Reinforcement Learning for NLP

Advanced Machine Learning for NLP

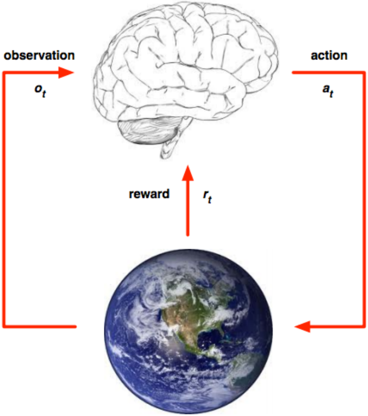
Jordan Boyd-Graber

REINFORCEMENT OVERVIEW, POLICY GRADIENT

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

	QA	MT
State	Words Seen	Foreign Words Seen
Reward	Answer Accuracy	Translation Quality
Actions	Answer / Wait	Translate / Wait

State

- Experience is a sequence of observations, actions, rewards

$$o_1, r_1, a_1, \ldots, a_{t1}, o_t, r_t \tag{1}$$

- The state is a summary of experience

$$s_t = f(o_1, r_1, a_1, \ldots, a_{t1}, o_t, r_t) \tag{2}$$

- In a fully observed environment

$$s_t = f(o_t) \tag{3}$$

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: $\pi(a \mid s) = p(a \mid s)$

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 π
  - with discount factor γ (future rewards mean less than immediate)

$$Q^{\pi}(s,a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s,a\right] \tag{4}$$

A Value Function is Great!

- An optimal value function is the maximum achievable value

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) = Q^{\pi^*}(s,a) \tag{5}$$

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

$$\pi^* = \arg\max_{a} Q(s,a) \tag{6}$$

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 π^*
- This is the policy achieving maximum future reward

Model-based RL

- Build a model of the environment
- Plan (e.g. by lookahead) using model

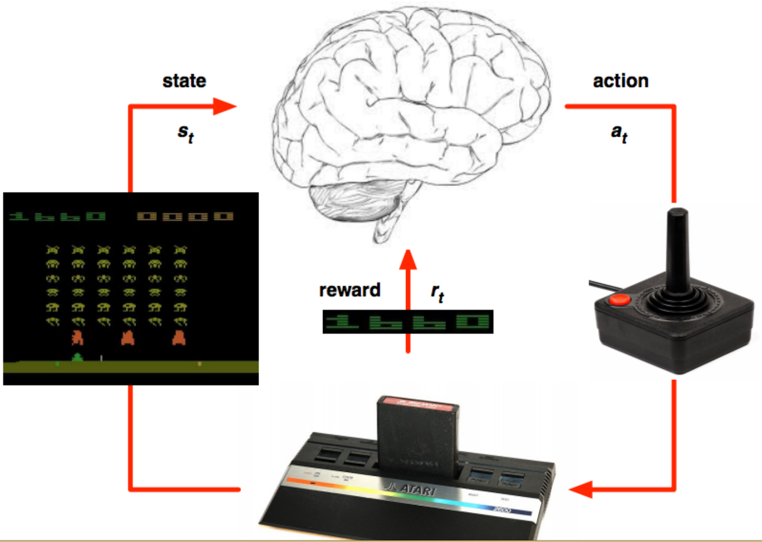
Deep Q Learning

- Optimal Q-values should obey equation

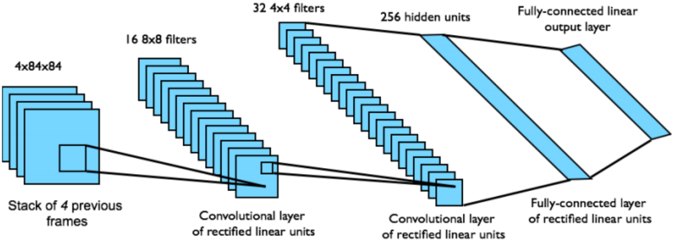
$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma Q(s',a') \,|\, s,a\right] \tag{7}$$

