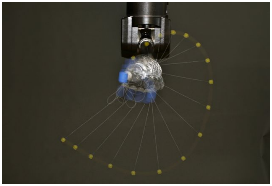
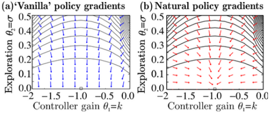
Reinforcement Learning

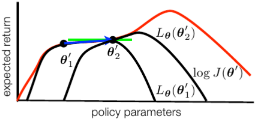
Policy Gradient

Marcello Restelli



March–April, 2015





Value Based and Policy–Based Reinforcement Learning

- Value Based
  - **Learn** value function
  - **Implicit** policy
- Policy Based
  - **No** value function
  - **Learn** policy
- Actor–Critic
  - **Learn** value function
  - **Learn** policy

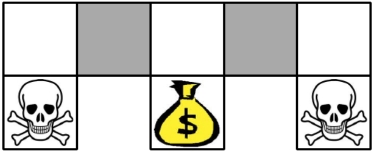
Advantages of Policy–Based RL

- Advantages:
  - Better **convergence** properties
  - Effective in **high–dimensional** or **continuous action** spaces
  - Can benefit from **demonstrations**
  - **Policy subspace** can be chosen according to the **task**
  - **Exploration** can be directly controlled
  - Can learn **stochastic policies**
- Disadvantages:
  - Typically converge to a **local** rather than a global optimum
  - Evaluating a policy is typically **inefficient** and **high variance**

Example: Rock–Paper–Scissor

- Two–player game of rock–paper–scissors
  - Scissors beats paper
  - Rock beats scissors
  - Paper beats rock
- Consider policies for **iterated** rock–paper–scissors
  - A deterministic policy is **easily exploited**
  - A **uniform random policy** is optimal (i.e., Nash equilibrium)

Example: Aliased Gridworld



- The agent **cannot differentiate** the gray states
- Consider **features** of the following form (for all N, E, S, W)

$$\phi(s, a) = \mathbf{1}(\text{wall to N}, a = \text{move E})$$

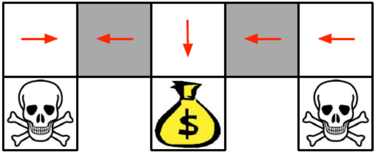
- Compare value–based RL, using an **approximate value function**

$$Q_\theta(s, a) = f(\phi(s, a), \theta)$$

- To policy–based RL, using a **parameterized policy**

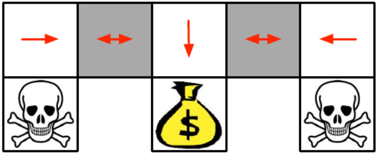
$$\pi_\theta(s, a) = g(\phi(s, a), \theta)$$

Example: Aliased Gridworld



- Under aliasing, an optimal **deterministic** policy will either
  - move W in both gray states
  - move E in both gray states
- Either way, it can get stuck and **never** reach the money
- Value–based RL learns a **near–deterministic policy**
- So it will traverse the corridor for a **long time**

Example: Aliased Gridworld



- An optimal **stochastic** policy will randomly move E or W in gray states

$$\pi_\theta(\text{wall to N and S, move E}) = 0.5$$

$$\pi_\theta(\text{wall to N and S, move W}) = 0.5$$

- It will reach the goal state in a few steps with high probability
- Policy–based RL can learn the optimal stochastic policy

Policy Objective Function

- **Goal**: given a policy $\pi_\theta(a|s)$ with parameters θ, find best θ
- But how do we **measure** the quality of a policy π_θ?
- We want to optimize the **expected return**

$$J(\theta) = \int_{\mathcal{S}} \mu(s) V^{\pi_\theta}(s) ds = \int_{\mathcal{S}} d^{\pi_\theta}(s) \int_{\mathcal{A}} \pi(a|s) R(s,a) da ds$$

