Carnegie Mellon

School of Computer Science

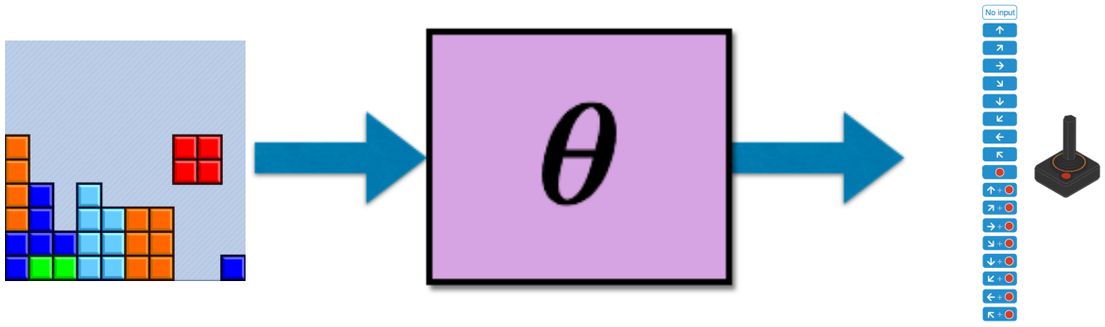
Deep Reinforcement Learning and Control

Evolutionary Methods for Policy Search

Fall 2025, CMU 10-703

Instructors: Katerina Fragkiadaki & Aviral Kumar

Reinforcement Learning



$\pi(a \mid s, \theta)$

Given an initial state distribution $\mu_0(s_0)$, estimate parameters θ of a policy π_θ so that, the trajectories τ sampled from this policy have maximum returns, i.e., sum of rewards $R(\tau)$.

$$\max_{\theta}. \quad U(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[R(\tau) \mid \pi_\theta, \mu_0(s_0)\right]$$

τ : trajectory, a sequence of state, action, rewards, a game fragment or a full game:

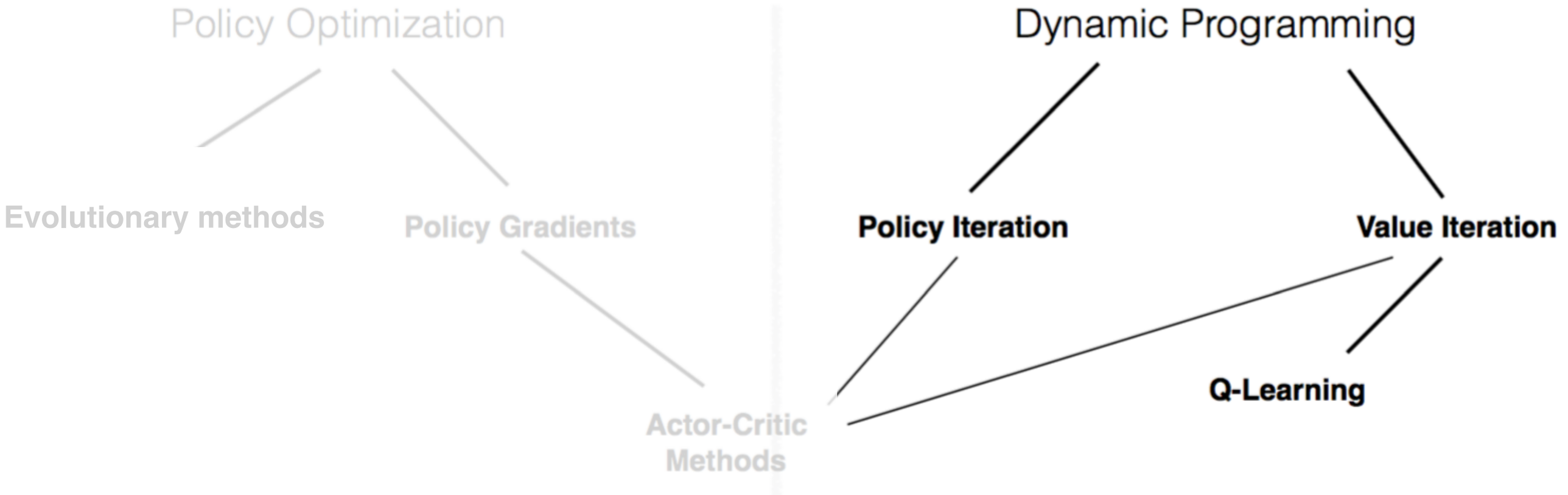
$$\tau : s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots s_T, a_T, r_T$$

$R(\tau)$: reward of a trajectory: (discounted) sum of the rewards of the individual state/actions

$$R(\tau) = \sum_{t=1}^{T} r_t$$

Policy Optimization and RL

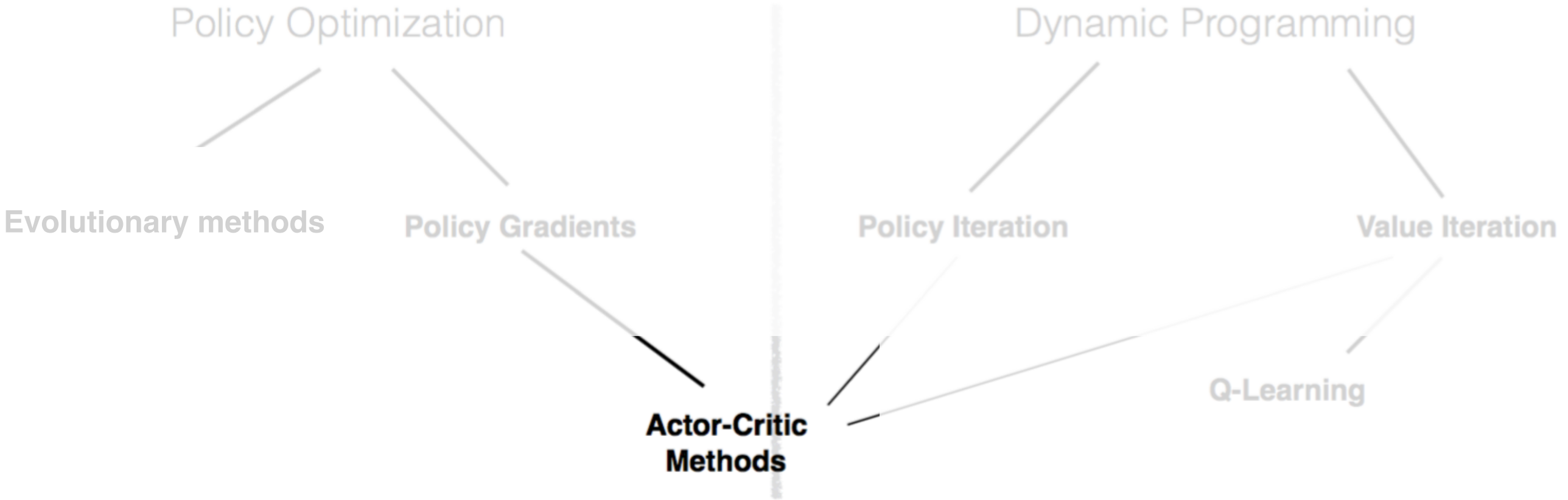
$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \,|\, \pi_\theta, \mu_0(s_0) \right]$$



Dynamic programming methods search over values for states without explicitly estimating policies. They use recursive updates that capture how one state results from another.

Black-box Policy Optimization

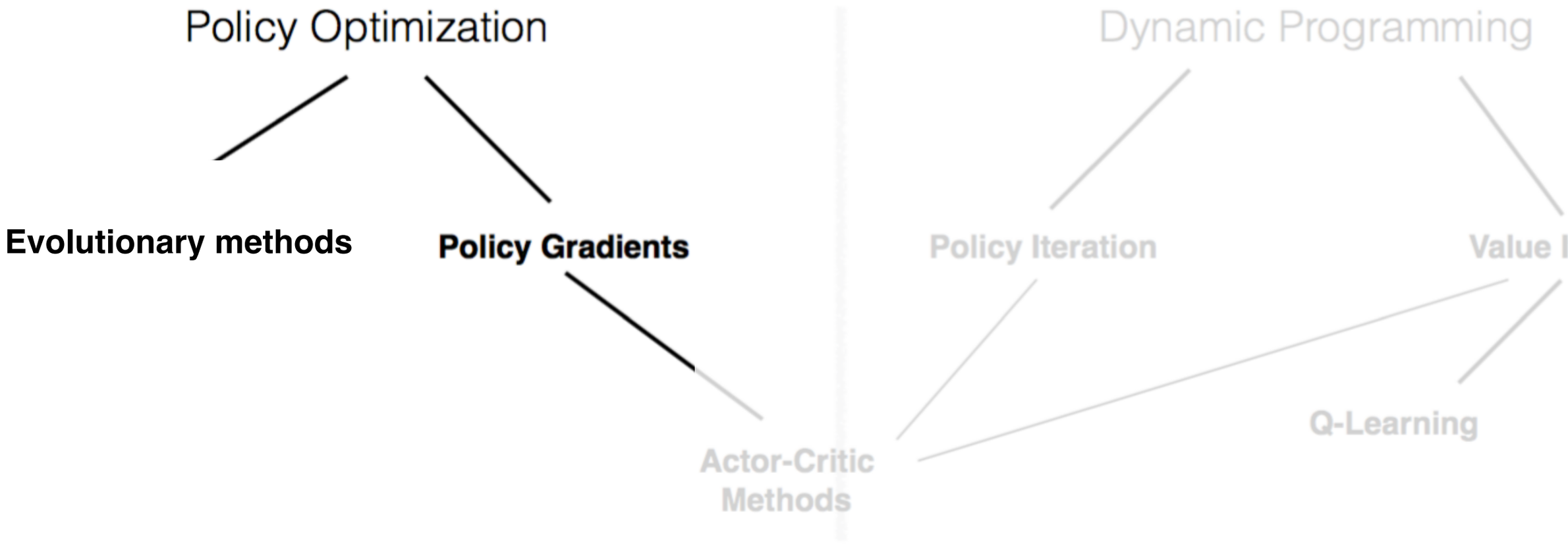
$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \,|\, \pi_\theta, \mu_0(s_0) \right]$$



Actor critic method both search over policies and over state values, and use the state values, as opposed to the raw rewards to guide policy search. Estimate of value function are usually way less noisy than raw rewards

Black-box Policy Optimization

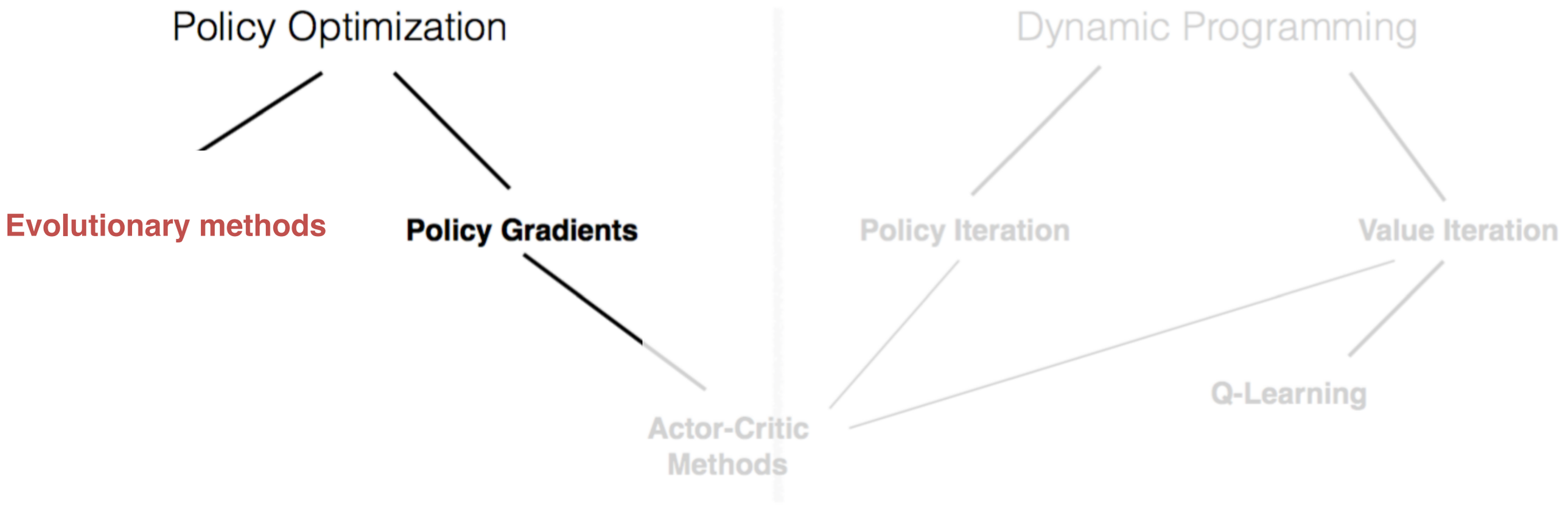
$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \,|\, \pi_\theta, \mu_0(s_0) \right]$$



Policy optimization just searches over policy parameters without computing values for states: it just looks are the raw rewards obtained as we search over the parameters. It does not exploit the structure of the states

Black-box Policy Optimization

$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \,|\, \pi_\theta, \mu_0(s_0) \right]$$



Policy optimization searches over policy parameters without computing values for states. It does not exploit any recursions, or reward decompositions over timesteps.

