Learning to Play Against Any Mixture of Opponents

Max Olan Smith [1] **Thomas Anthony** [2] **Yongzhao Wang** [1] **Michael P. Wellman** [1]

Abstract

Intuitively, experience playing against one mixture of opponents in a given domain should be relevant for a different mixture in the same domain. We propose a transfer learning method, *Q-Mixing*, that starts by learning Q-values against each pure-strategy opponent. Then a Q-value for *any* distribution of opponent strategies is approximated by appropriately averaging the separately learned Q-values. From these components, we construct policies against all opponent mixtures without any further training. We empirically validate Q-Mixing in two environments: a simple grid-world soccer environment, and a social dilemma game. We find that Q-Mixing is able to successfully transfer knowledge across any mixture of opponents. We next consider the use of observations during play to update the believed distribution of opponents. We introduce an opponent classifier—trained in parallel to Q-learning, reusing data—and use the classifier results to refine the mixing of Q-values. We find that Q-Mixing augmented with the opponent policy classifier performs better, with higher variance, than training directly against a mixed-strategy opponent.

1. Introduction

Reinforcement learning (RL) agents commonly interact in environments with other agents, whose behavior may be uncertain. For any particular probabilistic belief over the behavior of another agent (henceforth, *opponent*), we can learn to play with respect to that opponent distribution (henceforth, *mixture*), for example by training in simulation against opponents sampled from the mixture. If the mixture changes, ideally we would not have to train from scratch, but rather could *transfer* what we have learned to construct a policy to play against the new mixture.

Traditional RL algorithms include no mechanism to explicitly prepare for variability in opponent mixtures. Instead, current solutions either learn a new behavior or update a previously learned behavior. In lieu of that, researchers designed methods for learning a single behavior successfully across a set of strategies (Wang & Sandholm, 2003), or quickly adapting in response to new strategies (Jaderberg et al., 2019). In this work, we explicitly tackle the unique problem of responding to new opponent mixtures without requiring further simulations for learning.

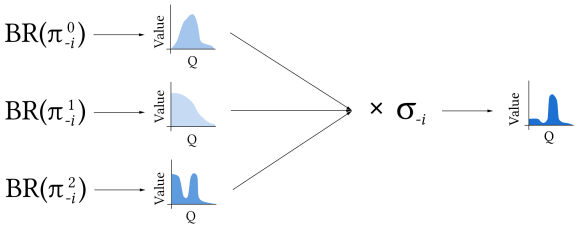

Figure 1. Q-Mixing concept. BR to each of the pure strategies π_{-i} are learned separately. The Q-value for a given opponent mixed strategy σ_{-i} is then derived by combining these components.

We propose a new algorithm, *Q-Mixing*, that effectively transfers learning across opponent mixtures. The algorithm is designed within a population-based learning regime (Jaderberg et al., 2019; Lanctot et al., 2017), where training is conducted against a known distribution of opponent policies. Q-Mixing initially learns value-based best responses (BR), represented as Q-functions, with respect to each of the opponent's pure strategies. It then transfers this knowledge against any given opponent mixture by weighting the Q-functions according to the probability that the opponent plays each of its pure strategies. The idea is illustrated in Figure 1. This calculation is an approximation of the Q-values against the mixed-strategy opponent, with error due to misrepresenting the future belief of the opponent by the current belief. The result is an approximate BR policy against any mixture, constructed without additional training.

This situation is common in game-theoretic approaches to multiagent RL such as Policy-Space Response Oracles (PSRO) (Lanctot et al., 2017) where a population of agents is trained and reused. Smith et al. (2021) showed that combining PSRO and Q-Mixing results in large reductions of PSRO's cumulative training time.

We experimentally validate our algorithm on: (1) a simple

[1]University of Michigan [2]Deepmind. Correspondence to: Max Olan Smith <mxsmith@umich.edu>.

grid-world soccer game, and (2) a sequential social dilemma game. Our experiments show that Q-Mixing is effective in transferring learned responses across opponent mixtures.

We also address two potential issues with combining Q-functions according to a given mixture. The first is that the agent receives information *during play* that provides evidence about the opponent's strategy. We address this by introducing an opponent classifier to predict which opponent we are playing for a particular state and reweighting the Q-values to focus on the likely opponents. The second issue is that the complexity of the policy produced by Q-Mixing grows linearly with the support of the opponent's mixture. This can make simulation and decision making computationally expensive. We propose and show that policy distillation (Rusu et al., 2015) can compress a Q-Mixing policy while maintaining performance.

