Lecture 3: Introduction to Deep Reinforcement Learning

Chunlin Chen & Zhi Wang

Department of Control and Systems Engineering
Nanjing University

Nov. 19th, 2019

Table of Contents

For large/continuous state/action spaces

- **Curse of dimensionality**: Computational requirements grow exponentially with the number of state variables
  - Theoretically, all state-action pairs need to be visited infinite times to guarantee an optimal policy
  - In many practical tasks, almost every state encountered will never have been seen before
- **Generalization**: How can experience with a limited subset of the state space be usefully generalized to produce a good approximation over a much larger subset?

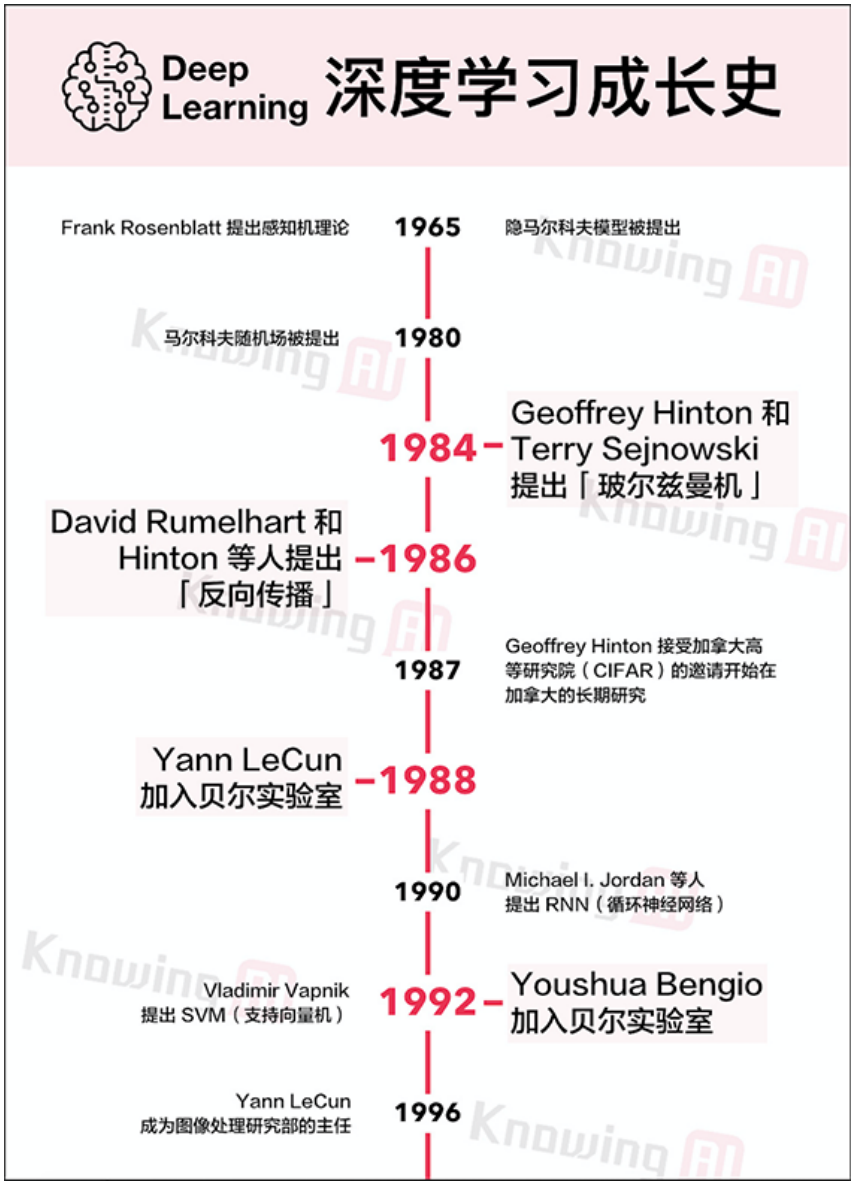
Function approximation

- It takes examples from a desired function (e.g., a value function) and attempts to generalize from them to construct an approximation to the entire function
  - Linear function approximation: $Q(s,a) = \sum_i \phi_i(s,a) w_i$
  - Neural network approximation: $Q(s,a) = Q(s,a;\boldsymbol{\theta})$
- Function approximation is an instance of supervised learning, the primary topic studied in machine learning, artificial neural networks, pattern recognition, and statistical curve fitting
  - In theory, any of the methods studied in these fields can be used in the role of function approximator within RL algorithms
  - RL with function approximation involves a number of **new issues** that do not normally arise in conventional supervised learning, e.g., non-stationarity, bootstrapping, and delayed targets

Deep reinforcement learning (DRL)

- The revolution of Deep Learning
  - Turing award 2018 to deep learning godfathers
- DRL = theories of RL + the help of deep function approximators

Deep reinforcement learning (DRL)



What do you think machine learning is?

Machine Learning

what society thinks I do

what my friends think I do

what my parents think I do

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i$$

$$\alpha_i \geq 0, \forall i$$

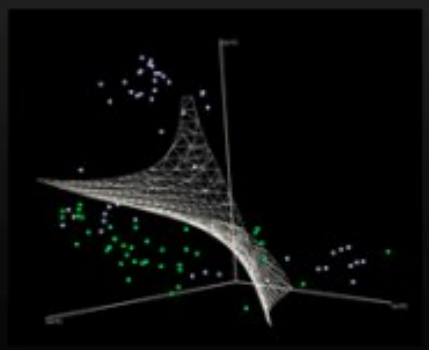
$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i, \sum_{i=1}^{l} \alpha_i y_i = 0$$

$$\nabla \hat{g}(\theta_t) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(x_i, y_i; \theta_t) + \nabla r(\theta_t).$$

$$\theta_{t+1} = \theta_t - \eta_t \nabla \ell(x_{i(t)}, y_{i(t)}; \theta_t) - \eta_t \cdot \nabla r(\theta_t)$$

$$\mathbb{E}_{i(t)}[\ell(x_{i(t)}, y_{i(t)}; \theta_t)] = \frac{1}{n} \sum_i \ell(x_i, y_i; \theta_t).$$