Black-box policy optimization

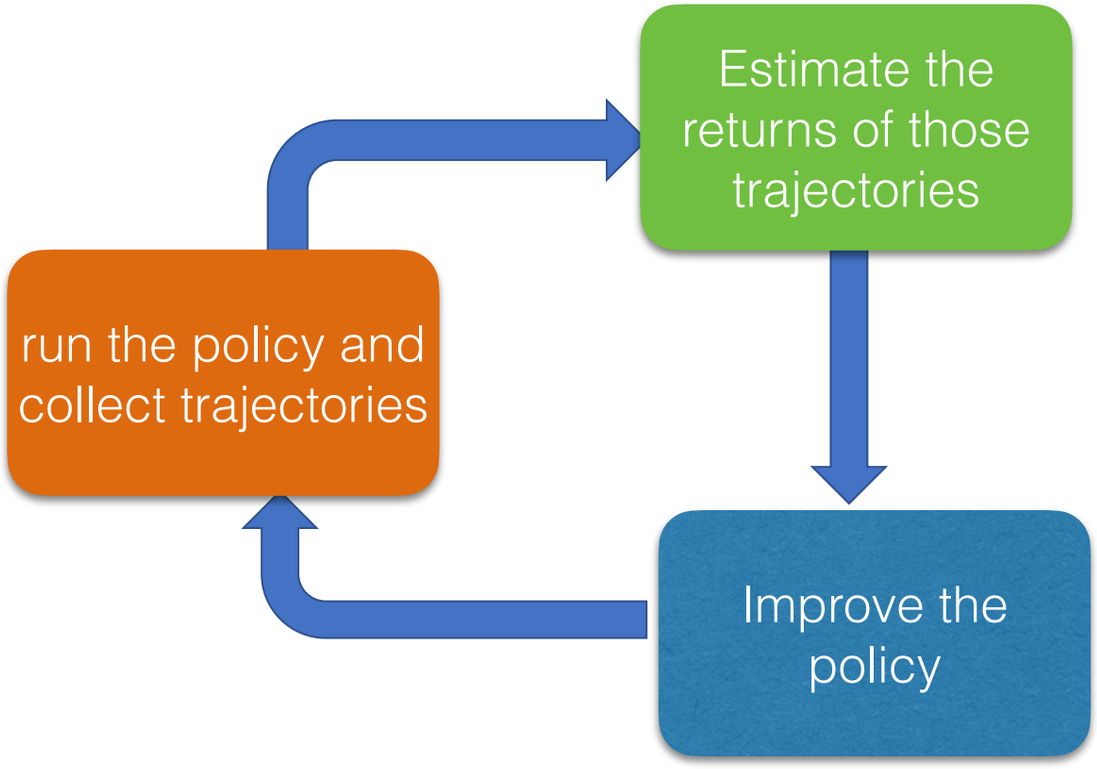
Initialize the policy parameters θ randomly.

1. Perturb policy parameters,

2. Run the resulting policy, collect trajectories and evaluate their returns.

3. Promote **the policy para**meters that resulted in trajectories that gave the largest return improvement.

4. GOTO 1.



- No gradient information
- no information regarding the structure of the reward, that it is additive over states or that states are interconnected in a particular way.

Evolutionary methods for policy search

$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \,|\, \pi_\theta, \mu_0(s_0) \right]$$

General algorithm:

Initialize a population of parameter vectors (genotypes)

1. *Make random perturbations (mutations) to each parameter vector*
2. *Evaluate the perturbed parameter vector (fitness)*
3. *Keep the perturbed vector if the result improves (selection)*
4. *GOTO 1*

Simple and biologically plausible…

Gaussian Density

Perhaps the most common probability density

$$\mathcal{N}\left(y|\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

σ^2 is the variance of the density and μ is the mean.

Multivariate Gaussian Density

Perhaps the most common probability density

$$\mathcal{N}(\mathbf{y} \mid \mu, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k \mid \boldsymbol{\Sigma} \mid}} \exp\left(-\frac{1}{2}(\mathbf{y}-\mu)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\mu)\right)$$

Cross-entropy method (CEM)

- Policy parameters are sampled from a multivariate Gaussian distribution with a diagonal covariance matrix.
- We will update the mean and variances of the parameter elements towards samples that have highest fitness scores.

Input: parameter space Θ, number of parameter vectors n, proportion $\rho \leq 1$, noise η
Initialize: Set the parameter $\mu = \bar{0}$ and $\sigma^2 = 100I$ (I is the identity matrix)
for $k = 1, 2, \ldots$ **do**
  Generate a random sample of n parameter vectors $\{\theta_i\}_{i=1}^{n} \sim \mathcal{N}(\mu, \sigma^2 I)$
  For each θ_i, play L games and calculate the average number of rows removed (score) by the controller
  Select $\lfloor \rho n \rfloor$ parameters with the highest score $\theta'_1, \ldots, \theta'_{\lfloor \rho n \rfloor}$
  Update μ and σ: $\mu(j) = \frac{1}{\lfloor \rho n \rfloor} \sum_{i=1}^{\lfloor \rho n \rfloor} \theta'_i(j)$ and $\sigma^2(j) = \frac{1}{\lfloor \rho n \rfloor} \sum_{i=1}^{\lfloor \rho n \rfloor} [\theta'_i(j) - \mu(j)]^2 + \eta$

Works embarrassingly well in low-dimensions, e.g., to search for a linear policy over the 22 Bertsekas features for Tetris.

Cross-entropy method

Work embarrassingly well in low-dimensions

Method	Mean Score	Reference
Nonreinforcement learning		
Hand-coded	631,167	Dellacherie (Fahey, 2003)
Genetic algorithm	586,103	(Böhm et al., 2004)
Reinforcement learning		
Relational reinforcement learning+kernel-based regression	≈50	Ramon and Driessens (2004)
Policy iteration	3183	Bertsekas and Tsitsiklis (1996)
Least squares policy iteration	<3000	Lagoudakis, Parr, and Littman (2002)
Linear programming + Bootstrap	4274	Farias and van Roy (2006)
Natural policy gradient	≈6800	Kakade (2001)
CE+RL	21,252	
CE+RL, constant noise	72,705	
CE+RL, decreasing noise	348,895	

István Szita and András Lörincz. "Learning Tetris using the noisy cross-entropy method". In: *Neural computation* 18.12 (2006), pp. 2936–2941

$\mu \in \mathbb{R}^{22}$

Much later:

Approximate Dynamic Programming Finally Performs Well in the Game of Tetris

[NIPS 2013]

Victor Gabillon
INRIA Lille - Nord Europe, Team SequeL, FRANCE
victor.gabillon@inria.fr

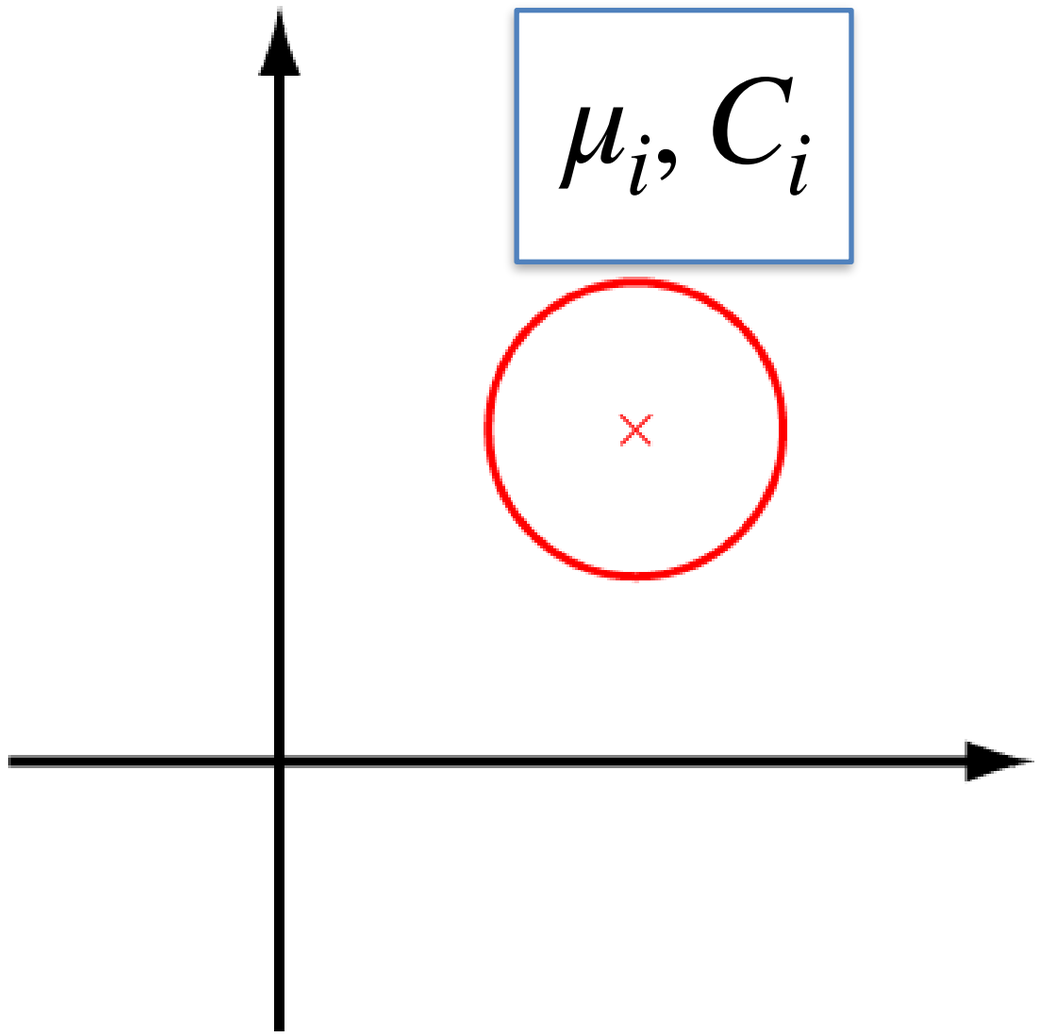
Mohammad Ghavamzadeh*
INRIA Lille - Team SequeL & Adobe Research
mohammad.ghavamzadeh@inria.fr

Bruno Scherrer
INRIA Nancy - Grand Est, Team Maia, FRANCE
bruno.scherrer@inria.fr

Covariance Matrix Adaptation (CMA-ES)

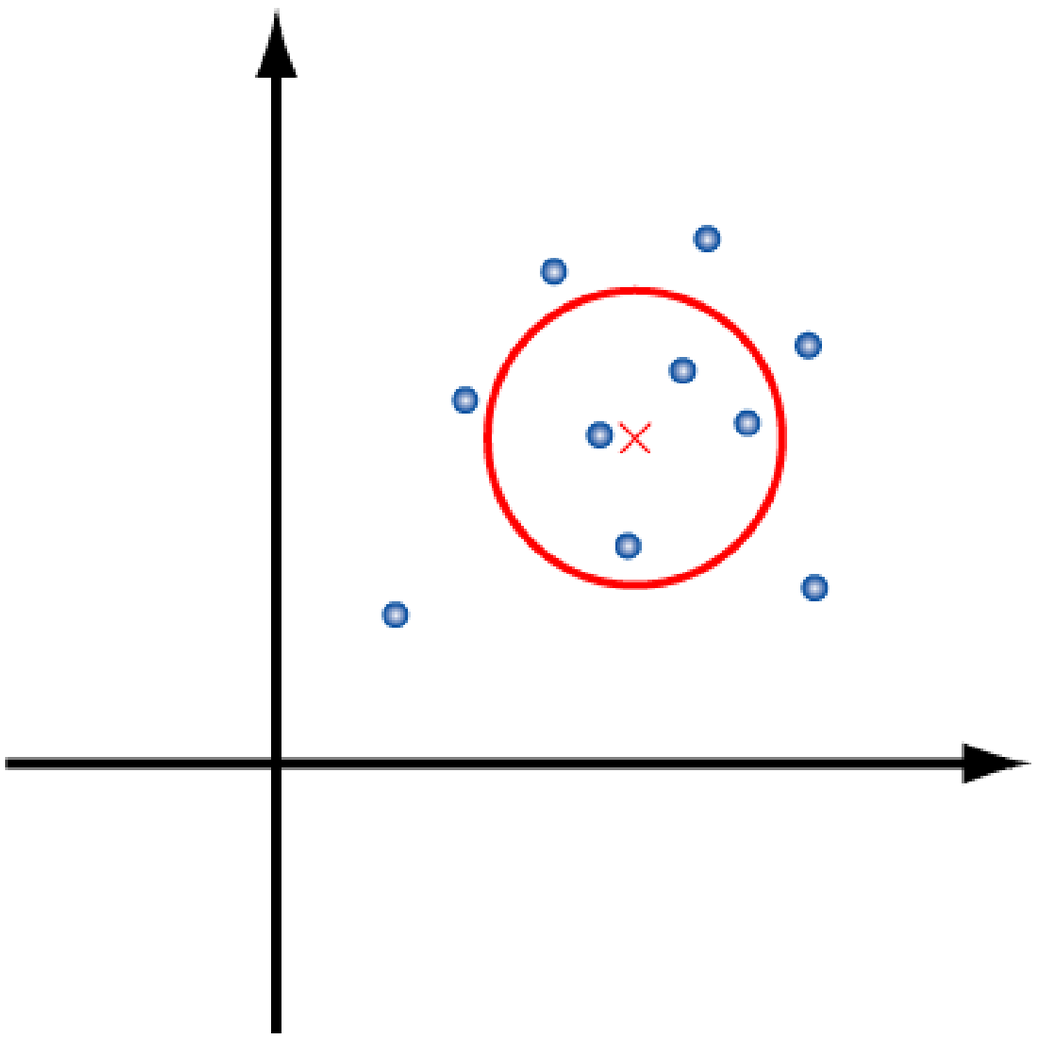
- Policy parameters are sampled from a multivariate Gaussian distribution with a full covariance matrix.
- We will update the mean and the covariance matrix to model samples that have highest fitness scores.

