Imitation Learning on Atari using Non-Expert Human Annotations

Ameya Panse, Tushar Madheshia, Anand Sriraman, Shirish Karande
TCS Research - TRDDC
54-B, Hadapsar Industrial Estate
Pune - 411040, Maharashtra, India
{ameya.panse, tushar.madheshia, anand.sriraman, shirish.karande}@tcs.com

Abstract

In this paper, we explore the problem of learning a policy from non-expert human demonstrators. We use a consensus algorithm to estimate consensus actions and learn worker skill levels. We iteratively update the skill levels while training an RL agent using learned weights for demonstrations over the entire training period. We perform our experiments in the Atari Learning Environment (ALE) available on OpenAI Gym and show initial results.

Introduction

Deep reinforcement learning has been shown to be very successful in solving problems such as playing Atari games (Mnih et al. 2013) and Go (Silver et al. 2016). However, initial learning via reinforcement learning can be extremely slow and requires a large amount of interactions with the environment to achieve substantial performance. Incorporating human knowledge can help accelerate the training for reinforcement learning agents. Imitation learning allows an agent to learn from human demonstrations by mimicking their behaviour on a task.

Many algorithms have been proposed for Imitation Learning. However, most of the prior work, e.g. DAgger (Ross, Gordon, and Bagnell 2011) and it's extension AggreVaTe (Ross and Bagnell 2014), use expert demonstrations to teach an agent.In this paper, we look at the problem of learning from non-expert human demonstrators. We model the humans' skill levels and learn the consensus actions at the various states. By using learned weighting of various demonstrations, we can perform better than by treating all demonstrations equally. We also use demonstration data throughout the training of the agent, rather than just a bootstrapping method to improve initial performance. We base our work upon demonstrations performed for Atari games by non-expert volunteers. In crowdsourcing literature, several algorithms have been proposed to obtain consensus labels from a set of worker labels. EM-based approaches such as (Welinder and Perona 2010) have been quite popular to model both worker skill levels as well as to obtain the consensus label. Deep neural networks have also been used to obtain crowd consensus (Albarqouni et al. 2016). We use an approach similar to Welinder and Perona, but modify the algorithm to obtain workers' action probability distributions at each state.

(Gao et al. 2018) comes the closest to our approach where they learn from imperfect demonstrations throughout their training. We differ from their approach, as we use an iterative algorithm to learn the consensus policy across demonstrations and use weighted demonstrations by modeling the worker's skill level. Our loss function and regularization methods are also different.

Preliminaries

Reinforcement Learning

The Reinforcement Learning problem that we consider is defined by a Markov Decision Process (MDP). A MDP is characterized by a tuple $< S, A, R, T, \gamma >$, where S is the set of states, A is the set of actions, $R(s, a)$ is the reward function, $T(s, a, s') = P(s'|s, a)$ is the transition probability, and γ is the discount factor. An agent in a particular state, interacts with the environment by taking an action, and receives the reward while transitioning to the next state.

The goal of the agent is to learn a policy π such that the agent maximizes the future discounted reward:

$$\pi = \underset{\pi}{\operatorname{argmax}} \sum_t \gamma^t \mathbb{E}_{s_t,a_t\ \pi}[R_t]$$

Proximal Policy Optimization

In Policy Gradient Methods, the policy gradient is estimated and is used in an stochastic gradient ascent algorithm. A variant of the Policy Gradient Methods, Proximal Policy Optimization (Schulman et al. 2017), where the policy updates are constrained by size while maximizing the clipped objective.

$$L_{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

$$\underset{\theta}{\operatorname{argmax}}\ L_{CLIP}(\theta)$$

subject to

$$L_{KL} = KL(\pi_{\theta_{old}}(\cdot\,|s_t), \pi_\theta(\cdot\,|s_t))] \le \delta$$

where, KL is the Kullback Leibler Divergence. The constraint is applied by using a penalty as follows :

$$\underset{\theta}{\operatorname{argmax}}\ L_{PPO} = L_{CLIP} - \beta\, L_{KL} \tag{1}$$

