ICML - Author Response

1. $SQLoss$: Theory

Equation 6 (section 2.3) describes the gradient for standard policy gradient. It has two terms. The $log\pi^1(u_t^1|s_t)$ term maximises the likelihood of reproducing the training trajectories $[(s_{t-1}, u_{t-1}, r_{t-1}), (s_t, u_t, r_t), (s_{t+1}, u_{t+1}, r_{t+1}), \dots]$. The return term pulls down trajectories that have poor return. The overall effect is to reproduce trajectories that have high returns. We refer to this standard loss as $Loss$ for the following discussion.

Lemma 1. For agents trained with random exploration in the IPD, $Q_\pi(D|s_t) > Q_\pi(C|s_t)$ for all s_t.

Let $Q_\pi(a_t|s_t)$ denote the expected return of taking a_t in s_t. Let $V_\pi(s_t)$ denote the expected return from state s_t.

$$\begin{aligned}
Q_\pi(C|CC) = 0.5[(-1) + V_\pi(CC)] + 0.5[(-3) + V_\pi(CD)]\\
Q_\pi(C|CC) = -2 + 0.5[V_\pi(CC) + V_\pi(CD)]\\
Q_\pi(D|CC) = -1 + 0.5[V_\pi(DC) + V_\pi(DD)]\\
Q_\pi(C|CD) = -2 + 0.5[V_\pi(CC) + V_\pi(CD)]\\
Q_\pi(D|CD) = -1 + 0.5[V_\pi(DC) + V_\pi(DD)]\\
Q_\pi(C|DC) = -2 + 0.5[V_\pi(CC) + V_\pi(CD)]\\
Q_\pi(D|DC) = -1 + 0.5[V_\pi(DC) + V_\pi(DD)]\\
Q_\pi(C|DD) = -2 + 0.5[V_\pi(CC) + V_\pi(CD)]\\
Q_\pi(D|DD) = -1 + 0.5[V_\pi(DC) + V_\pi(DD)]
\end{aligned} \tag{9}$$

Since $V_\pi(CC) = V_\pi(CD) = V_\pi(DC) = V_\pi(DD)$ for randomly playing agents, $Q_\pi(D|s_t) > Q_\pi(C|s_t)$ for all s_t.

Lemma 2. Agents trained to only maximise the expected reward in IPD will converge to mutual defection.

This lemma follows from Lemma 1. Agents initially collect trajectories from random exploration. They use these trajectories to learn a policy that optimises for long-term return. These learned policies always play D as described in Lemma 1.

Equation 7 describes the gradient for $SQLoss$. The $log\pi^1(u_{t-1}^1|s_t)$ term maximises the likelihood of taking u_{t-1} in s_t. The imagined episode return term pulls down trajectories that have poor imagined return.

Lemma 3. Agents trained on random trajectories using only $SQLoss$ oscillate between CC and DD.

For IPD, $s_t = (u_{t-1}^1, u_{t-1}^2)$. The $SQLoss$ maximises the likelihood of taking u_{t-1} in s_t when the return of the imagined trajectory $\hat{R}_t(\hat{\tau}_1)$ is high.

Consider state CC, with $u_{t-1}^1 = C$. $\pi^1(D|CC)$ is randomly initialised. The $SQLoss$ term reduces the likelihood of $\pi^1(C|CC)$ because $\hat{R}_t(\hat{\tau}_1) < 0$. Therefore, $\pi^1(D|CC) > \pi^1(C|CC)$.

Similarly, for CD, the $SQLoss$ term reduces the likelihood of $\pi^1(C|CD)$. Therefore, $\pi^1(D|CD) > \pi^1(C|CD)$. For DC, $\hat{R}_t(\hat{\tau}_1) = 0$, therefore $\pi^1(D|DC) > \pi^1(C|DC)$. Interestingly, for DD, the $SQLoss$ term reduces the likelihood of $\pi^1(D|DD)$ and therefore $\pi^1(C|DD) > \pi^1(D|DD)$.

Now, if s_t is CC or DD, then s_{t+1} is DD or CC and these states oscillate. If s_t is CD or DC, then s_{t+1} is DD, s_{t+2} is CC and again CC and DD oscillate. This oscillation is key to the emergence of cooperation as explained in section 2.3.1.

