1704.03732 - Deep Q-learning from Demonstrations

- **Yunqiu Xu**
- DeepMind AAAI 2018的论文, 和之前在arxiv上看得有一些修改, 现在对imitation learning有一定了解后再看下
- Here are some implementations which I can leverage:
  - https://github.com/go2sea/DQfD
  - https://github.com/nabergh/doom_dqfd
- Further reading:
  - 1709.10089 - Overcoming Exploration in Reinforcement Learning with Demonstrations

---

1. Introduction

- Challenges:
  - DRL: requires large data to train
  - Learn from simulation (sim-to-real):
    - Hard to cast into real world
    - Hard to find accurate simulator
- Our work: DQfD
  - Leverage demonstrations(previous control) to pretrain the agent
  - Then the agent can perform well from the start of learning, and continue to improve by using its self-generated data
  - **Combine TD loss and classification loss**:
    - Classification loss: imitate the desmonstrator
    - TD loss: learn a self-consistent value function
  - **Prioritized replay**: balance the ratio of demonstration and self-generated data

2. Background and Related Work

- Background: MDP, different DQNs
- Related work: DAGGER:
    - Requires expert to be available (on-policy) during training
    - Does not combine imitation with RL
- Related work: Deeply AggreVaTeD:
    - Same requirement as DAGGER
    - The expert must provide a value function as well as actions
    - Only does imitation, can not learn to improve
- Similar work: RL with Expert Demonstraitons:
    - Similar to our work: combine TD and classification loss
    - Our difference:
        - Our agent is pre-trained on the demonstration data, then use self-generated data
        - The use of prioritized replay mechanism
- Similar work: Schaal 1996, Learning from Demonstration
    - Use demonstrations to pretrain agent
    - But do not use supervised loss
- Related work: One-Shot Imitation Learning
    - Input the entire demonstration in addition to current state
    - A distribution of demonstrations with different initial and goal states
    - Can not learn to improve from demonstrations
- Similar work: Accelerated DQN with Expert Trajectories (ADET)
    - Combine TD and classification loss
    - 使用已训练好的一个agent (expert) 来生成demonstration
    - Difference:
        - 使用交叉熵loss (我们用large margin loss)
        - 并未预训练agent使其在与环境的第一次互动时就表现良好