Method

Let $S = \{s_i\}$ be the set of states observed. At each state s_{ij} a worker that sees the state, takes an action z_{ij}. The workers are asked to complete an episode, and every state-action pair of the worker is recorded. Let π_j be the policy of the worker. If we have multiple annotations for each state, then it is easy to setup a standard consensus algorithm and to estimate the consensus policy π_{cns} to be use for guided exploration. However, in most practical cases, it is infeasible to assume that every state has even a single annotation, let alone multiple. Hence we need to extrapolate π_j to the states not seen by the worker to arrive at a consensus. We make use of Deep Neural Networks for this generalization of the policy to unseen states. We used a convolutional neural network with three convolutional layers, similar to the Deep Q-Network in (Mnih et al. 2015).

Parameterized Policy and Distillation

We consider the parameterized policy of our agent π_θ, where θ are the parameters, such as the weights and biases of our network. We want to make use of the confidence values of each action, produced by the network for better estimates of the skill and difficulty parameters. Hence, the policy is learnt in conjunction with the other parameters.

(Hinton, Vinyals, and Dean 2015) introduced Knowledge Distillation, wherein a small student network accurately learns from a large teacher network by matching soft labels. Inspired by this, our primary contribution in Eq. (2) make use of the consensus policy to guide the exploration of the parameterized policy by adding a regularization loss to match the soft actions of p_θ and π_{cns}.

$$L_D(\theta) = \hat{\mathbb{E}}_s[(\pi_\theta(\cdot|s) - \pi_{cns}(\cdot|s))^2] \qquad (2)$$

We scale the distillation loss by α. We reduce α over time, since the optimal policy need not match the crowd policy. We estimate the distillation loss by a number of random samples from the observed states.

Worker Skill and Difficulty

Let w_j be the parameters encoding the skill level of the worker j. The skill level should represent the confidence of the workers actions. For example, an expert should have a high skill level near compared to a non-expert. For each worker, we estimate their skill level. The action probabilities of a state are weighted according to the skill levels of the workers annotating the state and the inherent difficulty of the state i, encoded by d_i.

We model the worker skill as $0 \leq w_j \leq 1$, where a highly skilled worker has w_j near to 1. We assume a prior of mixed Beta Distributions to model different types of workers (high skill, low skill, spammers).We model $0 \leq d_i \leq 1$ denoting the difficulty level of the state i. Further parameterization of the worker and difficulty based on the time elapsed, so as to take into account the improvement of the worker is being considered as a part of future work.

Parameter Estimation

Let $A = \{a_k\}$ be that set of all possible actions. We define the joint probability distribution over the observed states $S = \{s_i\}$, worker annotations $Z = \{z_{ij}\}$, policy parameters θ, worker parameters $W = \{w_j\}$ and difficulty parameters, $D = \{d_i\}$ as

$$p(Z, W, D, \theta|S) = p(\theta) \prod_i (p(d_i) \prod_k p(a_k|s_i, \theta)) \prod_j p(w_j) \prod_{i,j} p(z_{ij}|s_i, d_i, w_j) \qquad (3)$$

We now estimate the parameters by alternating maximization algorithms (Branson, Van Horn, and Perona 2017) :

$$\hat{\pi}_{cns}(a_k|s_i) = p(a_k|s_i, \hat{\theta}) \prod_j p(z_{ij}|a_k, \hat{d}_i, \hat{w_j}) \qquad (4)$$

$$\hat{a_i} = \operatorname*{argmax}_{a_k} \hat{\pi}_{cns}(a_k|s_i) \qquad (5)$$

$$\hat{d}_i = \operatorname*{argmax}_{d_i} p(d_i) \prod_j p(z_{ij}|\hat{a_i}, d_i, \hat{w_j}) \qquad (6)$$