Key Contributions:(1) Theoretically relate (in idealized setting) the Q-value for an opponent mixture to Q-values for mixture components. (2) A new transfer learning algorithm, *Q-Mixing*, that uses this relationship to construct approximate policies against any given opponent mixture, without additional training. (3) Augmenting this algorithm with runtime opponent classification, to account for observations that can inform predictions of opponent policy during play. (4) Demonstrate that policy distillation can reduce Q-Mixing's complexity (in both memory and computation).

2. Preliminaries

At any time $t \in \mathcal{T}$, an agent receives the environment's state $s^t \in \mathcal{S}$, or an observation $o^t \in \mathcal{O}$, a partial state. (Even if an environment's state is fully observable, the inclusion of other agents with uncertain behavior makes the overall system partially observable.) From said observation, the agent takes an action $a^t \in \mathcal{A}$ receiving a reward $r^t \in \mathbb{R}$. The agent's *policy* describes its behavior given each observation $\pi : \mathcal{O} \to \Delta(\mathcal{A})$. Actions are received by the environment, and a next observation is determined following the environment's transition dynamics $p : \mathcal{O} \times \mathcal{A} \to \mathcal{O}$.

The agent's goal is to maximize its reward over time; called the *return*: $G^t = \sum_l^\infty \gamma^l r^{t+l}$, where γ is the discount factor weighting the importance of immediate rewards. Return is used to define the values of being in of a given observation:

$$V(o^t) = \mathbb{E}_\pi \left[\sum_{l=0}^{\infty} \gamma^l r(o^{t+l}, a^{t+l}) \right],$$

and taking an action given an observation:

$$Q(o^t, a^t) = r(o^t, a^t) + \gamma \mathbb{E}_{o^{t+1} \in \mathcal{O}} \left[V(o^{t+1}) \right].$$

For multiagent settings, we index the agents and distinguish the agent-specific components with subscripts. Agent i's policy is $\pi_i : \mathcal{O}_i \to \Delta(\mathcal{A}_i)$, and the opponent's policy[1] is the negated index, $\pi_{-i} : \mathcal{O}_{-i} \to \Delta(\mathcal{A}_{-i})$. Boldface elements are joint across the agents (e.g., joint-action $\boldsymbol{a}$).

Agent i has a strategy set Π_i comprising the possible policies it can employ. Agent i may choose a single policy from Π_i to play as a *pure strategy*, or randomize with a *mixed strategy* $\sigma_i \in \Delta(\Pi_i)$. Note that the pure strategies π_i may themselves choose actions stochastically. For a mixed strategy, we denote the probability the agent plays a particular policy π_i as $\sigma_i(\pi_i)$. A *best response* (BR) to an opponent's strategy σ_{-i} is a policy with maximum return against σ_{-i}.

Agent i's prior belief about its opponent is represented by an opponent mixed-strategy, $\sigma^0_{-i} \equiv \sigma_{-i}$. The opponent plays the mixture by sampling a policy according to σ_{-i} at the start of the episode. They are locked into the sampled policy's behavior for the entire duration of the episode. The agent's updated belief at time t about the opponent policy faced is denoted σ^t_{-i}.

We introduce the term *Strategy Response Value* (SRV) to refer to the observation-value against an opponent's strategy.

Definition 1 (Strategic Response Value)**.** *An agent's π_i* strategic response value *is its expected return given an observation, when playing π_i against a specified opponent strategy:*

$$V_{\pi_i}(o^t_i \mid \sigma^t_{-i}) = \mathbb{E}_{\sigma^t_{-i}} \left[\sum_{\boldsymbol{a}} \boldsymbol{\pi}(a_i \mid o^t_i) \sum_{o'_i, r_i} p(o'_i, r_i \mid o^t_i, \boldsymbol{a}) \delta \right]$$

where $\delta \equiv r_i + \gamma V_{\pi_i}(o'_i \mid \sigma^{t+1}_{-i})$. Let the optimal SRV *be*

$$V^*_i(o^t_i \mid \sigma^t_{-i}) = \max_{\pi_i} V_{\pi_i}(o^t_i \mid \sigma^t_{-i}).$$

From the SRV, we define the Strategic Response Q-Value (SRQV) for a particular opponent strategy.

Definition 2 (Strategic Response Q-Value)**.** *An agent's π_i* strategic response Q-value *is its expected return for an action given an observation, when playing π_i against a specified opponent strategy:*

$$Q_{\pi_i}(o^t_i, a^t_i \mid \sigma^t_{-i}) = \mathbb{E}_{\sigma^t_{-i}} \left[r^t_i \right] + \gamma \mathbb{E}_{o^{t+1}_i} \left[V_{\pi_i}(o^{t+1}_i \mid \sigma^{t+1}_{-i}) \right],$$

where $r^t_i \equiv r_i(o^t_i, a^t_i, a^t_{-i})$. Let the optimal SRQV *be*

$$Q^*_i(o^t_i, a^t_i \mid \sigma^t_{-i}) = \max_{\pi_i} Q_{\pi_i}(o^t_i, a^t_i \mid \sigma^t_{-i}).$$

3. Q-Mixing

Our goal is to transfer the Q-learning effort across different opponent mixtures. We consider the scenario where

[1] Our methods are defined here for environments with two agents. Extension to greater numbers while maintaining computational tractability is a topic for future work.

we first learn against each opponent's pure strategy. From this, we construct a Q-function for a given distribution of opponents from the policies trained against each opponent's pure strategy.

