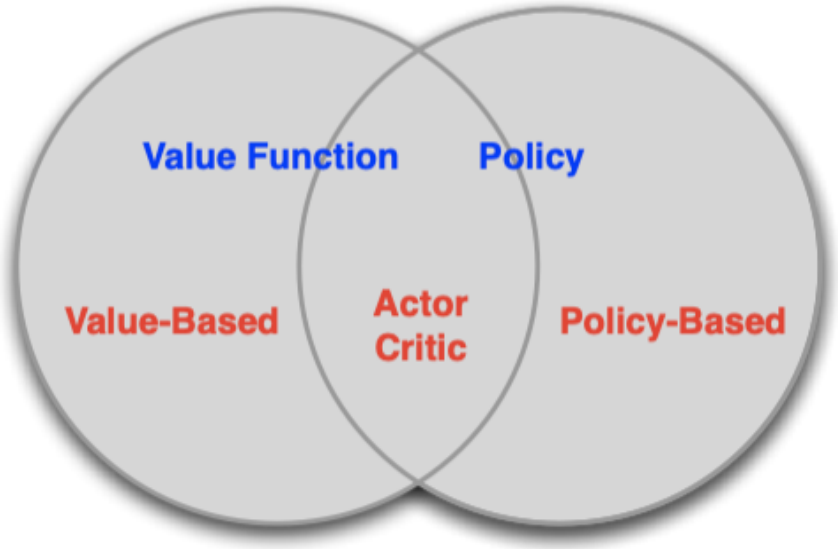
RL methods

- Value-based:
  - Learns value function
  - Implicit policy (e.g. greedy)
- Policy-based:
  - No value function
  - Learns policy
- Actor Critic
  - Learns value function
  - Learns policy



Policy Gradient

- Previously, we approximated value of action-value function using parameter θ,
  - e.g., deep Q learning
- Now we will directly parametarize and optimize the policy: $\pi_\theta(s, a) = P(a|s, \theta)$
- The objective J becomes a function over θ!
- Policy Gradient Descent!

Why Policy Gradient?

- Advantages:
  - Can learn stochastic policies
- Effective in high-dimensional or continuous action spaces
  - (see deterministic Policy Gradient)
- Better convergence properties
- Disadvantages
  - Typically converge to a local rather than global optimum
  - Evaluating a policy is typically inefficient and high variance

The objective function

- The expected reward (as the objective):

$$J(\theta) = \sum_{s \in S} d^{\pi}(s) V^{\pi}(s) = \sum_{s \in S} d^{\pi}(s) \left(\sum_{a \in A} \pi_{\theta}(a|s) Q^{\pi}(s,a) \right)$$

where $d^{\pi}(s) = lim_{t \to \infty} p(S_t = s | s_0, \pi_{\theta})$ is the *stationary distribution* of Markov chain when the agent starts from s_0 and following policy π_θ for *t* steps.

stationary distribution means when have the infinite time steps, the probability of $S_t = s$ *is stable* $p(S_t = s|s_0, \pi_\theta) = p(S_{t+1} = s|s_0, \pi_\theta)$*, when given initial state and policy.*

Policy Gradient Theorem

Policy Gradient Methods for Reinforcement Learning with Function Approximation

Richard S. Sutton, David McAllester, Satinder Singh, Yishay Mansour
AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932

Abstract

Function approximation is essential to reinforcement learning, but the standard approach of approximating a value function and determining a policy from it has so far proven theoretically intractable. In this paper we explore an alternative approach in which the policy is explicitly represented by its own function approximator, indepen-

Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." *Advances in neural information processing systems*. 2000.

Policy Gradient Theorem

- **Policy Gradient Theorem**: For an MDP, the gradient of expected reward

$$\nabla_\theta J(\theta) = \sum_{s \in S} d^\pi(s) \left(\sum_{a \in A} Q^\pi(s,a) \nabla_\theta \pi_\theta(a|s) \right)$$

- It provides a nice reformation of the derivative of the objective function that not involves the derivative of the state distribution $\frac{\partial d^\pi(s)}{\partial \theta}$

Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." *Advances in neural information processing systems*. 2000.