- where d^{π_θ} is the **stationary distribution**

Policy Optimization

- Policy based reinforcement learning is an **optimization** problem
- Find θ that maximizes $J(\theta)$
- Some approaches **do not use gradient**
  - Hill climbing
  - Simplex
  - Genetic algorithms
- Greater **efficiency** often possible using gradient
  - Gradient descent
  - Conjugate gradient
  - Quasi–Newton
- We focus on **gradient descent**, many extensions possible
- And on methods that exploit **sequential structure**

Greedy vs Incremental

- **Greedy** updates

$$\theta_{\pi'} = arg\max_{\theta} \mathbb{E}_{\pi_\theta}[Q^{\pi}(s,a)]$$

- $V^{\pi_0} \xrightarrow[\text{change}]{\text{small}} \pi_1 \xrightarrow[\text{change}]{\text{large}} V^{\pi_1} \xrightarrow[\text{change}]{\text{large}} \pi_2 \xrightarrow[\text{change}]{\text{large}}$
- Potentially **unstable** learning process with **large policy jumps**
- **Policy Gradient** updates

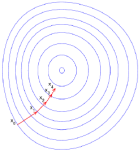
$$\theta_{\pi'} = \theta_{\pi} + \alpha \left.\frac{dJ(\theta)}{d\theta}\right|_{\theta=\theta^{\pi}}$$

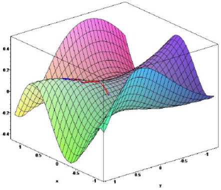
- $V^{\pi_0} \xrightarrow[\text{change}]{\text{small}} \pi_1 \xrightarrow[\text{change}]{\text{small}} V^{\pi_1} \xrightarrow[\text{change}]{\text{small}} \pi_2 \xrightarrow[\text{change}]{\text{small}}$
- **Stable** learning process with **smooth policy improvement**

Policy Gradient

- Let $J(\theta)$ be any **policy objective function**
- Policy gradient algorithms search for a **local maximum** in $J(\theta)$ by ascending the gradient of the policy, w.r.t. parameters θ

$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$



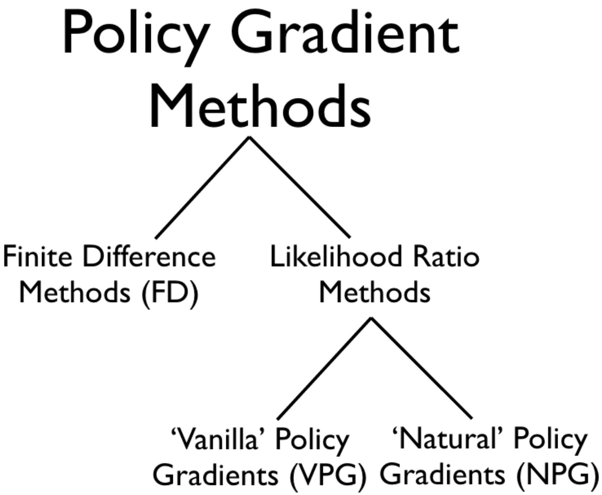


- Where $\nabla_\theta J(\theta)$ is the **policy gradient**

$$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix}$$

- and α is a step–size parameter

Policy Gradient Methods



Computing Gradients by Finite Differences

- **Black–box** approach
- To **evaluate** policy gradient of $\pi(a|s)$
- For each dimension $k \in [1, n]$
  - Estimate k–th **partial derivative** of objective function w.r.t. θ
  - By **perturbing** θ by small amount ϵ in k–th dimension

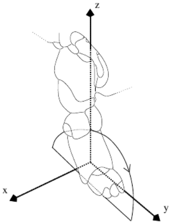
$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

  where u_k is unit vector with 1 in k–th component, 0 elsewhere
- Uses n **evaluations** to compute policy gradient in n dimensions

$$g_{FD} = \left(\Delta\Theta^{\mathrm{T}}\Delta\Theta\right)^{-1}\Delta\Theta^{\mathrm{T}}\Delta J$$

- Simple, noisy, inefficient, but sometimes effective
- Works for arbitrary policies, even if policy is **not differentiable**

