

# **TTIC 31230, Fundamentals of Deep Learning**

David McAllester, Winter 2019

## **Deep Reinforcement Learning**

# Definition of Reinforcement Learning

RL is defined by the following properties:

- An environment with **state**.
- State changes are influenced by **sequential decisions**.
- Reward (or loss) depends on **making decisions** that lead to **desirable states**.

# Reinforcement Learning Examples

- Board games (chess or go)
- Atari Games (pong)
- Robot control (driving)
- Dialog
- Life

# Policies

A policy is a way of behaving.

Formally, a (nondeterministic) policy maps a state to a probability distribution over actions.

$\pi(a_t|s_t)$  probability of action  $a_t$  in state  $s_t$

## Imitation Learning

Construct a training set of state-action pairs  $(s, a)$  from experts.

Define stochastic policy  $\pi_{\Phi}(s)$ .

$$\Phi^* = \operatorname{argmin}_{\Phi} E_{(s,a) \sim \text{Train}} - \ln \pi_{\Phi}(a \mid s)$$

This is just cross-entropy loss where we think of  $a$  as a “label” for  $s$ .

## Dangers of Imperfect Imitation Learning

Perfect imitation learning would reproduce expert behavior. Imitation learning is **off-policy** — the state distribution in the training data is different from that defined by the policy being learned.

Imitating experts, such as expert fire eaters, can be dangerous. “Don’t try this at home”.

Also, it is difficult to exceed expert performance by imitating experts. But this can happen.

## Markov Decision Processes (MDPs)

An MDP consists of a set  $\mathcal{S}$  of states, a set  $\mathcal{A}$  of allowed actions, a reward function  $R$  and a next-state probability function  $P_T$ . We will use the following notation.

$s_t \in \mathcal{S}$  is the state at time  $t$

$a_t \in \mathcal{A}$  is the action taken at time  $t$ .

$r_t = R(s_t, a_t) \in \mathbb{R}$  is the reward at time  $t$

$P_T(s_{t+1}|s_t, a_t)$  is the probability of  $s_{t+1}$  given  $s_t$  and  $a_t$ .

The function  $R(s, a)$  can allow for a cost of the action  $a$ .

## Optimizing Reward

In RL we maximize reward rather than minimize loss.

$$\pi^* = \operatorname{argmax}_{\pi} R(\pi)$$

$$R(\pi) = E_{\pi} \sum_{t=0}^T r_t \quad \text{episodic reward (go)}$$

$$\text{or } E_{\pi} \sum_{t=0}^{\infty} \gamma^t r_t \quad \text{discounted reward (financial planning)}$$

$$\text{or } \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^T r_t \quad \text{asymptotic average reward (driving)}$$



# The Value Function

For discounted reward:

$$V^\pi(s) = E_\pi \sum_t \gamma^t r_t \mid \pi, s_0 = s$$

$$V^*(s) = \sup_\pi V^\pi(s)$$

$$\pi^*(a|s) = \operatorname{argmax}_a R(s, a) + \gamma E_{s' \sim P_T(s'|s, a)} V^*(s')$$

$$V^*(s) = \max_a R(s, a) + \gamma E_{s' \sim P_T(\cdot|s, a)} V^*(s')$$

## Value Iteration

Suppose the state space and action space are finite.

In that case we can do value iteration.

$$V_0(s) = 0$$

$$V_{i+1}(s) = \max_a R(s, a) + \gamma E_{s' \sim P_T(\cdot|s,a)} V_i(s')$$

If all rewards are non-negative then

$$V_{i+1}(s) \geq V_i(s) \quad V_i(s) \leq V^*(s) \quad \text{so } \lim_{i \rightarrow \infty} V_i(s) \text{ exists}$$

