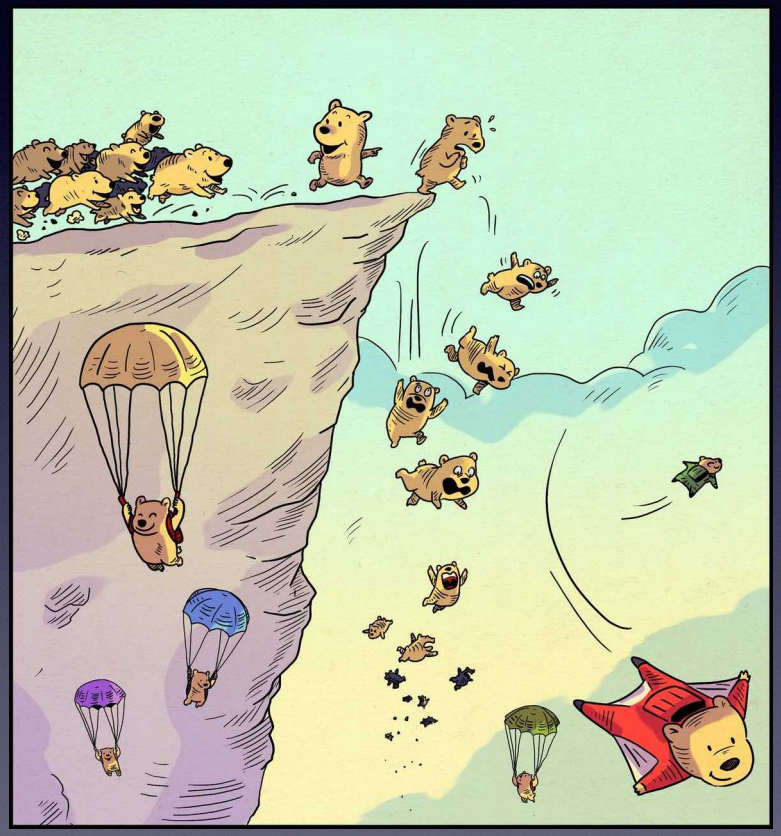
Evolution Strategies

Distributed deep reinforcement learning



(blog.otoro.net)

Steven Schmatz
@stevenschmatz

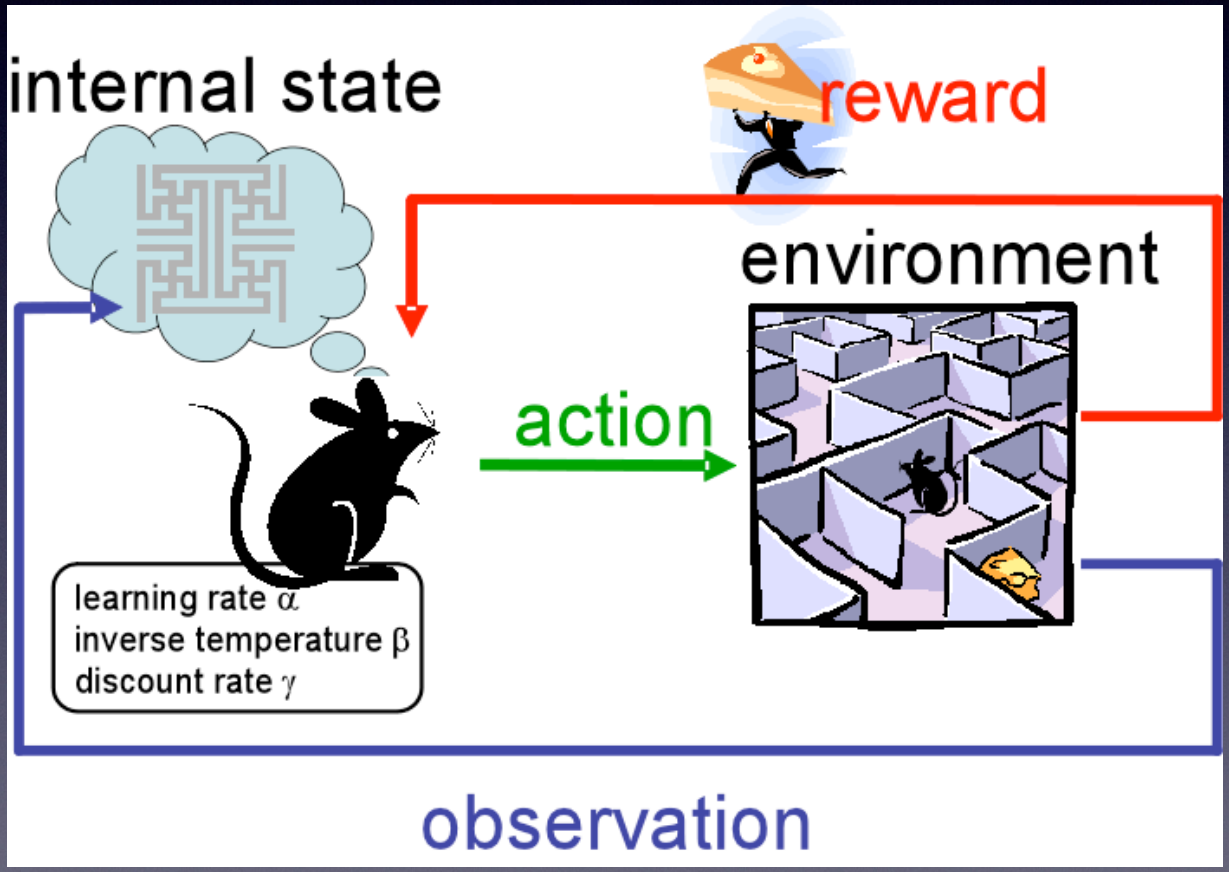
Deep Reinforcement Learning

Agenda

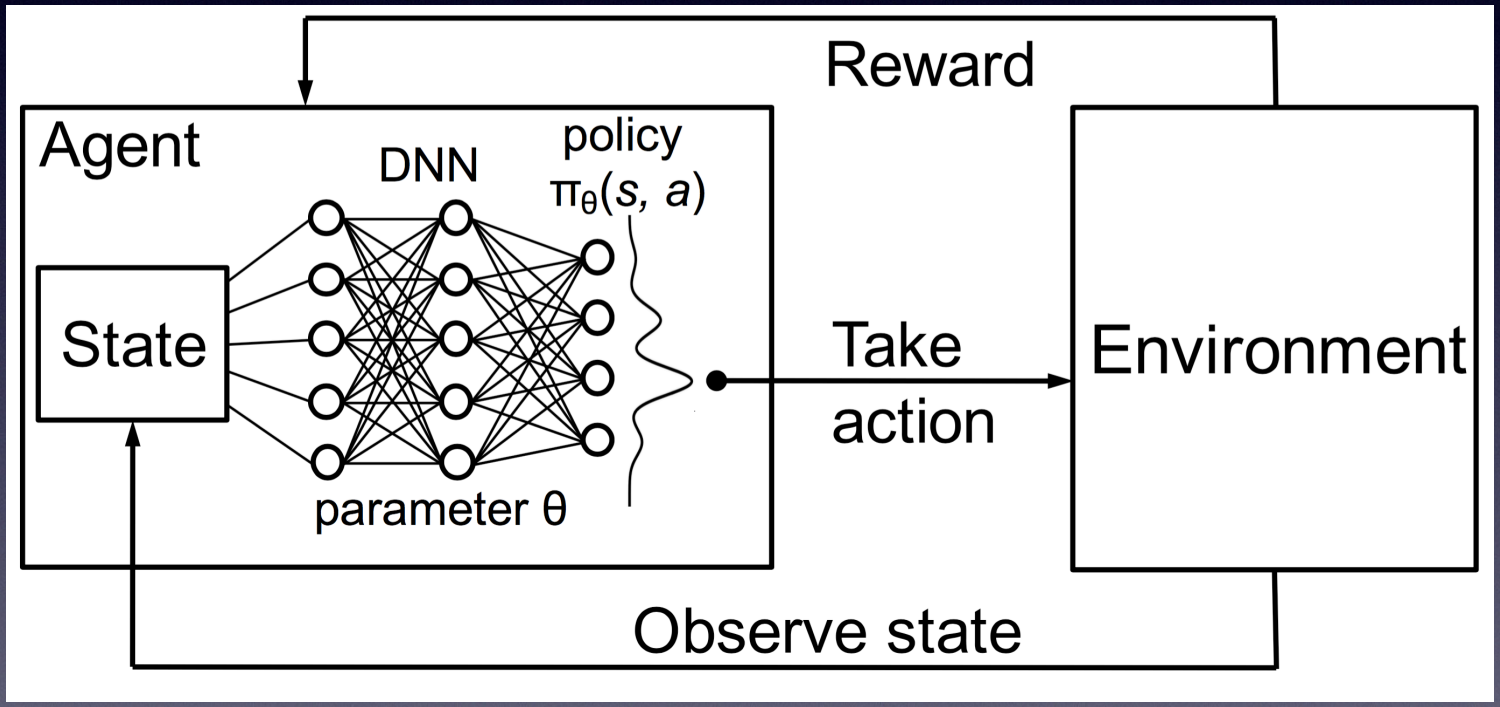
1. Why is deep reinforcement learning hard?
2. How does evolution strategies (ES) help?
3. Advice on applying ES to real-world problems

RL in a nutshell

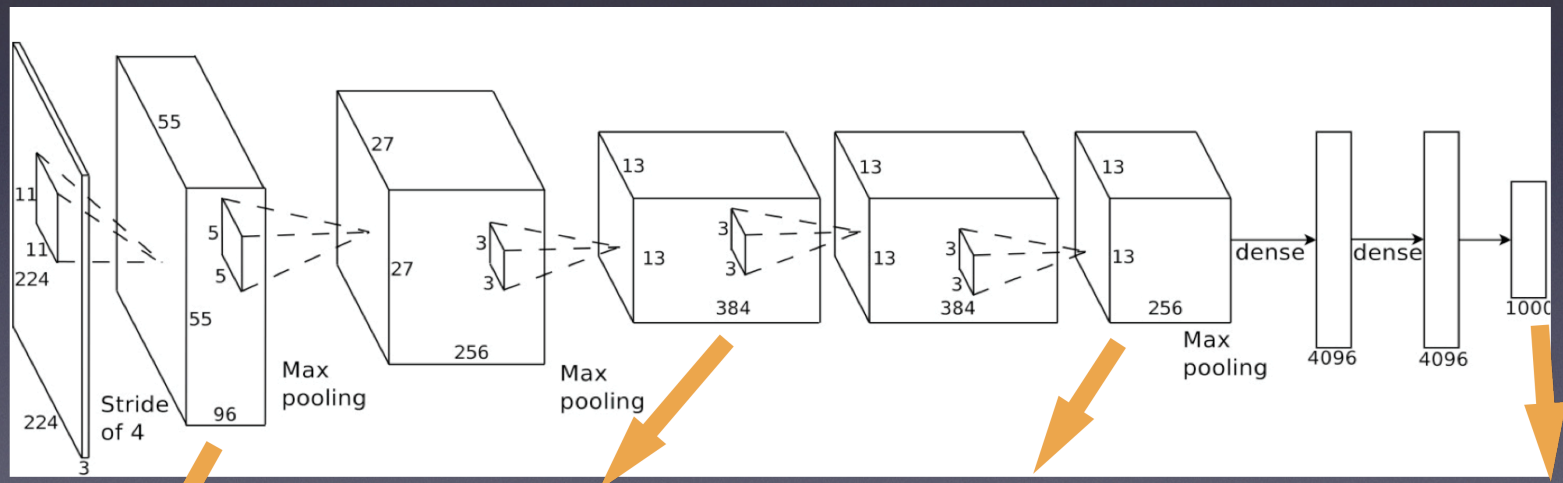
(reinforcement learning)