what other programmers think I do

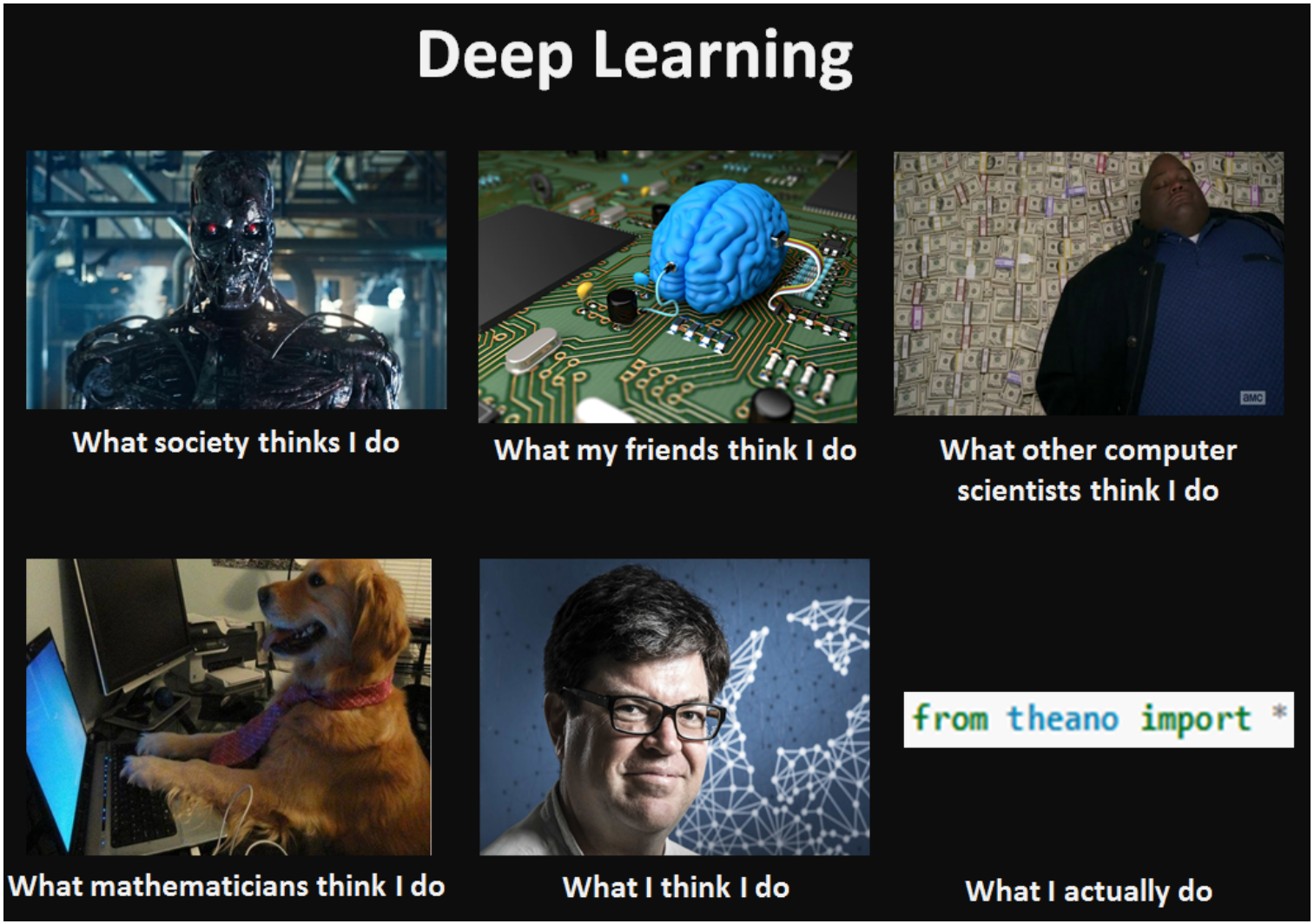


what I think I do

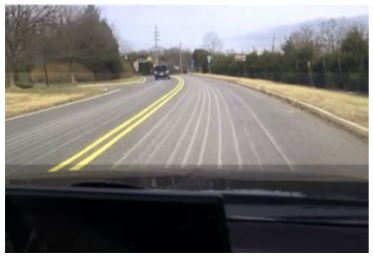
```
>>> from sklearn import svm
```

what I really do

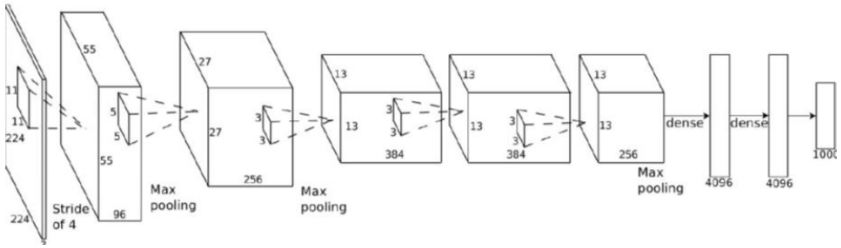
What do you think deep learning is?



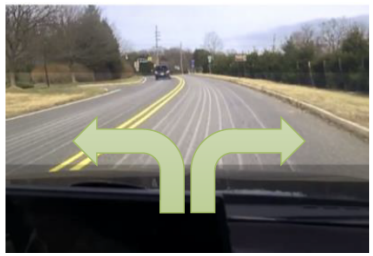
Imitation learning - Imitating expert demonstrations



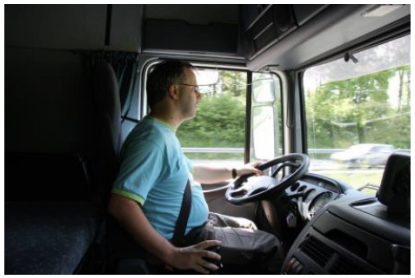
$\mathbf{o}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

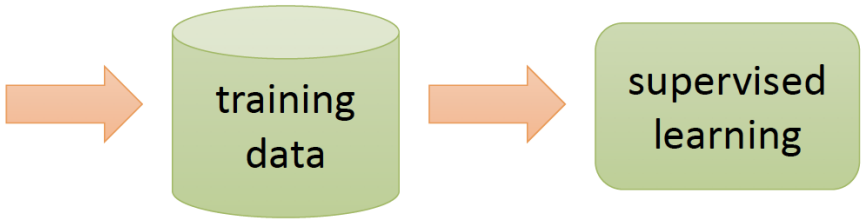


$\mathbf{a}_t$



$\mathbf{o}_t$

$\mathbf{a}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

- Supervised training has better stability than many RL methods

Table of Contents

Imitation learning - Imitating expert demonstrations

- One of the biggest challenge is collecting expert demonstrations
  - Unless it has a huge business potential, the attached cost can be prohibitive
- The trained policy is only as good as the demonstrations
  - Initialize a policy from expert demonstrations, finetune it using RL
- Error accumulates fast in a trajectory and put us into situations that we never deal with before
  - Analogous to the key challenge in mode-based RL

