

# Lecture 4: Introduction to Deep Reinforcement Learning

Zhi Wang & Chunlin Chen

Department of Control and Systems Engineering  
Nanjing University

Oct. 22nd, 2020

# Table of Contents

1 From Tabular Algorithms to Deep RL

2 DRL Algorithms

3 Imitation Learning

## 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:  $Q(s, a) = Q(s, a; \theta)$
- 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



Yoshua Bengio



Geoffrey Hinton



Yann LeCun

# 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

$$L_2 = \|\mathbf{w}\|^2 + \sum_i \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_i \alpha_i$$

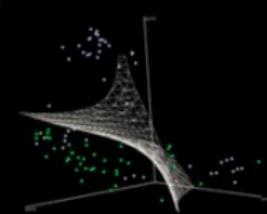
$$\alpha_i \geq 0, \forall i$$

$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i, \sum_i \alpha_i y_i = 0$$

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

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

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



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

what other programmers  
think I do

what I think I do

what I really do

# What do you think deep learning is?

## Deep Learning



What society thinks I do



What my friends think I do



What other computer scientists think I do



What mathematicians think I do



What I think I do

```
from theano import *
```

What I actually do

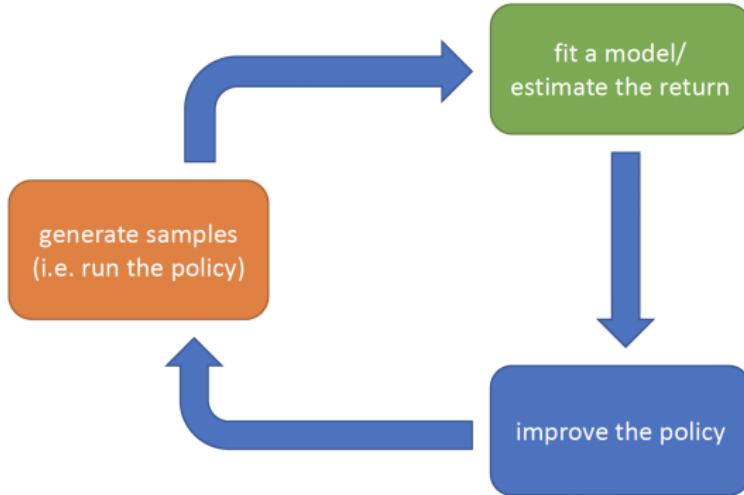
# Table of Contents

1 From Tabular Algorithms to Deep RL

2 DRL Algorithms

3 Imitation Learning

# The anatomy of a DRL algorithm



- Following the **Generalized Policy Iteration** framework
  - ... Generate samples => Policy evaluation => Policy improvement ...

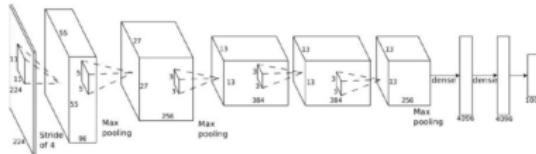
$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

