Reinforcement Learning

Lecture 4 Advanced Techniques

Nan Ye

School of Mathematics and Physics
The University of Queensland

Roadmap

- Introduction and overview

  motivation, bandits, big picture
- Classical ideas

  temporal difference methods, policy gradient, . . .
- Deep Reinforcement learning

  neural networks, DQN, DDPG, . . .
- Advanced techniques

  representation learning, stabilization, few-shot learning
- Applications

  AlphaGo, AlphaTensor, . . .

Representation Learning in DRL

representation learning = learning useful high-level features

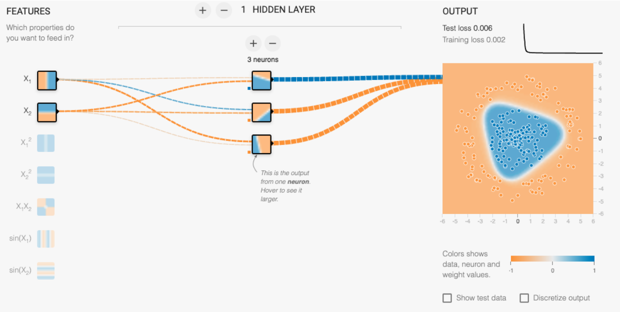
e.g., using CNNs to learn features indicating the presence of noses, eyes, . . .

DRL relies on DNNs for representation learning

such learning is often guided by the reward alone

more supervisory signals can be constructed to learn better representations

recall...



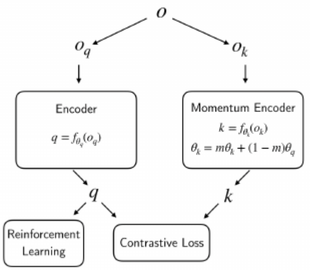
https://playground.tensorflow.org/

a sigmoid unit approximately learns the concept of a circular area in 2D plane

- In deep neural networks ($>$ 1 hidden layer), deeper layers are capable of learning higher-level features.
- This allows learning accurate models from raw features without handcrafting high-level features.

CURL

Contrastive Unsupervised Representations for Reinforcement Learning

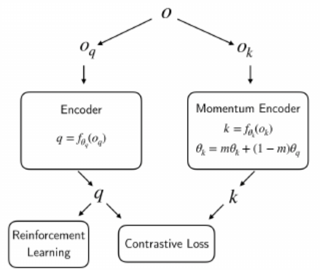


o_q and o_k are augmented versions of the observation o

source: (Laskin, Srinivas, and Abbeel, 2020)

- can be combined with any reinforcement learning algorithm
- additional supervisory signal is a contrastive loss (enforce similar representations for similar observations)

Laskin, Srinivas, and Abbeel, CURL: Contrastive unsupervised representations for reinforcement learning, 2020



o_q and o_k are augmented versions of the observation o

source: (Laskin, Srinivas, and Abbeel, 2020)

InfoNCE loss with bilinear similarity score for contrastive learning

$$L = \frac{\exp(q^\top W k_+)}{\exp(q^\top W k_+) + \sum_{i=0}^{K-1} \exp(q^\top W k_i)}.$$

where $q = f_{\theta_q}(o_q)$, $k_+ = f_{\theta_k}(o_k)$, $k_i = f_{\theta_k}(o_i)$ for another observation o_i

```
# f_q, f_k: encoder networks for anchor
# (query) and target (keys) respectively.
# loader: minibatch sampler from ReplayBuffer
# B-batch_size, C-channels, H,W-spatial_dims
# x : shape : [B, C, H, W]
# C = c * num_frames; c=3 (R/G/B) or 1 (gray)
# m: momentum, e.g. 0.95
# z_dim: latent dimension
f_k.params = f_q.params
W = rand(z_dim, z_dim) # bilinear product.
for x in loader: # load minibatch from buffer
 x_q = aug(x) # random augmentation
 x_k = aug(x) # different random augmentation
 z_q = f_q.forward(x_q)
 z_k = f_k.forward(x_k)
 z_k = z_k.detach() # stop gradient
 proj_k = matmul(W, z_k.T) # bilinear product
 logits = matmul(z_q, proj_k) # B x B
  # subtract max from logits for stability
 logits = logits - max(logits, axis=1)
 labels = arange(logits.shape[0])
 loss = CrossEntropyLoss(logits, labels)
 loss.backward()
 update(f_q.params) # Adam
 update(W) # Adam
 f_k.params = m*f_k.params+(1-m)*f_q.params
```