Policy Gradient Theorem: the proof

$$\nabla_\theta\left(\mathrm{E}\left[\sum_{k=1}^{\infty}\gamma^{k-1}r_{t+k}\middle|s_t=s_0,\pi\right]\right)=\nabla_\theta V^\pi(s_0)$$

$$=\nabla_\theta\left(\sum_{a\in A}Q^\pi(s_0,a)\pi_\theta(a|s_0)\right)$$

$$=\sum_{a\in A}\left(Q^\pi(s_0,a)\nabla_\theta\pi_\theta(a|s_0)+\pi_\theta(a|s_0)\nabla_\theta Q^\pi(s_0,a)\right)$$ **Product rule**

$$=\sum_{a\in A}\left(Q^\pi(s_0,a)\nabla_\theta\pi_\theta(a|s_0)+\pi_\theta(a|s_0)\nabla_\theta\left(\sum_{s',r}P(s',r|s,a)(r+V^\pi(s'))\right)\right)$$ **Extend Q^π for future state**

$$=\sum_{a\in A}\left(Q^\pi(s_0,a)\nabla_\theta\pi_\theta(a|s_0)+\pi_\theta(a|s_0)\left(\sum_{s',r}P(s',r|s_0,a)\nabla_\theta V^\pi(s')\right)\right)$$ **State transition and r are independent of θ**

$$=\sum_{a\in A}\left(Q^\pi(s_0,a)\nabla_\theta\pi_\theta(a|s_0)+\pi_\theta(a|s_0)\left(\sum_{s'}P(s'|s_0,a)\nabla_\theta V^\pi(s')\right)\right)$$ **Marginalize r out**

$$=\sum_{a\in A}\left(Q^\pi(s_0,a)\nabla_\theta\pi_\theta(a|s_0)\right)+\sum_{a\in A}\pi_\theta(a|s_0)\left(\sum_{s'}P(s'|s_0,a)\nabla_\theta V^\pi(s')\right)$$

$$=\sum_{a\in A}\left(Q^\pi(s_0,a)\nabla_\theta\pi_\theta(a|s_0)\right)+\sum_{s'}\left(\sum_{a\in A}\pi_\theta(a|s_0)P(s'|s_0,a)\right)\nabla_\theta V^\pi(s')$$

Policy Gradient Theorem: the proof

We thus have the following recursive form of the gradient:

$$\nabla_\theta V^\pi(s_0) = \sum_{a\in A}\left(Q^\pi(s_0,a)\nabla_\theta \pi_\theta(a|s_0)\right) + \sum_{s'}\left(\sum_{a\in A}\pi_\theta(a|s_0)P(s'|s_0,a)\right)\nabla_\theta V^\pi(s')$$

Equivalently we have:

$$\nabla_\theta V^\pi(s_0) = \varphi(s_0) + \sum_{s'} P^\pi(s'|s_0)\nabla_\theta V^\pi(s')$$

where $\varphi(s_0) \equiv \sum_{a\in A}\left(Q^\pi(s_0,a)\nabla_\theta \pi_\theta(a|s_0)\right)$ and Markov transition $P^\pi(s'|s_0) \equiv \sum_{a\in A}\pi_\theta(a|s_0)P(s'|s_0,a)$

Policy Gradient Theorem: the proof

We now consider the following visitation sequence:

$$s_0 \xrightarrow{a\sim\pi_\theta(.|s_0)} s' \xrightarrow{a\sim\pi_\theta(.|s')} s'' \xrightarrow{a\sim\pi_\theta(.|s'')} \dots \xrightarrow{a\sim\pi_\theta(.|.)} s$$

and denote the probability of transitioning from state s_0 to state s with policy π^θ after k step as

$$P^\pi(s''|s_0,k) \equiv \sum_{s'} P^\pi(s''|s',k-1)P^\pi(s'|s_0)\,,$$

where we have $P^\pi(s|s_0,0) = 1$ if $s = s_0$ and $P^\pi(s|s_0,0) = 0$ if $s \neq s_0$ (because it happened already!)

