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.

$\pi(a_t|s_t)$ 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 $\pi_\Theta(s)$.

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

This is just cross-entropy loss.

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 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.

$$r_t = R(s_t, a_t) \quad \text{reward at time } t$$

$$T(s_{t+1}|s_t, a_t) \quad \text{state transition probability}$$

The state space, action space, R and T define a **Markov Decision Process** or **MDP**.

Optimizing Reward

In RL we maximize reward rather than minimize loss.

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

$$R(\pi) = \mathrm{E}\left[\sum_{t=0}^{T} r_t\right] \qquad \text{episodic reward (go)}$$

$$\text{or } \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \qquad \text{discounted reward}$$

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

The Value Function

For discounted reward:

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

$$V^{*}(s) = \max_{\pi} V^{\pi}(s)$$

$$\pi^{*}(a|s) = \operatorname*{argmax}_{a} E_{s' \sim T(s'|s,a)} V^{*}(s')$$

$$V^{*}(s) = \max_{a} R(s,a) + \gamma E_{s' \sim T(\cdot|s,a)} V^{*}(s')$$

Value Iteration

Suppose the state space and action space are finite.

In that case we can do value iteration.

$$V_0(s) = 0$$

$$V_{i+1}(s) = \max_a \; R(s,a) + \gamma E_{s' \sim T(\cdot|s,a)} \, V_i(s')$$

If all rewards are non-negative then

$$V_{i+1}(s) \geq V_i(s) \quad V_i(s) \leq V^*(s)$$

Value Iteration

Theorem: **For discounted reward**

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

Proof:

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

The Q Function

For discounted reward:

$$Q^{\pi}(s,a) = \mathrm{E}\left[\sum_t \gamma^t r_t \mid \pi,\ s_0 = s,\ a_0 = a\right]$$

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

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

$$Q^{*}(s,a) = R(s,t) + \gamma E_{s' \sim T(\cdot|s,a)} \max_{a'} Q^{*}(s',a')$$

Q-Learning

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

Define the **Bellman error**

$$\left(Q_\Theta(s,a) - \left(R(s,a) + \gamma\ E_{s' \sim S(\cdot|s,a)}\ \max_{a'}\ Q_\Theta(s', a')\right)\right)^2$$

Algorithm: run the policy $\text{argmax}_a\ Q_\Theta(s_t, a)$ and repeat

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

The Theorem

Theorem: **For discounted reward**, if the Bellman error is zero then the induced policy is optimal.

This Q-learning theorem is similar in strength to the contrastive divergence theorem (gradient is zero at solution).

Weaker than the pseudo-likelihood theorem (optimization problem is correct assuming universal expressiveness).

Issues with Q-Learning

Problem 1: Nearby states in the same run are highly correlated.

Problem 2: SGD on Bellman error tends to be unstable. (Weakness of the Q-learning theorem?)

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

$$\Theta \text{ -= } \eta \nabla_\Theta \left(Q_\Theta(s_t, a_t) - (r_t + \gamma \operatorname*{argmax}_a Q_\Psi(s_{t+1}, a)) \right)^2$$

Replay is Off-Policy

Note that the replay buffer is (slightly) **off-policy**. This seems to be important for stability.

