Reinforcement Learning

Policy Search

Continuous state/action space, policy gradient, natural policy gradient, actor-critic, natural actor-critic

Vien Ngo
MLR, University of Stuttgart

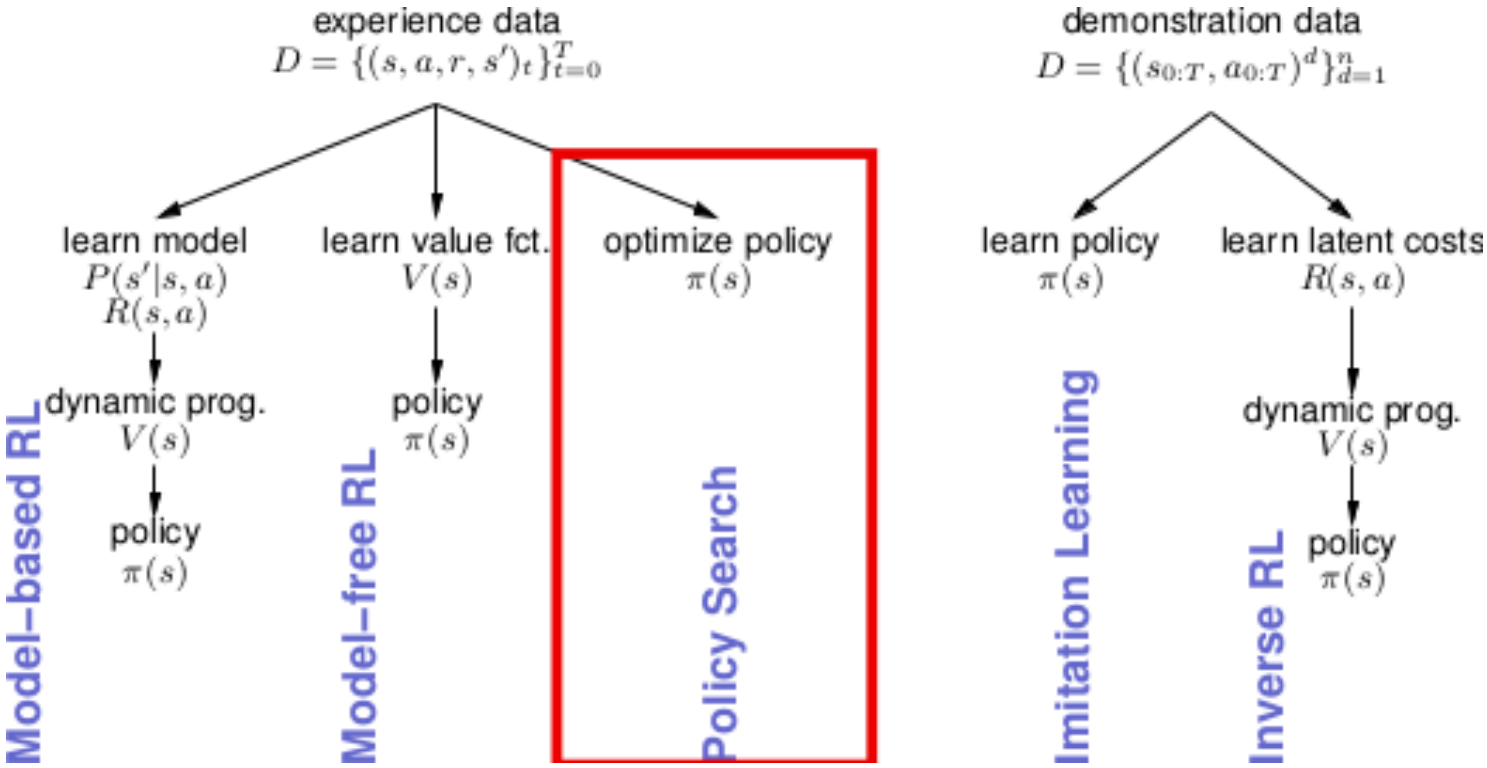
Outline

- Policy search overview: exploration, evaluation, updates
- Policy gradient methods
  – Plain policy gradient
  – Second-order policy gradient: Gauss-Newton, Natural gradient
  – Actor-critic, natural actor-critic.
- Inference-based methods
- Information-theoretic methods
- Global optimization methods

Missing

- Bayesian policy gradient methods
- Model-based policy search

Policy Search



Policy Search: motivations

- Value-based methods:
  – Inefficient in high-dimensional space
  – Value function approximation suffers from bias due to the use of bootstrapping (value updates based on current estimates)
  – The value function (using function approximation) might be discontinuous (e.g. in under-actuated systems).

Policy Search: motivations

- Value-based methods:
  – Inefficient in high-dimensional space
  – Value function approximation suffers from bias due to the use of bootstrapping (value updates based on current estimates)
  – The value function (using function approximation) might be discontinuous (e.g. in under-actuated systems).

- Advantages:
  – Guaranteed convergence properties
  – Learning in high-dimensional space problems
  – Enable imitation learning, learning from demonstration effectively
  – Integrate other optimal control policies straightforwardly

- Disadvantages: computationally inefficient

Policy Search

- Directly optimize over the parameter space $\theta \in \Theta$ of paramterized policies. For example,
  – Linear policies: $\pi(s) = \theta^\top \phi(s)$
  – Stochastic Gaussian policies $\pi(a|s, \theta) \sim e^{-f(s,a)}$
    - Linear parametrization: $f(s, a) = \frac{1}{2}\big(W\phi(s) - a\big)^\top \Sigma^{-1}\big(W\phi(s) - a\big)$, where assuming that $a \in R^n$, each $\theta_i \in \Theta$ for each action dimension, and

$$W = \begin{bmatrix} \theta_1^\top \\ \cdots \\ \theta_n^\top \end{bmatrix}$$

      hence $\theta = (W, \Sigma)$
    - Functional policy: $f(s, a) = \frac{1}{2}\big(h(s) - a\big)^\top \Sigma^{-1}\big(h(s) - a\big)$; hence $\theta = (h, \Sigma)$

  – (Recurrent) neural network: $a = f(\theta, s)$ (e.g. autonomous helicopter, AlphaGO, robotic control tasks, etc.)

Policy Search

Algorithm 1 Model-Free Policy Search

1: **while** (!converged) **do**

2: EXPLORATION: Generating trajectories $\mathcal{D} = \{\tau_i\}$ using $\pi(\theta_k)$

3: EVALUATION: Evaluating the quality of $\pi(\theta_k; \mathcal{D})$

4: UPDATE: Updating $\pi(\theta_{k+1})$ given the evaluations and trajectories.

(Deisenroth, Neumann and Peters, 2011)

Exploration Strategies

Exploration in Model-Free Policy Search

- Exploration in Action Space versus Exploration in Parameter Space
- Episode-based versus Step-based Exploration
- Uncorrelated versus Correlated Exploration

Exploration in Model-Free Policy Search

- Exploration in action space: an exploration noise is added directly to the executed actions (e.g. REINFORCE, eNAC)
  e.g. exploration at each step (e.g. PoWER, PI^2): $u_t = \phi_t(x)^\top \theta$

$$\pi_\theta(u_t|x_t) = \mathcal{N}\big(u_t|\theta^\top \phi(x_t), \phi(x_t)^\top \Sigma \phi(x_t)\big)$$

- Exploration in parameter sapce: the parameter is perturbed in the beginning of an episode (or can be used at each time step).

Exploration in Model-Free Policy Search

- Uncorrelated Exploration: Σ_θ is diagonal
- Correlated Exploration: Σ_θ is full. This stragegy often gives better learning speed, however requires large amount of data to accurately estimate Σ (CMA-ES is an efficient algorithm, but should not be used when $|\Theta| > 50$)

Policy Evaluation Strategies

- Step-based Policy Evaluation
- Episode-based Policy Evaluation

Step-based Policy Evaluation

Given a set of sampled trajectoris $\mathcal{D} = \left\{\xi^i = (x_t^i, u_t^i)_{t=0}^{T_i}\right\}_{i=1}^{M}$

- the expected cost-to-go of (x_t^i, u_t^i)

$$Q_t^i = Q_t^\pi(x_t^i, u_t^i) = \mathbb{E}_{\pi_\theta(\xi)}\left[\sum_{k=t}^{T_i} c(x_k, u_k) \middle| x_t = x_t^i, u_t = u_t^i\right]$$
$$\approx \sum_{k=t}^{T_i} c(x_k^i, u_k^i)$$

- step-based update is based on a data set $\mathcal{D}' = \{\{x_t^i, u_t^i, Q_t^i\}_{t=0}^{T_i}\}_{i=1}^{M}$
- algorithms: REINFORCE, G(PO)MDP, NAC, eNAC, PoWER, PI^2.

Episode-based Policy Evaluation

Given a set of sampled trajectoris $\mathcal{D} = \left\{\xi^i = (x_t^i, u_t^i)_{t=0}^{T_i}\right\}_{i=1}^{M}$

- using the expected cost of a parameter θ^i

$$C^i = C(\theta^i) = \mathbb{E}_{\pi_\theta(\xi)}\left[\sum_{k=0}^{T_i} c(x_k, u_k) \middle| \theta = \theta^i\right]$$

- episode-based update is based on a data set $\mathcal{D}' = \{\theta^i, C^i\}_{i=1}^{M}$
- algorithms: Episode-based REPS, Episode-based PI^2, PEPG, NES, CMA-ES, RWR

Step-based vs. Episode-based

- Episode-based might suffer from a large variance of the estimated accumulated costs.
- Episode-based does exploration directly in parameter space of the policy.
- The choice between two depends on the given problem.

