FOLLOW THE NEURALLY-PERTURBED LEADER FOR ADVERSARIAL TRAINING

Ari Azarafrooz
ari.azarafrooz@gmail.com

ABSTRACT

Game-theoretic models of learning are a powerful set of models that optimize multi-objective architectures. Among these models are zero-sum architectures that have inspired adversarial learning frameworks. We extend these two-player frameworks by introducing a *mediating neural agent* whose role is to augment the observation of the players to achieve certain maximum entropic objectives.

We show that the new framework can be utilized for 1) efficient online training in multi-modal and adaptive environments and 2) addressing the ergodic convergence and *cyclic* dynamics issues of adversarial training. We also note the proposed training framework resembles the 'follow the *perturbed* leader' learning algorithms where perturbations are the result of actions of the mediating agent.

We validate our theoretical results by applying them to the games with convex and non-convex loss as well as generative adversarial architectures. Moreover, we customize the implementation of this algorithm for adversarial imitation learning applications where we validate our assertions by using a procedurally generated game environment as well as synthetic data.

Pytorch implementation of the algorithms and experiments can be found in https://github.com/azarafrooz/FTNPL.

1 INTRODUCTION

A wide range of recent learning architectures expresses the learning formulation as a multi-objective and game-theoretic problem. They are useful to build log-likelihood free deep generative models (1; 2), learn disentangled representations (3; 4), learn adversarial imitation (5), learn complex behaviors (6), incorporate hierarchical modeling to mitigate the reinforcement (RL) exploration issues (7; 8; 9), formulate curiosity (10; 11; 12) and imagination objectives in RL (13), tighten the lower bound for mutual information estimates (14), compute synthetic gradients (15), etc. However, the behavior of the gradient-based methods of training in these architectures is even more complicated than those of single objective ones. For example, (16; 17) show that gradient-based methods suffer from recurrent dynamics, slow convergence, and the inability to measure convergence in zero-sum type games. The existence of cyclic behavior necessitates a slow learning rate and convergence. (16) proposed a new gradient-based method by utilizing the dynamics of Hamiltonian and Potential games.

The focus of our paper is on online training of *adversarial* architectures using *regret minimization* framework (18; 19; 20). Adversarial training using regret minimization framework also suffers from *cyclic* behaviors (21; 17; 22; 23). However, as we show in this paper, it provides a mathematically elegant framework for designing novel training algorithms that avoid cyclic dynamics. Another difficulty of standard regret minimization methods is that they fail in *non-convex* settings. To address the non-convexity issue, (24) invokes an offline oracle to introduce random noise. We propose a novel algorithm to address both of these issues. We show that a neural network mediator can remove the cyclic behaviors by perturbing the dynamics of the game. Moreover, there is no need for convexity assumption in our approach. The mediator perturbs the dynamic of the game by augmenting the observation of the players with so-called 'correlated' codes. The nature of such codes is related to the notion of correlated equilibrium (25; 26) and similar to disentangled codes in (3; 4). Thanks to these correlated codes, the proposed method is also an efficient approach for learning in multi-modal and adaptive environments.

2 PROBLEM FORMULATION

2.1 GAME THEORETICAL PRELIMINARIES

Figure 1: Generalizations of pure Nash equilibria. 'PNE' stands for pure Nash equilibria; 'MNE' for mixed Nash equilibria; 'CorEq' for correlated equilibria; and 'No Regret (CCE)' for coarse correlated equilibria.

Consider a zero-sum game between two players, agent π and discriminator D with strategies $\phi \in \Phi, \omega \in \Omega$ respectively. In a deep learning game, each player is a neural network with its parameters as strategies. Let $L(\phi, \omega)$ denote the loss of the game and $\mu \in \triangle\ \Phi \times \Omega$ be *joint* mixed strategy, where $\triangle$ and $\times$ denote the probability simplex and cartesian product respectively. Let us also define the *marginal* mixed strategies μ_π, μ_D. For example, $\mu_\pi(\phi)$ is the probability that an agent plays strategy ϕ. A Nash equilibrium gets achieved when no player has an incentive to deviate unilaterally. If a Nash equilibrium gets achieved over the probability distributions of strategies, it is a mixed Nash equilibrium (MNE). If one relaxes the assumptions on the equilibrium, other game-theoretic concepts can be derived, as shown in Fig. 1. The least restrictive of these is known as *coarse* correlated equilibria (CCE). It corresponds to the *empirical* distributions that arise from the repeated joint-play by no-regret learners. An essential concept required for the development of our algorithm is correlated equilibrium (CorEq) (25). Computing CorEq amounts to solving a linear program. As a result, it is computationally less expensive than computing NE, which amounts to computing a fixed point.