# Types DRL algorithms

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

- **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



$\mathbf{o}_t$

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

$\mathbf{a}_t$

$\mathbf{s}_t$  – state

$\mathbf{o}_t$  – observation

$\mathbf{a}_t$  – action

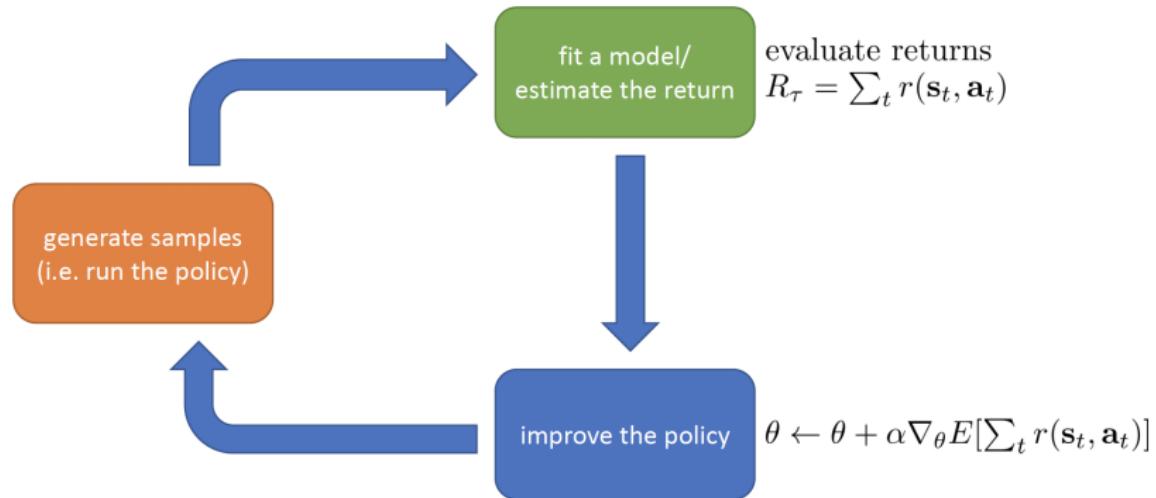
$\pi_\theta(\mathbf{a}_t | \mathbf{o}_t)$  – policy

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

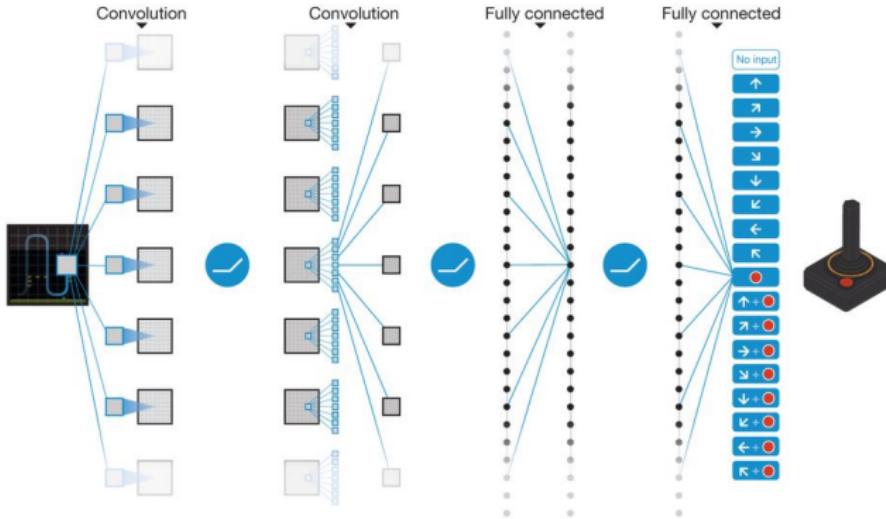


- Usually using deep neural networks
- Optimize the policy network by backpropagation