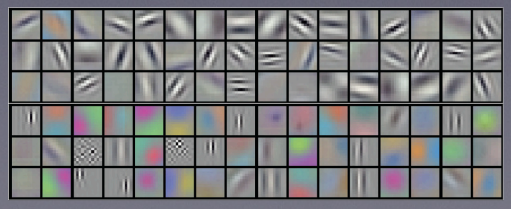


Deep RL in a nutshell

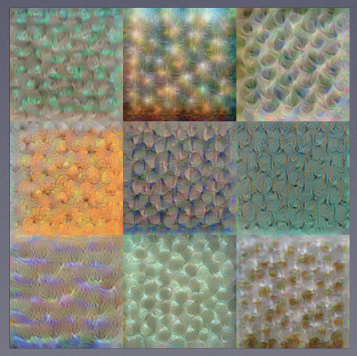


Deep CNNs are useful.





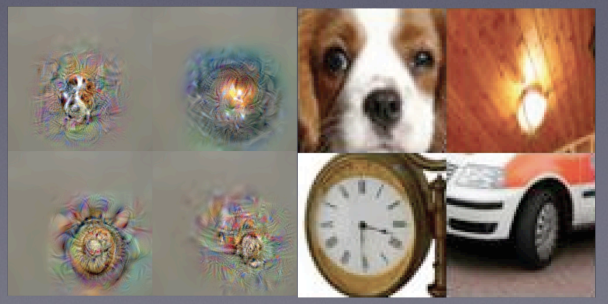
Conv 1: Edge+Blob



Conv 3: Texture

Numerical

Data-driven



Conv 5: Object Parts

cock

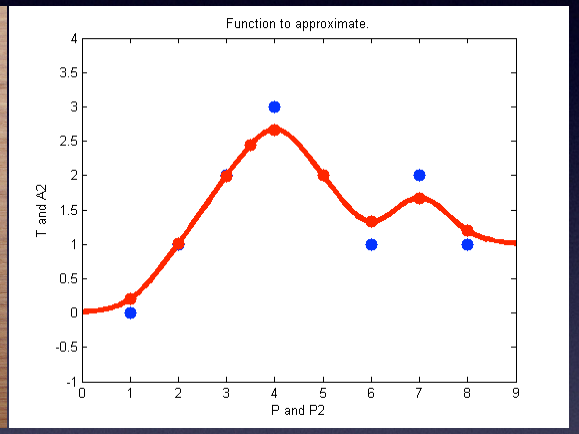
ship

Fc8: Object Classes

Assumptions of supervised learning

Stationary distribution

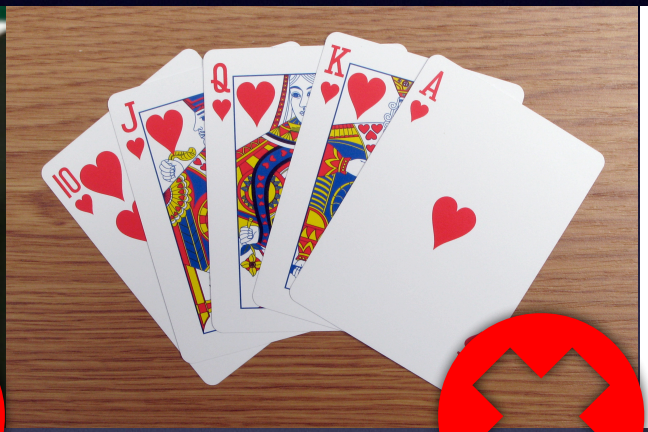
Independence of examples



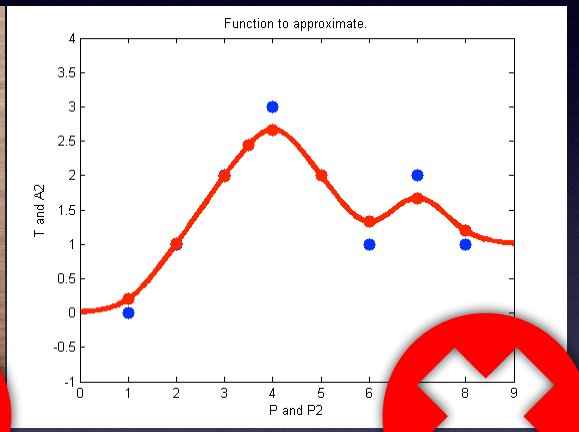
Clear input-output **relationship**

RL violates these assumptions. 😭

Stationary distribution



Independence of examples



Clear input-output **relationship**

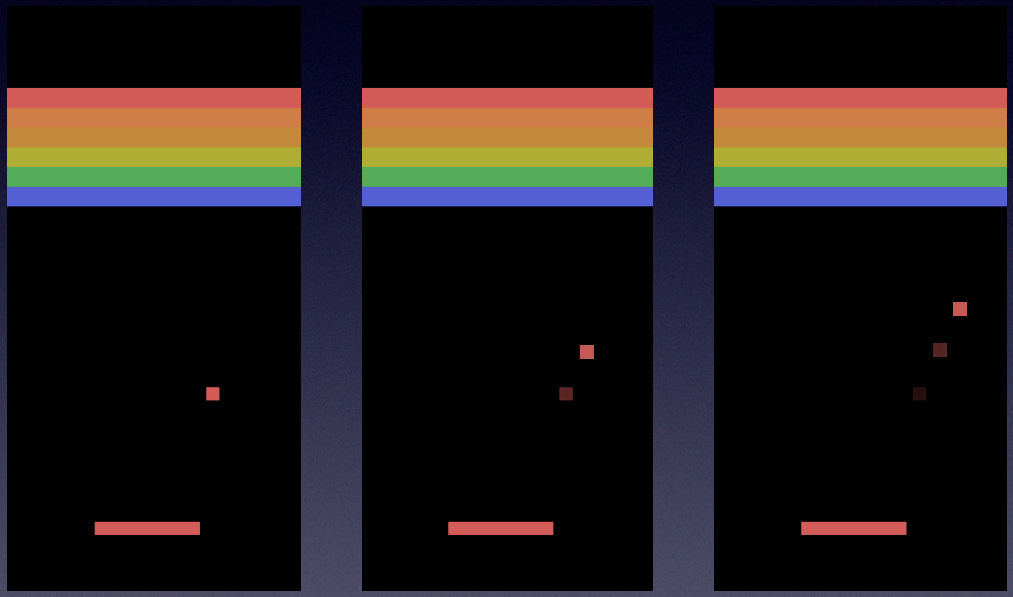
RL violates these assumptions. 😭

Stationary distribution

The training data changes as you act differently.

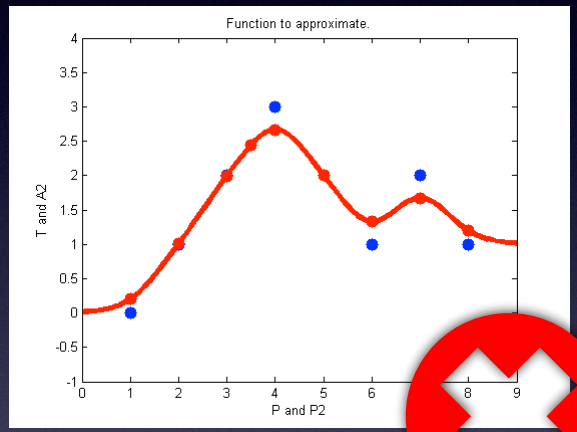
RL violates these assumptions. 😭

Independence of examples



Adjacent game frames are usually very similar.

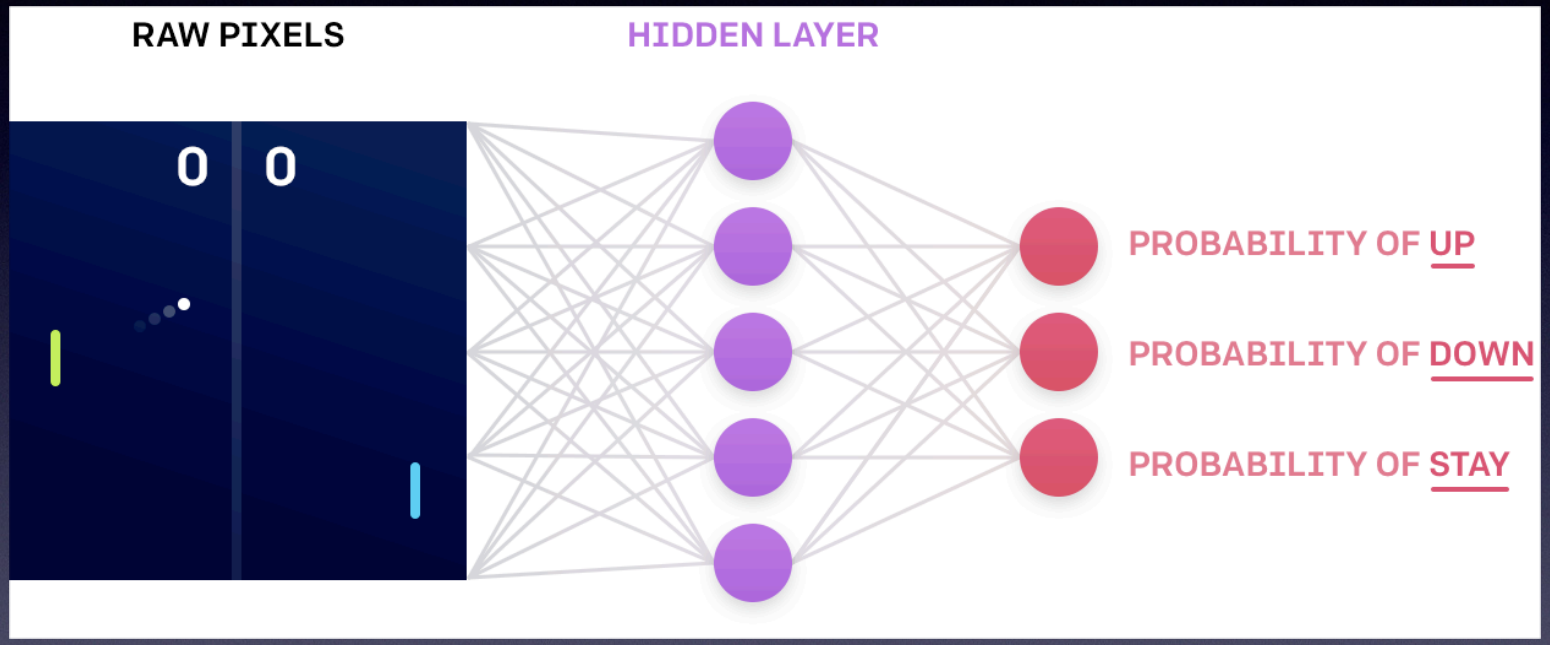
RL violates these assumptions. 😭



Clear input-output **relationship**

There can be a large delay between action and reward.