2.2 NO-REGRET LEARNING

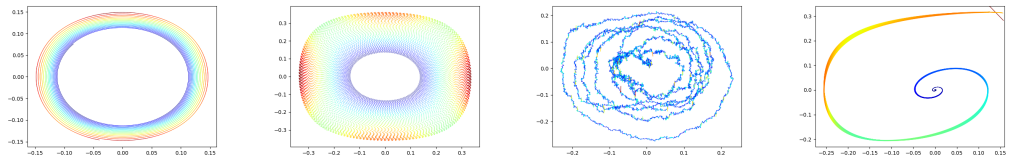
One gradient-based method of learning in games is no-external-regret learning. External regret is the difference between the actual loss achieved and the smallest possible loss that could have been achieved on the sequence of plays by playing a *fixed* action. For example in the context of the aforementioned zero-sum game, the regret for π and D is $\max_{\phi \in \Phi} \sum_{t=0}^{T-1} L(\phi_t, \omega_t) - L(\phi, \omega_t)$ and $\min_{\omega \in \Omega} \sum_{t=0}^{T-1} L(\phi_t, \omega_t) - L(\phi_t, \omega)$ respectively. Regret minimization algorithms ensure that long term regret is *sublinear* in the number of time steps. It is known that the optimal minimax regret of zeros-sum games is $\mathcal{O}(\log(T))$ (28; 29). There are several classes of algorithms that can yield sub-linear regret. One well-known class of no-regret learners is *Follow The Regularized Leader (FTRL)*(30):

$$\phi_t = \arg\min \sum_{i<t} L(\phi_i, \omega_i) + h(\phi_i)\ \ ;\ \ \omega_t = \arg\max \sum_{i<t} L(\phi_i, \omega_i) - h(\omega_i) \tag{1}$$

One common choice is ℓ_2 regularization $h(\theta) = \|\theta\|^2$ which we used in the following examples to illustrate the problem visually. FTRL algorithms are not suitable for non-convex losses. To address the non-convex optimization case, (24) proposed a *follow the perturbed leader (FTPL)* by choosing $h(\theta) = \sigma\theta$ where $\sigma \sim (Exp(\zeta))$ is an exponential noise introduced by an oracle.

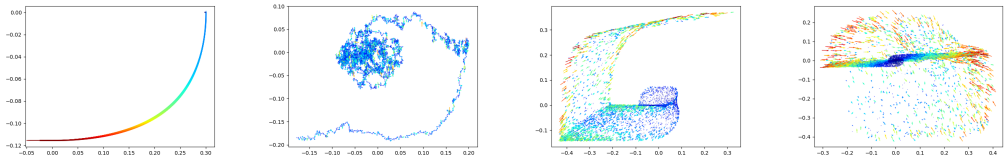
Ergodic convergence and cyclic dynamics Let $h^T = (\mu_0, ..., \mu_T)$ be the history of past strategies when the game is played repeatedly up to time T and $\mu_t = (\phi_t, \omega_t)$. It is known that that no-regret dynamics converges to MNE in zero-sum games (28; 31) in an *ergodic* sense. Ergodic convergence [1] implies that the time average of h^T will converge to the MNE. Previous works relied on this notion of convergence for training adversarial networks (18). However, this notion can be misleading. It is as meaningful as the statement that "moon converges to earth" instead of stating that the moon follows a trajectory that has the earth as its center (32). Ergodic convergence is also related to the dynamics of FTRL in adversarial games which is known to exhibit recurrent dynamics (21). This

[1] It is a.k.a as weak convergence but we avoid this term as much as we can to not get the reader confused with the weak convergence of random variables



(a) FTRL: small leaning rate (b) FTRL: large learning rate (c) FTPL (d) FTNPL

Figure 2: Training dynamic trajectories in example 1. x-axis and y-axis are $\phi[0]$ and $\omega[0]$ respectively. $\|\mu_t - \mu_{t-1}\|^2$ are encoded using colors to track convergence. Small values are blue and largest are red.



(a) FTRL (b) FTPL (c) FTNPL with one code (d) FTNPL with two codes

Figure 3: Training dynamic trajectories in example 2. x-axis and y-axis are $\phi[0]$ and $\omega[0]$ respectively. $\|\mu_t - \mu_{t-1}\|^2$ are encoded using colors to track convergence. Small values are blue and largest are red.

cyclic behavior is common across all choices of regularizers h and learning rates. We also show in the following example that ergodic convergence and cyclic behavior also hold true for the FTPL.

Example 1. *Consider a specific type of zero-sum game known as matching Pennies game with* $L(\phi, \omega) = \phi \boldsymbol{A} \omega^T$ *with* $A = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$. *MNE is* $\mu^* = (0, 0)$ *for this game.*

Trajectories of FTRL training dynamic for this example is visualized in Fig. 2a and Fig. 2b. Note in Fig. 2b, how a large learning rate leads to an even weaker convergence (in the sense defined above). Fig. 2c shows that FTPL also exhibits recurrent dynamics. Aside from the weak convergence, another implication of this cyclic behavior is the slow learning rate, as discussed in (16). This is because gradient-based algorithms do not follow the steepest path toward fixed points due to the 'rotational force'.

Non-convexity FTRL with ℓ_2 regularization does not converge in non-convex situations as shown in the following example.