# DRL - Directly approximate policies

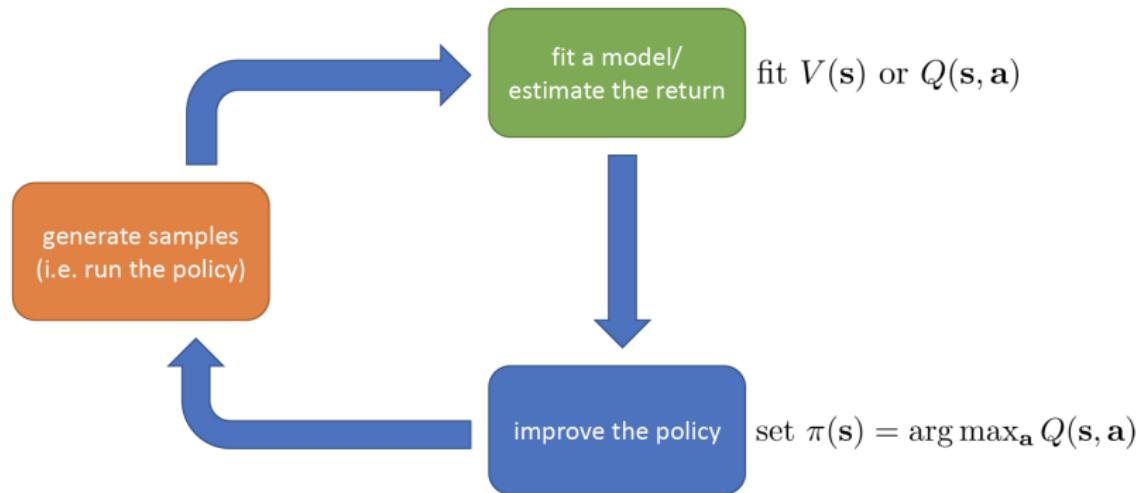


# DRL - Approximate value functions



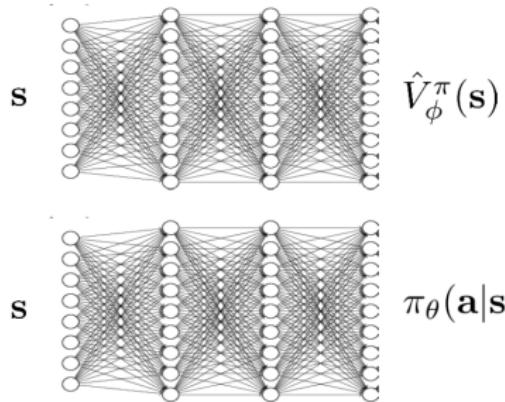
- Usually using deep neural networks
- Optimize the deep Q-network by minimizing the **Bellman residual**