## Value Iteration

Theorem: **For discounted reward**

$$V_{\infty}(s) \doteq \lim_{i \rightarrow \infty} V_i(s) = V^*(s)$$

## Proof

$$\begin{aligned}\Delta &\doteq \max_s V^*(s) - V_\infty(s) \\&= \max_s \left( \max_a R(s, a) + E_{s'|a} \gamma V^*(s') \right. \\&\quad \left. - \max_a R(s, a) + E_{s'|a} \gamma V_\infty(s') \right) \\&\leq \max_s \max_a \left( \begin{aligned} &R(s, a) + E_{s'|a} \gamma V^*(s') \\ &- R(s, a) + E_{s'|a} \gamma V_\infty(s') \end{aligned} \right) \\&= \max_s \max_a E_{s'|a} \gamma (V^*(s') - V_\infty(s)) \\&\leq \gamma \Delta\end{aligned}$$

## The $Q$ Function

For discounted reward:

$$Q^\pi(s, a) = E_\pi \sum_t \gamma^t r_t \mid \pi, s_0 = s, a_0 = a$$

$$Q^*(s, a) = \sup_{\pi} Q^\pi(s, a)$$

$$\pi^*(a|s) = \operatorname{argmax}_a Q^*(s, a)$$

$$Q^*(s, a) = R(s, a) + \gamma E_{s' \sim P_T(\cdot|s, a)} \max_{a'} Q^*(s', a')$$

## $Q$ Function Iteration

It is possible to define  $Q$ -iteration by analogy with value iteration, but this is generally not discussed.

Value iteration is typically done for finite state spaces. Let  $S$  be the number of states and  $A$  be the number of actions.

One update of a  $Q$  table takes  $O(S^2 A^2)$  time while one update of value iteration is  $O(S^2 A)$ .

## **$Q$ -Learning**

When learning by updating the  $Q$  function we typically assume a parameterized  $Q$  function  $Q_\Phi(s, a)$ .

**Bellman Error:**

$$\text{Bell}_\Phi(s, a) \doteq \left( Q_\Phi(s, a) - \left( R(s, a) + \gamma \mathbb{E}_{s' \sim P_T(s'|s, a)} \max_{a'} Q_\Phi(s', a') \right) \right)^2$$

**Theorem:** If  $\text{Bell}_\Phi(s, a) = 0$  for all  $(s, a)$  then the induced policy is optimal.

**Algorithm:** Generate pairs  $(s, a)$  from the policy  $\arg\max_a Q_\Phi(s_t, a)$  and repeat

$$\Phi \leftarrow \Phi - \eta \nabla_\Phi \text{Bell}_\Phi(s, a)$$

## Issues with $Q$ -Learning

Problem 1: Nearby states in the same run are highly correlated. This increases the variance of the cumulative gradient updates.

Problem 2: SGD on Bellman error tends to be unstable. Failure of  $Q_\Phi$  to model unused actions leads to policy change (exploration). But this causes  $Q_\Phi$  to stop modeling the previous actions which causes the policy to change back ...

To address these problems we can use a **replay buffer**.



## Using a Replay Buffer

We use a replay buffer of tuples  $(s_t, a_t, r_t, s_{t+1})$ .

Repeat:

1. Run the policy  $\operatorname{argmax}_a Q_\Phi(s, a)$  to add tuples to the replay buffer. Remove oldest tuples to maintain a maximum buffer size.
2.  $\Psi = \Phi$
3. for  $N$  times select a random element of the replay buffer and do

$$\Phi \leftarrow \Phi - \eta \nabla_\Phi (Q_\Phi(s_t, a_t) - (r_t + \gamma \max_a Q_\Psi(s_{t+1}, a)))^2$$

## Replay is Off-Policy

Note that the replay buffer is from a **mixture of policies** and is **off-policy** for  $\operatorname{argmax}_a Q_{\Phi}(s, a)$ . This seems to be important for stability.

This seems related to the issue of stochastic vs. deterministic policies. More on this later.