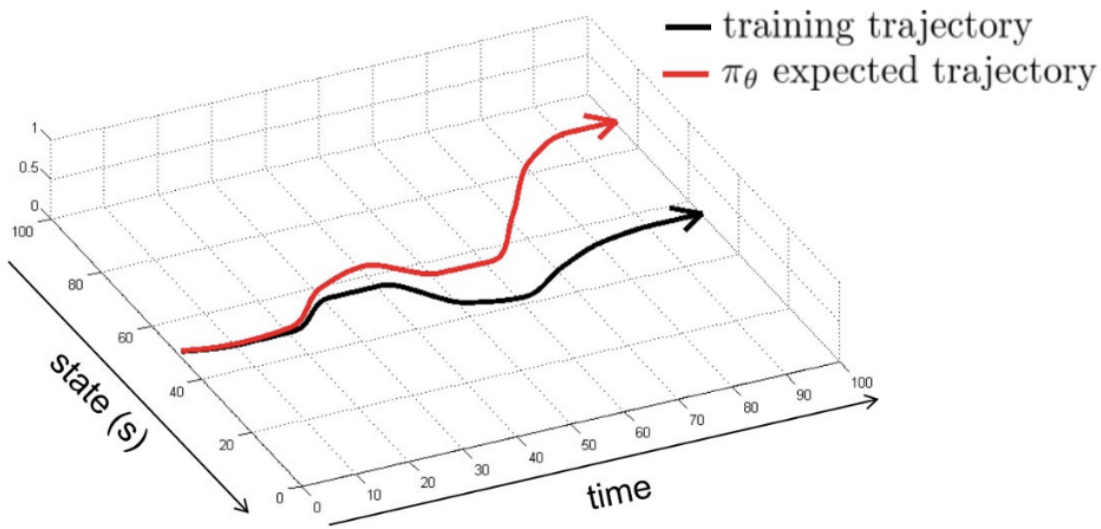
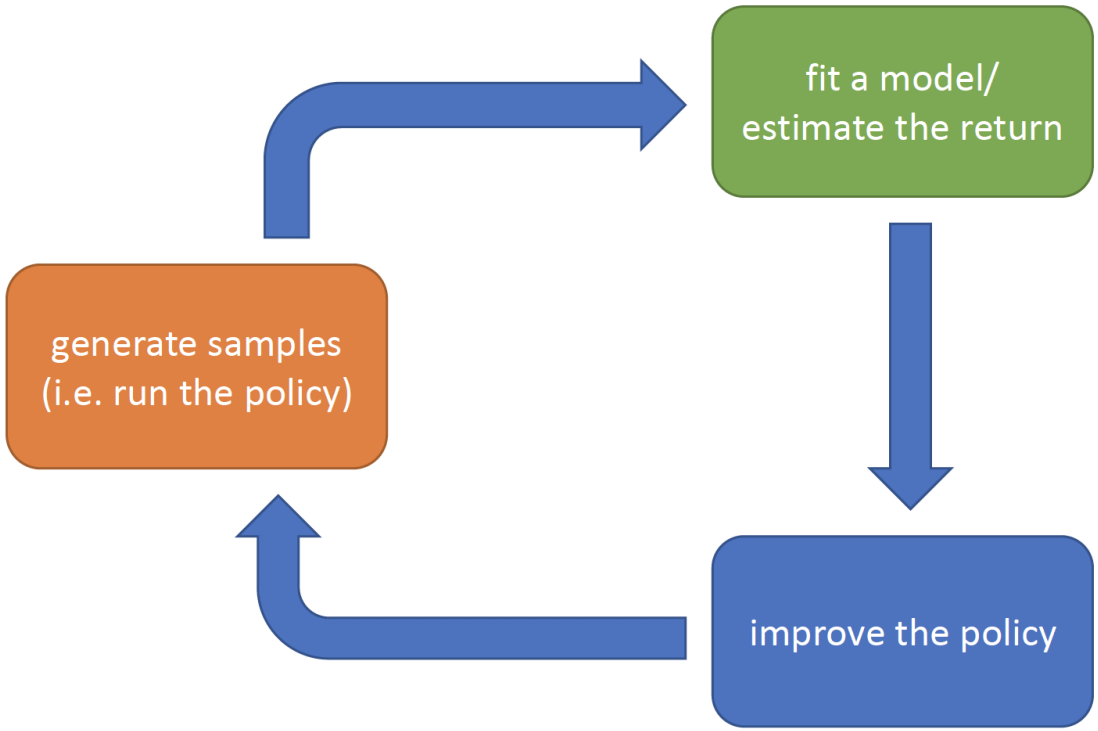


Table of Contents

The anatomy of a DRL algorithm



- Following the **Generalized Policy Iteration** framework
  - ... Generate samples $\Rightarrow$ Policy evaluation $\Rightarrow$ Policy improvement ...

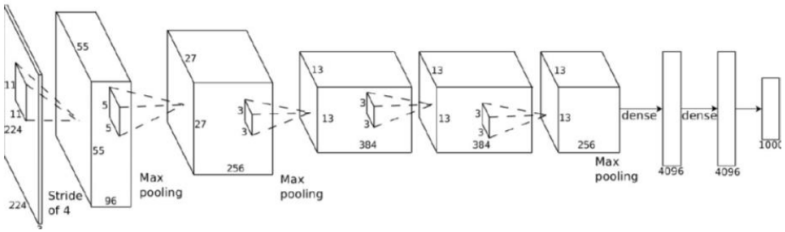
$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \ldots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

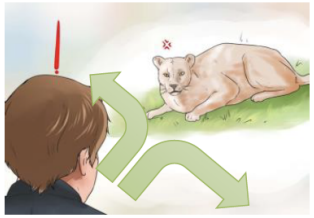
Types DRL algorithms

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathbb{E}_{\tau \sim p_{\boldsymbol{\theta}}(\tau)} \left[\sum_t r(s_t, a_t) \right]$$

- **Policy gradients**: Directly approximate the policy and differentiate the above objective
- **Value function-based**: Estimate state- or action-value function or of the optimal policy (no explicit policy)
- **Actor-critic**: Estimate state- or action-value function or of the current policy, use it to improve policy
- **Model-based RL**: Estimate the transition model, $p(s', r|s, a)$
  - Use it for planning (no explicit policy), use it to improve a policy, etc

