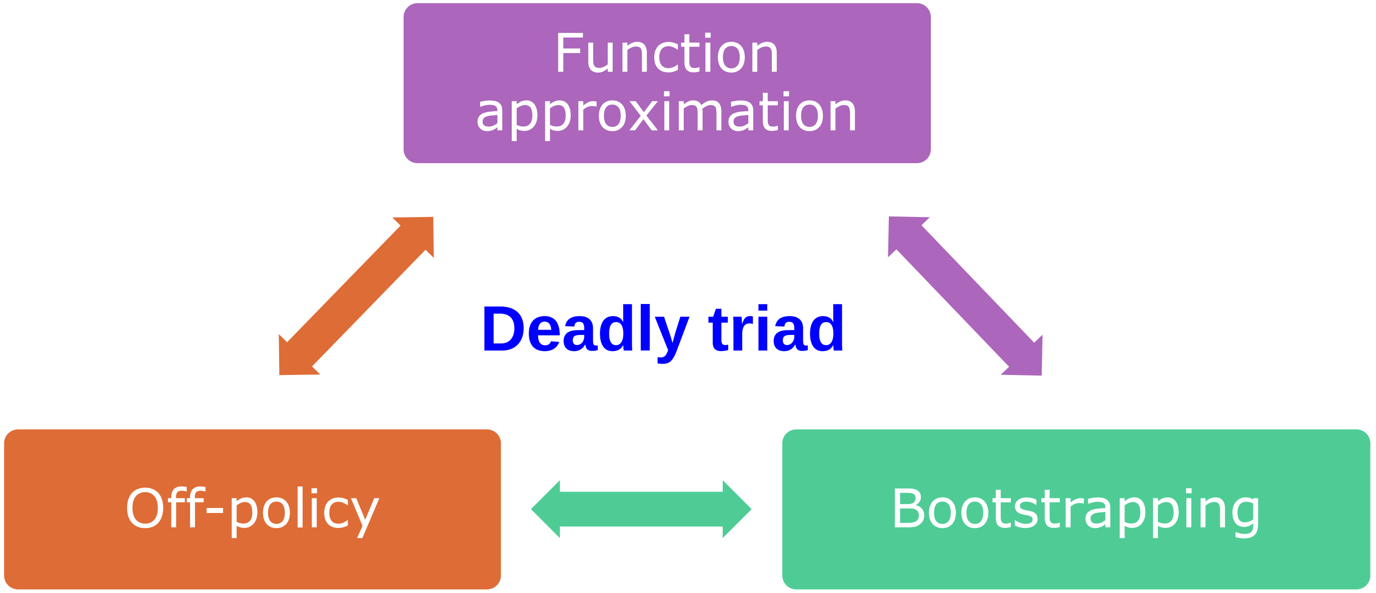
For $\boldsymbol{\rho_{t:t}}$, $\boldsymbol{s}$ is a fixed variable

- Proof:

$$\begin{aligned}
&\mathbb{E}_b\left\{(1-\rho_{t:t})V(s;w)\frac{\partial V(s;w)}{\partial w}\right\}\\
&=\mathbb{E}_{s\sim d_b}\left\{\mathbb{E}_{a\sim b}\left\{(1-\rho_{t:t})V(s;w)\frac{\partial V(s;w)}{\partial w}\right\}\right\}\\
&=\mathbb{E}_{s\sim d_b}\left\{V(s;w)\frac{\partial V(s;w)}{\partial w}\mathbb{E}_{a\sim b}\{(1-\rho_{t:t})\}\right\}\\
&=\mathbb{E}_{s\sim d_b}\left\{V(s;w)\frac{\partial V(s;w)}{\partial w}\sum_a\left[b(a|s)\left(1-\frac{\pi(a|s)}{b(a|s)}\right)\right]\right\}\\
&=\mathbb{E}_{s\sim d_b}\left\{V(s;w)\frac{\partial V(s;w)}{\partial w}\cdot 0\right\}\\
&=0.
\end{aligned}$$

Deadly Triad Issue

- **Risk of instability arises**
  - (1) Function approximation
  - (2) Bootstrapping
  - (3) Off-policy