Policy Update Strategies

Policy Update Strategies

- Policy gradient methods: Finite Difference methods, REINFORCE, G(PO)MDP, Natural Gradient, Natural Actor-Critic (NAC), episodic NAC, etc.
- Expectation-Maximization Policy search
- Information-theoretic techniques
- Bayesian optimization techniques

Policy gradient

Policy gradient

- Family of randomized policy $\mu(s, a) = p(a|s)$ (deterministic policy is a special case).
- A performance measure is

$$J(\mu) = E_\mu \left[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \right]$$

Policy gradient

- Family of randomized policy $\mu(s,a) = p(a|s)$ (deterministic policy is a special case).
- A performance measure is

$$J(\mu) = E_\mu\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots\right]$$

- Parameterized policy family $\mu_\theta(s,a)$ by a parameter space $\theta \in \Re^d$.
- The parametric performance measure becomes

$$J(\theta) = E_\theta\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots\right]$$

Policy gradient

- Family of randomized policy $\mu(s,a) = p(a|s)$ (deterministic policy is a special case).
- A performance measure is

$$J(\mu) = E_\mu\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots\right]$$

- Parameterized policy family $\mu_\theta(s,a)$ by a parameter space $\theta \in \Re^d$.
- The parametric performance measure becomes

$$J(\theta) = E_\theta\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots\right]$$

- Solution: gradient-ascent algorithms

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k)$$

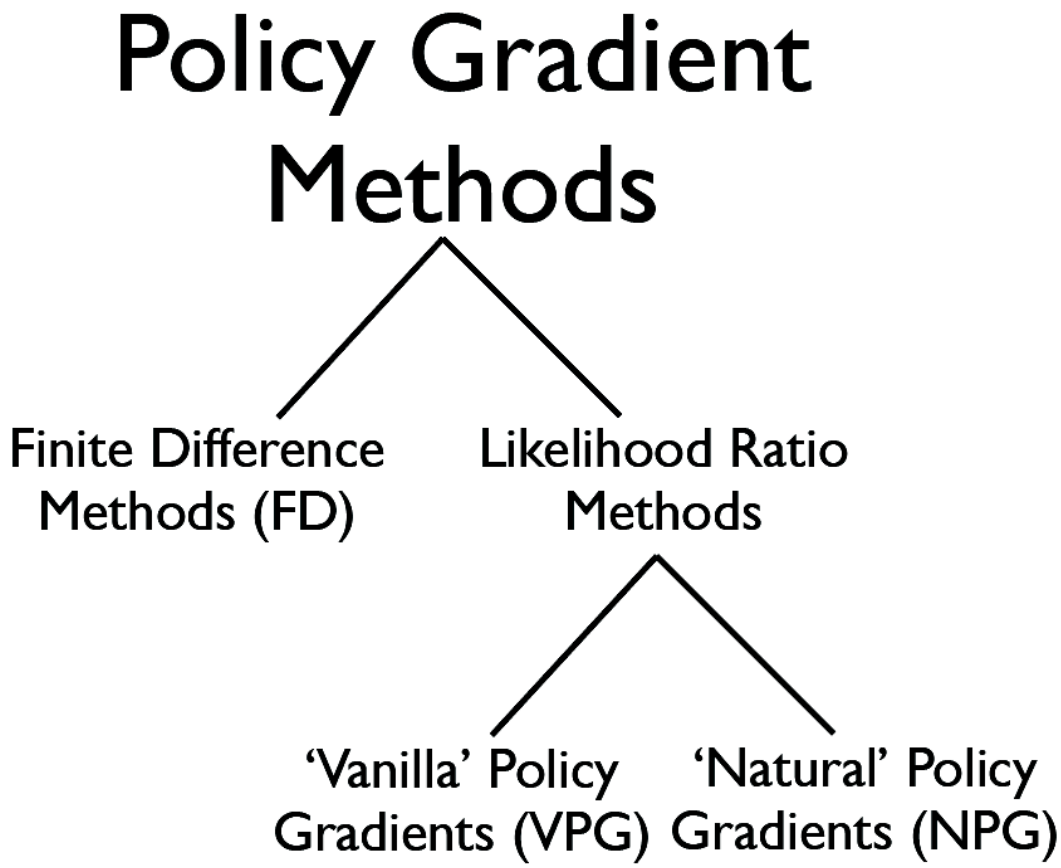
Policy gradient

- Policy gradient updates

$$\theta_{\mu'} = \theta_{\mu} + \alpha \nabla_{\theta} J(\theta_{\mu})$$

- Guarantee the performance improvement: $J(\theta_{\mu'}) \geq J(\theta_{\mu}) \Rightarrow \mu'$ at least better than or equal to μ

Policy gradient



(from Jan Peters' tutorial at ICRA 2012)

Policy gradient: Black-box approaches

- Approximate the gradient using supervised learning (regression).

Policy gradient: Black-box approaches

- Approximate the gradient using supervised learning (regression).
- Collect data $\mathcal{D} = \{\delta\theta_i, \delta J_i\}_{i=1}^{M}$ (the sampled gradients). By
  – perturbing the parameters: $\theta + \delta\theta$
  – applying the new policy $\mu(\theta + \delta\theta)$ to get $\delta J_i = J(\theta + \delta\theta) - J(\theta)$

Policy gradient: Black-box approaches

- Approximate the gradient using supervised learning (regression).
- Collect data $\mathcal{D} = \{\delta\theta_i, \delta J_i\}_{i=1}^{M}$ (the sampled gradients). By
  – perturbing the parameters: $\theta + \delta\theta$
  – applying the new policy $\mu(\theta + \delta\theta)$ to get $\delta J_i = J(\theta + \delta\theta) - J(\theta)$
- the finite different (FD) gradient estimation by regression

$$g_{FD}(\theta) = (\Delta\Theta^\top \Delta\Theta)^{-1} \Delta\Theta^\top \Delta J$$

where

$$\Delta\Theta = \begin{bmatrix} \delta\theta_1^\top \\ \dots \\ \delta\theta_M^\top \end{bmatrix}, \Delta J = \begin{bmatrix} \delta J_1 \\ \dots \\ \delta J_M \end{bmatrix}$$

Policy gradient: Black-box approaches

- Approximate the gradient using supervised learning (regression).
- Collect data $\mathcal{D} = \{\delta\theta_i, \delta J_i\}_{i=1}^{M}$ (the sampled gradients). By
  – perturbing the parameters: $\theta + \delta\theta$
  – applying the new policy $\mu(\theta + \delta\theta)$ to get $\delta J_i = J(\theta + \delta\theta) - J(\theta)$
- the finite different (FD) gradient estimation by regression

$$g_{FD}(\theta) = (\Delta\Theta^\top \Delta\Theta)^{-1} \Delta\Theta^\top \Delta J$$

where

$$\Delta\Theta = \begin{bmatrix} \delta\theta_1^\top \\ \dots \\ \delta\theta_M^\top \end{bmatrix}, \Delta J = \begin{bmatrix} \delta J_1 \\ \dots \\ \delta J_M \end{bmatrix}$$

- gradient update

$$\theta \leftarrow \theta + \alpha g_{FD}(\theta)$$

(Jan Peters, Scholarpedia 2010)

- This method works well if the return J is not too noisy.

Policy gradient: Likelihood Ratio Gradient

directly exploit the Markov property

Policy gradient: Likelihood Ratio Gradient

- rewrite the performance measure $J(\theta)$

$$\begin{aligned} J(\theta) &= E_\theta\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots\right] \\ &= \int p(\xi|\mu_\theta) R(\xi) d\xi \end{aligned}$$

where $\xi = \{s_0, a_0, r_0, s_1, a_1, r_1, \dots\}$ is a trajectory.

$$\begin{aligned} R(\xi) &= r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \\ p(\xi|\mu_\theta) &= p(s_0) \prod p(s_{t+1}|s_t, a_t) \mu_\theta(a_t|s_t) \end{aligned}$$

Policy gradient: Likelihood Ratio Gradient

- Gradient derivation

$$\begin{aligned}\nabla_\theta J(\theta) &= \int \nabla_\theta p(\xi|\mu_\theta) R(\xi) d\xi \\ &= \int p(\xi|\mu_\theta) \nabla_\theta \log p(\xi|\mu_\theta) R(\xi) d\xi \\ &\text{(the trick is } \nabla f = f \nabla \log f) \\ &= E\Big[\nabla_\theta \log p(\xi|\mu_\theta) R(\xi)\Big]\end{aligned}$$

- Using Monte-Carlo simulations: sampling M tracjectories ξ_i from policy μ_θ.

$$\begin{aligned}\nabla_\theta J(\theta) &\approx \frac{1}{M} \sum_{i=1}^{M} \nabla_\theta \log p(\xi_i|\mu_\theta) R(\xi_i) \\ &= \frac{1}{M} \sum_{i=1}^{M} \sum_{t=0}^{T_i} \nabla_\theta \log \mu_\theta(a_t|s_t) R(\xi_i)\end{aligned}$$

because $p(s_{t+1}|s_t, a_t)$ not depends on θ, so its gradient w.r.t θ is 0.

Policy gradient: Likelihood Ratio Gradient

A vanilla policy gradient algorithm

- initialize θ_0
- for $k = 0 : \infty$ (until convergence)
  - generate data: M trajectories ξ_i from policy μ_{θ_k}
  - evaluate: $\nabla_\theta J(\theta_k)$ from $\{\xi_i\}_{i=1}^M$
  - update: θ: $\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k)$
- end for

Variance Reduction

- Vanilla policy gradient method suffers from large variance.

