Reinforcement Learning

Prof. Volkan Cevher
volkan.cevher@epfl.ch

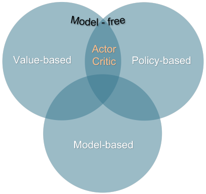
Lecture 8: Deep and Robust Reinforcement Learning

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-568 (Spring 2024)

License Information for Reinforcement Learning (EE-568)

Literature recap: Overview of reinforcement learning approaches



Value-based RL (Critic)

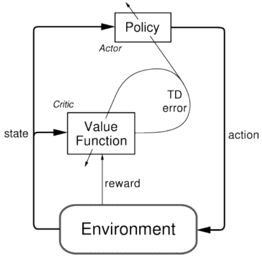
- Learn the optimal value functions $V^\star, Q^\star$
- Algorithms: Monte Carlo, SARSA, Q-learning, etc.
- Use temporal difference (low variance)
- Does not scale to large action spaces

Policy-based RL (Actor)

- Learn the optimal policy via gradient methods
- Algorithms: PG, NPG, TRPO, PPO, etc.
- Scales to large or continuous action spaces
- High variance, sample inefficiency

Actor-Critic (AC) methods

◦ AC methods aim at combining the advantages of actor-only methods and critic-only methods.



Interaction of Actor-Critic [25].

◦ The actor uses the policy gradient to update the learning policy.

◦ The critic uses TD learning to estimate the value function.

EE-568 perspective: Policy improvement updates as bilevel optimization

◦ Recall from Lecture 2 that the policy iteration updates can be written as follows

$$\pi^{k+1}(\cdot|s) = \operatorname{argmax}_{\pi \in \Delta_{\mathcal{A}}} \langle \pi, Q^{\pi^k}(s, \cdot) \rangle,$$

- Q^{π^k} is the fixed point of the operator $\mathcal{T}^{\pi^k}$, i.e. $Q^{\pi^k} = \mathcal{T}^{\pi^k} Q^{\pi^k}$.
- Equivalently, we can write $Q^{\pi^k} = \operatorname{argmin}_{Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}} \|Q - \mathcal{T}^{\pi^k} Q\|^2$.
- Hence, we can equivalently express the policy improvement update as a bilevel optimization problem

$$\begin{aligned} \pi^{k+1}(\cdot|s) &= \operatorname{argmax}_{\pi \in \Delta_{\mathcal{A}}} \langle \pi, q^\star(s, \cdot) \rangle \\ \text{s.t.} \quad q^\star &= \operatorname{argmin}_{Q \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}} \|Q - \mathcal{T}^{\pi^k} Q\|^2. \end{aligned}$$

Remarks: ◦ Methods that solve this program with function approximation are known as Actor-Critic methods.

◦ The outer problem updates are called actor updates.

◦ The inner problem updates are called critic updates.

Actor-Critic methods

- Actor-critic algorithms follow an approximate policy gradient:

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{1-\gamma} \mathbb{E}_{s \sim \lambda_\mu^{\pi_\theta}} \left[\mathbb{E}_{a \sim \pi_\theta(\cdot|s)} \left[Q_w(s,a) \nabla_\theta \log \pi_\theta(a \mid s) \right] \right].$$

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{1-\gamma} \mathbb{E}_{s \sim \lambda_\mu^{\pi_\theta}} \left[\mathbb{E}_{a \sim \pi_\theta(\cdot|s)} \left[A_w(s,a) \nabla_\theta \log \pi_\theta(a \mid s) \right] \right].$$

- Actor: adjust the policy parameter θ using policy gradient using the value function estimated by the critic.

- Critic: update the parameter w to estimate action-value or advantage function.

$$Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$$
$$A_w(s,a) \approx Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

Bias in Actor-Critic methods

- Recall action value expression of policy gradient

$$\nabla_\theta J(\pi_\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim\lambda_\mu^{\pi_\theta}}\left[\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a \mid s)\right]\right].$$

- Policy gradient estimators used by actor-critic algorithms:

$$\hat{\nabla}_\theta J(\pi_\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim\lambda_\mu^{\pi_\theta}}\left[\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q_w(s,a)\nabla_\theta \log \pi_\theta(a \mid s)\right]\right].$$

- Approximating the policy gradient using value function approximation Q_w could introduce bias.

- Luckily, if the value function approximation Q_w is chosen carefully, one may avoid such bias.

Compatible function approximation theorem

Compatible function approximation theorem [26]

Suppose the following two conditions are satisfied:

- Value function approximation at $w^\star$ is compatible to the policy, i.e.,

$$\nabla_w Q_{w^\star}(s,a) \;=\; \nabla_\theta \log \pi_\theta(a \mid s).$$

- Value function parameter $w^\star$ minimizes the mean-squared error, i.e.,

$$\min_w \; \mathbb{E}_{s\sim\lambda_\mu^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}[(Q_w(s,a) - Q^{\pi_\theta}(s,a))^2].$$

Then the policy gradient using critic $Q_{w^\star}(s,a)$ is exact:

$$\nabla_\theta J(\theta) \;=\; \frac{1}{1-\gamma}\mathbb{E}_{s\sim\lambda_\mu^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}[\nabla_\theta \log \pi_\theta(a \mid s) Q_{w^\star}(s,a)].$$

Remarks:

- Proof follows immediately from first-order optimality condition.
- Example: $Q_w(s,a) = \nabla_\theta \log \pi_\theta(a \mid s)^\top w$.

Variant I: Online Action-Value Actor-Critic

Online Action-Value Actor-Critic Algorithm

Initialize θ_0, w_0, state $s_0 \sim \mu$, $a_0 \sim \pi_{\theta_0}(\cdot \mid s_0)$.

for each step of the episode $t = 0, ..., T$ **do**

Obtain (r_t, s_{t+1}, a_{t+1}) from π_{θ_t}.

Compute policy gradient estimator: $\hat{\nabla}_\theta J(\pi_{\theta_t}) = Q_{w_t}(s_t, a_t) \nabla_\theta \log \pi_{\theta_t}(a_t \mid s_t)$.

Actor update θ: $\theta_{t+1} = \theta_t + \alpha_t \hat{\nabla}_\theta J(\pi_{\theta_t})$.

Compute temporal difference: $\delta_t = r_t + \gamma Q_{w_t}(s_{t+1}, a_{t+1}) - Q_{w_t}(s_t, a_t)$.

Critic update: $w_{t+1} = w_t - \beta_t \delta_t \nabla_w Q_{w_t}(s_t, a_t)$.

end for

Remarks:

- Uses temporal difference to estimate the value function Q^{π_θ}.
- Examples for Q_w: linear value function approximation $Q_w(s, a) = \phi(s, a)^\top w$.