Example 2. *Assume* π *to have the same loss as in Ex. 1 but let the loss for* D *to be defined ReLU*$(-\phi \boldsymbol{A} \omega^T)$ *where ReLU*$(x) = \max(0, x)$. *MNE stays the same* $\mu^* = (0, 0)$ *but FTRL does not converge to MNE as shown in Fig 3a. Unlike FTRL, FTPL converges to MNE but in an ergodic sense. We visualized the learning trajectories of FTPL in 3b.*

3 Follow the Neurally-Perturbed Leader

Another choice for regularizer $h(\theta)$ in FTRL is the entropy function $H(\theta)$. This choice of regularizer leads to Multiplicative Weights (MW) algorithm (33). However, instead of introducing entropy regularizer $h(.)$ to FTRL, we introduce a mediator agent $\mathcal{M}$ that learns to minimize $D_{KL}(\mu^* || \mu_t)$ (or equivalently to maximize entropy $H(\theta)$). It achieves this by augmenting the observations via generate codes c_t. As a result of the mediator's actions c_t, the loss of the modified game is $L(\phi_t, \omega_t, c_t)$.

The meditator learning objective can be formulated as a reinforcement learning framework with action c_t and the reward term formulated in eq. 5 of the Appendix. The mediator learns to augment players' observation s.t no users can predict the other player's next move, therefore encouraging a maximum entropic behavior in the training dynamics of the game.

We refer to such a no-regret algorithm 'follow the neurally perturbed leader (FTNPL)' since it can be viewed as FTPL with a neural network agent $\mathcal{M}$ as an oracle. The scheme of FTNPL is visualized in

Algorithm 1 Follow the Neurally-Perturbed Leader (FTNPL)

Input:. Code size C, queue size K.
Initialize: Initial parameters $(\phi_0, \omega_0, \psi_0)$ for agent π, discriminator D and mediator $\mathcal{M}$ respectively. Initial observable information in the game $\mathcal{I}_0$, empty queues of size K $h_D = h_\pi = \emptyset$, h_π.insert(ϕ_0), h_D.insert(ω_0), $\mathbf{u}_D = [], \mathbf{u}_\pi = []$
for $i = 0, 1, 2, ...$ **do**
$c_i = \mathcal{M}_{\psi_i}(\mathcal{I}_i)$

for $\phi \in h_\pi$ **do**
$\mathbf{u}_\pi$.append$(L(\phi_i, \omega_i, c_i))$
end for
$\phi_{i+1} \leftarrow \nabla_{\phi_i} \sum \mathbf{u}_\pi$

for $\omega \in h_D$ **do**
$\mathbf{u}_D$.append$(-L(\phi_i, \omega_i, c_i))$
end for
$\omega_{i+1} \leftarrow \nabla_{\omega_i} \sum \mathbf{u}_D$

$\psi_{i+1} \leftarrow -\nabla_{\psi_i} \hat{\mathbb{E}}_{\chi_i} \log \mathcal{M}_{\psi_i}(\mathcal{I}_i) r_m(\mathbf{u}_D, \mathbf{u}_\pi)$ with r_m defined in eq. 5
h_π.insert(ϕ_{i+1}), h_D.insert(ω_{i+1}), $\mathbf{u}_\pi = []$, $\mathbf{u}_D = []$,
end for

Fig. 4 and the algorithm is fully described in 1. In the Appendix section, we show that FTNPL does not require a convexity assumption and that converges to MNE without cyclic behavior (instead of ergodic convergence).

3.1 FTNPL Implementation

The description of the algorithm is given in 1. At every time step, the mediator uses the available information in the game $\mathcal{I}_t$ to generate correlated codes c_t. $\mathcal{I}$ can take the form of pair of latest strategies of the games (ϕ, ω) (in the case of games), pair of observations and actions (s, a) (in the case of imitation learning) or the real data in the form of generative networks. Both players update their parameters using an FTL algorithm. At every step of the game, the mediator updates its parameters according to the reward function in Eq. 5. In practice, we use the second power of g instead of $ReLU$ function. We also parameterize the mediator policy using the reparameterization trick (37). In the convex case (e.g Ex. 1), mediator action is implemented as the mean of the parameterized policy distribution. However, in the non-convex case (e.g Ex. 2), mediator actions have to be random samples from the parameterized policy distribution for the game to converge. For all the experiments except Fig. 2d, we implemented the mediator action as random samples rather than mean.

Since we are dealing with mixed strategies rather than pure strategies, we equip the algorithms with queues of discriminators and agents, in contrast with the classical approach, where there is a single discriminator and single agent. In practice, we keep the queue size K small since the runtime and memory of FTNPL algorithm grow linearly with K. Unlike (18), FTNPL requires no special queuing update. Thanks to the theoretical guarantees of FTNPL, none of the previous GAN training hacks such as choices of entropy regularization, grad penalty, or parameter clipping for the discriminator are required. FTNPL removes the recurrent dynamics as well as difficulties of past training methodologies.