Let us now go back to unroll the gradient of the value function:

$$\nabla_\theta V^\pi(s_0) = \varphi(s_0) + \sum_{s'} P^\pi(s'|s_0)\nabla_\theta V^\pi(s')$$

Policy Gradient Theorem: the proof

Let us now go back to unroll the gradient of the value function:

$$\nabla_\theta V^\pi(s_0) = \varphi(s_0) + \sum_{s'} P^\pi(s'|s_0)\nabla_\theta V^\pi(s')$$

$$= \varphi(s_0) + \sum_{s'} P^\pi(s'|s_0,1)[\varphi(s') + \sum_{s''} P^\pi(s''|s')\nabla_\theta V^\pi(s'')]$$

$$= \varphi(s_0) + \left[\sum_{s'} P^\pi(s'|s_0,1)\varphi(s')\right] + [\sum_{s'} P^\pi(s'|s_0,1) \sum_{s''} P^\pi(s''|s')\nabla_\theta V^\pi(s'')]$$

$$= \varphi(s_0) + \left[\sum_{s'} P^\pi(s'|s_0,1)\varphi(s')\right] + [\sum_{s''}\sum_{s'} P^\pi(s'|s_0,1)P^\pi(s''|s')\nabla_\theta V^\pi(s'')]$$

$$= \varphi(s_0) + \left[\sum_{s'} P^\pi(s'|s_0,1)\varphi(s')\right] + [\sum_{s''} P^\pi(s'|s_0,2)\nabla_\theta V^\pi(s'')]$$

$$= \sum_{s\in S}\sum_{k=0}^{\propto} P^\pi(s|s_0,k)\varphi(s) = \sum_{s\in S}\sum_{k=0}^{\propto}[P^\pi(s|s_0,k) \sum_{a\in A}\left(Q^\pi(s,a)\nabla_\theta \pi_\theta(a|s)\right)]$$

Policy Gradient Theorem: the proof

If we define $d^{\pi}(s) \equiv \sum_{k=0}^{\infty} P^{\pi}(s|s_0, k)$ * as the non-normalized visitation probabilities of state s (starting from s_0), the following

$$\nabla_{\theta} V^{\pi}(s_0) = \sum_{s \in S} \sum_{k=0}^{\infty} [P^{\pi}(s|s_0, k) \sum_{a \in A} \left(Q^{\pi}(s, a) \nabla_{\theta} \pi_{\theta}(a|s)\right)]$$

becomes

$$\nabla_{\theta} V^{\pi}(s_0) = \sum_{s \in S} d^{\pi}(s) \sum_{a \in A} \left(Q^{\pi}(s, a) \nabla_{\theta} \pi_{\theta}(a|s)\right)$$

* Note that if a Markov chain is ergodic it holds that there is a unique stationary distribution of the chain regardless the starting state. Thus $d^{\pi}(s)$ is a unique (non-normalized) stationary visiting prob. regardless of s_0.

REINFORCE

In order to make an unbiased estimation, the gradient can be further written as:

$$\nabla_\theta V^\pi(s_0) = \sum_{s\in S} d^\pi(s) \sum_{a\in A} \left(Q^\pi(s,a)\nabla_\theta \pi_\theta(a|s)\right)$$

$$\propto \sum_{s\in S} \mu(s) \sum_{a\in A} \left(\pi_\theta(a|s) Q^\pi(s,a) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}\right) \qquad \mu(s) = \frac{d^\pi(s)}{\sum_{s'} d^\pi(s')}$$

$$= \mathrm{E}_{s\sim\mu, a\sim\pi}[Q^\pi(s,a)\nabla_\theta \mathrm{In}\pi_\theta(a|s)]$$

$$= \mathrm{E}_{s\sim\mu, a\sim\pi}[G_t \nabla_\theta \mathrm{In}\pi_\theta(a|s)]$$

where $\mathrm{E}_{s\sim d^\pi, a\sim\pi}[G_t|s,a] = Q^\pi(s,a)$

So the gradient update is $\theta_{t+1} = \theta_t + \alpha G_t \nabla_\theta \mathrm{In}\pi_\theta(a|s)$

Why it is Monte Carlo Policy Gradient?