3.1. Single-State Setting

Let us first consider a simplified setting with a single state. This is essentially a problem of bandit learning, where our opponent's strategy will set the reward of each arm for an episode. Intuitively, our expected reward against a mixture of opponents is proportional to the payoff against each opponent weighted by their respective likelihood.

As shown in Theorem 1, weighting the component SRQV by the opponent's distribution supports a BR to that mixture. We call this relationship *Q-Mixing-Prior* and define it in Theorem 1 (proof provided in Section A).

Theorem 1 (Single-State Q-Mixing)**.** *Let $Q_i^*(\cdot \mid \pi_{-i})$, $\pi_{-i} \in \Pi_{-i}$, denote the optimal strategic response Q-value against opponent policy π_{-i}. Then for any opponent mixture $\sigma_{-i} \in \Delta(\Pi_{-i})$, the optimal strategic response Q-value is given by*

$$Q_i^*(a_i \mid \sigma_{-i}) = \sum_{\pi_{-i} \in \Pi_{-i}} \sigma_{-i}(\pi_{-i}) Q_i^*(a_i \mid \pi_{-i}).$$

3.2. Leveraging Information from the Past

Next, we consider the RL setting where both agents are able to influence an evolving observation distribution. As a result of the joint effect of agents' actions on observations, the agents have an opportunity to gather information about their opponent during an episode. Methods in this setting need to (1) use information from the past to update its belief about the opponent, and (2) grapple with uncertainty about the future. To bring Q-Mixing into this setting we need to quantify the agent's current belief about their opponent and their future uncertainty.

During a run with an opponent's pure strategy drawn from a distribution, the actual observations experienced generally depend on the identity of this pure strategy. Let $\psi_i : \mathcal{O}_i^{0:t} \to \Delta(\Pi_{-i})$ represent the agent's current belief about the opponent's policy using the observations during play as evidence. From this prediction, we propose an approximate version of Q-Mixing that accounts for past information. The approximation works by first predicting the relative likelihood of each opponent given the current observation. Then it weights the Q-value-based BRs against each opponent by their relative likelihood.

Figure 2 provides an illustration of the benefits and limitations of this new prediction-based Q-Mixing. At any given timestep t during the episode, the information available to an agent about the opponents may be insufficient to perfectly identify their policy. The yellow area above a timestep represents the uncertainty reduction from an updated prediction of the opponent σ_{-i}^t compared to the baseline prediction of the prior σ_{-i}^0. Crucially, this definition of Q-Mixing does not consider updating the opponent distribution from new information in the future (blue area in Figure 2).

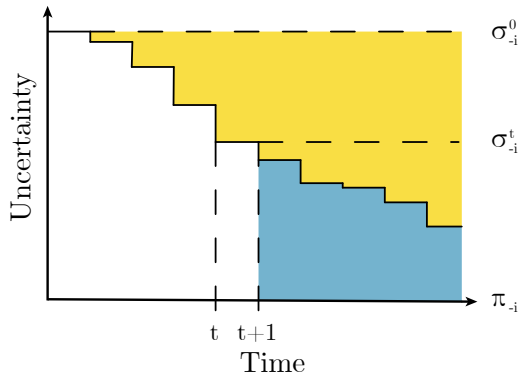

Figure 2. Opponent uncertainty over time. The yellow area represents uncertainty reduction as a result of updating belief about the distribution of the opponent. The blue area represents approximation error incurred by Q-Mixing.

Let the previously defined ψ be the *opponent policy classifier* (OPC), which predicts the opponent's policy. In this work, we consider a simplified version of this function, that operates only on the agent's current observation $\psi_i : \mathcal{O}_i \to \Delta(\Pi_{-i})$. We then augment Q-Mixing to weight the importance of each BR as follows:

$$Q_{\pi_i}(o_i, a_i \mid \sigma_{-i}) = \sum_{\pi_{-i}} \psi_i(\pi_{-i} \mid o_i, \sigma_{-i}) Q_{\pi_i}(o_i, a_i \mid \pi_{-i}).$$

We refer to this quantity as *Q-Mixing*, or *Q-Mixing-X*, where X describes ψ (e.g., *Q-Mixing-Prior* defines ψ as the prior uncertainty of the opponent σ_{-i}^0). By continually updating the opponent distribution during play, the adjusted Q-Mixing result better responds to the actual opponent.