# DRL - Approximate value functions

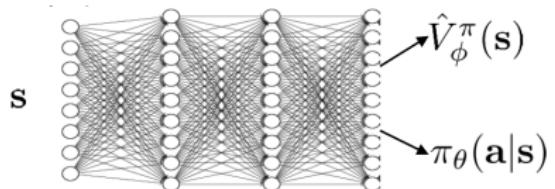


# DRL - Approximate both

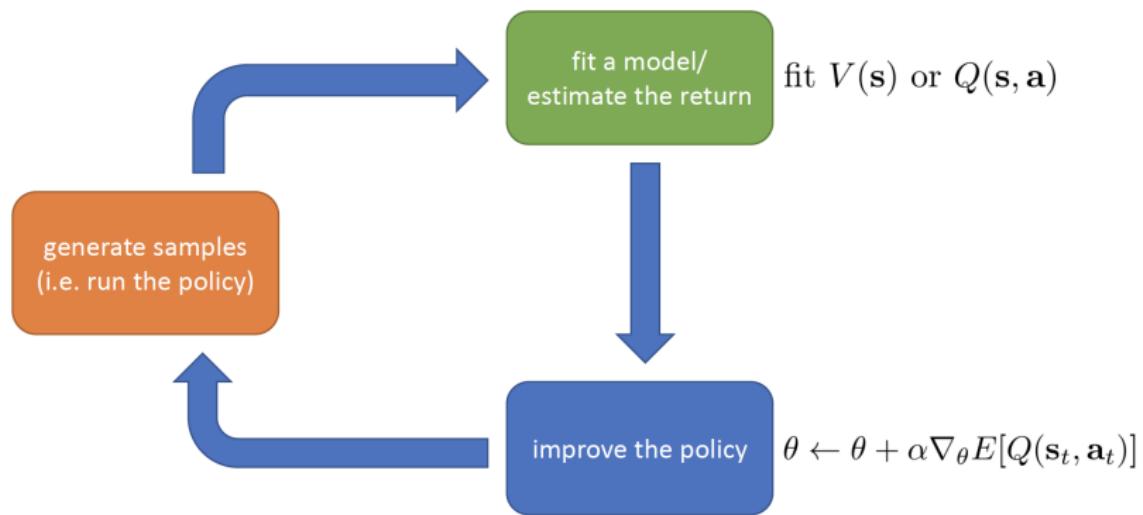
two network design



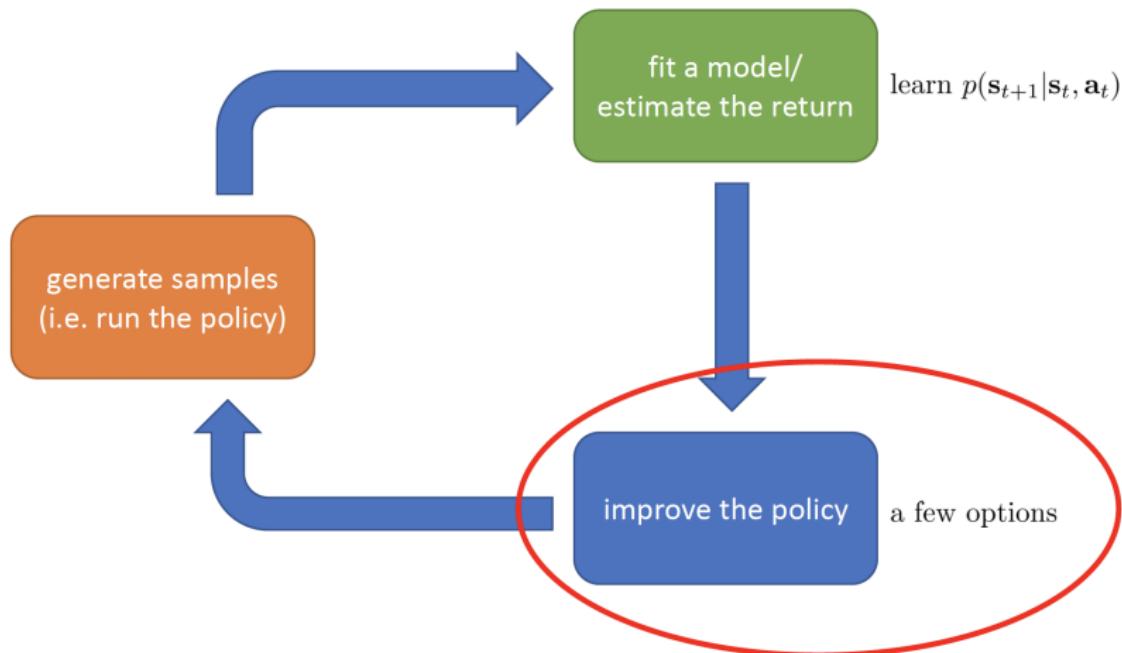
shared network design



# DRL - Approximate both



# DRL - Model-based



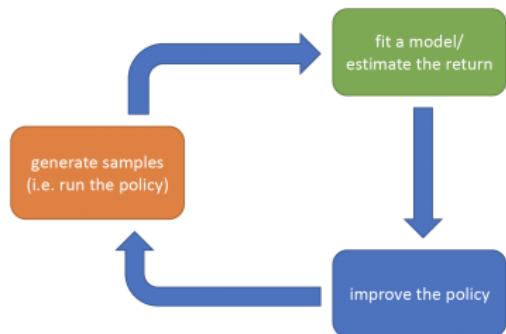
improve the policy

a few options

- 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

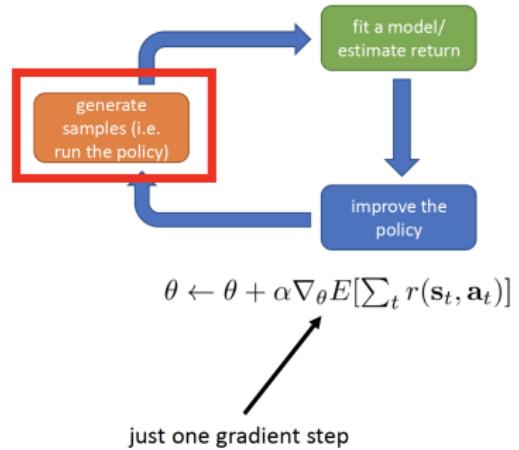
# Why so many DRL algorithms?

- 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?