$$\hat{w_j} = \operatorname*{argmax}_{w_j} p(w_j) \prod_i p(z_{ij}|\hat{a_i}, \hat{d}_i, w_j) \qquad (7)$$

$$\hat{\theta} = \operatorname*{argmax}_{\theta} (L_{PPO}(\theta) - \alpha L_D(\theta)) \qquad (8)$$

where $p(a_k|s_i, \hat{\theta})$ is the confidence output from the agent, and $p(d_i), p(w_j)$ are priors, $p(z_{ij}|a, \hat{d}_i, \hat{w_j})$ is probability of worker j taking action z_{ij} given that a is the optimal action, (8) is solved by stochastic gradient ascent.

$$p(z_{ij}|a, \hat{d}_i, \hat{w_j}) = \begin{cases} \text{if } z_{ij} = a: & w_j(1 - d_i) \\ \text{else}: & \dfrac{1 - w_j(1 - d_j)}{|A| - 1} \end{cases}$$

Experiment Setup and Results

For our preliminary experiment, we wanted to choose three types of Atari games: one where humans were better than RL agents, one where the agent was significantly better, and one where both were performing similarly. We obtained the scores from (McKenzie et al. 2017) for human performance and from (Salimans et al. 2017) for the agent performance . Based on the ratio of human to agent score, we chose Bowling(ratio=5.35) Seaquest(ratio= 11.44), Bankheist(ratio= 1.01) and Breakout(ratio= 0.08) which were available on OpenAI Gym (Brockman et al. 2016). Bowling and Seaquest have a high ratio, indicating that humans can perform better than machines on this game. Hence, there is room to imitate humans to better train our agent. Bankheist having a ratio close to one, we do not expect much difference between pure reinforcement learning and our algorithm. For Breakout, our algorithm does worse due the fact that human performance is below the RL performance.

We invited 21 participants to volunteer and stored their gameplay data, including actions performed, states and rewards generated by the environment. The volunteers consisted of our colleagues, family and friends. We only explained the controls of the game to the players and did not

