

Reinforcement Learning

Policy Gradient Methods

Stefano V. Albrecht, Michael Herrmann

27 February 2024



THE UNIVERSITY of EDINBURGH
informatics

- Parameterised policies
- Softmax and Gaussian policies as examples
- Policy gradient theorem
- Policy gradient methods: REINFORCE, baselines, actor-critic family

Value Function Approximation

Previously: approximate value function with parameterised function

$$\hat{v}(s, \mathbf{w}) \approx v_{\pi}(s)$$

$$\hat{q}(s, a, \mathbf{w}) \approx q_{\pi}(s, a)$$

Policy was generated *implicitly* from value function (e.g. ϵ -greedy)

- **Compact:** number of parameters in $\mathbf{w}$ can be much smaller than $|\mathcal{S}|$
- **Generalises:** changing one parameter value may change value of many states/actions

Policy Function Approximation

Today: approximate *policy* with parameterised function

$$\pi(a|s, \theta) = \Pr\{A_t = a \mid S_t = s, \theta_t = \theta\}$$

$\theta \in \mathbb{R}^{d'}$ is *policy parameter vector*

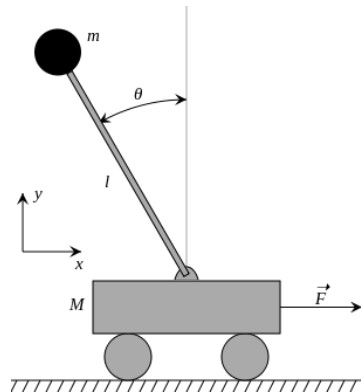
e.g. linear function, neural network, decision tree, ...

- **Compact:** number of parameters in θ can be much smaller than $|\mathcal{S}|$
- **Generalises:** changing one parameter value may change action in many states

Policy vs Value Approximation

Advantages of optimising policy directly:

- Can learn **stochastic policies**
(assign any probabilities to actions)
- Effective in high-dimensional and **continuous action spaces**
⇒ Important for robotics applications
- Better convergence properties,
typically to local optimum

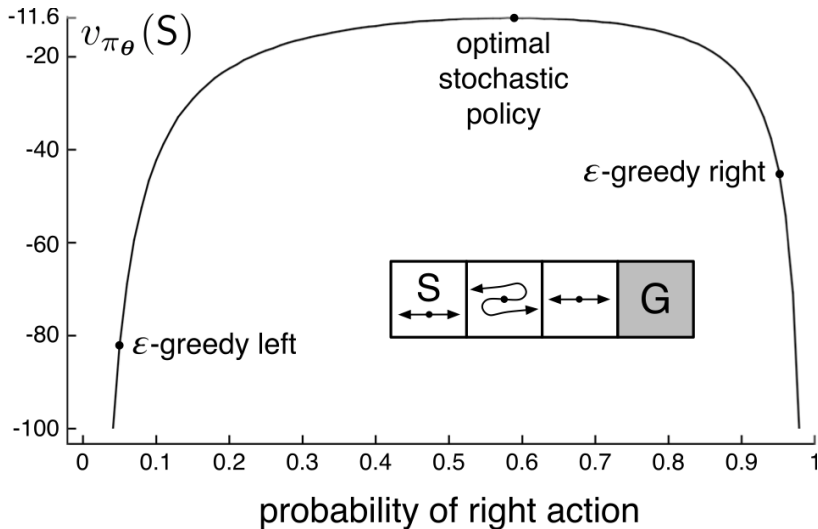


Example: Optimal Stochastic Policy in Short Corridor

Reward is -1 until goal state reached

Assume agent cannot distinguish between states, only between left/right action

$\epsilon = 0.1$



How to parameterise policy?

- We focus on **gradient-based optimisation** → need *differentiable* policy
- Examples:
 - Softmax for discrete actions
 - Gaussian for continuous actions
 - Deep neural network (next lectures)

Parameterised Policies

How to parameterise policy?

