Reinforcement Learning for NLP

Advanced Machine Learning for NLP
Jordan Boyd-Graber

Adapted from slides by David Silver, Pieter Abbeel, and John Schulman
• I used to say that RL wasn’t used in NLP . . .
• Now it’s all over the place
• Part of much of ML hype
• But what is reinforcement learning?
I used to say that RL wasn’t used in NLP . . .
Now it’s all over the place
Part of much of ML hype
But what is reinforcement learning?
  ○ RL is a general-purpose framework for decision-making
  ○ RL is for an agent with the capacity to act
  ○ Each action influences the agent’s future state
  ○ Success is measured by a scalar reward signal
  ○ Goal: select actions to maximise future reward
At each step $t$ the agent:
- Executes action $a_t$
- Receives observation $o_t$
- Receives scalar reward $r_t$

The environment:
- Receives action $a_t$
- Emits observation $o_{t+1}$
- Emits scalar reward $r_{t+1}$
### Example

<table>
<thead>
<tr>
<th>QA</th>
<th>MT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State</strong></td>
<td>Words Seen</td>
</tr>
<tr>
<td><strong>Reward</strong></td>
<td>Answer Accuracy</td>
</tr>
<tr>
<td><strong>Actions</strong></td>
<td>Answer / Wait</td>
</tr>
</tbody>
</table>
State

- Experience is a sequence of observations, actions, rewards
  \[o_1, r_1, a_1, \ldots, a_t, o_t, r_t\] (1)

- The state is a summary of experience
  \[s_t = f(o_1, r_1, a_1, \ldots, a_t, o_t, r_t)\] (2)

- In a fully observed environment
  \[s_t = f(o_t)\] (3)
What makes an RL agent?

- Policy: agent’s behaviour function
- Value function: how good is each state and/or action
- Model: agent’s representation of the environment
Policy

- A policy is the agent’s behavior
  - It is a map from state to action:
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $\pi(a | s) = p(a | s)$
Value Function

• A value function is a prediction of future reward: “How much reward will I get from action a in state s?”

• $Q$-value function gives expected total reward
  ◦ from state $s$ and action $a$
  ◦ under policy $\pi$
  ◦ with discount factor $\gamma$ (future rewards mean less than immediate)

\[
Q^\pi(s, a) = \mathbb{E} \left[ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a \right]
\]  
(4)
A Value Function is Great!

- An optimal value function is the maximum achievable value

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

\[ \text{(5)} \]

- If you know the value function, you can derive policy

\[ \pi^* = \arg\max_a Q(s, a) \]  

\[ \text{(6)} \]
Approaches to RL

Value-based RL
- Estimate the optimal value function $Q(s, a)$
- This is the maximum value achievable under any policy

Policy-based RL
- Search directly for the optimal policy $\pi^*$
- This is the policy achieving maximum future reward

Model-based RL
- Build a model of the environment
- Plan (e.g. by lookahead) using model
Deep Q Learning

- Optimal $Q$-values should obey equation
  \[
  Q^*(s, a) = \mathbb{E}_{s'} \left[ r + \gamma Q(s', a') \mid s, a \right]
  \]  
  (7)

- Treat as regression problem
- Minimize: 
  \[
  (r + \gamma \max_a Q(s', a', \vec{w}) - Q(s, a, \vec{w}))^2
  \]
- Converges to $Q$ using table lookup representation
- But diverges using neural networks due to:
  - Correlations between samples
  - Non-stationary targets
Deep RL in Atari

A diagram illustrating the concept of Deep RL in Atari with the following components:

- **State** ($s_t$) - The current state of the environment.
- **Action** ($a_t$) - The action taken by the agent.
- **Reward** ($r_t$) - The reward received by the agent for the current action in the current state.
DQN in Atari

- End-to-end learning of values $Q(s, a)$ from pixels $s$
- Input state $s$ is stack of raw pixels from last four frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step
Atari Results

A bar chart showing the performance of different Atari games. The chart is divided into categories, each representing a different game. The games are listed on the x-axis, and the performance is shown on the y-axis, with percentages indicating the success rate. The chart includes games such as Pac-Man, Space Invaders, and Ms. Pac-Man, with varying success rates in achieving human-level performance.
Policy-Based RL

- Advantages:
  - Better convergence properties
  - Effective in high-dimensional or continuous action spaces
  - Can learn stochastic policies

