Value Function Transfer for Deep Multi-Agent Reinforcement Learning Based on N-Step Returns

Liu et al.

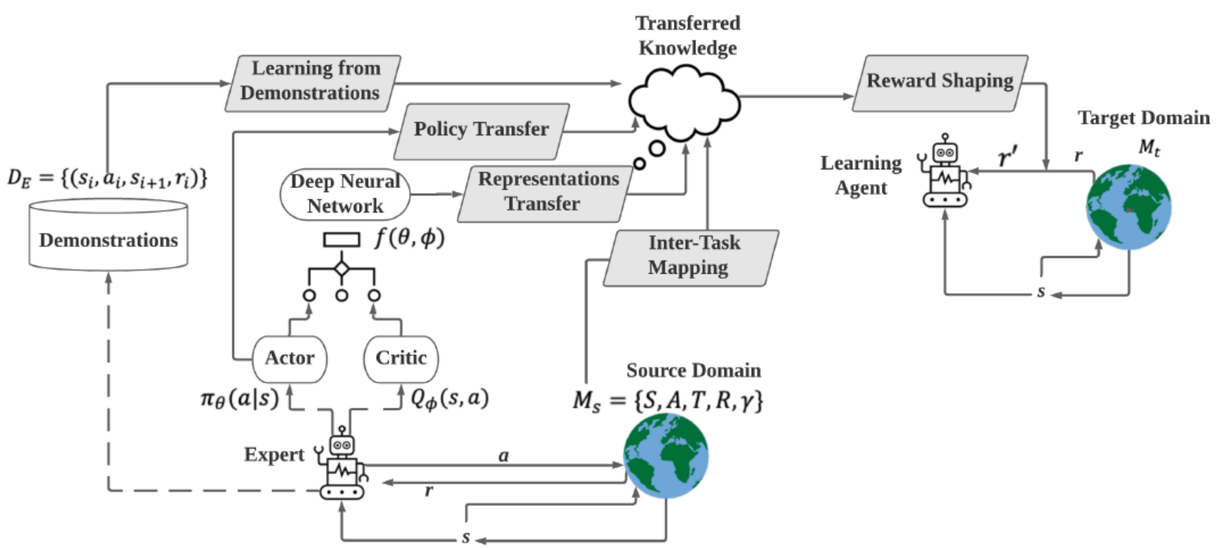
International Joint Conference on Artificial Intelligence(IJCAI) 2019

Speaker : Min Joon Kim

Feb 28th, 2024

Introduction

- **Transfer learning(TL)**
  - **(Definition)** Transfer learning is a technique in machine learning in which <u>knowledge learned from a task is re-used in order to boost performance</u> on a related task.
  - **TL in RL**
    - (1) Learning from demonstrations (2) Policy Transfer (3) Representations transfer
    - (4) Inter-task Mapping (5) Reward shaping



"Transfer learning in deep reinforcement learning: A survey" 2023

Problem definition

- **Value function transfer for deep multi-agent RL**
  - **Want to accelerate the multi-agent learning process with single agent knowledge.**