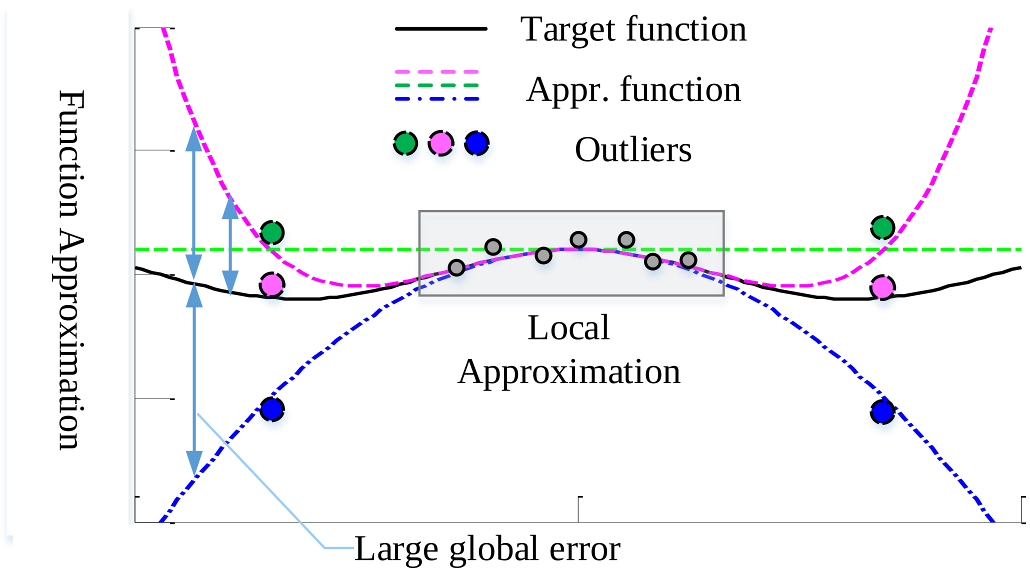


Deadly Triad Issue

Error accumulation for target value during RL iteration

$$v^{\text{target}} = \frac{\pi(a|s)}{b(a|s)}\left(r + \gamma V(s';w)\right)$$

- (1) Function approximation error is extremely large because of a few outliers in less explored area



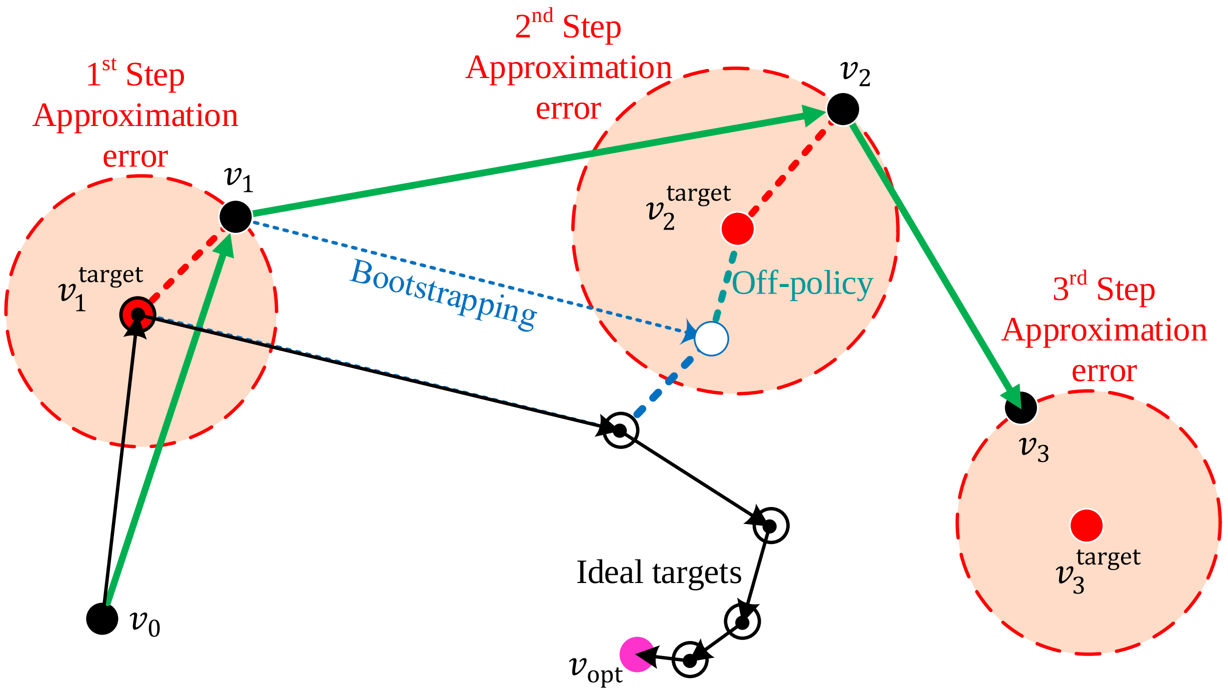
- (2) Off-policy enlarges the error with importance sampling ratio

Deadly Triad Issue

Error accumulation for target value during RL iteration

$$v^{\text{target}} = \frac{\pi(a|s)}{b(a|s)}\left(r + \gamma V(s';w)\right)$$

- (3) Bootstrapping induces current-step approximation error into next target value



Outline

1. Motivation from Real Tasks
2. Approximate functions
3. Value approximation
4. Policy approximation
5. Actor-Critic from Indirect RL

Fundamental of Policy Approximation

Greedy search in tabular PIM

$$\pi^g(s) = \arg\max_a \{Q(s,a)\}, \forall s \in \mathcal{S}$$

Policy approximation problem

- For parameterized policy $\pi_\theta(a|s)$

$$\theta = \arg\max_\theta \Bigg\{ \underbrace{\mathbb{E}_{s\sim d(s)} \left\{ \sum_a \pi_\theta(a|s) Q(s,a) \right\}}_{J(\theta)} \Bigg\}$$

Stochastic policy

- Its updating formula

$$\theta \leftarrow \theta + \beta \nabla_\theta J(\theta)$$

  - Here, $\nabla_\theta J(\theta)$ is called "indirect policy gradient"

