Meta-RL for Multi-Agent RL: Learning to Adapt to Evolving Agents

Matthias Gerstgrasser **David C. Parkes**
School of Engineering and Applied Sciences
Harvard University
{matthias,parkes}@seas.harvard.edu

Abstract

In Multi-Agent RL, agents learn and evolve together, and each agent has to interact with a changing set of other agents. While generally viewed as a problem of non-stationarity, we propose that this can be viewed as a Meta-RL problem. We demonstrate an approach for learning Stackelberg equilibria, a type of equilibrium that features a bi-level optimization problem, where the inner level is a "best-response" of one or more follower agents to an evolving leader agent. Various approaches have been proposed in the literature to implement this best-response, most often treating each leader policy and the learning problem it induces for the follower(s) as a separate instance. We propose that the problem can be viewed as a meta (reinforcement) learning problem: Learning to learn to best-respond to different leader behaviors, by leveraging commonality in the induced follower learning problems. We demonstrate an approach using contextual policies and show that it matches performance of existing approaches using significantly fewer environment samples in experiments. We discuss how more advanced meta-RL techniques could allow this to scale to richer domains.

1 Introduction

Multi-Agent Reinforcement Learning (RL) is considered a hard problem, in part because it features what is generally thought of as a non-stationarity issue: As agents train and evolve together, so does the learning problem that each individual agent change - if other agents behave differently in a joint environment, my own actions might lead to different results for myself. We propose that this can be viewed as a meta-RL or multitask RL problem as well: agents need to learn to adapt quickly to changing behavior of other agents.

We focus here specifically on *Stackelberg equilibria*, which arise naturally in settings such as security games [1, 22] or (indirect) mechanism design [16, 24, 5]. In these equilibria, we have an asymmetric setting: A leader who commits to a strategy, and one or more followers who respond. The leader aims to maximize their reward, knowing that followers in turn will best-respond to the leader's choice of strategy. In the remainder of this paper, we will briskly formalize the setting and equilibrium concept in section 2, discuss how existing approaches implement the follower best-response and propose a novel approach using meta-RL techniques in section **??**, and experimentally validate it in section 4.

Prior Work Our work builds on prior approaches on learning Stackelberg equilibria [11, 18, 26, 2, 30] as well as RL for (indirect) mechanism design [29, 28, 3, 5, 6]. We utilize contextual policies [25, 13, 7, 19, 31, 9] in our own approach.

6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans.

2 Preliminaries

Markov games. We consider partially observable stochastic Markov games, essentially a multi-agent generalization of a partially observable Markov Decision Process (POMDP).

Definition 1 (Markov Game)**.** *A Markov Game $\mathcal{M}$ with n agents is a tuple $(S, A, T, r, \Omega, O, \gamma)$, consisting of a state space S, an action space $A = (A_1, ..., A_n)$, a (stochastic) transition function $T : S \times A \to S$, a (stochastic) reward function $r : S \times A \to \mathbb{R}^n$, a set of observations $\Omega = (\Omega_1, ..., \Omega_n)$, a (stochastic) observation function $O : S \times A \to \Omega$, and a discount factor γ.*

At each step t of the game, every agent i chooses an action $a_{i,t}$ from their action space A_i, the game state evolves according to the joint action $(a_{1,t}, \dots, a_{n,t})$ and the transition function T, and agents receive observations and reward according to O and R. An agent's behavior in the game is characterized by its policy $\pi_i : o_i \mapsto a_i$, which maps observations to actions. (We focus here mainly on this memory-less case.) Each agent in a Markov Game individually seeks to maximize its own (discounted) total reward $\sum_t \gamma^t r_i(s_t, a_{i,t}, a_{-i,t})$. This gives rise to the usual definitions of Nash equilibria (NE), correlated equilibria (CE), and coarse correlated equilibria (CCE), which we do not repeat in full here. Note that strategies in Markov games and in each of these equilibrium definitions are policies: A pair of policies π_1, π_2 in a two-player Markov game is a Nash equilibrium if neither agent can increase their expected total reward by unilaterally deviating.