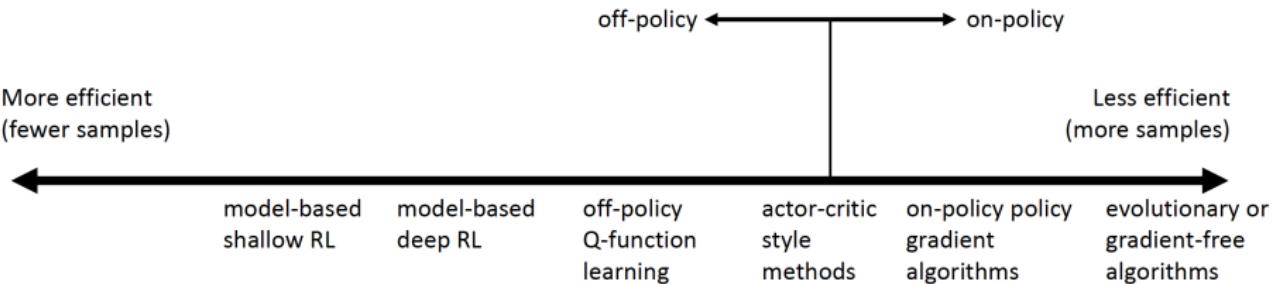


# Comparison: Sample efficiency

- 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



# 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???

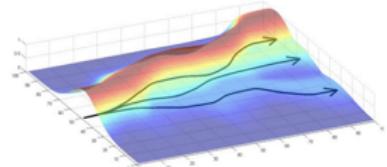
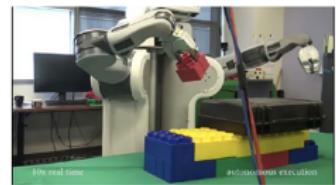
- Supervised learning: almost always gradient descent
- Reinforcement learning: often not gradient descent
  - Q-learning: Bellman fixed point iteration
  - Model-based RL: model is not optimized for expected reward
  - Policy gradient: is gradient descent, but also often the least efficient!

# 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

2018 IEEE International Conference on Robotics and Automation (ICRA)  
May 21-25, 2018, Brisbane, Australia

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

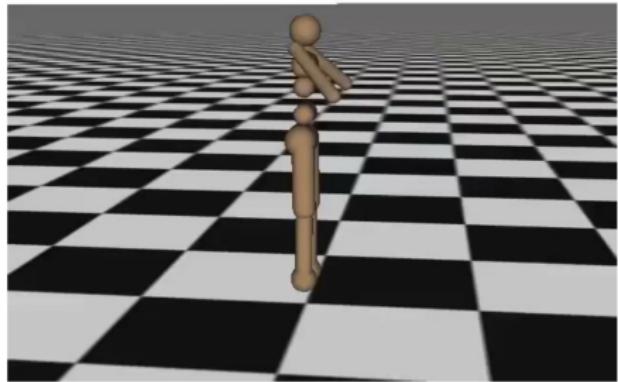
## Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, Sergey Levine  
University of California, Berkeley



## 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

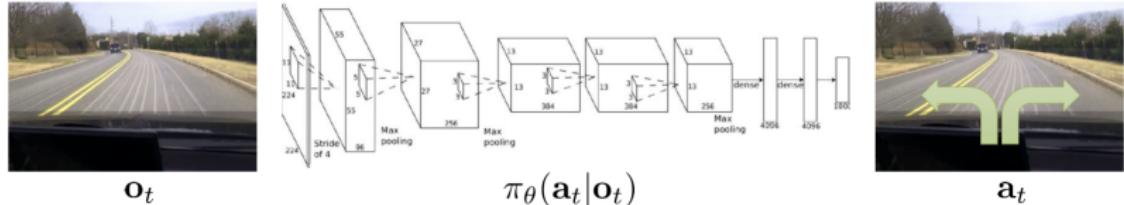
- Q learning from images for real world robotic grasping



# Table of Contents

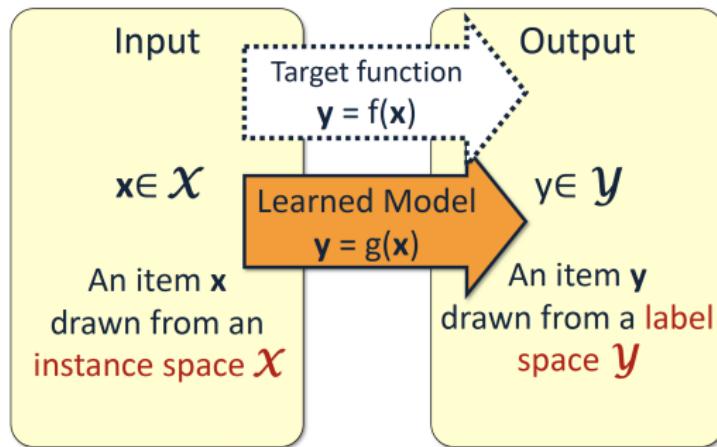
- 1 From Tabular Algorithms to Deep RL
- 2 DRL Algorithms
- 3 Imitation Learning

