Floyd-Warshall Reinforcement Learning: Combining Advantages from Model-Free and Model-Based RL Improving Performance in Multi-Goal Environments

Vikas Dhiman, Shurjo Banerjee, Jeffrey M. Siskind and Jason J. Corso

Abstract

Reinforcement learning algorithms are oftentimes classified as either *model-free* or *model-based* conditioned on whether a state-transition function is learned implicitly or explicitly. In the *model-free* case, algorithms such as Q-learning and policy gradients attempt to learn the optimal state-action value function that maximizes expected future reward for all state-action pairs percieved by an agent in an environment. The reward distribution and state dynamics that governs the agent's environment are implicitly memorized in the learning of this function. Due to the implicit nature of this process, *model-free* RL struggles with transferring learned behaviours to environments in which only the underlying reward distribution changes. In contrast, *model-based* RL explicity learns the state transition function allowing for the decoupling of an agent's learned behaviours to environments in which only the reward distribution changes. Oftentimes, this explicit modelling requires an additional planning step to predict state-transition dynamics making policy computation in such algorithms an expensive process. Inspired by both these paradigms and the Floyd-Warshall algorithm for path-planning on graphs, this work introduced Floyd-Warshall Reinforcement Learning (FWRL), a novel algorithm that combines advantages from both *model-based* and *model-free* approaches. The algorithm works by learning a goal-conditioned value function that transitions from model-based behavior to model-free behavior in well explored regions of an agent's state space. FWRL is shown to transfer knowledge about an environment when the reward location is dynamic compared to a model-based Q-learning baseline. FWRL is shown to meet the ground between model-free and model-based algorithms by being model-free in the more frequently visited regions while being model-based on less visited paths.

Introduction

Reinforcement learning is an attractive field of research as it allows for agents to learn complex, autonomous behaviors in a multitude of environments while requiring minimal supervision in the form of reward signals. This is readily evidenced from RL's recent well publicised successes from goal-agnostic activites such as playing atari games from purely visual input and defeating world GO and starcraft champions, to goal-driven ones such as recent applications in robotic navigation and manipulation. In the realm of goal-conditioned value functions, this work introduces Floyd-Warshall Reinforcement Learning, a new framework intended for goal-driven domains with specific consideration for transferring learned behaviours to environments in which the underlying reward distribution is dynamic.

Algorithms that underly RL are often classified as being either *model-based* or *model-free*, the distinction being whether an environment state-transition function is learned explicitly or implicitly. In *model-based* RL, the dynamics that govern an environment's transitions is modelled as separate step for policy computation. At any point in an episode, agent's use this model to predict future states and utilize this information to maximize possible reward. This formulation is known to be sample-efficient while normally not achieving asymptomatic performance. Due to the requirement of planning steps to predict future states, policy computation can be an expensive process. In contrast, in *model-free* RL, algorithms such as policy gradients and Q-learing learn the optimal state-action value function that maximizes expected future reward for every state-action tuple that the agent perceives. While highly sample-inefficient, agents trained under this paradigm have been shown to achieve asymptomatic performance in a variety of different problem spheres.

Both paradigms of RL suffer different disadvantages in transferring learned behaviors to *reward-dynamic* settings i.e. environments in which the underlying reward distrbution changes. While model-based RL allows for the separation of environment dynamics and reward, small errors in the modelling function lead to significant drops in performance. In *model-free* RL, on the other hand, the conflated representation of environment and reward makes any form of transfer difficult. These problems are exacerbated in multi-goal settings.

Inspired by the idea of combining advantages from both RL paradigms and the Floyd-Warshall algorithm in graph-based path planning, this work introduces Floyd-Warshall Reinforcement Learning, a new framework for combining model-based and model-free RL in goal-driven reward-dynamic settings. **Two line description of floyd-warshall's workings. don't think there should be equations.**

Ideas of combining both RL paradigms to improve agent performance are not new. In Temporal-Difference modelling, etc. et al derive's an exact relationship between both both of them. In Hindsight Experience Replay, previous

episode experience is used augment and bootstrap learning.

Experimentally, FWRL is shown to outperform both model-based, model-free and above combintations thereof in both a tabular and neural-network based setting. FWRL is found to outperform the next most significant baselines by as much x%.

In summary, this work introduces Floyd-Warshall Reinforcement Learning outperforming several strong baselines on a varierty of multi-goal reward-dynamic environments. The experimentation suite and all code is made available.

Claims

- Using FWRL is better than model-free learning like Q-learning in terms of generalizing to dynamic goals, static maps.
- Using FWRL is better than model-based learning because of less accumulation over long term, hence more accurate value functions.
- FWRL is better than Hindsight Experience Replay (HER) Andrychowicz and Kurach (2016) because it is a parametric representation of hindsight.
- FWRL is better than TDM Pong et al. (2018) because it does not require the distance reward function.

Related work

Goal-conditioned value functions

The idea of goal-conditioned value function is not new but has got attention because of revival of reinforcement learning based on deep neural networks. We build upon the goal-conditioned value functions of Schaul et al. (2015). Schaul et al. (2015) proposed an architecture and a matrix factorization based algorithm for faster learning of UVFA (Universal value function approximators). UVFA focused on fast estimation of goal-conditioned value functions using sparse matrix factorization but not on bridging the gap between model-based and model-free algorithms.