Stackelberg Equilibria. Unlike all the above equilibrium concepts, a Stackelberg equilibrium is not symmetric: There is a special player, the leader, who commits to their strategy first; the other player (the follower) then chooses their best strategy given the leader's choice of strategies. This makes the leader potentially more powerful. This Stackelberg concept again also extends to Markov games: Here a leader agent L decides on their strategy, and the remaining (follower) agents observe and best-respond to this strategy. The leader seeks to maximize their own reward, considering that followers will always best-respond to their choice of strategy. For instance, in an iterated prisoner's dilemma [20], a leader might commit to a tit-for-tat strategy, in turn leading the follower to cooperate. We formally define this in Appendix A.

3 Meta-RL for Stackelberg RL

Stackelberg learning approaches in the literature can broadly be grouped into two categories based on how they implement the follower best-response, **optimization-based approaches** such as [17, 27], and **learning-based approaches** such as [5, 3, 29]. What is common to both and crucial to our discussion is that they approach the Stackelberg learning problem using a nested outer-inner loop, where for every learning iteration of the leader, an inner loop iteratively computes the followers' best response.

As our key contribution, we go beyond this and explore using multi-task and **meta-RL** as a means of implementing the follower oracle. We can recognize that the follower games, $\mathcal{F}_{s_L}$, are in fact a family of related problems. For this reason, the follower oracle problem can be seen as a multitask or meta-RL problem, and solved using techniques from those fields. We make use of *contextual policies* [25], where a context ω describes the task an agent is supposed to solve. In our case, the context provides the specific MDP among a family of MDPs a follower finds itself in, and ω is a description of the leader policy. This context, ω, is concatenated to the follower agent's observation $o_{i,t}$, and agent i observes $(o_{i,t}, \omega)$ at timestep t.

We focus on settings with a state space small enough to allow the leader policy to be fully understood with a small number of queries (i.e., the leader's action for every possible state), and we directly use the leader's response to a fixed set of queries as the context ω. For instance, in an iterated prisoner's dilemma, we ask the leader three questions: "How do you act on the initial step of the game?", "How do you act if the opponent cooperated in the previous step?" and "How do you act if the opponent defected in the previous step?" Clearly, if these are the only three possible states, this is sufficient to characterize the leader policy. We further use a two-stage training approach. In Phase 1, we train the follower against a different, randomized leader policy in each episode. By the end of this phase, the policy is able to best-respond to all possible leader policies. In Phase 2, we train a leader policy against this follower, where the leader is queried at the beginning of each episode. In the iterated matrix game setting in our experiments, we explicitly define the context, ω, in this way. For settings

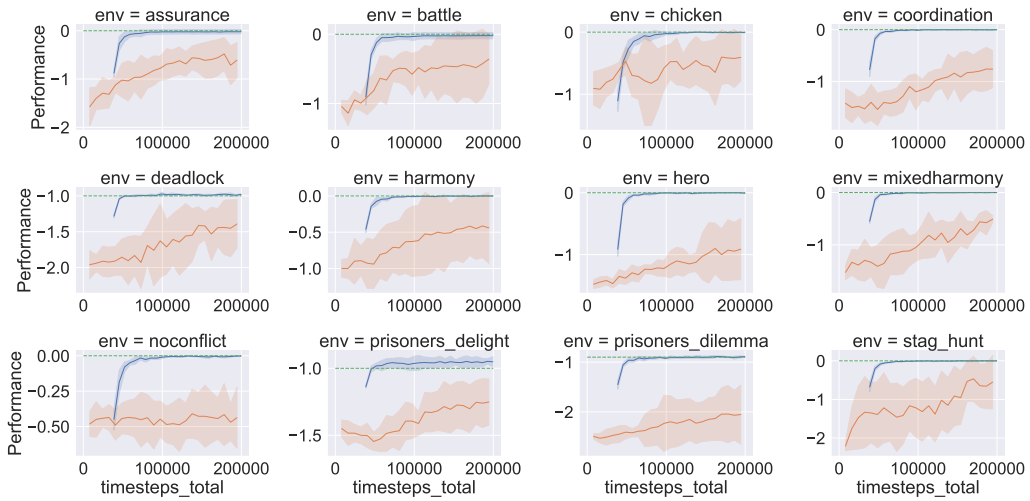

Figure 1: Blue: Performance of our novel PPO+Meta-RL approach on 12 canonical symmetric iterated matrix games. Orange: PPO+Q-learn [6]. Dashed green: Good Shepherd ES-MD [3] (final performance at 1.28B timesteps, estimated from Fig. 2 ibid.)

where this is not possible, the multitask and meta-RL literature provide a range of approaches that infer context, often using recurrent networks [25, 13, 7, 19, 31, 9].