Variant II: Advantage Actor-Critic

Advantage Actor-Critic (A2C)

Initialize θ_0, w_0, state $s_0 \sim \mu$.

for each step of the episode $t = 0, ..., T$ **do**

Take action $a_t \sim \pi_{\theta_t}(\cdot \mid s_t)$, obtain (r_t, s_{t+1}).

Estimate advantage function: $\delta_t = r_t + \gamma V_{w_t}(s_{t+1}) - V_{w_t}(s_t)$.

Compute policy gradient estimator: $\hat{\nabla}_\theta J(\pi_{\theta_t}) = \delta_t \nabla_\theta \log \pi_{\theta_t}(a_t \mid s_t)$.

Actor update: $\theta_{t+1} = \theta_t + \alpha_t \hat{\nabla}_\theta J(\pi_{\theta_t})$.

Critic update: $w_{t+1} = w_t - \beta_t \delta_t \nabla_w V_{w_t}(s_t)$.

end for

Remarks:

- Use $V_w(s)$ to approximate $V^{\pi_\theta}(s)$, for instance $V^w(s) \approx \phi(s)^\top w$.
- Use one step lookahead to estimate $Q^{\pi_\theta}(s_t, a_t) \approx r(s_t, a_t) + \gamma V^{\pi_\theta}(s_{t+1})$.
- Use advantage function to approximate the policy gradient.

Various Actor-Critic extensions

- Natural Actor-Critic [16]: use TRPO [22] or NPG[8] to update the actor
- Actor-Critic with generalized advantage estimator [23]: generalize advantage function with TD(λ)

$$\hat{A}_t^k(s_t, a_t) = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \cdots + \gamma^k V_w(s_{t+k}) - V_w(s_t)$$

$$\hat{A}_t^{\mathrm{GAE}}(s_t, a_t) = (1-\lambda)\sum_{k=1}^{\infty} \lambda^{k-1} \hat{A}_t^k(s_t, a_t)$$

- Soft Actor-Critic [6]: use entropy regularization in the objective to add strong convexity to the upper problem:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) + \rho \cdot \mathcal{H}(\pi(\cdot|s_t))\right], \text{ where } \mathcal{H}(\pi(\cdot|s)) = \mathbb{E}_{a\sim\pi(\cdot|s)}[-\log \pi(a|s)],$$

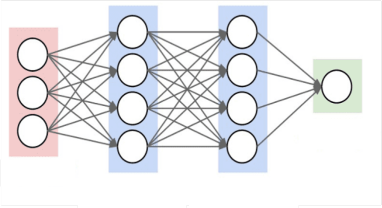
where ρ is a smoothing parameter (i.e, smooths the dual).

Convergence of Actor-Critic methods

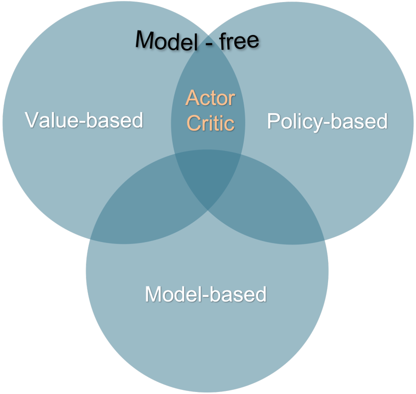
Remarks:
- The asymptotic analysis of two time-scale actor-critic methods (i.e., $\lim_{t\to\infty} \frac{\alpha_t}{\beta_t} = 0$) is in [2, 9].
- The proof is based on two-time-scale stochastic approximation and ODE analysis.
- Finite-sample analyses of actor-critic methods (tabular or LFA) have been studied recently [33].
- This work is based on the bilevel optimization perspective; see e.g., [33].

Deep reinforcement learning = DL + RL

- Tabular methods and linear function approximation are insufficient for large-scale RL applications.
- Using neural networks seems to be a must.



$+$



Neural networks

- Nested composition of (learnable) linear transformation with (fixed) nonlinear activation functions
- Example: a single-layer neural network (shallow neural network)

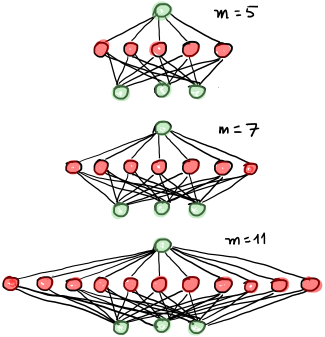


Figure: Networks of increasing width

$$f(\mathbf{x}; W, \boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i \cdot \sigma(w_i^\top \mathbf{x})$$

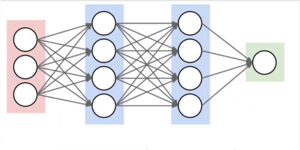
Activation function $\sigma(\cdot)$

- Identity: $\sigma(u) = u$
- Sigmoid: $\sigma(u) = \frac{1}{1+\exp(-u)}$
- Tanh: $\sigma(u) = \tanh(u)$
- Rectified linear unit (ReLU): $\sigma(u) = \max(0, u)$
- ...

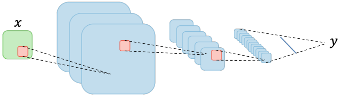
Deep neural networks

◦ More hidden layers, different activation functions, more general graph structure ....

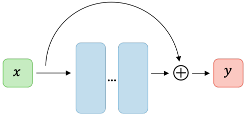
Feed forward network



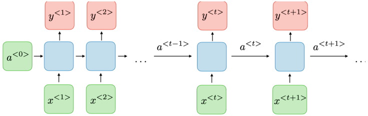
Convolutional network



Residual network



Recurrent network



Why neural networks?

- **Universal Approximation**
  - Any continuous function on a compact domain can be (uniformly) approximated to arbitrary accuracy by a single-hidden layer neural network with a non-polynomial activation function. [Cybenko, 1989; Hornik et al., 1989; Barron, 1993]
  - But the number of neurons can be large.

- **Benefits of depth**
  - A deep network cannot be approximated by a reasonably-sized shallow network.[34]
  - For example, there exists a function with $O(L^2)$ layers and width 2 which requires width $O(2^L)$ to approximate with $O(L)$ layers [27]. For more refined depth separation results see [20].