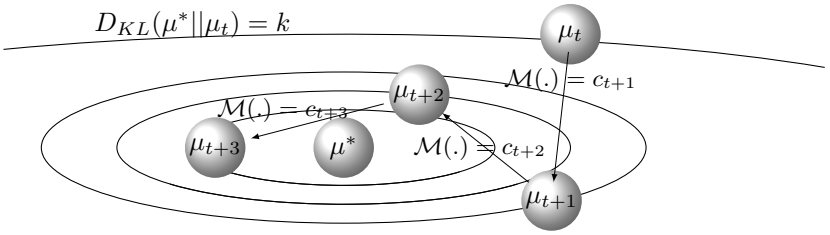
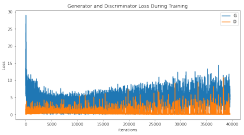
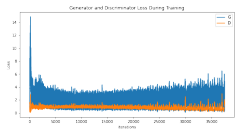


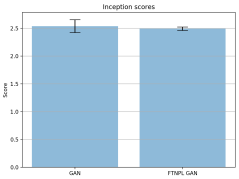
Figure 4: FTNPL scheme: $\forall \mu_t$ on the same orbit, $D_{KL}(\mu^*||\mu_t) = k$. Mediator $\mathcal{M}$ minimizes $D_{KL}(\mu^*||\mu_t)$ by learning from reward term defined in eq. 5.



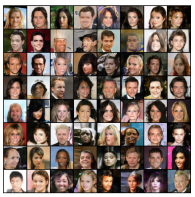
(a) GAN training dynamic applied to celebrity data.



(b) FTNPL-GAN training dynamic applied to celebrity data.



(c) Inception scores after 25 epochs.



(d) Generated images after 25 epochs.

Figure 5: FTNPL applied to GAN.

4 APPLICATIONS

We chose $K = 5$ and a code-size of $C = 2$ for all the experiments.

4.1 MATCHING PENNIES GAME

We applied FTNPL to example 1 and 2 with $\mathcal{I}$ being the pair of latest strategies of the games (ϕ, ω). It converges to MNE μ^* in both cases. The training dynamics do not exhibit recurrent dynamics as shown in Fig. 2d and converges to MNE even under non-convex losses as shown in Fig. 3c and Fig. 3d with code size of $C = 1$ and $C = 2$ respectively.

4.2 GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GAN) (1) is a well-known zero-sum deep learning architecture capable of generating synthetic samples from arbitrary distribution p_{data}. Player π gets random noise z as input and generates synthetic data $\pi_\phi(z)$. Player D then assigns scores $D_\omega(\pi_\phi(z))$. $L(\phi, \omega)$ is the Jenson-Shannon distance between $\pi_\phi(z)$ and p_{data}. We applied FTNPL to DCGAN (38) with $\mathcal{I}$ being the real image data. Our experiment using celeb data (39) shows that training dynamic of FTNPL is smoother compared with that of DCGAN as demonstrated in Fig. 5b and 5a. We stopped training for both models after 25 epochs and computed their inception scores (40). Fig. 5c shows that FTNPL leads to smaller variance for the inception score. Some generated samples are visualized in Fig. 5d.

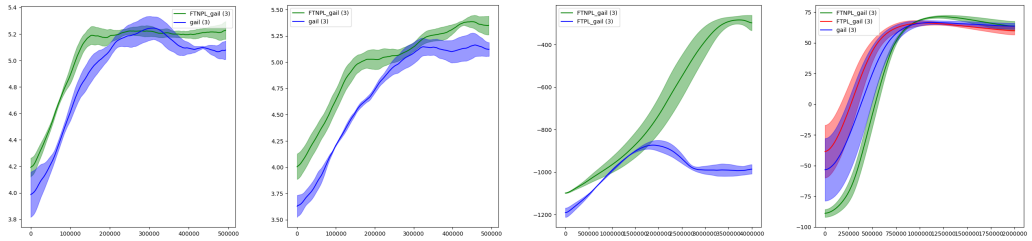
4.3 ADVERSARIAL IMITATION LEARNING

Algorithm 2 Correlated Rollout

Input: policy network π, mediator policy network $\mathcal{M}$, $i = 0$, number of steps n, an initial code $c_0 = 0_C$ with C the size of codes, initial state-action trajectory $\tau = []$, initial code trajectory $\tau_c = []$, environment env, s_0 = env.reset()
Output: state-action trajectory τ , code trajectory τ_c.

while not done and $i < n$ **do**
  $a = \pi(s_i, c_i)$
  τ_c.append(c_i), τ.append(s_i, a_i)
  $i+ = 1$
  s_i, done = environment(a)
  $c_i = \mathcal{M}(s_i, a)$
end while

Training dynamics of generative adversarial imitation learning (GAIL) (5) is more complicated than GAN. This is because the agent environment is a black box and this makes the optimization objective to be non-differentiable end-to-end. As a result, proper policy *rollouts* and Monte-Carlo estimation of policy gradients are required which makes the training dynamic more complicated. Therefore, the rest of the experiments are focused on the application of the FTNPL to GAIL. To formally explain the GAIL let $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$ be an infinite-horizon, discounted Markov decision process (MDP) with state-space $\mathcal{S}$, action space $\mathcal{A}$, transition probability distribution $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, reward function $r : \mathcal{S} \to \mathbb{R}$, distribution of the initial state s_0 $\rho_0 : \mathcal{S} \to \mathbb{R}$, the discount factor $\gamma \in (0, 1)$. In