An ancillary benefit of the opponent classifier is that poorly estimated Q-values tend to have their impact minimized. For example, if an observation occurs only against the second pure strategy, then the Q-value against the first pure strategy would not be trained well, and thus could distort the policy from Q-Mixing. These poorly trained cases correspond to unlikely opponents and get reduced weighting in the version of Q-Mixing augmented by the classifier.

3.3. Accounting for Future Uncertainty

To account for future uncertainty, Q-Mixing must be able to update its successor observation values given future observations. This can be done by expanding the Q-value into its components: expected reward under the current belief

in the opponent's policy, and our expected next observation value. By updating the second term to recursively reference a new opponent belief we can account for changing beliefs in the future. The extended formulation, Q-Mixing-Value-Iteration (QMVI), is given by:

$$\delta = r_i(o_i^t, a_i^t \mid \pi_{-i}) + \gamma \mathbb{E}_{o_i^{t+1}}\left[V^*(o_i^{t+1} \mid \sigma_{-i})\right],$$
$$Q_i^*(o_i^t, a_i^t \mid \sigma_{-i}) = \sum_{\pi_{-i} \in \Pi_{-i}} \psi_i(\pi_{-i} \mid o_i^t, \sigma_{-i}) \cdot \delta.$$

If we assume that we have access to both a dynamics model and the observation distribution dependent on the opponent, then we can directly solve for this quantity through Value Iteration (Algorithm 1). These requirements are quite strong, essentially requiring perfect knowledge of the system with regards to all opponents. The additional step of Value Iteration also carries a computational burden, as it requires iterating over the full observation and action spaces. Though these costs may render QMVI infeasible in practice, we provide Algorithm 1 in Section A as a way to ensure correctness in Q-values.

4. Experiments

4.1. Grid-World Soccer

We first evaluate Q-Mixing on a simple grid-world soccer environment (Littman, 1994; Greenwald & Hall, 2003). This environment has small state and action spaces, allowing for inexpensive simulation. With this environment we pose the following questions: (1) Can QMVI obtain Q-values for mixed-strategy opponents? (2) Can Q-Mixing transfer Q-values across all of the opponent's mixed strategies?

The soccer environment is composed of a soccer field, two players, one ball, and two goals. The player's objective is to acquire the ball and score a goal while preventing the opponent from scoring on their own goal. The scorer receives $+1$ reward, and the opponent receives -1 reward. The state of the environment consists of the entire field including the locations of the players and ball. This is represented as a 5×4 matrix with six possible values in each cell, referring to the occupancy of the cell. The players may move in any of the four cardinal directions or stay in place. Actions taken by the players are executed in a random order, and if the player possessing the ball moves last then the first player may take possession of the ball by colliding with them. A graphical example of the soccer environment can be seen in Figure 7.

In our experiments, we learn policies for Player 1 using Double DQN (van Hasselt et al., 2016). The state space is ravelled into a vector of length 120 that is then fed into a deep neural network. The network has two fully-connected hidden layers of size 50 with ReLU activation functions and an output layer over the actions with size 5. Player 2 plays a strategy over five policies, each using the same shape neural networks as Player 1, generated using the double oracle (DO) algorithm (McMahan et al., 2003). These policies are frozen for the duration of the experiments. Further details of the environment and experiments are in Section B.1.

4.1.1. Empirical Verification of Q-Mixing

We now turn to our first question: whether QMVI can obtain Q-values for mixed-strategy opponents. To answer this, we run the QMVI algorithm against a fixed opponent mixed strategy (Algorithm 1). We construct dynamics models for each opponent by considering the opponent's policy and the multiagent dynamics model as a single entity. Then we may approximate the relative observation-occupancy distribution by rolling out 30 episodes against each policy and estimating the distribution.

In our experiment, an optimal policy was reached in fourteen iterations. The resulting policy best-responded to each individual opponent and the mixture. This empirically validates our first hypothesis.

4.1.2. Coverage of Opponent Strategy Space

Our second question is whether Q-Mixing can produce high-quality responses for any opponent mixture. Our evaluation of this question employs the same five opponent pure strategies as the previous experiment. We first trained a baseline, *BR(Uniform)* or $\text{BR}(\sigma_{-i})$, directly against the uniform mixed-strategy opponent. The baseline was trained using 300000 simulation steps. The same hyperparameters were used to train against each of the opponent's pure strategies, with the simulation budget split equally. The Q-values trained respectively are used as the components for Q-Mixing, and an OPC is also trained from their replay buffers. The OPC has the same neural network architecture as the policies, but modifies the last layer to predict over the size of the opponent's strategy set.