Andrychowicz and Kurach (2016) introduced the idea of Hindsight experience replay (HER) two learn about the model from previous episodes even when the goal location has changed or not been achieved. Our method can be seen as parametric approximation of "Hindsight memory", that can help compress information from previous episodes instead of maintaining the entire history of replay memory.

Pong et al. (2018) propose temporal difference models that estimate goal directed Q function for a specific kind of reward function, in particular the distance from the goal and in contrast with limited temporal horizon.

Combining model-based and model-free methods

Navigation with mapping

(1) CMP from Saurabh Gupta: is metric, might not working in continuous spaces. (2) Semi-parameteric Topological mapping: is not end to end. (3) Neural Map: Is actually not mapping

Model free DRL

does not generalize to multi-goal environments.

Model based DRL

Needs more exploration. Find the paper that shows that Model based DRL can actually compete with Model free DRL as long as it models uncertainty.

Multi-goal navigation based papers

Mirowski 2017, 2018: No one shot map learning, does not generalizes to new maps.

Problem definition

Let the agent and its environment be represented by a state space $\mathcal{S}$ and the agent can traverse through state space by taking actions $a \in \mathcal{A}$ in a fixed action space $\mathcal{A}$. At every time step t, the agent takes action a_t and observes state $s_t \in \mathcal{S}$ and reward $r_t \in [-R_{\text{goal}}, R_{\text{goal}}]$. Consider an episode of T time steps. For every episode the environment chooses one of the states in the state space is the goal state $g \in \mathcal{S}$. We want to find the sequence of actions to take that maximizes the total reward over T time steps. Once the agent reaches the goal $\|s_t - g\| < \delta$, the agent receives the highest reward possible R_{goal} in the game and gets respawned in the environment at a randomg *spawn* state. This cycle continues enabling the agent to reach the goal multiple times, incentivizing rapid exploration and finding the shortest path to the goal. For the new episode a new goal location is chosen, but rest of the environment stays the same. The agent is allowed to remember about the environment from previous episode and use it for this episode.

$$\chi^*(s_t; g) = a_t^* = \arg\max_{a_t} \mathbb{E}_\chi \left[\sum_{t=0}^{T} r_t \right] \tag{1}$$

Why is this problem important?

Many real world problems can be formulated in this context. Consider a traveling postman problem who has moved into a new city. The postman has to explore the city and find the buildings that match the given address. The next time the postman gets the same address, they can use their experience to find out the building. Even when a new address is provided (in the next episode), the postman can use experience to find the new episode more quickly.

In robotics, tasks like picking and placing the object at a desired location can be formulated as goal-directed navigation.

Why is the problem hard?

Model-free Reinforcement learning methods assume that the rewards are being sampled from the a static reward function. In a problem where the goal location changes, hence the reward function also changes, it becomes hard to transfer the learned value-function or action-value function to the changed location. One alternative is to concatenate the goal location the state, making the new state space $[s_t, g]^\top \in \mathcal{S}^2$ larger. This method is wasteful in computation and more importantly in sample complexity.

Method

We present a model-free reinforcement learning method that easily transfers when goal location is dynamic. We accomplish this by maintaining a path based expected reward function from any state to any goal state. We call this algorithm Floyd-Warshall Reinforcement Learning, because of its similarity to Floyd-Warshall algorithm : a shortest-path planning algorithm on graphs. We define Floyd-Warshall value function as

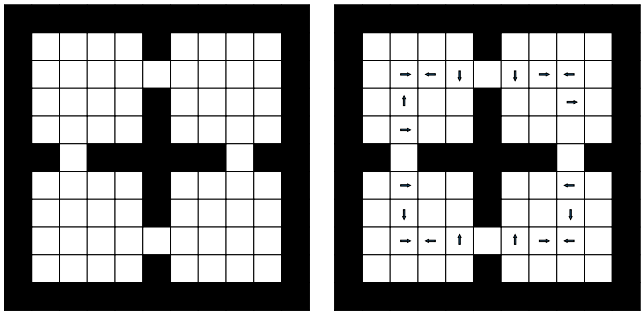

Figure 1: Left: Four room grid world. Right: Four room windy grid world with wind direction shown by arrows. The windy pushes the agent in the direction of wind with 0.25 probability irrespective of the action taken.

$$F_\chi(i, l, j) = \mathbb{E}_\chi \left[\sum_{t=0}^{t=k} r_t \middle| s_0 = i, a_0 = l, s_k = j \right]. \tag{2}$$

When the policy is optimal, the Floyd-Warshall function should satisfy the constraint

$$F^*(s_i, a_i, s_j) = \max_{s_k} \left[F^*(s_i, a_i, s_k) + \max_{a_k} F^*(s_k, a_k, s_j) \right]. \tag{3}$$

We summarize the algorithm in Alg 1.

Experiments

We setup two environments in the grid world and windy grid world.