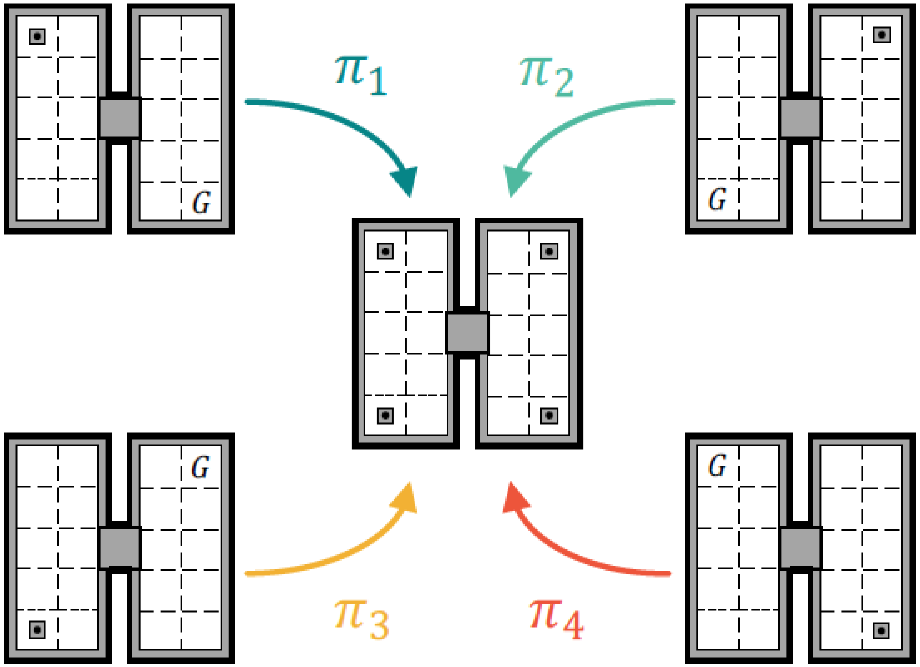


Figure 1: Transfer single-agent knowledge to multi-agent system

Key ideas

- **Propose the two knowledge transfer methods**
  - Direct value function transfer
  - N-step Return(NSR) based value function transfer
    - Using NSR value representing the MDP similarity between single-agent and multi-agent environment
  - Experiments are conducted in grid world, multi-agent particle environment and Ms. Pac-Man game.
    - Sparse interaction environment (independent agents)

Related works

<table>
<tr><th>Year</th><th>Paper</th><th>Note</th></tr>
<tr><td>2010</td><td>Learning multi-agent state space representations</td><td rowspan="3">Knowledge transfer in multi-agent
/
<u>Tabular domain</u></td></tr>
<tr><td>2011</td><td>Transfer learning for multi-agent coordination</td></tr>
<tr><td rowspan="2">2015</td><td rowspan="2">Learning in multi-agent systems with sparse interactions by knowledge transfer and game abstraction.</td></tr>
<tr><td>MDP similarity</td></tr>
<tr><td>2018</td><td>Value-decomposition networks for cooperative multi-agent learning based on team reward.</td><td>VDN</td></tr>
<tr><td>2018</td><td>QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning.</td><td>QMIX</td></tr>
</table>

- **Previous works**
  - Can be adopted only in tabular domain
  - Not much studies in value function transfer

- **Contribution**
  - Proposed value function transfer methods
  - Suggest a new MDP similarity metric using N-step return value

Direct Value Function Transfer

- Direct Value Function Transfer Network Architecture
  - Model A : Single-agent expert policy
  - Model B : Multi-agent network(QMIX)

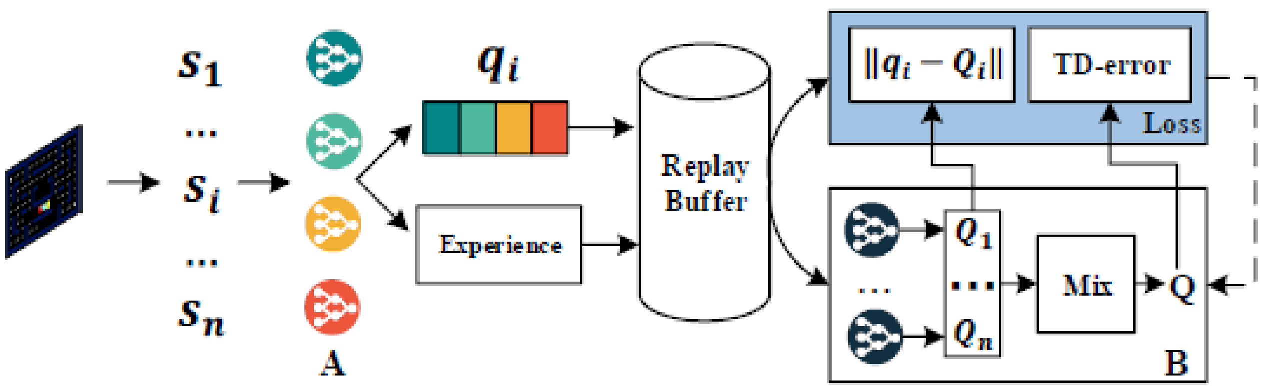


Figure 2: Direct Value Function Transfer Network Architecture. Model A represents single-agent expert policy network and model B represents multi-agent network.

