Sparse Rewards

Eduardo Morales, Hugo Jair Escalante

INAOE

Outline

Introduction

- One of the main challenges in RL is how to deal with sparse rewards, specially in large search spaces
- Several strategies have been proposed
  1. Curriculum learning/Goal-based methods
  2. Reward shaping
  3. User's demos, user's feedback, ...
  4. Model-based RL
  5. Exploration strategies

Universal Value Function Approximator (UVFA)

- Is an extension to DQN/value-based RL where there is more than one goal
- If $\mathcal{G}$ is the space of possible goals, each $g \in \mathcal{G}$ has a reward function $r_g : S \times A \rightarrow R$
- Each episode starts sampling a state-action pair from a sampling distribution
- The goal remains fixed during all the episode
- At each step information from the current state and goal is given, $\pi : S \times G \rightarrow A$ and obtains a reward $r_t = r_g(s_t, a_t)$
- The function Q now depends on the state-action pair and the goal: $Q^{\pi}(s_t, a_t, g) = \mathbb{E}[R_t | s_t, a_t, g]$

Curriculum Learning

- A relevant challenge in RL is how to deal with sparse rewards
- It is common for the user to carefully design a reward function for the system to work
- Idea: Create a curriculum where the easy tasks are learned first
- Originally the order of tasks was given to the system and later schemes were proposed to create it

Hindsight Experience Replay

- Similar to UVFA the value function is trained considering several goals, however, in this case the goals are automatically generated during learning
- After a sequence of an episode $(s_0, \ldots, s_T)$, the transitions are stored, not only with the original goal, but also with a set of other goals
- A *replay* is performed with goal $m(s_T)$ (i.e., the state that was reached at the end of the episode)

Hindsight Experience Replay

- Several strategies are considered to include sub-goals:
  1. *Final*: Use the final state reached in the episode
  2. *Future*: *Replay k* random states that are part of the same episode and that occurred after the state
  3. *Episode*: *Replay k* random states from the same episode
  4. *Random*: *Replay k* random states that have been visited during training
- In general, *future*, behaves better with $k = 4$

Self-play (Alice & Bob)

- Use an asymmetric self-play with two agents (Alice and Bob) to create increasingly complex goals
- Alice starts at an initial state (s_0) and after a sequence of actions it reaches a state s_t
- In reversible environments, the task for Bob is to start at s_t and return to s_0; in reset-able environments, the goal for Bob is to start at s_0 and reach s_t

Self-play (Alice & Bob)

- The interesting part is how to define the rewards that promote Alice to put increasingly challenging tasks to Bob, but not impossible for Bob:

$$R_B = -\gamma t_B$$

  where R_B is the total reward (return) in the episode for Bob and t_B is the time it took to complete it

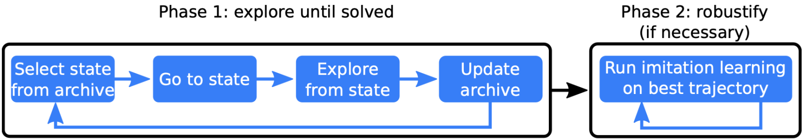
$$R_A = \gamma max(0, t_B - t_A)$$

  where R_A is the total reward (return) in the episode for Alice and t_A is the time it took to complete it
- The total time of each episode is bounded by t_{max}, if Bob does not finish the goal: $t_B = t_{max} - t_A$

Self-play (Alice & Bob)

- Alice wants Bob to take longer than her, but not much longer
- Alice does not want to take more than t_{max}, so it limits its step in order to make it simpler for Bob
- The best for Alice is to find easy tasks (small t_A) that Bob finds hard to solve (large t_B)

Go-Explore algorithm



[1]

[1] A. Ecoffet, J. Huizinga, J. Lehman, K.O. Stanley, J. Clune (2019). Go-Explore: a new approach for hard-exploration problems.