DRL - Directly approximate policies





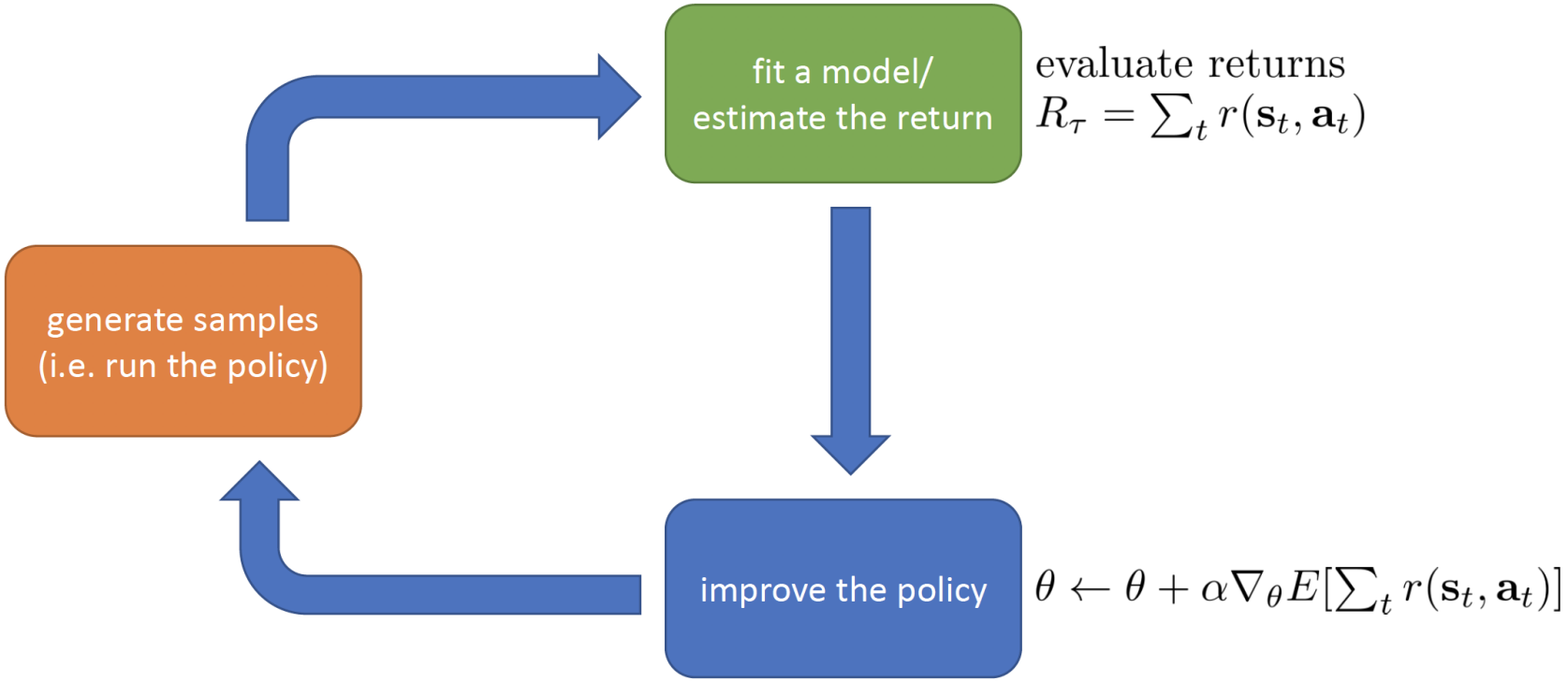
$\mathbf{o}_t$ $\quad \pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\quad \mathbf{a}_t$

$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

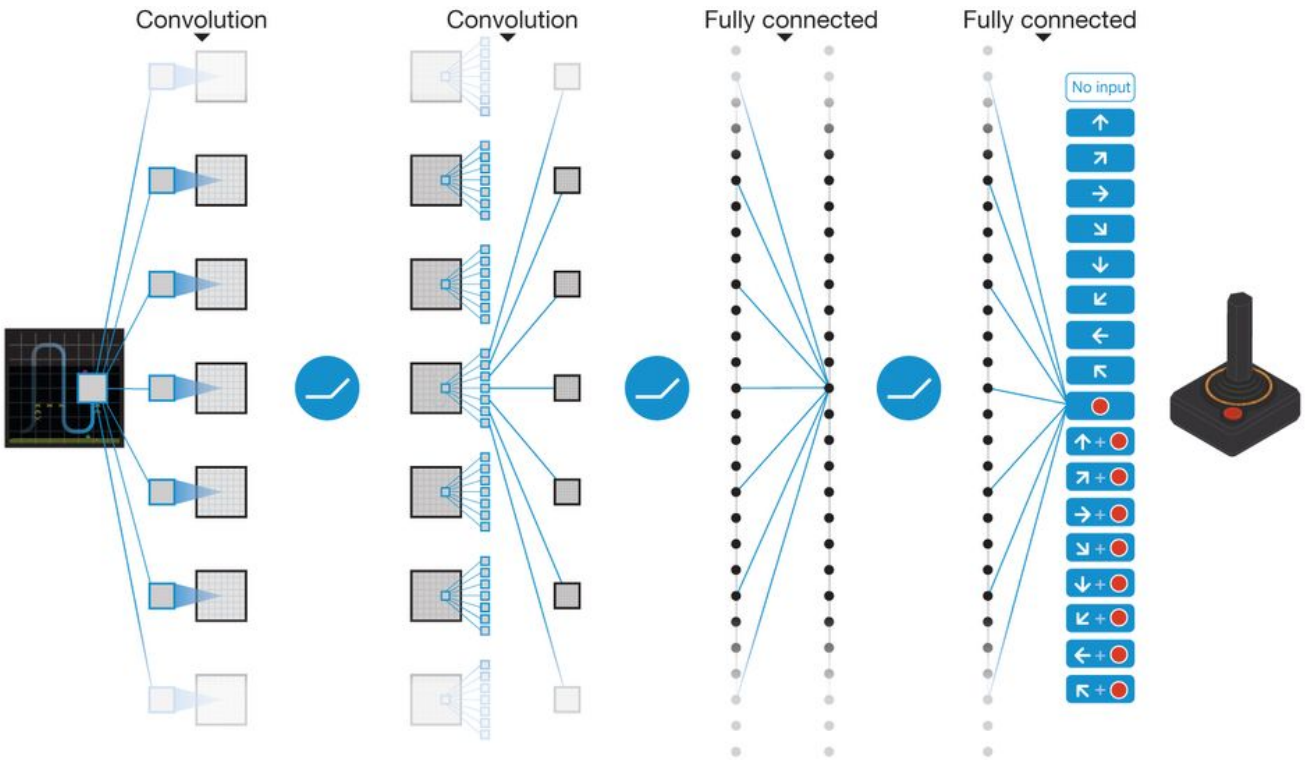
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)

- Usually using deep neural networks
- Optimize the policy network by backpropagation