Methods to Derive Expected Gradient

- **Two tricks to derive expected gradients**
  - (1) Log-likelihood trick (w/ high variance)

$$
\begin{aligned}
\nabla_\theta \mathbb{E}_{z\sim p_\theta}\{h(z)\} &= \nabla_\theta \sum_z p_\theta(z) h(z) \\
&= \sum_z p_\theta(z) \frac{\nabla_\theta p_\theta(z)}{p_\theta(z)} h(z) \\
&= \mathbb{E}_{z\sim p_\theta}\{\nabla_\theta \log p_\theta(z)\, h(z)\}
\end{aligned}
$$

  - (2) Reparameterization trick (w/ low variance)

$$
z\sim p_\theta \rightarrow \underline{\epsilon\sim p(\epsilon), z = g_\theta(\epsilon)}
$$

Select $g_\theta(\epsilon)$ that decouples θ from randomness

$$
\begin{aligned}
\nabla_\theta \mathbb{E}_{z\sim p_\theta}\{h(z)\} &= \nabla_\theta \mathbb{E}_{\epsilon\sim p(\epsilon)}\{h(g_\theta(\epsilon))\} \\
&= \mathbb{E}_{\epsilon\sim p(\epsilon)}\{\nabla_\theta h(g_\theta(\epsilon))\}
\end{aligned}
$$

Indirect On-Policy Gradient

Trick 1: Gradient with action-value function

- Use the same policy approximation problem as before

$$\nabla_\theta J(\theta) = \sum_s d(s) \sum_a \nabla_\theta \pi_\theta(a|s) Q(s,a)$$

  - Here, action-value $Q(s,a)$ is fixed

Key difference with policy gradients in direct RL

- Replace $d(s)$ with $d_\pi(s)$ in gradient estimation

$$\nabla_\theta J(\theta) = \sum_s d_\pi(s) \sum_a \nabla_\theta \pi_\theta(a|s) Q(s,a)$$

$$= \sum_s d_\pi(s) \sum_a \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)\, Q(s,a)$$

$$= \mathbb{E}_{s\sim d_\pi, a\sim\pi}\{\nabla_\theta \log \pi_\theta(a|s)\, Q(s,a)\}$$

- Equivalent to defining such policy approximation problem as

$$J(\theta) = \sum_s d_\pi(s) \sum_a \pi_\theta(a|s) Q(s,a)$$

Assume that $d_\pi(s)$ does not change with π_θ

Indirect On-Policy Gradient

Trick 1: Gradient with state-value function

Relation between $Q(s,a)$ and $V(s)$

$$Q(s,a) \cong \mathbb{E}_{s'\sim\mathcal{P}}\{r+\gamma V(s')\}$$

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{s\sim d_\pi,a\sim\pi}\{\nabla_\theta \log \pi_\theta(a|s)\, Q(s,a)\} \\ &= \mathbb{E}_{s\sim d_\pi,a\sim\pi,s'\sim\mathcal{P}}\{\nabla_\theta \log \pi_\theta(a|s)\left(r+\gamma V(s')\right)\} \\ &= \mathbb{E}_\pi\{\nabla_\theta \log \pi_\theta(a|s)\left(r+\gamma V(s')\right)\}\end{aligned}$$

Here, $\mathbb{E}_\pi$ represents state distribution, action distribution and next state distribution

- Its computing burden is lower than on-policy gradient with action-value function

Trick 2: Gradient for action-value function

$$\nabla_\theta J = \nabla_\theta \mathbb{E}_{s\sim d(s), a\sim\pi_\theta}\{Q(s,a)\}$$

- Apply reparameterization trick $a = g_\theta(s,\epsilon)$

Take Gaussian policy as example

$$\begin{cases} \pi_\theta(a|s) \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta^2(s)) \\ \Downarrow \\ g_\theta(s,\epsilon) = \mu_\theta(s) + \epsilon\cdot\sigma_\theta(s) \end{cases}$$

$$\epsilon \sim \mathcal{N}(0,1)$$

The key is to separate the randomness from parameterized part to avoid taking derivative w.r.t. random variable

- Gradient with reparametrized stochastic policy

$$\begin{aligned} \nabla_\theta \mathbb{E}_{s\sim d(s), a\sim\pi_\theta}\{Q(s,a)\} &= \nabla_\theta \mathbb{E}_{s\sim d(s), \epsilon\sim p(\epsilon)}\{Q(s, g_\theta(s,\epsilon))\} \\ &= \mathbb{E}_{s\sim d(s), \epsilon\sim p(\epsilon)}\{\nabla_\theta Q(s, g_\theta(s,\epsilon))\} \\ &= \mathbb{E}_{s\sim d(s), \epsilon\sim p(\epsilon)}\{\nabla_a Q(s,a)\nabla_\theta g_\theta(s,\epsilon)\} \end{aligned}$$