4 Experiments

We evaluate our contextual policy approach and general framework on an ensemble of iterated symmetric matrix games, standard benchmark domains that allow comparison to prior approaches, as well as a novel domain based on a modified Atari 2600 game.

Iterated Matrix Games. In these environments, we play a matrix games for $n = 10$ steps per episode, and give agents a one-step memory. Therefore, there are five states in these Markov games: one for the initial steps of each episode, and four for later steps depending on the two agents' previous actions. At each step, each agent has a choice of two actions (e.g. "cooperate" or "defect").

Figure 1 shows the performance of our Meta-RL approach using PPO for the leader. We compare against the approaches of [3] and [6]. For our PPO+Meta-RL approach, we plot the combined environment steps used by the meta-follower training plus the leader training on the x-axis. For [3], we estimate performance from Figure 2 therein. Note that this is the eventual performance at the end of training ([3] do not publish learning curves).

We see that final performance largely matches that of the "good shepherd" ES-MD approach, and by extension also matches or outperforms all their baselines (c.f. Figures 1 and 2 therein). Importantly, notice that our approach converges in around 50k environment steps, whereas [3] report performance at 1.28 billion environment steps in the ES-MD case. We give further details on comparing performance to [3] in Appendix D.

We also see that our approach outperforms the PPO+Q-learn approach of [6]. In Appendix D we show the PPO+Q-learn approach training for significantly longer, and see that where it does converge it does so around 500k environment steps at the earliest, whereas for most of the harder cases it still has not nearly reached optimal performance at 2M timesteps. We again note that our approach shows greatly improved sample efficiency.

Environments: Bilateral Trade on Atari 2600 As a second, significantly higher-dimensional and challenging domain, we present a bilateral trade scenario on a modified Atari 2600 game (which are a state-of-the-art benchmark domain in single-agent RL). We use a two-player version of the game "Space Invaders", and introduce an artificial resource constraint: Each agent can only fire in

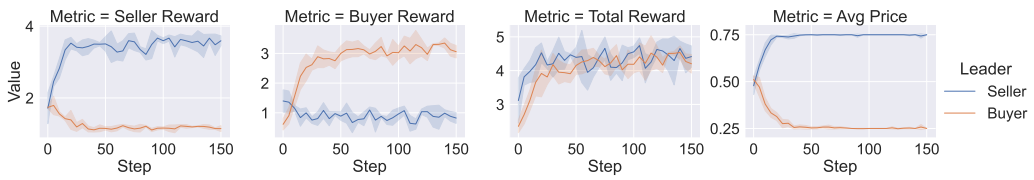

Figure 2: Performance of PPO+Meta-RL on Atari 2600 bilateral trade scenario. Plots show two Stackelberg equilibria: Agent 1 (seller) as leader (blue curves) and agent 2 (buyer) as leader (orange).

the game if they have a bullet available. Initially, neither player has any bullets available. Throughout the episode, we give bullets to player 1, one at a time at stochastic intervals. Player 1 can then choose to offer the sell this bullet to player 2 by offering them a price, or Player 1 can choose to use the bullet themselves. Player 2 in turn can choose to accept or reject a particular offer at a particular price. If a trade takes place, the sales price is added to player 1's reward, and subtracted from player 2's reward. Additionally, we introduce a reward scale imbalance: Each time player 1 successfully shoots an alien invader, they get a reward of 0.1. However each time player 2 shoots an alien, they get a much higher reward of 1.0. Noting that even well-trained AI agents do not hit every single shot they take, we should still expect that player 2 be able to generate just under 1.0 reward from each bullet they fire, and player 1 a much smaller reward of just under 0.1.