source: (Laskin, Srinivas, and Abbeel, 2020)

easy to implement

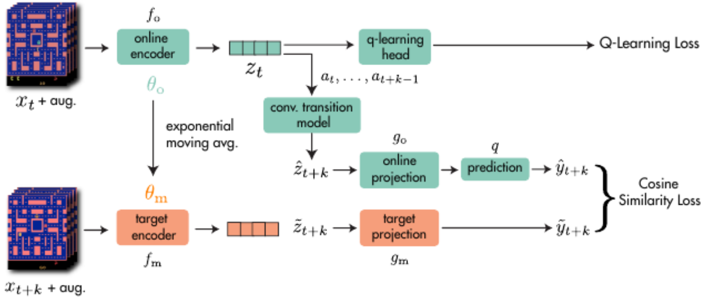
500K STEP SCORES	CURL	PLANET	DREAMER	SAC+AE	SLACV1	PIXEL SAC	STATE SAC
FINGER, SPIN	**926 ± 45**	561 ± 284	796 ± 183	884 ± 128	673 ± 92	179 ± 166	923 ± 21
CARTPOLE, SWINGUP	**841 ± 45**	475 ± 71	762 ± 27	735 ± 63	-	419 ± 40	848 ± 15
REACHER, EASY	**929 ± 44**	210 ± 390	793 ± 164	627 ± 58	-	145 ± 30	923 ± 24
CHEETAH, RUN	518 ± 28	305 ± 131	570 ± 253	550 ± 34	**640 ± 19**	197 ± 15	795 ± 30
WALKER, WALK	**902 ± 43**	351 ± 58	897 ± 49	847 ± 48	842 ± 51	42 ± 12	948 ± 54
BALL IN CUP, CATCH	**959 ± 27**	460 ± 380	879 ± 87	794± 58	852 ± 71	312± 63	974 ± 33
100K STEP SCORES							
FINGER, SPIN	**767 ± 56**	136 ± 216	341 ± 70	740 ± 64	693 ± 141	179 ± 66	811±46
CARTPOLE, SWINGUP	**582±146**	297±39	326±27	311±11	-	419±40	835±22
REACHER, EASY	**538±233**	20±50	314±155	274±14	-	145±30	746±25
CHEETAH, RUN	299 ±48	138±88	235± 137	267±24	**319±56**	197±15	616±18
WALKER, WALK	**403±24**	224±48	277±12	394±22	361±73	42±12	891±82
BALL IN CUP, CATCH	**769 ± 43**	0 ± 0	246 ± 174	391± 82	512 ± 110	312± 63	746±91

source: (Laskin, Srinivas, and Abbeel, 2020)

STATE SAC uses state as observation, others uses images as observations

works very well

Self-predictive Representation (SPR)



source: (Schwarzer et al., 2020)

learn a representation that can be used to predict that of a future observation

Stability

DRL algorithms use DNNs...

benefits: great representation learning

price: no free lunch – easily overfit, unstable performance

cure? slow down update, try adding constraints and regularizers...

covered: constraining representation learning
next: constraining and regularizing policy update – TRPO, PPO, SAC, ...

Trust Region Policy Optimization (TRPO)

Policy improvement theorem, adapted from (Schulman et al., 2015)

Consider a class of parametric stochastic policy $\{\pi_\theta : \theta \in \Theta\}$, let

$$H(\theta) = \operatorname*{argmax}_{\phi} \left[\underbrace{\sum_{s,a} \rho_{\pi_\theta}(s)\pi^s_\phi(a)A_{\pi_\theta}(s,a)}_{\text{make } \phi \text{ pick good actions}} - C(\pi_\phi) \max_s \underbrace{KL(\pi^s_\theta \| \pi^s_\phi)}_{\text{make } \phi \text{ close to } \theta}, \right]$$

then

$$V(\pi_{H(\theta)}) \geq V(\pi_\theta).$$

Notations:

- $\rho_\pi(s)$ is the discounted state distribution for π

  $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s,a)$ is the advantage function of π