Indirect Off-Policy Gradient

- Only have access to $d_b(s)$, i.e., stationary state distribution under the behavior policy
- Replace $d(s)$ with $d_b(s)$ in gradient estimation

$$\nabla_\theta J(\theta) = \sum_s d_b(s) \sum_a \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)\, Q(s,a)$$

$$= \sum_s d_b(s) \sum_a b(a|s) \frac{\pi_\theta(a|s)}{b(a|s)} \nabla_\theta \log \pi_\theta(a|s)\, Q(s,a)$$

$$= \mathbb{E}_{s\sim d_b, a\sim b}\left\{\frac{\pi_\theta(a|s)}{b(a|s)} \nabla_\theta \log \pi_\theta(a|s)\, Q(s,a)\right\}$$

$$= \mathbb{E}_b\left\{\frac{\pi_\theta(a|s)}{b(a|s)} \nabla_\theta \log \pi_\theta(a|s)\, Q(s,a)\right\}$$ w/ action-value function

$$= \mathbb{E}_b\left\{\frac{\pi_\theta(a|s)}{b(a|s)} \nabla_\theta \log \pi_\theta(a|s)\, \left(r + \gamma V(s')\right)\right\}$$ w/ state-value function

Two $\mathbb{E}_b$ have same meanings?

Indirect Off-Policy Gradient

- **Can we get rid of IS ratio in off-policy gradient?**
  - In the case of action-value function

$$\nabla_\theta J(\theta) = \sum_s d_b(s) \sum_a \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)\, Q(s,a)$$

Reason to have IS ratio

  - When we have samples form b policy, i.e., (s_b, a_b, s_b')
  - The next state s_b' is not necessary due to $Q(s,a)$, and the action a_b can be replaced with $a_\pi \sim \pi_\theta(a|s_b)$ to generate action distribution
  - That is equivalent to use a series of new pairs like (s_b, a_π)

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim d_b}\left\{\sum_a \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)\, Q\big(s, \pi_\theta(a|s)\big)\right\}$$

$$= \mathbb{E}_{s\sim d_b, a\sim\pi_\theta}\{\nabla_\theta \log \pi_\theta(a|s_b) Q(s,a)\}$$

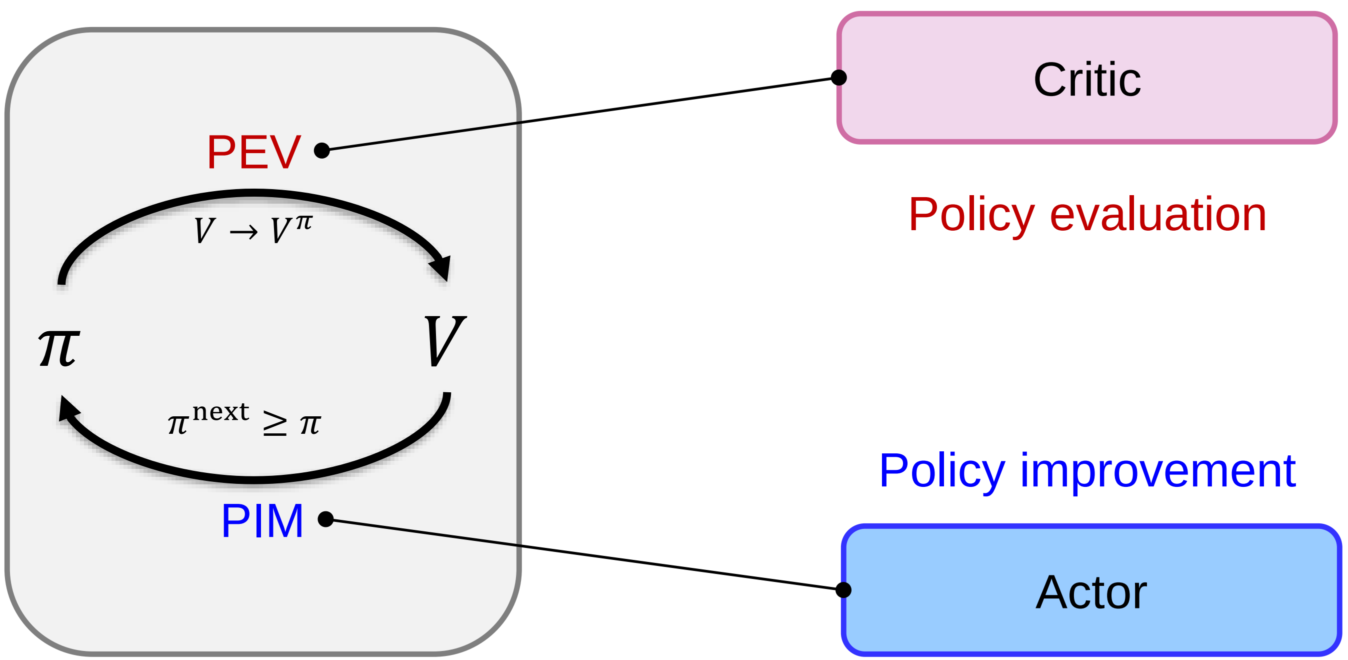
Outline

1. Motivation from Real Tasks
2. Approximate functions
3. Value approximation
4. Policy approximation
5. Actor-Critic from Indirect RL

Actor-Critic Architecture

Basis of actor-critic (AC) architecture

- The element "Actor" controls how the agent behaves with a learned new policy
- The element "Critic" evaluates the agent behavior by estimating its corresponding value function