Example: ATARI network architecture

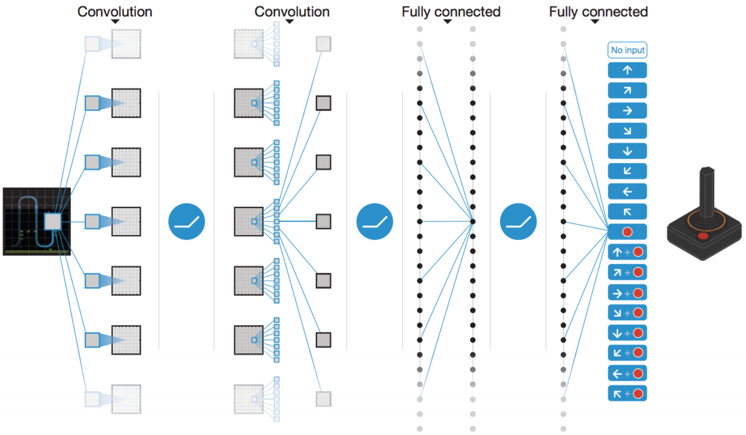


Figure: ATARI Network Architecture for $Q(s, a)$: History of frames as input. One output per action. [12]

Challenges with training neural networks in RL

- Deadly triad: divergence when combining function approximation, bootstrapping, and off-policy learning
- Non iid data
- Sample inefficiency
- High variance
- Overfitting
- Saddle points
- ...

A summary of the common fixes or RL tricks of the trade

- Better data: e.g., experience replay (mix online data and a buffer from past experience)
  - Reduce correlation, allow mini-batch update
- Better objective: e.g., use entropy regularization
  - Improve optimization landscape, encourage exploration
- Better optimizers: e.g., adaptive SGD such as Adam and RMSProp
  - Adaptive learning rates
- Better estimation: e.g., use eligibility traces, target works
  - Reduce overestimation bias, balance bias-variance tradeoff
- Better sampling: e.g., use prioritized replay (sample based on priority)
  - Prioritize transitions on which we can learn much
- Better implementation: e.g., parallel implementation (multithreading of CPU)
  - Speed up training, reduce correlation, allow better exploration
- Better architectures: e.g. dueling networks
  - Encode inductive biases that are good for RL

Value-based deep RL

- Idea: use neural networks for value function approximation
- Recall Q-learning:

Q Learning

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$

Q-learning with function approximation

$w_{t+1} \leftarrow w_t + \alpha_t[r_t + \gamma \max_a Q_{w_t}(s_{t+1}, a) - Q_{w_t}(s_t, a_t)]\nabla Q_{w_t}(s_t, a_t)$

Remarks:
- Note that Q-learning is not a(n unbiased) stochastic gradient descent method.
- Naive deep Q-learning could diverge due to sample correlation and moving targets.
- Deep Q-Networks (DeepMind, 2015) [12]: combine several techniques for stabilizing Q-learning.
- Experience replay (better data efficiency and make data more stationary).
- Target networks (prevent target objective from changing too fast).

Deep Q-Networks (DQN)

- Main idea: minimize the following mean-square error by SGD (or adaptive SGD)

$$\min_w L(w) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\left(r + \gamma \max_{a'} Q(s', a'; w^-) - Q(s, a; w) \right)^2 \right]$$

- The target parameter w^- is held fixed and updated periodically

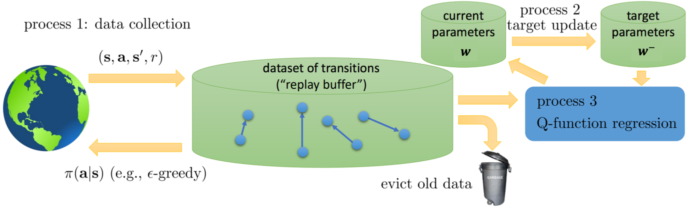


Figure: A more general view of DQN. Source: `https://zhuanlan.zhihu.com/p/468385820`

Another bilevel interpretation

- DQN aims at finding the optimal policy and state action value function directly.
  - Recall from lecture 2 that $\pi^\star(\cdot|s) = \mathrm{argmax}_{\pi \in \Delta_{\mathcal{A}}} \langle \pi, Q^\star(s,\cdot)\rangle$.
  - Recall from lecture 2 that we have that $Q^\star = \mathrm{argmin}_{Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}} \|Q - T^{\pi^\star} Q\|$.
- These facts lead to the following bilevel optimization problem

$$\pi^\star(\cdot|s) = \mathrm{argmax}_{\pi \in \Delta_{\mathcal{A}}} \langle \pi, Q^\star(s,\cdot)\rangle$$
$$\text{s.t.} \quad Q^\star = \mathrm{argmin}_{Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}} \|Q - T^{\pi^\star} Q\|.$$

Remarks:

- This bilevel problem is more complex than the one in Actor Critic.
- The reason is that the inner problem depends on the solution of the outer problem.
- This optimization template is implementable if the transition dynamics is known.
- DQN attempts to approximately solve this bilevel problem.

DQN in playing Atari games [12]

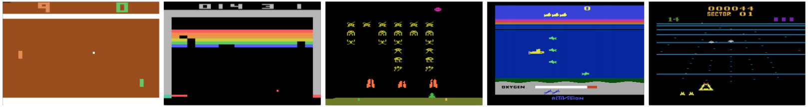


Figure: Five Atari 2600 Games: Pong, Breakout, Space Invaders, Seaquest, Beam Rider

	B. Rider	**Breakout**	**Enduro**	**Pong**	**Q*bert**	**Seaquest**	**S. Invaders**
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690

Figure: Average total reward for a fixed number of steps.

- DQN source code: https://github.com/deepmind/dqn

DQN extensions I

◦ Double DQN (DeepMind, 2016) [29]: Use separate networks to select best action and evaluate best action to reduce overestimation bias

$$\min_w L(w) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\left(r + \gamma Q(s', \arg\max_{a'} Q(s', a'; w); w^-) - Q(s, a; w) \right)^2 \right]$$

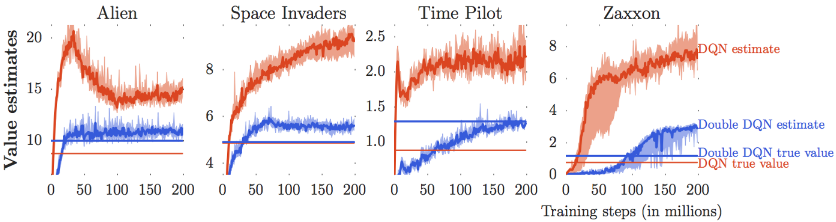


Figure: Value estimates by DQN (orange) and Double DQN (blue) on Atari games. The straight horizontal lines are computed by running the corresponding agents after learning concluded, and averaging the actual discounted return obtained from each visited state.