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

Deep RL in Atari

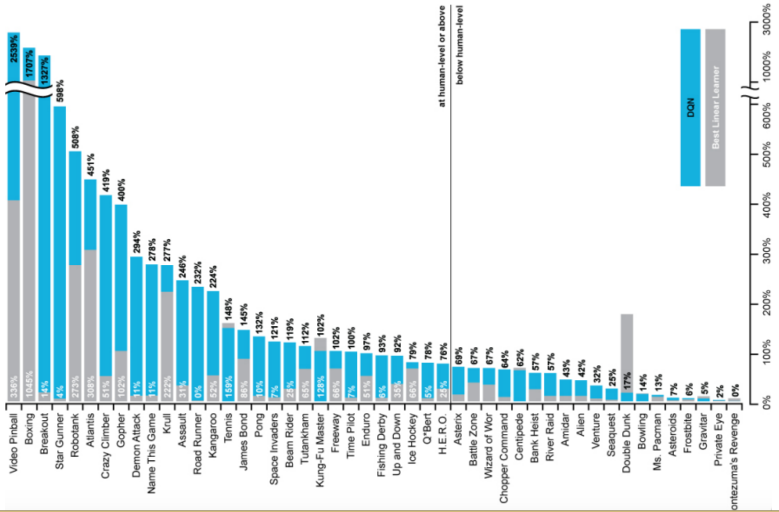


DQN in Atari



- 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

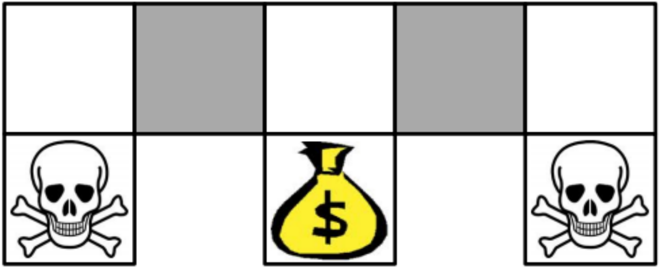


Policy-Based RL

- 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

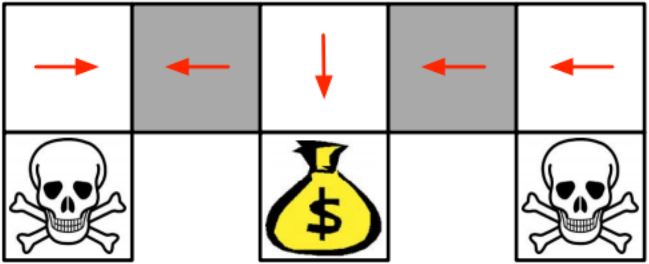
Optimal Policies Sometimes Stochastic



(Cannot distinguish gray states)

Optimal Policies Sometimes Stochastic

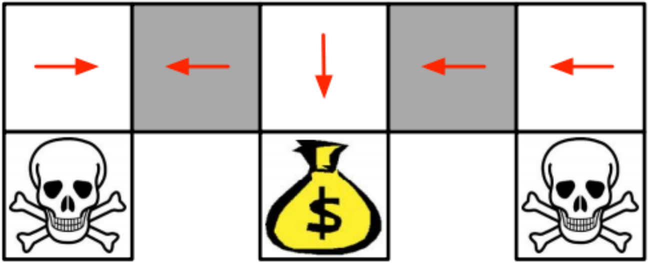
Deterministic



(Cannot distinguish gray states)

Optimal Policies Sometimes Stochastic

Deterministic

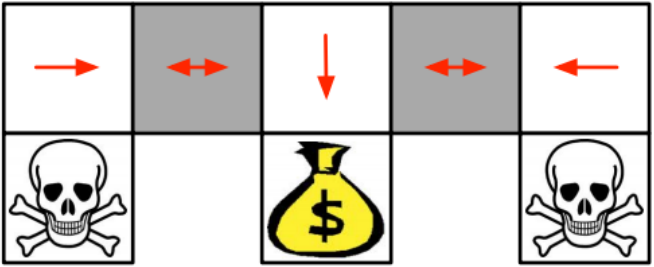


(Cannot distinguish gray states)