- We focus on **gradient-based optimisation** → need *differentiable* policy
- Examples:
 - Softmax for discrete actions
 - Gaussian for continuous actions
 - Deep neural network (next lectures)

How to optimise policy parameters?

- **Policy gradient theorem** leads to family of optimisation algorithms
- Monte Carlo, n-step TD, TD(λ), ...

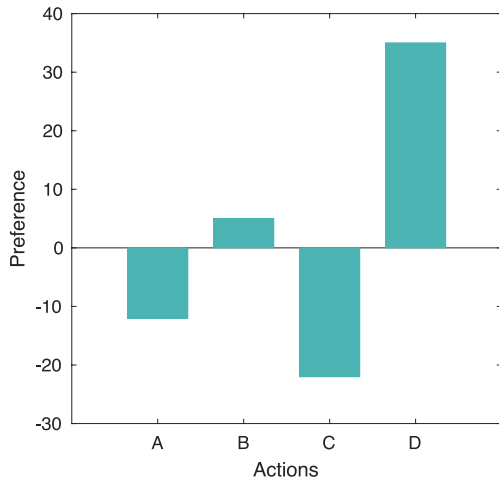
Softmax Policy for Discrete Actions

For discrete actions, can use **softmax** policy:

$$\pi(a|s, \theta) \doteq \frac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}}$$

- Action preference $h(s, a, \theta)$ can be parameterised arbitrarily, e.g. linear in features

$$h(s, a, \theta) = \theta^\top \mathbf{x}(s, a)$$



Gaussian Policy for Continuous Actions

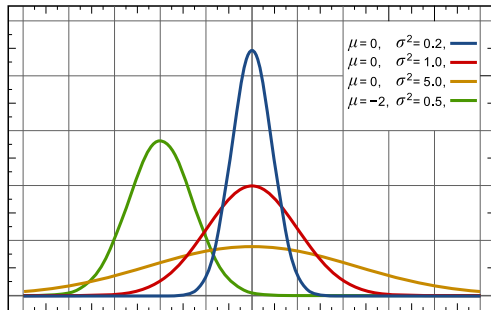
For continuous actions, can use **Gaussian** policy:

$$a \sim \mathcal{N}(\mu(s, \theta), \sigma^2)$$

- Mean μ can be parameterised arbitrarily, e.g. linear in features

$$\mu(s, \theta) \doteq \theta^\top \mathbf{x}(s)$$

- Variance σ^2 can be fixed or also parameterised (see book)



Policy Optimisation Objective

Goal: given policy representation $\pi(a|s, \theta)$, find optimal parameters θ

How to measure quality of θ ?

- In episodic tasks, can use **value of start state** s_0 :

$$J(\theta) \doteq v_{\pi_\theta}(s_0)$$

- In continuing tasks, can use **average reward**:

$$J(\theta) \doteq \sum_s P_\pi(s) \sum_a \pi(a|s, \theta) \sum_{s', r} p(s', r|s, a) r$$

$P_\pi(s)$ is steady-state distribution under π

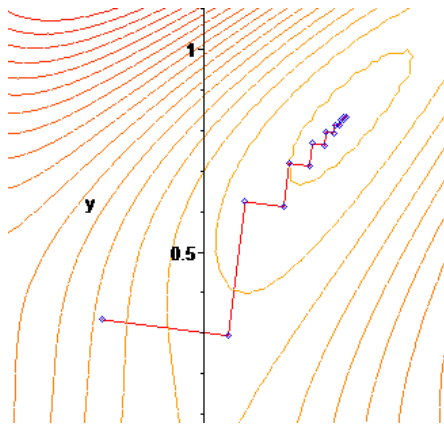
Policy Gradient

- Policy gradient algorithms search for a *local* maximum in $J(\theta)$ by ascending the gradient of π wrt θ

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

- $\nabla J(\theta)$ is the **policy gradient**

$$\nabla J(\theta) = \left(\frac{\partial J(\theta)}{\partial \theta_1}, \dots, \frac{\partial J(\theta)}{\partial \theta_{d'}} \right)$$