Deep Q-Learning



Model

$$f : \mathbb{R}^{84\times 84} \xrightarrow{\theta} \mathbb{R}$$

Training objective

$$\max_{\theta} \mathbb{E}\left[\sum_{t=0}^{H} R(s_t)|\theta\right]$$

Policy gradients

$$\max_{\theta} \mathbb{E}\left[\sum_{t=0}^{H} R(s_t)|\theta\right]$$

(our objective)

$$\Delta\theta_t = \alpha R_t \frac{\partial \pi(a_t|\theta_t)}{\partial \theta_t}$$

(our weight update)

Steven Schmatz
@stevenschmatz

Policy gradients

$$\Delta\theta_t = \alpha R_t \frac{\partial \pi(a_t|\theta_t)}{\partial \theta_t}$$

What if reward is noisy?

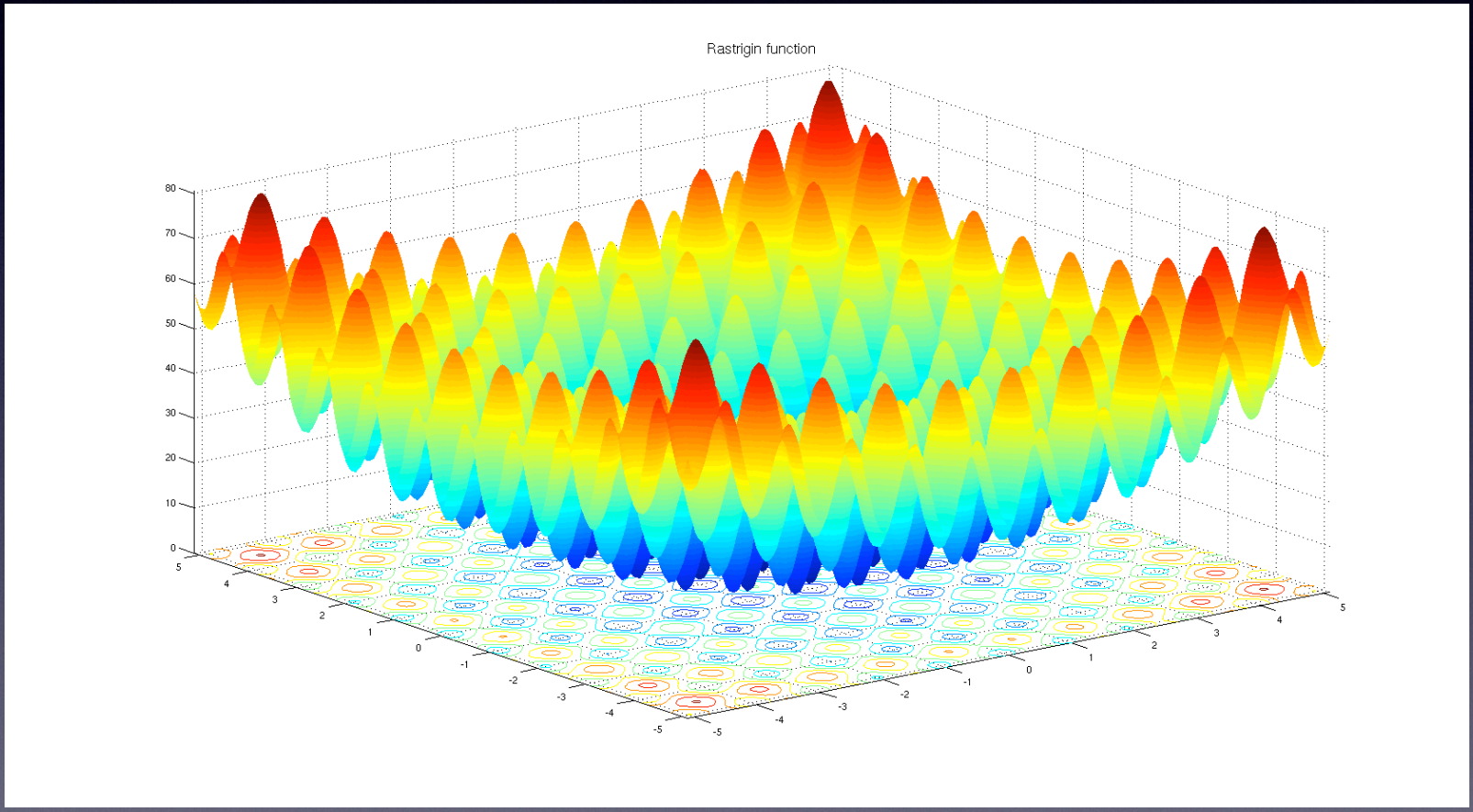
What if reward is non-differentiable?

What if reward is very far away?

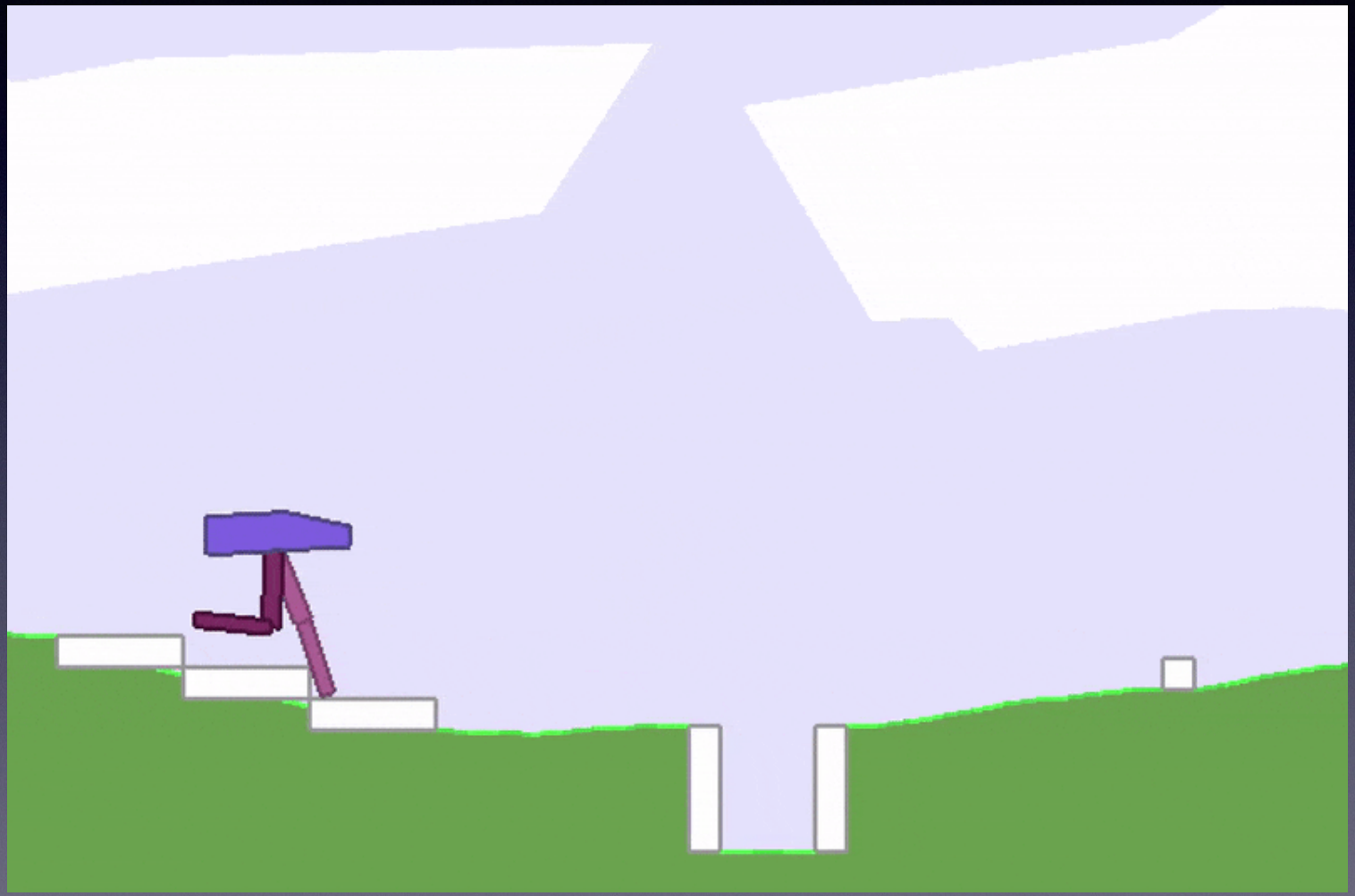
How far should we step?