Covariance Matrix Adaptation



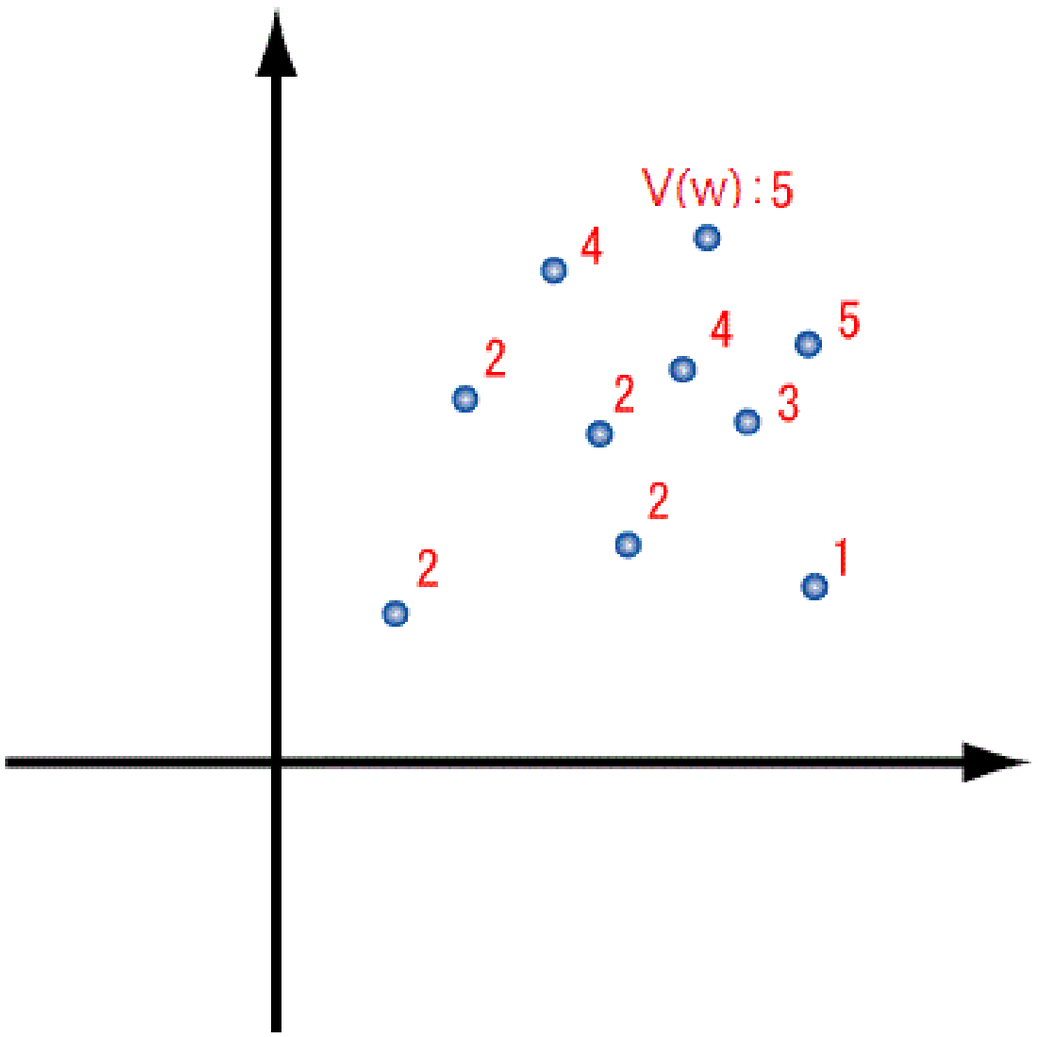
- Sample
- Select elites
- Update mean
- Update covariance
- iterate

Covariance Matrix Adaptation



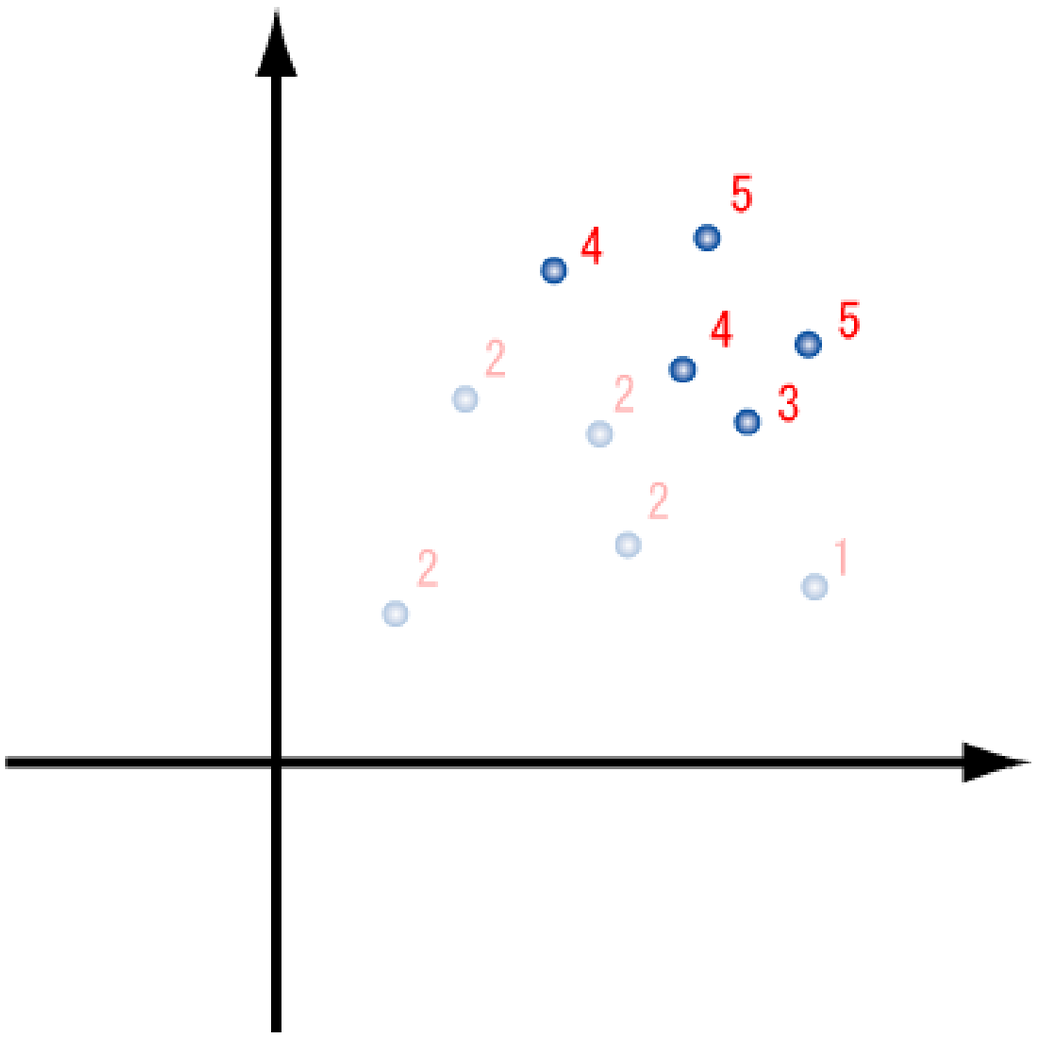
- Sample
- Select elites
- Update mean
- Update covariance
- iterate

Covariance Matrix Adaptation



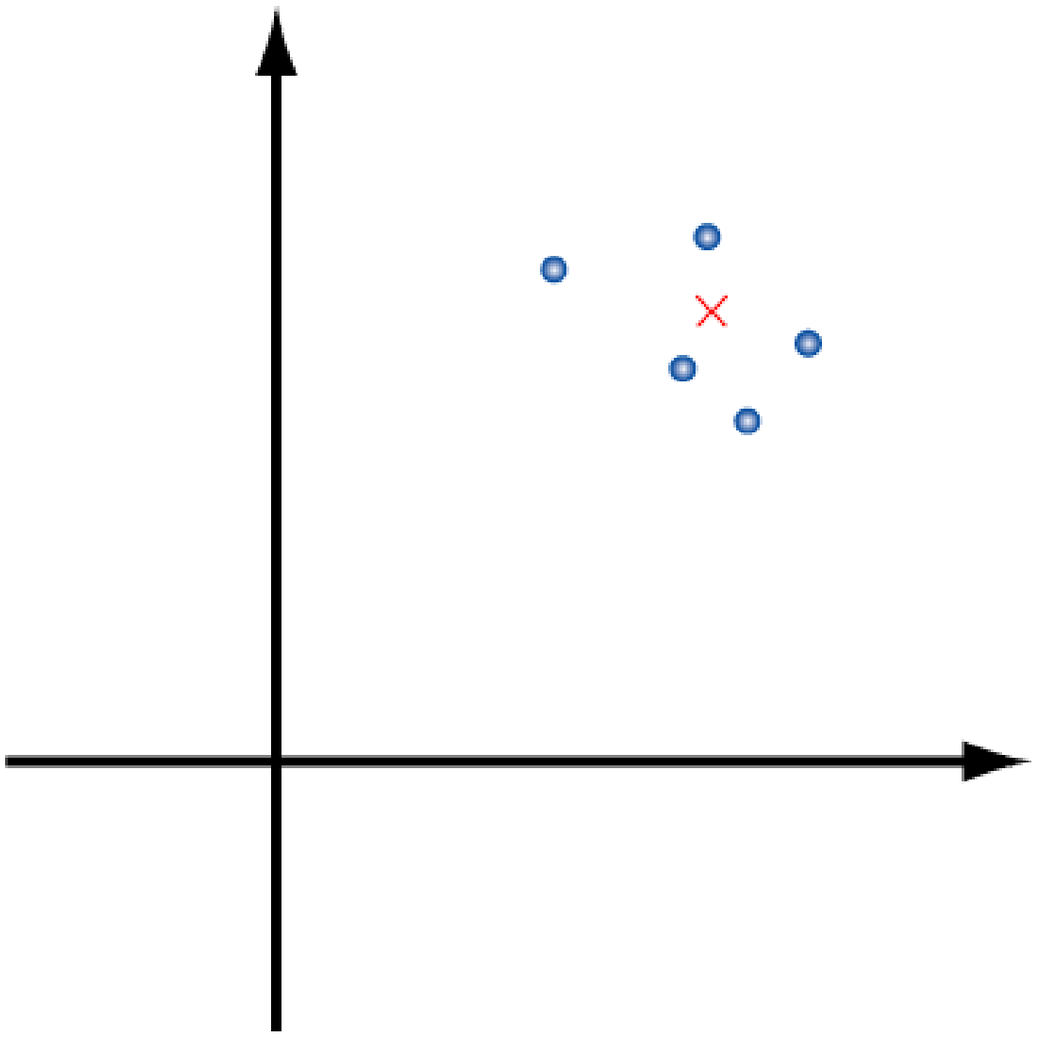
- Sample
- Select elites
- Update mean
- Update covariance
- iterate

Covariance Matrix Adaptation



- Sample
- Select elites
- Update mean
- Update covariance
- iterate

Covariance Matrix Adaptation

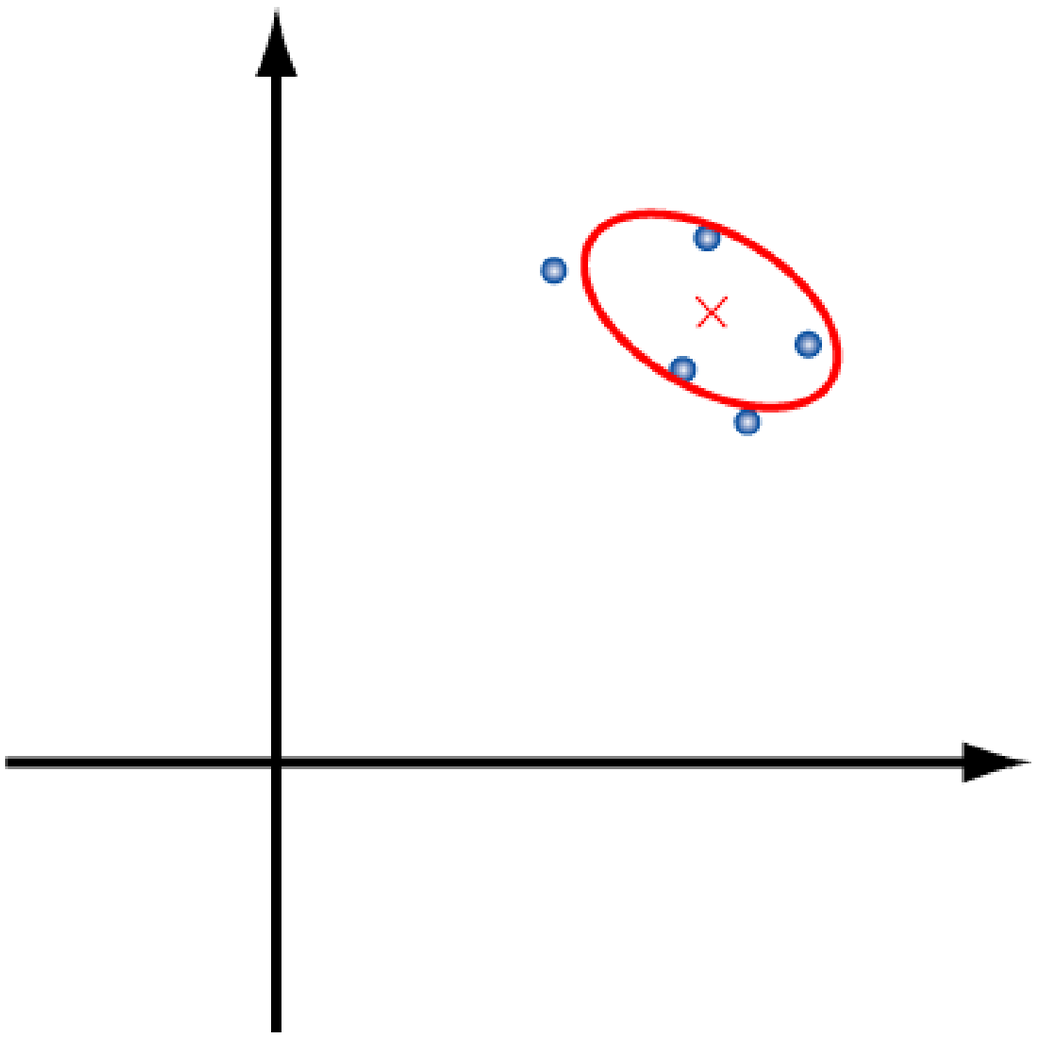


- Sample
- Select elites
- Update mean
- Update covariance
- iterate

$$\mu_{t+1} = \mu_t + \alpha \sum_{i=1}^{n_{elit}} w_i(\theta_i^{elit,t} - \mu_t)$$