Clearly there is more total reward generated if player 1 sells all their bullets to player 2, with the difference referred to as the "gains from trade" in economics. However, notice that this is not a mechanism design setting (there is no mechanism), and also that there are two Stackelberg equilibria: If player 1 is the leader, then their optimal strategy is to offer bullets to player 2 at just under player 2's average utility per bullet. Player 2 will best respond by accepting the trade, still generate small positive reward, and player 1 will receive almost the entirety of the gains from trade. In the second Stackelberg equilibrium, player 2 is the leader. Player 2's optimal strategy is to refuse any price higher than just above player 1's average utility per bullet; and player 1's best response is to offer to sell at that (low) price. In this scenario, player 1 will be left with little more reward than had they kept and used the bullets themselves, and player 2 will receive almost all the gains from trade.

Figure 2 shows that our Meta-RL algorithm is able to successfully learn this for both equilibria. In this experiment we use discrete prices $(0, 0.25, 0.5, 0.75.1.0)$ for compatibility with the discrete Atari environment, so the results shown are the exact optimum.

5 Conclusion and Discussion

We have proposed and evaluated a novel approach to Stackelberg learning that uses contextual policies to implement the follower oracle, showing similar performance to previous approaches at much reduced sample complexity in experiments. In our approach, we view different leader agents the follower interacts with, and the follower learning problems they induce, as a family of related tasks.

In our current implementation, we pre-train the follower meta-policy and then freeze it during leader training. This works because we are working relatively small domains, where we can effectively sample enough of the leader policy space for the follower meta-policy to generalize to any leader it sees during leader training. In larger domains, we envision to keep training the follower during leader training, making this even more of a meta-RL (as opposed to multitask RL) approach. In fact, one prior work [29] already uses a similar approach, although they frame it as a curriculum approach used mainly to bootstrap learning rather than a meta-RL approach, and they only give partial context to the follower. Nevertheless, we see this as further evidence that a "meta-RL for multiagent-RL" approach has merit.

Further, we focus here specifically on Stackelberg equilibria but believe the same principle of viewing interaction with different or changing other agents as a meta-RL problem applies to other contexts as well. We note in particular similarities with zero-shot and few-shot interaction problems [10, 8], and suggest that synergies between these areas and meta-RL could be a fruitful area for further work.

References

[1] B. An, M. Tambe, and A. Sinha. Stackelberg security games (SSG) basics and application overview. *Improving Homeland Security Decisions*, page 485, 2017.

[2] Y. Bai, C. Jin, H. Wang, and C. Xiong. Sample-efficient learning of Stackelberg equilibria in general-sum games. *Advances in Neural Information Processing Systems*, 34:25799–25811, 2021.

[3] J. Balaguer, R. Koster, C. Summerfield, and A. Tacchetti. The good shepherd: An oracle agent for mechanism design. *arXiv preprint arXiv:2202.10135*, 2022.

[4] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[5] G. Brero, D. Chakrabarti, A. Eden, M. Gerstgrasser, V. Li, and D. C. Parkes. Learning Stackelberg equilibria in sequential price mechanisms. In *Proc. ICML Workshop for Reinforcement Learning Theory*, 2021.

[6] G. Brero, N. Lepore, E. Mibuari, and D. C. Parkes. Learning to mitigate ai collusion on economic platforms. *Advances in Neural Information Processing Systems*, 35, 2022.

[7] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[8] H. Hu, A. Lerer, A. Peysakhovich, and J. Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

[9] J. Humplik, A. Galashov, L. Hasenclever, P. A. Ortega, Y. W. Teh, and N. Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

[10] A. Lazaridou and M. Baroni. Emergent multi-agent communication in the deep learning era. *arXiv preprint arXiv:2006.02419*, 2020.

[11] J. Letchford, V. Conitzer, and K. Munagala. Learning and approximating the optimal strategy to commit to. In *International symposium on algorithmic game theory*, pages 250–262. Springer, 2009.