(a) Coinrun with One level (b) Coinrun with Unbounded levels (c) Pendulum (d) Continuous Mountain Car

Figure 6: FTNPL applied to Imitation Learning.

the case of imitation learning, we are given access to a set of expert trajectories τ_E that are achieved using expert policy π_E. We are interested at estimating a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. To estimate π_E, GAIL optimizes the following:

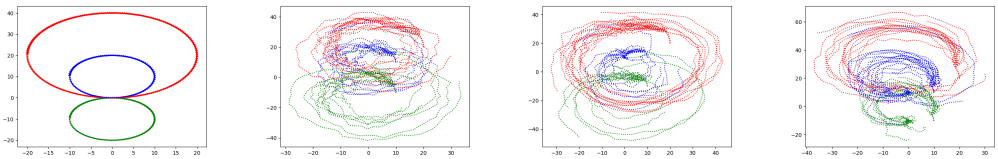
$$\min_{\pi} \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_{\pi}[\log D(s, a)] + \mathbb{E}_{\pi_E}[\log(1 - D(s, a))] - \lambda H(\pi) \tag{2}$$

, with the expected terms defined as $\mathbb{E}_{\pi}[D(s, a)] \triangleq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t D(s_t, a_t)]$, where $s_0 \sim \rho_0$, $a_t \sim \pi(a_t|s_t)$, $s_{t+1} \sim P(s_{t+1}|a_t, s_t)$, and $H(\pi) \triangleq \mathbb{E}_{\pi}[-\log \pi(a|s)]$ is the γ-discounted causal entropy. An adversary player D tries to distinguish state-action pairs generated during *rollout* using π from the demonstrated trajectories generated by π_E.

To apply FTNPL Alg. 1 to GAIL, we modify the rollout algorithm to Alg. 2 with $\mathcal{I} = (s, a)$. The concept of adding codes to the policy network is similar to infoGAIL (4). InfoGAIL uses fixed code to guide an entire trajectory. Moreover, it uses other regularization terms in the policy gradient optimization objective, to make sure that the codes would not be ignored. The correlated codes have different properties. First, they are generated per state-action (they also are fed into the discriminator) and therefore it addresses the multimodality and other types of variations within the trajectories as well. Second, there is no need to include any extra regularization terms including discounted causal entropy, i.e we assume $\lambda = 0$ in Eq. 2. Agent π uses PPO (41) for updates. We also used the utility definition of Wasserstein GAN (42) for our final implementation. The rest is a straightforward application of FTNPL 1 to GAIL. We also apply FTPL algorithm to GAIL as a baseline.

4.3.1 EXPRIEMENTS

Changing environments When learning from the expert trajectories π_E, one has to take into account the *internal* factors of variations such as learning in new environments that are not are demonstrated in the expert trajectories. For example, observations can vary for different levels of a game. Using the CoinRun environment (43), we set up an experiment to show that the introduction of correlated codes in FTNPL not only can address recurrent dynamic issues but also the internal factors of variations. CoinRun is a procedurally generated environment that has a configurable number of levels and difficulties. It can provide insight into an agent's ability to generalize to new and unseen environments. The game observations are high dimensional ($64 \times 64 \times 3$ RGB) and therefore it is also suitable for testing the efficiency of RL-based mediator of FTNPL under a complex environment. The only reward in the CoinRun environment is obtained by collecting coins, and this reward is a fixed positive constant. A collision with an obstacle results in the agent's death and levels vary widely in difficulty. The level terminates when the agent dies, the coin is collected, or after 1000 time steps. We first trained a PPO (41) agent using the 3-layer convolutional architecture proposed in (44). We stopped the training when it achieved a reward of 6.3 with 500 levels. We then generated expert trajectories with the same number of levels. However, in imitation learning training, two different levels are selected: one and an unbounded set of levels. A higher number of levels decrease the chance that a given environment gets encountered more than once. For the unbounded number of levels, this probability is almost 0. The selection of these different numbers of levels provides an insight into the adaptability and transferability of the imitation algorithms to new environments. We visualized the performance of FTNPL GAIL and GAIL, averaged over three different seeds. When there is only one level, the gap between the sample efficiency of GAIL and FTNPL-GAIL is not very wide as shown in Fig. 6a. However, with the increase in the number of levels (i.e changing



(a) Expert (b) GAIL (c) FTPL GAIL (d) FTNPL GAIL

Figure 7: Path of trajectories during training: The results are for the last 40K of the overall 200K steps.

environments), the FTNPL training outperforms GAIL more noticeably as demonstrated in 6b. We also did a few classic baselines for MountainCarContinuous-v0 in Fig. 6d and pendulum-v0 in Fig. 6c. FTNPL outperforms the other baselines and also has smaller variances across all the experiments.