DQN extensions II

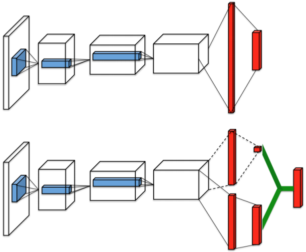
- DQN with prioritized experience replay [21]: Prioritize transitions in proportion to the absolute Bellman error

$$p \propto \left| r + \gamma \max_{a'} Q(s', a'; w) - Q(s, a; w) \right|$$

$\langle S_t, A_t, R_{t+1}, S_{t+1}, p_t \rangle$
$\langle S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, p_{t+1} \rangle$
$\langle S_{t+2}, A_{t+2}, R_{t+3}, S_{t+3}, p_{t+2} \rangle$
$\langle S_{t+3}, A_{t+3}, R_{t+4}, S_{t+4}, p_{t+3} \rangle$
$\dots$

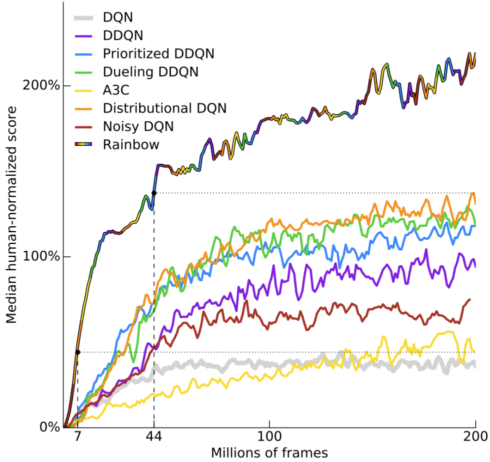
- Dueling DQN [30]: Split Q-networks into two streams to estimate value function and advantage function

$$Q(s, a; w, \alpha, \beta) = V(s; w, \beta) + \bar{A}(s, a; w, \alpha)$$



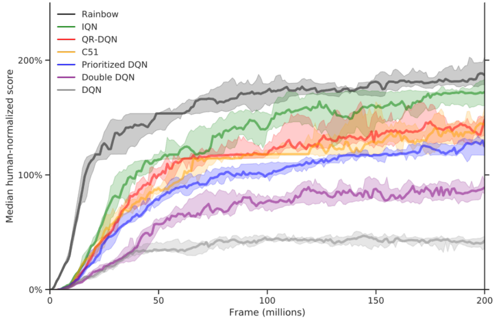
DQN mega extension

- Can these extensions be combined? Yes, Rainbow [7]!



The big zoo of DQN

Directory	Paper
dqn	Human Level Control Through Deep Reinforcement Learning
double_q	Deep Reinforcement Learning with Double Q-learning
prioritized	Prioritized Experience Replay
c51	A Distributional Perspective on Reinforcement Learning
qrdqn	Distributional Reinforcement Learning with Quantile Regression
rainbow	Rainbow: Combining Improvements in Deep Reinforcement Learning
iqn	Implicit Quantile Networks for Distributional Reinforcement Learning



Plot of median human-normalized score over all 57 Atari games for each agent

◦ Source code: https://github.com/deepmind/dqn_zoo

Policy-based/Actor-Critic DRL

◦ Combine the actor-critic approach with Deep Q Network

- Asynchronous Advantage Actor-Critic (A3C)) [11]
- Soft Actor Critic (SAC) [6]
- Deep deterministic policy gradient (DDPG) [10]: continuous control
- Twin Delayed DDPG (TD3) [4]: continuous control
- ....

A3C [11]

◦ Idea: advantage actor-critic + deep Q-network + asynchronous implementation

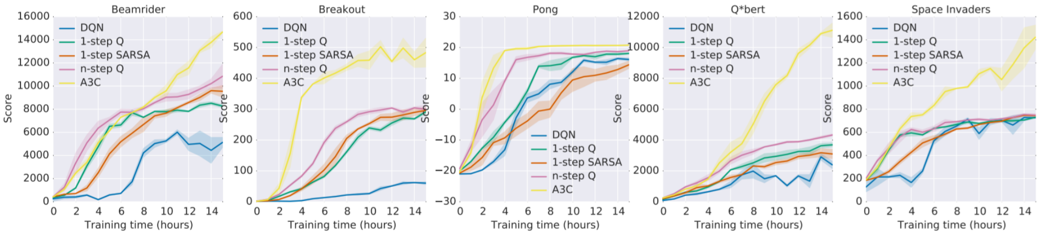
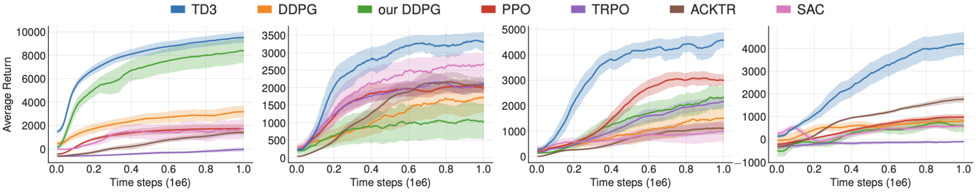


Figure: Comparison for DQN and A3C on five Atari 2600 games. 1-step Q means asynchronous one-step Q-learning.

DDPG [10] and TD3 [4]

- DDPG: deterministic policy gradient + deep Q-network
- Select action $a \sim \mu(s;\theta) + \mathcal{N}(0, \sigma^2)$ (add noise to enhance exploration)
- Policy update: $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_i \nabla_a Q_w(s_i, \mu(s_i;\theta))\nabla_\theta \mu(s_i;\theta)$

- TD3: DDPG + clipped action exploration + delayed policy update + pessimistic double Q-learning
  - Select action $a \sim \mu(s;\theta) + \epsilon$, $\epsilon \sim \text{clip}(N(0,\sigma^2), -c, c)$
  - Delayed policy update: update critic more frequent than policy



(a) HalfCheetah-v1 (b) Hopper-v1 (c) Walker2d-v1 (d) Ant-v1

Figure: Learning curves for the OpenAI gym continuous control tasks.

Summary

- **Deep Value-based Methods**
  - DQN
  - Double DQN
  - Dueling DQN
  - DQN with prioritized experience replay
  - Rainbow
  - ....

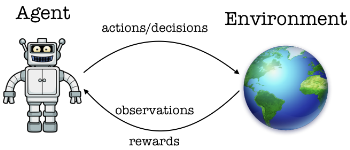
- **Deep Policy-based/Actor-Critic Methods**
  - TRPO
  - PPO
  - A3C
  - SAC
  - DDPG/TD3
  - ....

Question: So, which one should we choose in practice? when do they work well?