  $C(\pi) = \frac{2\gamma \max_{s,a} A_\pi(s,a)}{(1-\gamma)}$

  $\pi^s(a) := \pi(\cdot \mid s)$

  $KL(p\|q)$ is the KL-divergence

Schulman et al., Trust region policy optimization, 2015

theoretical TRPO policy update

$$\max_{\phi} J(\phi \mid \theta) = \mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta^s}\left[\frac{\pi_{\phi}^{s}(a)}{\pi_\theta^s(a)} A^{\pi_\theta}(s,a)\right]$$
$$\text{s.t.} \quad \underbrace{\mathbb{E}_{s\sim\rho_{\pi_\theta}}\left[KL\left(\pi_\theta^s \| \pi_{\phi}^s\right)\right] \leq \delta}_{\text{trust region}}$$

derivation from policy improvement theorem

turn KL regularizer to constraint

rewrite sum of weighted advantage as expectation

replace max KL by average KL

practical TRPO update

$$\begin{aligned} \max_{\phi} \quad & g^\top(\phi - \theta) \\ \text{s.t.} \quad & \frac{1}{2}(\phi - \theta)^\top H(\phi - \theta) \leq \delta. \end{aligned}$$

why? linearize objective, quadraticize constraint, using Taylor explansion

exact solution
$\theta + \sqrt{\frac{2\delta}{g^\top H^{-1} g}} H^{-1} g$, but may not satisfy KL constraint

approximate solution
$\theta + \alpha\sqrt{\frac{2\delta}{g^\top H^{-1} g}} H^{-1} g$, with $\alpha \in (0, 1)$ determined by backtracking, and $H^{-1} g$ computed using conjugate gradient

Algorithm TRPO (adapted from OpenAI Spinning Up)

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0

2: Hyperparameters: KL-divergence limit δ, backtracking coefficient α, maximum number of backtracking steps K

3: **for** $k = 0, 1, 2, ...$ **do**

4: Collect trajectories $\tau_1, \ldots, \tau_N$ by running policy π_{θ_k}.

5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

6: Estimate policy gradient as $\hat{g}_k = \frac{1}{N} \sum_i \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t$.

7: Compute $\hat{x}_k \approx \hat{H}_k^{-1} \hat{g}_k$ using conjugate gradient, where $\hat{H}_k$ is the Hessian of the sample average KL.

8: Update policy by backtracking for smallest $j \in \{0, 1, 2, ...K\}$ such that

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{\hat{x}_k^T \hat{H}_k \hat{x}_k}} \hat{x}_k$$

improves the sample loss and satisfies the sample KL-divergence constraint.

9: Fit value function by least squares regression on rewards-to-go $\hat{R}_t$:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{NT} \sum_i \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

TRPO is on-policy, works for both discrete or continuous A.

Proximal Policy Optimization (PPO)

TRPO makes updates more stable, but very complex!

PPO (Schulman et al., 2017) also makes updates stable, but much simpler.

PPO-penalty: uses a KL-regularizer with adaptive regularization strength

PPO-clip: clip the objective function to prevent large update

Schulman et al., Proximal Policy Optimization Algorithms, 2017

TRPO objective

$$\max_{\phi} \mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta^s} \left[\frac{\pi_\phi^s(a)}{\pi_\theta^s(a)} A^{\pi_\theta}(s,a) \right]$$

subject to the trust region constraint

PPO-clip objective

$$\max_{\phi} \mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta^s} \min\left(\frac{\pi_\phi^s(a)}{\pi_\theta^s(a)} A^{\pi_\theta}(s,a), \mathrm{clip}\left(\frac{\pi_\phi^s(a)}{\pi_\theta^s(a)}, 1-\epsilon, 1+\epsilon \right) A^{\pi_\theta}(s,a) \right)$$

no constraint

why clipping keeps the update small?

- positive A: larger ratio $r = \frac{\pi_\phi^s(a)}{\pi_\theta^s(a)}$ preferred, but no incentive to go above $1+\epsilon$.
- negative A: smaller r preferred, but no incentive to go below $1-\epsilon$.

Algorithm PPO-Clip (OpenAI Spinning Up)

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, ...$ **do**
3: Collect trajectories $\tau_1, \ldots, \tau_N$ by running policy π_{θ_k}.
4: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
5: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \underset{\theta}{\text{argmax}} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta^s(a)}{\pi_{\theta_k}^s(a)} A^{\pi_{\theta_k}}(s, a), \text{clip} \left(\frac{\pi_\theta^s(a)}{\pi_{\theta_k}^s(a)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right),$$

typically via stochastic gradient ascent with Adam.
6: Fit value function by least squares regression on rewards-to-go $\hat{R}_t$:

$$\phi_{k+1} = \underset{\phi}{\text{argmin}} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

PPO is on-policy

Few-shot RL

few/one/zero-shot learning: learn from few/one/zero examples

how is this possible?