[12] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

[13] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

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

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[16] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 129–140, 1999.

[17] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 895–902, 2008.

[18] B. Peng, W. Shen, P. Tang, and S. Zuo. Learning optimal strategies to commit to. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2149–2156, 2019.

[19] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

[20] D. Robinson and D. Goforth. *The topology of the 2x2 games: a new periodic table*, volume 3. Psychology Press, 2005.

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

[22] A. Sinha, F. Fang, B. An, C. Kiekintveld, and M. Tambe. Stackelberg security games: Looking beyond a decade of success. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, 2018.

[23] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[24] C. Swamy. The effectiveness of Stackelberg strategies and tolls for network congestion games. In *SODA*, pages 1133–1142. Citeseer, 2007.

[25] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[26] K. Wang, L. Xu, A. Perrault, M. K. Reiter, and M. Tambe. Coordinating followers to reach better equilibria: End-to-end gradient descent for stackelberg games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5219–5227, 2022.

[27] H. Xu, F. Fang, A. X. Jiang, V. Conitzer, S. Dughmi, and M. Tambe. Computing minimax strategy for discretized spatio-temporal zero-sum security games. In *International Joint Workshop on Optimization in Multi-Agent Systems and Distributed Constraint Reasoning (OPTMASDCR) at AAMAS*. Citeseer, 2014.

[28] J. Yang, E. Wang, R. Trivedi, T. Zhao, and H. Zha. Adaptive incentive design with multi-agent meta-gradient reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1436–1445, 2022.

[29] S. Zheng, A. Trott, S. Srinivasa, D. C. Parkes, and R. Socher. The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science Advances*, 8(18):eabk2607, 2022.

[30] H. Zhong, Z. Yang, Z. Wang, and M. I. Jordan. Can reinforcement learning find Stackelberg-Nash equilibria in general-sum markov games with myopic followers? *arXiv preprint arXiv:2112.13521*, 2021.

[31] L. Zintgraf, M. Igl, K. Shiarlis, A. Mahajan, K. Hofmann, and S. Whiteson. Variational task embeddings for fast adapta-tion in deep reinforcement learning. In *International Conference on Learning Representations Workshop (ICLRW)*, 2019.

A Stackelberg Equilibrium Definition

Stackelberg equilibria can be extended to multiple followers, where best-response is not always unique: Any choice of leader strategy s_L induces a Markov game between F_{s_L} between the followers, which could feature multiple equilibria and even types of equilibria. This is often abstracted away from formally by defining the follower best-response as a (possibly stochastic) oracle $\mathcal{E}(\mathcal{F}_{s_L})$. For this discussion, we adopt this notation even for the single-follower case.

Definition 2 (Stackelberg equilibrium)**.** *Given a Markov Game $\mathcal{M}$ and a follower best-response oracle $\mathcal{E}$, a leader strategy s_L together with a tuple of follower strategies $\mathbf{s}_F$ is a* Stackelberg

equilibrium*, if s_L maximizes the leaders expected reward under the condition that follower strategies are drawn from $\mathcal{E}(\mathcal{F}_{s_L})$:*

$$s_L \in_{s_L} \mathop{\mathbb{E}}_{\mathbf{s}_F \sim \mathcal{E}(\mathcal{F}_{s_L})} \Big[\sum_t \mathbb{E}[r_L(s_t, a_L, \mathbf{a}_F)] \Big], \tag{1}$$

where the second expectation is drawing actions and state transitions from their respective policies $s_L, \mathbf{s}_F$ and transition function T, and the reward function is r, all as in Definition 1.

B Continuous Follower Learning

A standard approach is to use a (no-regret or reinforcement) learning algorithm to implement the follower best-response. In some prior work [3], it is explicitly stated that learning starts "from scratch" i.e. with randomly initialized weights after every leader policy update. In other approaches [29, 5], followers train starting from their weights before the leader update, i.e. they train continuously without a weight re-initialization. The latter approach recognizes, as we do, that the follower learning problems are not entirely unrelated - in particular, if the leader policy only changes a little in each update, then it seems reasonable to assume that follower policies should also converge to a point only a little distance from their best-response to the previous leader policy. In turn, if they start from their previous weights, they may only need fewer training steps to reach there. While intuitively this makes sense, and may work in practice in many settings, we show here that this approach can also lead to learning failure.