$$\mu_{t+1} = \sum_{i=1}^{n_{elit}} w_i \theta_i^{elit,t}$$

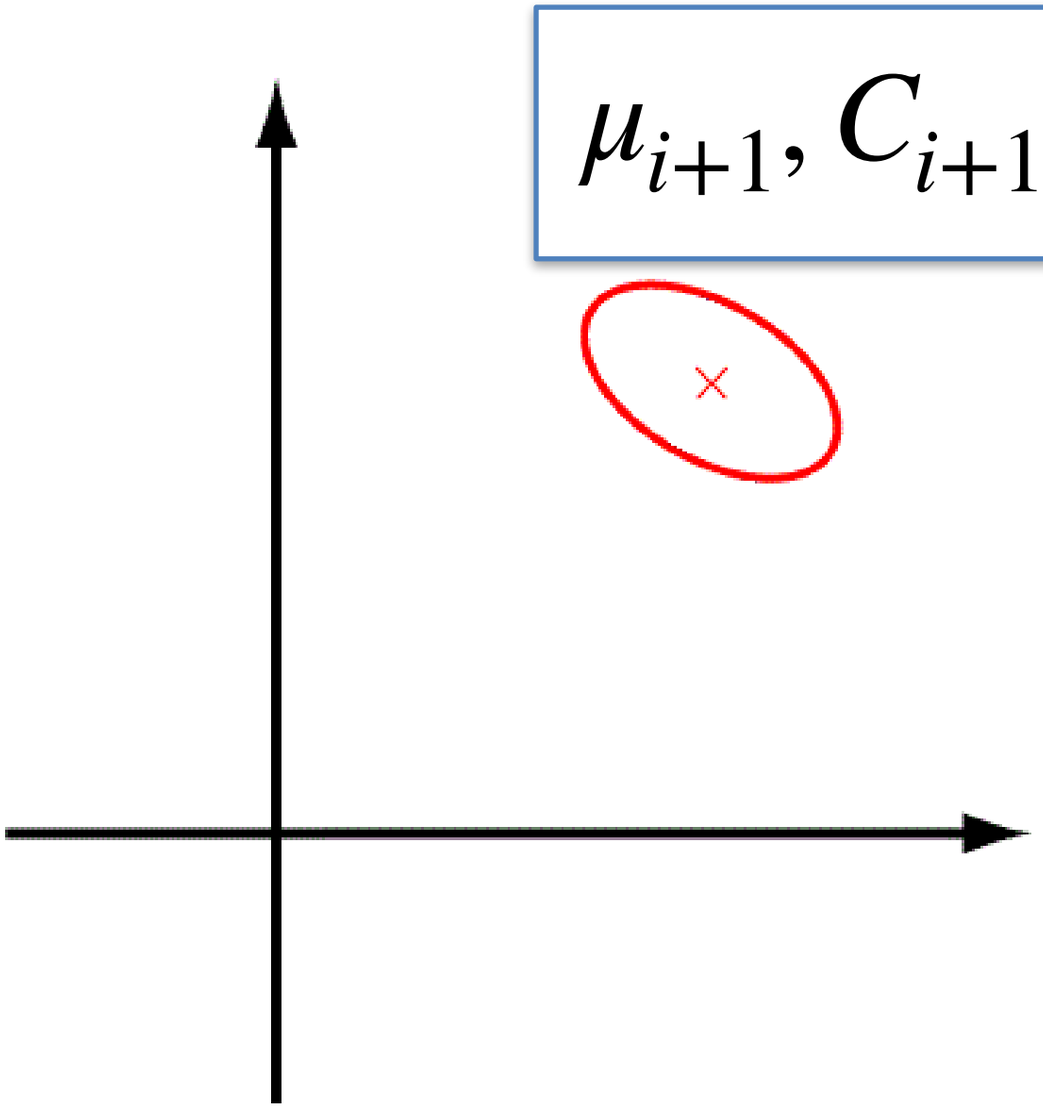
Covariance Matrix Adaptation



- Sample
- Select elites
- Update mean
- Update covariance
- iterate

$$\Sigma_{t+1} = Cov(\theta_1^{elit,t}, \theta_2^{elit,t}, \dots) + \epsilon I$$

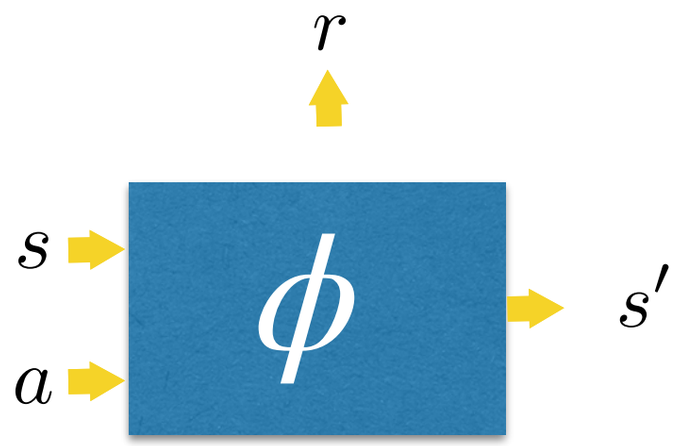
Covariance Matrix Adaptation



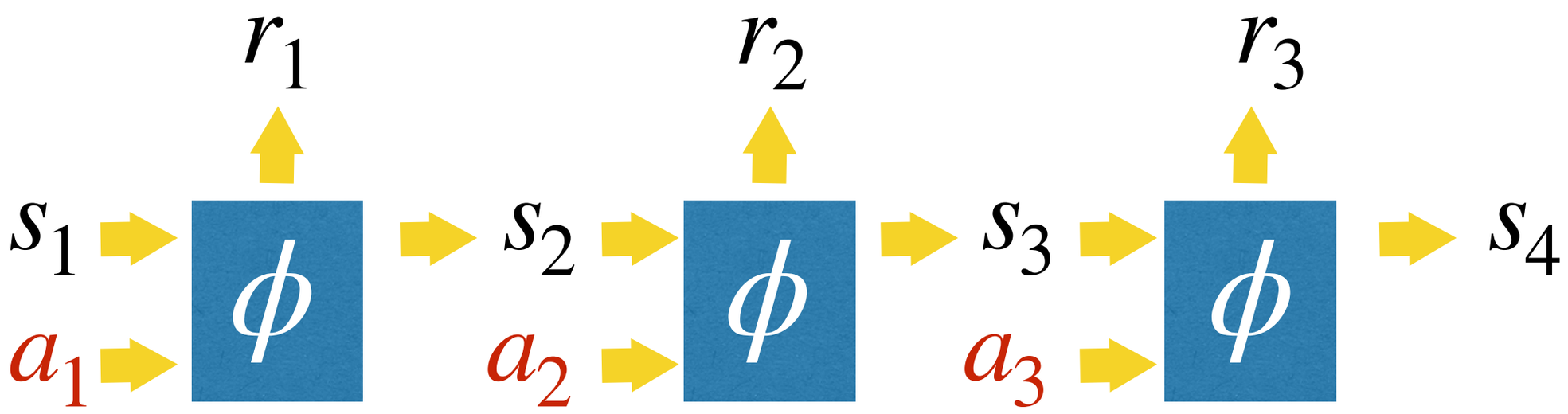
- Sample
- Select elites
- Update mean
- Update covariance
- iterate

Evolution for Action Selection

- A learned model:

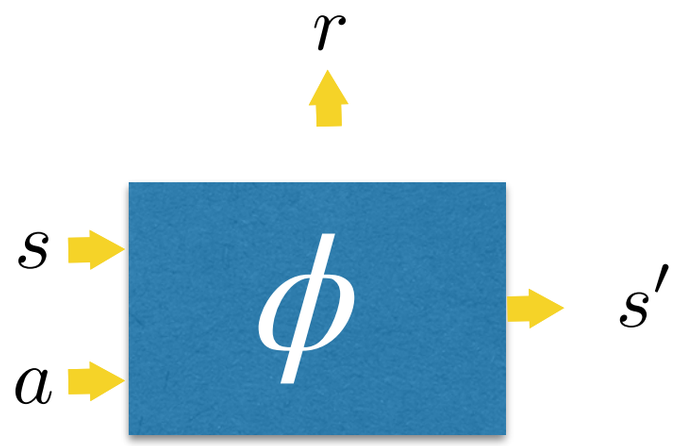


- Objective: maximizing sum of rewards

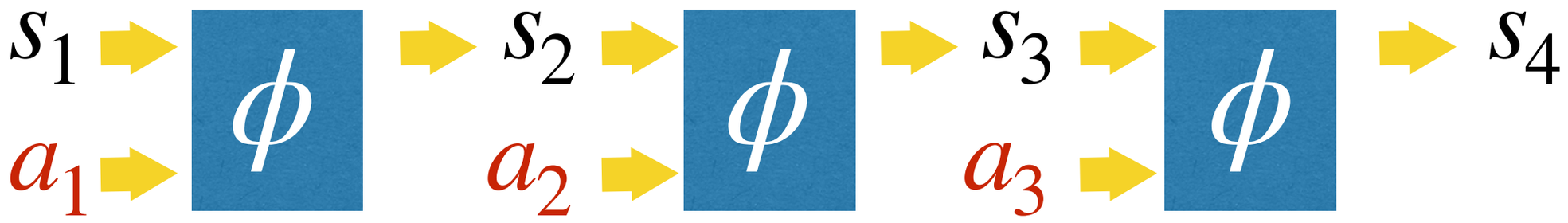


Evolution for Action Selection

- A learned model:



- Objective: reaching a desired state: $\|s_4 - s_*\|$



Natural Evolutionary Strategies (NES)

- In CEM and CMA-ES, we have been selecting the best (elite) parameter offsprings to update the parameter distribution.
- NES considers every offspring.

Natural Evolutionary Strategies

- Consider the parameters of our policy $\theta \in \mathbb{R}^d$ follow a Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and diagonal and fixed covariance matrix $\sigma^2 \mathbf{I}_d : \theta \sim P_\mu(\theta)$.

- Goal: $\max_{\mu} . \ \mathbb{E}_{\theta \sim P_\mu(\theta)} \mathrm{F}(\theta)$, where fitness score $\mathrm{F}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta, s_0 \sim \mu_0(s)} R(\tau)$.

Natural Evolutionary Strategies

- Consider the parameters of our policy $\theta \in \mathbb{R}^d$ follow a Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and diaginal and fixed covariance matrix $\sigma^2 \mathbf{I}_d : \theta \sim P_\mu(\theta)$.

- Goal: $\max_\mu . \ \mathbb{E}_{\theta \sim P_\mu(\theta)} \mathrm{F}(\theta)$, where fitness score $\mathrm{F}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta, s_0 \sim \mu_0(s)} R(\tau)$.

$$\nabla_\mu \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\mathrm{F}(\theta) \right] = \nabla_\mu \int P_\mu(\theta) \mathrm{F}(\theta) d\theta$$

Natural Evolutionary Strategies

- Consider the parameters of our policy $\theta \in \mathbb{R}^d$ follow a Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and diaginal and fixed covariance matrix $\sigma^2 \mathbf{I}_d : \theta \sim P_\mu(\theta)$.

- Goal: $\max_\mu . \ \mathbb{E}_{\theta \sim P_\mu(\theta)} \mathrm{F}(\theta)$, where fitness score $\mathrm{F}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta, s_0 \sim \mu_0(s)} R(\tau)$.

$$\begin{aligned} \nabla_\mu \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\mathrm{F}(\theta) \right] &= \nabla_\mu \int P_\mu(\theta) \mathrm{F}(\theta) d\theta \\ &= \int \nabla_\mu P_\mu(\theta) \mathrm{F}(\theta) d\theta \end{aligned}$$

Natural Evolutionary Strategies

- Consider the parameters of our policy $\theta \in \mathbb{R}^d$ follow a Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and diagonal and fixed covariance matrix $\sigma^2 \mathbf{I}_d : \theta \sim P_\mu(\theta)$.

- Goal: $\max_\mu . \ \mathbb{E}_{\theta \sim P_\mu(\theta)} \mathrm{F}(\theta)$, where fitness score $\mathrm{F}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta, s_0 \sim \mu_0(s)} R(\tau)$.

$$
\begin{aligned}
\nabla_\mu \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\mathrm{F}(\theta) \right] &= \nabla_\mu \int P_\mu(\theta) \mathrm{F}(\theta) d\theta \\
&= \int \nabla_\mu P_\mu(\theta) \mathrm{F}(\theta) d\theta \\
&= \int P_\mu(\theta) \frac{\nabla_\mu P_\mu(\theta)}{P_\mu(\theta)} \mathrm{F}(\theta) d\theta
\end{aligned}
$$

Natural Evolutionary Strategies

- Consider the parameters of our policy $\theta \in \mathbb{R}^d$ follow a Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and diagonal and fixed covariance matrix $\sigma^2 \mathbf{I}_d : \theta \sim P_\mu(\theta)$.

- Goal: $\max_\mu . \ \mathbb{E}_{\theta \sim P_\mu(\theta)} \mathrm{F}(\theta)$, where fitness score $\mathrm{F}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta, s_0 \sim \mu_0(s)} R(\tau)$.

$$
\begin{aligned}
\nabla_\mu \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\mathrm{F}(\theta) \right] &= \nabla_\mu \int P_\mu(\theta) \mathrm{F}(\theta) d\theta \\
&= \int \nabla_\mu P_\mu(\theta) \mathrm{F}(\theta) d\theta \\
&= \int P_\mu(\theta) \frac{\nabla_\mu P_\mu(\theta)}{P_\mu(\theta)} \mathrm{F}(\theta) d\theta \\
&= \int P_\mu(\theta) \nabla_\mu \log P_\mu(\theta) \mathrm{F}(\theta) d\theta
\end{aligned}
$$

Natural Evolutionary Strategies

- Consider the parameters of our policy $\theta \in \mathbb{R}^d$ follow a Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and diagonal and fixed covariance matrix $\sigma^2 \mathbf{I}_d : \theta \sim P_\mu(\theta)$.

