TTIC 31230, Fundamentals of Deep Learning

David McAllester, April 2017

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.

$$\pi(a_t|s_t) \quad \text{probability of action } a_t \text{ in state } s_t$$

Imitation Learning

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

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

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

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 dangersous. "Don't try this at home".

Also, imitating experts will never exceed expert performance (consider AlphaZero).

Markov Decision Processes (MDPs)

For an RL problem we work with an action policy π

s_t is the state at time t

r_t is the reward at time t

a_t is the action taken at time t.

$$\begin{array}{rl} r_t = R(s_t, a_t) & \text{reward at time } t \\ P_T(s_{t+1}|s_t, a_t) & \text{state transition probability} \end{array}$$

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

This Data defines a **Markov Decision Process** or **MDP**.

Optimizing Reward

In RL we maximize reward rather than minimize loss.

$$\pi^* = \underset{\pi}{\mathrm{argmax}} \;\; R(\pi)$$

$$\begin{array}{lll} R(\pi) & = \mathrm{E}\left[\sum_{t=0}^{T} r_t\right] & \text{episodic reward (go)} \\ & \text{or } \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] & \text{discounted reward (financial planning)} \\ & \text{or } \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} r_t & \text{asymptotic average reward (driving)} \end{array}$$

The Value Function

For discounted reward:

$$V^\pi(s) \; = \; \mathrm{E}\left[\sum_t \gamma^t r_t \;\mid \pi,\; s_0 = s\right]$$

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

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

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

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

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

Value Iteration

Theorem: **For discounted reward**

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

Proof

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

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-Learning

We will assume a parameterized Q function $Q_\Phi(s, a)$.

Bellman Error:

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

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

Algorithm: Generate pairs (s, a) from the policy $\mathrm{argmax}_a \; Q_\Phi(s_t, a)$ and repeat

$$\Phi \mathrel{-}= \eta \nabla_\Phi \; \mathrm{Bell}_\Phi(s, a)$$

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_Φ to model unused actions leads to policy change (exploration). But this causes Q_Φ 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 $\mathrm{argmax}_a\, Q_\Phi(s, a)$ 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

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

Replay is Off-Policy

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

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

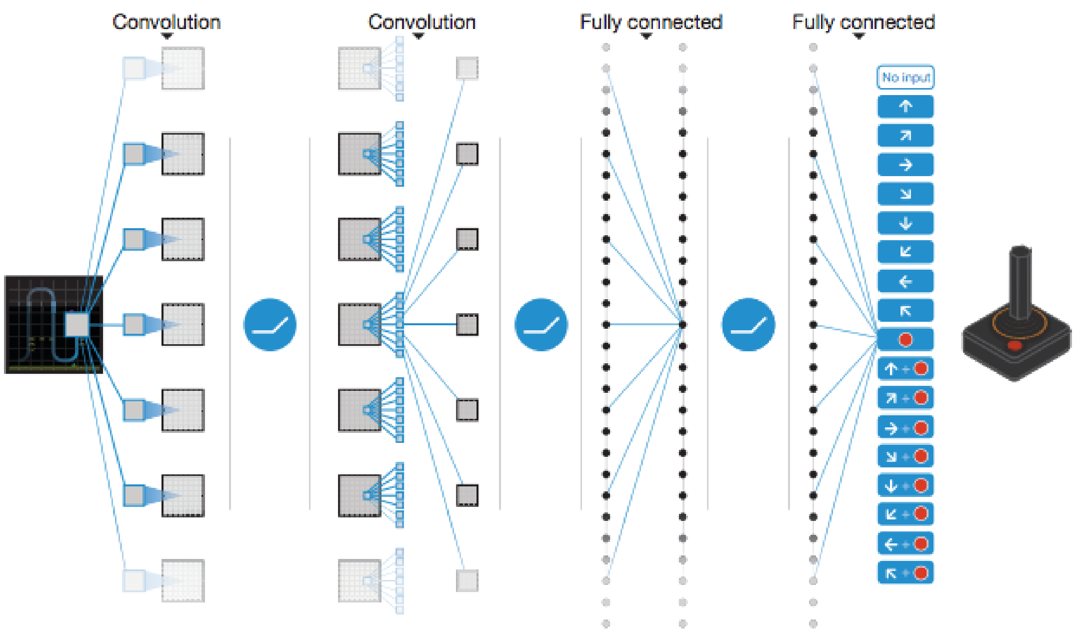
Multi-Step Q-learning

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

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 Φ)