Direct Value Function Transfer

- Pseudo code

Algorithm 1: Direct Value Function Transfer

Input: local value function $q_i(s_i, a)$ for each agent i, discount factor γ, exploration factor ϵ

Initialization. $Q(s, \vec{a}) \leftarrow \theta$, $\hat{Q}(s, \vec{a}) \leftarrow \theta'$, $Q_i(s_i, \vec{a}) \leftarrow \theta_i$;
foreach *episode* **do**
  Initialize state s;
  repeat
    foreach *agent* i **do**
      $(s_i, a_i) \leftarrow$ the component of agent i in $(s, \vec{a})$;
      $a_i \leftarrow$ max q_i index with ϵ-greedy policy;
    $\vec{a} \leftarrow [a_1, ..., a_n]$;
    store experience $(s, \vec{a}, r, s', done, [q_1, ..., q_n])$;
    $s \leftarrow s'$;
    Sample training experience from buffer;
    if *not done* **then**
      $y = r + \gamma \hat{Q}_{max}(s, \vec{a}; \theta')$;
    else
      $y = r$;
    $L(\theta) = \alpha \sum_1^N (q_i(s_i, a_i) - Q_i(s_i, a_i; \theta_i))^2 + (1 - \alpha)(y - Q(s, \vec{a}; \theta))^2$;
    Update θ by a gradient method w.r.t. $L(\theta)$;
    Every C steps reset $\hat{Q} = Q$;
  until $s = terminal$;

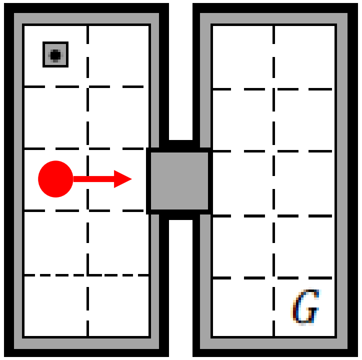
Store experience with Single-agents policy

Q-value from single agent(expert) – Q-value for agent i

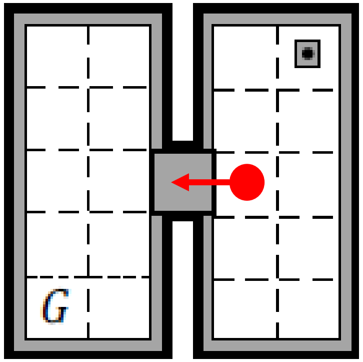
(TD loss for the global Q value)

Limitation of Direct Value Function Transfer

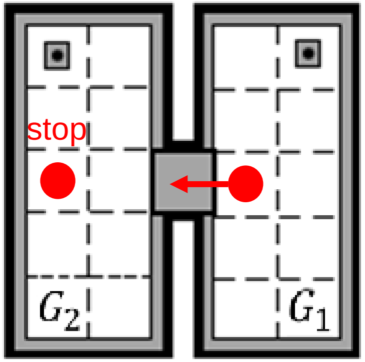
- Limitation and MDP similarity idea
  - What if the state is different between single-agent and multi-agent environment?
  - It causes negative transfer.



[Single agent1]



[Single agent2]



[Multi agent2]

  - So, they want to transfer only when the MDP is similar.
    - MDP similarity calculation : <u>single agent Q value</u> - <u>N-step reward(multi-agent)</u>

N-step Return based Value Function Transfer

- Pseudo code

Algorithm 2: NSR-based Value Function Transfer

Input: local value function $q_i(s_i, a)$ and single-agent policy π_i for each agent i, N (N $\geq$ 1), discount factor γ, exploration factor ϵ, δ

Initialization. NSR value function $\hat{\mathcal{R}}^N_{\pi_i} \leftarrow \psi_i$, Replay Buffer $\mathcal{B}_i$ for each agent i ;

foreach *episode* **do**
- Initialize state s_t, $t = 0$;
- **repeat**
  - **foreach** *agent* i **do**
    - $(s_{i,t}, a_i) \leftarrow$ the state of agent i in $(s_t, \vec{a})$;
    - $a_i \leftarrow$ max q_i index with probability δ;
    - $y_i = r_{t-N+1} + \gamma r_{t-N} + ... + \gamma^{N-1} r_t$;
    - Store $(s_{i,t-N+1}, y_i)$ in $\mathcal{B}_i$;
    - Update ψ_i by a gradient method w.r.t. $(y_i - \hat{\mathcal{R}}_{\pi_i}(s_{i,t-N+1}; \psi))^2$;
  - $t = t + 1$
- **until** $s_t =$ *terminal*;