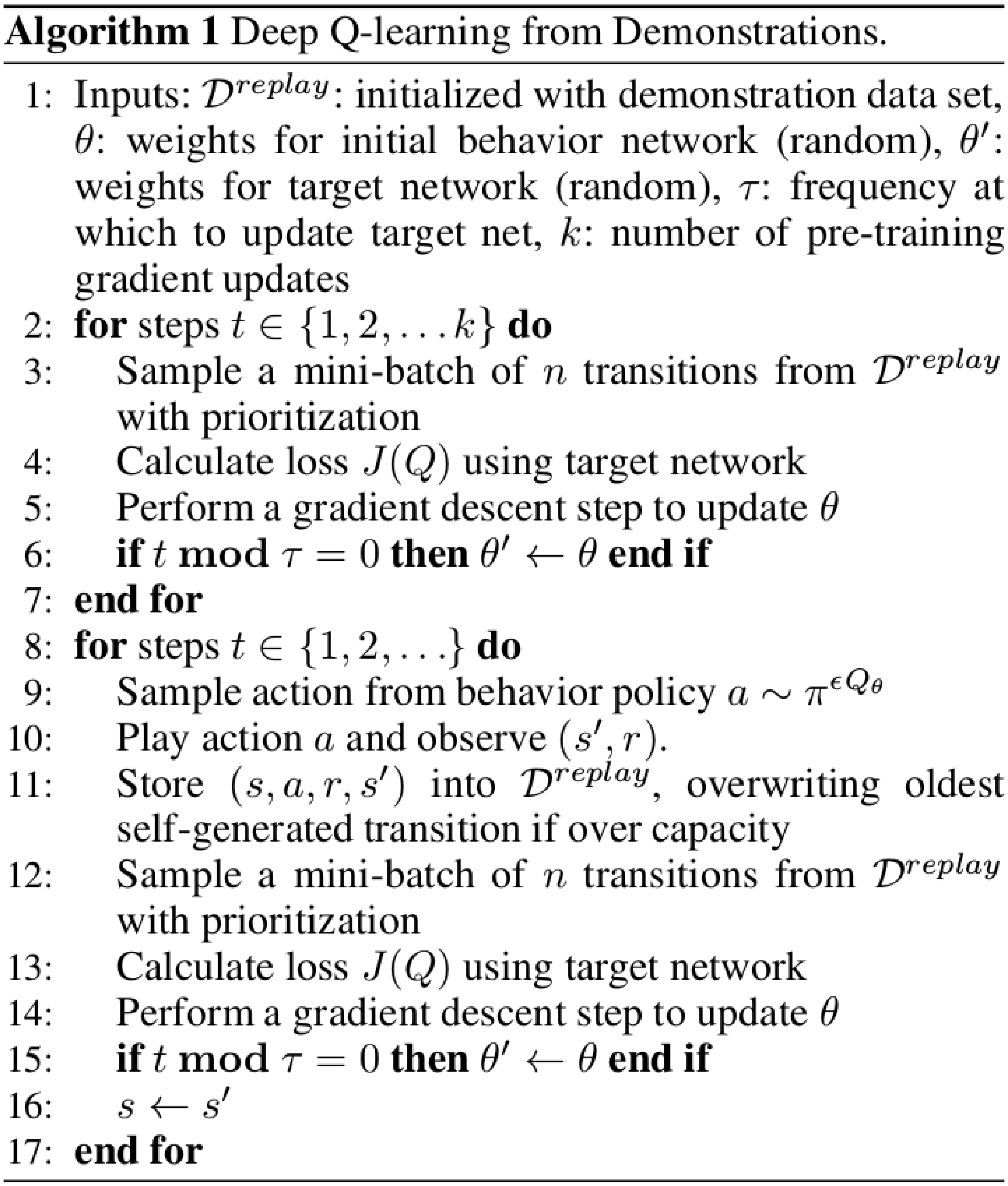
3. DQfD

- Pre-training phase:
    - Performed before real system, uses demonstrations only
    - Goal: learn to imitate the demonstrator with a value function
    - Losses: combine Q-loss, classification loss and L2 loss
- Combined loss:
    - Supervised loss: large margin

      $$J_E(Q) = max_{a\in A}[Q(s,a) + l(a_E, a)] - Q(s, a_E)$$

        - 为什么用这个不用交叉熵(前人): 对获得的value做下限制
        - $l(a_E, a)$: a_E is for expert, if $a_E = a$, the value is 0, otherwise the value is positive
        - 使用这个loss可以保证其他动作的value总是比专家动作的value低一点点
        - **During training (self-generated data), supervised loss will not be used** $\rightarrow \lambda_2 = 0$
    - Q loss:
        - one-step or n-step
        - 仅仅使用分类loss也不行, 需要使用Q-network进行提升
    - L2 loss: for regularization

      $$J(Q) = J_{DQ}(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q) + \lambda_3 J_{L2}(Q)$$
- Replay buffer D^{replay}:
    - Used to store demonstraton
    - Used in pre-training first, store expert's demonstration
    - During training, keep adding self-generated data, if full, **overwrite self-generated data**
    - **Expert's demonstration data will never change once added**
    - Prioritized mechanism:
        - Control the relative sampling of demonstration versus agent data
        - Use different positive constants ϵ_a and ϵ_d to balance the ratio of self-generated data and demonstration
- Pesudocode:



Algorithm 1 Deep Q-learning from Demonstrations.

```
1:  Inputs: $\mathcal{D}^{replay}$: initialized with demonstration data set, $\theta$: weights for initial behavior network (random), $\theta'$: weights for target network (random), $\tau$: frequency at which to update target net, $k$: number of pre-training gradient updates
2:  for steps $t \in \{1, 2, \dots k\}$ do
3:      Sample a mini-batch of $n$ transitions from $\mathcal{D}^{replay}$ with prioritization
4:      Calculate loss $J(Q)$ using target network
5:      Perform a gradient descent step to update $\theta$
6:      if $t$ mod $\tau = 0$ then $\theta' \leftarrow \theta$ end if
7:  end for
8:  for steps $t \in \{1, 2, \dots\}$ do
9:      Sample action from behavior policy $a \sim \pi^{\epsilon Q_\theta}$
10:     Play action $a$ and observe $(s', r)$.
11:     Store $(s, a, r, s')$ into $\mathcal{D}^{replay}$, overwriting oldest self-generated transition if over capacity
12:     Sample a mini-batch of $n$ transitions from $\mathcal{D}^{replay}$ with prioritization
13:     Calculate loss $J(Q)$ using target network
14:     Perform a gradient descent step to update $\theta$
15:     if $t$ mod $\tau = 0$ then $\theta' \leftarrow \theta$ end if
16:     $s \leftarrow s'$
17: end for
```

  - 初始化时注意 τ 的定义: 用于判定何时更新target network
  - 2-7行: pretraining
  - 8-17行: training
    - 和前面的区别就在于多了自己生成的数据
    - 注意11行, 只会复写self-generated data, 之前加进去的demonstration不变
    - 注意13行不需要计算分类loss了
- 和PDDDQN (综合之前的版本, 可参考Rainbow)的对比
  - D^{replay} 中永久保持demonstration, 仅仅重写self-generated data

    - 预训练时仅仅用demonstration, 此时不与环境进行互动
    - Combined losses
    - Demonstration priority

4. Experiment

- Environment: ALE
- Evaluation:
    - Full DQfD with human demonstrations
    - PDD DQN
        - No demonstration
        - No pretraining
        - No supervised losses and regularization
    - Supervised imitation (BC), no environment interaction
        - Cross entropy loss
        - No TD loss → only learns from pre-training
- Human demonstration:
    - Human playing : 3-12 times per game
    - During playing : log agent's state, actions, rewards and terminations
    - 5574-75472 transitions per game (very small demonstration dataset compared with AlphaGo and DQN)

5. Result and Discussion

- On-line scores:

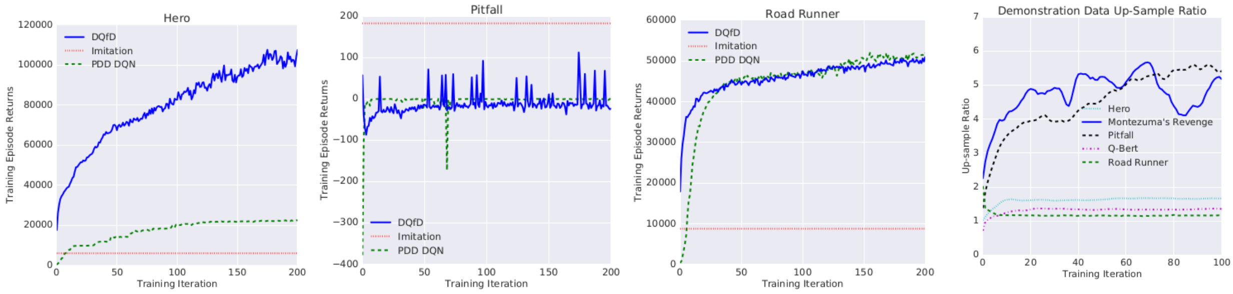
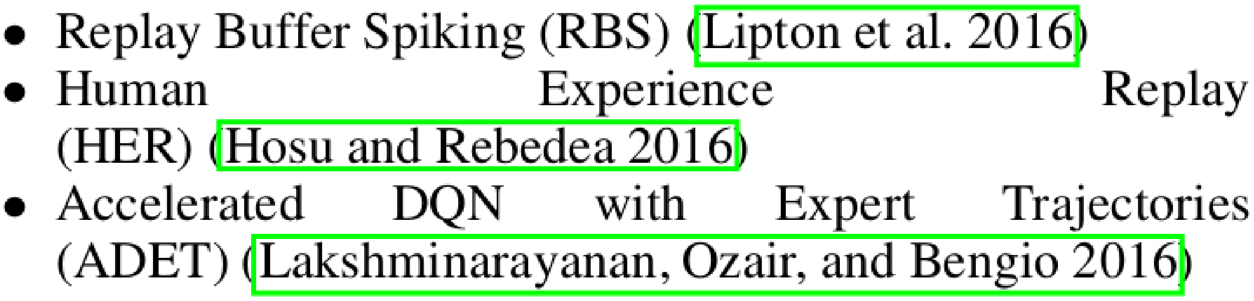