AIBO Walking Policies



Initial gate

White–Box approach

- Use an explorative, **stochastic** policy and make use of the knowledge of your policy
- We now compute the gradient **analytically**
- Assume we **know** the gradient $\nabla_\theta \pi_\theta(s, a)$

Likelihood Ratio Gradient

- For a cost function

$$J(\theta) = \int_{\mathbb{T}} p_\theta(\tau|\pi) R(\tau) d\tau$$

- we have the gradient

$$\nabla_\theta J(\theta) = \nabla_\theta \int_{\mathbb{T}} p_\theta(\tau|\pi) R(\tau) d\tau = \int_{\mathbb{T}} \nabla_\theta p_\theta(\tau|\pi) R(\tau) d\tau$$

- Using the trick

$$\nabla_\theta p_\theta(\tau|\pi) = p_\theta(\tau|\pi) \nabla_\theta \log p_\theta(\tau|\pi)$$

- We obtain

$$\begin{aligned} \nabla_\theta J(\theta) &= \int_{\mathbb{T}} p_\theta(\tau|\pi) \nabla \log p_\theta(\tau|\pi) R(\tau) d\tau \\ &= \mathbb{E}[\nabla_\theta \log p_\theta(\tau|\pi) R(\tau)] \\ &\approx \frac{1}{K} \sum_{k=1}^{K} \nabla_\theta \log p_\theta(\tau_k|\pi) R(\tau_k) \end{aligned}$$

- Needs only **samples**!

Characteristic Eligibility

- Why the previous result is **cool**?
- The definition of a **path probability**

$$p_\theta(\tau) = \mu(s_1)\Pi_{t=1}^{T} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

- implies

$$\log p_\theta(\tau) = \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \text{const}$$

- Hence, we can get the derivative of the distribution **without a model** of the system:

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

- The **characteristics eligibility** is $\nabla_\theta \log \pi_\theta(a|s)$

Softmax Policy

- We will use **softmax policy** as a running example
- Weight actions using **linear combination** of features $\phi(s,a)^\intercal\theta$
- Probability of action is proportional to exponential weight

$$\pi_\theta(s,a) \propto e^{\phi(s,a)^\intercal\theta}$$

- The **characteristic eligibility** is

$$\nabla_\theta \log \pi_\theta(a|s) = \phi(s,a) - \mathbb{E}_{\pi_\theta}[\phi(s,\cdot)]$$

Gaussian Policy

- In **continuous action spaces**, a Gaussian policy is natural
- **Mean** is a linear combination of state features $\mu(s) = \phi(s)^\mathsf{T}\theta$
- **Variance** may be fixed σ^2, or can also parameterized
- Policy is a Gaussian, $a \sim \mathcal{N}(\mu(s), \sigma)$
- The characteristic eligibility is

$$\nabla_\theta \log \pi_\theta(s, a) = \frac{(a - \mu(s))\phi(s)}{\sigma^2}$$

One–Step MDPs

- Consider a simple class of **one–step** MDPs
  - **Starting** in state $s \sim d(\cdot)$
  - **Terminating** after one time–step with reward $r = R(s, a)$
- Use **likelihood ratios** to compute policy gradient

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\pi_\theta}[r] \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) R(s, a) \\
\nabla_\theta J(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) R(s, a) \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) r]
\end{aligned}
$$

Policy Gradient Theorem

- The policy gradient theorem generalize the likelihood ratio approach to **multi–step MDPs**
- Replaces instantaneous reward r with **long–term value** $Q^{\pi}(s, a)$
- Policy gradient theorem applies to **start state** objective, **average reward** and **average value** objective

Theorem

For any differentiable policy $\pi_\theta(a|s)$, for any of the policy objective functions $J = J_1$, J_{avR}, or $\frac{1}{1-\gamma} J_{avV}$, the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)]$$