REINFORCE uses the complete return from time t, which includes all future rewards up until the endo for the episode.

REINFORCE: Monte-Carlo Policy Gradient

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

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 $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

REINFORCE with Baseline

- The policy gradient theorem can be generalized to include an arbitrary baseline $b(s)$ to reduce the variance of Monte-Carlo.

$$\nabla_\theta J(\theta) \propto \sum_{s \in S} \mu(s) \left(\sum_{a \in A} (Q^\pi(s,a) - b(s)) \nabla_\theta \pi_\theta(a|s) \right)$$

- The baseline can be any function as long as it does not vary with a because

$$\sum_{a \in A} b(s) \nabla_\theta \pi_\theta(a|s) = b(s) \sum_{a \in A} \nabla_\theta \pi_\theta(a|s) = b(s) \nabla_\theta 1 = 0$$

- The update is:

$$\theta_{t+1} = \theta_t + \alpha (G_t - b(s)) \nabla_\theta \mathrm{In} \pi_\theta(a|s)$$

Function Approximation

- Consider $Q^{\pi}(a, s)$ is approximated by a learned function approximator
  - Let f_w : *S* x *A* ->*R* be our approximation to Q, with parameter *w*
  - It is natural to learn f_w by following

$$\Delta w_t \propto \frac{\partial}{\partial w}\left[\hat{Q}^{\pi}(s_t, a_t) - f_w(s_t, a_t)\right]^2$$

$$\propto \left[\hat{Q}^{\pi}(s_t, a_t) - f_w(s_t, a_t)\right]\frac{\partial f_w(s_t, a_t)}{\partial w}$$

where $\hat{Q}^{\pi}(s_t, a_t)$ is some unbiased estimator of Q.

  - the local optimum condition gives

$$\sum_{s \in S} d^{\pi}(s)\left(\sum_{a \in A} \pi_{\theta}(a|s)\left[\hat{Q}^{\pi}(s_t, a_t) - f_w(s_t, a_t)\right]\frac{\partial f_w(s_t, a_t)}{\partial w}\right) = 0$$

REINFORCE with Value Function Approximation and Baseline

REINFORCE with Baseline (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm 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):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Function Approximation

- Policy Gradient with Function Approximation:
  - If f_w satisfies the local optimum condition (previous slide) and is compatible with the policy parameterization as follows:

$$\frac{\partial f_w(s_t,a_t)}{\partial w} = \frac{\partial \pi_\theta(a|s)}{\partial \theta} \frac{1}{\pi_\theta(a|s)} = \frac{\partial ln \pi_\theta(a|s)}{\partial \theta}$$

  - then we have:

$$\nabla_\theta J(\theta) = \sum_{s \in S} d^\pi(s) \left(\sum_{a \in A} f_w(s_t, a_t) \nabla_\theta \pi_\theta(a|s) \right)$$

  - The proof can be done by combining the local optimum condition (previous page) and the compatibility condition

Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." *Advances in neural information processing systems*. 2000.

Function Approximation

- Given the policy is parameterized as follow:

$$\pi_\theta(a|s) = \frac{e^{\theta^T \varphi_{s,a}}}{\sum_b e^{\theta^T \varphi_{s,b}}}$$

Softmax form

(θ considered as the last layer in a NN)

– then we have:

$$\frac{\partial f_w(s_t, a_t)}{\partial w} = \frac{\partial \pi_\theta(a|s)}{\partial \theta} \frac{1}{\pi_\theta(a|s)} = \varphi_{s,a} - \sum_b \pi_\theta(b|s)\varphi_{s,b}$$

- So $f_w(s_t, a_t) = w^T[\varphi_{s,a} - \sum_b \pi_\theta(b|s)\varphi_{s,b}]$

Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." *Advances in neural information processing systems*. 2000.