- Goal: $\max_\mu . \ \mathbb{E}_{\theta \sim P_\mu(\theta)} \mathrm{F}(\theta)$, where fitness score $\mathrm{F}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta, s_0 \sim \mu_0(s)} R(\tau)$.

$$
\begin{aligned}
\nabla_\mu \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\mathrm{F}(\theta) \right] &= \nabla_\mu \int P_\mu(\theta) \mathrm{F}(\theta) d\theta \\
&= \int \nabla_\mu P_\mu(\theta) \mathrm{F}(\theta) d\theta \\
&= \int P_\mu(\theta) \frac{\nabla_\mu P_\mu(\theta)}{P_\mu(\theta)} \mathrm{F}(\theta) d\theta \\
&= \int P_\mu(\theta) \nabla_\mu \log P_\mu(\theta) \mathrm{F}(\theta) d\theta \\
&= \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\nabla_\mu \log P_\mu(\theta) \mathrm{F}(\theta) \right]
\end{aligned}
$$

We approximate expectations by sampling!

Natural Evolutionary Strategies

- Consider the parameters of our policy $\theta \in \mathbb{R}^d$ follow a Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and diagonal and fixed covariance matrix $\sigma^2 \mathbf{I}_d : \theta \sim P_\mu(\theta)$.

- Goal: $\max_\mu . \ \mathbb{E}_{\theta \sim P_\mu(\theta)} \mathrm{F}(\theta)$, where fitness score $\mathrm{F}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta, s_0 \sim \mu_0(s)} R(\tau)$.

$$
\begin{aligned}
\nabla_\mu \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\mathrm{F}(\theta) \right] &= \nabla_\mu \int P_\mu(\theta) \mathrm{F}(\theta) d\theta \\
&= \int \nabla_\mu P_\mu(\theta) \mathrm{F}(\theta) d\theta \\
&= \int P_\mu(\theta) \frac{\nabla_\mu P_\mu(\theta)}{P_\mu(\theta)} \mathrm{F}(\theta) d\theta \\
&= \int P_\mu(\theta) \nabla_\mu \log P_\mu(\theta) \mathrm{F}(\theta) d\theta \\
&= \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\nabla_\mu \log P_\mu(\theta) \mathrm{F}(\theta) \right] \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \left[\nabla_\mu \log P_\mu(\theta) \mid_{\theta=\theta_i} \mathrm{F}(\theta_i) \right] \quad \theta_i \sim P_\mu(\theta)
\end{aligned}
$$

Policy Gradients

$$\max_{\theta} . \; U(\theta) = \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[R(\tau) \right]$$

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[R(\tau) \right] \\
&= \nabla_\theta \sum_\tau P_\theta(\tau) R(\tau) \\
&= \sum_\tau \nabla_\theta P_\theta(\tau) R(\tau) \\
&= \sum_\tau P_\theta(\tau) \frac{\nabla_\theta P_\theta(\tau)}{P_\theta(\tau)} R(\tau) \\
&= \sum_\tau P_\theta(\tau) \nabla_\theta \log P_\theta(\tau) R(\tau) \\
&= \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[\nabla_\theta \log P_\theta(\tau) R(\tau) \right]
\end{aligned}$$

Sample estimate:

$$\nabla_\theta U(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log P_\theta(\tau^{(i)}) R(\tau^{(i)})$$

Evolutionary Methods

$$\max_{\mu} . \; U(\mu) = \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[F(\theta) \right]$$

$$\begin{aligned}
\nabla_\mu U(\mu) &= \nabla_\mu \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[F(\theta) \right] \\
&= \nabla_\mu \int P_\mu(\theta) F(\theta) d\theta \\
&= \int \nabla_\mu P_\mu(\theta) F(\theta) d\theta \\
&= \int P_\mu(\theta) \frac{\nabla_\mu P_\mu(\theta)}{P_\mu(\theta)} F(\theta) d\theta \\
&= \int P_\mu(\theta) \nabla_\mu \log P_\mu(\theta) F(\theta) d\theta \\
&= \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\nabla_\mu \log P_\mu(\theta) F(\theta) \right]
\end{aligned}$$

Sample estimate:

$$\nabla_\mu U(\mu) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\mu \log P_\mu(\theta^{(i)}) F(\theta^{(i)})$$

Policy gradients VS Evolutionary methods

Considers distribution over actions

$$\max_{\theta} . \; U(\theta) = \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[R(\tau) \right]$$

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[R(\tau) \right] \\
&= \nabla_\theta \sum_\tau P_\theta(\tau) R(\tau) \\
&= \sum_\tau \nabla_\theta P_\theta(\tau) R(\tau) \\
&= \sum_\tau P_\theta(\tau) \frac{\nabla_\theta P_\theta(\tau)}{P_\theta(\tau)} R(\tau) \\
&= \sum_\tau P_\theta(\tau) \nabla_\theta \log P_\theta(\tau) R(\tau) \\
&= \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[\nabla_\theta \log P_\theta(\tau) R(\tau) \right]
\end{aligned}$$

Sample estimate:

$$\nabla_\theta U(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log P_\theta(\tau^{(i)}) R(\tau^{(i)})$$

Considers distribution over policy parameters

$$\max_{\mu} . \; U(\mu) = \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[F(\theta) \right]$$

$$\begin{aligned}
\nabla_\mu U(\mu) &= \nabla_\mu \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[F(\theta) \right] \\
&= \nabla_\mu \int P_\mu(\theta) F(\theta) d\theta \\
&= \int \nabla_\mu P_\mu(\theta) F(\theta) d\theta \\
&= \int P_\mu(\theta) \frac{\nabla_\mu P_\mu(\theta)}{P_\mu(\theta)} F(\theta) d\theta \\
&= \int P_\mu(\theta) \nabla_\mu \log P_\mu(\theta) F(\theta) d\theta \\
&= \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\nabla_\mu \log P_\mu(\theta) F(\theta) \right]
\end{aligned}$$

Sample estimate:

$$\nabla_\mu U(\mu) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\mu \log P_\mu(\theta^{(i)}) F(\theta^{(i)})$$

Policy gradients VS Evolutionary methods

- We are sampling in both cases…
  - PG: sampling in action space
  - ES: sampling in parameter space

Reminder: Sampling from a multivariate Gaussian

We want to generate samples $\theta_i \sim P_\mu(\theta)$ where P is the Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and diagonal and fixed covariance matrix $\sigma^2 \mathbf{I}_d$.

Imagine we have access to random vectors $\epsilon_i \in \mathbb{R}^d$, $\epsilon_i \sim \mathcal{N}(0, \mathbf{I}_d)$

$$\theta_1 = \mu + \sigma\epsilon_1$$

$$\theta_2 = \mu + \sigma\epsilon_2$$

The samples have the desired mean and variance

Reparametrization Trick

- A sample from a normal distribution $z \sim N(\mu, \Sigma)$ can be rewritten as follows:

$$z = \mu + \Sigma^{\frac{1}{2}}\epsilon \text{ where } \epsilon \sim N(0, \mathrm{I}).$$

- We just need a standard normal sampler to sample from an arbitrary normal distribution.

Cumulative Distribution Function (CDF)

Given a PDF $p(x)$,

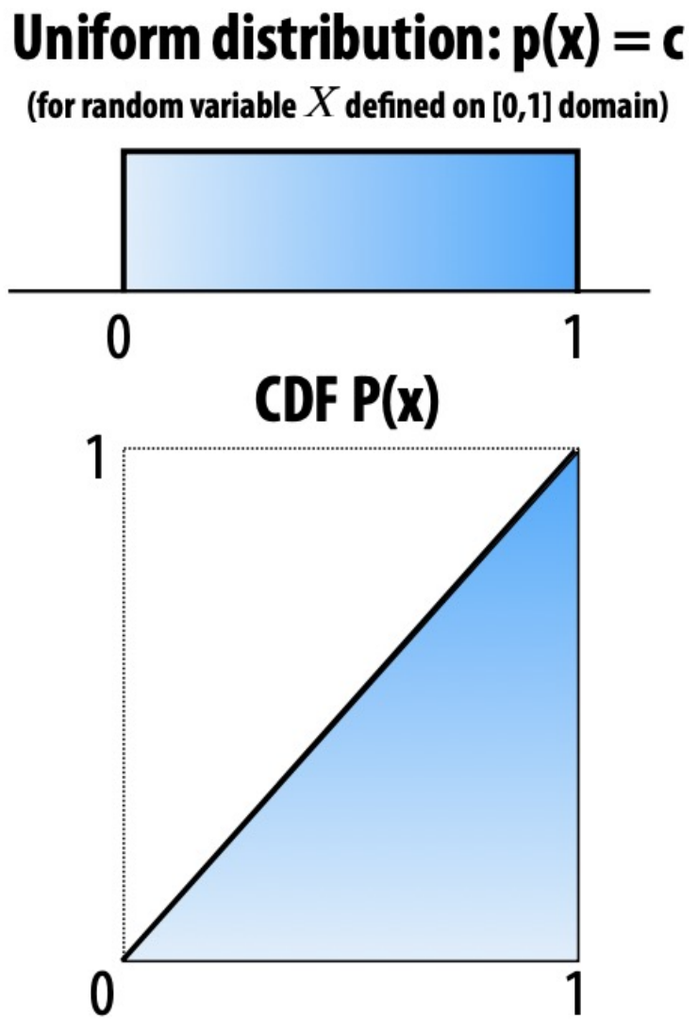
- **CDF** $F_X(x)$: $\mathbb{R} \to [0,1]$,

$$F_X(x) = \Pr(X \leq x) = \int_{-\infty}^{x} p(t)\mathrm{d}t$$

- $\Pr(a < X \leq b)$

$$= \int_{a}^{b} p(t)\mathrm{d}t = F_X(b) - F_X(a)$$

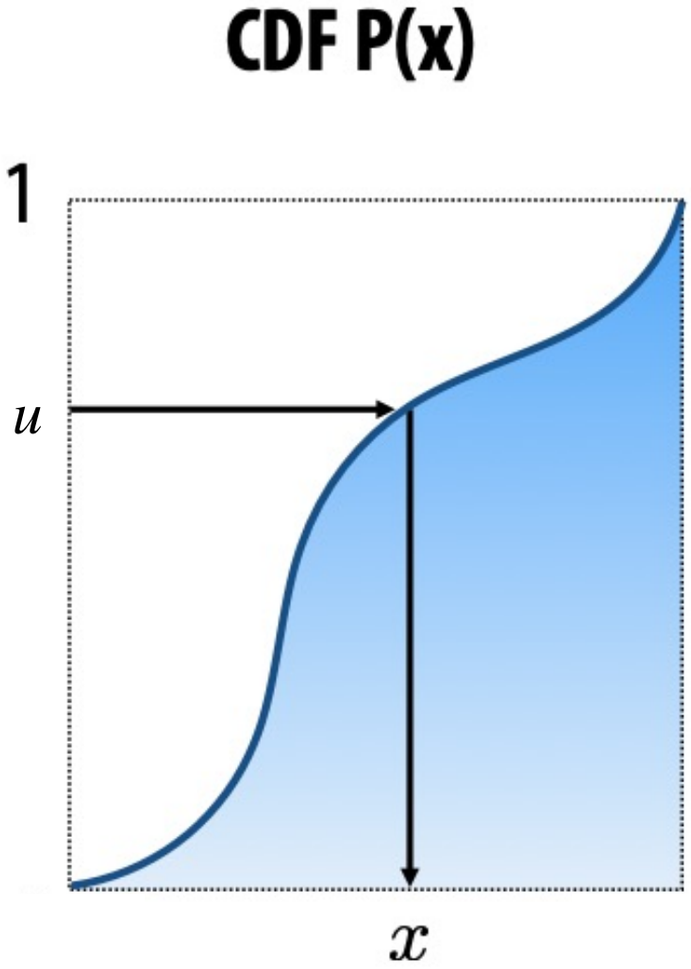
- $F_X(1) = 1$



Inverse Transform Sampling

To randomly sample based on the given PDF,

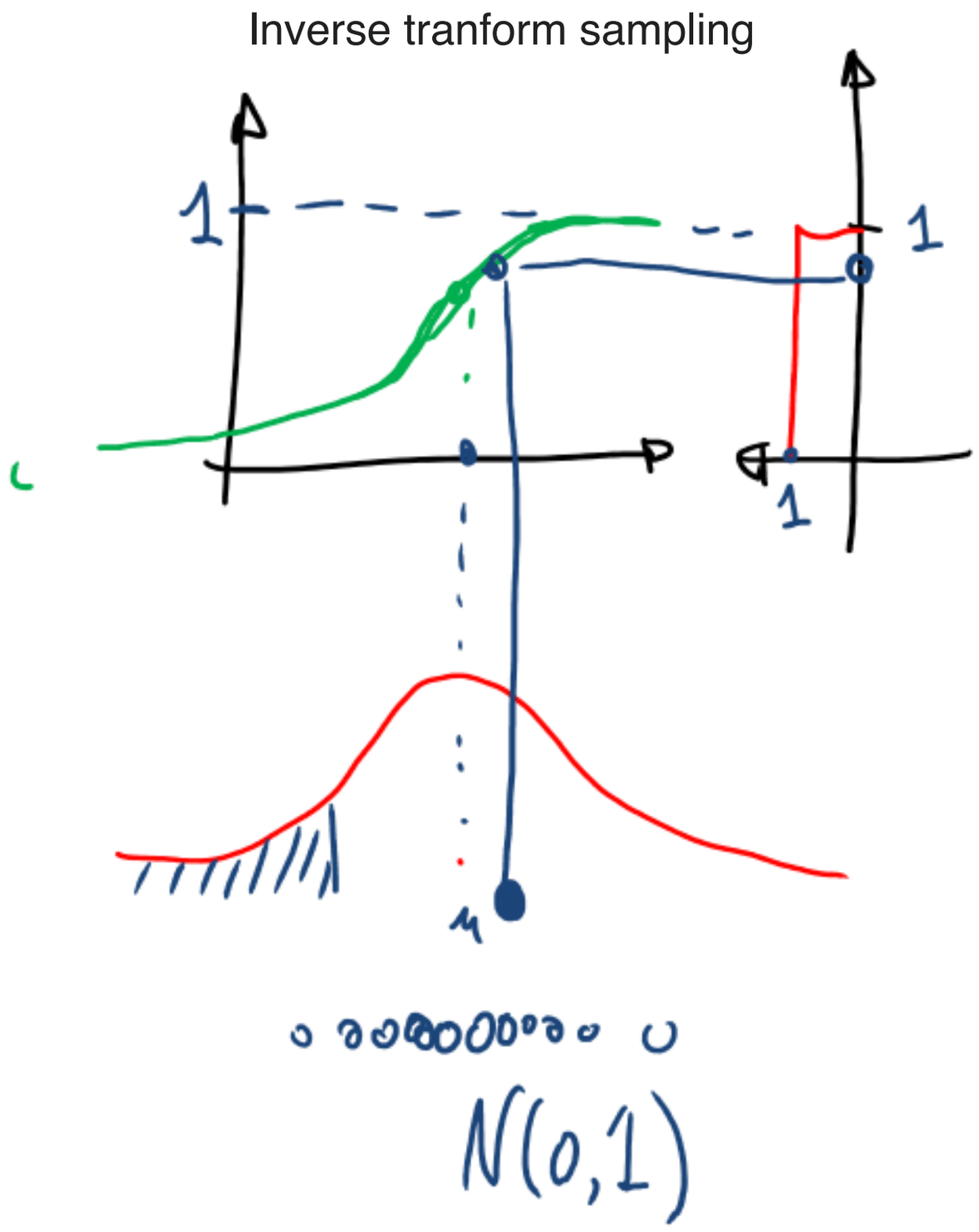
- Compute **CDF** $F_X(x)$
- Draw $u \sim \mathcal{U}(0,1)$.
- Take $x = F_X^{-1}(u)$.