Steven Schmatz
@stevenschmatz

Local optima



Local optima



Black-box optimization

$$f : \mathbb{R}^{84\times 84} \xrightarrow{\theta} \mathbb{R}$$

$$\theta \rightarrow$$  $$\rightarrow \text{total reward}$$

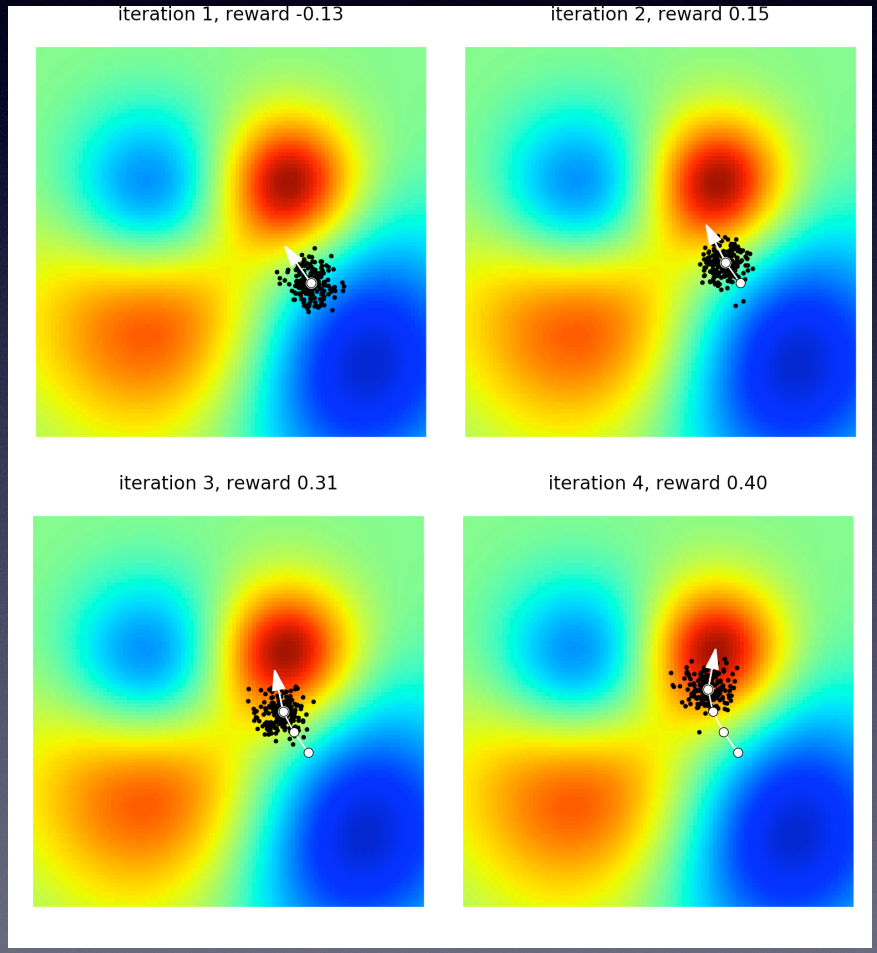
$$e : \theta \rightarrow \mathbb{R}$$

Steven Schmatz
@stevenschmatz

ES to the rescue!

At each iteration:

1. **Generate** candidate solutions from old candidates by adding noise
2. **Evaluate** a *fitness function* for each candidate
3. **Aggregate** the results and discard bad candidates.

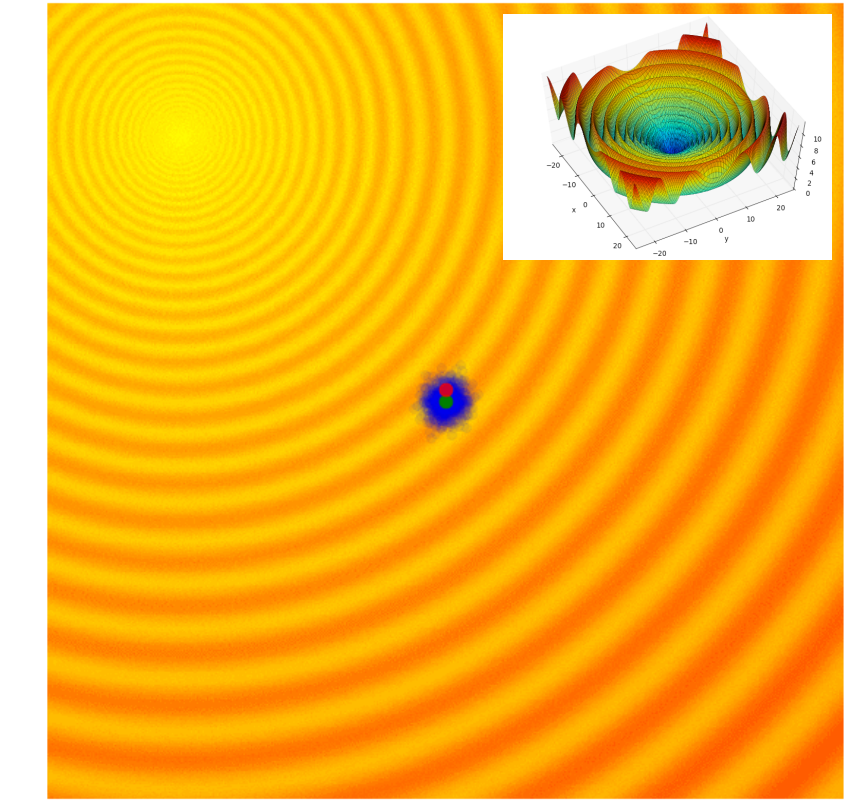


Simple ES

Basic idea:

Select the **single best** previous solution, and add Gaussian noise.

(Keep standard deviation fixed.)



Steven Schmatz
@stevenschmatz

Genetic ES

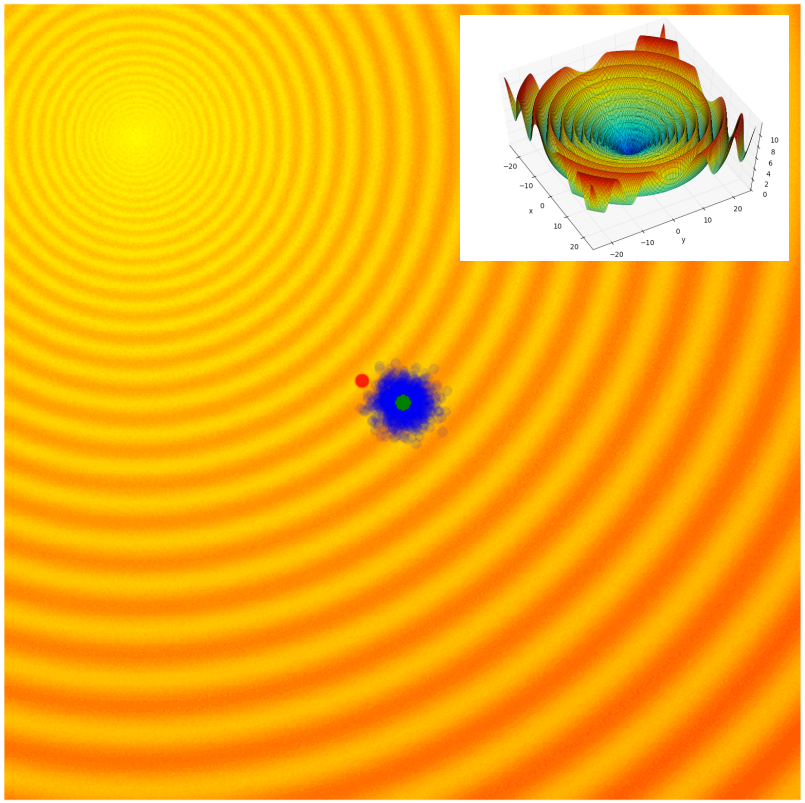
Basic idea:

Only keep the **top performing 10%** of solutions.

Randomly select two solutions.

Recombine them by randomly assigning each parameter value from either parent.

(and add fixed Gaussian noise.)



Example:

Combine (1, 2, 3) and (**4, 5, 6**):

- (1, **5, 6**)
- (**4**, 2**,** 3)

CMA–ES

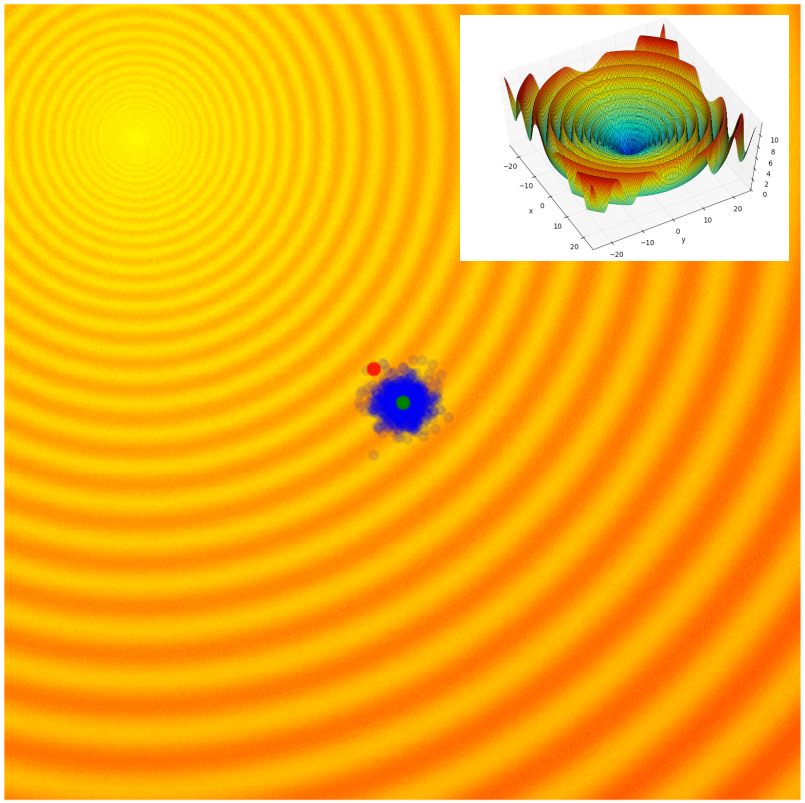
Basic idea:

Select the best 25% of the population.

Calculate a **covariance matrix** of these best 25%.

(represents a promising area to search for new candidates)

Generate new candidates using the per-parameter means and variances.



CMA–ES

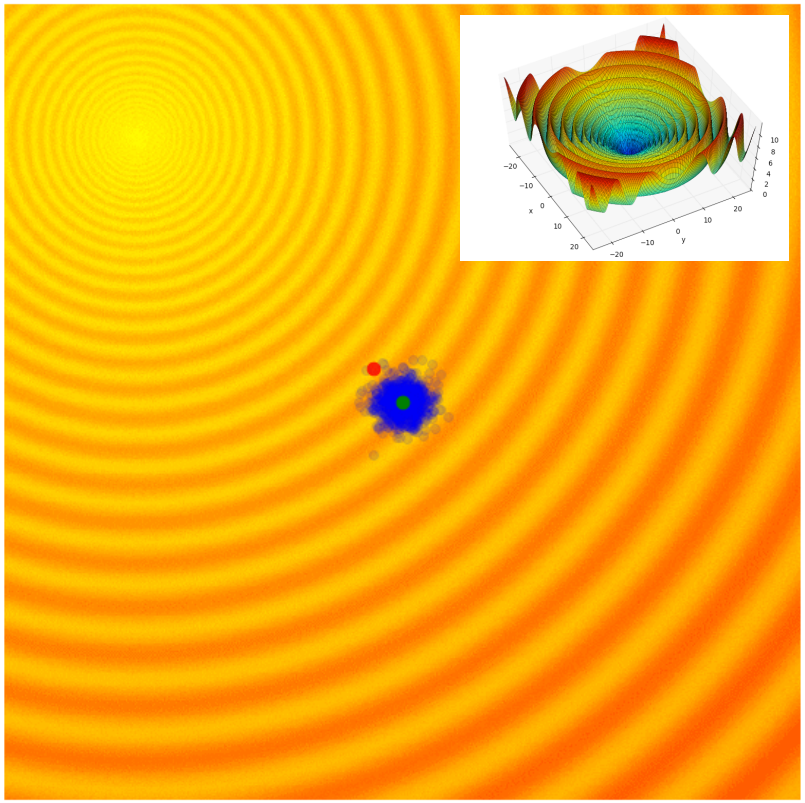
Basic idea:

Select the best 25% of the population.