# Imitation learning - Imitating expert demonstrations



- Behavior cloning
- Supervised training has better stability than many RL methods

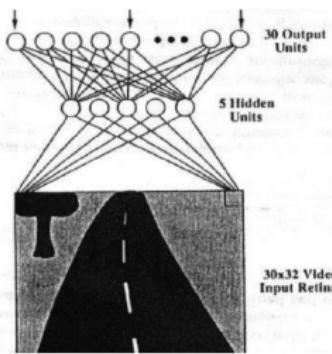
# Recall: supervised learning



- We need to choose what kind of model we want to learn
  - Linear model, nonlinear model...
  - Parametric model, nonparametric model...
  - Decision trees, neural networks, Gaussian processes...

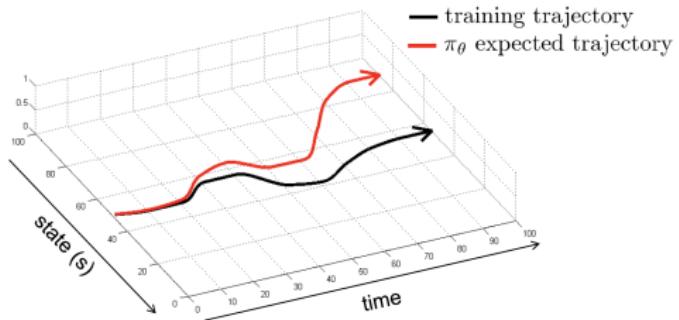
# The original deep imitation learning system

ALVINN: Autonomous Land Vehicle In a Neural Network  
1989

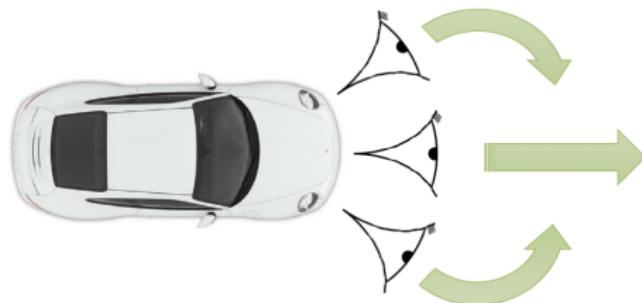
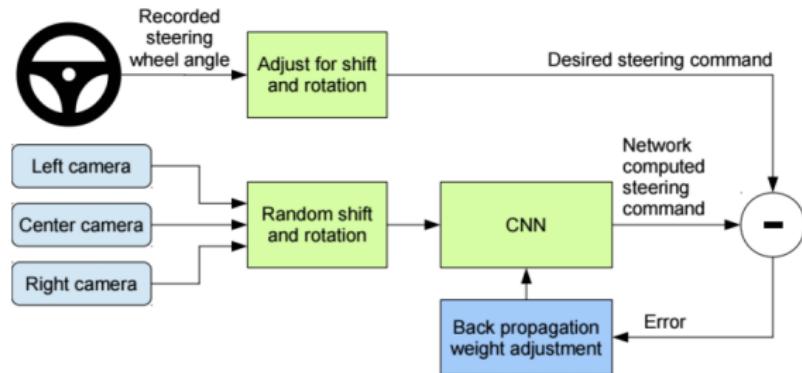


# Deep imitation learning system

- Input: states/observations  $o_t$
- Output: actions  $\hat{a}_t = \pi_\theta(\cdot | o_t)$
- Label/ground truth: actions  $a_t$  provided by human