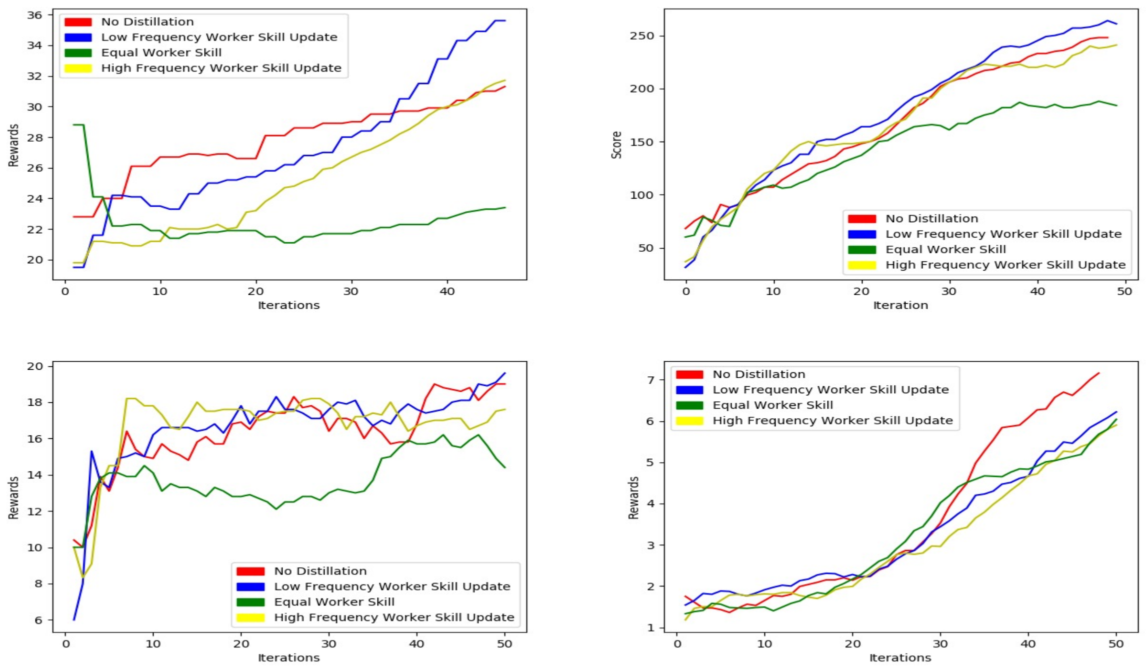

Figure 1: Scores of various training configurations after 50 iterations (200 epochs) on (in clockwise direction) Bowling, Seaquest, Bankheist and Breakout

elaborate on the specific game mechanics. This was done to ensure that they explored the game's reward mechanisms and would improve during the course of their episode.

After collecting the demonstration data, we trained the agent in four configurations. In the first configuration, we didn't incorporate the distillation loss during training to simulate the vanilla RL training without demonstration data. Next, we set equal skill levels to all the workers, and didn't update them during training. This was a baseline to understand how the algorithm performed if the worker skill level wasn't modeled and all demonstrations were treated as oracle demonstrations. Finally we ran two configurations where we updated the worker skill levels at a low frequency (an update every 10 iterations) and high frequency (an update every iteration). We train all configurations for 50 iterations, each iteration being 4 epochs.

In situations where, human knowledge is useful for imitation, as seen in Figure 1, updating the worker parameters with a low frequency gave us the fastest training improvement and highest average score.In the case where human and RL performance was similar (Bankheist), we do not see a significant difference between our algorithms and pure Reinforcement learning. Whereas, in situations where human performance is significantly worse that RL performance (like Breakout), our algorithm takes a hit since the incorporated human skills worsens the performance.

However, treating all workers as experts (equal high skill) lead to the worst performance in all cases, thus proving that worker modeling is necessary for high performance levels.

Discussion and Limitations

The number of observed states were very high in number. Hence, while updating the worker and difficulty parameters, it was unfeasible to run over all observed states due to memory constraints. Instead, we sampled all states where we had multiple crowd inputs and matched those with an equal number of randomly sampled states which totaled to 600 states.

The frequency of the parameter updates have an impact on the learning time and also the performance of the agent, and this relationship is not monotonic. Too low frequencies (Equal Skill Worker) and too high frequencies (High Frequency Skill Update), both do not produce the best results. An adaptive method of updates might boost the performance significantly.

In this paper, we have introduced a novel formulation for continuous use of non-expert demonstration data for RL. We have shown that modeling the worker skill levels, and using weighted demonstrations during training helps speed up the training significantly. In the future, we plan to scale up our experiments, by optimizing our web system and getting more demonstrations from a public crowd. We also plan on exploring more ways of modeling worker behaviour, e.g. learning in-game, shared worker parameters across games and modeling the interference created by the game delivery system like lag, jitter, etc.

References

Albarqouni, S.; Baur, C.; Achilles, F.; Belagiannis, V.; Demirci, S.; and Navab, N. 2016. Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images. *IEEE transactions on medical imaging* 35(5):1313–1321.

Branson, S.; Van Horn, G.; and Perona, P. 2017. Lean crowdsourcing: Combining humans and machines in an online system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7474–7483.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Gao, Y.; Lin, J.; Yu, F.; Levine, S.; Darrell, T.; et al. 2018. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

McKenzie, M.; Loxley, P.; Billingsley, W.; and Wong, S. 2017. Competitive reinforcement learning in atari games. In *Australasian Joint Conference on Artificial Intelligence*, 14–26. Springer.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Ross, S., and Bagnell, J. A. 2014. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*.

Ross, S.; Gordon, G.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635.

Salimans, T.; Ho, J.; Chen, X.; and Sutskever, I. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature* 529(7587):484–489.

Welinder, P., and Perona, P. 2010. Online crowdsourcing: rating annotators and obtaining cost-effective labels.