We evaluate each method against all opponent mixtures truncated to the tenths place (e.g., [0.1, 0.6, 0.1, 0.2, 0.0]), resulting in 860 strategy distributions. This is meant to serve as a representative coverage of the entire opponent's mixed-strategy space. For each one of these mixed-strategies, we simulate the performance of each method against that mixture for thirty episodes. We then collect the average payoff against each opponent mixture and sort the averages in descending order.

Figure 3 shows Q-Mixing's performance across the opponent mixed-strategy space. Learning in this domain is fairly easy, so both methods are expected to win against almost every mixed-strategy opponent. Nevertheless, Q-Mixing gen-

eralizes across strategies better, albeit with slightly higher variance. While the improvement of Q-Mixing is incremental, we interpret this first evaluation as validating the promise of Q-Mixing for coverage across mixtures.

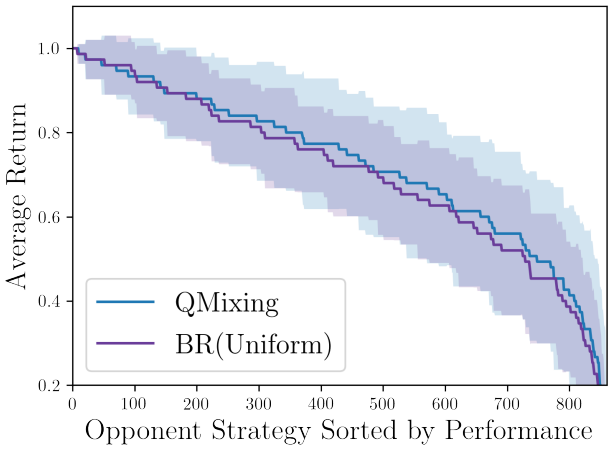

Figure 3. Q-Mixing's coverage of the opponent's strategy space in the soccer game. The strategies are sorted per-BR-method by the BR's return. Shaded region represents a 95% confidence interval over five random seeds ($df = 4,\ t = 2.776$). BR(Mixture) is trained against the uniform mixture opponent. The two methods use the same number of experiences.

4.2. Sequential Social Dilemma Game

In this section, we evaluate the performance of Q-Mixing in a more complex setting, in the domain of sequential social dilemmas (SSD). These dilemmas present each agent with difficult strategic decision where they must balance collective and self interest throughout repeated interactions. Like our grid-world soccer example, our SSD *Gathering* game has two players, but unlike the simpler game it is general sum. Most importantly, the game exhibits imperfect observation, that is, the environment is only partially observable and the players have private information. We investigated the following research questions on this environment: (1) Can Q-Mixing work with Q-values based on observations from a complex environment? (2) Can the use of an opponent policy classifier that updates the opponent distribution improve performance? (3) Can policy distillation be used to compress a Q-Mixing based policy, while preserving performance?

The Gathering game is a gridworld instantiation of a tragedy-of-the-commons game (Perolat et al., 2017). In this game, the players compete to harvest apples from an orchard; however, the regrowth rate of the apples is proportional to the number of nearby apples. Limited resources puts the players at odds, because they must work together to maintain the orchard's health while selfishly amassing apples. In addition to picking apples, the players may tag all players in a beam in front of them removing them from the game for a short fixed duration. Moreover, the agents face partial observation as they are only able to see a small window in front of them and do not know the full state of the game.

A visualization of the environment and more details are included in Section B.2. The policies used in this environment are implemented as deep neural networks with two hidden layers with 50 units and ReLU activations. rAgents can choose to move in the four cardinal directions, turn left or right, tag the other player, or perform no action. The policies are trained using the Double DQN algorithm (van Hasselt et al., 2016), and the opponent's parameters are always kept fixed (no training updates are performed) for the duration of the experiments.

All statistical results below are reported with a 95% confidence interval based on the Student's t-distribution ($df = 4,\ t = 2.776$). To generate this interval, we perform almost the entire experimental process, as described in each experiment's subsection, entirely across five random seeds. For each seed we do not resample hyperparameters nor generate new opponents. A final result is produced for each random seed, often resulting in a sample of payoffs for a variety of player profiles. Then the sample statistics from each random seed are utilized to construct the confidence interval.

4.2.1. Empirical Verification of Q-Mixing

We experimentally evaluate Q-Mixing-Prior on the Gathering game, allowing us to confirm our algorithm's robustness to complex environments. First, a set of three opponent policies are generated through DO. A BR is trained against each of the independent pure-strategies. A baseline BR(σ_{-i}) is trained directly against the uniform mixture of the same opponents. We evaluate Q-Mixing-Prior's ability to transfer strategic knowledge by first simulating its performance against the mixed-strategy opponent. Then we compare this performance to the performance that BR(σ_{-i})'s strategy that was learned by training directly against the mixed-strategy.