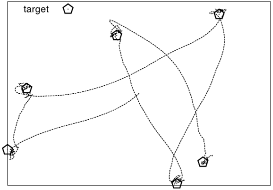
Monte–Carlo Policy Gradient

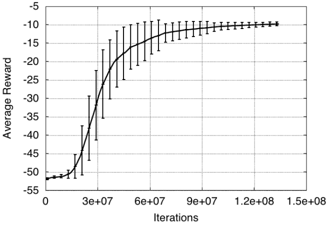
- Update parameters by **stochastic gradient ascent**
- Using **policy gradient theorem**
- Using return v_t as an **unbiased** sample of $Q^{\pi_\theta}(s_t, a_t)$

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) v_t$$

function REINFORCE()
  Initialize θ arbitrarily
  for all episodes $\{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
    for $t = 1$ to $T - 1$ **do**
      $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t, s_t) v_t$
    end for
  end for
  return θ
end function

Puck World Example





- **Continuous actions** exert small force on puck
- Puck is rewarded for getting **close to target**
- Target location is **reset** every 30 seconds
- Policy is trained using variant (conjugate) of Monte–Carlo policy gradient

Reducing Variance using Critic

- Monte–Carlo policy gradient still has a **high variance**
- We use a **critic** to estimate the action–value function

$$Q_w(s,a) \approx Q_w^{\pi_\theta}(s,a)$$

- Actor–critic algorithms maintain **two** sets of parameters
  - **Critic**: Updates **action–value function** parameters w
  - **Actor**: Updates policy parameters θ, in **direction** suggested by critic
- Actor–critic algorithms follow an **approximate policy gradient**

$$\begin{aligned}
\nabla_\theta J(\theta) &\approx \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_w(s,a)] \\
\Delta\theta &= \alpha \nabla_\theta \log \pi_\theta(a|s) Q_w(s,a)
\end{aligned}$$

Estimating the Action–Value Function

- The critic is solving a familiar problem: **policy evaluation**
- How good is policy π_θ for current parameters θ?
  - Monte Carlo policy evaluation
  - Temporal–Difference learning
  - TD(λ)
- Could also use e.g., **least–squares policy evaluation**

Action–Value Actor–Critic

- Simple actor–critic algorithm based on **action–value critic**
- Using linear value function approximation $Q_w(s, a) = \phi(s, a)^\intercal w$
  - **Critic**: Updates w by linear TD(0)
  - **Actor**: Updates θ by policy gradient

function QAC()
    Initialize s, θ
    Sample $a \sim \pi_\theta$
    for all step **do**
        Sample reward $r = R(s, a)$; sample transition $s' \sim P(\cdot|s, a)$
        Sample action $a' \sim \pi_\theta(s', a')$
        $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$
        $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$
        $w \leftarrow w + \beta \delta \phi(s, a)$
        $a \leftarrow a', s \leftarrow s'$
    end for
end function

Bias in Actor–Critic Algorithms

- Approximating the policy gradient introduces **bias**
- A biased policy gradient may **not** find the right solution
- Luckily, if we choose action–value function approximation **carefully**
- Then we can **avoid** introducing any bias
- i.e., We can still follow the **exact** policy gradient

Compatible Function Approximation

Theorem (Compatible Function Approximation Theorem)

If the following two conditions are satisfied:

1. *Value function approximation is compatible to the policy*

$$\nabla_w Q_w(s, a) = \nabla_\theta \log \pi_\theta(a|s)$$

2. *Value function parameters w minimize the mean–squared error*

$$\epsilon = \mathbb{E}_{\pi_\theta}[(Q^{\pi_\theta}(s, a) - Q_w(s, a))^2]$$

Then the policy gradient is exact

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_w(s, a)]$$

Proof of Compatible Function Approximation Theorem

If w is chosen to **minimize** mean–squared error, gradient of ϵ w.r.t. w must be zero:

$$\begin{aligned}
\nabla_w \epsilon &= 0 \\
\mathbb{E}_{\pi_\theta}[(Q^{\pi_\theta}(s,a) - Q_w(s,a))\nabla_w Q_w(s,a)] &= 0 \\
\mathbb{E}_{\pi_\theta}[(Q^{\pi_\theta}(s,a) - Q_w(s,a))\nabla_\theta \log \pi_\theta(a|s)] &= 0 \\
\mathbb{E}_{\pi_\theta}[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)] &= \mathbb{E}_{\pi_\theta}[Q_w(s,a)\nabla_\theta \log \pi_\theta(a|s)]
\end{aligned}$$