DRL - Directly approximate policies

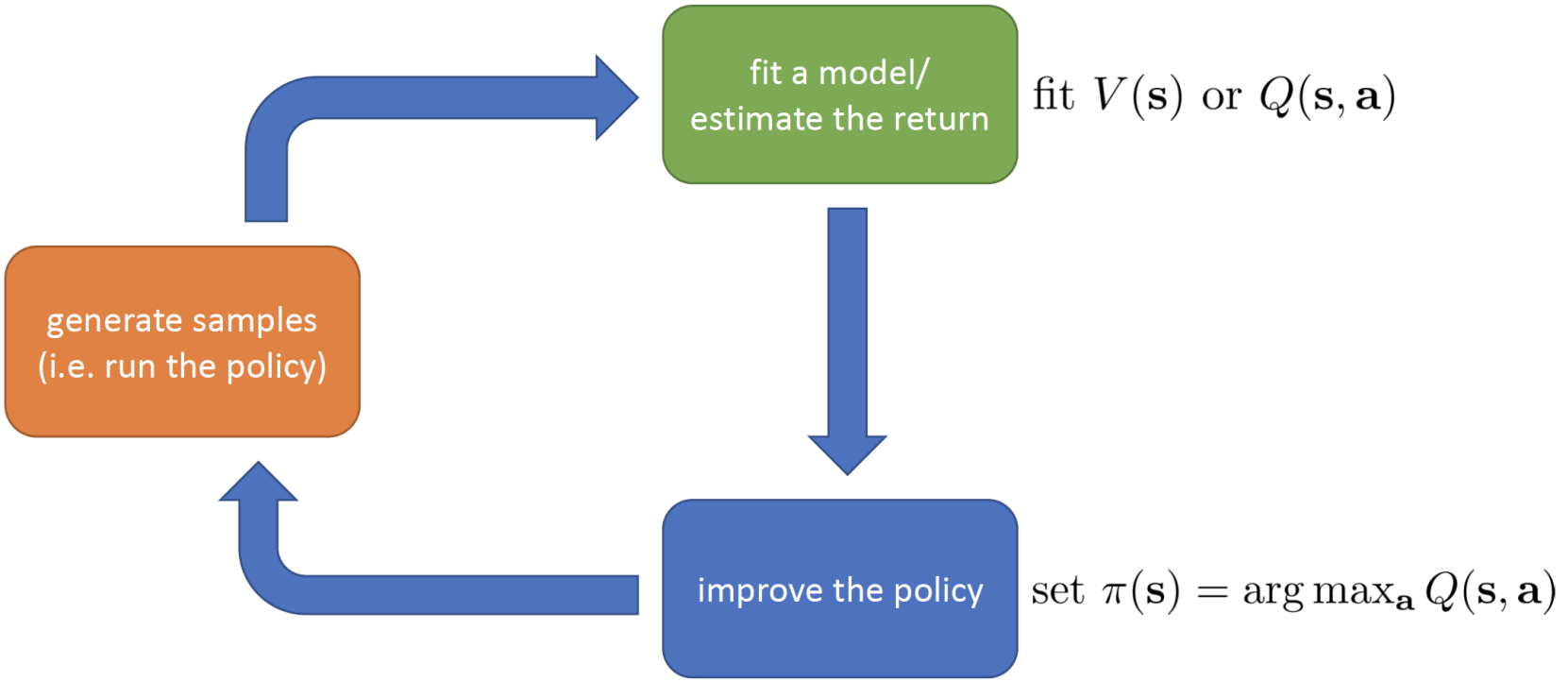


DRL - Approximate value functions



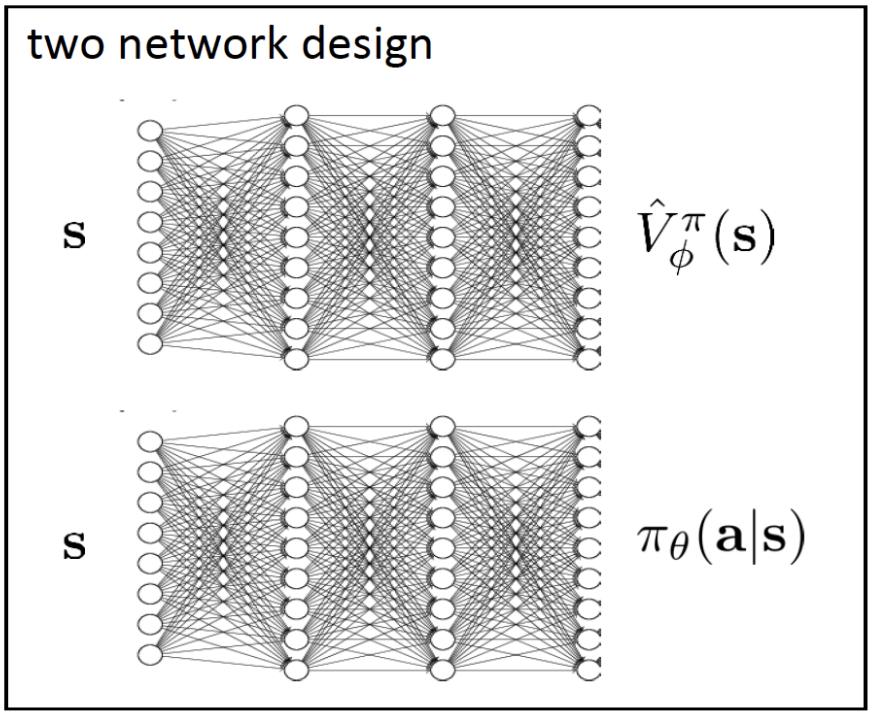
- Usually using deep neural networks
- Optimize the deep Q-network by minimizing the **Bellman residual**