Go-Explore: Phase 1

- Explore the state space and find one or more high-performing trajectories (build archive of interestingly different states or "cells" and trajectories to reach them)
- Procedure:
  1. Choose a cell heuristically (e.g., rarely visited, contributed to new cells, better score, shorter trajectory, ...),
  2. Return to that cell
  3. Explore from that location stochastically (other exploration strategies could be used),
  4. Add to archive new cells, how to reach them, cumulative reward, and length of trajectory
- Choose an abstract representation for cells

Go-Explore: Phase 2

- Use imitation learning, learning from demonstration (LfD), e.g., the Backward Algorithm that can improve upon its demonstrations:
  - Learn a policy from a state close to the goal,
  - Back the starting state to a slightly earlier place along the trajectory, and
  - Repeat the process
- In this case, the trajectory is treated as a curriculum, where the agent tries to minimize its loss function and not to accurately mimic the trajectory

Reward Phasing and Temporal Phasing

- Idea: Introduce a set of simplified tasks with progressive complexity (similar to CL)
- Given a task that cannot be solved efficiently (T_f) and a simplified task that can be efficiently solved (T_1)
- Construct a convex combination function between these two tasks which provides a task continuum
- Learn intermediate goals

Task Phasing algorithm

Input: initial (simplified) task, $\mathcal{K}_s$; target (complex) task, $\mathcal{K}_f$; step size, α

Output: optimized policy for $\mathcal{K}_f$, π_f^*

$\pi^* \leftarrow$ train$(\pi^*, \mathcal{K}_s)$
$\mathcal{K} \leftarrow \mathcal{K}_s$
$\beta \leftarrow 0$
while $\mathcal{K} \neq \mathcal{K}_f$ **do**
    $\beta \leftarrow \beta + \alpha$
    $\mathcal{K} \leftarrow Con(\beta, \mathcal{K}_s, \mathcal{K}_f)$
    $\pi^* \leftarrow$ re-train$(\pi^*, \mathcal{K}_\beta)$
end
return π^* [2]

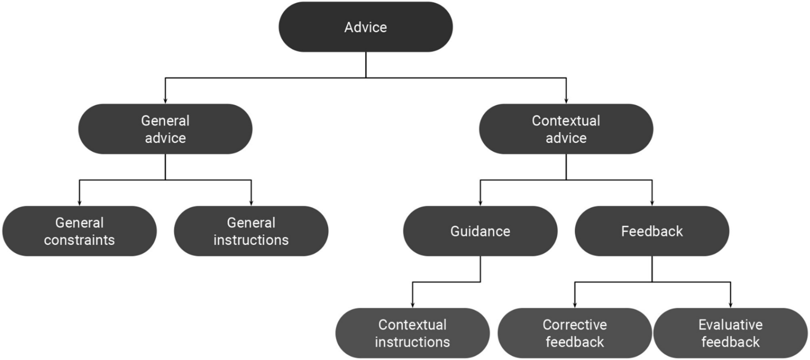
[2] V. Bajaj, G. Sharon, P. Stone (2022). Task Phasing: Automated Curriculum Learning from Demonstrations. arXiv:2210.10999v1

User's intervention

- Teaching a machine through interaction has been around AI for many years (e.g., the Child Machine - Turing 1950)
- There is a large number of possible teaching signals: instructions, demonstrations, suggestions, ...
- Common approaches: (i) Learning from advice, (ii) learning from evaluative feedback, and (iii) learning from demonstrations

Sparse Rewards
Eduardo Morales, Hugo Jair Escalante
Introduction
Curriculum Learning
Reward Shaping
User's Intervention
Model-based RL
Exploration Strategies

Learning from advice



[3]

- General advice can specify general constraints or instructions
- Contextual advice can provide guidance or feedback

[3] A. Najar, M. Chetouari (2021). Reinforcement Learning with Human Advice: A Survey. Frontiers in Robotics and AI.