Figure 1: On-line scores of the algorithms on the games of Hero, Pitfall, and Road Runner. On Hero and Pitfall, DQfD leverages the human demonstrations to achieve a higher score than any previously published result. The last plot shows how much more frequently the demonstration data was sampled than if data were sampled uniformly, for five different games.

    - Pretraining: DQfD is much better than PDD DQN at the beginning
        - **However, only naively adding (e.g. only pretraining or filling replay buffer) can not achieve similar performance**
    - Pritorized mechanism:
        - Even after pre-training, the agent still needs expert demonstration
        - The need of expert data grows when the game becomes more difficult, e.g. reaching new screens
- Loss ablation analysis



- Replay Buffer Spiking (RBS) (Lipton et al. 2016)
- Human Experience Replay (HER) (Hosu and Rebedea 2016)
- Accelerated DQN with Expert Trajectories (ADET) (Lakshminarayanan, Ozair, and Bengio 2016)

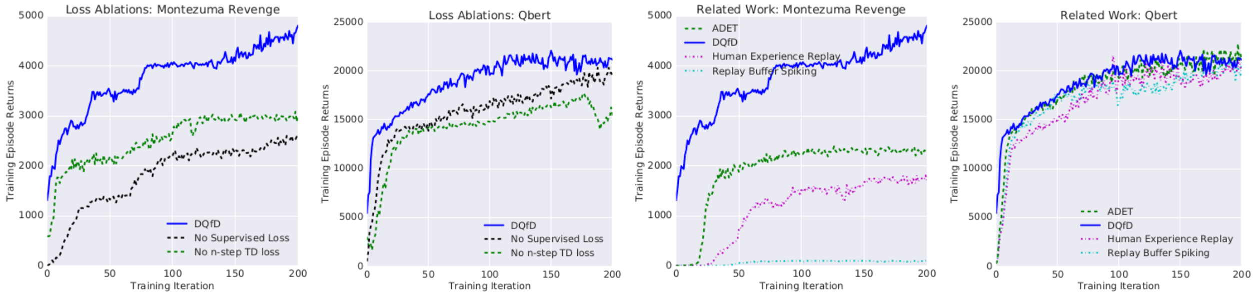
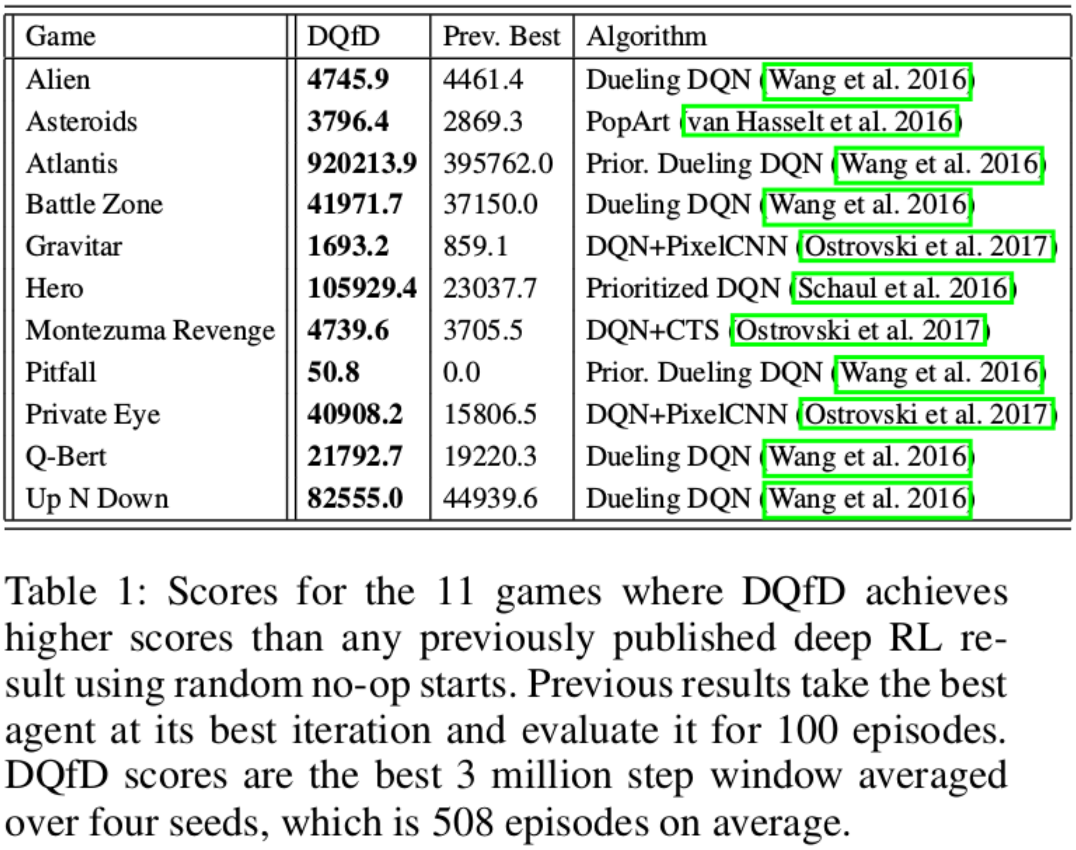