DRL - Approximate value functions

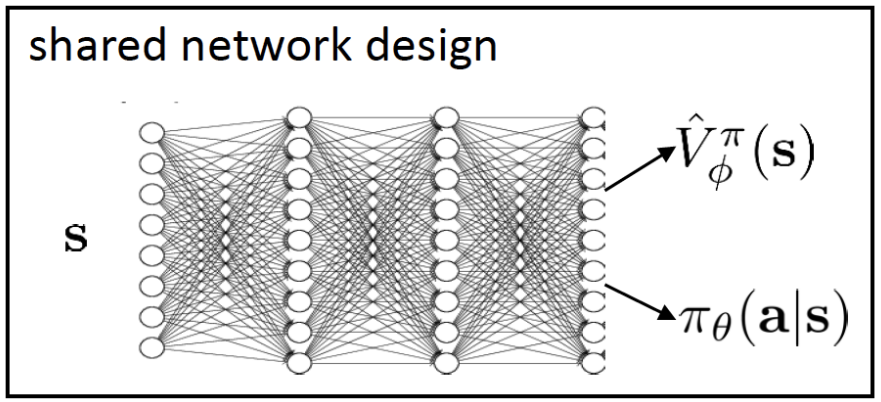


DRL - Approximate both

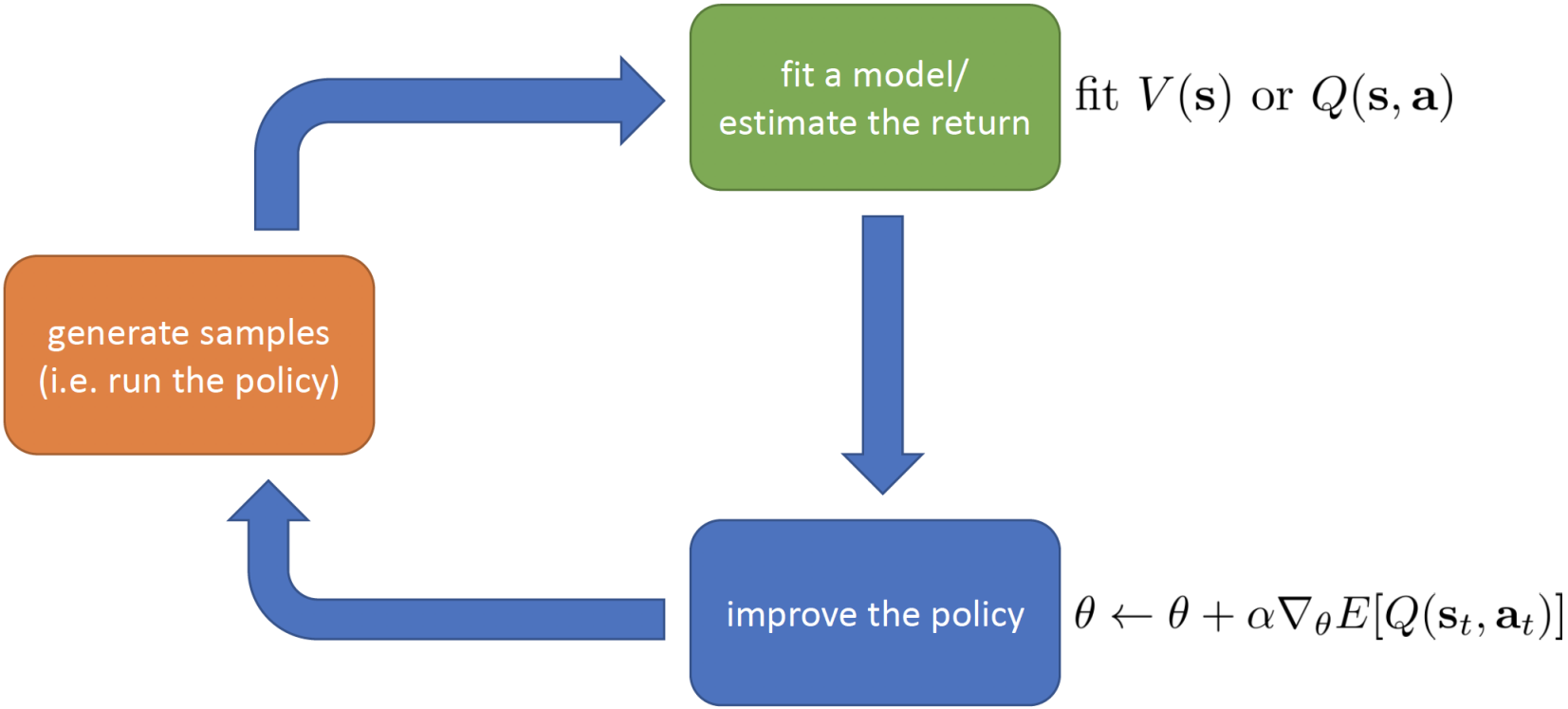
two network design



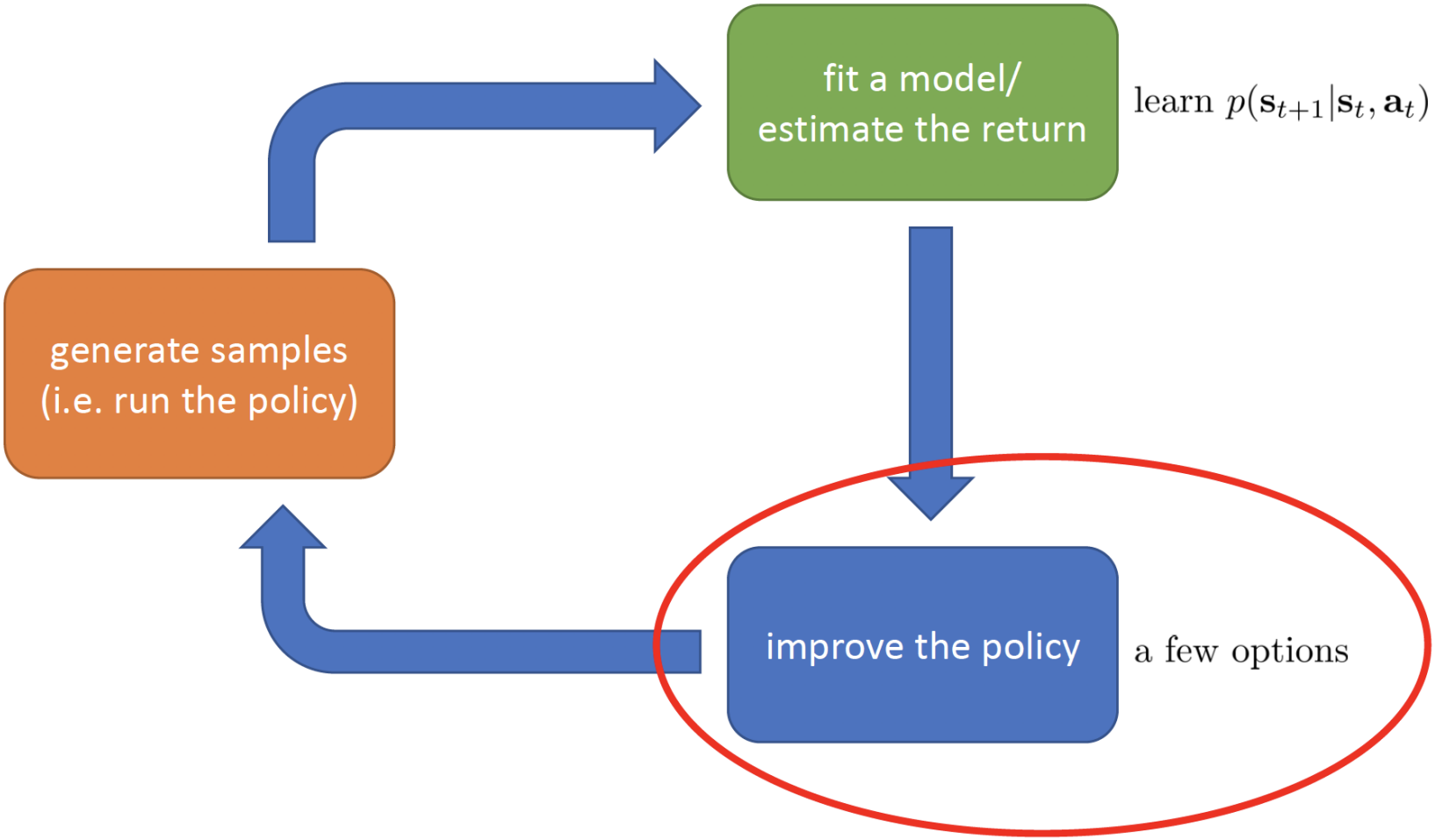
shared network design



DRL - Approximate both



DRL - Model-based

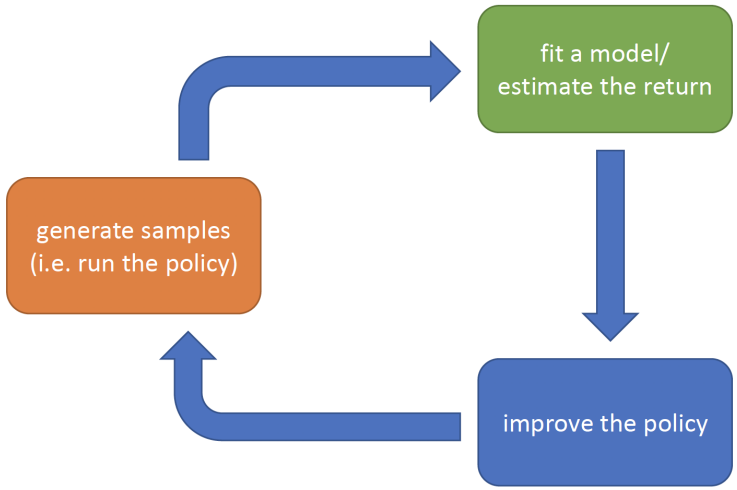


DRL - Model-based

- Just use the model to plan (no explicit policy)
  - Trajectory optimization / optimal control
  - Discrete planning in discrete action spaces, e.g., Monte Carlo tree search
- Backpropagate gradients into the policy
- Use the model to learn a value function
  - Dynamic programming
  - Generate simulated experience for model free learner

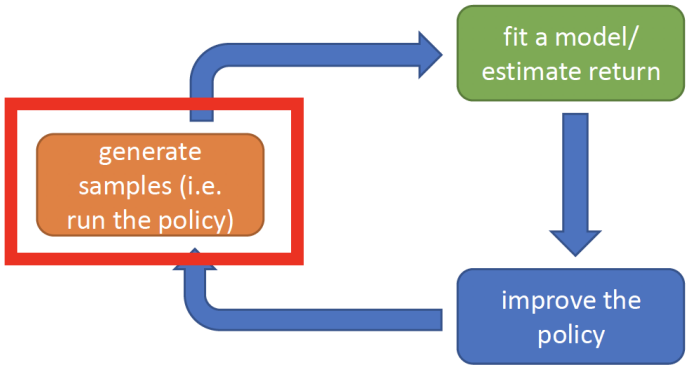
Why so many DRL algorithms?

- Different tradeoffs
  - Sample efficiency – model-based
  - Stability & easy to use – model-free
- Different assumptions
  - Stochastic or deterministic?