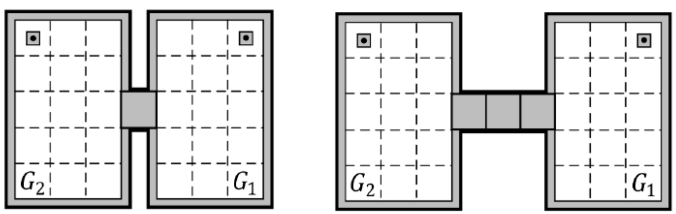
foreach *episode* **do**
- Initialize state s;
- **repeat**
  - $(s_i, a_i) \leftarrow$ the state of agent i in $(s, \vec{a})$;
  - **if** $\max_i \left| \hat{\mathcal{R}}_{\pi_i}(s_i) - q_i(s_i, \pi_i(s_i)) \right| \leq \tau$ **then**
    - $a_i \leftarrow$ argmax $q_i(s_i, a_i)$ for each agent i;
  - **else**
    - $a_i \leftarrow$ argmax $Q_i(s, a_i)$ with ϵ greedy for each agent i;
  - Learning by multi-agent method;
- **until** $s =$ *terminal*;

- Similarity threshold τ
- Transfer (models are similar)
- No transfer (models are different)

- Update the N-step return prediction network (Multi-agent environment)

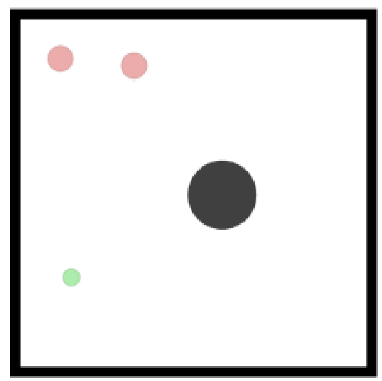
Experiments

- Expertiment settings
  - Each single agent expert policy is trained by DQN.
  - 3 environments (Grid world, Multi-agent Particle, Ms. Pacman)
  - Similarity threshold $\tau = 2, 3, 5, 7$



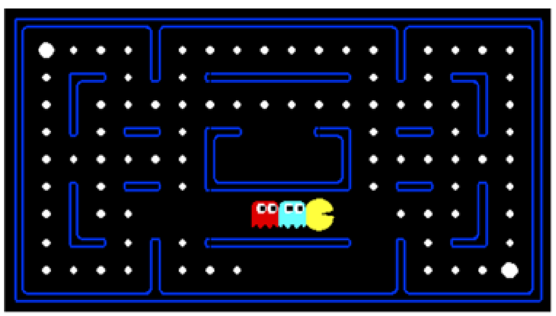
(a) map 1 (b) map 2

Grid world



(a) predator-prey

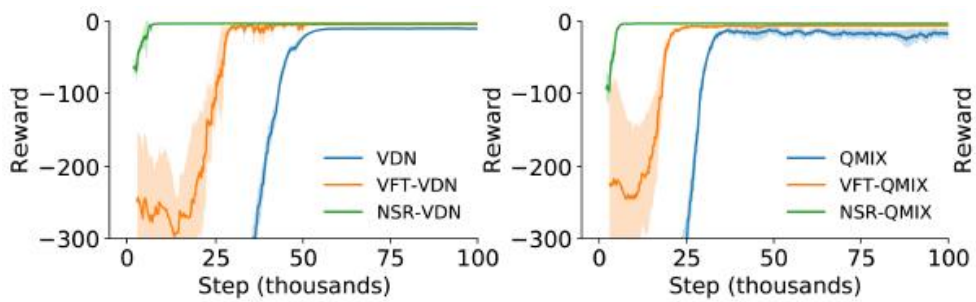
Multi-agent Particle Environment (MPE)



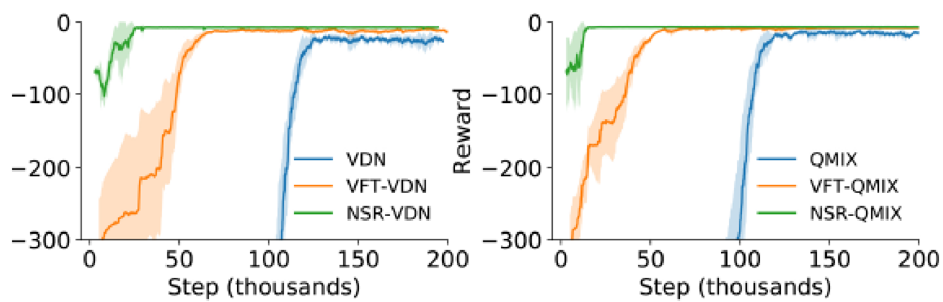
Ms. Pac-Man

Results

- Grid world
  - NSR-based transfer method showed more stable and faster convergence than others.
  - QMIX was slightly better than VDN algorithm.
  - About double steps are needed for the second environment(map2) learning.