Reparametrization Trick: Sampling from a general Gaussian density

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

$$x_i \sim \mathcal{N}(\mu, \sigma^2)$$
$$\sim \mu + \sigma\mathcal{N}(0,1)$$



$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_i \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right)$$

$\mathbf{x}$ μ Σ

$$\mathbf{x} \sim \mathcal{N}(\mu, \Sigma) \qquad x \sim \mu + L\mathcal{N}(0,I)$$

Cholesky decomposition $\Sigma = LL^T$

Reparametrization Trick: Sampling from a general Gaussian density

- A sample from a normal distribution $z \sim N(\mu, \Sigma)$ can be rewritten as follows:

$$z = \mu + \Sigma^{\frac{1}{2}}\epsilon \text{ where } \epsilon \sim N(0, \mathrm{I}).$$

- We just need a standard normal sampler to sample from an arbitrary normal distribution.

A concrete example

- Suppose $\theta \sim P_\mu(\theta)$ is a Gaussian distribution with mean μ, and covariance matrix $\sigma^2 \mathbf{I}_d$.

$$\log P_\mu(\theta) = -\frac{\|\theta - \mu\|^2}{2\sigma^2} + \text{const}$$

$$\nabla_\mu \log P_\mu(\theta) = \frac{\theta - \mu}{\sigma^2}$$

A concrete example

- Suppose $\theta \sim P_\mu(\theta)$ is a Gaussian distribution with mean μ, and covariance matrix $\sigma^2 \mathbf{I}_d$.

$$\log P_\mu(\theta) = -\frac{\|\theta - \mu\|^2}{2\sigma^2} + \text{const}$$

$$\nabla_\mu \log P_\mu(\theta) = \frac{\theta - \mu}{\sigma^2}$$

  - We draw two parameter samples θ_1, θ_2. For each sampled parameter vector, we run the policy and obtain a set of trajectories, or a single trajectory. Then:

$$\mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\nabla_\mu \log P_\mu(\theta) \mathrm{F}(\theta) \right] \approx \frac{1}{N} \sum_{i=1}^{N} \left[\nabla_\mu \log P_\mu(\theta) |_{\theta=\theta_i} \mathrm{F}(\theta_i) \right]$$

A concrete example

- Suppose $\theta \sim P_\mu(\theta)$ is a Gaussian distribution with mean μ, and covariance matrix $\sigma^2 \mathbf{I}_d$.

$$\log P_\mu(\theta) = -\frac{\|\theta - \mu\|^2}{2\sigma^2} + \text{const}$$

$$\nabla_\mu \log P_\mu(\theta) = \frac{\theta - \mu}{\sigma^2}$$

  - We draw two parameter samples θ_1, θ_2. For each sampled parameter vector, we run the policy and obtain a set of trajectories, or a single trajectory. Then:

$$\mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\nabla_\mu \log P_\mu(\theta) \mathrm{F}(\theta) \right] \approx \frac{1}{N} \sum_{i=1}^{N} \left[\nabla_\mu \log P_\mu(\theta) \big|_{\theta=\theta_i} \mathrm{F}(\theta_i) \right]$$

$$\approx \frac{1}{2} \left[\mathrm{F}(\theta_1) \frac{\theta_1 - \mu}{\sigma^2} + \mathrm{F}(\theta_2) \frac{\theta_2 - \mu}{\sigma^2} \right], \text{ where } \mathrm{F}(\theta_1) = R(\tau_1), \mathrm{F}(\theta_2) = R(\tau_2)$$

  - Q:Can we simplify this expression more?

A concrete example

- Suppose $\theta \sim P_\mu(\theta)$ is a Gaussian distribution with mean μ, and covariance matrix $\sigma^2 \mathbf{I}_d$.

$$\log P_\mu(\theta) = -\frac{\|\theta - \mu\|^2}{2\sigma^2} + \text{const}$$

$$\nabla_\mu \log P_\mu(\theta) = \frac{\theta - \mu}{\sigma^2}$$

  - We draw two parameter samples θ_1, θ_2. For each sampled parameter vector, we run the policy and obtain a set of trajectories, or a single trajectory. Then:

$$\mathbb{E}_{\theta \sim P_\mu(\theta)}\left[\nabla_\mu \log P_\mu(\theta) \mathrm{F}(\theta)\right] \approx \frac{1}{N}\sum_{i=1}^{N}\left[\nabla_\mu \log P_\mu(\theta)\,|_{\theta=\theta_i} \mathrm{F}(\theta_i)\right]$$

$$\approx \frac{1}{2}\left[\mathrm{F}(\theta_1)\frac{\theta_1 - \mu}{\sigma^2} + \mathrm{F}(\theta_2)\frac{\theta_2 - \mu}{\sigma^2}\right], \text{ where } \mathrm{F}(\theta_1) = R(\tau_1), \mathrm{F}(\theta_2) = R(\tau_2)$$

$$\boxed{\begin{aligned}\theta_1 &= \mu + \sigma\epsilon_1 \\ \theta_2 &= \mu + \sigma\epsilon_2\end{aligned}}$$

A concrete example

- Suppose $\theta \sim P_\mu(\theta)$ is a Gaussian distribution with mean μ, and covariance matrix $\sigma^2 \mathbf{I}_d$.

$$\log P_\mu(\theta) = -\frac{\|\theta - \mu\|^2}{2\sigma^2} + \text{const}$$

$$\nabla_\mu \log P_\mu(\theta) = \frac{\theta - \mu}{\sigma^2}$$

  - We draw two parameter samples θ_1, θ_2. For each sampled parameter vector, we run the policy and obtain a set of trajectories, or a single trajectory. Then:

$$\mathbb{E}_{\theta \sim P_\mu(\theta)}\left[\nabla_\mu \log P_\mu(\theta) \mathrm{F}(\theta)\right] \approx \frac{1}{N}\sum_{i=1}^{N}\left[\nabla_\mu \log P_\mu(\theta)|_{\theta=\theta_i} \mathrm{F}(\theta_i)\right]$$

$$\boxed{\begin{aligned}\theta_1 &= \mu + \sigma\epsilon_1 \\ \theta_2 &= \mu + \sigma\epsilon_2\end{aligned}}$$

$$\approx \frac{1}{2}\left[\mathrm{F}(\theta_1)\frac{\theta_1 - \mu}{\sigma^2} + \mathrm{F}(\theta_2)\frac{\theta_2 - \mu}{\sigma^2}\right], \text{ where } \mathrm{F}(\theta_1) = R(\tau_1), \mathrm{F}(\theta_2) = R(\tau_2)$$

$$= \frac{1}{2\sigma}\left[\mathrm{F}(\theta_1)\epsilon_1 + \mathrm{F}(\theta_1)\epsilon_2\right]$$

Natural Evolutionary Strategies (NES)

- In CEM and CMA-ES, we have been selecting the best (elite) parameter offsprings.
- NES considers every offspring.

Algorithm 1: Evolutionary Strategies

1. **Input**: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
2. **for** $t = 0,1,2,\cdots$ do
3. Sample $\epsilon_1, \cdots, \epsilon_n \sim \mathcal{N}(0, \mathbf{I}_d)$
4. Compute returns $F_i = F\left(\mu_t + \sigma\epsilon_i\right)$ for $i = 1,2,\cdots,n$
5. Set $\mu_{t+1} \leftarrow \mu_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i$

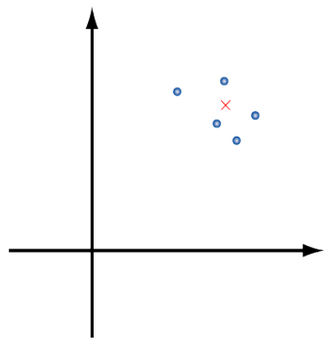
6. end for

Compare the two update rules for the mean

NES: $$\mu_{t+1} = \mu_t + \frac{\alpha}{n\sigma} \sum_{i=1}^{n} \mathrm{F}(\theta_i)\epsilon_i$$

All offsprings participate

CMA-ES: $$\mu_{t+1} = \mu_t + \alpha \sum_{i=1}^{n_{elit}} w_i(\theta_i^{elit,t} - \mu_t)$$



- Sample
- Select elites
- Update mean
- Update covariance
- iterate

Question: Can evolutionary methods scale?

- Evolutionary methods work well on relatively low-dimensional problems (small number of parameter dimensions).
- Can they be used to optimize policies represented as neural networks of thousands parameters?

Evolution Strategies as a Scalable Alternative to Reinforcement Learning

Tim Salimans **Jonathan Ho** **Xi Chen** **Szymon Sidor** **Ilya Sutskever**
OpenAI

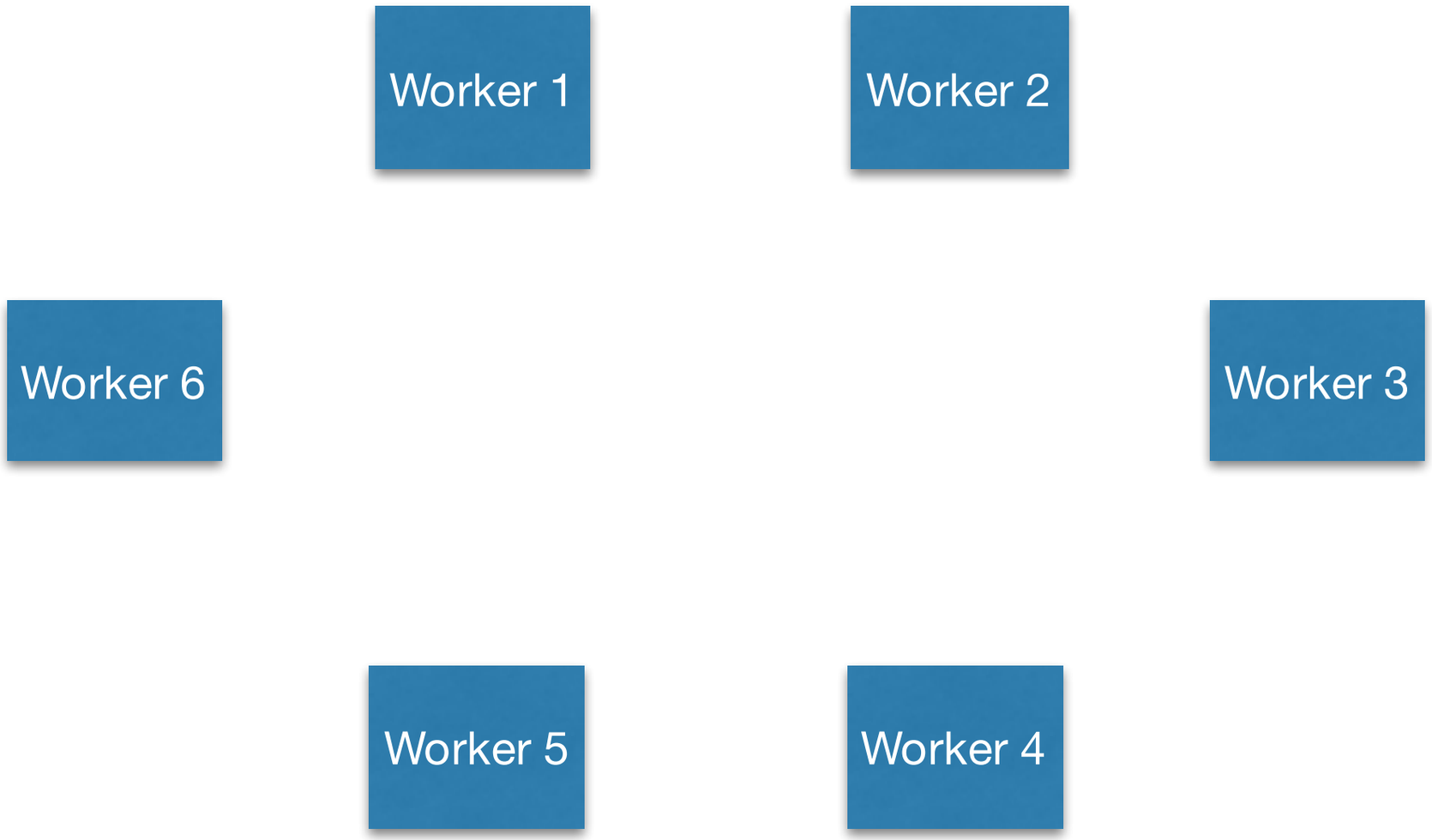
Algorithm 1: Evolutionary Strategies

1. **Input**: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
2. **for** $t = 0,1,2,\cdots$ do
3. Sample $\epsilon_1, \cdots, \epsilon_n \sim \mathcal{N}(0, \mathbf{I}_d)$
4. Compute returns $F_i = F\left(\mu_t + \sigma\epsilon_i\right)$ for $i = 1,2,\cdots,n$
5. Set $\mu_{t+1} \leftarrow \mu_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i$
6. **end for**