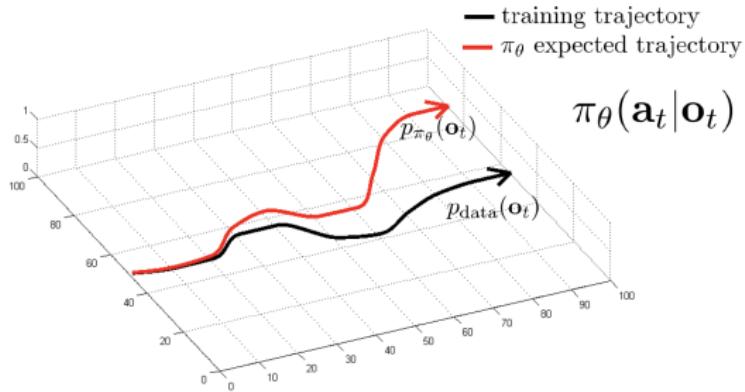


# The original deep imitation learning system



# Challenge: distribution shift

- 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



**Can we make  $p_{data}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$ ?**

# Alternative solution

Can we make  $p_{data}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$ ?

Idea: instead of being clever about  $p_{\pi_\theta}(\mathbf{o}_t)$ , be clever about  $p_{data}(\mathbf{o}_t)$

## DAgger: Dataset Aggregation

Goal: collect training data from  $p_{\pi_\theta}(\mathbf{o}_t)$  instead of  $p_{data}(\mathbf{o}_t)$

How? Just run  $p_{\pi_\theta}(\mathbf{o}_t)$

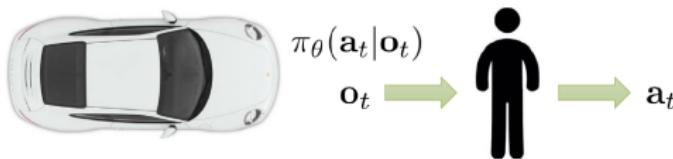
But need labels  $\mathbf{a}_t$ !

- 
1. train  $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$  from human data  $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_N, \mathbf{a}_N\}$
  2. run  $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$  to get dataset  $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
  3. Ask human to label  $\mathcal{D}_\pi$  with actions  $\mathbf{a}_t$
  4. Aggregate:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

# What is the problem?

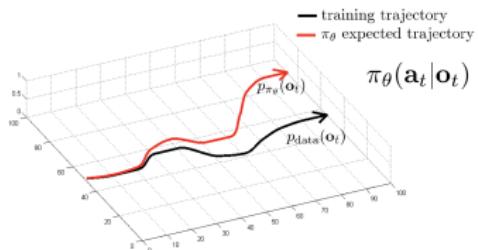
- One of the biggest challenge is collecting expert demonstrations
  - Unless it has a huge business potential, the attached cost can be prohibitive
- The trained policy is only as good as the demonstrations
  - Initialize a policy from expert demonstrations, finetune it using RL

- 
1. train  $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$  from human data  $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_N, \mathbf{a}_N\}$
  2. run  $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$  to get dataset  $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
  3. Ask human to label  $\mathcal{D}_\pi$  with actions  $\mathbf{a}_t$
  4. Aggregate:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$



# Deep imitation learning in practice

- DAgger addresses the problem of distributional “drift”
- What if our model is so good that it doesn't drift?
- Need to mimic expert behavior very accurately
- But don't overfit



# Why might we fail to fit the expert?

- RL: temporally correlated
- Imitation learning: Non-Markovian behavior

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

behavior depends only  
on current observation

$$\pi_{\theta}(\mathbf{a}_t | \mathbf{o}_1, \dots, \mathbf{o}_t)$$

behavior depends on  
all past observations

If we see the same thing  
twice, we do the same thing  
twice, regardless of what  
happened before