  - Continuous or discrete?
  - Episodic or infinite horizon?
- Different things are easy or hard in different settings
  - Easier to represent the policy?
  - Easier to represent the model?



Comparison: Sample efficiency

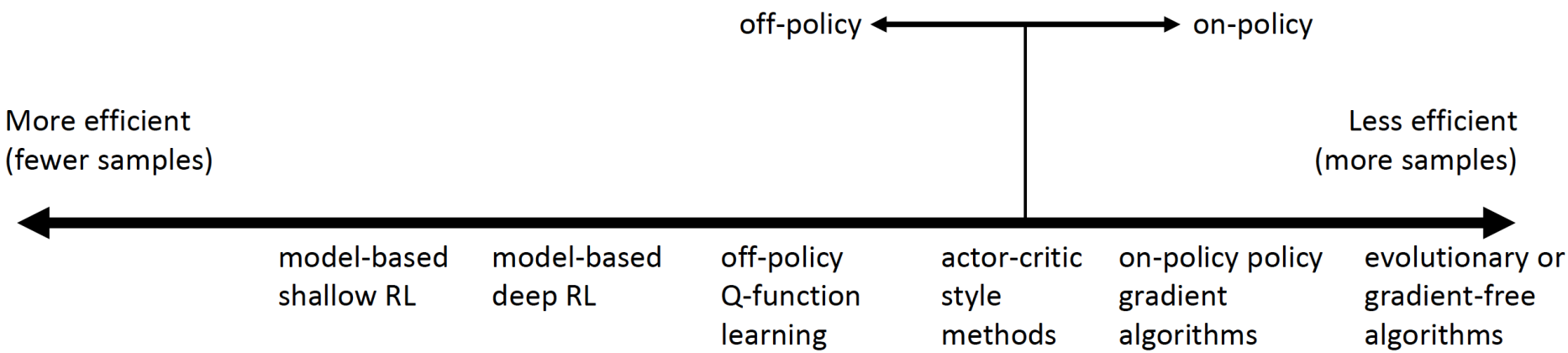
- Sample efficiency = how many samples do we need to get a good policy?
- Most important question: is the algorithm off-policy?
  - Off-policy: able to improve the policy without generating new samples from that policy
  - On-policy: each time the policy is changed, even a little bit, we need to generate new samples



$$\theta \leftarrow \theta + \alpha \nabla_\theta E[\textstyle\sum_t r(\mathbf{s}_t, \mathbf{a}_t)]$$

just one gradient step

Comparison: Sample efficiency



Comparison: Stability and ease of use

- Does it converge?
- And if it converges, to what?
- And does it converge every time?

Why is any of this even a question???