Variance Reduction

- Vanilla policy gradient method suffers from large variance.

- Variance reduction using a baseline:
  – retain non-biased estimate gradients
  – reduce variance of gradient estimates

Variance Reduction

- Vanilla policy gradient method suffers from large variance.
- Variance reduction using a baseline:
  – retain non-biased estimate gradients
  – reduce variance of gradient estimates
- REINFORCE and G(PO)MDP algorithms

Variance Reduction: REINFORCE

- Introduce a baseline b that satisfies (un-biased)

$$\begin{aligned}\nabla_{\theta_i}^{RF} J(\theta) &= E_{p_\theta(\xi)}\left[\sum_{t=0}^{T-1} \nabla_{\theta_i} \log \mu_\theta(a_t|s_t)(R(\xi) - b)\right] \\ &= E_{p_\theta(\xi)}\left[\sum_{t=0}^{T-1} \nabla_{\theta_i} \log \mu_\theta(a_t|s_t) R(\xi))\right]\end{aligned}$$

- Non-bias gradient estimate: $\nabla_\theta J(\theta) = \nabla_\theta^{RF} J(\theta)$

Variance Reduction: REINFORCE

- Introduce a baseline b that satisfies (un-biased)

$$\begin{aligned}\nabla_{\theta_i}^{RF} J(\theta) &= E_{p_\theta(\xi)}\left[\sum_{t=0}^{T-1} \nabla_{\theta_i} \log \mu_\theta(a_t|s_t)(R(\xi) - b)\right] \\ &= E_{p_\theta(\xi)}\left[\sum_{t=0}^{T-1} \nabla_{\theta_i} \log \mu_\theta(a_t|s_t) R(\xi))\right]\end{aligned}$$

- Non-bias gradient estimate: $\nabla_\theta J(\theta) = \nabla_\theta^{RF} J(\theta)$

- Choose b s.t minimize the variance of $\nabla_\theta^{RF} J(\theta)$

$$\begin{aligned}\frac{\partial}{\partial b_i} \text{Var}\left[\nabla_{\theta_i}^{RF} J(\theta)\right] &= \frac{\partial}{\partial b_i}\left\{E\left[\left(\nabla_{\theta_i}^{RF} J(\theta)\right)^2\right] - E\left[\nabla_{\theta_i}^{RF} J(\theta)\right]^2\right\} \\ &= \frac{\partial}{\partial b_i}\left\{E\left[\left(\nabla_{\theta_i}^{RF} J(\theta)\right)^2\right]\right\} = 0\end{aligned}$$

where i is the i^{th}-dim of θ.

- The resulting baseline b is (for each dimension i)

$$b_i = \frac{E_{p_\theta(\xi)}\left[\left(\sum_{t=0}^{T-1} \nabla_{\theta_i} \log \mu_\theta(a_t|s_t)\right)^2 R(\xi)\right]}{E_{p_\theta(\xi)}\left[\left(\sum_{t=0}^{T-1} \nabla_{\theta_i} \log \mu_\theta(a_t|s_t)\right)^2\right]}$$

Variance Reduction: G(PO)MDP Algorithm

- The rewards at a given time are independent from future actions

$$E_{p_\theta(\xi)}\Big[\partial_\theta \log \mu_\theta(a_t|s_t,t) r_j\Big] = 0, \quad \forall j < t$$

- The gradient computed by G(PO)MDP is

$$\nabla_{\theta_i}^{G(PO)MDP} J(\theta) = E_{p_\theta(\xi)}\left[\sum_{j=0}^{T-1}\sum_{t=0}^{j} \nabla_{\theta_i} \log \mu_\theta(a_t|s_t)(r_j - b_j)\right]$$

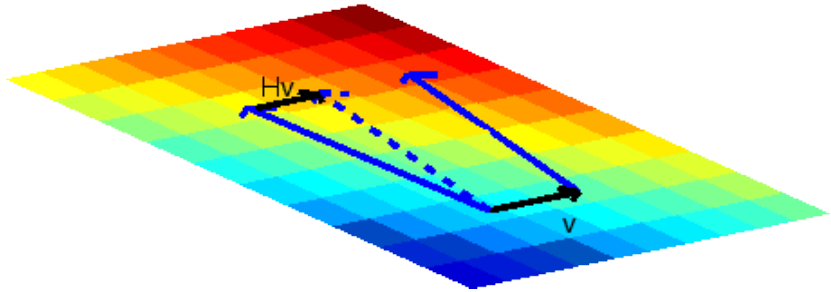
- The bias is

$$b_{ji} = \frac{E_{p_\theta(\xi)}\left[\Big(\sum_{t=0}^{j} \nabla_{\theta_i} \log \mu_\theta(a_t|s_t)\Big)^2 r_j\right]}{E_{p_\theta(\xi)}\left[\Big(\sum_{t=0}^{j} \nabla_{\theta_i} \log \mu_\theta(a_t|s_t)\Big)^2\right]}$$

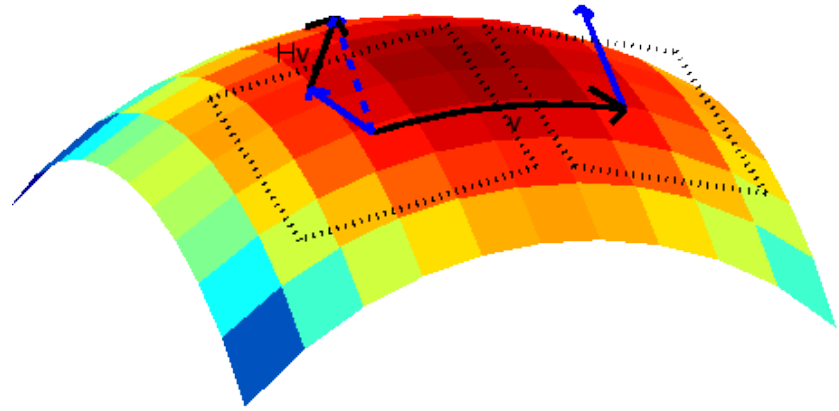
Natural policy gradient

Natural policy gradient

Steepest gradient



in a flat space



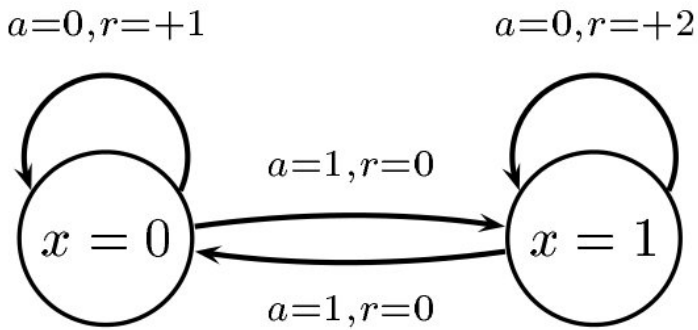
in a manifold

(pictures from http://www.netlib.org/utk/people/JackDongarra/)

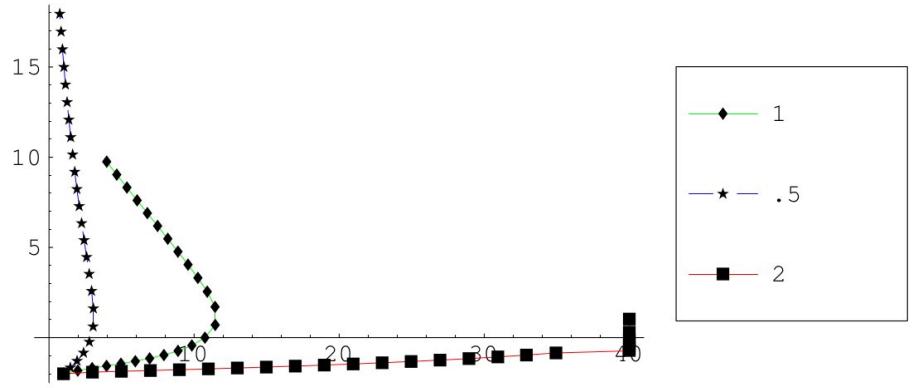
Natural policy gradient

Fisher Information Matrix

- Covariant to reparametrization (of the same representative power).
- The matrix is positive definite, its rotation is less than 90 degree. Thus all convergence properties of gradient methods are the same.



A Two state MDP.



Log odds of the policies:

$$\pi(a=0|x;\theta) = \frac{1}{1+\exp(\lambda\theta_0\delta_0(x)+\theta_1\delta_1(x))}$$

Natural policy gradient

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

where $G(\theta)$ is the Fisher Information Matrix (Amari, 1998).

$$G(\theta) = \mathbb{E}_{\xi \sim p(\xi;\theta)} \Big[\nabla_\theta \log p(\xi;\theta) \nabla_\theta \log p(\xi;\theta)^\top \Big]$$

Proof and derivation?