The training curves for BR(σ_{-i}) is presented in Figure 4. From this graph we can see that Q-Mixing-Prior is able to achieve performance stronger than BR(σ_{-i}) through transfer learning. This is possible because the pure-strategy BR are able to learn stronger policies from specialization compared to the mixed-strategy BR. For example, when playing only against π^0_{-i} the best-response BR(π^0_{-i}) achieves a return of 227.93 ± 21.01 while BR(σ_{-i}) achieves 207.98 ± 15.92. This verifies our hypothesis that Q-Mixing is able to transfer knowledge under partial observation.

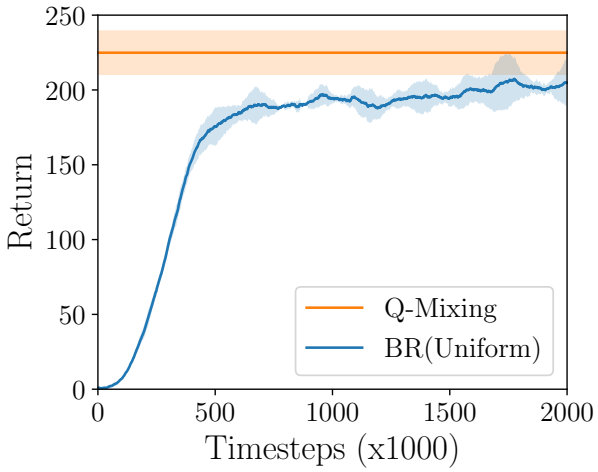

Figure 4. Learning curve of BR(σ_{-i}) compared to the performance of Q-Mixing-Prior on the Gathering game. Shaded region represents a 95% confidence interval over five random seeds ($df = 4,\ t = 2.776$).

4.2.2. Opponent Classification

Our next research question is: can the use of an OPC that updates the opponent distribution in Q-Mixing improve its performance? During play against an opponent sampled from the mixed-strategy, the player is able to gather evidence about which opponent they are playing. We hypothesize that leveraging this evidence to weight the importance of the respective BR's Q-values higher will improve Q-Mixing's performance.

To verify this hypothesis, we train an OPC using the replay buffers associated with each BR policy. These are the same buffers that were used to train the BRs, and cost no additional compute to collect. This data is used to train an OPC that outputs a distribution over opponent pure strategies for each observation. The OPC is implemented with a deep neural network two hidden layers with 50 units and ReLU activations. To train this classifier we take each experience from the pure-strategy BR replay buffers and assign them each a class label for each respective pure-strategy. The classifier is then trained to predict this label using a cross-entropy loss.

We evaluate Q-Mixing-OPC by testing the performance on a representative coverage of the mixed-strategy opponents illustrated in Figure 5. We found that the Q-Mixing-OPC policy performed stronger against the full opponent strategy coverage supporting our hypothesis that an OPC can identify the opponent's pure-strategy and enable Q-Mixing to chose the correct BR policy. However, our method has a much larger variance, which can be explained by the OPC's misclassification of the opponent resulting in poorly chosen actions throughout the episode.

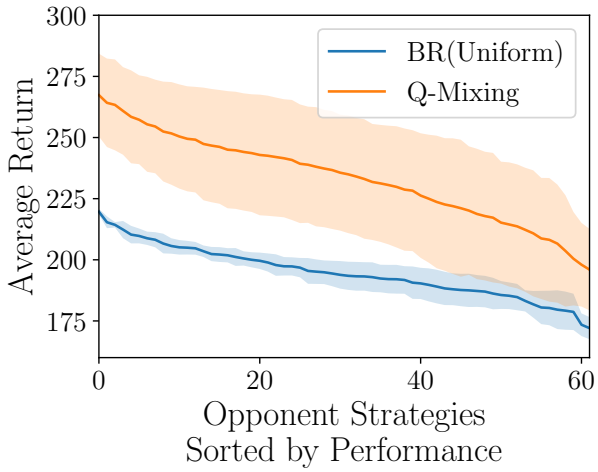

Figure 5. Q-Mixing with an OPC's coverage of the opponent's strategy space on the Gathering game. Each strategy is a mixture over the 3 opponent's truncated to the tenths place. The strategies are sorted by the respective BR's performance. Shaded regions represent a 95% confidence interval over five random seeds ($df = 4,\ t = 2.776$).

4.2.3. Policy Distillation

In Q-Mixing we need to compute Q-values for each of the opponent's pure strategies. This can be a limiting factor in parametric policies, like deep neural networks, where our policy's complexity grows linearly in the size of the support of the opponent's mixture. This can become unwieldy in both memory and computation. To remedy these issues, we propose using policy distillation to compress a Q-Mixing policy into a smaller parametric space (Hinton et al., 2014).