Advantage Function

- We can rewrite the policy gradient using the advantage function $A^{\pi_\theta}(s, a)$

$$A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$$

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

- The advantage function can significantly reduce variance
- So the critic should really estimate the advantage function
- For example, by estimating both $V^{\pi_\theta}(s)$ and $Q^{\pi_\theta}(s, a)$
- Using two function approximators and two parameter vectors,

$$V_v(s) \approx V^{\pi_\theta}(s)$$

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

$$A(s, a) = Q_w(s, a) - V_v(s)$$

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

Advantage Function

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

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

- is an unbiased estimate of the advantage function

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

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

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

- In practice we can use an approximate TD error. This approach only requires one set of critic parameters v

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

One-step Actor-Critic

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

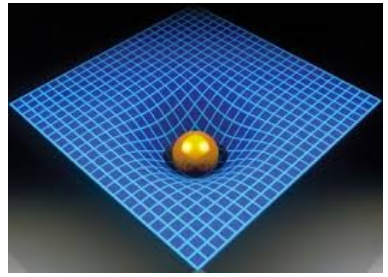
Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Comparison

- Unbias estimator of Q: Monte-Carlo Policy Gradient
- Functional approximate:
  - Learning Q functions directly
  - Advantage function: One-step Actor Critic, TD error

Natural Gradient

- Gradient descent take steps on parameter space, which consider every optimize direction is equal and take the same distance on parameter space.
- However, in practice, the underlying space of parameters is curved and distorted



Amari, Shun-Ichi, and Scott C. Douglas. "Why natural gradient?." *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*. Vol. 2. IEEE, 1998.

Natural Gradient Descent

- A trusted region approach:

$$\arg\min_{\Delta\theta} \mathcal{L}(\theta + \Delta\theta)$$
$$\text{s. t. } KL(p_\theta || p_{\theta+\Delta\theta}) = const.$$

- We:
  - Construct a surrogate function in trust region
  - Approximate in the original objective function in the neighborhood of θ
  - The neighborhood happens at policy space (KL divergence of policy)

Natural Gradient Descent

- A trusted region approach:

$$\arg\min_{\Delta\theta} \mathcal{L}(\theta + \Delta\theta)$$
$$\text{s. t. } KL(p_\theta || p_{\theta+\Delta\theta}) = const.$$

- As $\Delta\theta$ is small, we can approximate the KL divergence by its second order Taylor series:

$$\begin{aligned} KL(p_\theta \parallel p_{\theta+\Delta\theta}) &\approx (\mathbb{E}_{\mathbf{z}}[\log p_\theta] - \mathbb{E}_{\mathbf{z}}[\log p_\theta]) \\ &- \mathbb{E}_{\mathbf{z}}[\nabla \log p_\theta(\mathbf{z})]\,\Delta\theta - \frac{1}{2}\Delta\theta^T \mathbb{E}_{\mathbf{z}}\left[\nabla^2 \log p_\theta\right]\Delta\theta \\ &= \frac{1}{2}\Delta\theta^T \mathbb{E}_{\mathbf{z}}\left[-\nabla^2 \log p_\theta(\mathbf{z})\right]\Delta\theta \\ &= \frac{1}{2}\Delta\theta^T \mathbf{F}\Delta\theta \end{aligned}$$

$$\mathbb{E}_{\mathbf{z}}[\nabla \log p_\theta(\mathbf{z})] = \sum_{\mathbf{z}}\left(p_\theta(\mathbf{z})\frac{1}{p_\theta(\mathbf{z})}\frac{\partial p_\theta(\mathbf{z})}{\partial\theta}\right) = \frac{\partial}{\partial\theta}\left(\sum_\theta p_\theta(\mathbf{z})\right) = \frac{\partial 1}{\partial\theta} = 0.$$

Pascanu, Razvan, and Yoshua Bengio. "Revisiting natural gradient for deep networks." *arXiv preprint arXiv:1301.3584* (2013)

Natural Gradient Descent

- With the approximated KL, we now use a Lagrangian, where L(θ + Δθ) by its first order Taylor series L(θ) + ∇L(θ)Δθ:

L (θ + Δθ)