Deep RL resources

- OpenAI Spinning up: https://spinningup.openai.com/
- The awesome list of deep RL (libraries and tutorials): https://github.com/kengz/awesome-deep-rl

Reinforcement learning



- Environment: Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \gamma, \mu, r)$
- Agent: Parameterized deterministic policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$, where $\theta \in \Theta$

Reinforcement learning (RL) game

At time step $t = 0$: $S_0 \sim \mu(\cdot)$

for $t = 1, 2, \ldots$ **do**:

agent observes the environment's state $S_t \in \mathcal{S}$

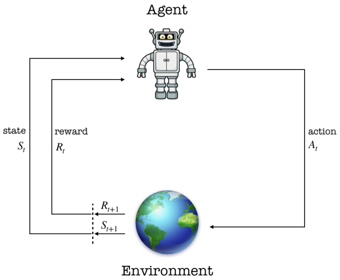
agent chooses an action $A_t = \pi_\theta(S_t) \in \mathcal{A}$

agent receives a reward $R_{t+1} = r(S_t, A_t)$

agent finds itself in a new state $S_{t+1} \sim T(\cdot \mid S_t, A_t)$

Exploration vs. exploitation in RL

- Challenge: Exploration vs. exploitation!



- Objective (non-concave): $$\max_{\theta\in\Theta} \; J(\theta) \; := \; \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \;\middle|\; \pi_\theta, \mathcal{M}\right]$$
  - The environment only reveals the rewards after actions
  - Exploitation: Maximize objective by choosing the appropriate action
  - Exploration: Gather information on other actions

An optimization interpretation

◦ Objective (non-concave): $\max_{\theta\in\Theta} \; J(\theta) \; := \; \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \;\middle|\; \pi_\theta, \mathcal{M}\right]$

◦ Exploitation: Progress in the gradient direction

$$\theta_{t+1} \;\leftarrow\; \theta_t + \eta_t \widehat{\nabla_\theta J(\theta_t)}$$

◦ Exploration: Add stochasticity while collecting the episodes

- noise injection in the action space [24, 10]

$$a = \pi_\theta(s) + \mathcal{N}(0, \sigma^2 I)$$

- noise injection in the parameter space [18]

$$\tilde{\theta} = \theta + \mathcal{N}(0, \sigma^2 I)$$

Reinforcement learning with Langevin dynamics I

- Explore via an infinite dimensional concave-problem (linear in p):

$$\underset{p\in\mathcal{M}(\Theta)}{\text{maximize}} \quad \mathop{\mathbb{E}}_{\theta\sim p}[J(\theta)]$$

  - $\mathcal{M}(\Theta)$ is the (infinite dimensional) space of all probability distributions on Θ.
  - $p^\star = \arg\max_p \mathop{\mathbb{E}}_{\theta\sim p}[J(\theta)]$ is a delta measure centered at $\theta^\star = \arg\max_\theta J(\theta)$.

Reinforcement learning with Langevin dynamics II

- Exploit via a well-known entropy smoothing trick:

$$\underset{p \in \mathcal{M}(\Theta)}{\text{maximize}} \quad \mathbb{E}_{\theta \sim p}[J(\theta)] + \beta H(p)$$

  - $H(p) = \mathbb{E}_{\theta \sim p}[-\log p(\theta)]$ is the entropy of the distribution p.
  - the optimal solution takes the form $p^\star_\beta(\theta) \propto \exp\left(\frac{1}{\beta} J(\theta)\right)$.

- Our proposal for explore-exploit
  - Use Langevin dynamics [31] to draw samples from $p^\star_\beta(\theta)$
  - Use homotopy on the smoothing parameter β

Learning robust policies

- Why robust RL? In short: Generalization under environmental changes
  - upshots: self-driving car in varying environmental conditions
  - trends: from simple parametric models to super expressive neural networks
  - challenges: computational costs as well as the difficulty of training
- Highlight: Robust Adversarial Reinforcement Learning (RARL) [17]
  - train an **agent** neural net
  - train an **adversary** neural net
  - setup a minimax game between the two
- Several variants exist [14, 32]
- Action Robust RL [28]

Two-Player Zero-Sum Markov Game

- Players:
  - Environment: Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \bar{\mathcal{A}}, T, \gamma, r, \mu)$
  - Agent: parameterized deterministic policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$, where $\theta \in \Theta$
  - Adversary: parameterized deterministic policy $\nu_\omega : \mathcal{S} \to \bar{\mathcal{A}}$, where $\omega \in \Omega$

Two-Player Zero-Sum Markov Game

At time step $t = 0$: $S_0 \sim \mu(\cdot)$

for $t = 1, 2, \ldots$ **do**:

both players observe the environment's state $S_t \in \mathcal{S}$

both players choose the actions $A_t = \pi_\theta(S_t) \in \mathcal{A}$, and $\bar{A}_t = \nu_\omega(S_t) \in \bar{\mathcal{A}}$

the agent gets a reward $R_{t+1} = r(S_t, A_t, \bar{A}_t)$ while the adversary gets $-R_{t+1}$

both players find themselves in a new state $S_{t+1} \sim T(\cdot \mid S_t, A_t, \bar{A}_t)$

- Performance objective:

$$\max_{\theta \in \Theta} \min_{\omega \in \Omega} \; J(\theta, \omega) \; := \; \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \;\middle|\; \pi_\theta, \nu_\omega, \mathcal{M} \right]$$

Robust Adversarial Reinforcement Learning (RARL)

- A natural *pure* strategy-based minimax objective

$$\max_{\theta \in \Theta} \min_{\omega \in \Omega} \; J(\theta, \omega).$$

  - θ: an **agent** neural net
  - ω: an **adversary** neural net
  - *highly* non-concave/non-convex objective

- Theoretical challenges
  - a saddle point might NOT exist [3]
  - no provably convergent algorithm

- Practical challenges
  - the simple (alternating) SGD does NOT work well in practice
  - adaptive methods (Adam, RMSProp,...) highly unstable, heavy tuning

RARL: From pure to mixed Nash Equilibrium

◦ Objective of RARL is a pure strategy formulation:

$$\max_{\theta \in \Theta} \min_{\omega \in \Omega} \; J(\theta, \omega).$$

◦ A new objective of RARL: Our **mixed** strategy proposal via game theory

$$\max_{p \in \mathcal{M}(\Theta)} \min_{q \in \mathcal{M}(\Omega)} \; \boldsymbol{E}_{\theta \sim p} \boldsymbol{E}_{\omega \sim q} \left[J(\theta, \omega) \right].$$

where $\mathcal{M}(\mathcal{Z}) \coloneqq \{$all (regular) probability measures on $\mathcal{Z}\}$.