In an experiment, we trained a leader and follower in the "Battle of the Sexes" single-shot matrix game from Example **??**, using a standard inner-loop outer-loop approach. The follower best-response is implemented using a simple tabular Q-learning approach. We ran two versions of this: In one, the follower Q-table was reset between leader updates; in the other, it was not. Figure 3 shows leader performance for four different seeds for both approaches. As can be seen, when the follower policy is reset between leader updates, learning consistently converges to the unique Stackelberg equilibrium in the game. When follower learning is continuous, however, the system converges to a different behavior. In one seed, learning failed entirely; in the other three, agents converge to a non-Stackelberg behavior, specifically the other pure Nash equilibrium, where they coordinate on the follower's preferred choice. This happens because the follower gets "stuck" in that behavior: Notice that for a uniformly random leader policy which goes to each restaurant with equal probability, the follower's best response is to go to their preferred restaurant (yielding happiness 2 with probability 0.5, compared to happiness 1 with probability 0.5 when going to less-preferred restaurant). In fact, this is true for any leader policy that goes to the follower's preferred restaurant with probability at least $1/3$. Therefore, for a randomly initialized leader policy, that is the best-response behavior the follower will likely learn. With continuous follower learning, the follower might then never move away from this behavior, even if the leader explores a policy where this is not the best-response. In turn, the leader may never explore that part of its policy space for long enough.

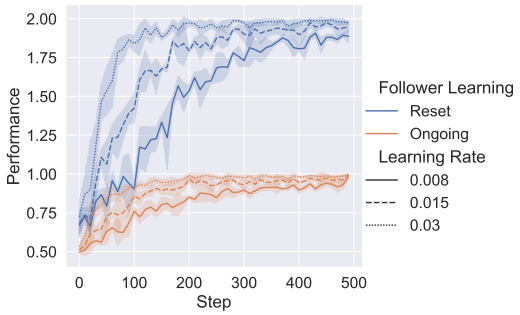

Figure 3: Leader reward in "Battle of the Sexes" with Q-learning follower best-response with weight re-initialization between leader updates (blue) and continuous learning (orange).

Whether this failure happens in practice will depend on the specific setting, choice of leader and follower learning algorithms and their hyperparameters. However, we take the fact that it *can* happen as further evidence that a more rigorous approach on generalizing between leader policies is warranted.

C Further Experiment Details

Environments. We use the 12 canonical symmetric matrix games identified in [20] and also used by [3]. During training, we scale rewards to be centered at 0, i.e. taking values -1.5, -0.5, 0.5, 1.5, but we report results offset to match the reward scales used by [3]. This has no effect on comparability of results. We use $n = 10$ steps per episode.

Algorithm. Algorithm 1 details the two-phase learning algorithm we use. In all the experiments shown in the main text, we use policy gradient to train the follower meta-policy in the pre-training loop. We use PG [23], PPO [21] and DQN [14, 15] in the main training loop, as indicated in the respective figures. We use linear models, and disable exploration in the leader policy while pre-training the follower and vice versa. Table D lists the hyperparameters used for each of these algorithms. Any hyperparameters not listed were left at default values in **rllib** version 2.0.0. All experiments were run with a single rollout worker (per experiment), and using Torch.

Equilibrium Verification. At the end of every experiment, we freeze the leader policy and further train the follower policy for $n = 50$ iterations. Unlike in the pre-training phase, we here train them only against the specific leader policy trained in the main training loop. This is to further verify that the policies indeed form a Stackelberg equilibrium, and in particular that the follower meta-policy is best-responding to the trained leader. If this is the case, we should not see any change in leader or follower performance in this post-training phase. If the follower meta-policy was *not* already best-responding to the leader, we may see an increase in follower performance during this post-training phase. In all of the experiments in this paper (except the ones designed to show failure modes) we see no follower improvement, i.e. behavior consistent with a Stackelberg equilibrium. This is not shown in the training curves in the figures, but can be reproduced from the source code.