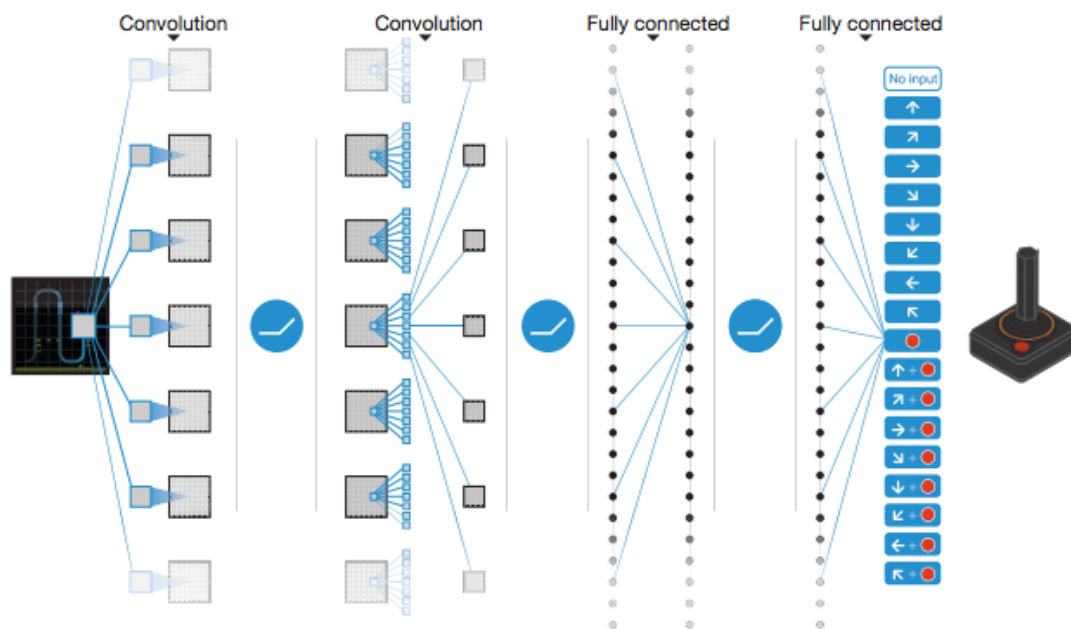
## Multi-Step $Q$ -learning

$$\Phi \leftarrow \sum_t \nabla_{\Phi} \left( Q_{\Phi}(s_t, a_t) - \sum_{\delta=0}^D \gamma^{\delta} r_{(t+\delta)} \right)^2$$

# Human-level control through deep RL (DQN)

Mnih et al., Nature, 2015. (Deep Mind)

We consider a CNN  $Q_{\Phi}(s, a)$ .



## Watch The Video

<https://www.youtube.com/watch?v=V1eYniJ0Rnk>

## Asynchronous Q-Learning (Simplified)

No replay buffer. Many asynchronous threads each repeating:

$\tilde{\Phi} = \Phi$  (retrieve  $\Phi$ )

using policy  $\operatorname{argmax}_a Q_{\tilde{\Phi}}(s, a)$  compute

$$s_t, a_t, r_t, \dots, s_{t+K}, a_{t+K}, r_{t+K}$$

$$\Phi \mathrel{+=} \eta \sum_{i=t}^{t+K-D} \nabla_{\tilde{\Phi}} \left( Q_{\tilde{\Phi}}(s_i, a_i) - \sum_{\delta=0}^D \gamma^{\delta} r_{i+\delta} \right)^2 \text{ (update } \Phi)$$

# The REINFORCE Algorithm

Williams, 1992

## REINFORCE is a Policy Gradient Algorithm

We assume a parameterized policy  $\pi_{\Phi}(a|s)$ .

$\pi_{\Phi}(a|s)$  is normalized while  $Q_{\Phi}(s, a)$  is not.



## Policy Gradient Theorem (Episodic Case)

$$\Phi^* = \operatorname{argmax}_{\Phi} E_{\pi_{\Phi}} R$$

$$\nabla_{\Phi} E_{\pi_{\Phi}} R = \sum_{s_0, a_0, s_1, a_1, \dots, s_T, a_T} \nabla_{\Phi} P(s_0, a_0, s_1, a_1, \dots, s_T, a_T) R$$

$$\begin{aligned} \nabla_{\Phi} P(\dots) R &= P(S_0) \nabla_{\Phi} \pi(a_0) P(s_1) \pi(a_1) \cdots P(s_T) \pi(a_T) R \\ &\quad + P(S_0) \pi(a_0) P(s_1) \nabla_{\Phi} \pi(a_1) \cdots P(s_T) \pi(a_T) R \\ &\quad \vdots \\ &\quad + P(S_0) \pi(a_0) P(s_1) \pi(a_1) \cdots P(s_T) \nabla_{\Phi} \pi(a_T) R \end{aligned}$$

$$= P(\dots) \left( \sum_t \frac{\nabla_{\Phi} \pi_{\Phi}(a_t)}{\pi_{\Phi}(a_t)} \right) R$$