Calculate a **covariance matrix** of these best 25%.

(represents a promising area to search for new candidates)

Generate new candidates using the per-parameter means and variances.



Problem: $O(n^2)$ 😭

Natural ES

Basic idea:

Treat the problem a bit differently:

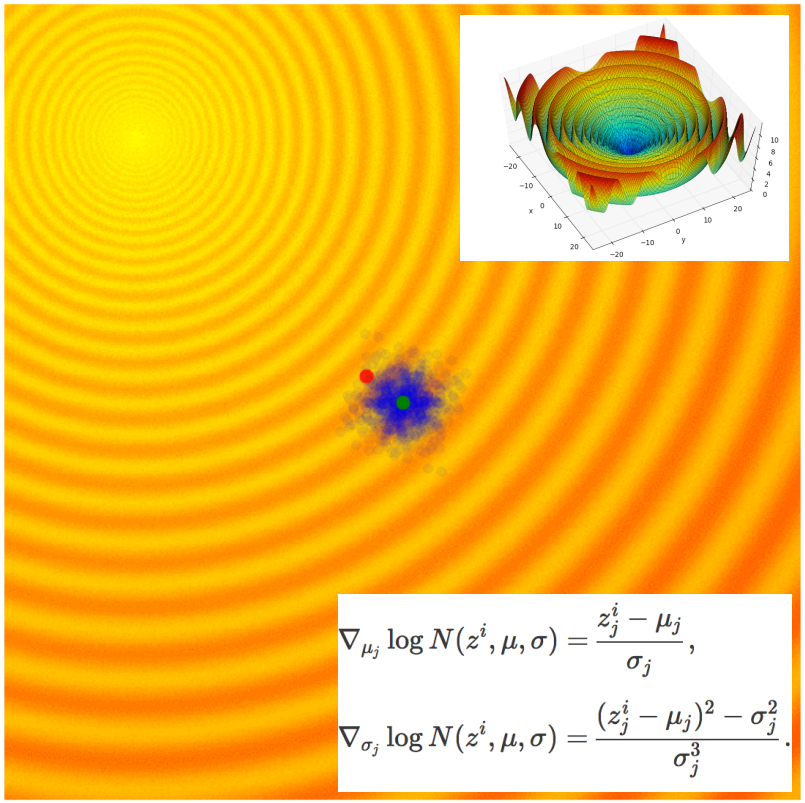
$$J(\theta) = \mathbb{E}_\theta \left[\textsc{Fitness}(z) \right]$$

$$\nabla_\theta J(\theta) = \mathbb{E}_\theta \left[\textsc{Fitness}(z) \nabla \log \pi(z^{(i)}, \theta) \right]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \textsc{Fitness}(z^{(i)}) \underbrace{\nabla_\theta \log \pi(z^{(i)}, \theta)}_{\text{generate prob. } z^{(i)}}$$

Then use the gradient with your favorite SGD optimizer:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$



Steven Schmatz
@stevenschmatz

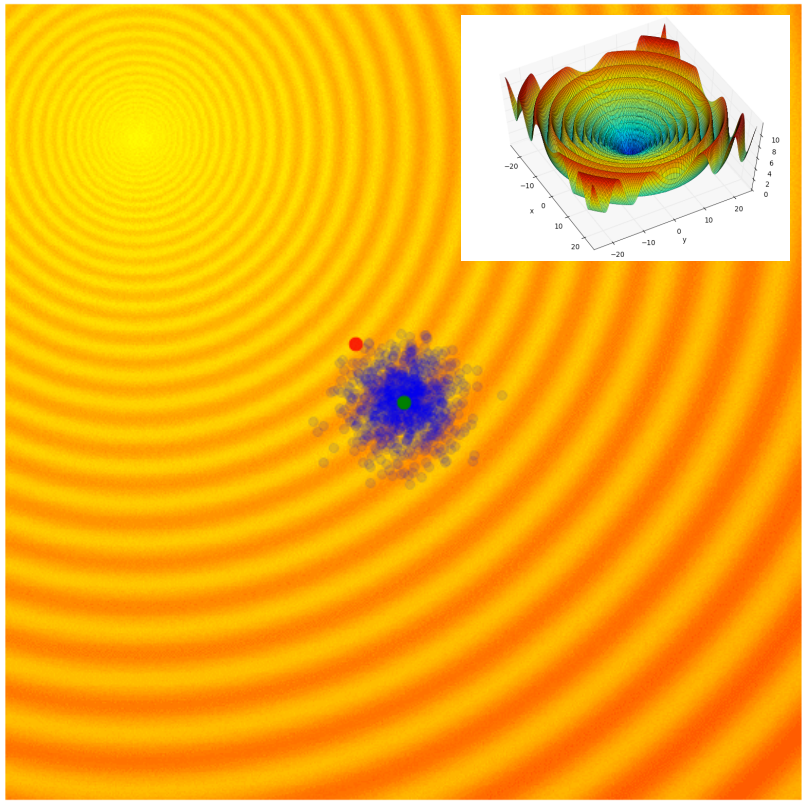
OpenAI ES

Basic idea:

Similar to Natural ES, but σ constant.

$$J(\theta) = \mathbb{E}_\theta \left[\textsc{Fitness}(z)\right]$$
$$= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)} \left[F(\theta + \sigma\epsilon)\right]$$
$$\nabla_\theta J(\theta) = \frac{1}{\sigma}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)} \left[F(\theta + \sigma\epsilon)\epsilon\right]$$

for $t = 0, 1, 2, \ldots$ **do**
  Sample $\epsilon_1, \ldots \epsilon_n \sim \mathcal{N}(0, I)$
  Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$ for $i = 1, \ldots, n$
  Set $\theta_{t+1} \leftarrow \theta_t + \alpha\frac{1}{n\sigma}\sum_{i=1}^{n} F_i\epsilon_i$
end for



Steven Schmatz
@stevenschmatz

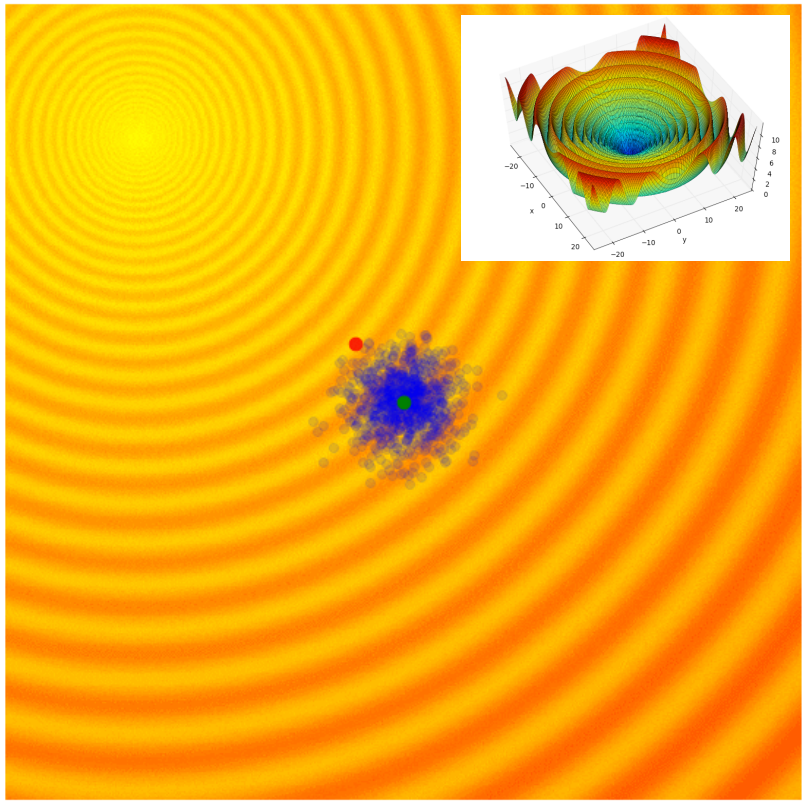
OpenAI ES

Basic idea:

Similar to Natural ES, but σ constant.

$$J(\theta) = \mathbb{E}_\theta \left[\textsc{Fitness}(z)\right]$$
$$= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)} \left[F(\theta + \sigma\epsilon)\right]$$
$$\nabla_\theta J(\theta) = \frac{1}{\sigma}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)} \left[F(\theta + \sigma\epsilon)\epsilon\right]$$

for $t = 0, 1, 2, \ldots$ **do**
  Sample $\epsilon_1, \ldots \epsilon_n \sim \mathcal{N}(0, I)$
  Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$ for $i = 1, \ldots, n$
  Set $\theta_{t+1} \leftarrow \theta_t + \alpha\frac{1}{n\sigma}\sum_{i=1}^{n} F_i\epsilon_i$
end for

