DTU Course 02456 Deep learning
6 Reinforcement learning
2017 updates

Ole Winther

Dept for Applied Mathematics and Computer Science
Technical University of Denmark (DTU)

October 8, 2017

Objectives of 2017 RL update

- P1: Deep Q learning
- brief intro taken from this blogpost
- P2: Evolutionary strategies
- a very simple way gradient-free approach to optimisation and RL

Part 1:
Deep Q learning

Deep Q learning - Bellman equation

- R_t is the future discounted cumulative reward at t:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = r_t + \gamma R_{t+1}$$

Deep Q learning - Bellman equation

- R_t is the future discounted cumulative reward at t:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = r_t + \gamma R_{t+1}$$

- Quality $Q(s_t, a_t)$ at time t in state s_t take action a_t and then take best possible action from there on:

$$Q(s_t, a_t) = \max_{a_{t+1}, \ldots, a_\infty} R_{t+1}$$

- If Q is known then the optimal policy is:

$$\pi(s_t) = \text{argmax}_{a_t} Q_t(s_t, a_t)$$

Deep Q learning - Bellman equation

- R_t is the future discounted cumulative reward at t:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = r_t + \gamma R_{t+1}$$

- Quality $Q(s_t, a_t)$ at time t in state s_t take action a_t and then take best possible action from there on:

$$Q(s_t, a_t) = \max_{a_{t+1}, \ldots, a_\infty} R_{t+1}$$

- If Q is known then the optimal policy is:

$$\pi(s_t) = \text{argmax}_{a_t} Q_t(s_t, a_t)$$

- Bellman equation = recursion for Q for $s_t, a_t \rightarrow r_{t+1}, s_{t+1}$,

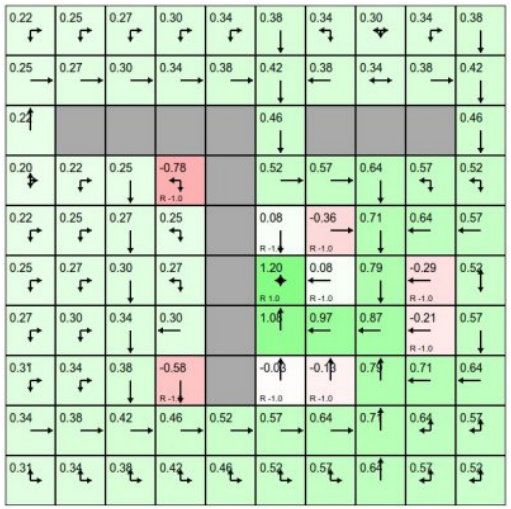
$$Q(s_t, a_t) = r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$$

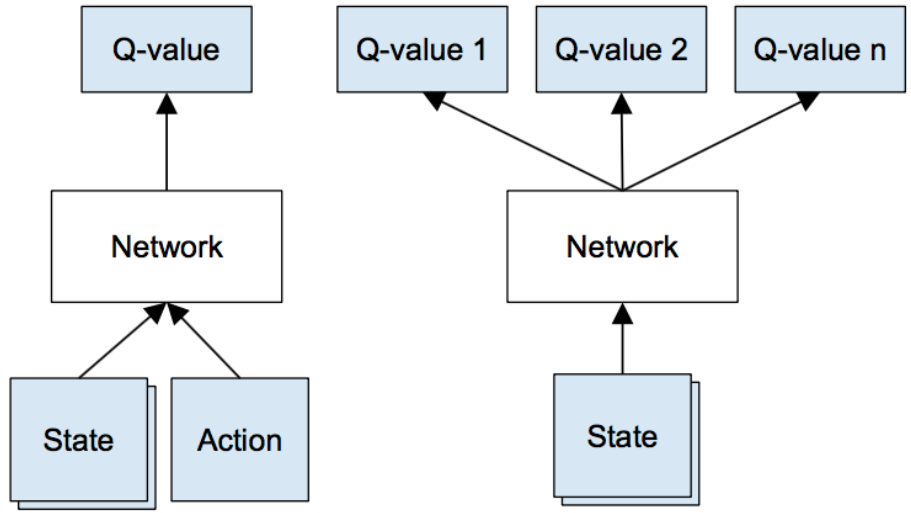
Deep Q learning - learning algorithm

- Bellman equation = recursion for Q for $s_t, a_t \rightarrow r_{t+1}, s_{t+1}$,

$$Q(s_t, a_t) = r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$$

- Learning:
- For grid world use tabulation (dynamic programming)
- For general state space use learning approach (= Deep Q learning).