## Policy Gradient Theorem (Episodic Case)

$$\nabla_{\Phi} P(\dots)R = P(\dots) \left( \sum_t \frac{\nabla_{\Phi} \pi_{\Phi}(a_t|s_t)}{\pi_{\Phi}(a_t|s_t)} \right) R$$

$$\nabla_{\Phi} E_{\pi_{\Phi}} R = E_{\pi_{\Phi}} \left( \sum_t \nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t) \right) R$$

## Policy Gradient Theorem

$$\begin{aligned} & \nabla_{\Phi} E_{\pi_{\Phi}} R \\ &= E_{\pi_{\Phi}} \left( \sum_t \nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t) \right) R \\ &= E_{\pi_{\Phi}} \left( \sum_t \nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t) \right) \left( \sum_t r_t \right) \\ &= E_{\pi_{\Phi}} \sum_{t,t'} \nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t) r_{t'} \end{aligned}$$

## Policy Gradient Theorem

$$\nabla_{\Phi} E_{\pi_{\Phi}} R = \sum_{t,t'} E_{s_t,a_t,r_{t'}} \nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t) r_{t'}$$

For  $t' < t$  we have

$$\begin{aligned} E_{r_{t'},s_t,a_t} r_{t'} \nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t) &= E_{r_{t'},s_t} r_{t'} \sum_{a_t} \pi_{\Phi}(a_t|s_t) \nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t) \\ &= E_{r_{t'},s_t} r_{t'} \sum_{a_t} \nabla_{\Phi} \pi_{\Phi}(a_t|s_t) \\ &= E_{r_{t'},s_t} r_{t'} \nabla_{\Phi} \sum_{a_t} \pi_{\Phi}(a_t|s_t) \\ &= 0 \end{aligned}$$

# REINFORCE

$$\nabla_{\Phi} E_{\pi_{\Phi}} R = E_{\pi_{\Phi}} \sum_{t, t' \geq t} (\nabla_{\Phi} \ln \pi_{\Phi}(a_t | s_t)) r_{t'}$$

Sampling runs and computing the above sum over  $t$  is Williams' REINFORCE algorithm.

# Optimizing Discrete Decisions with Non-Differentiable Loss

The REINFORCE algorithm is used generally for non-differentiable loss functions.

For example error rate and BLEU score are non-differentiable — they are defined on the result of discrete decisions.

$$\Phi^* = \operatorname{argmax}_{\Phi} E_{w_1, \dots, w_n \sim P_{\Phi}} \text{BLEU}$$

## The Variance Issue

REINFORCE typically suffers from high variance of the gradient samples requiring very small learning rates and very long convergence times.

$$\nabla_{\Phi} E_{\pi_{\Phi}} R = E_{\pi_{\Phi}} \sum_{t, t' \geq t} (\nabla_{\Phi} \ln \pi_{\Phi}(a_t | s_t)) r_{t'}$$

We have to consider

- the variation over the choice of  $a_t$
- the variation over  $r_{t'}$  given  $a_t$

## The Variance Issue

$$\nabla_{\Phi} E_{\pi_{\Phi}} R = E_{\pi_{\Phi}} \sum_{t, t' \geq t} \nabla_{\Phi} \ln \pi_{\Phi}(a_t | s_t) \textcolor{red}{r}_{t'}$$

We can reduce the variation over  $r_{t'}$  given  $s_t$  and  $a_t$  by shifting to the following.

$$\nabla_{\Phi} E_{\pi_{\Phi}} R = E_{\pi_{\Phi}} \sum_{t, t'} \nabla_{\Phi} \ln \pi_{\Phi}(a_t | s_t) \textcolor{red}{E}_{r_{t'} | s_t, a_t} \textcolor{red}{r}_{t'}$$

Before saying how this can be computationally approximated, we state the above expression somewhat differently.