Lemma 4. For agents trained using both standard loss and $SQLoss$, $\pi(C|CC) > \pi^1(D|CC)$.

For CD, DC, both the standard loss and $SQLoss$ push the policy towards D. For DD, with sufficiently high κ, the $SQLoss$ term overcomes the standard loss and pushes the agent towards C. For CC, initially both the standard loss and $SQLoss$ push the policy towards D. However, as training progresses, the incidence of CD and DC diminish because of $SQLoss$ as described in Lemma 3. Therefore, $V_\pi(CD) \approx V_\pi(DC)$ since agents immediately move from both states to DD. Intuitively, agents lose the opportunity to exploit the other agent. In equation 9, with $V_\pi(CD) \approx V_\pi(DC)$, $Q_\pi(C|CC) > Q_\pi(D|CC)$ and the standard loss pushes the policy so that $\pi(C|CC) > \pi(D|CC)$. This depends on the value of κ. For very low values, the standard loss overcomes $SQLoss$ and agents defect. For very high values, $SQLoss$ overcomes standard loss and agents oscillate between cooperation and defection. For moderate values of κ (as shown in our experiments), the two loss terms work together so that $\pi(C|CC) > \pi(D|CC)$.

G. Reward modifications needed to make this work with Q-learning

1. In state CC, for example, if you take D, you get the normal next state reward, however, if you take C, SQLoss kicks in to give the extra SQLoss reward.

2. For CD, SQLoss kicks in for C

3. For DC, SQLoss kicks in for D

4. For DD, SQLoss kicks in for D So, interestingly, every time you stick to the status quo, you get this extra reward term, otherwise you do not

H. extra stuff

Therefore, $\pi^1(C|CC) < \pi^1(D|CC)$ post the update.

Note that $\pi(u_t|s_t)$ For $s_t = CC$, $u_{t-1}^1 = C$ and $\hat{R}_t^1(\hat{\tau}_1) << 0$.

Initially, $\pi^1(u_t^1|s_t)$ is random. Consider a gradient update to π^1 using only the $SQLoss$ term from Equation 7.

Therefore, $\pi^1(C|CC) < \pi^1(D|CC)$ post the update. Similarly, $\pi^1(D|DD) < \pi^1(C|DD)$. For $s_t = CD, u_{t-1}^1 = C$ and $\hat{R}_t^1(\hat{\tau}_1) << 0$. Therefore, $\pi^1(C|CD) < \pi^1(D|CD)$. Note that, for $s_t = DC, u_{t-1}^1 = D$ and $\hat{R}_t^1(\hat{\tau}_1) = 0$. Therefore, $\pi^1(D|DC) > \pi^1(C|DC)$.

If we use **only** $SQLoss$ gradient updates, and $s_0 = CC$, then both agents play D (since $\pi^1(C|CC) < \pi^1(D|CC)$) to reach $s_1 = DD$. In $s_1 = DD$ both agents play C to reach $s_2 = CC$ and this $[CC, DD]$ sequence oscillates. If $s_0 = DC$, then agent 1 plays D, and agent 2 plays C (for agent 2 the state is CD) to reach $s_1 = DD$ and subsequently a $[DD, CC]$ sequence. The same is true for $s_0 = CD$. Hence, regardless of s_0, using **only** $SQLoss$ leads to a $[CC, DD]$ sequence (intuition is described in section 2.3.1).

In the absence of $SQLoss$, consider policy $\pi^1(u_t^1|s_t)$, which is initially random. After playing a certain number of games against $\pi^2(u_t^2|s_t)$ (also random), let $E^1(u_t|s_t)$ denote the expected reward when playing u_t in s_t. $E^1(C|CC) = 0.5(-1) + V(CC) + 0.5(-3) + V(CD) = -2 + V(CD) + V(CC)$ (If a_1 plays C, then with $p = 0.5$ it stays in CC, with $p = 0.5$ it goes to CD, and r_t denotes the reward of subsequent random play). Similarly, $E^1(D|CC) = 0.5(0) + V(DC) + 0.5(-2) + V(DD) = -1 + V(DC) + V(DD)$. Therefore, $E^1(D|CC) > E^1(C|CC)$. Also, $E^1(C|DC) = 0.5(-1) + V(CC) + 0.5(-3) + V(CD) = -2 + V(CC) + V(CD)$, $E^1(D|DC) = 0.5(0) + V(DC) + 0.5(-2) + V(DD) = -1 + V(DC) + V(DD)$. Therefore, $E^1(D|DC) > E^1(C|DC)$. Similarly, $E^1(D|CD) > E^1(C|CD)$ and $E^1(D|DD) > E^1(C|DD)$. Note that for random agents, $V(CC) = V(DD) = V(DC) = V(CD)$. After the gradient update $\pi^1(D|s_t) > \pi^1(C|s_t)$ and $\pi^2(D|s_t) > \pi^2(C|s_t)$. As agents learn, $V(DD) \to -2$, $V(CC) \to -3$, because cooperative agents are exploited. Therefore, a_1 and a_2 converge to mutual defection.