Actor-Critic Architecture

Indirect RL

- Bellman equation

$$v^*(s) = \max_a \mathbb{E}_\pi\{r + \gamma v^*(s')\}$$

$$\Downarrow$$

$$\pi^*$$

Critic update $w \leftarrow w - \alpha \cdot \partial J(w)/\partial w$

Policy Evaluation

$$\min_w J_{\text{Critic}}(w) = \mathbb{E}\{\phi(\#^\pi(\cdot), \#(\cdot; w))\}$$

$$\# = \{V\} \text{ or } \{Q\}$$

- Value function is fixed
- Find better policy w.r.t V or Q

- Evaluate policy by its value function

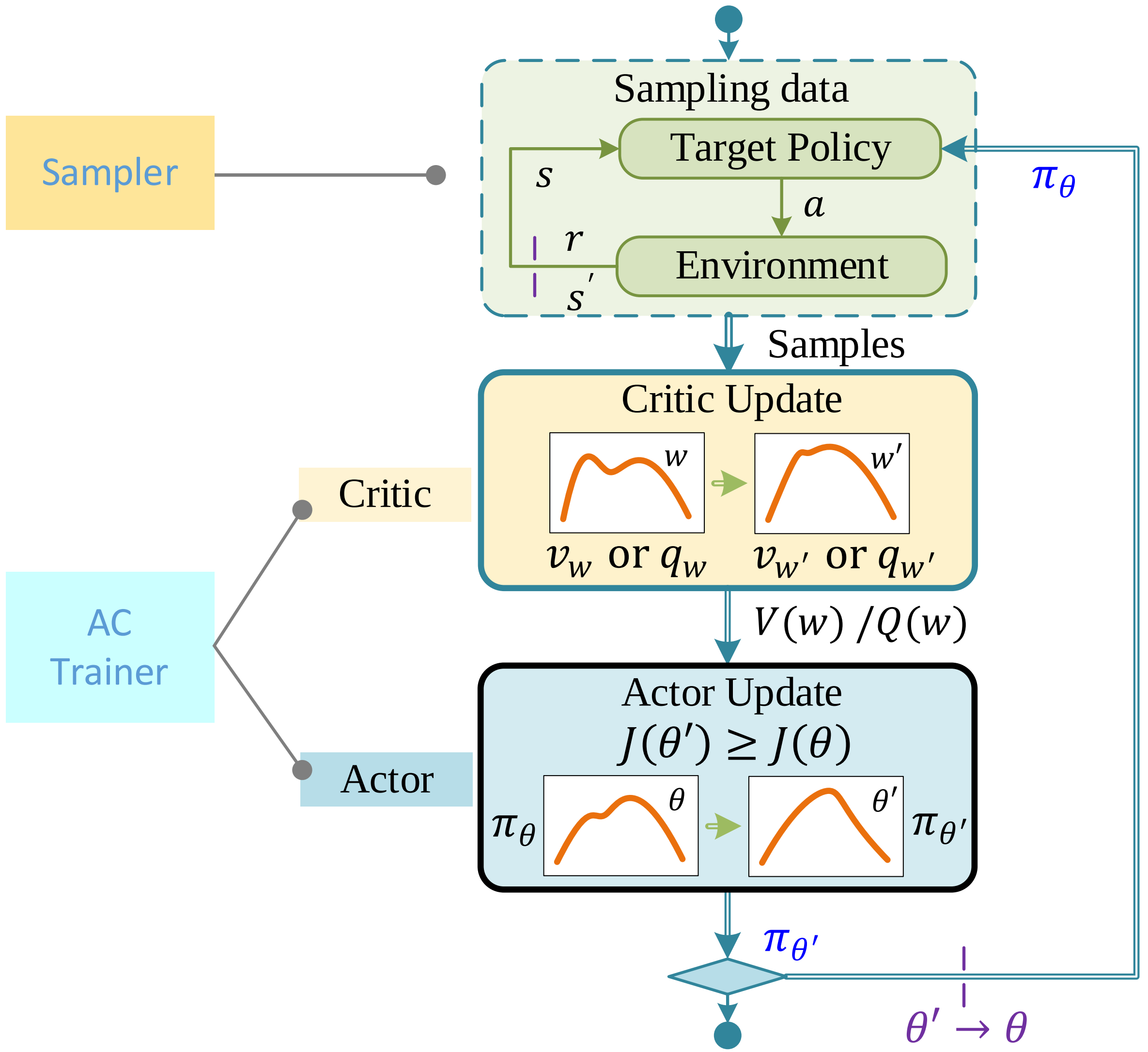
Policy Improvement

$$\max_\theta J_{\text{Actor}}(\theta) = \mathbb{E}_{a \sim \pi(\theta)}\{\#\}$$

$$\# = \{r + \gamma V(s')\} \text{ or } \{Q(s,a)\}$$

Actor update $\theta \leftarrow \theta + \beta \cdot \partial J(\theta)/\partial \theta$

Actor-Critic Architecture



Flow chat of
typical actor-critic
algorithms

(on-policy + off-policy)