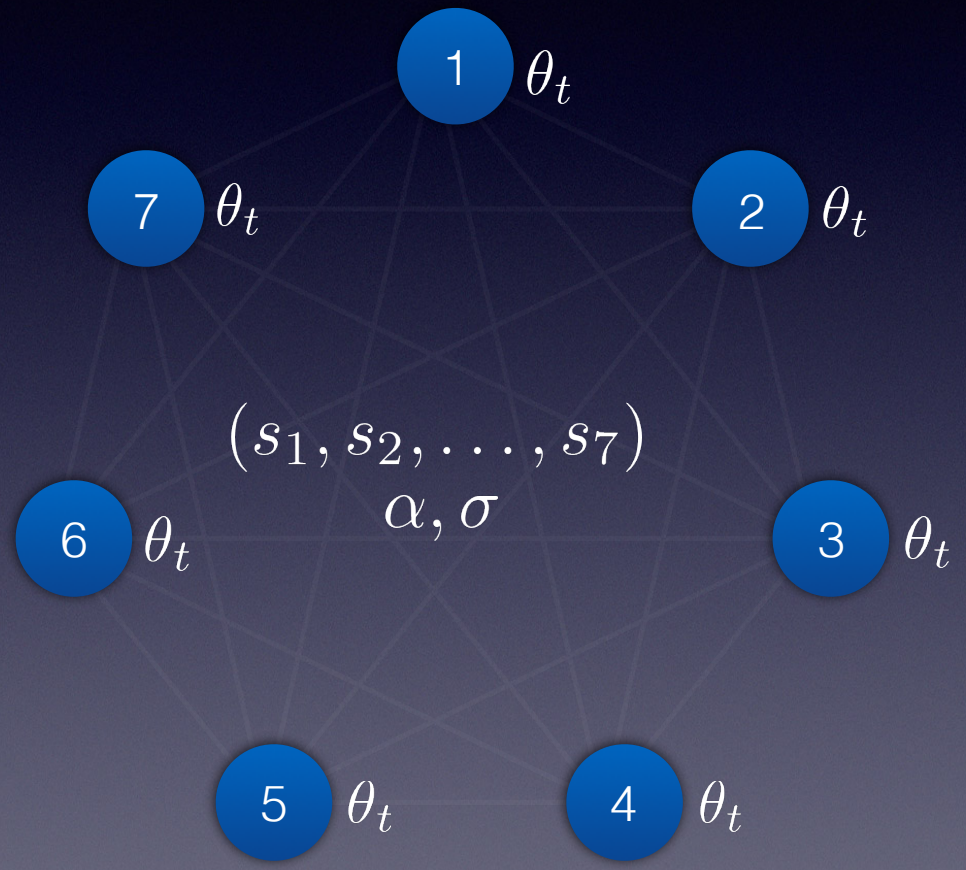


Note: to parallelize we only need to know (F_i, ϵ_i) pairs!

Parallelization

Initialize

1. Create shared list of all random seeds, one per worker; and θ_t



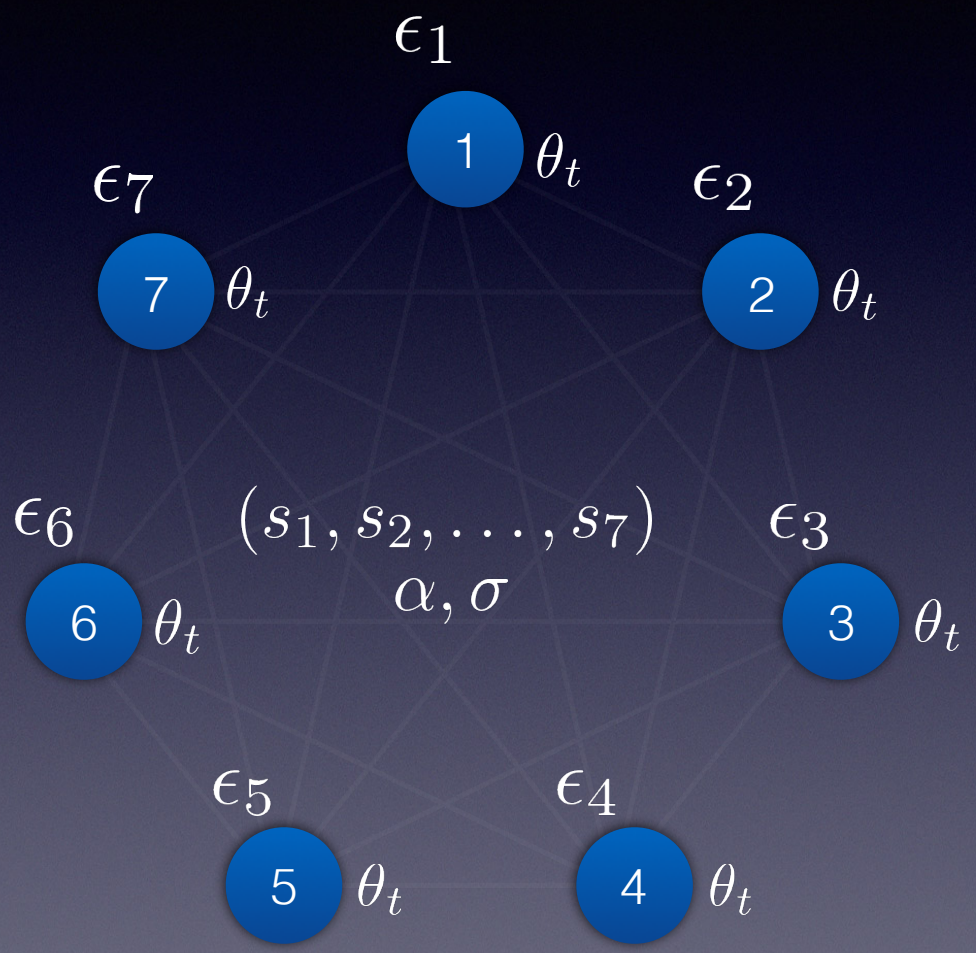
Parallelization

Initialize

1. Create shared list of all random seeds, one per worker; and θ_t

Repeat:

1. Sample $\epsilon_i \sim \mathcal{N}(0, 1|s_i)$



Steven Schmatz
@stevenschmatz

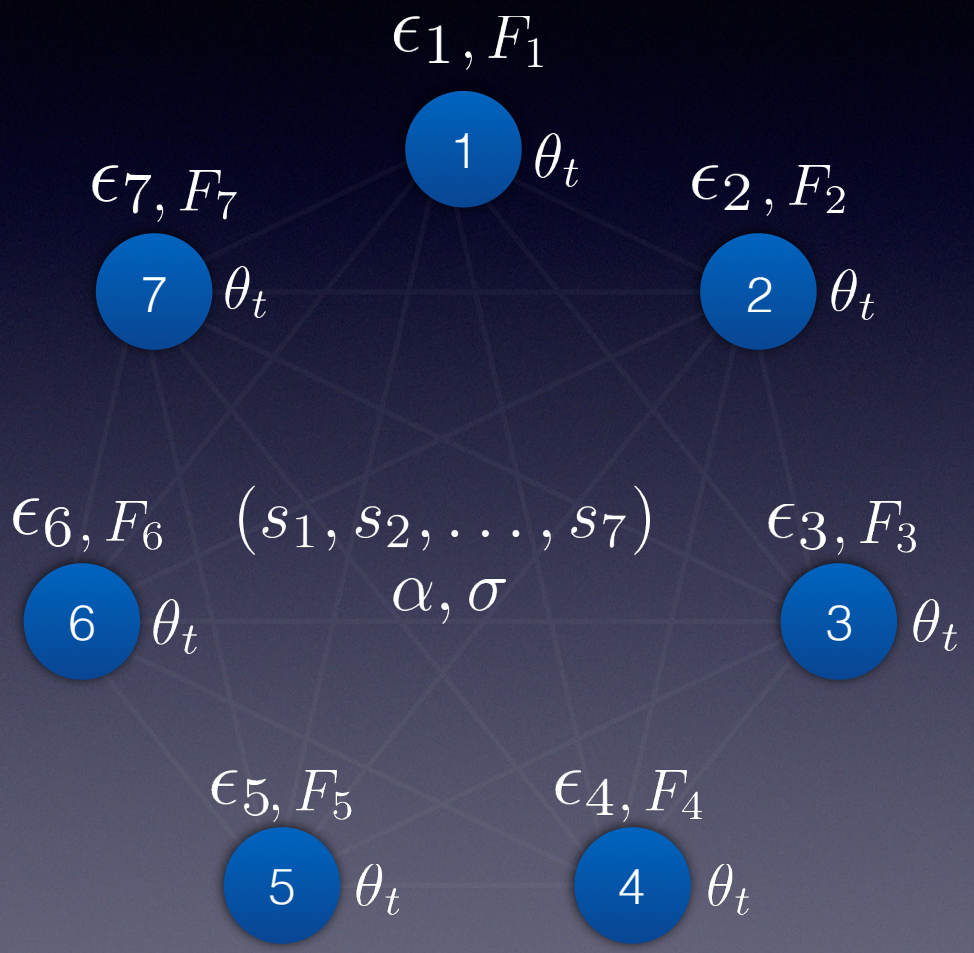
Parallelization

Initialize

1. Create shared list of all random seeds, one per worker; and θ_t

Repeat:

1. Sample $\epsilon_i \sim \mathcal{N}(0, 1|s_i)$
2. Evaluate $F(\theta_t + \sigma\epsilon_i)$



Steven Schmatz
@stevenschmatz

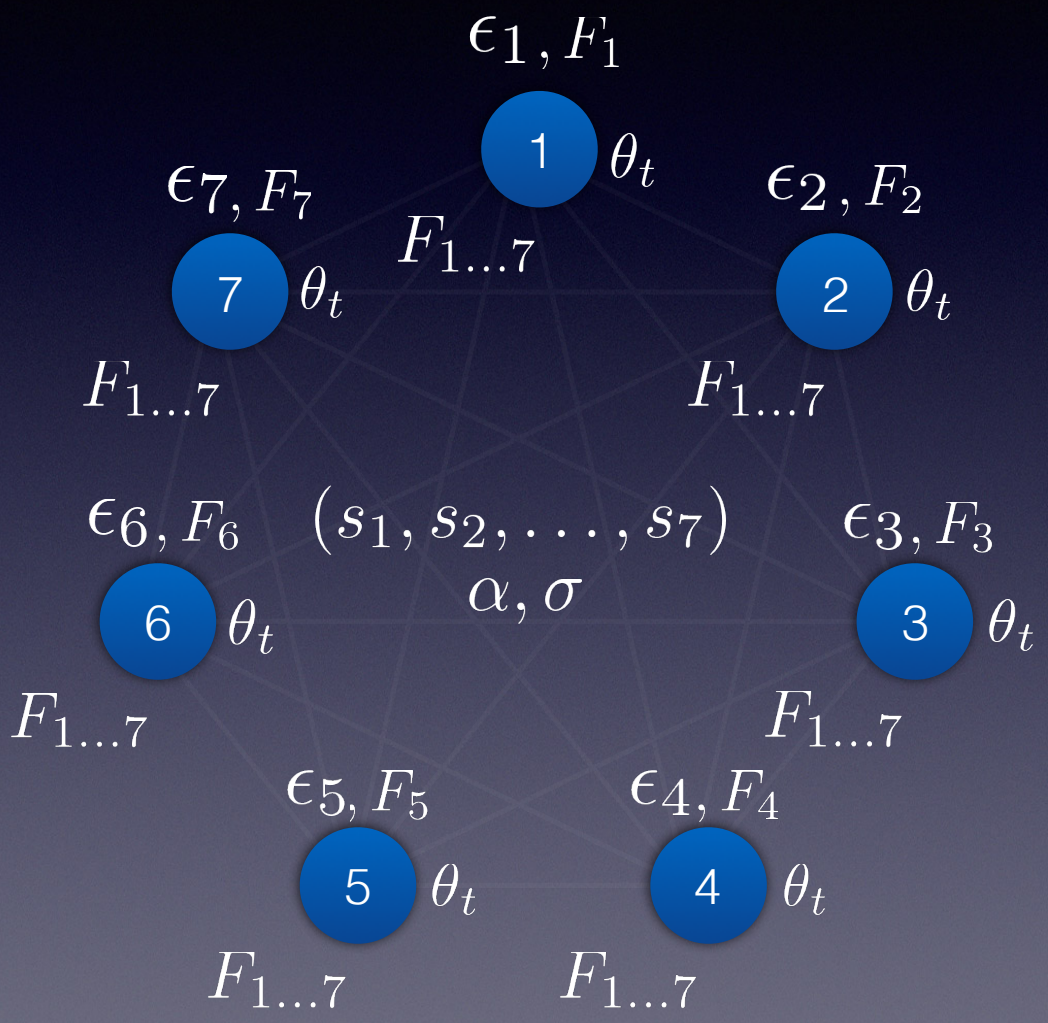
Parallelization

Initialize

1. Create shared list of all random seeds, one per worker; and θ_t

Repeat:

1. Sample $\epsilon_i \sim \mathcal{N}(0, 1|s_i)$
2. Evaluate $F(\theta_t + \sigma\epsilon_i)$
3. Communicate F to all nodes



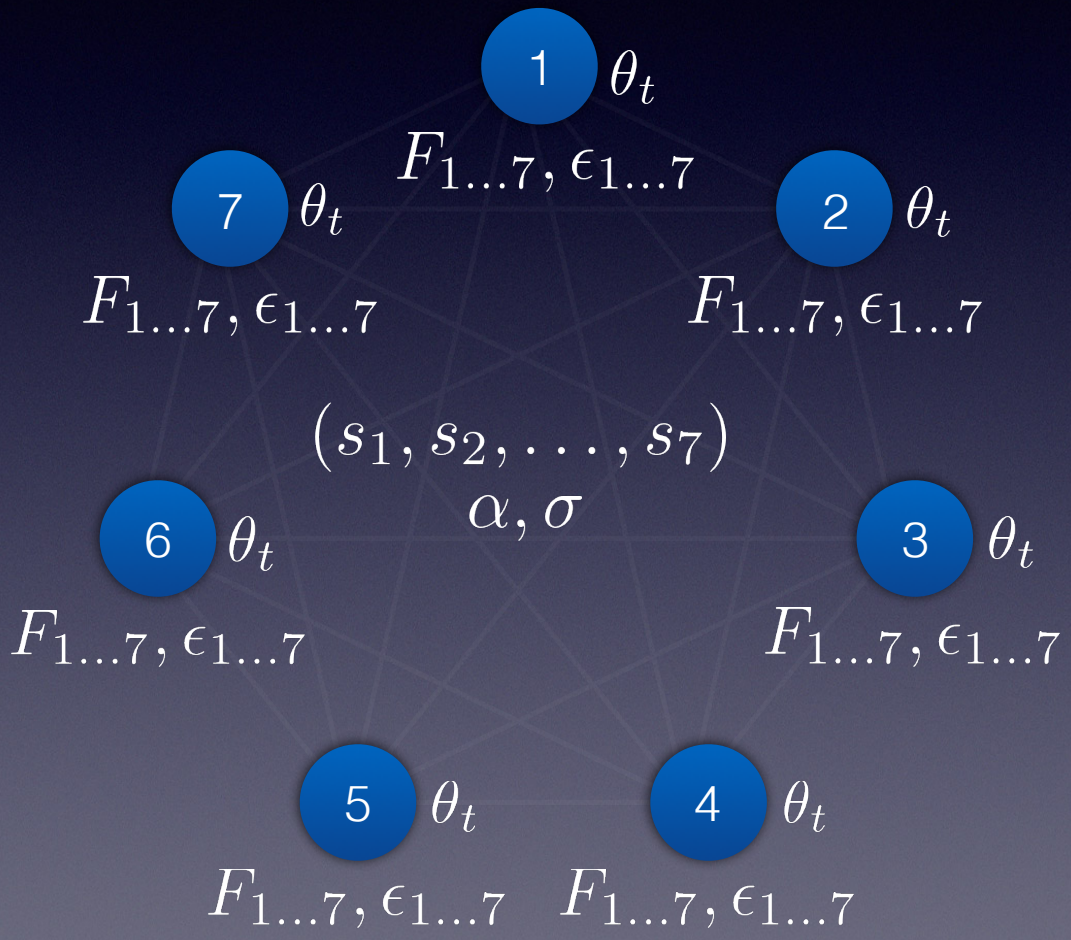
Parallelization

Initialize

1. Create shared list of all random seeds, one per worker; and θ_t

Repeat:

1. Sample $\epsilon_i \sim \mathcal{N}(0, 1|s_i)$
2. Evaluate $F(\theta_t + \sigma\epsilon_i)$
3. Communicate F to all nodes
4. **Reconstruct** ϵ_i for all other nodes using known random seeds



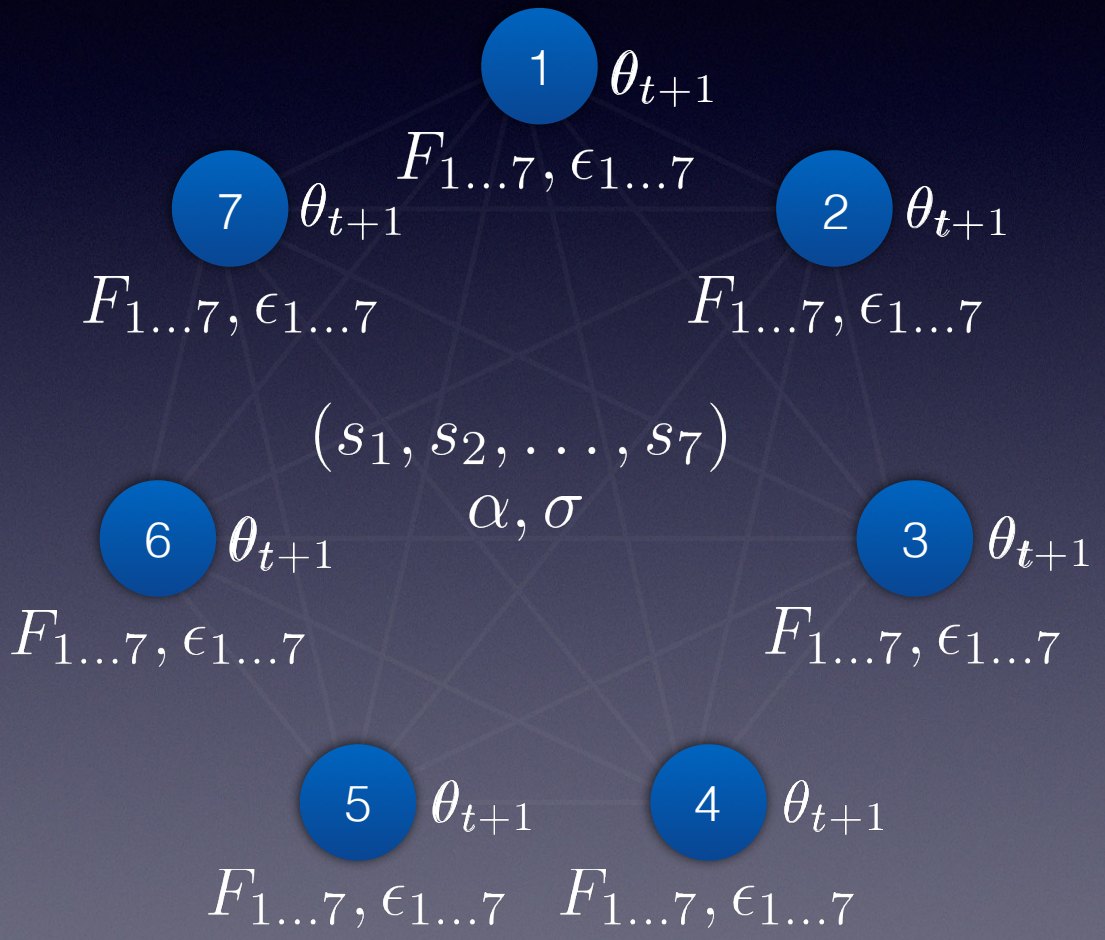
Parallelization

Initialize

1. Create shared list of all random seeds, one per worker; and θ_t

Repeat:

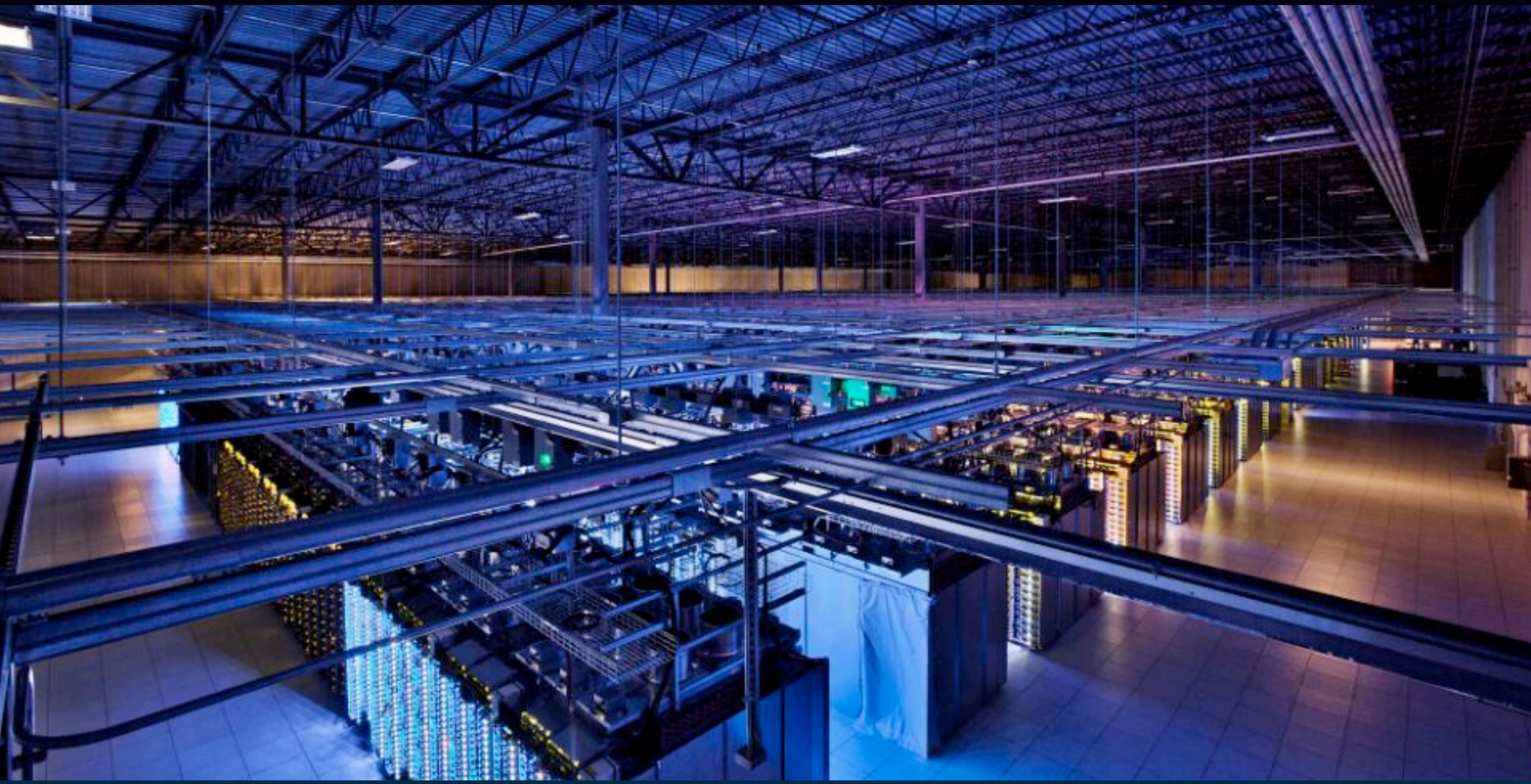
1. Sample $\epsilon_i \sim \mathcal{N}(0, 1|s_i)$
2. Evaluate $F(\theta_t + \sigma\epsilon_i)$
3. Communicate F to all nodes
4. **Reconstruct** ϵ_i for all other nodes using known random seeds
5. $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{j=1}^{n} F_j \epsilon_j$