Often very unnatural for  
human demonstrators

## Recall: Markov property

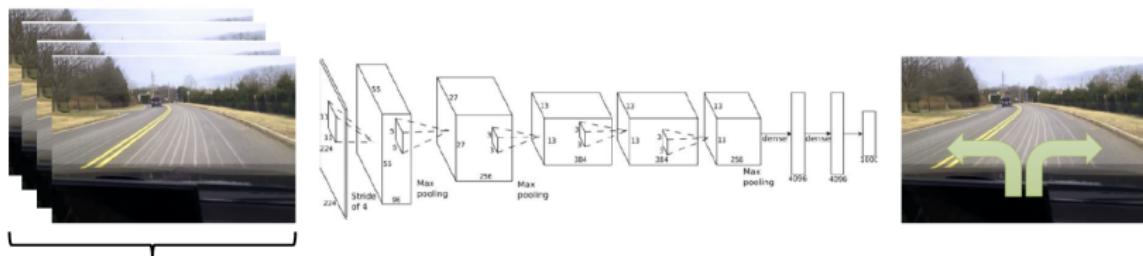
$$p(s', r | s, a) = \Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}$$

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) = 1, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

- The probabilities given by  $p$  completely characterize the environment's dynamics
- **Markov Property**
  - The probability of each possible value for  $S_t$  and  $R_t$  depends only on the immediately preceding state  $S_{t-1}$  and action  $A_{t-1}$ , not at all on earlier states and actions
  - $p(S_t, R_t | S_{t-1}, A_{t-1}) = p(S_t, R_t | S_{t-1}, A_{t-1}, S_{t-2}, A_{t-2}, S_{t-3}, A_{t-3}, \dots)$
- Recall supervised learning  $p(X_i | X_j) = 0$

# How can we use the whole history?

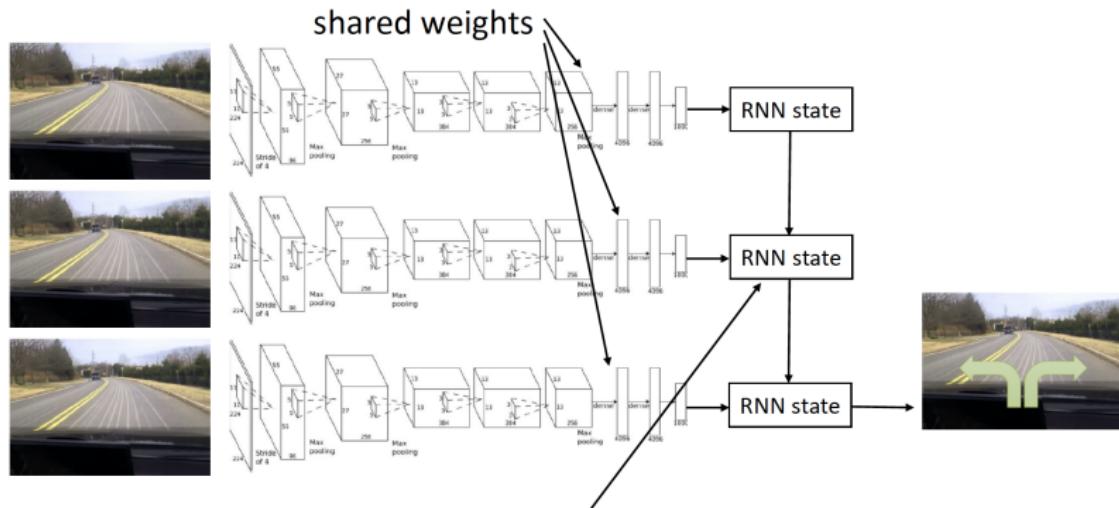
- Use multiple frames as the input
- The input dimension will be too high



variable number of frames,  
too many weights

# How can we use the whole history?

- Use recurrent neural network (RNN) as the encoder
- Embed the summary of past states into the internal **memory**



Typically, LSTM cells work better here

# Recap: Imitation learning



- Often (but not always) insufficient by itself
  - Distribution mismatch problem
- Sometimes work well
  - Samples from a stable trajectory distribution
  - Add more **on-policy** data, e.g., using Dagger
  - Better models that fit more accurately

# 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
  - Know the main types of DRL algorithms, and their differences
  - Understand the anatomy of imitation learning

# References

- Lecture 2 & 4 of CS285 at UC Berkeley, *Deep Reinforcement Learning, Decision Making, and Control*
  - <http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-2.pdf>
  - <http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-4.pdf>

# THE END