In the policy distillation framework, a larger neural network referred to as the *teacher* is used as a training target for a smaller neural network called the *student*. In our experiment, the Q-Mixing policy is the teacher to a student neural network that is the size of a single best-response policy. The student is trained in a supervised learning framework, where the dataset is the concatenated replay buffers from training pure-strategy best-responses. This is the same dataset that was used in opponent classifying, which was notably generated without running any additional simulations. A batch of data is sampled from the replay-buffer and the student predicts Q^S the teacher's Q^T Q-values for each action. The student is then trained to minimize the KL-divergence between the predicted Q-values and the teacher's true Q-values. There is a small wrinkle, the policies produce Q-values, and KL-divergence is a metric over probability distributions. To make this loss function compatible, the Q-values are trans-

formed into a probability distribution by softmax with temperature τ. The temperature parameter allows us to control the softness of the maximum operator. A high temperature produces actions that have a near-uniform probability, and as the temperature is lowered the distribution concentrates weight on the highest Q-Values (Sutton & Barto, 2018). The benefit of a higher temperature is that more information can be passed from the teacher to the student about each state. Additional training details are described in Section B.3.

The learning curve of the student is reported in Figure 6. We found that the student policy was able to recover the performance of Q-Mixing-Prior, albeit with slightly higher variance. This study did not include any effort to optimize the student's performance, thus further improvements with the same methodology may be possible. This result confirms our hypothesis that policy distillation is able to effectively compress a policy derived by Q-Mixing.

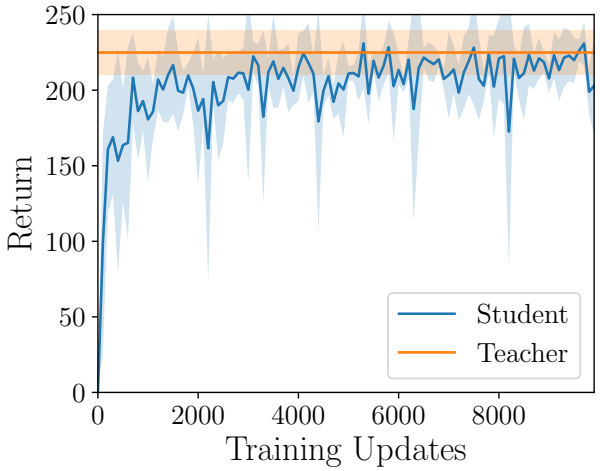

Figure 6. Policy distillation simulation performance over training on the Gathering game. The teacher Q-Mixing-Prior is used as a learning target for a smaller student network. Shaded region represents a 95% confidence interval over five random seeds ($df = 4,\ t = 2.776$).

5. Related Work

5.1. Multiagent Learning

The most relevant line of work in multiagent learning studies the interplay between centralized learning and decentralized execution (Tan, 1993; Kraemer & Banerjee, 2016). In this regime, agents are able to train with additional information about the other agents that would not be available during evaluation (e.g., the other agent's state (Rashid et al., 2018), actions (Claus & Boutilier, 1998), or a coordination signal (Greenwald & Hall, 2003)). The key question then becomes: how to create a policy that can be evaluated without additional information? A popular approach is to decompose the joint-action value into independent Q-values for each agent (Guestrin et al., 2001; He et al., 2016; Sunehag et al., 2018; Rashid et al., 2018; Mahajan et al., 2019). An alternative approach is to learn a centralized critic, which can train independent agent policies (Gupta et al., 2017; Lowe et al., 2017; Foerster et al., 2018b). Some work has proposed constructing meta-data about the agent's current policies as a way to reduce the learning instability present in environments where other agents' policies are changing (Foerster et al., 2017; Omidshafiei et al., 2017).

A set of assumptions that can be made is that all players have fixed strategy sets. Under these assumptions, agents could maintain more sophisticated beliefs about their opponent (Zheng et al., 2018), and extend this to recursive-reasoning procedures (Yang et al., 2019). These lines of work focus more on other-player policy identification and are a promising future direction for improving the quality of the OPC. One more potential extension of the OPC is to consider alternative objectives. Instead of focusing exclusively on predicting the opponent, in safety-critical situations an agent will want to consider an objective that accounts for inaccurate prediction of their opponent. The Restricted Nash Response Johanson et al. (2007) encapsulates this measure by balancing maximal performance if the prediction is correct balanced with reasonable performance if the prediction is inaccurate. While both of these directions of work focus around opponent-policy prediction, they do so under a largely different problem statement. Most notably, these works do not consider varying the distribution of the opponent policies, and as such extending these works to fit this new problem statement would constitute its own study.