So $Q_w(s,a)$ can be substituted directly into the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_w(s,a)]$$

All–Action Gradient

- By integrating over all possible actions in a state, the gradient becomes

$$\begin{aligned}\nabla_\theta J(\theta) &= \int_{\mathcal{S}} d^{\pi_\theta}(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a|s) Q_w(s,a) da ds \\ &= \int_{\mathcal{S}} d^{\pi_\theta}(s) \int_{\mathcal{A}} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^{\mathrm{T}} w da ds \\ &= F(\theta) w\end{aligned}$$

- It can be shown that the **all–action matrix** $F(\theta)$ is equal to the Fisher information matrix $G(\theta)$

$$\begin{aligned}G(\theta) &= \int_{\mathcal{S}} d^{\pi_\theta}(s) \int_{\mathcal{A}} \pi_\theta(a|s) \nabla_\theta \log\left(d^{\pi_\theta}(s)\pi_\theta(a|s)\right) \nabla_\theta \log\left(d^{\pi_\theta}(s)\pi_\theta(a|s)\right)^{\mathrm{T}} da ds \\ &= \int_{\mathcal{S}} d^{\pi_\theta}(s) \int_{\mathcal{A}} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^{\mathrm{T}} da ds \\ &= F(\theta)\end{aligned}$$

Reducing Variance Using a Baseline

- We subtract a **baseline function** $B(s)$ from the policy gradient
- This can **reduce variance**, without changing expectation

$$\begin{aligned}
\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)B(s)] &= \int_{\mathcal{S}} d^{\pi_\theta}(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a|s)B(s)dads \\
&= \int_{\mathcal{S}} d^{\pi_\theta} B(s) \nabla_\theta \int_{\mathcal{A}} \pi_\theta(a|s)dads \\
&= 0
\end{aligned}$$

- A **good** baseline is the state value function $B(s) = V^{\pi_\theta}(s)$
- So we can rewrite the policy gradient using the **advantage function** $A^{\pi_\theta}(s,a)$

$$\begin{aligned}
A^{\pi_\theta}(s,a) &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s,a)]
\end{aligned}$$

Estimating the Advantage Function

- The compatible function approximator is **mean–zero**!

$$\int_{\mathcal{A}} \nabla_\theta \log \pi_\theta(a|s) w da = \int_{\mathcal{A}} \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} w da = 0$$

- So the critic should really estimate the **advantage function**
- The advantage function can significantly **reduce variance** of policy gradient
- Traditional value function learning methods (e.g., TD) **cannot be applied**
- Using **two** function approximators and **two** parameter vectors

$$\begin{aligned} V_v(s) &\approx V^{\pi_\theta}(s) \\ Q_w(s,a) &\approx Q^{\pi_\theta}(s,a) \\ A(s,a) &= Q_w(s,a) - V_v(s) \end{aligned}$$

- And **updating** both value functions by e.g., TD learning

Estimating the Advantage Function

- For the true value function $V^{\pi_\theta}(s)$, the TD–error δ^{π_θ}

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

- is an **unbiased** estimate of the advantage function

$$\begin{aligned} \mathbb{E}_{\pi_\theta}[\delta^{\pi_\theta}] &= \mathbb{E}_{\pi_\theta}[r + \gamma V^{\pi_\theta}(s')|s,a] - V^{\pi_\theta}(s) \\ &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\ &= A^{\pi_\theta}(s,a) \end{aligned}$$

- So we can use the **TD error** to compute the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\delta^{\pi_\theta}]$$