Main contribution: Parallelization with a need for tiny only cross-worker communication

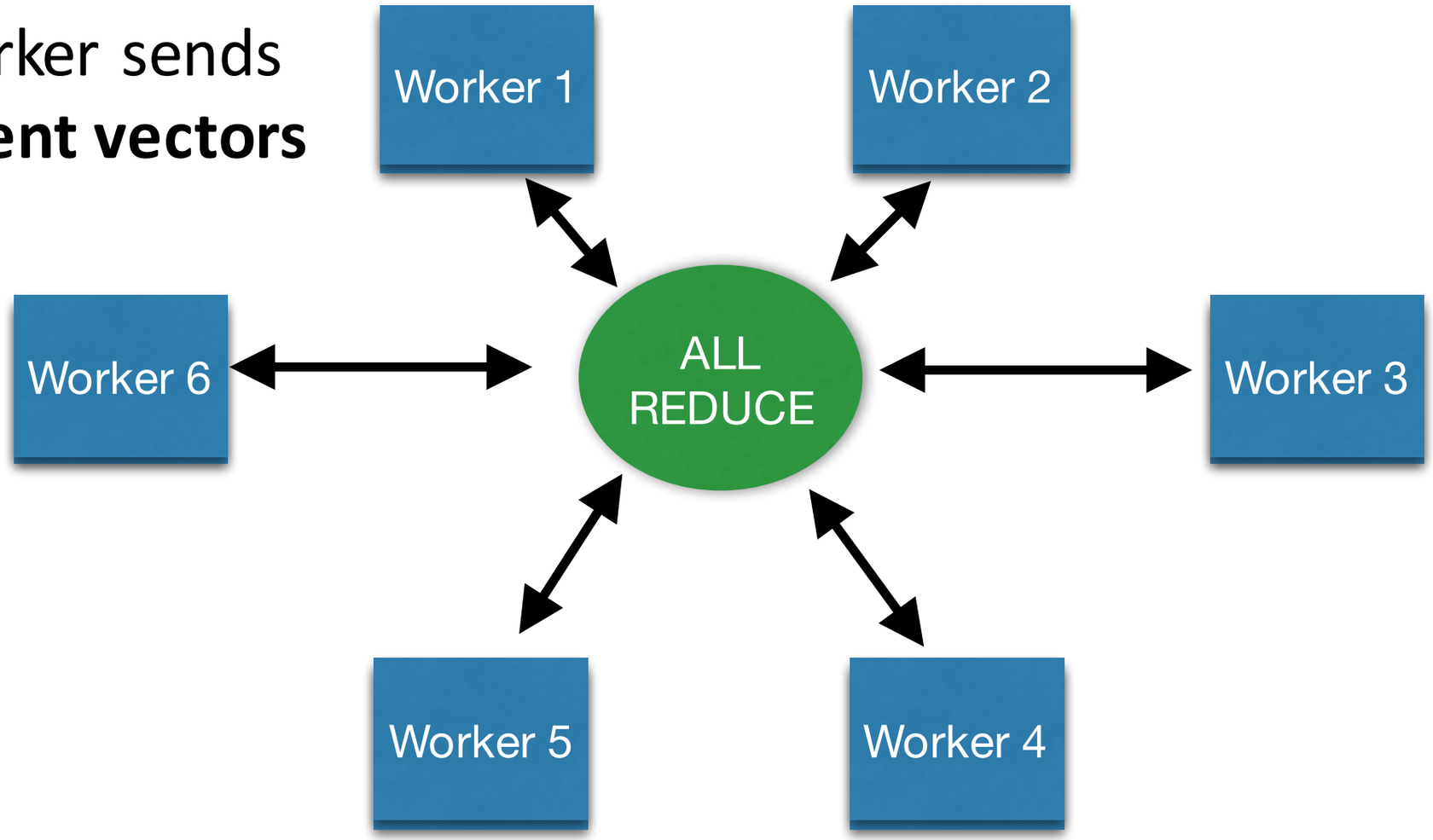
Distributed stochastic gradient descent

Every worker get a set of examples, and computes a gradient vector.
The master averages those gradient vectors.

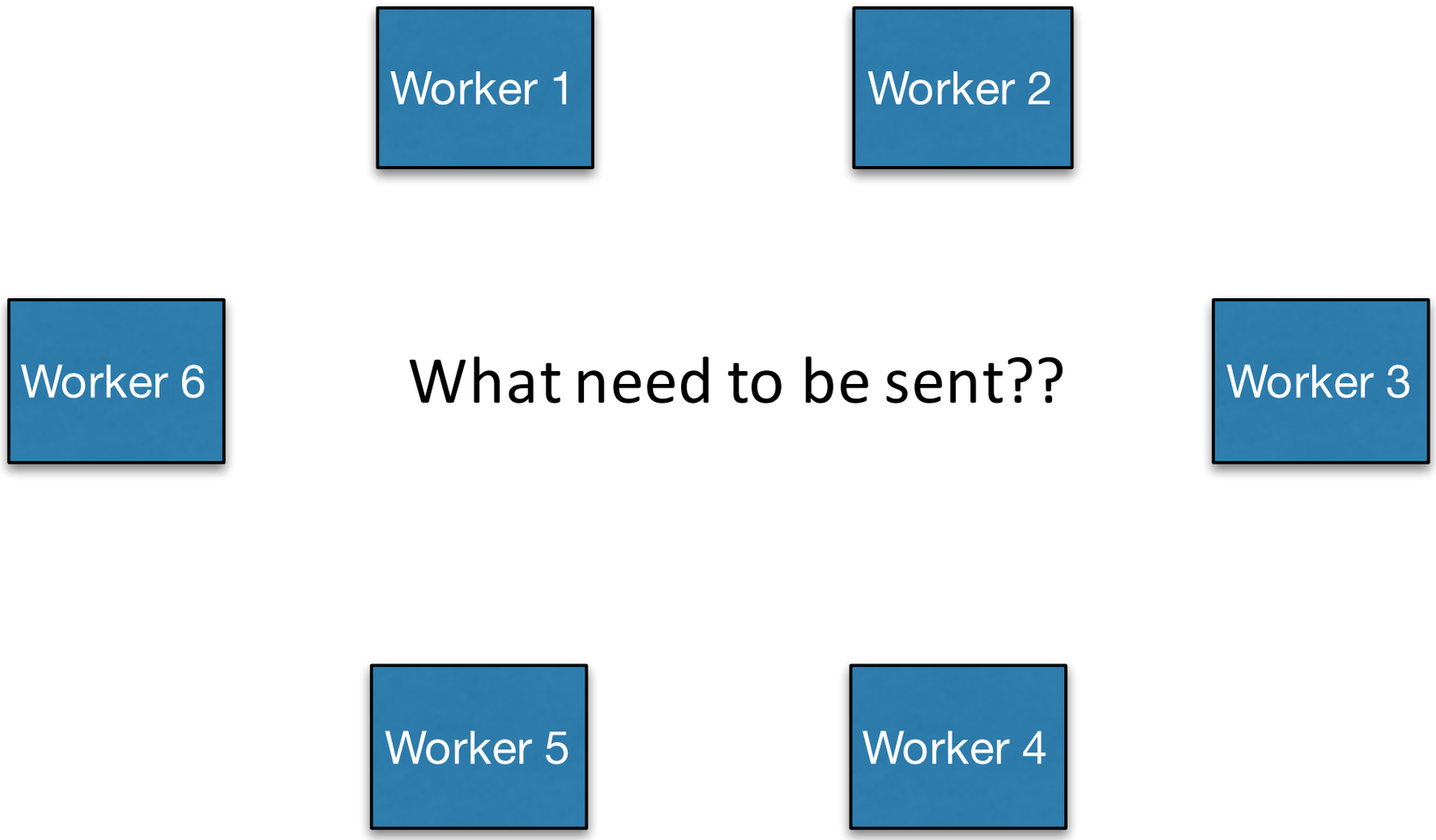


Distributed stochastic gradient descent

Each worker sends
big gradient vectors



Distributed Evolution



Distributed Evolution

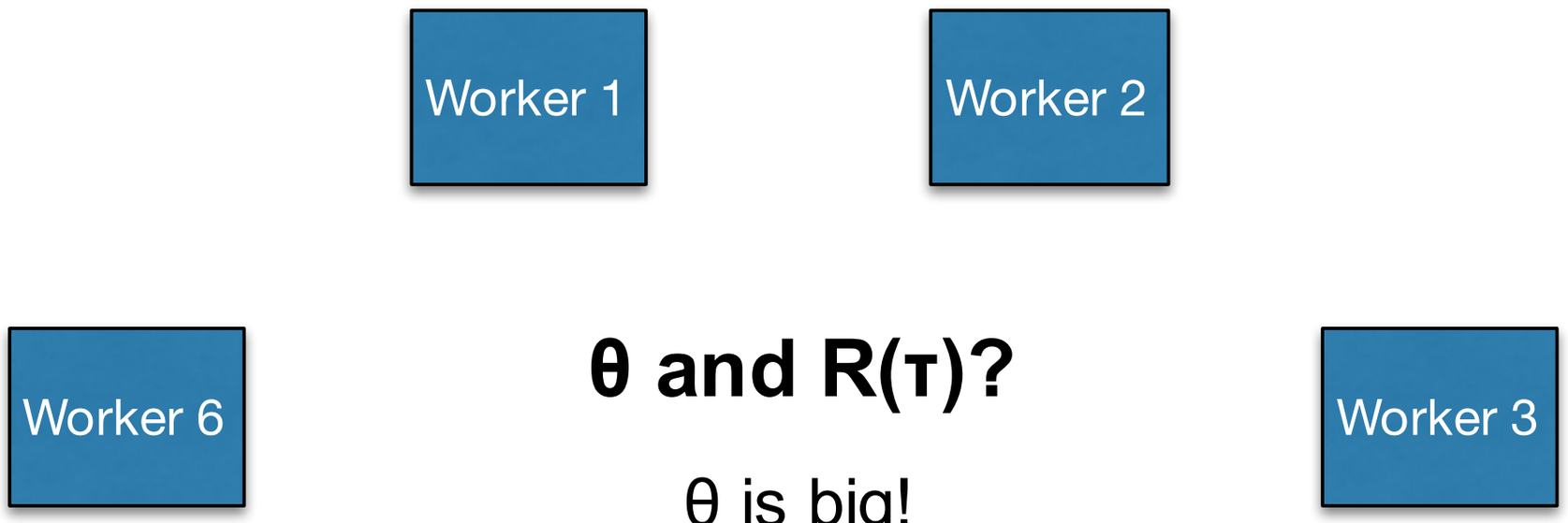
Worker 1

Worker 2

Algorithm 1: Evolutionary Strategies

1. **Input**: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
2. **for** $t = 0,1,2,\cdots$ do
3. Sample $\epsilon_1, \cdots, \epsilon_n \sim \mathcal{N}(0, \mathbf{I}_d)$
4. Compute returns $F_i = F\left(\mu_t + \sigma\epsilon_i\right)$ for $i = 1,2,\cdots,n$
5. Set $\mu_{t+1} \leftarrow \mu_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i$
6. **end for**