KL term

$$\mathcal{L}(\theta) + \nabla\mathcal{L}(\theta)\Delta\theta + \frac{1}{2}\lambda\Delta\theta^T\mathbf{F}\Delta\theta$$

Lagrangian multiplier

Calculate the gradient of with respect to Δθ, the optimal value is achieved when gradient = 0

- which gives:

$$\nabla_N\mathcal{L}(\theta) \stackrel{\text{def}}{=} \nabla\mathcal{L}(\theta)\mathbb{E}_{\mathbf{z}}\left[\left(\nabla \log p_\theta(\mathbf{z})\right)^T\left(\nabla \log p_\theta(\mathbf{z})\right)\right]^{-1}$$

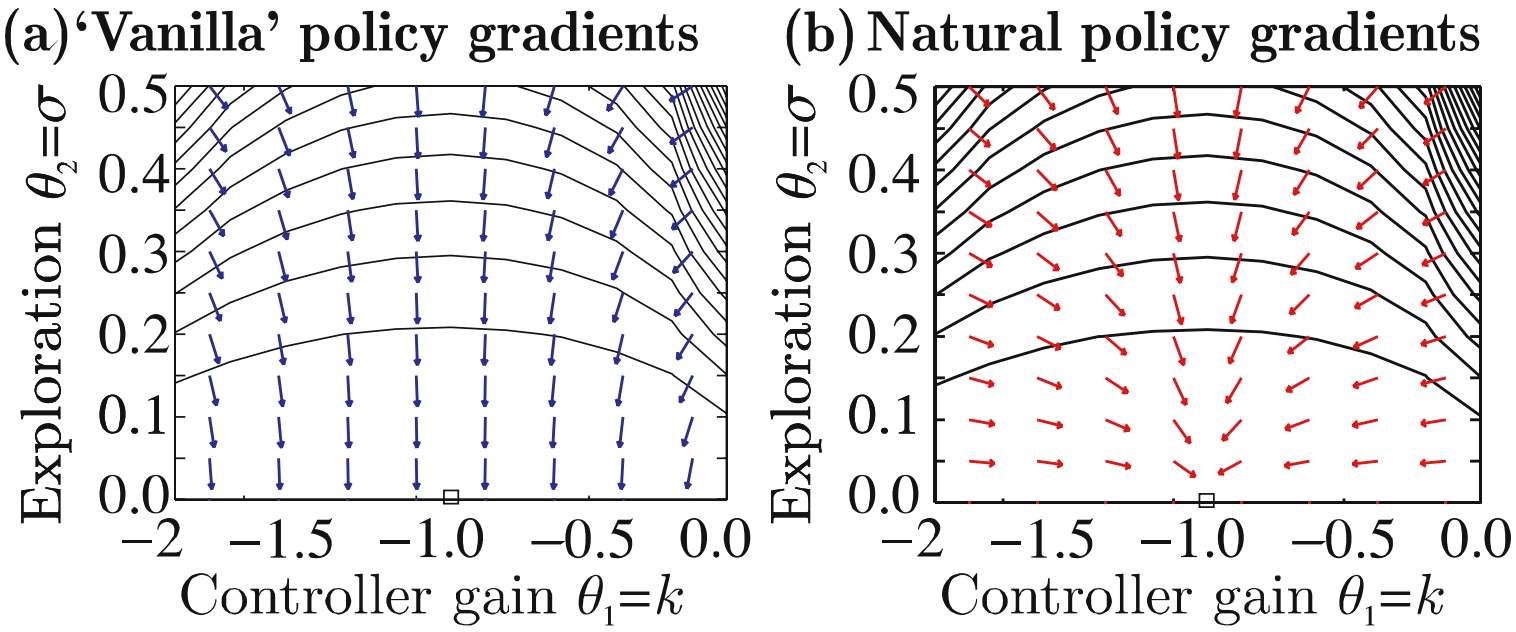
$$\stackrel{\text{def}}{=} \nabla\mathcal{L}(\theta)\mathbf{F}^{-1}.$$

The Fisher Information Matrix form can be obtained from the expected value of the Hessian

Pascanu, Razvan, and Yoshua Bengio. "Revisiting natural gradient for deep networks." *arXiv preprint arXiv:1301.3584* (2013)

Natural Policy Gradient

- Policy gradient methods may be quite inefficient partially caused by the large plateaus in the expected return landscape
  - where the gradients are small and often do not point directly towards the optimal solution.