Natural Policy Gradient: Steepest descent

- The optimization (maxmize $J(\theta)$) is over the space of trajectories, when considering $p(\xi|\mu_\theta)$ is a function of parameter ξ (instead of θ) of dimension $|\theta|$.
- We try to define a Riemannian structure on the manifold of trajectories ξ

Natural Policy Gradient: Steepest descent

- The optimization (maxmize $J(\theta)$) is over the space of trajectories, when considering $p(\xi|\mu_\theta)$ is a function of parameter ξ (instead of θ) of dimension $|\theta|$.
- We try to define a Riemannian structure on the manifold of trajectories ξ
- The steepest descent should minimize the $J(\theta + \delta\theta)$ after update s.t.

$$\mathsf{KL}\Big(p(\xi;\theta)||\phi(\xi;\theta+\delta\theta)\Big) \leq \epsilon$$

Natural Policy Gradient: Steepest descent

- The optimization (maxmize $J(\theta)$) is over the space of trajectories, when considering $p(\xi|\mu_\theta)$ is a function of parameter ξ (instead of θ) of dimension $|\theta|$.
- We try to define a Riemannian structure on the manifold of trajectories ξ
- The steepest descent should minimize the $J(\theta + \delta\theta)$ after update s.t.

$$\mathrm{KL}\Big(p(\xi;\theta)||\phi(\xi;\theta+\delta\theta)\Big) \leq \epsilon$$

$$\mathrm{KL}\Big(p(\xi;\theta)||\phi(\xi;\theta+\delta\theta)\Big) \approx \frac{1}{2}\delta\theta^\top G\delta\theta = \langle\delta\theta,\delta\theta\rangle_G$$

- The steepest descent should minimize the $J(\theta + \delta\theta)$ after update. This is formulated as an optimization problem

$$\max J(\theta+\delta\theta) = J(\theta) + \delta\theta\nabla J(\theta) \quad \text{subject to } \langle\delta\theta,\delta\theta\rangle_G \leq \epsilon$$

Natural Policy Gradient: Steepest descent

- The optimization (maxmize $J(\theta)$) is over the space of trajectories, when considering $p(\xi|\mu_\theta)$ is a function of parameter ξ (instead of θ) of dimension $|\theta|$.
- We try to define a Riemannian structure on the manifold of trajectories ξ
- The steepest descent should minimize the $J(\theta + \delta\theta)$ after update s.t.

$$\mathrm{KL}\Big(p(\xi;\theta)||\phi(\xi;\theta+\delta\theta)\Big) \leq \epsilon$$

$$\mathrm{KL}\Big(p(\xi;\theta)||\phi(\xi;\theta+\delta\theta)\Big) \approx \frac{1}{2}\delta\theta^\top G\delta\theta = \langle\delta\theta,\delta\theta\rangle_G$$

- The steepest descent should minimize the $J(\theta + \delta\theta)$ after update. This is formulated as an optimization problem

$$\max J(\theta+\delta\theta) = J(\theta) + \delta\theta\nabla J(\theta) \quad \text{subject to } \langle\delta\theta,\delta\theta\rangle_G \leq \epsilon$$

- Using Lagrange multiplier

$$\mathcal{L}(\theta,\lambda) = J(\theta) + \delta\theta\nabla J(\theta) + \lambda(\sum_{i,j} G_{i,j}\delta\theta_i\delta\theta_j - \epsilon)$$

- Taking derivative $\delta\theta$,

$$\nabla J(\theta) + \lambda G\delta\theta = 0$$

- Therefore,

$$\delta = G^{-1}\nabla J(\theta)$$

Natural policy gradient

- The metric is the distances on probability spaces. The KL-divergence between two distributions is a natural divergence on changes in a distribution.

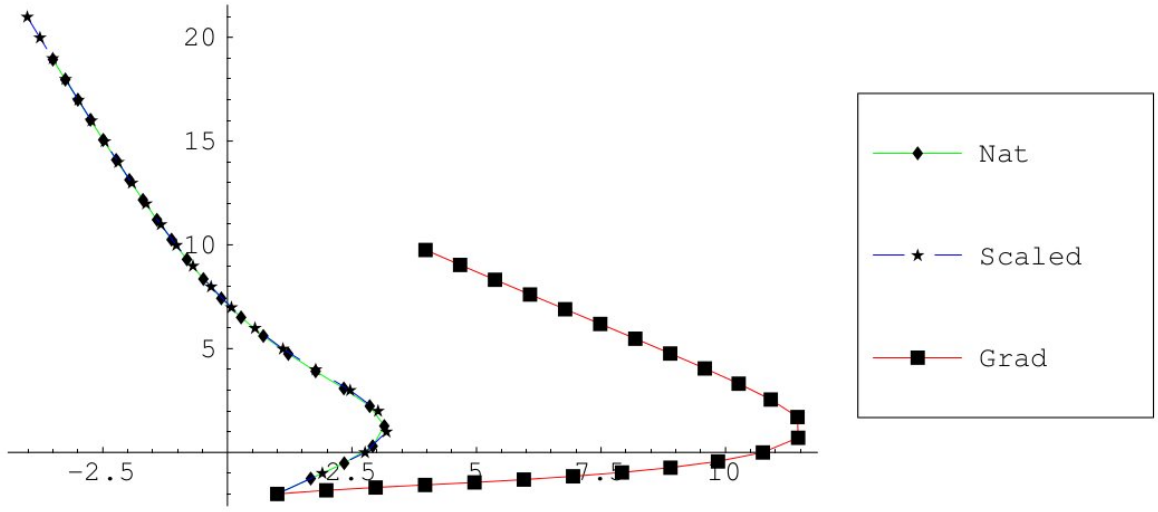
$$G(\theta) = \mathbb{E}_{\xi \sim p(\xi;\theta)} \Big[\nabla_\theta \log p(\xi;\theta) \nabla_\theta \log p(\xi;\theta)^\top \Big]$$

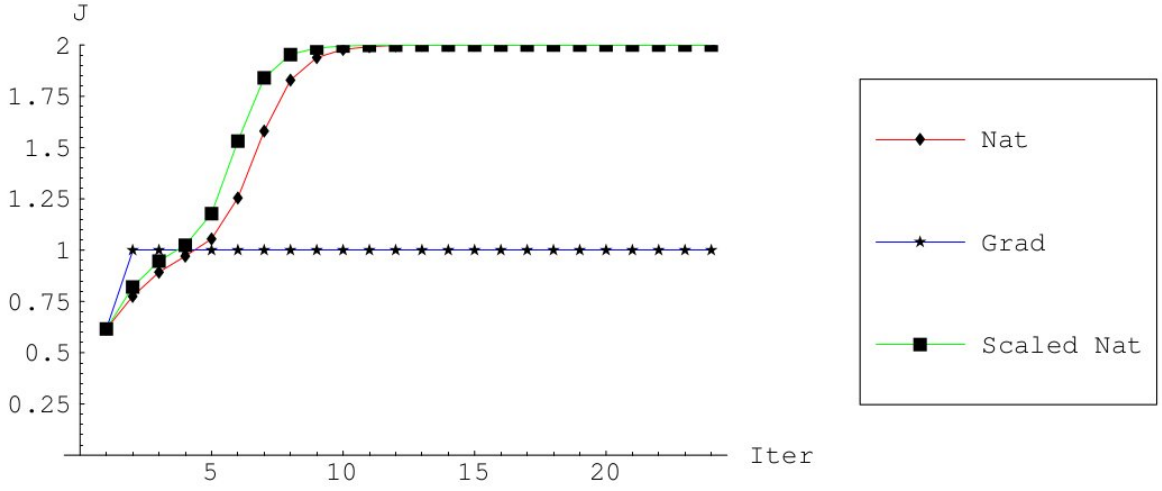
- Estimate the Fisher information matrix using sampled trajectories.

$$\begin{aligned} G(\theta) &\approx \frac{1}{M} \sum_{i=1}^{M} [\nabla_\theta \log p(\xi_i|\mu_\theta)] \times [\nabla_\theta \log p(\xi_i|\mu_\theta)]^\top \\ &= \frac{1}{M} \sum_{i=1}^{M} [\sum_{t=0}^{T_i} \nabla_\theta \log \mu_\theta(a_t|s_t) \nabla_\theta \log \mu_\theta(a_t|s_t)^\top] \end{aligned}$$

Natural Policy Gradient: Example

Two-State MDP

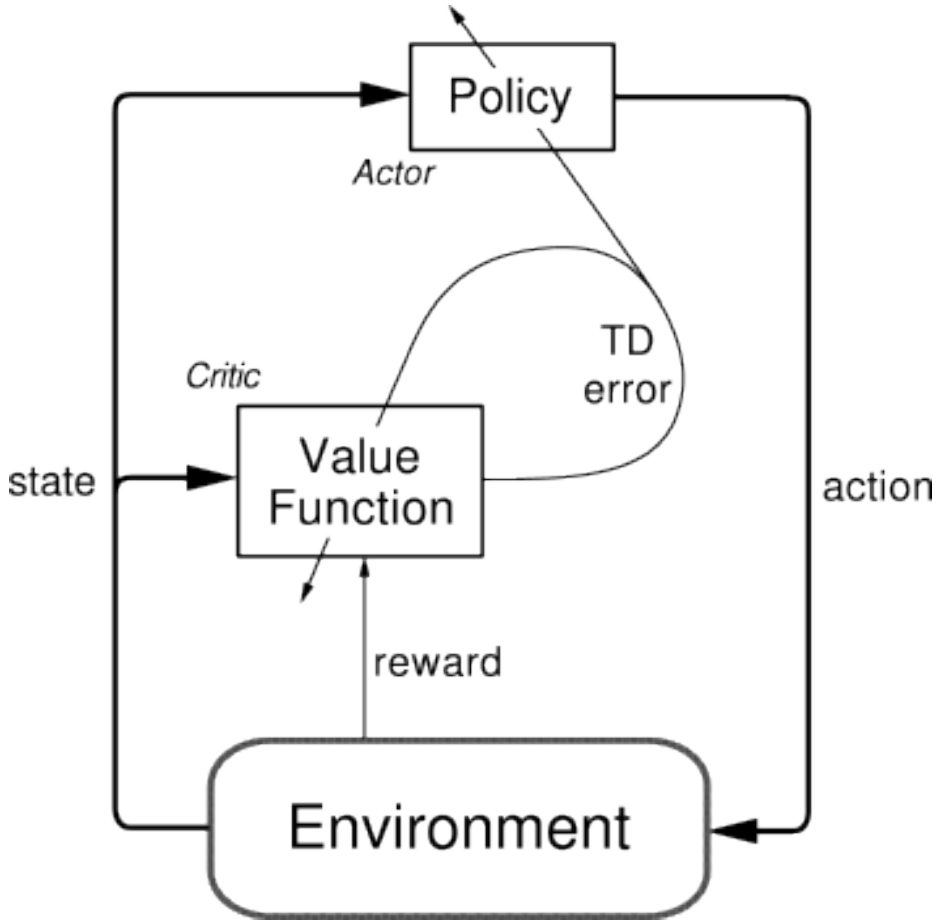




Actor-Critic Framework

Combining advatages of value-based and policy-based methods

Actor-Critic methods



- The policy structure is known as the actor, ($\rightarrow$ tuned by gradient updates.)
- The estimated value function is known as the critic ($\rightarrow$ tuned by TD errors).