Distributed Evolution



Distributed Evolution

Algorithm 2 Parallelized Evolution Strategies

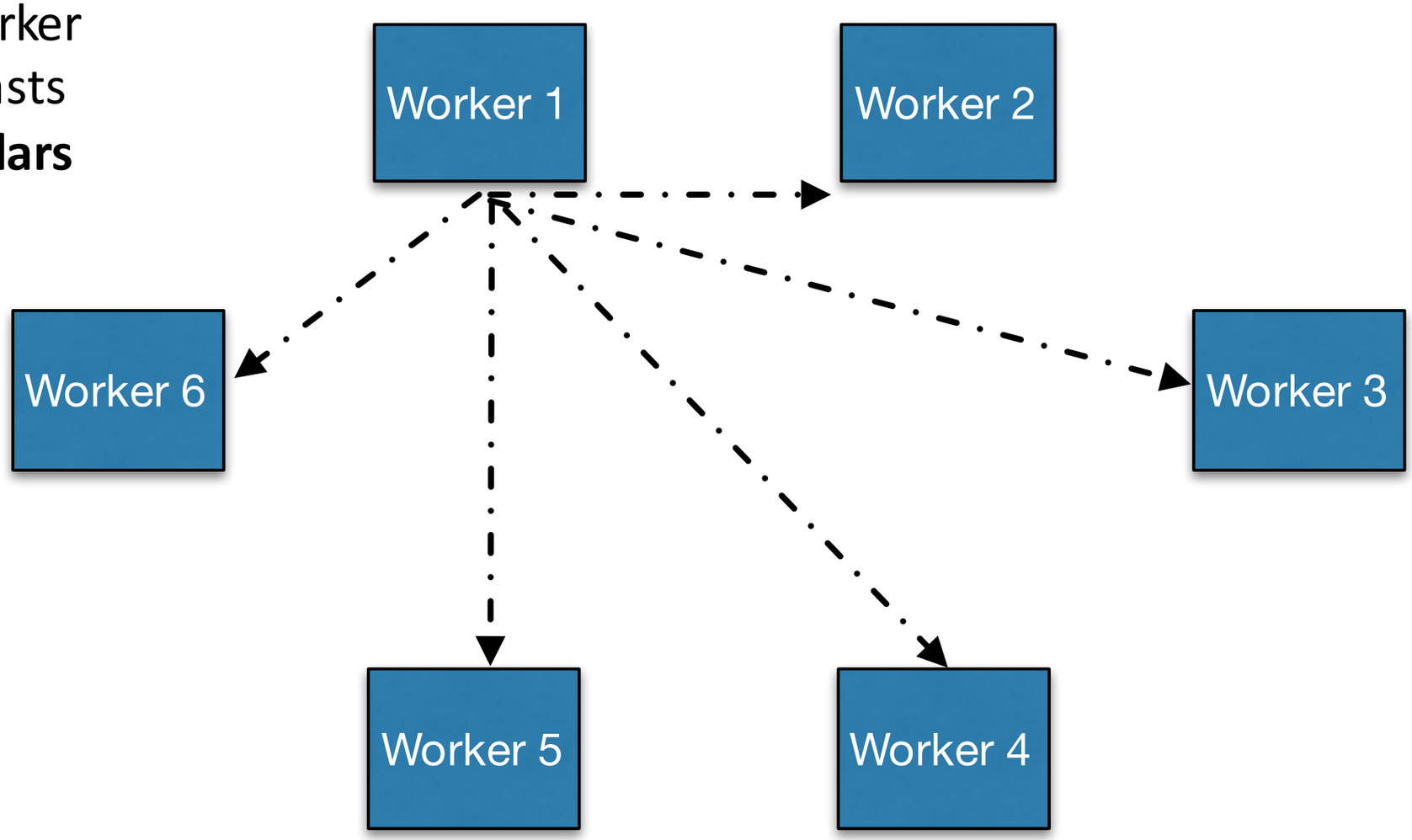
Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
Initialize: n workers with known random seeds, and initial parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
  for each worker $i = 1, \ldots, n$ **do**
    Sample $\epsilon_i \sim \mathcal{N}(0, I)$
    Compute returns $F_i = F(\mu_t + \sigma\epsilon_i)$
  end for
  Send all scalar returns F_i from each worker to every other worker
  for each worker $i = 1, \ldots, n$ **do**
    Reconstruct all perturbations ϵ_j for $j = 1, \ldots, n$
    Set $\mu_{t+1} \leftarrow \mu_t + \alpha \frac{1}{n\sigma} \sum_{j=1}^{n} F_j \epsilon_j$
  end for
end for

Same μ_t for all workers!

[Salimans, Ho, Chen, Sutskever, 2017]

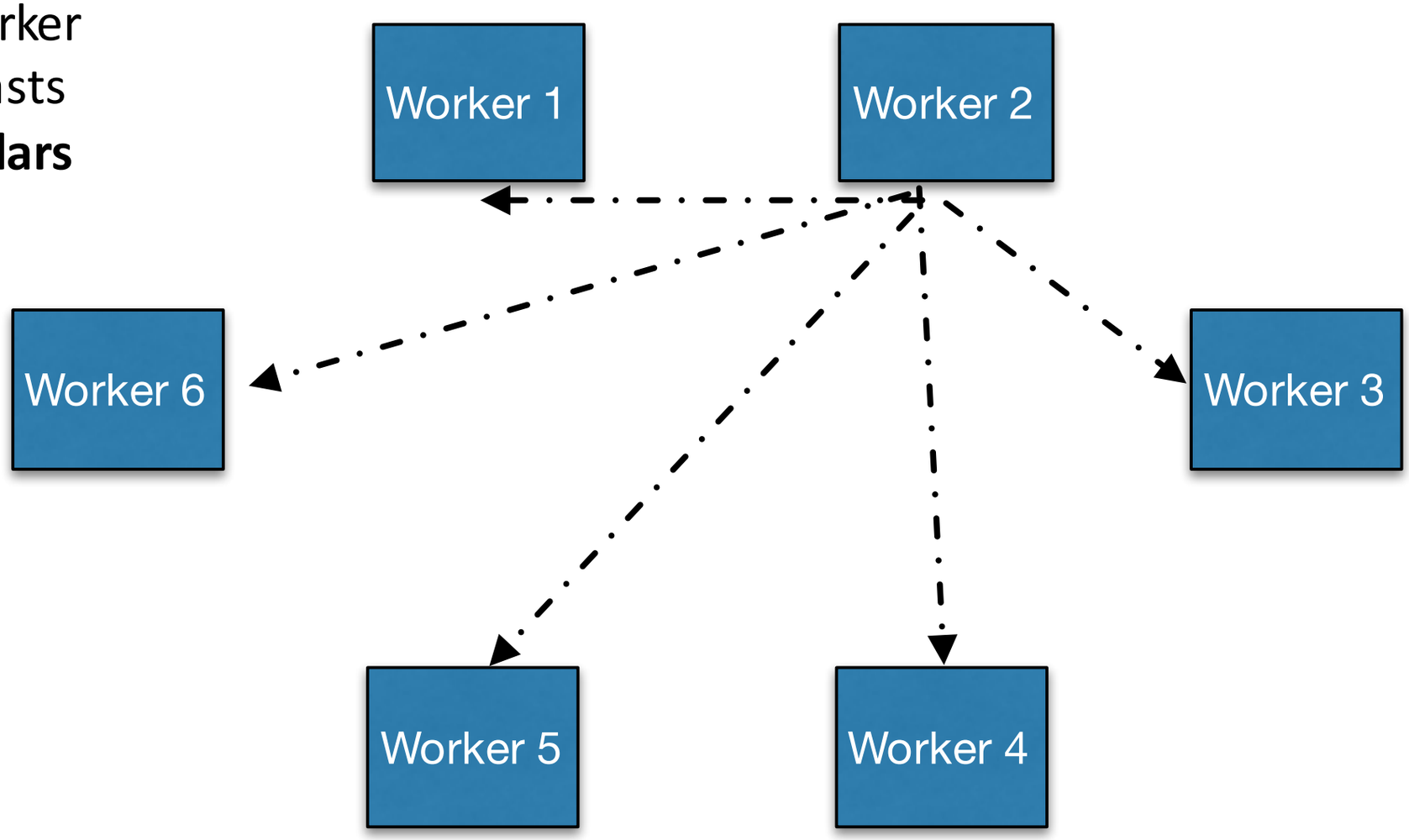
Distributed Evolution

Each worker broadcasts **tiny scalars**



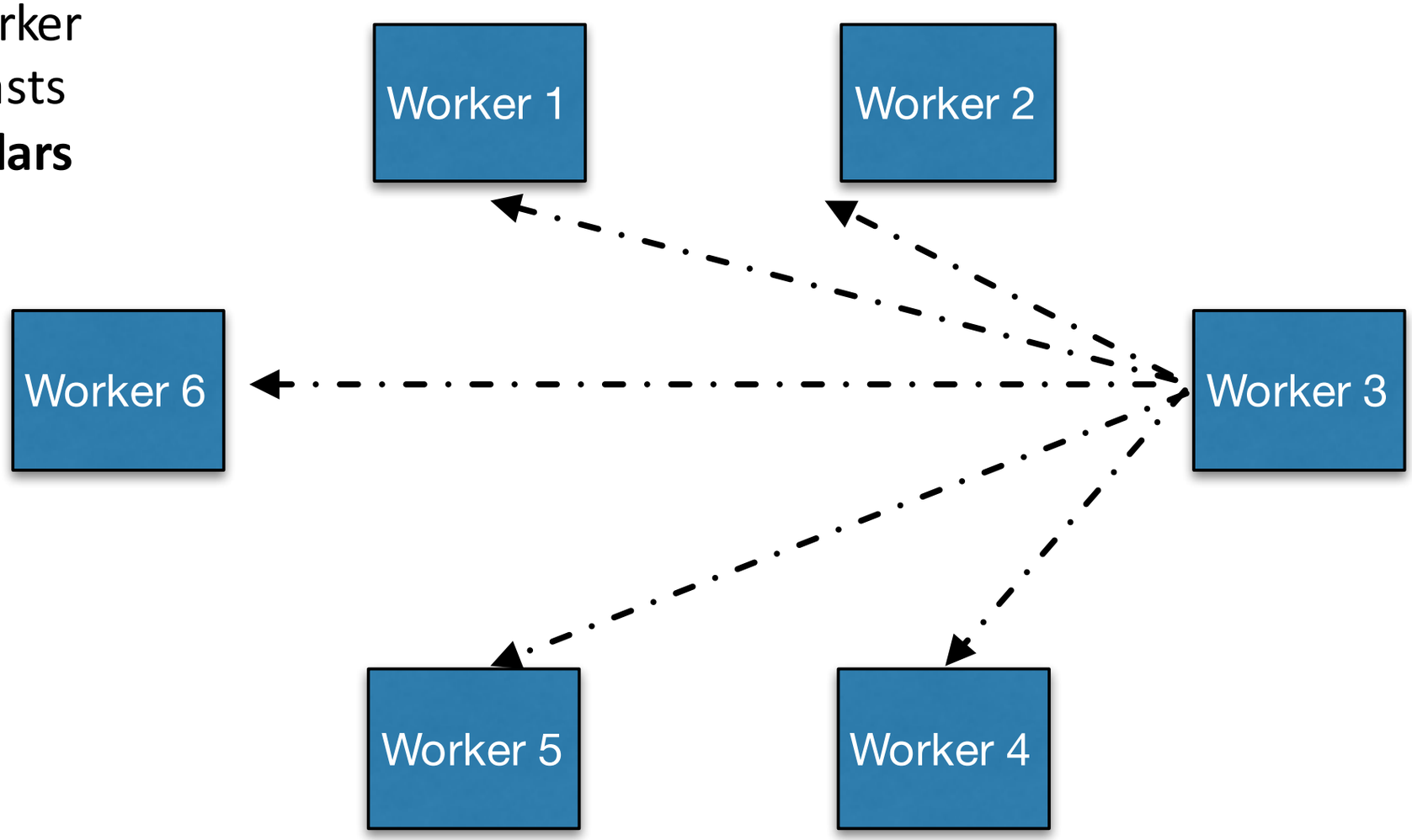
Distributed Evolution

Each worker broadcasts **tiny scalars**



Distributed Evolution

Each worker broadcasts **tiny scalars**



Distributed Evolution Scales Very Well :-)

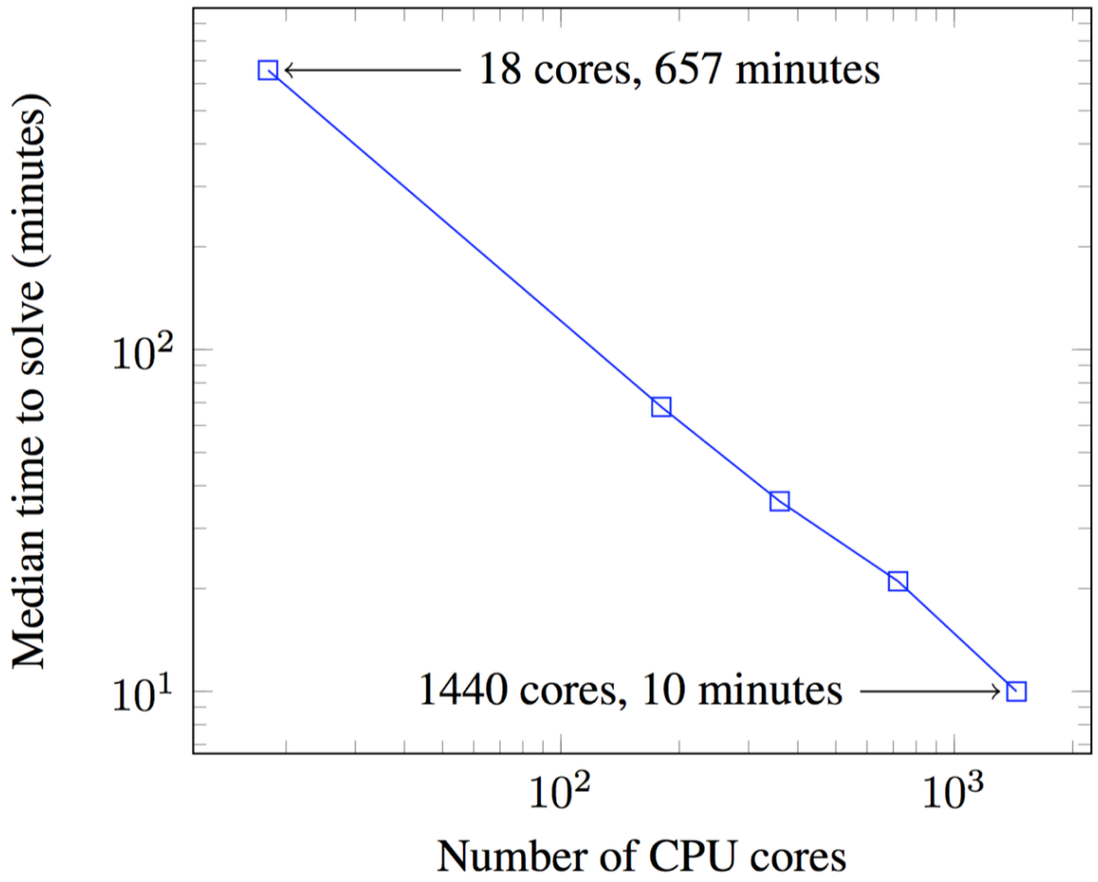


Figure 1. Time to reach a score of 6000 on 3D Humanoid with different number of CPU cores. Experiments are repeated 7 times and median time is reported.

Salimans, Ho, Chen, Sustskever. 2017

Limitations

Q: Can Evolutionary Search get stuck in local optima?
A: Yes, sure it can and it does.

Q: how can we get ES to be less stuck in local optima?

ES can get stuck in local optima

- How can we help search methods search better?

It turns out, searching for policies:

1.in multiple related tasks

2.and in multiple related environments

can help ES search methods do better. We do not have any formal guarantees, but empirically it turns out this is the case.