## Policy Gradient Theorem

$$\nabla_{\Phi} E_{\pi_{\Phi}} R = E_{\pi_{\Phi}} \sum_{t, t' \geq t} (\nabla_{\Phi} \ln \pi_{\Phi}(a_t | s_t)) \textcolor{red}{E}_{s_{t'}, a_{t'} | s_t, a_t} \textcolor{red}{r}_{t'}$$

$$= E_{\pi_{\Phi}} \sum_t (\nabla_{\Phi} \ln \pi_{\Phi}(a_t | s_t)) \textcolor{red}{Q}^{\pi_{\Phi}}(s_t, a_t)$$

$$Q^{\pi}(s, a) = E_{\pi} \sum_t r_t \mid s_0 = s, a_0 = a$$

## Policy Gradient Theorem

$$\nabla_{\Phi} E_{\pi_{\Phi}} R = E_{\pi_{\Phi}} \sum_t (\nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t)) Q^{\pi_{\Phi}}(s_t, a_t)$$

**The point** is that we can now approximate  $Q^{\pi_{\Phi}}$  with neural network  $Q_{\Phi}$  where the networks  $\pi_{\Phi}$  and  $Q_{\Phi}$  can use different, perhaps overlapping, parts of  $\Phi$ .

We reduced the variance at the cost of approximating the expected future reward.

# The Actor-Critic Algorithm

$$\nabla_{\Phi} E_{\pi_{\Phi}} R \approx E_{\pi_{\Phi}} \sum_t (\nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t)) Q_{\Phi}(s_t, a_t)$$

$\pi_{\Phi}$  is the “actor” and  $Q_{\Phi}$  is the “critic”

## The Actor-Critic Algorithm

$$\nabla_{\Phi} E_{\pi_{\Phi}} R \approx E_{\pi_{\Phi}} \sum_t (\nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t)) Q_{\Phi}(s_t, a_t)$$

We can sample an episode and then do

$$\begin{aligned} \Phi & += \sum_t \eta_1 (\nabla_{\Phi} \ln \pi_{\Phi}(a_i|s_i)) Q_{\Phi}(s_t, a_t) \\ \Phi & -= \sum_t \eta_2 \nabla_{\Phi} \left( Q_{\Phi}(s_t, a_t) - \sum_{t' \geq t} r_{t'} \right)^2 \end{aligned}$$

The two updates typically apply to different (but perhaps overlapping) subsets of the parameters  $\Phi$ .

## Variance from the Choice of $a_t$

To address the variance due to the choice of  $a_t$  we first make the following observation for any function  $V(s)$  of states.

$$\begin{aligned} & E_{s_t, a_t} (\nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t)) V(s_t) \\ &= E_{s_t} \sum_{a_t} (\pi_{\Phi}(a_t|s_t) \nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t)) V(s_t) \\ &= E_{s_t} \sum_{a_t} (\nabla_{\Phi} \pi_{\Phi}(a_t|s_t)) V(s_t) \\ &= E_{s_t} V(s_t) \sum_{a_t} (\nabla_{\Phi} \pi_{\Phi}(a_t|s_t)) \end{aligned}$$

## Variance from the Choice of $a_t$

$$\begin{aligned} & E_{s_t, a_t} \left( \nabla_{\Phi} \ln \pi_{\Phi}(a_t | s_t) \right) V(s_t) \\ &= E_{s_t} V(s_t) \sum_{a_t} \left( \nabla_{\Phi} \pi_{\Phi}(a_t | s_t) \right) \\ &= E_{s_t} V(s_t) \nabla_{\Phi} \sum_{a_t} \pi_{\Phi}(a_t | s_t) \\ &= 0 \end{aligned}$$