Actor-Critic Architecture

Four kinds of AC algorithms from indirect RL

	Action-value function	State-value function
Deterministic policy	$J_{\text{Critic}} = \mathbb{E}_{s,a}\left\{\left(q - Q(w)\right)^2\right\}$	$\cancel{J_{\text{Critic}} = \mathbb{E}_{s}\left\{\left(v - V(w)\right)^2\right\}}$
	$J_{\text{Actor}} = \mathbb{E}_{s}\{Q(s, \pi_\theta(s), w)\}$	$\cancel{J_{\text{Actor}} = \mathbb{E}_{s,s'}\{r + \gamma V(s'; w)\}}$
Stochastic policy	$J_{\text{Critic}} = \mathbb{E}_{s,a}\left\{\left(q - Q(w)\right)^2\right\}$	$J_{\text{Critic}} = \mathbb{E}_{s}\left\{\left(v - V(w)\right)^2\right\}$
	$J_{\text{Actor}} = \mathbb{E}_{s,a}\{Q(s, a; w)\}$	$J_{\text{Actor}} = \mathbb{E}_{s,a,s'}\{r + \gamma V(s'; w)\}$

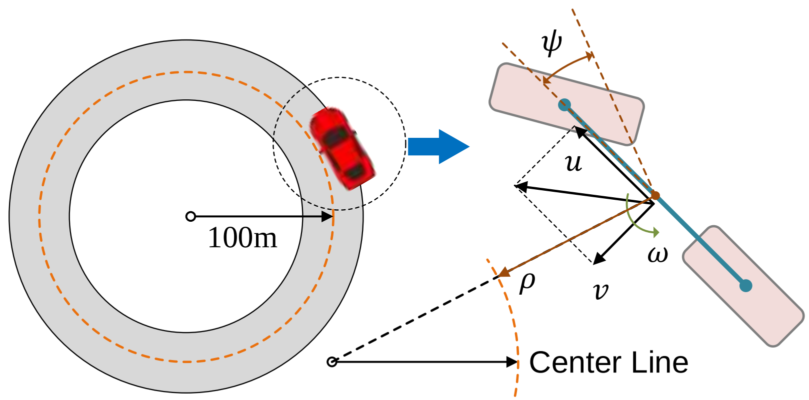
Why such a model-free actor-critic algorithm does NOT exist, i.e., deterministic policy gradient with state-value function?

$$\nabla_a v(s') = \frac{\partial v(s)}{\partial s'} \frac{\partial s'}{\partial a} = \frac{\partial v(s)}{\partial s'} \frac{\partial f(s,a)}{\partial a}$$

Example: Lane-keeping Control

Autonomous driving on a circular road

- To minimize the weighted sum of accumulative errors of lateral position, yaw direction and longitudinal speed



State:

$$s = [\rho, \psi, u, v, \omega] \in \mathbb{R}^5$$

Action:

$$a = [\delta, a_x] \in \mathbb{R}^2$$

Reward:

$$r(s) = c_0 - c_\rho|\rho| - c_\psi \psi^2 - c_u\left|u - u_{\text{exp}}\right| - c_\delta \delta^2 - c_a a_x^2 - I_{\text{fail}}$$

with $$I_{\text{fail}} = \begin{cases} 100, & \text{if out of lane} \\ 0, & \text{otherwise} \end{cases}.$$

Example: Lane-keeping Control

Environment

- Coordinate transformation

$$\dot{\rho} = -u \sin\psi - v\cos\psi$$
$$\dot{\psi} = \omega - \frac{u\cos\psi - v\sin\psi}{\rho}$$

- Bike dynamic model

$$a_x + a_{\text{noise}} = \dot{u} - v\omega$$
$$F_{Y1}\cos\delta + F_{Y2} + F_{\text{ramp}} = m(\dot{v} + u\omega)$$
$$aF_{Y1}\cos\delta - bF_{Y2} = I_{ZZ}\dot{\omega}$$

- Fiala tire model

$$F_{Y\#} = \begin{cases} -C_{\#}\tan\alpha_{\#}\left(\frac{C_{\#}^2(\tan\alpha_{\#})^2}{27(\mu_{\#}F_{Z\#})^2} - \frac{C_{\#}|\tan\alpha_{\#}|}{3\mu_{\#}F_{Z\#}} + 1\right), & |\alpha_{\#}| \le |\alpha_{\max,\#}| \\ \mu_{\#}F_{Z\#}, & |\alpha_{\#}| > |\alpha_{\max,\#}| \end{cases}$$

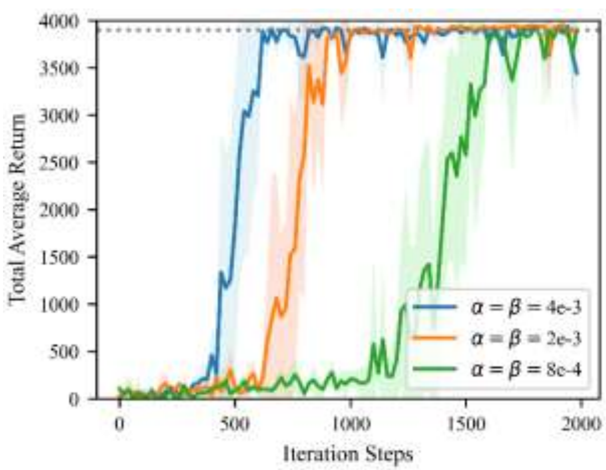
$$\alpha_{\max,\#} = \frac{3\mu_{\#}F_{Z\#}}{C_{\#}}, \quad \mu_{\#} = \frac{\sqrt{(\mu F_{Z\#})^2 - (F_{X\#})^2}}{F_{Z\#}}$$
$$\# = 1,2$$