- In practice we can use an **approximate TD error**

$$\delta_v = r + \gamma V_v(s') - V_v(s)$$

- This approach only requires **one** set of critic parameters v

Actors at Different Time–Scales

- As the critic, also the actor can estimate policy gradient at many time–scales

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s,a)]$$

- **Monte–Carlo** policy gradient uses error from complete return

$$\Delta\theta = \alpha(v_t - V_v(s_t))\nabla_\theta \log \pi_\theta(a_t|s_t)$$

- Actor–critic policy gradient uses **one–step TD error**

$$\Delta\theta = \alpha(r + \gamma V_v(s_{t+1}) - V_v(s_t))\nabla_\theta \log \pi_\theta(a_t|s_t)$$

Policy Gradient with Eligibility Traces

- Just like **forward–view** TD(λ), we can mix over time–scales

$$\Delta\theta = \alpha({\color{red}v_t^\lambda} - V_v(s_t))\nabla_\theta \log \pi_\theta(a_t|s_t)$$

- where $v_t^\lambda - V_v(s_t)$ is a **biased** estimate of advantage function
- Like **backward–view** TD(λ), we can also use eligibility traces
- By equivalence with TD(λ), substituting $\phi(s) = \nabla_\theta \log \pi_\theta(a|s)$

$$\begin{aligned}
\delta &= r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)\\
e_{t+1} &= \lambda e_t + \nabla_\theta \log \pi_\theta(a|s)\\
\Delta\theta &= \alpha\delta e_t
\end{aligned}$$

- This update can be applied **online**, to **incomplete sequences**

Alternative Policy Gradient Directions

- Gradient ascent algorithms can follow **any** ascent direction
- A good ascent direction can significantly **speed convergence**
- Also, a policy can often be **re–parameterized** without changing action probabilities
- For example, increasing score of all actions in a softmax policy
- The vanilla gradient is **sensitive** to these re–parameterization

Natural Policy Gradient

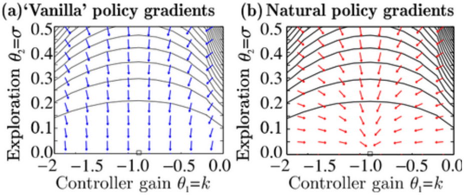
- A more efficient gradient in learning problems is the **natural gradient**
- It finds ascent direction that is closest to vanilla gradient, when changing policy by a small, fixed amount

$$\tilde{\nabla}_\theta J(\theta) = G^{-1}(\theta)\nabla_\theta J(\theta)$$

- Where $G(\theta)$ is the **Fisher information matrix**

$$G(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s)^{\mathrm{T}}]$$

- Natural policy gradients are **independent** of the chosen policy parameterization
- They correspond to **steepest ascent in policy space** and not in the parameter space
- **Convergence to a local minimum** is guaranteed



Natural Actor Critic

- Using **compatible** function approximation

$$\nabla_w A_w(s,a) = \nabla_\theta \log \pi_\theta(a|s)$$

- So the natural policy gradient **simplifies**

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s,a)] \\
&= \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\mathrm{T} w] \\
&= G(\theta) w \\
\tilde{\nabla}_\theta J(\theta) &= w
\end{aligned}$$

- i.e., update actor parameters in direction of critic parameters

$$\theta_{t+1} \leftarrow \theta_t + \alpha_t w_t$$

Episodic Natural Actor Critic

- Critic: Episodic Evaluation
  - Sufficient Statistics

$$\Phi = \begin{bmatrix} \phi_1 & \phi_2 & \dots & \phi_N \\ 1 & 1 & \dots & 1 \end{bmatrix}^{\mathrm{T}}$$

$$R = \begin{bmatrix} R_1 & R_2 & \dots & R_N \end{bmatrix}^{\mathrm{T}}$$

  - Linear Regression

$$\begin{bmatrix} w \\ J \end{bmatrix} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}R$$

- Actor: Natural Policy Gradient Improvement

$$\theta_{t+1} = \theta_t + \alpha_t w_t$$

Learning Ball in a Cup

Ball in a cup