Implementation and Environment. All experiments were implemented using Ray / RLlib 2.0.0 [12]. Experiments were run on recent Intel Xeon processors with a single core and 2GB RAM per experiment.

Hyperparameter Tuning. Learning rates and batch sizes were tuned using grid search, with some additional tuning using HyperOpt and BayesOpt Python packages, yielding no further improvement however.

D Further Details on Performance Comparisons

In Figure 1 we compare our Meta-RL approach with the PPO+Q-learn approach of [6] and the ES-MD approach of [3].

For [5], we implement follower Q-learning using information therein. Hyperparameters for both the leader and the follower were tuned using the HyperOpt package [4]. In Figure 1 we plot learning curves up to 200k timesteps, as our approach converges before that point. We show in Figure **??** learning curves until 2M timesteps. We can see that in some cases PPO+Q-learn eventually converges to the optimum, while in the majority of cases this still has not happened by 2M timesteps.

For [3], we estimate their performance from Figure 2 therein. Notice that that figure is *not* a learning curve, but represents a single inner loop at the end of their training procedure. In the ES-MD case, [3] report their performance after 1.28 billion environment steps. In the Diff-MD case, a comparison of sample complexity is difficult, as that approach uses a description of the environment rather than sample access. The closest we can come to a like-for-like comparison is noting that [3] report performance for Diff-MD after 500k computed expected reward per 10-step episode. In some sense this could be seen to be equivalent to 5M environment steps as a lower bound.

Algorithm 1 Contextual Policy

Pre-Training Loop
Initialize follower policy π_F
for each pre-training iteration **do**
  for each episode per sample batch **do**
    Sample a random leader policy π_L^r
    for each $o_L \in O_L$ **do**
      Query π_L^r for $\pi_L^r(o_L)$
    end for
    Set context $\omega = \pi_L^r(o_L), o_L \in O_L$
    for each episode step **do**
      Return $o_{L,t}$ to leader, $(\omega, o_{F,t})$ to follower
      Step environment using $a_L = \pi_L^r(o_L), a_F = \pi_F(\omega, o_{F,t})$
    end for
  end for
  Update follower policy π_F using collected sample batch using PG/PPO/DQN
end for
Main Training Loop
Initialize leader policy π_L
for each training iteration **do**
  for each episode per sample batch **do**
    for each $o_L \in O_L$ **do**
      Query π_L for $\pi_L(o_L)$
    end for
    Set context $\omega = \pi_L(o_L), o_L \in O_L$
    for each episode step **do**
      Return $o_{L,t}$ to leader, $(\omega, o_{F,t})$ to follower
      Step environment using $a_L = \pi_L^r(o_L), a_F = \pi_F(\omega, o_{F,t})$
    end for
  end for
  Update leader policy π_L using collected sample batch using PG/PPO/DQN
end for

Table 1: Hyper-Parameter Configuration Table

Follower Policy Gradient			
Hyper-Parameter	**Value**	**Hyper-Parameter**	**Value**
algorithm	PG	rollout_fragment_length	100
lr	0.02	train_batch_size	100
iterations	500	batch_mode	complete_episodes
Leader Policy Gradient			
Hyper-Parameter	**Value**	**Hyper-Parameter**	**Value**
algorithm	PG	rollout_fragment_length	100
lr	0.156	train_batch_size	100
iterations	1200	batch_mode	complete_episodes
Leader PPO			
Hyper-Parameter	**Value**	**Hyper-Parameter**	**Value**
algorithm	PPO	rollout_fragment_length	1000
lr	0.008	train_batch_size	1000
entropy_coeff	0.0	sgd_minibatch_size	1000
iterations	500	batch_mode	complete_episodes
Leader DQN			
Hyper-Parameter	**Value**	**Hyper-Parameter**	**Value**
algorithm	SimpleQ	rollout_fragment_length	10
lr	0.001	train_batch_size	1024
learning_starts	5000	exploration_type	ParameterNoise
exploration_initial_stddev	1.0	exploration_random_timesteps	0
iterations	20000	batch_mode	complete_episodes