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

Multi-Step Q-learning

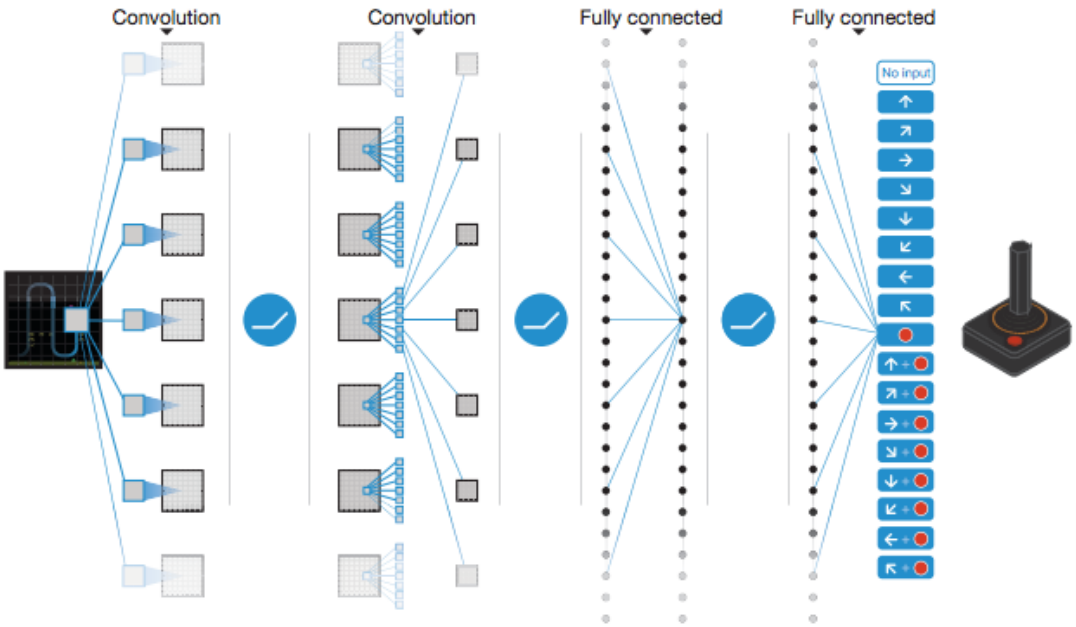
$$\Theta \mathrel{-}= \sum_t \nabla_\Theta \left(Q_\Theta(s_t, a_t) - \sum_{\delta=0}^{K} \gamma^\delta r_{(t+\delta)} \right)^2$$

Deep Q-Learning (DQN)

Human-level control through deep reinforcement learning, Mnih et al., Nature, 2015. (Deep Mind)

Deep Q-Networks

We consider a **Deep Q-network** — a deep network with parameters Θ and computing a Q-value $Q_\Theta(s, a)$.



Watch The Video

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

Asynchronous Q-Learning (Simplified)

No replay buffer. Many aynchronous threads each repeating:

$\tilde{\Theta} = \Theta$ (retrieve global Θ)

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

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

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

Update global Θ:

$$\Theta \texttt{ -= } \eta \sum_{i=t}^{t+K} \nabla_{\tilde{\Theta}} \left(Q_{\tilde{\Theta}}(s_i, a_i) - R_i \right)^2$$

Policy Gradient

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

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

$$\Phi \mathrel{+}= \eta \nabla_\Phi \, R(\Phi)$$

Policy Gradient Theorem (Episodic Case)

$$\mathrm{E}\left[R \mid \Phi\right] = \sum_{s_0,a_0,s_1,a_1,\ldots,s_T,a_T} P(s_0, a_0, s_1, a_1, \ldots, s_T, a_T)\ R$$

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

$$= P(\ldots)\left(\sum_i \frac{\nabla_\Phi\ \pi_\Phi(a_i)}{\pi_\Phi(a_i)}\right) R$$

Policy Gradient Theorem (Episodic Case)

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

$$
= P(\ldots)\left(\sum_i \frac{\nabla_\Phi \, \pi_\Phi(a_i)}{\pi_\Phi(a_i)}\right) R
$$

$$
\nabla_\Phi \, \mathrm{E}\left[R \mid \Phi\right] = \mathrm{E}\left[R \sum_t \nabla_\Phi \, \ln \pi_\Phi(a_t|s_t)\right]
$$

Policy Gradient Theorem

$$\nabla_\Phi \, \mathrm{E}\left[R \mid \Phi\right]$$

$$= \mathrm{E}\left[R \sum_t \nabla_\Phi \, \ln \pi_\Phi(a_t|s_t)\right]$$

$$= E \sum_{t_1,t_2} r_{t_1} \nabla_\Phi \, \ln \pi_\Phi(a_{t_2}|s_{t_2})$$

$$= E \sum_{t_1<t_2} r_{t_1} \nabla_\Phi \, \ln \pi_\Phi(a_{t_2}|s_{t_2}) + E \sum_{t_1 \geq t_2} r_{t_1} \nabla_\Phi \, \ln \pi_\Phi(a_{t_1}|s_{t_2})$$

