Floyd warshall deep reinforcement learning

Abstract

Problem: Multi-goal navigation without mapping
Model free deep reinforcement learning is to learn $\chi(a|s)$ or $Q(s,a)$
Model based deep reinforcement learning is to learn $P_T(s_{t+1}|s_t,a_t)$ and $V(s)$. and planning on a graph to get the shortest path. Note that $P_T(.)$ is highly sparse and keeping a list of non-zero s_{t+1} is much better than keeping all the value of $P_T(.)$ for all $s_{t+1} \in \mathcal{S}$.
Floyd-Warshall deep reinforcement learning is to generalize model based DRL to directly learn the $F(s_j|s_i,a_i)$ which is the cost of reaching state s_j starting from s_i when the first action taken is a_i. If we directly try to learn $F(.)$ we are likely to get conflicting results that do not obey FW identity $F(s_j|s_i,a_i) = \min_{s_k} \min_a F(s_j|s_k,a) + F(s_k|s_i,a_i)$
It is expected that since we will be visiting nearby states more often, so the $F(.)$ will be consistent over small distances but will grow inconsistent over large distances. We can draw few samples from $F(.)$ to check for it's inconsistencies and then plan over graph over higher ranges.

Claims

- Using Floyd Warshall value function leads to better generalization in case of static maps and random goals.
- Hypothesis: Multi-goal navigation is more common than we think. Does FW algo improves performance in attari games.

Related work

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.

Method

See Alg **??** Over simplified. Ignoring the cost of going through the entire state space.

Algorithm 1: How to solve small windy grid world with randomized goals?

```
Data: Graph G_0 = (V, E);
Initialize F(s_i, a_i, s_j; θ_F) = 100 ;
Initialize Q(s_i, a_i; θ_Q) = 1 ;
Initialize α_V = 0.1, α_= 0.9 ;
Let minimum path cost F_0 = 0.05 ;
Observe z_0 from environment ;
s_0 = Φ_o(z_0; θ_E) ;
for t ← 1 to  do
    Take action a_{t-1};
    Observe z_t, r_t;
    Encode state s_t = Φ_o(z_{1:t}; θ_E);
    /* Initialize new FW values */
    F(s, a, s_t) = min{F(s, a, s_t), F(s, a, s_{t-1}) + F_0}   ∀s ∈ S, a ∈ A ;
    /* Q-Value update */
    Q(s_{t-1}, a_{t-1}) = (1 - α_Q)(r_t + γ max_{a_k} Q(s_t, a_k)) + α_Q Q(s_{t-1}, a_{t-1});
    if s_t is visited the first time then
        for (s_i, s_k, a_k) ∈ (S × S × A) do
            /* Run the Floyd Warshall update */
            F(s_k, a_k, s_i) = min{F(s_k, a_k, s_i), F(s_k, a_k, s_t) + min_{a∈A} F(s_t, a, s_i)} ;
            Q(s_k, a_k) = max{Q(s_k, a_k), max_a Q(s_i, a) - F(s_k, a_k, s_i)} ;
Result: To follow the shortest path s_i to s_j, follow the
        neighbors with highest Q;
χ(s_k) = arg max_{a_k∈A} Q(s_k, a_k);
```

Experiments

- Grid world: Set up a random goal static maze scenario, compare with normal Q-learning.
- Deepmind Lab: Set up a random goal static maze scenario, compare with normal Q-learning.
- Atari games: Compare performance with normal Q-learning. Analyze games in which FW does better. Show that those games have dynamic goals rather than static.