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

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

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

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)

$$\Phi^* = \underset{\Phi}{\operatorname{argmax}}\; E_{\pi_\Phi}\, R$$

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

$$\begin{aligned}
\nabla_\Phi\, P(\ldots)R \;=\;& P(S_0)\nabla_\Phi\; \pi(a_0)P(s_1)\pi(a_1)\cdots P(s_T)\pi(a_T)\; R \\
&+P(S_0)\pi(a_0)P(s_1)\nabla_\Phi\; \pi(a_1)\cdots P(s_T)\pi(a_T)\; R \\
&\vdots \\
&+P(S_0)\pi(a_0)P(s_1)\pi(a_1)\cdots P(s_T)\nabla_\Phi\; \pi(a_T)\; R \\
\\
=\;& P(\ldots)\left(\sum_t \frac{\nabla_\Phi\; \pi_\Phi(a_t)}{\pi_\Phi(a_t)}\right) R
\end{aligned}$$

Policy Gradient Theorem (Episodic Case)

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

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

Policy Gradient Theorem

$$\nabla_\Phi \; E_{\pi_\Phi} \; R$$

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

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

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

Policy Gradient Theorem

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

For $t' < t$ we have

$$\begin{aligned}
E_{\pi_\Phi} r_{t'} \nabla_\Phi \; \ln \pi_\Phi(a_t|s_t) &= E_{s_0,a_0,\ldots,s_t} \; r_{t'} \sum_{a_t} \pi_\Phi(a_t|s_t) \; \nabla_\Phi \; \ln \pi_\Phi(a_t|s_t) \\
&= E_{s_0,a_0,\ldots,s_t} \; r_{t'} \sum_{a_t} \nabla_\Phi \; \pi_\Phi(a_t|s_t) \\
&= E_{s_0,a_0,\ldots,s_t} \; r_{t'} \; \nabla_\Phi \sum_{a_t} \pi_\Phi(a_t|s_t) \\
&= 0
\end{aligned}$$

REINFORCE

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

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.

$$\Phi^* = \operatorname*{argmax}_{\Phi} \; E_{w_1,\ldots,w_n \sim P_\Phi} \; \mathrm{BLEU}$$

The Variance Issue

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

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

We have to consider

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

The Variance Issue

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

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

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

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

Policy Gradient Theorem

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

$$=\; E_{\pi_\Phi} \sum_t \left(\nabla_\Phi \; \ln \pi_\Phi(a_t|s_t)\right) \quad Q^{\pi_\Phi}(s_t, a_t)$$

$$Q^\pi(s,a) = E_\pi \; \textstyle\sum_t \; r_t \;|\; s_0 = s,\;\; a_0 = a$$

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^{π_Φ} with neural network Q_Φ where the networks π_Φ and Q_Φ can use different, perhaps overlapping, parts of Φ.

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

The Actor-Critic Algorithm

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

π_Φ is the "actor" and Q_Φ is the "critic"

The Actor-Critic Algorithm

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

We can sample an episode and then do

$$\Phi \;\; \texttt{+=} \; \sum_t \eta_1 \, (\nabla_\Phi \; \ln \pi_\Phi(a_i|s_i)) \, Q_\Phi(s_t, a_t)$$

$$\Phi \;\; \texttt{-=} \; \sum_t \eta_2 \; \nabla_\Phi \; \left(Q_\Phi(s_t, a_t) - \sum_{t' \geq t} r_{t'} \right)^2$$

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

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.

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

Variance from the Choice of a_t

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

Variance from the Choice of a_t

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

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

$$= \; \nabla_\Phi \; E_{\pi_\Phi} \; R$$

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 π_Φ 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

$$\Phi \;\texttt{+=}\; \sum_t \eta_1 \, (\nabla_\Phi \; \ln \pi_\Phi(a_i|s_i)) \, Q_\Phi(s_t, a_t)$$

$$\Phi \;\texttt{-=}\; \sum_t \eta_2 \; \nabla_\Phi \; \left(Q_\Phi(s_t, a_t) - \sum\nolimits_{t' \geq t} r_{t'}\right)^2$$

$$\Phi \;\texttt{-=}\; \sum_t \eta_3 \; \nabla_\Phi \; (V_\Phi(s_t) - Q_\Phi(s_t, a))^2$$

Asynchronous Methods for Deep RL (A3C)

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

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

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

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

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

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

Issue: Policies must be Exploratory

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

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