Imitating mixture of state-action trajectories Another type of variation in π_E is the *external* one, such as when one is learning from a mixture of expert demonstrations. For this, we used the Synthetic 2D-CircleWorld experiment of (4). The goal is to select a direction strategy at time t using the observations of $t-4$ to t such that a path would mimic those demonstrated in expert trajectories τ_E. These expert trajectories are stochastic policies that produce circle-like trajectories. They contain three different modes as shown in Fig. 7. Proper imitation learning should have the ability to distinguish the mixture of experts from each other. The results in Fig. 7 demonstrate the path of learned trajectories during the last 40K of the overall 200K steps of training. It can be seen that FTNPL-GAIL can distinguish the expert trajectories and imitate the demonstrations more efficiently than FTPL-GAIL and GAIL.

5 Conclusion

We extended the 2-player adversarial learning frameworks by introducing a mediating neural agent whose role is to augment the observation of the players. We then proposed a novel follow the perturbed leader algorithm for training such 3-player architectures that guarantees convergence to mixed Nash equilibrium without recurrent dynamics and the loss-convexity assumption. As demonstrated in our experiments, it is also suitable for training in environments with various factors of variations thanks to its online learning nature.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[2] J. U. Schmidhuber, "Learning factorial codes by predictability minimization," 1992.

[3] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *NIPS*, 2016.

[4] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," in *NIPS*, 2017.

[5] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NIPS*, 2016.

[6] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions," *ArXiv*, vol. abs/1901.01753, 2019.

[7] A. S. Vezhnevets, S. Osindero, T. Schaul, N. M. O. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *ICML*, 2017.

[8] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *NIPS*, 2016.

[9] A. Azarafrooz and J. Brock, "Hierarchical soft actor-critic: Adversarial exploration via mutual information optimization," *ArXiv*, vol. abs/1906.07122, 2019.

[10] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," 1991.

[11] J. Schmidhuber, "Unsupervised minimax: Adversarial curiosity, generative adversarial networks, and predictability minimization," *ArXiv*, vol. abs/1906.04493, 2019.

[12] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 488–489, 2017.

[13] S. Racanière, T. Weber, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. M. O. Heess, Y. Li, R. Pascanu, P. W. Battaglia, D. Hassabis, D. Silver, and D. Wierstra, "Imagination-augmented agents for deep reinforcement learning," *ArXiv*, vol. abs/1707.06203, 2017.

[14] B. Poole, S. Ozair, A. van den Oord, A. A. Alemi, and G. Tucker, "On variational bounds of mutual information," in *ICML*, 2019.

[15] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu, "Decoupled neural interfaces using synthetic gradients," *ArXiv*, vol. abs/1608.05343, 2016.

[16] D. Balduzzi, S. Racanière, J. Martens, J. N. Foerster, K. Tuyls, and T. Graepel, "The mechanics of n-player differentiable games," *ArXiv*, vol. abs/1802.05642, 2018.

[17] J. P. Bailey and G. Piliouras, "Multi-agent learning in network zero-sum games is a hamiltonian system," *ArXiv*, vol. abs/1903.01720, 2019.

[18] P. Grnarova, K. Y. Levy, A. Lucchi, T. Hofmann, and A. Krause, "An online learning approach to generative adversarial networks," *ArXiv*, vol. abs/1706.03269, 2017.

[19] E. Hazan, K. Singh, and C. Zhang, "Efficient regret minimization in non-convex games," in *ICML*, 2017.

[20] N. Kodali, J. Hays, J. D. Abernethy, and Z. Kira, "On convergence and stability of gans," 2018.

[21] P. Mertikopoulos, C. H. Papadimitriou, and G. Piliouras, "Cycles in adversarial regularized learning," *ArXiv*, vol. abs/1709.02738, 2018.

[22] J. P. Bailey and G. Piliouras, "Multiplicative weights update in zero-sum games," in *EC '18*, 2018.

[23] G. Piliouras and J. S. Shamma, "Optimization despite chaos: Convex relaxations to complex limit sets via poincaré recurrence," in *SODA*, 2014.

[24] A. Gonen and E. Hazan, "Learning in non-convex games with an optimization oracle," in *COLT*, 2018.

[25] R. J. Aumann, "Correlated equilibrium as an expression of bayesian rationality," 1987.

[26] A. Blum and Y. Mansour, "From external to internal regret," *J. Mach. Learn. Res.*, vol. 8, pp. 1307–1324, 2005.

[27] L. E. Ortiz, R. E. Schapire, and S. M. Kakade, "Maximum entropy correlated equilibria," in *AISTATS*, 2006.

[28] Y. Freund and R. E. Schapire, "Adaptive game playing using multiplicative weights," 1999.

[29] A. Rakhlin and K. Sridharan, "Optimization, learning, and games with predictable sequences," *ArXiv*, vol. abs/1311.1869, 2013.