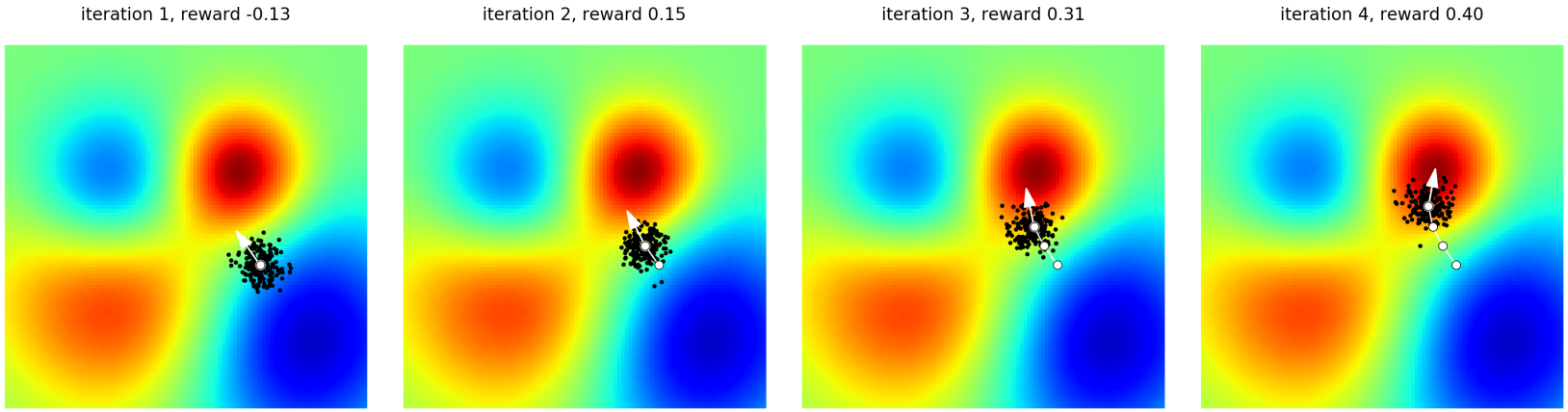
Part 2:
Evolutionary strategies

Evolution Strategies as a Scalable Alternative to Reinforcement Learning

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

https://arxiv.org/pdf/1703.03864.pdf
https://blog.openai.com/evolution-strategies/

Buy a lot of computer and forget about gradients



- Parameters θ
- Return (reward) $F(\theta)$
- Evaluate new parameter settings

$$\theta + \sigma\epsilon_i \ ,$$

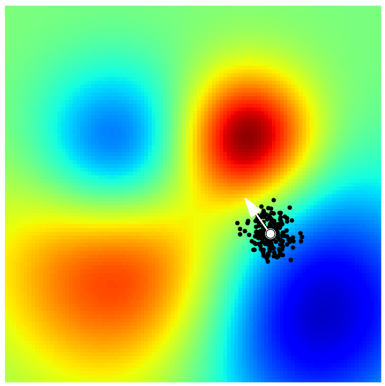
- $\epsilon_i \sim \mathcal{N}(0, I)$, $i = 1, \ldots, n$.
- Update in direction that makes $F(\theta)$ larger!

The algorithm

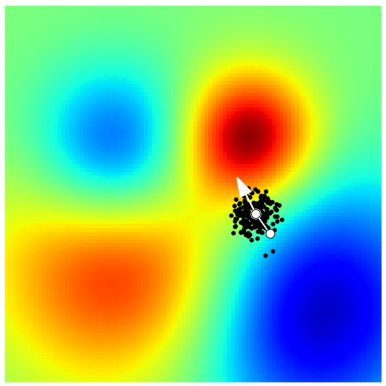
Algorithm 1 Evolution Strategies

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

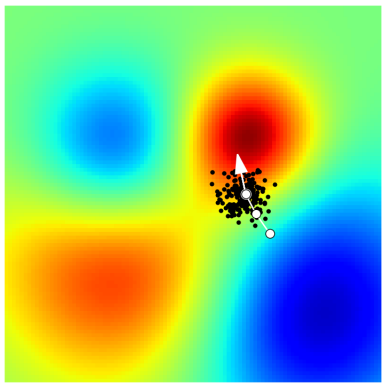
iteration 1, reward -0.13



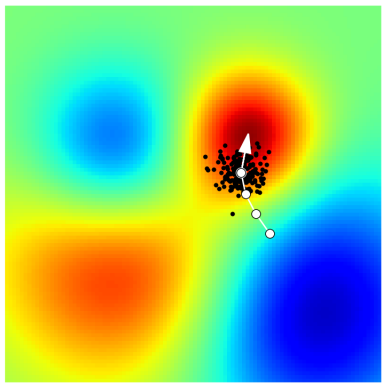
iteration 2, reward 0.15



iteration 3, reward 0.31



iteration 4, reward 0.40



Derivation

- $F(\theta)$ might be non-differentiable
- So replace with smoothed version:

$$\mathbb{E}_{\psi \sim \mathcal{N}(\theta, \sigma^2 I)} \left[F(\psi) \right]$$

- The derivative is (reuse policy gradient result):

$$\nabla_\theta \mathbb{E}_{\psi \sim \mathcal{N}(\theta, \sigma^2 I)} \left[F(\psi) \right] = \mathbb{E}_{\psi \sim \mathcal{N}(\theta, \sigma^2 I)} \left[F(\psi) \nabla_\theta \log \mathcal{N}(\psi | \theta, \sigma^2) \right]$$

- Replace average by samples $\psi_i = \theta + \epsilon_i$, $i = 1, \ldots, n$:

$$\mathbb{E}_{\psi \sim \mathcal{N}(\theta, \sigma^2 I)} \left[F(\psi) \nabla_\theta \log \mathcal{N}(\psi | \theta, \sigma^2) \right] \approx \frac{1}{n} \sum_{i=1}^{n} F(\psi_i) \frac{\psi_i - \theta}{\sigma^2}$$

- with $\psi_i = \theta + \sigma \sigma_i$.

Variance of estimators

- Really hand-wavy here
- Policy gradient uses a stochastic actions $a \sim p(a|s)$ during training
- So if simulation is T steps long then

$$\text{Variance gradient estimator} \propto T$$

- Evolutionary strategies can use deterministic actions
- so suitable for long simulations with very delayed rewards.

Variance of estimators

- Really hand-wavy here
- Policy gradient uses a stochastic actions $a \sim p(a|s)$ during training
- So if simulation is T steps long then

$$\text{Variance gradient estimator} \propto T$$

- Evolutionary strategies can use deterministic actions
- so suitable for long simulations with very delayed rewards.
- Downside: (stochastic) finite difference gradient estimation scales badly with effective model dimensionality.
- In practice: robust, highly parallel and 3-10$\times$ the compute as highly customised RL algorithms!

Thanks!
Ole Winther