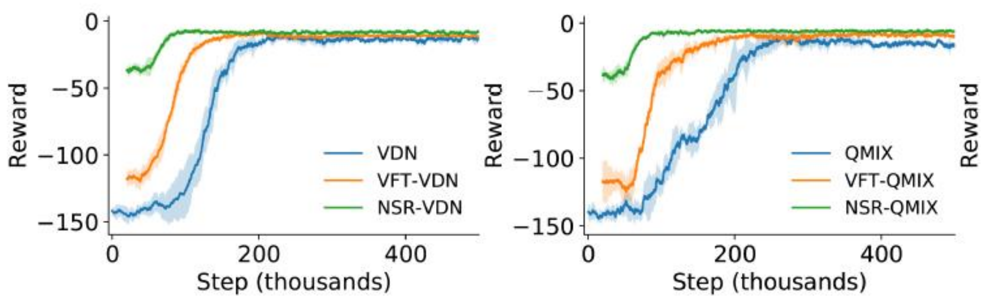
(a) map1-VDN (b) map1-QMIX



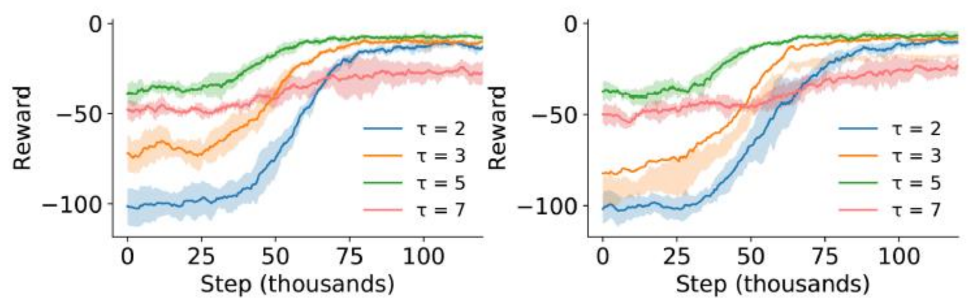
(c) map2-VDN (d) map2-QMIX

Results

- Multi-agent Particle Environment(MPE)
  - NSR-based transfer method showed much faster convergence than others.
  - Proposed method converges in 90,000 steps, while the VDN needs 200,000 steps.
  - They found the appropriate threshold($\tau = 5$) experimentally in figure (g), (h)



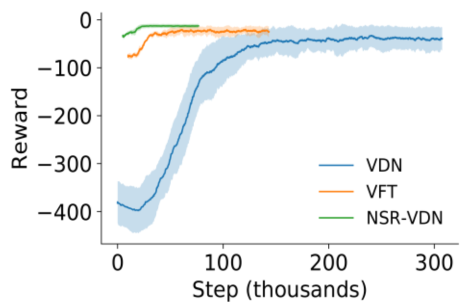
(e) MPE-VDN
(f) MPE-QMIX



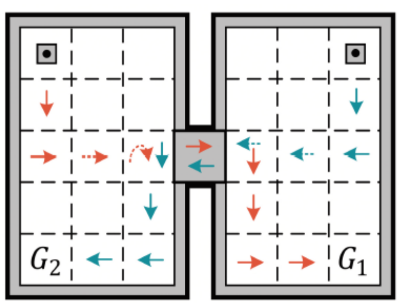
(g)
(h)

Results

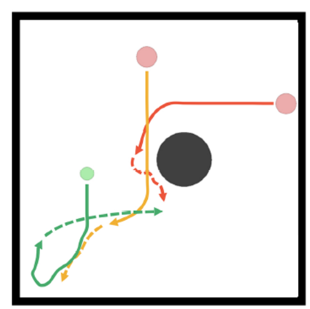
- Ms. Pac-Man environment
  - Proposed method showed best performance, but there is not much improvement compared to the direct value function transfer method.



- Learned policy (Grid world, MPE environment)
  - Learned by single-agent(solid arrow), Learned by multi-agent(dotted arrow)



(b) Policy Display



(b) Policy Display

Conclusion

- **Conclusion**
  - They suggested value function transfer method for the Multi-agent RL.
  - Proposed method showed faster and stable convergence than MARL algorithms.
  - Achieved about 2 to 5 times faster convergence than existing models.

- **Limitations**
  - Only for sparse interactions (independent agents)
  - Similarity calculation makes the additional burden in the training steps.