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 τ be state-action $s_0, u_0, \dots, s_H, u_H$. Utility of policy π parametrized by θ is

$$U(\theta) = \mathbb{E}_{\pi_\theta, U}\left[\sum_t^H R(s_t, u_t); \pi_\theta\right] = \sum_{tau} P(\tau; \theta) R(\tau). \tag{8}$$

Our goal is to find θ:

$$\max_\theta U(\theta) = \max_\theta \sum_t p(\tau; \theta) R(\tau) \tag{9}$$

Likelihood Ratio Policy Gradient

$$\sum_t p(\tau;\theta)R(\tau) \tag{10}$$

Taking the gradient wrt θ:

$$\tag{11}$$

Likelihood Ratio Policy Gradient

$$\sum_t p(\tau;\theta)R(\tau) \tag{10}$$

Taking the gradient wrt θ:

$$\nabla_\theta U(\theta) = \sum_\tau R(\tau)\frac{P(\tau;\theta)}{P(\tau;\theta)}\nabla_\theta P(\tau;\theta) \tag{11}$$

$$\tag{12}$$

Move differentiation inside sum (ignore $R(\tau)$ and then add in term that cancels out

Likelihood Ratio Policy Gradient

$$\sum_t p(\tau;\theta)R(\tau) \tag{10}$$

Taking the gradient wrt θ:

$$\nabla_\theta U(\theta) = \sum_\tau R(\tau)\frac{P(\tau;\theta)}{P(\tau;\theta)}\nabla_\theta P(\tau;\theta) \tag{11}$$

$$= \sum_\tau P(\tau;\theta)\frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)}R(\tau) \tag{12}$$

$$\tag{13}$$

Move derivative over probability

Likelihood Ratio Policy Gradient

$$\sum_t p(\tau;\theta)R(\tau) \tag{10}$$

Taking the gradient wrt θ:

$$\begin{aligned}
\nabla_\theta U(\theta) &= \sum_\tau R(\tau)\frac{P(\tau;\theta)}{P(\tau;\theta)}\nabla_\theta P(\tau;\theta) && (11)\\
&= \sum_\tau P(\tau;\theta)\frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)}R(\tau) && (12)\\
&= \sum_\tau P(\tau;\theta)\nabla_\theta\left[\log P(\tau;\theta)\right]R(\tau) && (13)
\end{aligned}$$

Assume softmax form

Likelihood Ratio Policy Gradient

$$\sum_t p(\tau;\theta)R(\tau) \tag{10}$$

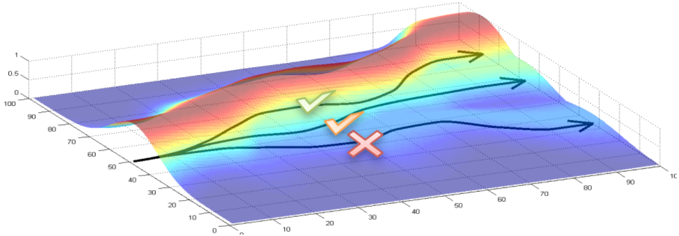
Taking the gradient wrt θ:

$$= \sum_\tau P(\tau;\theta)\nabla_\theta \left[\log P(\tau;\theta)\right] R(\tau) \tag{11}$$

Approximate with empirical estimate for m sample paths from π

$$\nabla_\theta U(\theta) \approx \frac{1}{m}\sum_1^m \nabla_\theta \log P(r^i;\theta)R(\tau^i) \tag{12}$$

Policy Gradient Intuition



- Increase probability of paths with positive R
- Decrease probability of paths with negagive R

Extensions

- Consider baseline b (e.g., path averaging)

$$\nabla_\theta U(\theta) \approx \frac{1}{m} \sum_1^m \nabla_\theta \log P(r^i;\theta)(R(\tau^i) - b(\tau)) \tag{13}$$

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