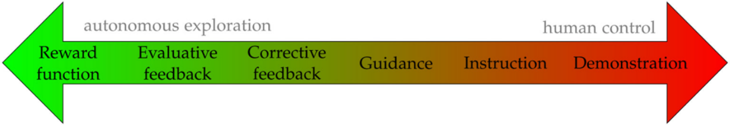
Learning from advice

- The advice has to be interpreted, hand-coding the mapping
- The interpretations can be learned from examples to covert the advice, for instance, into rules
- Can be given as pairs of demonstrations and descriptions of the demonstrations
- Use RL to interpret advice

Shaping with advice

- Advice can be integrated into the learning process at the reward function, the value function, the policy or the decision
- In Reward shaping the advice is translated into intermediate rewards (that may include a decay factor)
- Value shaping can be given in the form of constraints on action values (i.e., If cond then Q $>$ value), action preferences (e.g., If cond then prefer a1 over a2), or to modify the value function (e.g., Q = Q + shape)
- Policy shaping can integrate the advice directly into the policy, scaling the gradient policy with a shaping factor, include an additional term in the TD error, or in the selection of the action
- In the Decision, the advice can directly bias the output of the policy (e.g., "turn right")

Learning from advice



[4]

- Evaluative feedback: Normally simpler and easier to provide (e.g., good, better, ...)
- Corrective feedback: Require encoding the mapping between the instruction and the action, but can also include correcting demonstrations
- Guidance: Inform the agent of future aspects of the task (e.g., next action to perform, a region to explore, actions to try, ...)
- Instructions: Inform more directly about optimal actions
- Demonstration: Define a sequence of state-action pairs representing a solution to the task.

[4] A. Najar, M. Chetouari (2021). Reinforcement Learning with Human Advice: A Survey. Frontiers in Robotics and AI.

Learning Shaping

- Shaping can be difficult to define by hand
- Shaping can be specify in terms of distances to goal
- DDL idea: Distance evaluation and policy improvement
- Roll out the policy multiple times to sample trajectories
- Learn a parameterized distance function between pair of states visited by a given policy
- Learn a policy using the learned distance as a negative reward
- Select as goal a vicinity of a previously visited goal/state or selected by the user

Bi-level Optimization

- Defining an adequate shaping reward function can be difficult
- Idea: Use a bi-level optimization approach:
  - Low Level: Optimize a policy using the current reward shaping function
  - High Level: Optimize a parameterized reward shaping function
- Use an alternating optimization method: optimize one and then optimize the other, and continue

DDL Algorithm

Input: ϕ, ψ ▷ Initial policy and distance parameters
Input: $\mathcal{D}$ ▷ Empty replay pool
repeat
  $\tau \sim \rho_\pi,\ \mathcal{D} \leftarrow \mathcal{D} \cup \tau$ ▷ Sample a new trajectory
  for $i = 0$ **to** N_d **do**
    $\psi \leftarrow \psi - \lambda_d \hat{\nabla} \mathcal{L}_d(\psi; \pi)$ ▷ Minimize distance loss
  end for
  $\mathbf{g} \leftarrow \text{choose\_goal}(\mathcal{D})$ ▷ Choose goal state
  for $i = 0$ **to** N_π **do**
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla} \mathcal{L}_\pi(\phi; d, \mathbf{g})$ ▷ Minimize policy loss
  end for
until converged [5]

[5] K. Hartikainen, X. Geng, T. Haarnoja, S. Levine (2020). Dynamical distance learning for semi-supervised and unsupervised skill discovery. In Proc. ICLR.

Imitation Learning

- Imitation learning can be implemented by a supervised learning approach, finding a policy that follows closely the traces given by an expert
- DAgger (Dataset Aggregation) trains a deterministic policy.
- First it gathers data using the expert's policy and trains a new policy that mimics the expert on those trajectories
- At iteration "n" is uses the π_n policy to collect more trajectories and add those trajectories to the dataset, the next policy (π_{n+1}) is the policy that best mimics the expert of the whole dataset

DAgger algorithm

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation. [6]

[6] S. Ross, G. Gordon, J.A. Bagnell (2011). A reduction of imitation learning and structure prediction to no-regret online learning. arXiv: 1011.0686v3.