[30] S. Shalev-Shwartz and Y. Singer, "Convex repeated games and fenchel duality," in *NIPS*, 2006.

[31] N. Immorlica, A. T. Kalai, B. Lucier, A. Moitra, A. Postlewaite, and M. Tennenholtz, "Dueling algorithms," in *STOC '11*, 2011.

[32] C. H. Papadimitriou and G. Piliouras, "From nash equilibria to chain recurrent sets: Solution concepts and topology," in *ITCS '16*, 2016.

[33] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: a meta-algorithm and applications," *Theory of Computing*, vol. 8, pp. 121–164, 2012.

[34] J. Hofbauer, S. Sorin, and Y. Viossat, "Time average replicator and best-reply dynamics," *Math. Oper. Res.*, vol. 34, pp. 263–269, 2009.

[35] P. Mertikopoulos and W. H. Sandholm, "Learning in games via reinforcement and regularization," *Math. Oper. Res.*, vol. 41, pp. 1297–1324, 2014.

[36] P. D. Taylor and L. B. Jonker, "Evolutionarily stable strategies and game dynamics," 1978.

[37] A. Blum, N. Haghtalab, and A. D. Procaccia, "Variational dropout and the local reparameterization trick," in *NIPS*, 2015.

[38] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015.

[39] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[40] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *ArXiv*, vol. abs/1606.03498, 2016.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[42] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML-17, pp. 214–223, JMLR.org, 2017.

[43] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," *ArXiv*, vol. abs/1812.02341, 2018.

[44] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

A Appendix

CorEq is concerned with *joint* mixed strategies (e.g $\mu(\phi, \omega)$) which is and stronger and more restrictive notion than CCE. We therefore first define a simpler notion of MCorEq. We are particularly interested in its maximum entropic version similar to (27).

Marginal correlated equilibrium (MCorEq): Let $u_\pi(\phi) = \mathbb{E}_{\omega\sim\mu_D} L(\phi, \omega)$, $u_D(\omega) = -\mathbb{E}_{\phi\sim\mu_\pi} L(\phi, \omega)$ be the *marginal loss*. Also let $g_D(k, \hat{k}) = u_D[\hat{k}] - u_D[k]$ and $g_\pi(k, \hat{k}) = u_\pi[\hat{k}] - u_\pi[k]$ be discriminator's and agent's *marginal payoff gain* for selecting strategy k instead of $\hat{k}$. Marginal payoff g quantifies the motivation of the users to switch to other strategies. MCorEq is then a marginal mixed strategy (μ_π, μ_D) such that $\forall(\phi, \hat{\phi}), \forall(\omega, \hat{\omega})$, $\mu_\pi(\phi) > 0, \mu_D(\omega) > 0$, $g_\pi(\phi, \hat{\phi}) <= 0$ and $g_D(\omega, \hat{\omega}) <= 0$, i.e no user benefits (in the marginal loss sense) from switching strategies.

MeMCorEq: It is the solution to the convex optimization problem of $\mu^* = \arg\max_{\mu\in MCorEq} H(\mu)$ where H is the entropy. It is therefore a convex optimization problem.

Theorem 1. *When a continuous time MW algorithm is applied to zero-sum game with a fully mixed Nash equilibrium $\mu^* = (\mu_D^*, \mu_\pi^*)$, cross entropy between each evolving strategy $\mu(t)$ and the players' mixed Nash equilibrium $H(\mu^*, \mu) = -\sum_{\phi\in\Phi} \mu_\pi^*(\phi) \ln \mu_\pi(\phi) - \sum_{\omega\in\Omega} \mu_D^*(\omega) \ln \mu_D(\omega)$ remains constant.*

Proof. Various works including (34; 35) have shown that the continuous-time version of MW algorithm follows a replicator dynamic (36) as is described by:

$$\frac{\dot{\mu}_\pi(\phi)}{\mu_\pi(\phi)} = u_\pi(\phi) - \sum_{\hat{\phi}\in\Phi} \mu_\pi(\hat{\phi}) u_\pi(\hat{\phi}) \;\; ; \;\; \frac{\dot{\mu}_D(\omega)}{\mu_D(\omega)} = u_D(\omega) - \sum_{\hat{\omega}\in\Omega} \mu_D(\hat{\omega}) u_D(\hat{\omega}) \tag{3}$$

where as before $u_\pi(\phi), u_D(\omega)$ are the marginal loss and $\dot{u} = \frac{du}{dt}$.