- External disturbance

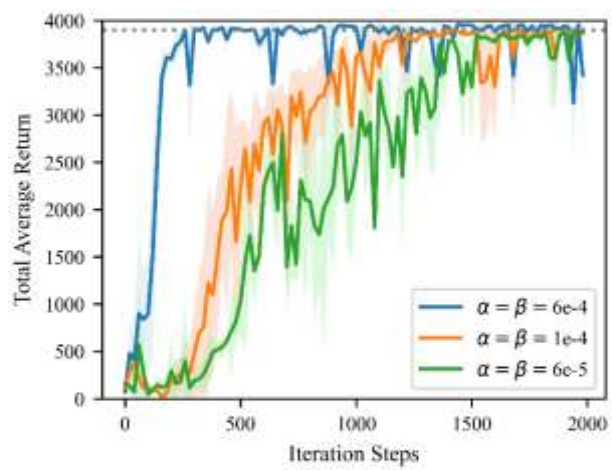
$$F_{\text{ramp}} \sim \mathcal{N}\left(\mu_{\text{ramp}}, \sigma_{\text{ramp}}^2\right)$$
$$a_{\text{noise}} \sim \mathcal{N}\left(\mu_{\text{acce}}, \sigma_{\text{acce}}^2\right),$$

Different learning rates

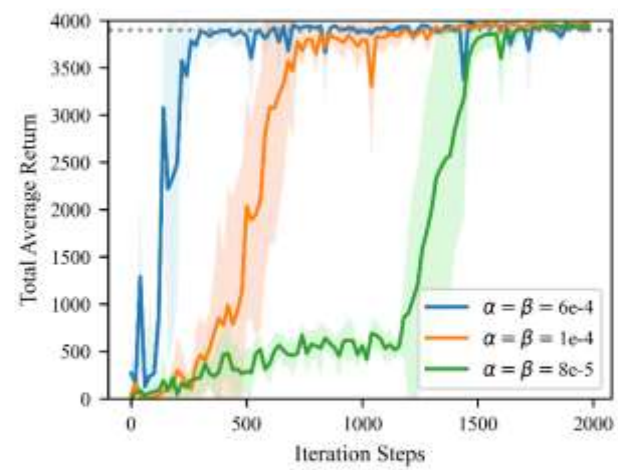
Stochastic + V



Stochastic + Q



Deterministic + Q

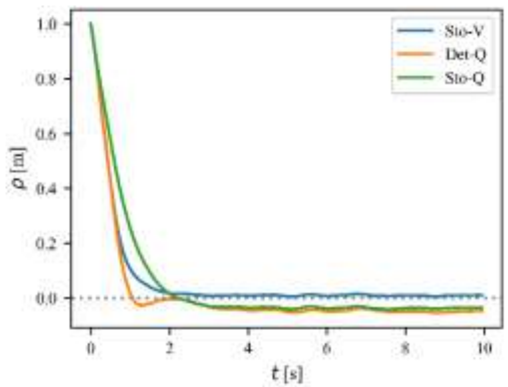


(1) A small learning rate slows training and a too large rate causes instability
(2) Action-value function (Q) benefits fast policy improvement at the start
(3) A deterministic policy has lower variance than a stochastic policy