Efficiency

- The only information communicated at each iteration is a **single scalar** per machine.
- Most distributed update mechanisms (A3C, Gorila) must communicate entire parameter lists.
- Result: **linear horizontal parallelization**.

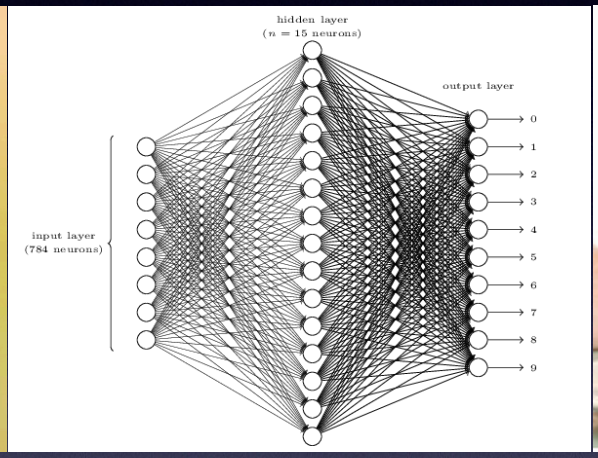
Efficiency



Benefits

Non-differentiable policies!

(hard attention!)



No backprop!

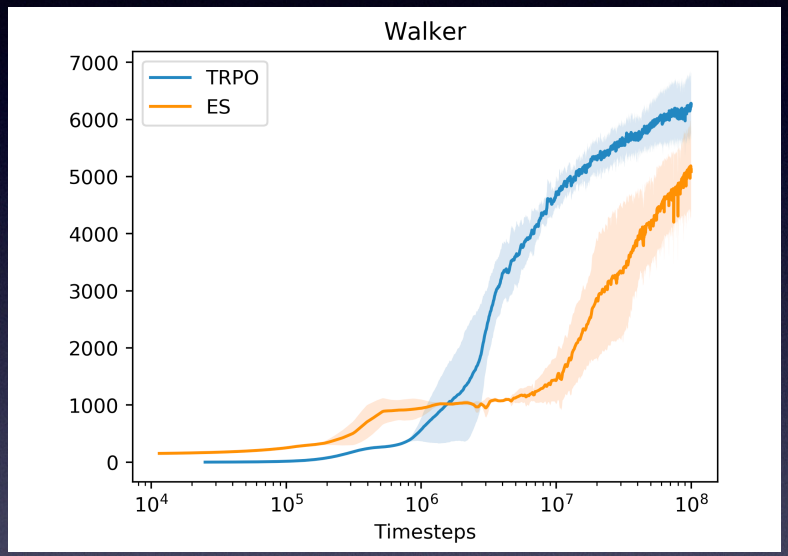
3x computation time decrease!

Sparse rewards!

Learn long-term policies in hard environments!

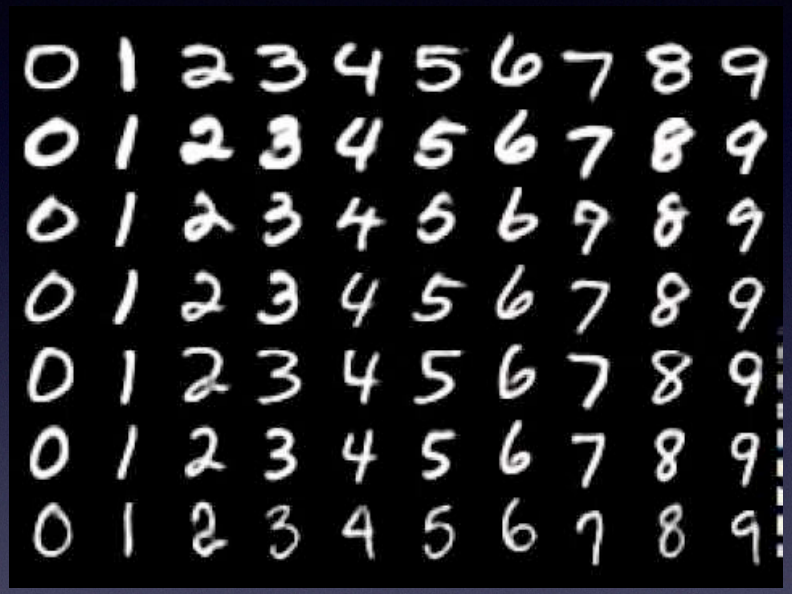
And **much** cheaper than GPUs!

Drawbacks



Data inefficient

About 3–10x less data efficient



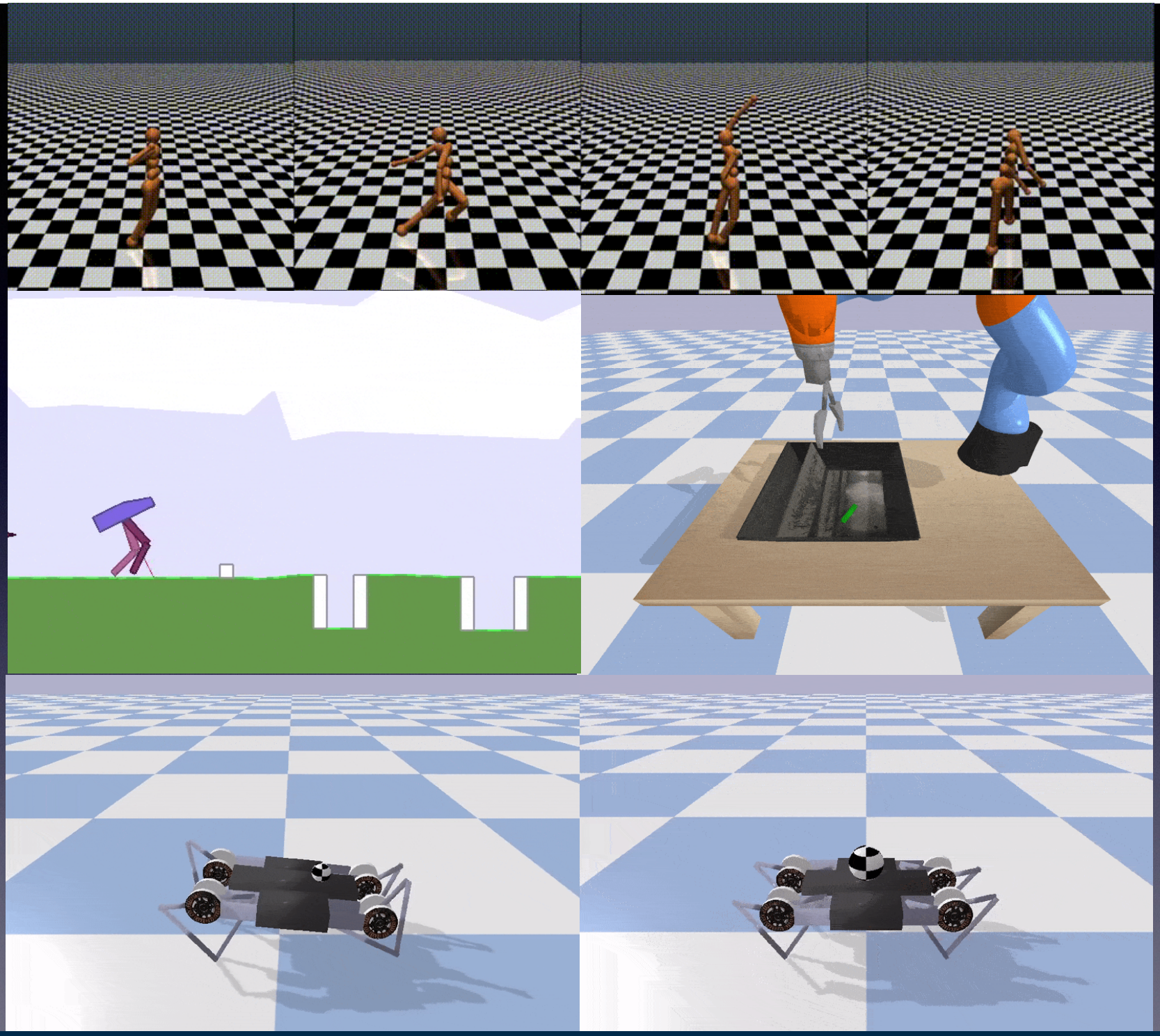
Not useful for **supervised learning**.

(good, reliable gradients)

Bottom Line

If you have a **large amount of CPU cores** (>100), or if you have **sparse rewards**, evolution strategies may be a good bet.

Steven Schmatz
@stevenschmatz



Appendix

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

$$\nabla_\theta J(\theta) = \int_{\mathbb{T}} \nabla_\theta p_\theta(\tau) r(\tau) d\tau = \int_{\mathbb{T}} p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) r(\tau) d\tau = E\left\{ \nabla_\theta \log p_\theta(\tau) r(\tau) \right\}.$$

Steven Schmatz
@stevenschmatz