Four room grid world

Four room grid world is a grid world with four rooms connected to each other as shown in Figure 1.

Four room windy world

Four room windy world is a grid world with four rooms connected to each other as shown in Figure 1. Some of the grid cells in have wind shown by arrow and the agent gets pushed around by the wind with 0.25 probability irrespective of the action taken.

Results

Fig 2 and Fig 3 show the results.

Future work

Items to improve the algorithm:

- TODO: Justify the computational cost of constraint The cost of going through the entire state space. How do you extend to a network? and large state spaces.
  1. Observation 1: If there is only one goal, then the computation should not be any more than Q-learning. This can be accomplished by assuming that transitivity is

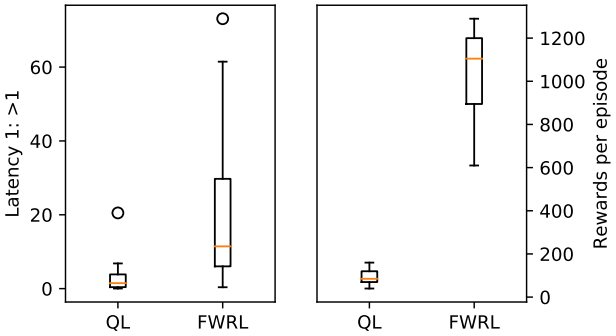

Figure 2: Results on Grid world

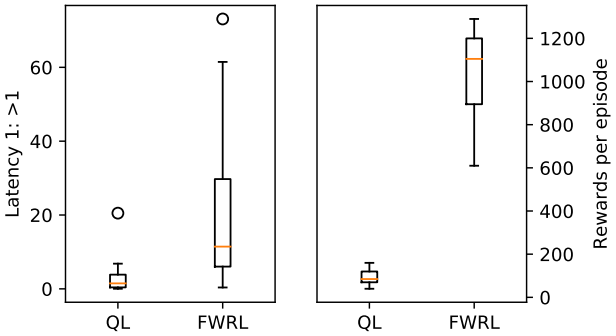

Figure 3: Results on windy world

satisfied till s_{t-1} and needs to be extend to only the next step. This sounds similar to the Floyd-Warshall dynamic programming update. However, this assumes that s_t is being visited for the first time. If the state s_t is being visited for the second time, the earlier value may be the shorter path for it.

- We need Q-value for exploration.

Function $\chi^*(s_t, s_g, Q(.,.), F(.,.,.))$

if $s_g = \phi$ *or all*$(F(s_t, :, s_g) = -\infty)$ **then**
  `/* Exploration mode */`
  $a^* = \arg\max_a Q(s_t, a)$;
else
  `/* Exploitation mode */`
  $a^* = \arg\max_a F(s_t, a, s_g)$;
return a^*

Algorithm 1: Floyd-Warshall Reinforcement Learning (Tabular setting)

Let $r_g \leftarrow 10$;
`/* By default all states are unreachable */`
Initialize $F(s_i, a_i, s_j; \theta_F) \leftarrow -\infty$;
Initialize $Q(s_i, a_i) \leftarrow 1$;
Initialize $s_g = \phi$;
Set $t \leftarrow 0$;
Observe z_t ;
$s_t = \Phi_o(z_t; \theta_E)$;
for $t \leftarrow 1$ **to** T **do**
  `/* See Function` χ^* `*/`
  Take action $a_{t-1} \sim$ Egreedy$(\chi^*(s_{t-1}, s_g, Q, F))$;
  Observe z_t, r_t ;
  $s_t = \Phi_o(z_t; \theta_E)$;
  if $r_t >= r_g$ **then**
    `/* Reached the goal */`
    $s_g \leftarrow s_t$;
    `/* Respawning does not need update of value functions */`
    continue;
  $Q(s_{t-1}, a_{t-1}) \leftarrow r_t + \max_a Q(s_t, a)$;
  $F(s_{t-1}, a_{t-1}, s_t) \leftarrow r_t$;
  for $s_k \in \mathcal{S}, a_k \in \mathcal{A}, s_l \in \mathcal{S}$ **do**
    $F(s_k, a_k, s_l) \leftarrow$
    $\max\{F(s_k, a_k, s_l), F(s_k, a_k, s_t) +$
    $\max_{a_p \in \mathcal{A}} F(s_t, a_p, s_l)\}$;
Result: $\chi^*(s_k, s_g, Q, F)$

- Grid world: Set up a random goal static maze scenario, compare with normal Q-learning.
- Lava world: Set up a Lava world like Schaul et al. (2015) and test on it.
- TODO: Atari games: Compare performance with normal Q-learning.

- TODO: Deepmind Lab: Set up a random goal static maze scenario, compare with normal Q-learning.

References

Andrychowicz, M., and Kurach, K. 2016. Learning efficient algorithms with hierarchical attentive memory. *arXiv preprint arXiv:1602.03218*.

Pong, V.; Gu, S.; Dalal, M.; and Levine, S. 2018. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*.

Schaul, T.; Horgan, D.; Gregor, K.; and Silver, D. 2015. Universal value function approximators. In *International Conference on Machine Learning*, 1312–1320.