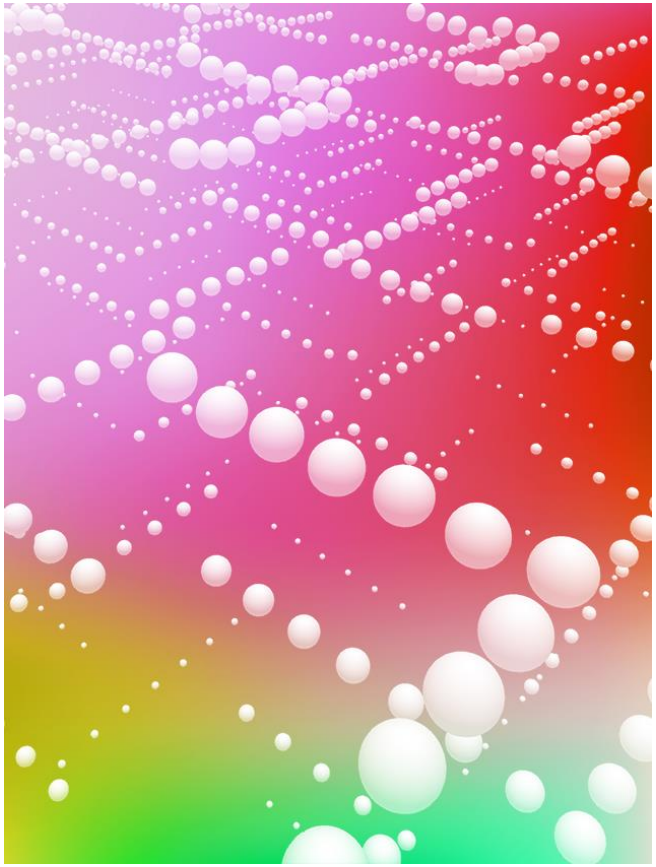
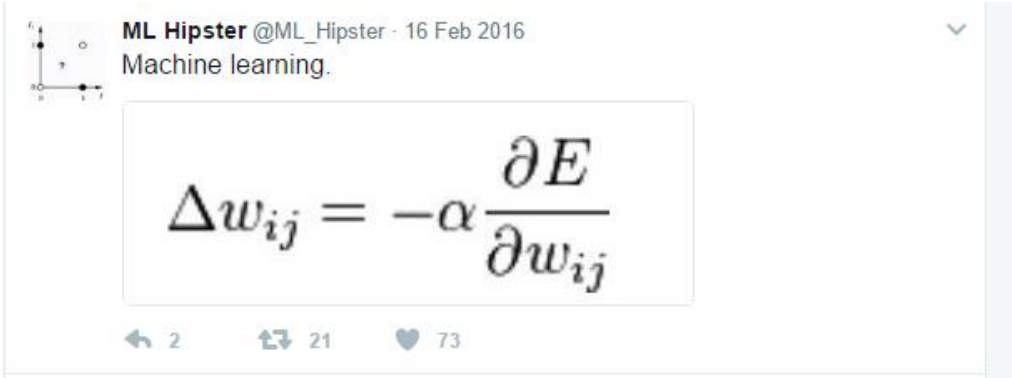
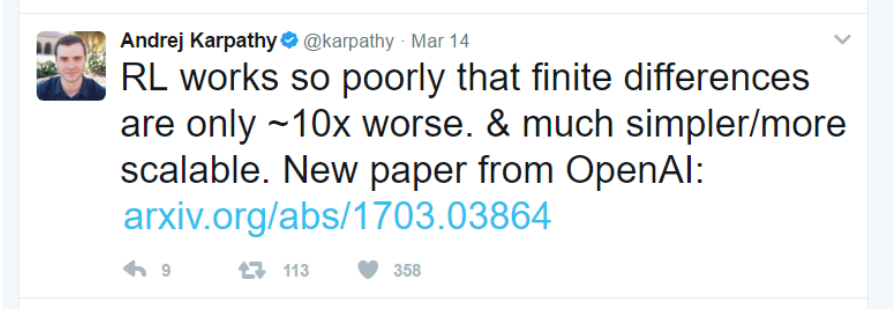
Evolution Strategies for RL



https://blog.openai.com/evolution-strategies/





Policy Gradient Learning

- Idea: policy π_θ gives us average value $J_{avV}(\theta)$
- Approximate $\nabla_\theta J_{avV}(\theta)$ by following policy π_θ, keeping track of the rewards, and computing a weighted sum of $\nabla \pi_\theta(a|s)$, and perform gradient ascent
  - $\theta \leftarrow \theta + \alpha \nabla_\theta J_{avV}(\theta)$

“Evolutionary” strategy

- Perturb θ n times by adding Gaussian noise to it (“produce n mutations”) to obtain $\theta_1, \theta_2, \dots, \theta_n$
- Follow $\pi_{\theta_1}, \dots, \pi_{\theta_n}$, and keep track of the rewards
- Obtain a new θ that’s a combination of $\theta_1, \theta_2, \dots, \theta_n$, weighted by the rewards (“evolve θ”)
- Repeat

- Easy to parallelize (follow different policies on different CPUs)
- Similar to computing the gradient using finite differences
  - Vary θ, try it, adjust θ according to how this went

“Evolutionary” strategy

Algorithm 1 Evolution Strategies

Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
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

https://arxiv.org/pdf/1703.03864.pdf

the number of CPUs available: By using hundreds to thousands of parallel workers, ES can solve 3D humanoid walking in 10 minutes and obtain competitive results on most Atari games after one hour of training time. In addition, we