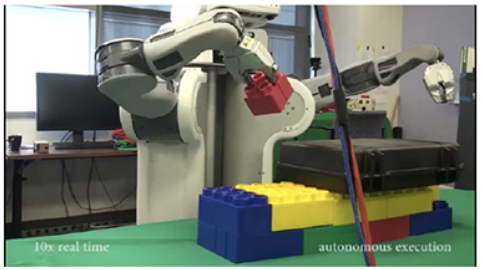
- Supervised learning: almost always gradient descent
- Reinforcement learning: often not gradient descent
  - Q-learning: Bellman fixed point iteration
  - Model-based RL: model is not optimized for expected reward
  - Policy gradient: is gradient descent, but also often the least efficient!

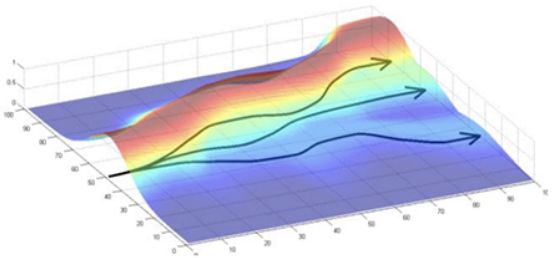
Comparison: Stability and ease of use

- Value function fitting
  - At best, minimizes error of fit ("Bellman error"), not the same as expected reward
  - At worst, doesn't optimize anything. Many popular deep RL value fitting algorithms are not guaranteed to converge to anything in the nonlinear case
- Model-based RL
  - Model minimizes error of fit. This will converge.
  - No guarantee that better model = better policy
- Policy gradient
  - The only one that actually performs gradient descent (ascent) on the true objective

Comparison: Assumptions

- Common assumption 1: full observability
  - Generally assumed by value function fitting methods
  - Can be mitigated by adding recurrence
- Common assumption 2: episodic learning
  - Often assumed by pure policy gradient methods
  - Assumed by some model-based RL methods
- Common assumption 3: continuity or smoothness
  - Assumed by some continuous value function learning methods
  - Often assumed by some model-based RL methods





Examples of specific algorithms

- Policy gradient methods
  - REINFORCE
  - Natural policy gradient
  - Trust region policy optimization
- Value function fitting methods
  - Fitted value iteration, fitted Q-iteration
  - Deep Q-network, deep Q-learning
- Actor critic algorithms
  - Deep deterministic policy gradient (DDPG)
  - Asynchronous advantage actor critic (A3C)
  - Soft actor critic (SAC)
- Model based RL algorithms
  - Dyna
  - Guided policy search

We'll learn about most of these in the next few weeks!

Example 1: Atari games with deep Q-networks

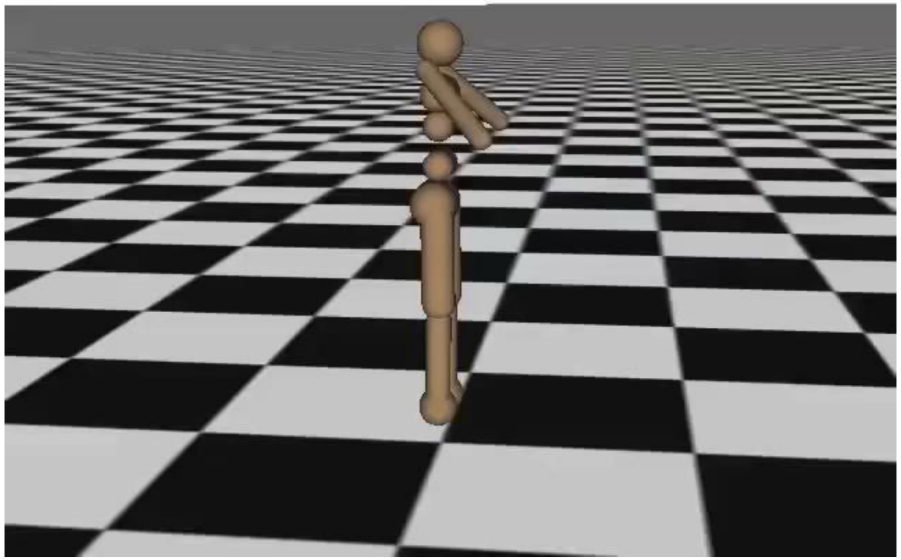
- Playing Atari with deep reinforcement learning
- Q-learning with convolutional neural networks

Example 2: robots and model-based RL

- End-to-end training of deep visuomotor policies
- Guided policy search (model-based RL) for image based robotic manipulation

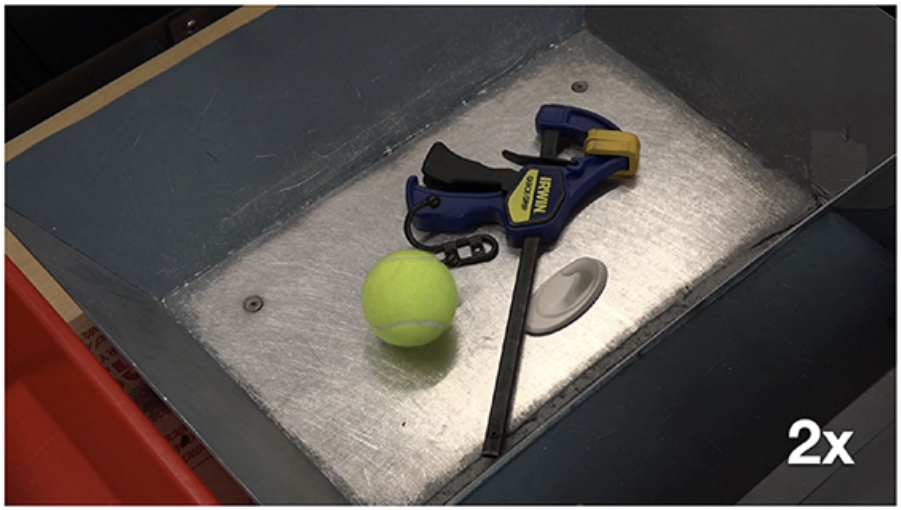
Example 3: walking with policy gradients

- High-dimensional continuous control with generalized advantage estimation
- Trust region policy optimization with value function approximation



Example 4: robotic grasping with Q-functions

- Q learning from images for real world robotic grasping



Learning objectives of this lecture

- You should be able to...
  - Understand the function approximation mechanism for RL problems with large or continuous state-action spaces
  - Understand the anatomy of imitation learning
  - Know the main types of DRL algorithms, and their differences

References

- Lecture 4 of CS285 at UC Berkeley, *Deep Reinforcement Learning, Decision Making, and Control*
  - `http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-4.pdf`

THE END