Policy Gradient Theorem

$$\nabla_\Phi \, \mathrm{E}\left[R \mid \Phi\right]$$

$$= E \sum_{t_1 < t_2} r_{t_1} \nabla_\Phi \, \ln \pi_\Phi(a_{t_2}|s_{t_2}) + E \sum_{t_1 \geq t_2} r_{t_1} \nabla_\Phi \, \ln \pi_\Phi(a_{t_1}|s_{t_2})$$

$$= \sum_{t_1 < t_2} E_{s_{t_1}} \left(E \, r_{t_1}|s_{t_1}\right) E_{s_{t_2}|s_{t_1}} \sum_{a_{t_2}} \pi_\Phi(a_{t_2}|s_{t_2}) \nabla_\Phi \, \ln \pi_\Phi(a_{t_2}|s_{t_2}) + \cdots$$

$$\sum_{t_1 < t_2} E_{s_{t_1}} \left(E \, r_{t_1}|s_{t_1}\right) E_{s_{t_2}|s_{t_1}} \sum_{a_{t_2}} \nabla_\Phi \, \pi_\Phi(a_{t_2}|s_{t_2}) \quad = 0$$

Policy Gradient Theorem

$$\nabla_\Phi \, \mathrm{E}\left[R \mid \Phi\right]$$

$$= E \sum_{t_1 \leq t_2} \nabla_\Phi \, \ln \pi_\Phi(a_{t_1}|s_1) \, r_{t_2}$$

$$= E \sum_{t} \nabla_\Phi \, \ln \pi_\Phi(a_t|s_t) \, R_{\geq t}$$

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

Optimizing Discrete Decisions

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.

Policy Gradient Theorem

$$\nabla_\Phi \, \mathrm{E}\left[R \mid \Phi\right] = E \sum_t \nabla_\Phi \, \ln \pi_\Phi(a_t|s_t) \, R_{\geq t}$$

$$= \sum_s \rho(s) \sum_a \left(\nabla_\Phi \, \pi_\Phi(a|s)\right) Q^{\pi_\Phi}(s,a)$$

$\rho(s)$ is the expected number of occurances of s

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

The Actor-Critic Algorithm

$$\begin{aligned}\nabla_\Phi \, \mathrm{E}\left[R \mid \Phi\right] &= E \sum_t \nabla_\Phi \, \ln \pi_\Phi(a_t|s_t) \, Q^{\pi_\Phi}(s_t, a_t) \\ &\approx E \sum_t \nabla_\Phi \, \ln \pi_\Phi(a_t|s_t) \, Q_\Psi(s_t, a_t)\end{aligned}$$

We can reduce variance by using an estimator $Q_\Psi(s, a)$ for $Q^{\pi_\Phi}(s, a)$.

π_Φ is the "actor" and Q_Ψ is the "critic".

The Actor-Critic Algorithm

$$\nabla_\Phi \, \mathrm{E}\left[R \mid \Phi\right] \approx E \sum_t \nabla_\Phi \, \ln \pi_\Phi(a_t|s_t) \, Q_\Psi(s_t, a_t)$$

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

$$\Psi \; \texttt{-=} \; \eta_2 \sum_t \nabla_{\tilde{\Psi}} \left(Q_\Psi(s_t, a_t) - R_{\geq t}\right)^2$$

π_Φ is the "actor" and Q_Ψ is the "critic".

Advantage-Actor-Critic Theorem

$$\nabla_\Phi R(\Phi) = \sum_s \rho(s) \sum_a \left(\nabla_\Phi \, \pi_\Phi(a|s)\right) \left(Q^{\pi_\Phi}(s,a) - V^{\pi_\Phi}(s)\right)$$

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

Asynchronous Advantage Actor-Critic (A3C)

Asynchronous Methods for Deep Reinforcement Learning, Mnih et al., Arxiv, 2016 (Deep Mind)

Asynchronous Advantage Actor-Critic (A3C)

$\tilde{\Phi} = \Phi; \tilde{\Psi} = \Psi$ (retrieve global Φ and Ψ)

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

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

Update global Φ and Ψ

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

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

Issue: Policies must be Exploratory

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

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