(Introduction to RL, Sutton & Barto)

Policy Gradient Theorem Algorithm

- $R(\xi)$ can be estimated by $Q_t^\pi(s_t, a_t)$

$$\begin{aligned}\nabla_\theta J(\theta) &= E\Big[\sum_{t=0}^{T} \nabla_\theta \log \mu_\theta(a_t|s_t)\big(\sum_{j=t}^{T} r_j\big)\Big] \\ &= E\Big[\sum_{t=0}^{T} \nabla_\theta \log \mu_\theta(a_t|s_t) Q_t^\pi(s_t, a_t)\Big]\end{aligned}$$

(Baxter & Bartlett, 2001)

Policy Gradient Theorem Algorithm

- $R(\xi)$ can be estimated by $Q_t^\pi(s_t, a_t)$

$$\begin{aligned}\nabla_\theta J(\theta) &= E\Big[\sum_{t=0}^{T} \nabla_\theta \log \mu_\theta(a_t|s_t)\big(\sum_{j=t}^{T} r_j\big)\Big] \\ &= E\Big[\sum_{t=0}^{T} \nabla_\theta \log \mu_\theta(a_t|s_t) Q_t^\pi(s_t, a_t)\Big]\end{aligned}$$

(Baxter & Bartlett, 2001)

- Policy gradient with function approximation

$$Q_t^\pi(s_t, a_t) = \phi(s_t, a_t)^\top \cdot \mathbf{w}$$

- Compatible function approximation: how to choose $\phi(s_t, a_t)$? such that
  - – does not introduce bias
  - – reduce the variance

Compatible function approximation

The function approximation is compatible with the policy parametrization.

- The feature function is found $\phi(s,a) = \nabla_\theta \log \mu_\theta(a|s)$, by solving the least-square error

$$\mathbb{E}_{\phi(\xi)}\left[\sum_{t=0}^{T-1}(Q(s_t,a_t) - \phi(s_t,a_t)^\top w)^2\right]$$

(Sutton, 1999; Konda & Tsitsiklis, 1999)

Compatible function approximation

The function approximation is compatible with the policy parametrization.

- The feature function is found $\phi(s,a) = \nabla_\theta \log \mu_\theta(a|s)$, by solving the least-square error

$$\mathbb{E}_{\phi(\xi)}\left[\sum_{t=0}^{T-1}(Q(s_t,a_t) - \phi(s_t,a_t)^\top w)^2\right]$$

(Sutton, 1999; Konda & Tsitsiklis, 1999)

- The gradient with compatible function approximation

$$\nabla_\theta J(\theta) = \underbrace{E\Big[\sum_{t=0}^{T}\nabla_\theta \log \mu_\theta(a_t|s_t)\big(\nabla_\theta \log \mu_\theta(a_t|s_t)\big)^\top\Big]}_{\text{Fisher information matrix}}\mathbf{w}$$
$$= G_\theta \mathbf{w} \quad (\textbf{Actor-critic algorithms})$$

Compatible function approximation

The function approximation is compatible with the policy parametrization.

- The feature function is found $\phi(s,a) = \nabla_\theta \log \mu_\theta(a|s)$, by solving the least-square error

$$\mathbb{E}_{\phi(\xi)}\left[\sum_{t=0}^{T-1}(Q(s_t,a_t) - \phi(s_t,a_t)^\top w)^2\right]$$

(Sutton, 1999; Konda & Tsitsiklis, 1999)

- The gradient with compatible function approximation

$$\nabla_\theta J(\theta) = \underbrace{E\Big[\sum_{t=0}^{T}\nabla_\theta \log \mu_\theta(a_t|s_t)\big(\nabla_\theta \log \mu_\theta(a_t|s_t)\big)^\top\Big]}_{\text{Fisher information matrix}}\mathbf{w}$$
$$= G_\theta \mathbf{w} \quad (\textbf{Actor-critic algorithms})$$

- The natural policy gradient with compatible function approximation

$$\nabla_\theta^{NG} J(\theta) = G_\theta^{-1} G_\theta \mathbf{w} = \mathbf{w} \ (\textbf{Natural actor-critic algorithms})$$

Actor-critic algorithms

A regular-gradient actor-critic algorithm (a stochastic update version)

- initialize θ_0
- For $t = 0 : \infty$ (until convergence)
  - choose an action $a_t \sim \mu_{\theta_t}(a_t|s_t)$
  - Take a_t, observe r_t, and s_{t+1}.
  - Compute TD error: $\delta_t = r_t + \gamma\phi(s_{t+1}, a_{t+1})^\top \mathbf{w}_t - \phi(s_t, a_t)^\top \mathbf{w}_t$.
  - Critic update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\delta_t\phi(s_t, a_t)$
  - Actor update: $\theta_{t+1} = \theta_t + \beta\phi(s_t, a_t)\phi(s_t, a_t)^\top \mathbf{w}_t$
- end For

Episodic Natural Actor-Critic (eNAC)

Episodic Natural Actor-Critic (eNAC)

- Policy Gradient Theorem: replace the baseline by $V(s_t)$

$$\begin{aligned}\nabla_\theta^{RF} J(\theta) &= E_{p_\theta(\xi)}\left[\sum_{t=0}^{T-1} \nabla_\theta \mu_\theta(a_t|s_t)(Q(a_t,s_t) - b(s_t))\right] \\ &= E_{p_\theta(\xi)}\left[\sum_{t=0}^{T-1} \nabla_\theta \mu_\theta(a_t|s_t)(Q(a_t,s_t) - V(s_t))\right]\end{aligned}$$

- $A(s_t, a_t) = Q(a_t, s_t) - V(s_t)$ is called advantage functions

Episodic Natural Actor-Critic (eNAC)

- Policy Gradient Theorem: replace the baseline by $V(s_t)$

$$\begin{aligned}\nabla_\theta^{RF} J(\theta) &= E_{p_\theta(\xi)}\left[\sum_{t=0}^{T-1} \nabla_\theta \mu_\theta(a_t|s_t)(Q(a_t,s_t) - b(s_t))\right] \\ &= E_{p_\theta(\xi)}\left[\sum_{t=0}^{T-1} \nabla_\theta \mu_\theta(a_t|s_t)(Q(a_t,s_t) - V(s_t))\right]\end{aligned}$$

- $A(s_t, a_t) = Q(a_t, s_t) - V(s_t)$ is called advantage functions
- Bellman equation for a single transition of one sample trajectory

$$A(s_t, a_t) + V_t(s_t) = r(s_t, a_t) + \gamma V_t(s_{t+1})$$

Episodic Natural Actor-Critic (eNAC)

- Policy Gradient Theorem: replace the baseline by $V(s_t)$

$$\begin{aligned}\nabla_\theta^{RF} J(\theta) &= E_{p_\theta(\xi)}\left[\sum_{t=0}^{T-1} \nabla_\theta \mu_\theta(a_t|s_t)(Q(a_t,s_t) - b(s_t))\right] \\ &= E_{p_\theta(\xi)}\left[\sum_{t=0}^{T-1} \nabla_\theta \mu_\theta(a_t|s_t)(Q(a_t,s_t) - V(s_t))\right]\end{aligned}$$

- $A(s_t, a_t) = Q(a_t, s_t) - V(s_t)$ is called advantage functions
- Bellman equation for a single transition of one sample trajectory

$$A(s_t, a_t) + V_t(s_t) = r(s_t, a_t) + \gamma V_t(s_{t+1})$$

- For the whole path

$$\sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \mu_\theta(a_t|s_t)\mathbf{w} + V_0(s_0) = \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) + \gamma^T r_T(s_T)$$

Note: For multiple starting states s_0: $V_0(s) = \psi(s)^\top \mathbf{v}$

- For a set of trajectories $\mathcal{D} = \{\xi_i\}_{i=1}^N = \left\{\{s_0^0, a_0^0, s_1^0, a_1^0, \ldots, s_{T_0-1}^0\}, \ldots\right\}_{i=1}^N$, denote

$$\phi^i = \left[\left(\sum_{t=0}^{T_i-1} \gamma^t \nabla_\theta \log \mu_\theta(a_t^i|s_t^i)\right)^\top, \psi(s_0^i)^\top\right]^\top, \quad \Phi = [\phi^1, \phi^2, \ldots, \phi^N]^\top$$

$$R = \left[\sum_{t=0}^{T_1-1} \gamma^t r(s_t^1, a_t^1) + \gamma^T r_T(s_T^1), \ldots, \sum_{t=0}^{T_N-1} \gamma^t r(s_t^N, a_t^N) + \gamma^T r_T(s_T^N)\right]^\top$$