However, with the $SQLoss$ term, $E^1(C|CC) = -2 + V(CD) + V(CC) + \kappa * (-1)$. Also, $E^1(D|DD) = -1 + V(DD) + V(DC) + \kappa * (-2)$. $E^1(C|CD) = -2 + V(CD) + V(CC) + \kappa * (-3)$. $E^1(D|DC) = -1 + V(DC) + V(DD) + \kappa * (0)$. As before, for random agents, $V(CC) = V(DD) = V(DC) = V(CD)$. The key idea is that with the addition of $SQLoss$, $E^1(C|DD) > E^1(D|DD)$. Now, to achieve cooperation, we need $E^1(C|CC) > E^1(D|CC)$. $E^1(C|CC) = -2 + V(CD) + V(CC) + \kappa * (-1)$. $E^1(D|CC) = -1 + V(DC) + V(DD)$. Since $\pi^1(D|CD) > \pi^1(C|CD)$ and $\pi^1(D|DC) > \pi^1(C|DC)$, over time $V(CD)V(CD)V(DD)$

Therefore, a_1 and a_2 initially move to a $[CC, DD]$ sequence.

pushes agents in the IPD towards either CC or DD. In the absence of $SQLoss$, the IPD reward $R_t^1(\tau_1)$ (Equation 5) pushes agents individually towards DC, and therefore leads to mutual defection. However, with $SQLoss$, DC is diminished, $[DD, CC]$ remain and the IPD reward moves agents towards CC since $R_t(CC) > R_t(CC)$.

However, with the $SQLoss$ term pushes agents in the IPD towards either CC or DD. In the absence of $SQLoss$, the IPD reward $R_t^1(\tau_1)$ (Equation 5) pushes agents individually towards DC, and therefore leads to mutual defection. However, with $SQLoss$, DC is diminished, $[DD, CC]$ remain and the IPD reward moves agents towards CC since $R_t(CC) > R_t(CC)$.

I. GameDistill - Architecture Details

GameDistill consists of two components.

First, the state sequence encoder (Step 2, Section 2.4) that takes as input a sequence of 4 consecutive game states (input size is 4x4x3x3, where 4x3x3 is the dimension of the game state) and outputs a single feature representation. We encode each state in the sequence using a common trunk of 3 Convolution layers with kernel-size 3x3, followed by fully-connected layer with 100 neurons to obtain a feature-representation. This unified feature vector is then given as input to the different prediction branches of the network. The 3 branches, color-of-picked-coin (classification task), self-reward (regression task), and the opponent-reward (regression task) (as shown in Figure 2), independently use this feature vector as input to compute appropriate output. They take as input the feature vector and use a dense hidden layer (with 100 neurons) with linear activation to predict the output. Linear activation allows us to cluster the feature vectors (embeddings) using a linear clustering algorithm, such as Agglomerative Clustering. We use the *Binary-Cross-Entropy (BCE)* loss for classification tasks and *mean-squared error (MSE)* loss for the regression tasks in the prediction branches. We use the *Adam* (Kingma & Ba, 2014) optimizer with learning-rate of $3e-3$.

Second, the oracle network (Step 4, Section 2.4), that predicts an action for an input state. We encode the input state using a sequence of 3 convolution layers with kernel-size of 2x2 and *relu* as the activation function. To predict the action, we use 2 fully-connected layers with Cross Entropy as the loss function and *relu* as the activation. We use *L2* regularization, and *Gradient Descent* optimizer (learning rate $1e-3$) for all the experiments.