## Variance from the Choice of $a_t$

$$\begin{aligned} & E_{s_t, a_t} \sum_t (\nabla_{\Phi} \ln \pi_{\Phi}(a_t | s_t)) (Q^{\pi_{\Phi}}(s_t, a_t) - V(s_t)) \\ &= E_{s_t, a_t} \sum_t (\nabla_{\Phi} \ln \pi_{\Phi}(a_t | s_t)) Q^{\pi_{\Phi}}(s_t, a_t) \\ &= \nabla_{\Phi} E_{\pi_{\Phi}} R \end{aligned}$$

## Variance from the Choice of $a_t$

In particular we have

$$\nabla_{\Phi} E_{\pi_{\Phi}} R = E_{\pi_{\Phi}} \sum_t (\nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t)) (Q^{\pi_{\Phi}}(s_t, a_t) - V^{\pi_{\Phi}}(s_t))$$

$$V^{\pi_{\Phi}}(s) = E_{a \sim \pi_{\Phi}(a|s)} Q^{\pi_{\Phi}}(s, a)$$

$Q^{\pi_{\Phi}}(s, a) - V^{\pi_{\Phi}}(s)$  is the “advantage” of deterministically using  $a$  rather than sampling an action.

Nondeterminism of the policy  $\pi_{\Phi}$  provides for exploration.



## Advantage-Actor-Critic Algorithm

$$\nabla_{\Phi} E_{\pi_{\Phi}} R \approx E_{\pi_{\Phi}} \sum_t (\nabla_{\Phi} \ln \pi_{\Phi}(a_t|s_t)) (Q_{\Phi}(s_t, a_t) - V_{\Phi}(s_t))$$

We can sample an episode and then do

$$\begin{aligned}\Phi & += \sum_t \eta_1 (\nabla_{\Phi} \ln \pi_{\Phi}(a_i|s_i)) (Q_{\Phi}(s_t, a_t) - V_{\Phi}(s_t)) \\ \Phi & -= \sum_t \eta_2 \nabla_{\Phi} \left( Q_{\Phi}(s_t, a_t) - \sum_{t' \geq t} r_{t'} \right)^2 \\ \Phi & -= \sum_t \eta_3 \nabla_{\Phi} (V_{\Phi}(s_t) - Q_{\Phi}(s_t, a))^2\end{aligned}$$

## Asynchronous Methods for Deep RL (A3C)

Mnih et al., Arxiv, 2016 (Deep Mind)

$\tilde{\Phi} = \Phi$  (retrieve global  $\Phi$ )

using policy  $\pi_{\tilde{\Phi}}$  compute  $s_t, a_t, r_t, \dots, s_{t+K}, a_{t+K}, r_{t+K}$

$$R_i = \sum_{\delta=0}^D \gamma^{i+\delta} r_{(i+\delta)}$$

$$\Phi \ += \ \eta \sum_{i=t}^{t+K-D} \left( \nabla_{\tilde{\Phi}} \ln \pi_{\tilde{\Phi}}(a_i | s_i) \right) \left( R_i - V_{\tilde{\Phi}}(s_i) \right)$$

$$\Phi \ -= \ \eta \sum_{i=t}^{t+K-D} \nabla_{\tilde{\Phi}} \left( V_{\tilde{\Phi}}(s_i) - R_i \right)^2$$

## Issue: Policies must be Exploratory

The optimal policy is deterministic —  $a(s) = \operatorname{argmax}_a Q(s, a)$ .

However, a deterministic policy never samples alternative actions.

Typically one forces a random action some small fraction of the time.

## **Issue: Discounted Reward**

DQN and A3C use discounted reward on episodic or long term problems.

Presumably this is because actions have near term consequences.

This should be properly handled in the mathematics.

## Observation: Continuous Actions are Differentiable

In problems like controlling an inverted pendulum, or robot control generally, a continuous loss can be defined and the gradient of loss of with respect to a deterministic policy exists.

## More Videos

<https://www.youtube.com/watch?v=g59nSURxYgk>

<https://www.youtube.com/watch?v=rAai4QzcYbs>

**END**