- For a set of trajectories $\mathcal{D} = \{\xi_i\}_{i=1}^N = \left\{\{s_0^0, a_0^0, s_1^0, a_1^0, \ldots, s_{T_0-1}^0\}, \ldots\right\}_{i=1}^N$, denote

$$\phi^i = \left[\left(\sum_{t=0}^{T_i-1} \gamma^t \nabla_\theta \log \mu_\theta(a_t^i|s_t^i)\right)^\top, \psi(s_0^i)^\top\right]^\top, \quad \Phi = [\phi^1, \phi^2, \ldots, \phi^N]^\top$$

$$R = \left[\sum_{t=0}^{T_1-1} \gamma^t r(s_t^1, a_t^1) + \gamma^T r_T(s_T^1), \ldots, \sum_{t=0}^{T_N-1} \gamma^t r(s_t^N, a_t^N) + \gamma^T r_T(s_T^N)\right]^\top$$

- by regression

$$\begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} = [\Phi^\top \Phi]^{-1} \Phi^\top R$$

- For a set of trajectories $\mathcal{D} = \{\xi_i\}_{i=1}^N = \left\{\{s_0^0, a_0^0, s_1^0, a_1^0, \ldots, s_{T_0-1}^0\}, \ldots\right\}_{i=1}^N$, denote

$$\phi^i = \left[\left(\sum_{t=0}^{T_i-1} \gamma^t \nabla_\theta \log \mu_\theta(a_t^i|s_t^i)\right)^\top, \psi(s_0^i)^\top\right]^\top, \quad \Phi = [\phi^1, \phi^2, \ldots, \phi^N]^\top$$

$$R = \left[\sum_{t=0}^{T_1-1} \gamma^t r(s_t^1, a_t^1) + \gamma^T r_T(s_T^1), \ldots, \sum_{t=0}^{T_N-1} \gamma^t r(s_t^N, a_t^N) + \gamma^T r_T(s_T^N)\right]^\top$$

- by regression

$$\begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} = [\Phi^\top \Phi]^{-1} \Phi^\top R$$

- The eNAC gradient is

$$\nabla_\theta^{eNAC} J(\theta) = \mathbf{w}$$

Episodic natural actor-critic (eNAC)

- initialize θ_0
- For $t = 0 : \infty$ (until convergence)
  - sample a set of trajectories $\{\xi_i\}$ from $\pi(\theta_t)$
  - Critic update: by regression

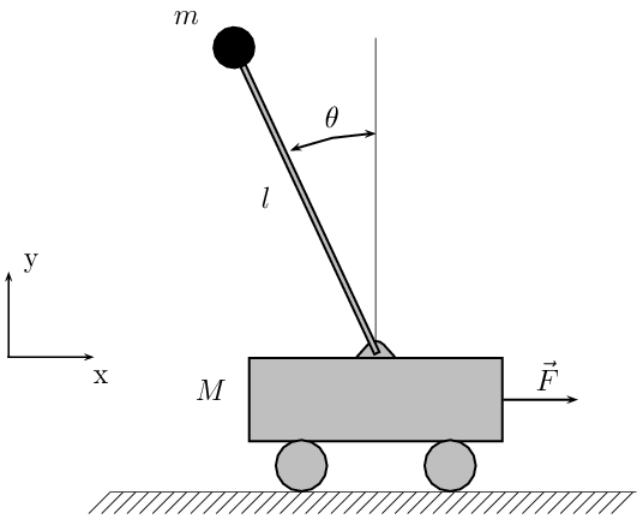
$$\begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} = [\Phi^\top \Phi]^{-1} \Phi^\top R$$

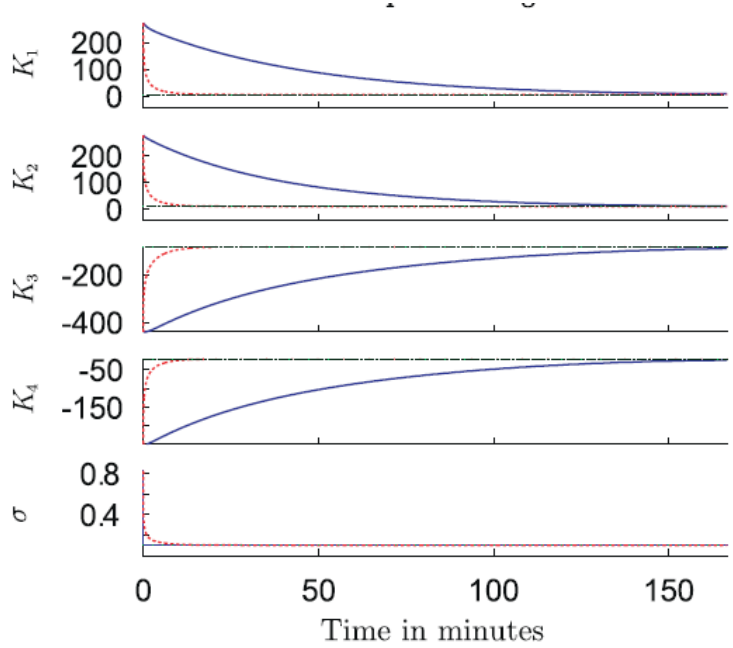
  - Actor update:

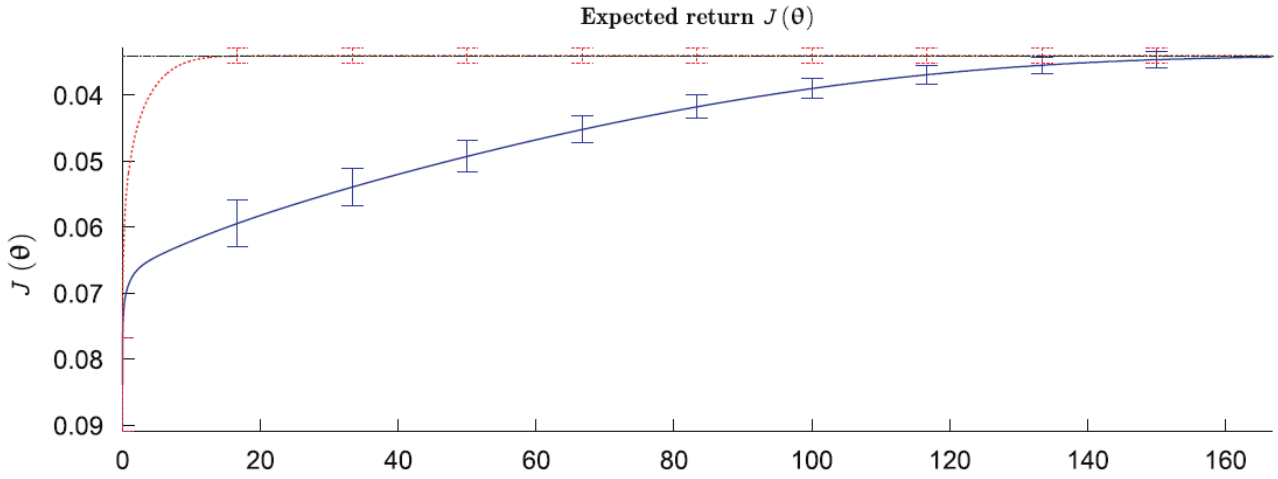
$$\theta = \theta + \alpha \mathbf{w}$$

- end For

Example: Cart-pole







red: eNAC, blue: vanilla PG

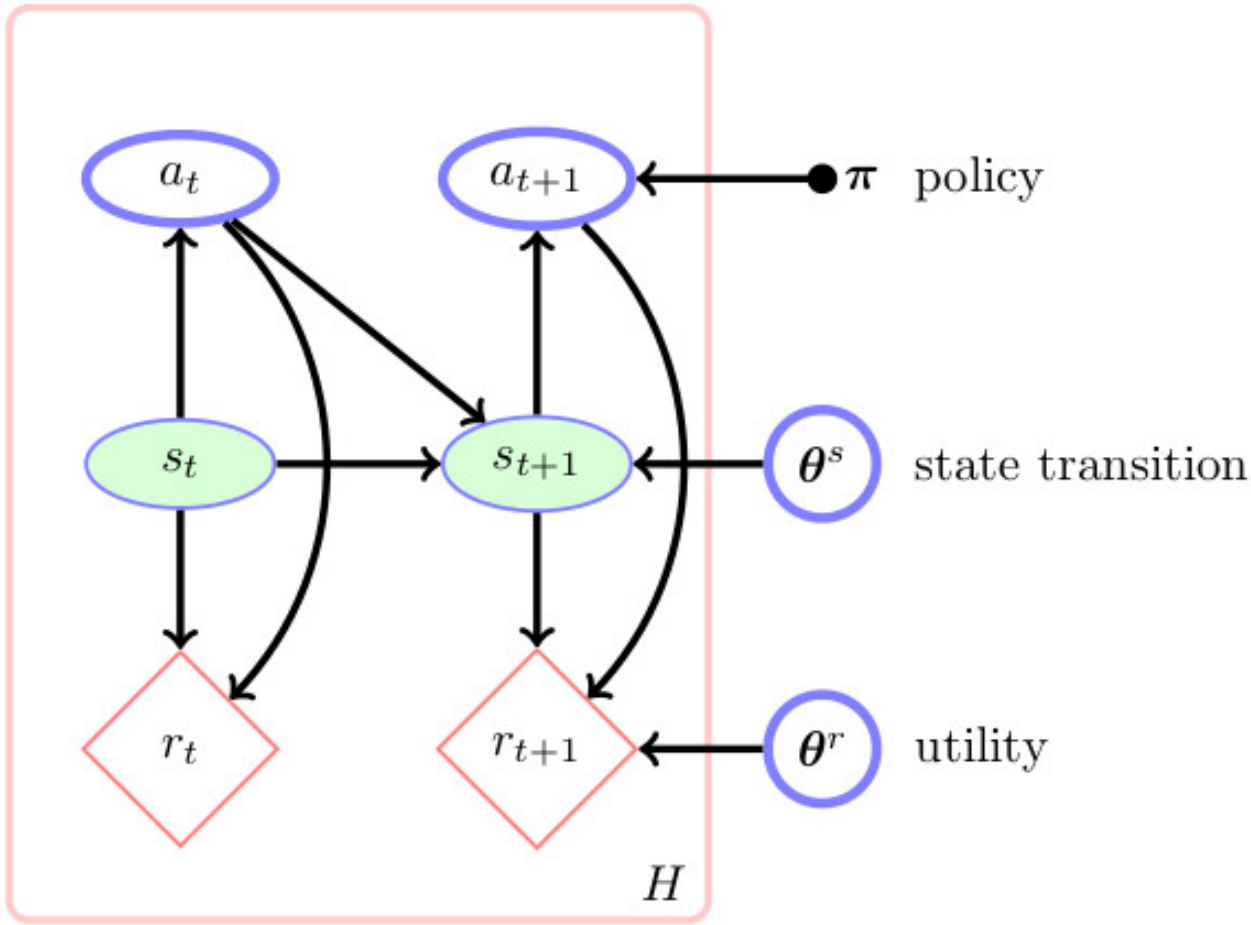
Peters & Schaal, Neurocomputing, 2008

Inference-based Policy Search

Inference-based Policy Search

- No problems with step-sizes, e.g. unstable learning process or slow convergence.
- Invariant to transformation of the parameters
- Using inference to infer a new policy.

Inference for Policy Search



Furmston & Barber, 2010

Inference-based Policy Search

- Assume rewards are positive, then they are transformed to the probability distribution, e.g. $p(R|\xi) \propto \exp\big(R(\xi)\big)$
- Assume the policy is parameterized by θ: $\pi(\theta)$
- The objective is to maximize the log-marginal likelihood

$$\log p(R;\theta) = \int p(R|\xi)p(\xi;\theta)d\xi$$

- For a standard EM-algorithm, a variational distribution $q(\xi)$ is used

$$\log p(R;\theta) = \underbrace{\mathcal{L}(q(\xi);\theta)}_{\text{Lower bound}} + \mathsf{KL}\big(q(\xi)||p(\xi|R;\theta)\big)$$

where the lower bound of the log evidence is

$$\mathcal{L}(q;\theta) = \int q(\xi) \log \frac{p(R|\xi)p(\xi;\theta)}{q(\xi)} d\xi$$

and the Kullback-Leibler is

$$\mathsf{KL}\big(q(\xi)||p(\xi|R;\theta)\big) = -\int q(\xi) \log \frac{p(\tau|R;\theta)}{q(\xi)} d\xi$$

- For a standard EM-algorithm, a variational distribution $q(\xi)$ is used

$$\log p(R;\theta) = \underbrace{\mathcal{L}(q(\xi);\theta)}_{\text{Lower bound}} + \mathrm{KL}\big(q(\xi)||p(\xi|R;\theta)\big)$$

  where the lower bound of the log evidence is

$$\mathcal{L}(q;\theta) = \int q(\xi) \log \frac{p(R|\xi)p(\xi;\theta)}{q(\xi)} d\xi$$

  and the Kullback-Leibler is

$$\mathrm{KL}\big(q(\xi)||p(\xi|R;\theta)\big) = -\int q(\xi) \log \frac{p(\tau|R;\theta)}{q(\xi)} d\xi$$

- E-step: set $q(\xi) = p(\xi|R;\theta)$
- M-Step: maximize $L(q;\theta)$ w.r.t θ while q is fixed.

Policy learning by Weighting Exploration with Returns (PoWER)

- Two key insights: structured exploration and inference-based.

Policy learning by Weighting Exploration with Returns (PoWER)

- Two key insights: structured exploration and inference-based.
- Structured exploration:

$$a_t = \left(\theta + \epsilon_t\right)^\top \phi(s_t)$$

  where $\epsilon_t \propto \mathcal{N}(0, \Sigma)$
- Hence, the controller is: $p(a_t|s_t) = \mathcal{N}(a_t|\theta^\top \phi(s_t), \phi(s_t)^\top \Sigma \phi(s_t))$

- E-step: $q(\xi) = p(\xi|R;\theta_k)$
- M-step: $\theta_{k+1} = \text{argmax}_\theta\, L(\theta; q(\theta_k))$

$$
\begin{aligned}
\mathcal{L}(q;\theta) &= \int q(\xi) \log \frac{p(R|\xi)p(\xi;\theta)}{q(\xi)} d\xi \\
&= \int p(\xi|R;\theta_k) \log \frac{p(R|\xi)p(\xi;\theta)}{p(\xi|R;\theta_k)} d\xi \\
&\propto \int p(R|\xi)p(\xi;\theta_k) \log \frac{p(R|\xi)p(\xi;\theta)}{p(R|\xi)p(\xi;\theta_k)} d\xi \\
&= \int p(R|\xi)p(\xi;\theta_k) \log \frac{p(\xi;\theta)}{p(\xi;\theta_k)} d\xi \\
&= \int p(R|\xi)p(\xi;\theta_k) \log \frac{p(\xi;\theta)}{p(\xi;\theta_k)} d\xi
\end{aligned}
$$

- M-step: Solve for θ_{k+1}

$$
\begin{aligned}
0 = \partial_\theta L(q;\theta_k) &= \int p(R|\xi) p(\xi;\theta_k) \partial_\theta \log p(\xi;\theta) d\xi \\
&= \mathsf{E}_{p(\xi;\theta_k)}\Big(p(R|\xi) \partial_\theta \log p(\xi;\theta) \Big) \\
&= \mathsf{E}_{p(\xi;\theta_k)}\Big(\sum_{t=1}^{T} \partial_\theta \log \pi(a_t|s_t) Q(s_t,a_t) \Big)
\end{aligned}
$$

- M-step: Solve for θ_{k+1}

$$
\begin{aligned}
0 = \partial_\theta L(q;\theta_k) &= \int p(R|\xi)p(\xi;\theta_k)\partial_\theta \log p(\xi;\theta) d\xi \\
&= \mathsf{E}_{p(\xi;\theta_k)}\Big(p(R|\xi)\partial_\theta \log p(\xi;\theta)\Big) \\
&= \mathsf{E}_{p(\xi;\theta_k)}\Big(\sum_{t=1}^{T} \partial_\theta \log \pi(a_t|s_t) Q(s_t,a_t)\Big)
\end{aligned}
$$

- Results:

$$
\theta_{k+1} = \theta_k + \left(\mathsf{E}_{p(\xi;\theta_k)}\Big(\sum_{t=1}^{T} W_t Q(s_t,a_t)\Big)\right)^{-1} \mathsf{E}_{p(\xi;\theta_k)}\Big(\sum_{t=1}^{T} W_t \epsilon_t Q(s_t,a_t)\Big)
$$

where $W_t = \phi(s_t)\phi(s_t)^\top \big(\phi(s_t)^\top \Sigma \phi(s_t)\big)^{-1}$

Data generation at each step: Example of generating one sampled trajectory

- Input: θ_k

- for $t = 0 : T$
  - Generate $\epsilon_t = \mathcal{N}(0, \Sigma)$
  - Compute action: $a_t = (\theta_k + \epsilon_t)^\top \phi(s_t)$
  - Execute a_t, osberve s_{t+1}, r_t
  - A data-tuple: $(s_t, a_t, \epsilon_t, s_{t+1}, r_t)$
- end for

Information-theoretic Policy Search

Information-theoretic Policy Search

- Information-theoretic bounds the update of the trajectory distributions, i.e. regularization limits the information loss in policy update. For example, natural policy gradient.
- Regularization helps avoid premature on local optima as in EM-based methods.

Information-theoretic Policy Search

- Information-theoretic bounds the update of the trajectory distributions, i.e. regularization limits the information loss in policy update. For example, natural policy gradient.
- Regularization helps avoid premature on local optima as in EM-based methods.

- Information-theoretic policy exploits advantages from inference-based (updates without a step-size), and natural policy gradient (information loss bound).
- For examples: Relative Entropy Policy Search (REPS).

Relative Entropy Policy Search (REPS)

- Formulate the policy search problem as optimization

$$\max_{\pi} \int_{\theta} \phi(\theta) R(\theta)$$
$$\text{s.t : KL}\Big(\phi(\theta)||q(\theta)\Big) \leq \epsilon \text{ (information loss bound)}$$
$$\int \phi(\theta) = 1 \text{ (a distribution)}$$

where $q(\theta)$ is an old policy (used to generate data)

Relative Entropy Policy Search (REPS)

- Formulate the policy search problem as optimization

$$\max_{\pi} \int_{\theta} \phi(\theta) R(\theta)$$
$$\text{s.t : KL}\Big(\phi(\theta)||q(\theta)\Big) \leq \epsilon \text{ (information loss bound)}$$
$$\int \phi(\theta) = 1 \text{ (a distribution)}$$

  where $q(\theta)$ is an old policy (used to generate data)

- Analytic solution

$$\phi(\theta) \propto q(\theta) \exp(\frac{R(\theta)}{\eta})$$

  – η is found by solving the dual problem (convex optimization).