◦ Existence of NE $(p^\star, q^\star)$: Glicksberg's existence theorem [5].

A re-thinking of RARL via the mixed Nash equilibrium

◦ **Upshot:** Our mixed Nash Equilibrium proposal ≡ bi-linear matrix games

$$\max_{p\in\mathcal{M}(\Theta)} \min_{q\in\mathcal{M}(\Omega)} \boldsymbol{E}_{\theta\sim p}\boldsymbol{E}_{\omega\sim q}\left[J(\theta,\omega)\right]$$

$$\Updownarrow$$

$$\max_{p\in\mathcal{M}(\Theta)} \min_{q\in\mathcal{M}(\Omega)} \langle p, Gq\rangle$$

- Caveat: **Infinite dimensions!!!**

◦ Key ingredients moving forward

- $\langle p, h\rangle := \int h dp$ for a measure p and function h (Riesz representation)
- the linear operator G and its adjoint $G^\dagger$:

$$\begin{aligned}(Gq)(\theta) &:= \boldsymbol{E}_{\omega\sim q}\left[J(\theta,\omega)\right]\\ (G^\dagger p)(\omega) &:= \boldsymbol{E}_{\theta\sim p}\left[J(\theta,\omega)\right],\end{aligned}$$

where $G : \mathcal{M}(\Omega) \to \mathcal{F}(\Theta)$, and $G^\dagger : \mathcal{M}(\Theta) \to \mathcal{F}(\Omega)$.

Training Phase

◦ We use the following special adversary with $\alpha = 0.1$ (Noisy Action Robust MDP):

Noisy Action Robust MDP Game

for $t = 1, 2, \ldots$ **do**:

both players observe the environment's state $S_t \in \mathcal{S}$

both players choose the actions $A_t = \mu(S_t) \in \mathcal{A}$, and $A'_t = \nu(S_t) \in \mathcal{A}$

the resulting action $\bar{A}_t = (1 - \alpha)A_t + \alpha A'_t$ is executed in the environment $\mathcal{M}$

the agent gets a reward $R_{t+1} = r(S_t, \bar{A}_t)$ while the adversary gets $-R_{t+1}$

both players find themselves in a new state S_{t+1}

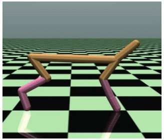
◦ We train the policy based on specific environment parameters

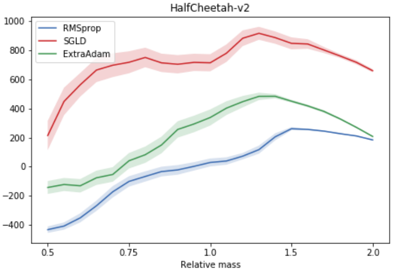
- ◦ i.e., standard relative mass variables in OpenAI gym.

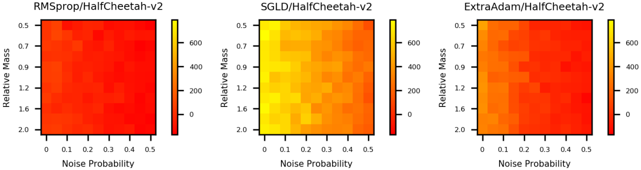
Testing Phase

- Robustness under Adversarial Disturbances (x-axis of the heatmap):
  - measure performance in the presence of an adversarial disturbance.
- Robustness to Test Conditions (y-axis of the heatmap):
  - measure performance with respect to varying test conditions.

Experimental evaluation via MuJoCo







Wrap up!

- That's it folks!
- We hope to have made you passioned and skilled to start your project !
- If not done yet register your project via the following form `https://forms.gle/ssLrrH5FSAGs42wQ6`.
- From now on Thursdays at 1 pm the TAs will be available to answer your questions.
- We can not guarantee support in other time slots.
- Please send your poster by 23rd May 2024 via Moodle.
- Posters and apero on 30th May 2024! Details will follow.

References I

[1] Amir Beck and Marc Teboulle.
Mirror descent and nonlinear projected subgradient methods for convex optimization.
Operations Research Letters, 31(3):167–175, 2003.
55

[2] Vivek S Borkar and Vijaymohan R Konda.
The actor-critic algorithm as multi-time-scale stochastic approximation.
Sadhana, 22(4):525–543, 1997.
12

[3] Partha Dasgupta and Eric Maskin.
The existence of equilibrium in discontinuous economic games, i: Theory.
The Review of economic studies, 53(1):1–26, 1986.
40

[4] Scott Fujimoto, Herke Hoof, and David Meger.
Addressing function approximation error in actor-critic methods.
In *International Conference on Machine Learning*, pages 1582–1591, 2018.
28, 30

[5] Irving L Glicksberg.
A further generalization of the kakutani fixed point theorem, with application to nash equilibrium points.
Proceedings of the American Mathematical Society, 3(1):170–174, 1952.
41

References II

[6] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine.
Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.
In *International Conference on Machine Learning*, pages 1856–1865, 2018.
11, 28

[7] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver.
Rainbow: Combining improvements in deep reinforcement learning.
In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
26

[8] S. Kakade.
A natural policy gradient.
In *Advances in Neural Information Processing Systems (NeurIPS)*, 2001.
11

[9] Vijay Konda and John Tsitsiklis.
Actor-critic algorithms.
Advances in neural information processing systems, 12, 1999.
12

[10] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra.
Continuous control with deep reinforcement learning.
arXiv preprint arXiv:1509.02971, 2015.
28, 30, 35

References III

[11] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu.
Asynchronous methods for deep reinforcement learning.
In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.
28, 29

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al.
Human-level control through deep reinforcement learning.
Nature, 518(7540):529–533, 2015.
17, 20, 23

[13] Arkadi Nemirovski.
Prox-method with rate of convergence o $(1/t)$ for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems.
SIAM Journal on Optimization, 15(1):229–251, 2004.
55

[14] Arnab Nilim and Laurent El Ghaoui.
Robust control of markov decision processes with uncertain transition matrices.
Operations Research, 53(5):780–798, 2005.
38

References IV

[15] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al.
Training language models to follow instructions with human feedback.
Advances in neural information processing systems, 35:27730–27744, 2022.
57, 58, 59, 60

[16] Jan Peters and Stefan Schaal.
Natural actor-critic.
Neurocomputing, 71(7-9):1180–1190, 2008.
11

[17] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta.
Robust adversarial reinforcement learning.
In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2817–2826. JMLR. org, 2017.
38

[18] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz.
Parameter space noise for exploration.
arXiv preprint arXiv:1706.01905, 2017.
35