Figure 2: The left plots show on-line rewards of DQfD with some losses removed on the games of Montezuma's Revenge and Q-Bert. Removing either loss degrades the performance of the algorithm. The right plots compare DQfD with three algorithms from the related work section. The other approaches do not perform as well as DQfD, particularly on Montezuma's Revenge.

- Score for 11 games



Game	DQfD	Prev. Best	Algorithm
Alien	**4745.9**	4461.4	Dueling DQN (Wang et al. 2016)
Asteroids	**3796.4**	2869.3	PopArt (van Hasselt et al. 2016)
Atlantis	**920213.9**	395762.0	Prior. Dueling DQN (Wang et al. 2016)
Battle Zone	**41971.7**	37150.0	Dueling DQN (Wang et al. 2016)
Gravitar	**1693.2**	859.1	DQN+PixelCNN (Ostrovski et al. 2017)
Hero	**105929.4**	23037.7	Prioritized DQN (Schaul et al. 2016)
Montezuma Revenge	**4739.6**	3705.5	DQN+CTS (Ostrovski et al. 2017)
Pitfall	**50.8**	0.0	Prior. Dueling DQN (Wang et al. 2016)
Private Eye	**40908.2**	15806.5	DQN+PixelCNN (Ostrovski et al. 2017)
Q-Bert	**21792.7**	19220.3	Dueling DQN (Wang et al. 2016)
Up N Down	**82555.0**	44939.6	Dueling DQN (Wang et al. 2016)

Table 1: Scores for the 11 games where DQfD achieves higher scores than any previously published deep RL result using random no-op starts. Previous results take the best agent at its best iteration and evaluate it for 100 episodes. DQfD scores are the best 3 million step window averaged over four seeds, which is 508 episodes on average.

6. Summary and Future Work

- DQfD:
  - Pretraining using demonstration, then train with both demonstration and self-generated data
  - Combined losses
  - Pritorized mechanism
  - Small demonstration set with good performance
- Future work:
  - Derive more value from demonstrations:
    - Human learn from demonstration in a different way
    - Some information may be inaccessible for current system
  - Apply DQfD in continuous environment (classification loss → regression loss)