Model-based RL

- The idea is to learn/use a model (transition function, reward function, causal model) that can be used for planning, guide the exploration process, ...
- In many cases, the goal is to lean a dynamic model, which can be: (i) forward model $(s_t, a_t) \rightarrow s_{t+1}$, (ii) backward/reverse model $s_{t+1} \rightarrow (s_t, a_t)$, (iii) inverse model $(s_t, s_{t+i}) \rightarrow a_t$
- The system needs to decide when/how to use the model: In all states (DP), reachable estates (DynaQ), prioritized states (Prioritized Sweeping), current state (AlphaGo)
- Also decide when to learn a model and how much data to use: After each episode, after N episodes, batches

Exploration Strategies

- All RL algorithms follows some form of exploration strategy, the most common being $\epsilon-$ greedy
- An adequate exploration strategy is required with sparse rewards
- In RL exploration means how to find useful rewards
- This is challenging, as rewards can be sparse, there may be local minima (maxima) or areas of flat rewards

Exploration Strategies

- Exploration can be divided into: Efficiency exploration (sample efficient) and Safe exploration (ensure safety during exploration)
- Efficiency-based exploration can be further divided into: (i) Imitation-based (use a policy from an expert) and (ii) self-taught methods
- Self-taught methods can be further divided into: (i) Planning methods, (ii) intrinsic rewards, and (iii) random methods
- Planning methods can be: (i) Goal-based and (ii) probabilistic
- Intrinsic reward can be: (i) Reward diverse behaviors and (ii) reward novel states

Reward Novel States

- Rewards are given for discovering new states (intrinsic reward) and can be divided into:
  1. Prediction error methods: The intrinsic reward is measured as the distance between the prediction of a state from a model and that state

     $$r_{int} = f(z(s_{t+1}) - M(z(s_t, a_t)))$$

     They can be further divided into: State representation prediction (using, for instance, auto-encoders, random network distillation, inverse dynamic features) and uncertainty about the environment (using a Bayesian approach, ensembles, information-gain)
  2. Count-based methods: Each state has associated a counter with the number of visits, low count means high intrinsic reward
  3. Memory methods: They are based on how easy it is to distinguish one state from another, the easiest the higher the intrinsic reward

Reward Diverse Behaviors

- The idea is to collect as many different experiences as possible, the objective is exploration rather than finding rewards
- Approaches:
  - Evolutionary methods: Use EA to obtain diverse samples (population)
  - Policy learning: Measure the diversity among several policies, e.g., using KL divergence. Idea: to have as many diverse policies as possible (e.g., Diversity is all you need (DIAYN))

Conclusions

- How to deal with sparse rewards is an active research area
- Several methods have been proposed and in many cases more than one method is used in the same algorithm
- Other related areas not covered in these notes: Hierarchical RL, MARL, ...

References

1. P. Ladosz, L. Wang, M. Kim, H. Oh (2022). Exploration in Deep Reinforcement Learning: A Survey
2. A. Najar, M. Chetouari (2021). Reinforcement Learning with Human Advice: A Survey. Frontiers in Robotics and AI.
3. S. Ross, G. Gordon, J.A. Bagnell (2011). A reduction of imitation learning and structure prediction to no-regret online learning. arXiv: 1011.0686v3.
4. K. Hartikainen, X. Geng, T. Haarnoja, S. Levine (2020). Dynamical distance learning for semi-supervised and unsupervised skill discovery. In Proc. ICLR.
5. V. Bajaj, G. Sharon, P. Stone (2022). Task Phasing: Automated Curriculum Learning from Demonstrations. arXiv:2210.10999v1
6. T.M. Moerland, J. Broekens, A. Plaat, C.J. Jonker (2022). Model-based reinforcement learning: A survey. arXiv:2006.16712v4