References V

[19] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn.
Direct preference optimization: Your language model is secretly a reward model.
Advances in Neural Information Processing Systems, 36, 2024.
60, 62

[20] Itay Safran, Ronen Eldan, and Ohad Shamir.
Depth separations in neural networks: What is actually being separated?
In Alina Beygelzimer and Daniel Hsu, editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 2664–2666. PMLR, 25–28 Jun 2019.
16

[21] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver.
Prioritized experience replay.
arXiv preprint arXiv:1511.05952, 2015.
25

[22] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz.
Trust region policy optimization.
In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.
11

[23] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel.
High-dimensional continuous control using generalized advantage estimation.
arXiv preprint arXiv:1506.02438, 2015.
11

References VI

[24] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller.
Deterministic policy gradient algorithms.
In *ICML*, 2014.
35

[25] Richard S Sutton and Andrew G Barto.
Reinforcement learning: An introduction.
MIT press, 2018.
4

[26] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour.
Policy gradient methods for reinforcement learning with function approximation.
In *Advances in neural information processing systems*, pages 1057–1063, 2000.
8

[27] Matus Telgarsky.
Benefits of depth in neural networks.
In *Conference on learning theory*, pages 1517–1539. PMLR, 2016.
16

[28] Chen Tessler, Yonathan Efroni, and Shie Mannor.
Action robust reinforcement learning and applications in continuous control.
In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.
38

References VII

[29] Hado Van Hasselt, Arthur Guez, and David Silver.
Deep reinforcement learning with double q-learning.
In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
24

[30] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas.
Dueling network architectures for deep reinforcement learning.
In *International Conference on Machine Learning*, pages 1995–2003, 2016.
25

[31] Max Welling and Yee W Teh.
Bayesian learning via stochastic gradient langevin dynamics.
In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.
37

[32] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem.
Robust markov decision processes.
Mathematics of Operations Research, 38(1):153–183, 2013.
38

[33] Yue Wu, Weitong Zhang, Pan Xu, and Quanquan Gu.
A finite time analysis of two time-scale actor critic methods.
arXiv preprint arXiv:2005.01350, 2020.
12

References VIII

[34] Dmitry Yarotsky.
Error bounds for approximations with deep relu networks.
Neural Networks, 94:103–114, 2017.
16

[35] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving.
Fine-tuning language models from human preferences.
arXiv preprint arXiv:1909.08593, 2019.
57, 58, 59, 60

Supplementary: Entropic mirror descent iterates in infinite dimension

- Negative Shannon entropy and its Fenchel dual: (dz :=Lebesgue)
  - $\Phi(p) = \int p \log \frac{dp}{dz}$.
  - $\Phi^\star(h) = \log \int e^h$.
  - $d\Phi$ and $d\Phi^\star$: Fréchet derivatives.[1]

Theorem (Infinite-dimensional mirror descent, informal)

For a learning rate η, a probability measure p, and an arbitrary function h, we can equivalently define

$$p_+ = \mathbf{MD}(p, h) \quad \equiv \quad p_+ = d\Phi^\star\left(d\Phi(p) - \eta h\right) \equiv \quad dp_+ = \frac{e^{-\eta h} dp}{\int e^{-\eta h} dp}.$$

Moreover, most the essential ingredients in the analysis of finite-dimensional prox methods can be generalized to infinite dimension.

- Continuous analog of the entropic mirror descent [1]
  - Mirror-prox also possible [13]

[1] Under mild regularity conditions on the measure/function.

Supplementary: Entropic mirror descent in infinite dimension: rates

◦ Algorithm:

Algorithm 1 Infinite-Dimensional Entropic Mirror Descent

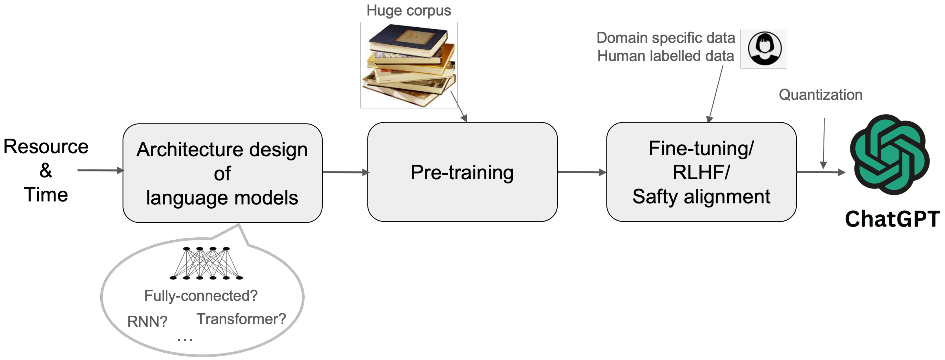
Input: Initial distributions p_1, q_1, and learning rate η
for $t = 1, 2, \ldots, T-1$ **do**
$p_{t+1} = \mathrm{MD}_\eta\left(p_t, -Gq_t\right)$
$q_{t+1} = \mathrm{MD}_\eta\left(p_t, G^\dagger p_t\right)$
end for
Output: $\bar{p}_T = \frac{1}{T}\sum_{t=1}^T p_t$ and $\bar{q}_T = \frac{1}{T}\sum_{t=1}^T q_t$

Theorem (Convergence Rates)

Let $\Phi(p) = \int dp \log \frac{dp}{dz}$. *Then*

1. *Entropic MD* $\Rightarrow O(T^{-\frac{1}{2}})$*-NE.*
2. *If only stochastic derivatives* $(\hat{G}^\dagger p$ *and* $-\hat{G}q)$ *are available, then Entropic MD* $\Rightarrow O(T^{-\frac{1}{2}})$*-NE in expectation.*

Supplementary: Reinforcement learning from human feedback (RLHF) in LLM [35, 15]



Supplementary: Reinforcement learning from human feedback (RLHF) in LLM [35, 15]

- Notation:
  - Policy: a language model π.
  - State: input sentence s.
  - Action: output sentence a, follows distribution $\pi(\cdot|s)$.
- Building LLM - Step 1: Pre-train an LLM based on unlabeled corpus.
- Building LLM - Step 2: Supervised fine-tune via collected demonstration, denoted by π^{SFT}.