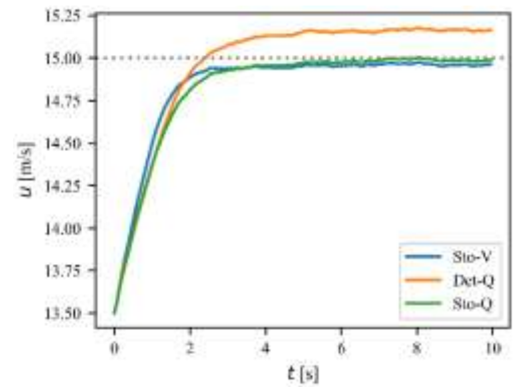
Example: Lane-keeping Control

Self-driving performance

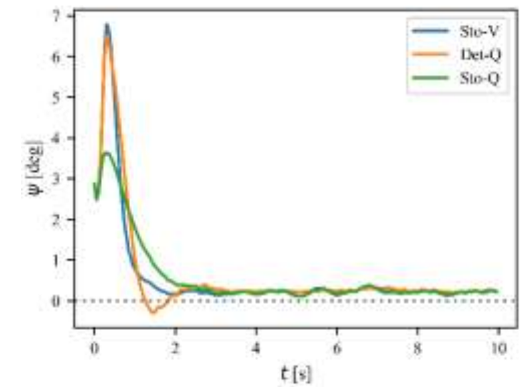
Lateral position error



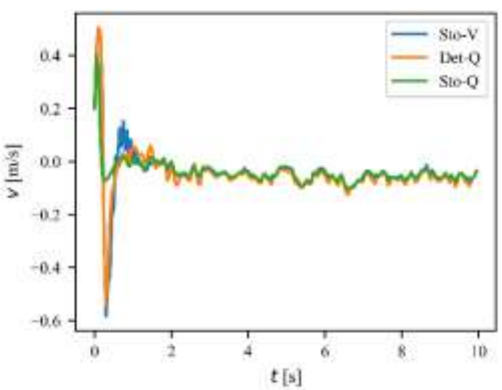
Longitudinal speed



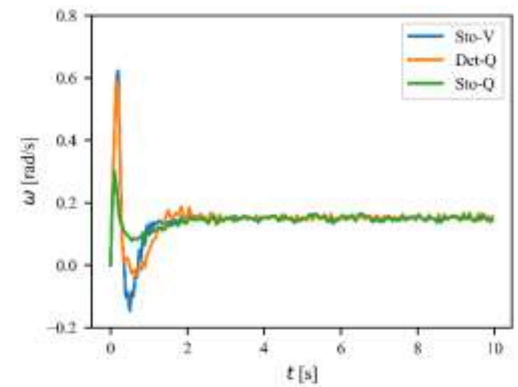
Heading angle error



Lateral velocity



Yaw rate



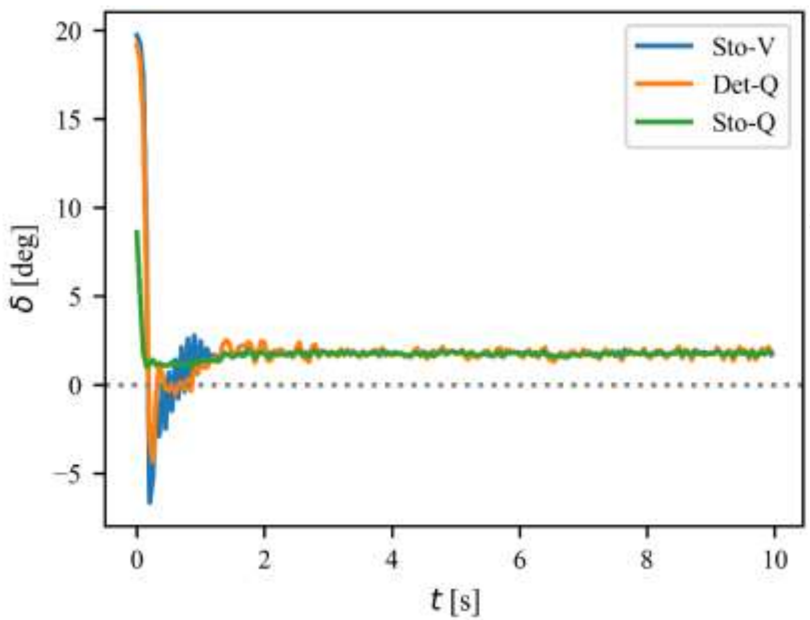
(1) Learned policies are not perfectly optimal and show different behaviors

(2) Stochastic policy exhibits similar smoothing behavior to deterministic one

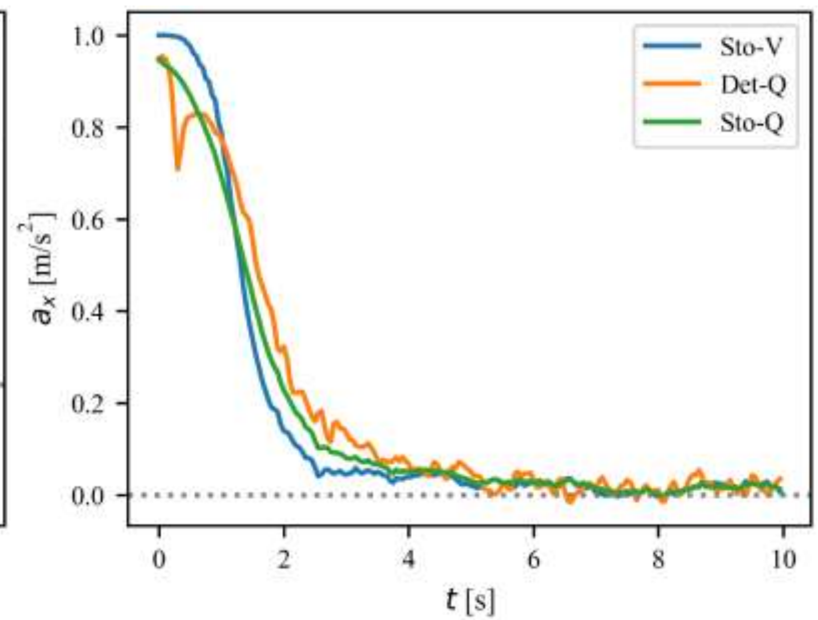
Example: Lane-keeping Control

Self-driving performance

Frontal steering angle



Longitudinal acceleration



(1) Actions cannot settle down to zero due to the noisy uncertainties

(2) Stochastic policy with action-value (i.e., Sto-Q) has the smallest action vibration, while Det-Q and Sto-V have relatively high action fluctuation

The End!