Policy Gradient Theorem

Policy Gradient Theorem:

For any *differentiable* policy π , the policy gradient is

$$\nabla J(\theta) = \sum_s d_\pi(s) \sum_a q_\pi(s, a) \nabla \pi(a|s, \theta)$$

$d_\pi(s)$ is the *on-policy distribution* under π :

- For start-state value: $d_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \Pr\{S_t = s \mid s_0, \pi\}$
- For average reward: $d_\pi(s) = \lim_{t \rightarrow \infty} \Pr\{S_t = s \mid \pi\}$ (steady-state dist.)

Note: does not require derivative of environment dynamics $p(s', r|s, a)$!

Sampling Policy Gradient

Since $d_\pi(s)$ is on-policy, we can *sample* approximate gradient:

$$\begin{aligned}\nabla J(\theta) &= \sum_s d_\pi(s) \sum_a q_\pi(s, a) \nabla \pi(a|s, \theta) \\&= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t, a) \nabla \pi(a|S_t, \theta) \right] \\&= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t, \theta) q_\pi(S_t, a) \frac{\nabla \pi(a|S_t, \theta)}{\pi(a|S_t, \theta)} \right] \\&= \mathbb{E}_\pi \left[q_\pi(S_t, A_t) \frac{\nabla \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} \right] \\&= \mathbb{E}_\pi [q_\pi(S_t, A_t) \nabla \ln \pi(A_t|S_t, \theta)]\end{aligned}$$

General Gradient Update

General gradient update: $\theta_{t+1} = \theta_t + \alpha(q_\pi(S_t, A_t) \nabla \ln \pi(A_t|S_t, \theta_t))$

A policy gradient method needs to:

General Gradient Update

General gradient update: $\theta_{t+1} = \theta_t + \alpha (q_\pi(S_t, A_t) \nabla \ln \pi(A_t | S_t, \theta_t))$

A policy gradient method needs to:

- Compute/approximate $\nabla \ln \pi(A_t | S_t, \theta_t)$
 - Softmax policy: $\nabla \ln \pi(a | s, \theta) = \mathbf{x}(s, a) - \sum_{a'} \pi(a' | s, \theta) \mathbf{x}(s, a')$
 - Gaussian policy: $\nabla \ln \pi(a | s, \theta) = (a - \mu(s, \theta)) \mathbf{x}(s) / \sigma^2$

General Gradient Update

General gradient update: $\theta_{t+1} = \theta_t + \alpha (q_\pi(S_t, A_t) \nabla \ln \pi(A_t|S_t, \theta_t))$

A policy gradient method needs to:

- Compute/approximate $\nabla \ln \pi(A_t|S_t, \theta_t)$
 - Softmax policy: $\nabla \ln \pi(a|s, \theta) = \mathbf{x}(s, a) - \sum_{a'} \pi(a'|s, \theta) \mathbf{x}(s, a')$
 - Gaussian policy: $\nabla \ln \pi(a|s, \theta) = (a - \mu(s, \theta)) \mathbf{x}(s) / \sigma^2$
- Approximate $q_\pi(S_t, A_t)$
 - e.g. Monte Carlo: use G_t , since $\mathbb{E}_\pi[G_t|S_t, A_t] = q_\pi(S_t, A_t)$
 - $\Rightarrow$ REINFORCE algorithm

REINFORCE Update Rule

REINFORCE update rule:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)}$$

- $\nabla \pi(A_t|S_t, \theta_t)$
- G_t
- $\pi(A_t|S_t, \theta_t)^{-1}$

REINFORCE Update Rule

REINFORCE update rule:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)}$$

- $\nabla \pi(A_t|S_t, \theta_t)$ – direction in parameter space that most increases the probability of repeating action A_t on future visits to state S_t
- G_t
- $\pi(A_t|S_t, \theta_t)^{-1}$