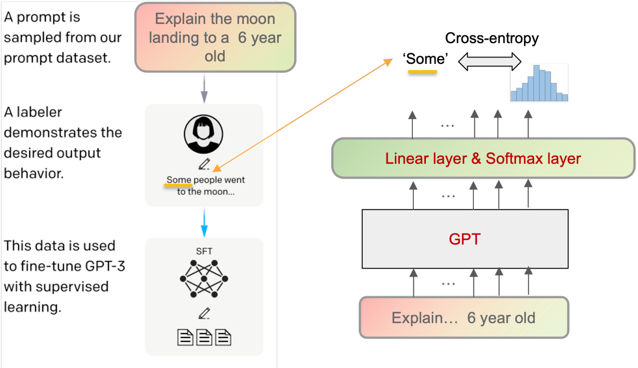


Figure: Step 2: Supervised fine-tune.

Supplementary: Reinforcement learning from human feedback (RLHF) in LLM [35, 15]

- Building LLM - Step 3: Using RLHF to further improve π^{SFT} based on some data pairs $(s, a_w \succ a_l)$.
  - s: "Write me a poem about the history of jazz."
  - Generate a_w and a_l according to $\pi^{\mathsf{SFT}}(\cdot|s)$.
  - a_w: "In smoky halls where shadows dance, A rhythm born of circumstance..."
  - a_l: "In the heart of New Orleans, where the streets hummed..."
  - $a_w \succ a_l$ means a_w is better than a_l, annotated by human preference.

Supplementary: Reinforcement learning from human feedback (RLHF) in LLM [35, 15]

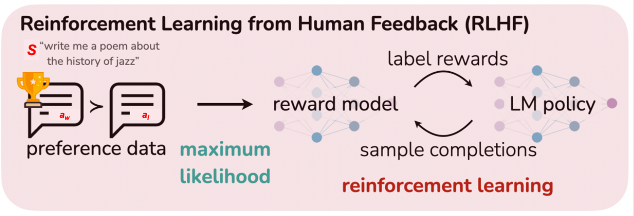


Figure: Diagram of RLHF, from [19].

- The Bradley-Terry (BT) model assumes these pairs follows the distribution $p^\star$

$$p^\star(a_w \succ a_l \mid s) = \frac{\exp\left(r^\star(s, a_w)\right)}{\exp\left(r^\star(s, a_w)\right) + \exp\left(r^\star(s, a_l)\right)} \triangleq \sigma(r^\star(s, a_w) - r^\star(s, a_l)). \tag{1}$$

where σ is the sigmoid function, $r^\star$ is some unknown latent reward model.

- This reward function can be learned by adding a linear layer into π^{SFT}, denoted by r_ϕ with parameters ϕ.

- Given pairs $\mathcal{D} = \left\{s^{(i)}, a_w^{(i)}, a_l^{(i)}\right\}_{i=1}^N$ (assumed sampled from $p^\star$), learn r_ϕ by maximum likelihood.

$$\max L(r_\phi) = \mathbb{E}_{(s, a_w, a_l) \sim \mathcal{D}}\left[\log \sigma(r_\phi(s, a_w) - r_\phi(s, a_l))\right]. \tag{2}$$

Supplementary: Reinforcement learning from human feedback (RLHF) in LLM

◦ The learned reward function is used to provide feedback for fine-tuning LLMs (getting a new policy π_θ).

$$\begin{aligned}
\max_{\pi_\theta} L_{\mathsf{RLHF}}(\pi_\theta) &= \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(a|s)} r_\phi(s,a) - \underbrace{\beta \mathrm{KL}(\pi_\theta(a \mid s) \,||\, \pi^{\mathsf{SFT}}(a \mid s))}_{\text{ensuring the policy doesn't change a lot.}} \\
&= \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(a|s)} \underbrace{r_\phi(s,a) - \beta(\log \pi_\theta(a \mid s) - \log \pi^{\mathsf{SFT}}(a \mid s))}_{r_{\mathsf{PPO}}(s,a)} .
\end{aligned} \tag{3}$$

◦ This process is optimized via PPO with the reward function: $r_{\mathsf{PPO}}(s,a)$.

Supplementary: Direct Preference Optimization (DPO) [19]

∘ In RLHF, we need to first fit an explicit reward model. In DPO, we don't need.

Direct Preference Optimization:
Your Language Model is Secretly a Reward Model

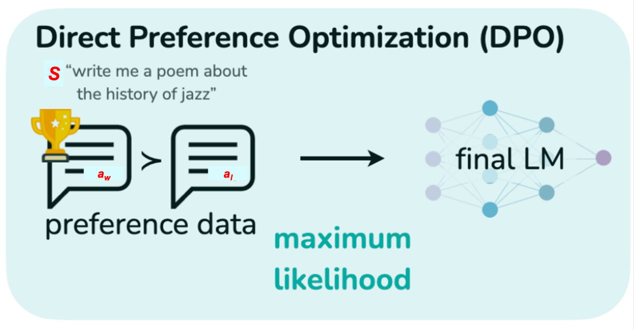


Figure: Diagram of DPO, from [19].

Supplementary: Direct Preference Optimization (DPO)

- We start with Eq. 3 under a general reward function $r^\star$. It is easy to prove that the optimal solution is:

$$\pi^\star(a \mid s) = \frac{1}{Z(s)} \pi^{\mathsf{SFT}}(a \mid s) \exp\left(\frac{1}{\beta} r^\star(s, a)\right), \tag{4}$$

where $Z(s) = \sum_y \pi^{\mathsf{SFT}}(a \mid s) \exp\left(\frac{1}{\beta} r^\star(s, a)\right)$ is the partition function.

- But we can not obtain $\pi^\star(a \mid s)$ in this way as it is expensive to estimate $Z(s)$.
- Alternatively, let us rearrange Eq. 4 as follows:

$$r^\star(s, a) = \beta \log \frac{\pi^\star(a \mid s)}{\pi^{\mathsf{SFT}}(a \mid s)} + \beta \log Z(s). \tag{5}$$

- Substituting Eq. 5 into Eq. 1, the partition function cancels out and we get:

$$p^\star(a_w \succ a_l \mid s) = \sigma\left[\left(\beta \log \frac{\pi^\star(a_w \mid s)}{\pi^{\mathsf{SFT}}(a_w \mid s)} - \beta \log \frac{\pi^\star(a_l \mid s)}{\pi^{\mathsf{SFT}}(a_l \mid s)}\right)\right]. \tag{6}$$

- Hence, the object of DPO becomes:

$$\max \textcolor{red}{L_{\mathsf{DPO}}(\pi_\theta)} = -\mathbb{E}_{(s, a_w, a_l) \sim \mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(a_w \mid s)}{\pi^{\mathsf{SFT}}(a_w \mid s)} - \beta \log \frac{\pi_\theta(a_l \mid s)}{\pi^{\mathsf{SFT}}(a_l \mid s)}\right)\right]. \tag{7}$$