1d linear quadratic regulation, the differences between 'vanilla' and natural policy gradients

Peters, Jan, and Stefan Schaal. "Natural actor-critic." *Neurocomputing* 71.7-9 (2008): 1180-1190.

Natural Policy Gradient

- The natural policy gradient is a second order optimization and works regardless of how the model is parameterized (model invariant)
  - It does not follow the steepest direction in parameter space but the steepest direction with respect to the Fisher metric given by

$$\nabla_\theta^{nat} \pi_\theta(s,a) = G_\theta^{-1} \nabla_\theta \pi_\theta(s,a)$$

  - where G_θ is the Fisher information matrix

$$G_\theta = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a)^T\right]$$

Peters, Jan, and Stefan Schaal. "Natural actor-critic." *Neurocomputing* 71.7-9 (2008)

Natural Actor-Critic

- Using compatible function approximation,

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

- So the natural policy gradient simplifies,

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)\right] \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a)^T w\right] \\
&= G_\theta w \\
\nabla_\theta^{nat} J(\theta) &= w
\end{aligned}$$

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

Compatibility

– If f_w satisfies the local optimum condition (previous slide) and is compatible with the policy parameterization as follows:

$$\frac{\partial f_w(s_t,a_t)}{\partial w} = \frac{\partial \pi_\theta(a|s)}{\partial \theta}\frac{1}{\pi_\theta(a|s)} = \frac{\partial ln \pi_\theta(a|s)}{\partial \theta}$$

– then we have:

$$\nabla_\theta J(\theta) = \sum_{s\in S} d^\pi(s) \left(\sum_{a\in A} f_w(s_t,a_t) \nabla_\theta \pi_\theta(a|s)\right)$$

Peters, Jan, and Stefan Schaal. "Natural actor-critic." *Neurocomputing* 71.7-9 (2008)

Practical Implementation of Natural Gradient Descent

- Trust region policy optimization (TRPO)

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}(\theta_k, \theta)$$
$$\text{s.t. } \bar{D}_{KL}(\theta||\theta_k) \leq \delta$$

$$\mathcal{L}(\theta_k, \theta) = \mathop{\mathrm{E}}_{s,a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a)\right]$$

$$\bar{D}_{KL}(\theta||\theta_k) = \mathop{\mathrm{E}}_{s\sim\pi_{\theta_k}}\left[D_{KL}\left(\pi_\theta(\cdot|s)||\pi_{\theta_k}(\cdot|s)\right)\right].$$

- With Taylor expansion over objective and constraints, we can apply Lagrangian duality to yield the solution

$$\mathcal{L}(\theta_k, \theta) \approx g^T(\theta - \theta_k)$$
$$\bar{D}_{KL}(\theta||\theta_k) \approx \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k)$$

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g.$$

Practical Implementation of Natural Gradient Descent

- Proximal Policy Optimization (PPO)
  - Approximate TRPO by taking clipping
  - Serves as a regularizer by removing incentives for the policy to change dramatically

$$\theta_{k+1} = \arg\max_{\theta} \underset{s,a\sim\pi_{\theta_k}}{\mathrm{E}} \left[L(s, a, \theta_k, \theta) \right],$$

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \quad \mathrm{clip} \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right),$$

Take multiple gradient steps to maximize the objective function.

References

- Peters, Jan, and Stefan Schaal. "Natural actor-critic." *Neurocomputing* 71.7-9 (2008)
- Pascanu, Razvan, and Yoshua Bengio. "Revisiting natural gradient for deep networks." *arXiv preprint arXiv:1301.3584* (2013)
- Amari, Shun-Ichi, and Scott C. Douglas. "Why natural gradient?." *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*. Vol. 2. IEEE, 1998.
- Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." *Advances in neural information processing systems*. 2000.