REINFORCE Update Rule

REINFORCE update rule:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)}$$

- $\nabla \pi(A_t|S_t, \theta_t)$ – direction in parameter space that most increases the probability of repeating action A_t on future visits to state S_t
- G_t – make gradient magnitude proportional to return (better actions get larger updates)
- $\pi(A_t|S_t, \theta_t)^{-1}$

REINFORCE Update Rule

REINFORCE update rule:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)}$$

- $\nabla \pi(A_t|S_t, \theta_t)$ – direction in parameter space that most increases the probability of repeating action A_t on future visits to state S_t
- G_t – make gradient magnitude proportional to return (better actions get larger updates)
- $\pi(A_t|S_t, \theta_t)^{-1}$ – make gradient magnitude inversely proportional to probability of A_t to normalise against frequency of observed A_t (like importance sampling)

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$

Algorithm parameter: step size $\alpha > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

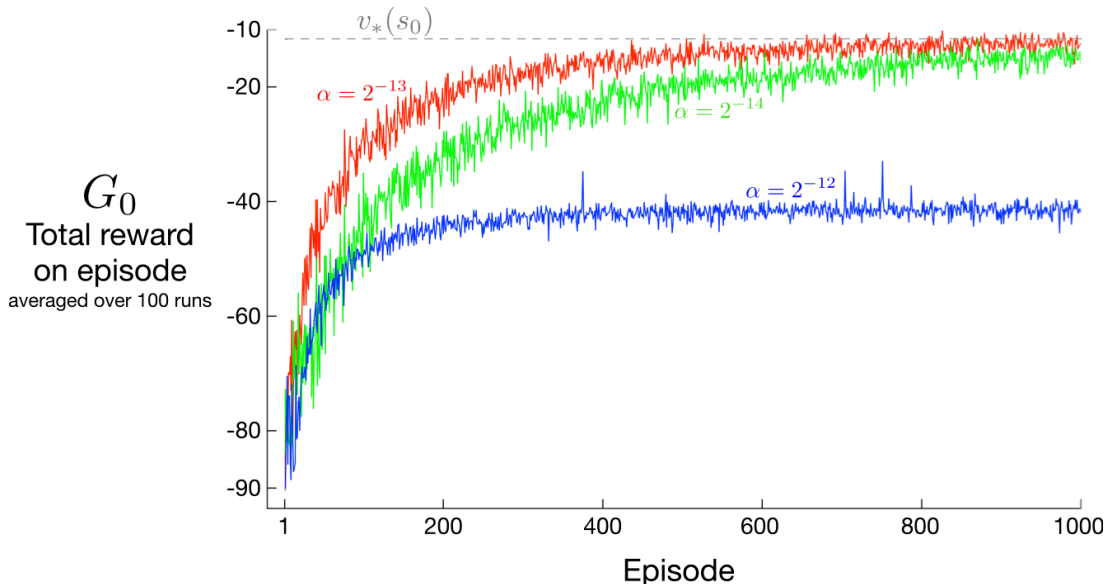
 Generate an episode $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$

 Loop for each step of the episode $t = 0, 1, \dots, T - 1$:

$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \tag{G_t}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$$

REINFORCE in Corridor Example



Baseline to Reduce Variance in Updates

Can generalise policy update to include **baseline**:

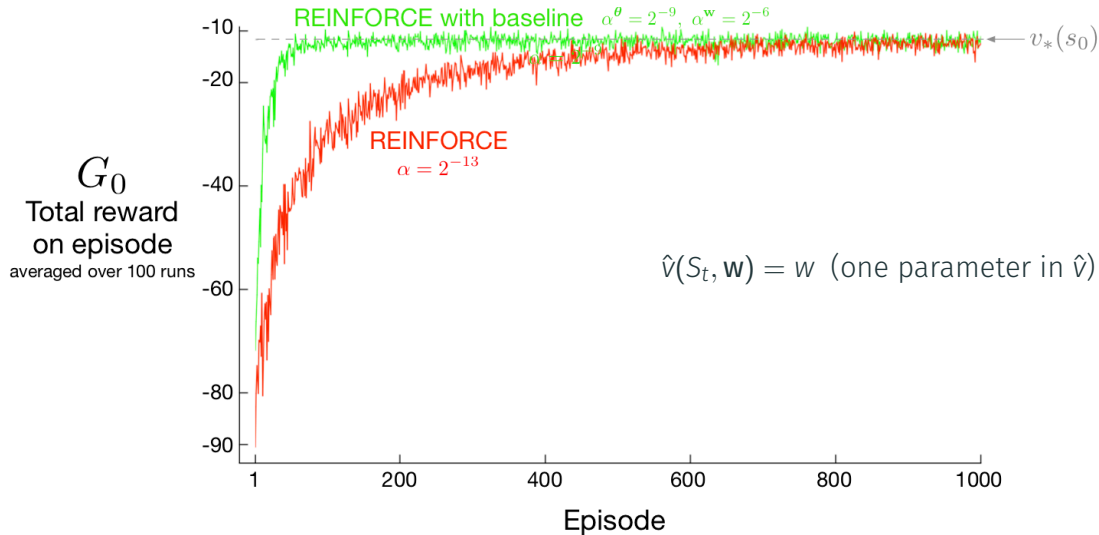
$$(q_{\pi}(S_t, A_t) - b(S_t)) \nabla \ln \pi(A_t | S_t, \theta)$$

Does not change expectation:

$$\begin{aligned} \mathbb{E}_{\pi}[\nabla \ln \pi(A_t | S_t, \theta) b(S_t)] &= \sum_s d_{\pi}(s) \sum_a \nabla \pi(a | s, \theta) b(s) \\ &= \sum_s d_{\pi}(s) b(s) \nabla \sum_a \pi(a | s, \theta) \\ &= \sum_s d_{\pi}(s) b(s) \nabla 1 = 0 \end{aligned}$$

But can reduce *variance* of updates, e.g. use $b(s) = \hat{v}(S_t, \mathbf{w})$

REINFORCE with Baseline in Corridor Example



Actor-Critic Methods

REINFORCE uses MC updates:

- large variance in updates (as any MC method)
- has to wait until end of episode (as any MC method)

Policy gradient can also use TD methods → then called **Actor-Critic method**
e.g. semi-gradient TD(0):

$$\theta_{t+1} = \theta_t + \alpha (R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})) \nabla \ln \pi(A_t | S_t, \theta)$$

- *Critic* updates value function parameters $\mathbf{w}$
- *Actor* updates policy function parameters θ

Actor-Critic with Semi-Gradient TD(0)

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$

Input: a differentiable state-value function parameterization $\hat{v}(s, \mathbf{w})$

Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

 Initialize S (first state of episode)

$I \leftarrow 1$

 Loop while S is not terminal (for each time step):

$A \sim \pi(\cdot|S, \boldsymbol{\theta})$

 Take action A , observe S', R

$\delta \leftarrow R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$ (if S' is terminal, then $\hat{v}(S', \mathbf{w}) \doteq 0$)

$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S, \mathbf{w})$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$

$I \leftarrow \gamma I$

$S \leftarrow S'$

More advanced policy gradient methods:

- Natural Policy Gradient
- Trust Region Policy Optimisation
- Proximal Policy Optimisation
- Deterministic Policy Gradient

(Search on Google Scholar)

Required:

- RL book, Chapter 13 (13.1–13.5)

End of examinable material. For extra exam revision, see Tutorials 8 & 9.

Optional:

- S. Bhatnagar, R. Sutton, M. Ghavamzadeh, M. Lee (2009). Natural Actor–Critic Algorithms. *Automatica*, 45(11)
- Marc P. Deisenroth, G. Neumann, J. Peters (2013). A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics*, Vol. 2: No. 1–2, pp 1-142