It suffices to take time derivative of cross entropy term and plug in Eq. 3 and note the zero-sum nature of the game:

$$\begin{aligned} \frac{dH(\mu^*, \mu(t))}{dt} &= -\sum_{\phi} \mu_\pi^*(\phi) \frac{\dot{\mu}_\pi(\phi)}{\mu_\pi(\phi)} - \sum_{\omega} \mu_D^*(\omega) \frac{\dot{\mu}_D(\omega)}{\mu_D(\omega)} = \\ -\sum_{\phi} \mu_\pi^*(\phi)[u_\pi(\phi) - \sum_{\hat{\phi}\in\Phi} \mu_\pi(\hat{\phi}) u_\pi(\hat{\phi})] &- \sum_{\omega} \mu_D^*(\omega)[u_D(\omega) - \sum_{\hat{\omega}\in\Omega} \mu_D(\hat{\omega}) u_D(\hat{\omega})] = 0 \end{aligned} \tag{4}$$

□

Lemma 2. *Maximizing Nash entropy $H(\mu^*)$ implies convergence of MW to MNE is no longer weak/ergodic. Therefore no recurrent/cyclic dynamics exist.*

Proof. The degree of weakness in convergence measures of FTRL can be quantified using KL divergence $D_{KL}(\mu^*||\mu)$. Since $H(\mu^*, \mu) = H(\mu^*) + D_{KL}(\mu^*||\mu)$, and $H(\mu^*, \mu)$ is constant from Theorem 1, $D_{KL}(\mu^*||\mu) \to 0$ is equivalent to Maximizing Nash entropy $H(\mu^*)$. □

Let us refer to trajectory of $\forall \mu_t$ that $D_{KL}(\mu^*||\mu_t) = k$ as *KL orbit* as visualized in Fig 4. Lemma 2 then builds a useful intuition on how to avoid the cyclic behavior and to guarantee convergence (instead of ergodic convergence) to the MNE. By slowly maximizing $H(\mu^*)$, we can travel toward MNE one KL orbit at a time, until we reach an orbit with radius 0, i.e $D_{KL}(\mu^*||\mu_t) = 0$. At this point, convergence is no longer weak and no recurrent dynamic exists. However, it is not feasible to control $H(\mu^*)$ without interfering with the game as Nash μ^* is predetermined by $L(\phi, \omega)$. We propose to use a *neural network mediator agent* $\mathcal{M}$ that perturbs the original dynamic of the game by introducing auxiliary codes to players. We will show that a proper reward for $\mathcal{M}$ can be set to maximize Nash entropy $H(\mu^*)$ under the new perturbed game. However, without a game-theoretic formulation, players will *ignore* these perturbations. To address this, we use our predefined notion of MeMCorEq. The derivation MeMCorEq in our setup is not straightforward, as mediator $\mathcal{M}$ is perturbing the game dynamics sequentially as demonstrated in Fig. 4.

Theorem 3. *Any training dynamic of FTNPL applied to zero-sum game will converge to MNE* ***without recurrent dynamics*** *when the mediator appends correlated codes c to the inputs of both players according to the following reward function:*

$$r_m = -\sum_{i=0}^{T} \sum_{j=0}^{T} ReLU(g_\pi(\phi_i, \phi_j, c) + g_D(\omega_i, \omega_j, c)) \tag{5}$$

where $h^T = (\mu_0 ..., \mu_T)$ is history of strategies of the game up to time T and $\mu_t = (\phi_t, \omega_t)$, $ReLU(x) = \max(x, 0)$ and g is the marginal payoff gain defined in 2.1.

Proof. Dual problem of MeMCorEq is $\inf_{\lambda>=0} \ln(Z(\lambda)$ with the following relationship between the dual variables λ and the primal variables μ:

$$\begin{aligned} \ln(Z(\lambda)) = &- \ln \mathbb{E}_{\omega\sim\mu_D}[\exp(\sum_{\phi} \sum_{\hat{\phi}} \lambda_{\phi,\hat{\phi}} L(\phi, \omega) - L(\hat{\phi}, \omega))] \\ &- \mathbb{E}_{\phi\sim\mu_\pi}[\exp(\sum_{\omega} \sum_{\hat{\omega}} \lambda_{\omega,\hat{\omega}} L(\phi, \omega) - L(\phi, \hat{\omega}))] \end{aligned} \tag{6}$$

Instead of learning the Lagrangian multipliers λ, the mediator learns to introduce code c to the loss function $L(\phi, \omega, c)$ as if λ is absorbed into the loss function. To make sure that the Lagrangian constraints $\lambda >= 0$ are satisfied, we introduce ReLU function to the equation. This followed by the Jenson inequality yields:

$$\ln(Z(c)) \leq$$
$$\mathbb{E}_{\omega\sim\mu_D}[\sum_{\phi}\sum_{\hat{\phi}} ReLU(L(\hat{\phi},\omega,c) - L(\phi,\omega,c))] + \mathbb{E}_{\phi\sim\mu_\pi}[\sum_{\omega}\sum_{\hat{\omega}} ReLU(L(\phi,\hat{\omega},c) - L(\phi,\omega,c))] =$$
$$\sum_{i=0}^{T}\sum_{j=0}^{T} ReLU(g_\pi(\phi_i,\phi_j,c)) + ReLU(g_D(\omega_i,\omega_j,c)) \tag{7}$$

The last equality comes from the fact that MeMCorq is a stricter notion than CCE and therefore like other no-regret learning algorithms, the average of past strategies can be used as a proxy for computing MNE μ^*. The proof is then complete using the result of Lemma 2 and definition of MeMCorq.

□