- *domain adaptation/transfer learning: pre-train a model, adapt to a new domain*
- *meta-learning: train a model using multiple tasks, picking up task-agnostic patterns and task-specific patterns at the same time*

A Survey of Zero-shot Generalisation in Deep Reinforcement Learning

Robert Kirk ROBERT.KIRK.20@UCL.AC.UK
University College London, Gower St, London WC1E 6BT, United Kingdom

Amy Zhang AMYZHANG@FB.COM
University of California, Berkeley, Berkeley CA, United States
Meta AI Research,

Edward Grefenstette E.GREFENSTETTE@UCL.AC.UK
University College London, Gower St, London WC1E 6BT, United Kingdom

Tim Rocktäschel TIM.ROCKTASCHEL@UCL.AC.UK
University College London, Gower St, London WC1E 6BT, United Kingdom

Abstract

The study of zero-shot generalisation (ZSG) in deep Reinforcement Learning (RL) aims to produce RL algorithms whose policies generalise well to novel unseen situations at deployment time, avoiding overfitting to their training environments. Tackling this is vital if we are to deploy reinforcement learning algorithms in real world scenarios, where the environment will be diverse, dynamic and unpredictable. This survey is an overview of this nascent field. We rely on a unifying formalism and terminology for discussing different ZSG problems, building upon previous works. We go on to categorise existing benchmarks for ZSG, as well as current methods for tackling these problems. Finally, we provide a critical discussion of the current state of the field, including recommendations for future work. Among other conclusions, we argue that taking a purely procedural content generation approach to benchmark design is not conducive to progress in ZSG, we suggest fast online adaptation and tackling RL-specific problems as some areas for future work on methods for ZSG, and we recommend building benchmarks in underexplored problem settings such as offline RL ZSG and reward-function variation.

Kirk et al., A survey of zero-shot generalisation in deep reinforcement learning, 2023

Does Zero-Shot Reinforcement Learning Exist?

Ahmed Touati, Jérémy Rapin, Yann Ollivier *

March 2, 2023

Abstract

A *zero-shot RL agent* is an agent that can solve *any* RL task in a given environment, instantly with no additional planning or learning, after an initial reward-free learning phase. This marks a shift from the reward-centric RL paradigm towards "controllable" agents that can follow arbitrary instructions in an environment. Current RL agents can solve families of related tasks at best, or require planning anew for each task. Strategies for approximate zero-shot RL have been suggested using successor features (SFs) [BBQ+18] or forward-backward (FB) representations [TO21], but testing has been limited.

After clarifying the relationships between these schemes, we introduce improved losses and new SF models, and test the viability of zero-shot RL schemes systematically on tasks from the Unsupervised RL benchmark [LYL+21]. To disentangle universal representation learning from exploration, we work in an offline setting and repeat the tests on several existing replay buffers.

SFs appear to suffer from the choice of the elementary state features. SFs with Laplacian eigenfunctions do well, while SFs based on auto-encoders, inverse curiosity, transition models, low-rank transition matrix, contrastive learning, or diversity (APS), perform unconsistently. In contrast, FB representations jointly learn the elementary and successor features from a single, principled criterion. They perform best and consistently across the board, reaching 85% of supervised RL performance with a good replay buffer, in a zero-shot manner.

Touati, Rapin, and Ollivier, Does Zero-Shot Reinforcement Learning Exist?, 2022

Roadmap

- Introduction and overview

  motivation, bandits, big picture

- Classical ideas

  temporal difference methods, policy gradient, ...

- Deep Reinforcement learning

  neural networks, DQN, DDPG, ...

- Advanced techniques

  representation learning, stabilization, few-shot learning

- Applications

  AlphaGo, AlphaTensor, ...

References I

- Kirk, R. et al. (2023). A survey of zero-shot generalisation in deep reinforcement learning. In: *Journal of Artificial Intelligence Research* 76, pp. 201–264.
- Laskin, M., A. Srinivas, and P. Abbeel (2020). CURL: Contrastive unsupervised representations for reinforcement learning. In: *International conference on machine learning*. PMLR, pp. 5639–5650.
- Schulman, J. et al. (2015). Trust region policy optimization. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1889–1897.
- Schulman, J. et al. (2017). Proximal Policy Optimization Algorithms. In: *arXiv preprint arXiv:1707.06347*.
- Schwarzer, M. et al. (2020). Data-efficient reinforcement learning with self-predictive representations. In: *arXiv preprint arXiv:2007.05929*.
- Touati, A., J. Rapin, and Y. Ollivier (2022). Does Zero-Shot Reinforcement Learning Exist? In: *arXiv preprint arXiv:2209.14935*.