- Disadvantages:
  - Typically converge to a local rather than global optimum
  - Evaluating a policy is typically inefficient and high variance
Optimal Policies Sometimes Stochastic
Optimal Policies Sometimes Stochastic

(Cannot distinguish gray states)
Optimal Policies Sometimes Stochastic

Deterministic

(Cannot distinguish gray states)
Optimal Policies Sometimes Stochastic

Deterministic

(Cannot distinguish gray states)
Value-based RL learns near deterministic policy!
Optimal Policies Sometimes Stochastic

Stochastic

(Cannot distinguish gray states, so flip a coin!)
Likelihood Ratio Policy Gradient

Let $\tau$ be state-action $s_0, u_0, \ldots, s_H, u_H$. Utility of policy $\pi$ parametrized by $\theta$ is

$$U(\theta) = \mathbb{E}_{\pi_\theta, U} \left[ \sum_{t=0}^{H} R(s_t, u_t); \pi_\theta \right] = \sum_{\tau \in \mathcal{A}} P(\tau; \theta) R(\tau). \quad (8)$$

Our goal is to find $\theta$:

$$\max_{\theta} U(\theta) = \max_{\theta} \sum_{t} p(\tau; \theta) R(\tau) \quad (9)$$
Likelihood Ratio Policy Gradient

\[ \sum_{t} p(\tau; \theta) R(\tau) \]  (10)

Taking the gradient wrt \( \theta \):

(11)
Likelihood Ratio Policy Gradient

\[ \sum_t p(\tau; \theta)R(\tau) \]  

Taking the gradient wrt \( \theta \):

\[ \nabla_\theta U(\theta) = \sum_\tau R(\tau) \frac{P(\tau; \theta)}{P(\tau; \theta)} \nabla_\theta P(\tau; \theta) \]  

Move differentiation inside sum (ignore \( R(\tau) \) and then add in term that cancels out
Likelihood Ratio Policy Gradient

\[ \sum_t p(\tau; \theta) R(\tau) \]  

(10)

Taking the gradient wrt \( \theta \):

\[ \nabla_\theta U(\theta) = \sum_\tau R(\tau) \frac{P(\tau; \theta)}{P(\tau; \theta)} \nabla_\theta P(\tau; \theta) \]  

(11)

\[ = \sum_\tau P(\tau; \theta) \frac{\nabla_\theta P(\tau; \theta)}{P(\tau; \theta)} R(\tau) \]  

(12)

(13)

Move derivative over probability
Likelihood Ratio Policy Gradient

\[ \sum_t p(\tau; \theta) R(\tau) \]  

(10)

Taking the gradient wrt \( \theta \):

\[ \nabla_\theta U(\theta) = \sum_{\tau} R(\tau) \frac{P(\tau; \theta)}{P(\tau; \theta)} \nabla_\theta P(\tau; \theta) \]  

(11)

\[ = \sum_{\tau} P(\tau; \theta) \frac{\nabla_\theta P(\tau; \theta)}{P(\tau; \theta)} R(\tau) \]  

(12)

\[ = \sum_{\tau} P(\tau; \theta) \nabla_\theta [\log P(\tau; \theta)] R(\tau) \]  

(13)

Assume softmax form
Likelihood Ratio Policy Gradient

\[
\sum_t p(\tau; \theta) R(\tau)
\]  

(10)

Taking the gradient wrt \( \theta \):

\[
= \sum_{\tau} P(\tau; \theta) \nabla_{\theta} \left[ \log P(\tau; \theta) \right] R(\tau)
\]  

(11)

Approximate with empirical estimate for \( m \) sample paths from \( \pi \)

\[
\nabla_{\theta} U(\theta) \approx \frac{1}{m} \sum_{1}^{m} \nabla_{\theta} \log P(r^i; \theta) R(\tau^i)
\]  

(12)
Policy Gradient Intuition

- Increase probability of paths with positive $R$
- Decrease probability of paths with negative $R$
Extensions

• Consider baseline $b$ (e.g., path averaging)

$$\nabla_\theta U(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(r^i; \theta)(R(\tau^i) - b(\tau))$$ (13)

• Combine with value estimation (critic)
  ◦ Critic: Updates action-value function parameters
  ◦ Actor: Updates policy parameters in direction suggested by critic