Policy Search via Global Optimization

- CMA-ES and Bayesian Optimization

Policy Search via Global Optimization

- Parametric policy $\pi(a|s;\theta)$
- Update the policy with only evaluations $J(\theta)$

Covariance Matrix Adaptation-Evolutionary Strategy (CMA-ES)

- A parametric distribution of θ, $\mu(\theta; w)$

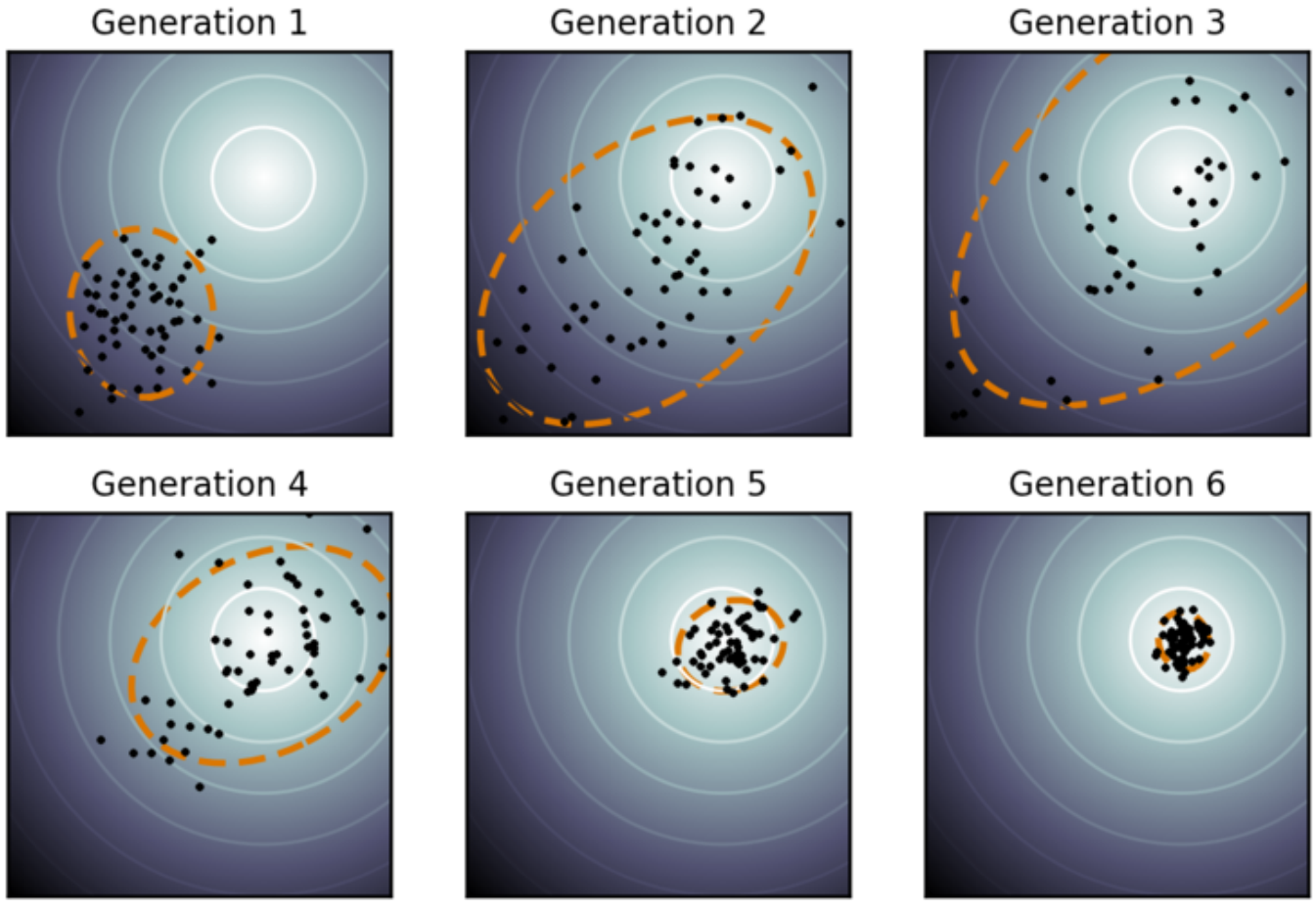
Input: initial parameter w, function $J(\theta)$, update function $h(w, D)$

Output: final w^* and best point θ

1: **repeat**
2: Sample $\{\theta_i\}_{i=1}^n \sim p(\theta; w)$
3: Evaluate samples, $D = \{(\theta_i, J(\theta_i))\}_{i=1}^n$
4: Update $w \leftarrow h(w, D)$
5: **until** w converges

- for a simplest example of $\mu(\theta; w) = \mathcal{N}(\theta; \hat{\theta}, \Sigma), w = \{\hat{\theta}, \Sigma\}$
- updating function:
  – given $D = \{(\theta_i, J(\theta_i))\}_{i=1}^n$, select l best candidates $D_l = \text{best}_l(D)$
  – calculate $\hat{\theta}$ is the weighted mean $\hat{\theta}_{k+1}$ of D_l, order them by return $J(\theta_i)$, the weight $d^{[i]} = \log(l+1) - \log(i)$
  – Calculate Σ proportional to $(\hat{\theta}_{k+1} - \hat{\theta}_k)$
- CMA-ES is similar to many episode-based policy search methods

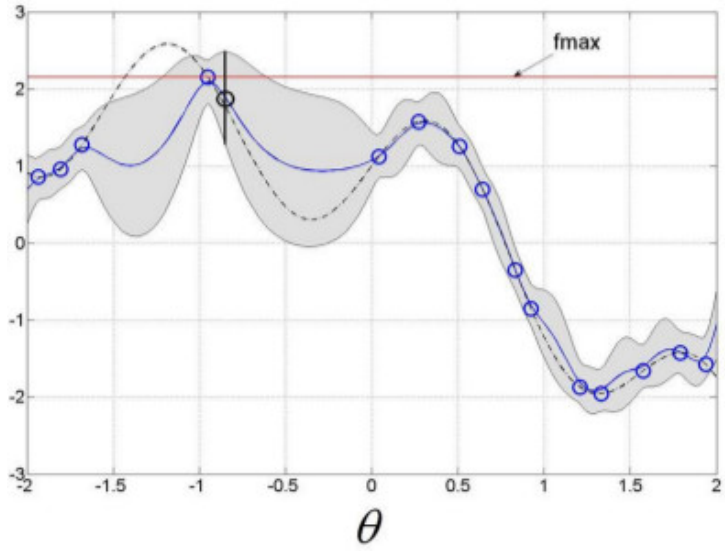
Neuroevolution strategies for episodic reinforcement learning, by Heidrich-Meisner, Igel, 2009


Generation 1
Generation 2
Generation 3
Generation 4
Generation 5
Generation 6

Bayesian Optimization

Bayesian Optimization

- objective function $J(\theta) \sim GP\big(0, k(\theta, \theta')\big)$



Aaron Wilson, Alan Fern, Prasad Tadepalli, JMLR 2014

Input: Data $D_0 = []$

1: **repeat**
2: Update the covariances **K**
3: Select the next parameter $\theta_{k+1} = \arg\max_\theta u(\theta; D_k)$
4: Collect N episodes from the policy $\pi(a|s; \theta_{k+1})$
5: Compute the performance of θ_{k+1}: $J(\theta_{k+1}) = \frac{1}{N} \sum_{i=1}^{N} R_i$
6: Update $D_{k+1} = D_k \cup \{\theta_{k+1}, J(\theta_{k+1})\}$
7: **until** Converge

- prediction

$$\mu\big(J(\theta)|D_n\big) = \mathbf{k}(\theta)^\top \mathbf{K}^{-1} J$$
$$\sigma^2\big(J(\theta)|D\big) = k(\theta,\theta) - \mathbf{k}(\theta)^\top \mathbf{K}^{-1}\mathbf{k}(\theta)$$

where $J = [J(\theta_1), \ldots, J(\theta_n)]^\top$, and $\mathbf{K} = [k(\theta_i, \theta_j)]$

- prediction

$$\mu\big(J(\theta)|D_n\big) = \mathbf{k}(\theta)^\top \mathbf{K}^{-1} J$$
$$\sigma^2\big(J(\theta)|D\big) = k(\theta,\theta) - \mathbf{k}(\theta)^\top \mathbf{K}^{-1}\mathbf{k}(\theta)$$

  where $J = [J(\theta_1), \ldots, J(\theta_n)]^\top$, and $\mathbf{K} = [k(\theta_i, \theta_j)]$

- acquisition function $u(\theta; D)$
  – Probability of improvement (PI): $\theta = \arg\max p(x|\theta, D), x \sim \mathcal{N}(x; \mu(\theta), \sigma^2(\theta))$

$$u(\theta) = \Phi\Big(\frac{\mu(\theta) - J^*}{\sigma(\theta)}\Big)$$

  – Expected improvement (EI)

$$u(\theta) = \int_{-\infty}^{J^*} \mathcal{N}(J|\mu(\theta), \sigma^2(\theta))(J^* - J)dJ = \sigma(\theta)\big(y\Phi(y) + \mathcal{N}(y; 0, 1)\big)$$

  where $y = \Phi\Big(\frac{\mu(\theta) - J^*}{\sigma(\theta)}\Big)$
  – Upper confidence bound (UCB)

$$u(\theta) = \mu(\theta) + \alpha\sigma(\theta)$$

References

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters: A Survey on Policy Search for Robotics. Foundations and Trends in Robotics, (2013)

Gerhard Neumann, Jan Peters: A tutorial on Policy Search: Methods and Applications at ICML 2015 (http://icml.cc/2015/tutorials/PolicySearch.pdf)