Instead of building or using complete models of opponents, one may use an implicit representation of their opponents. By choosing to build an explicit model of their opponent they circumvent needing a large amount of data to reconstruct the opponent policy. An additional benefit is that there are less likely to be errors in the model that need to be overcome, because a perfect reconstruction of a complex policy is no longer necessary. He et al. (2016) proposes DRON which uses a learned latent action prediction of the opponent as conditioning information to the policy (in a similar nature to the opponent-actions in the joint-action value area). They also show another version DRON which uses a Mixture-of-Experts (Jacobs et al., 1991) operation to marginalize over the possible opponent behaviors. More formally, they compute the marginal $\sum_{a_{-i}} \pi_{-i}(a_{-i}|s_i)Q_i(s_i, a_i, a_{-i})$, which is over the action-space and utilized to condition the expected Q-value. Q-Mixing is built off a similar style of marginalization; however, it marginalizes over the policy-space of the opponent instead of the action-space. Moreover, Q-Mixing depends on independent BR Q-values against

each opponent-policy, where DRON learns a single Q-network. Bard et al. (2013) proposes implicitly modelling opponents through the payoffs received from playing against a portfolio of the agent's policies.

Most multiagent learning work focuses on the simultaneous learning of many agents, where there is not a distribution over a static set of opponent policies. This difference in methods can have strong influences on the final learned policies. For example, when a policy is trained concurrently with another particular opponent-policy they may overfit to each other's behavior. As a result, the final learned policy may be unable to coordinate or play well against any other opponent policy (Bard et al., 2020) Another potential problem is that each agent now faces a dynamic learning problem, where they must learn a moving target (the other agent's policy) (Foerster et al., 2018a; Tesauro, 2003).

5.2. Multi-Task Learning

Multiagent learning is analogous to multi-task learning. In this reconstruction, each strategy/policy is analogous to solving a different task. And the opponent's strategy would be the distribution over tasks. Similar analogies to tasks can be made with objectives, goals, contexts, etc. (Kaelbling, 1993; Ruder, 2017).

The multi-task community has roughly separated learnable knowledge into two categories (Snel & Whiteson, 2014). *Task relevant* knowledge pertains to a particular task (Jong & Stone, 2005; Walsh et al., 2006); meanwhile, *domain relevant* knowledge is common across all tasks (Caruana, 1997; Foster & Dayan, 2002; Konidaris & Barto, 2006). Work has been done that bridges the gap between these settings; for example, knowledge about a task could be a curriculum to utilize over tasks (Czarnecki et al., 2018). In task relevant learning, a leading method is to identify state information that is irrelevant to decision making, and abstract it away (Jong & Stone, 2005; Walsh et al., 2006). Our work falls into the same task relevant category, where we are interested in learning responses to particular opponent policies. What differentiates our work from the previous work is that we learn Q-values for each task independently, and do not ignore any information.

Progressively growing neural networks is another similar line of work (Rusu et al., 2016), focused on a stream of new tasks. Schwarz et al. (2018) also found that network growth could be handled with policy distillation.

5.3. Transfer Learning

Transfer learning is the study of reusing knowledge to learn new tasks/domains/policies. Within transfer learning, we look at either how knowledge is transferred, or what kind of knowledge is transferred. Previous work on how to transfer knowledge has tended to follow one of two main directions (Pan & Yang, 2010; Lampinen & Ganguli, 2019). The *representation transfer* direction considers how to abstract away general characteristics about the task that are likely to apply to later problems. Ammar et al. (2015) present an algorithm where an agent collects a shared general set of knowledge that can be used for each particular task. The second direction directly transfers parameters across tasks; appropriately called *parameter transfer*. Taylor et al. (2005) show how policies can be reused by creating a projection across different tasks' state and action spaces.

In the literature, transferring knowledge about the opponent's strategy is considered intra-agent transfer (Silva & Costa, 2019). The focus of this area is on *adapting to other agents*. One line of work in this area focuses on ad hoc teamwork, where an agent must learn to quickly interact with new teammates (Barrett & Stone, 2015; Bard et al., 2020). The main approach relies on already having a set of policies available, and learning to select which policy will work best with the new team (Barrett & Stone, 2015). Another work proposes learning features that are independent of the game, which can either be qualities general to all games or strategies (Banerjee & Stone, 2007). Our study differs from these in its focus on the opponent's policies as the source of information to transfer.

6. Conclusions

This paper introduces Q-Mixing, an algorithm for transferring knowledge across distributions of opponents. We show how Q-Mixing relies on the theoretical relationship between an agent's action-values, and the strategy employed by the other agents. A first empirical confirmation of the approach is demonstrated using a simple grid-world soccer environment. In this environment, we show how experience against pure strategies can transfer to construction of policies against mixed-strategy opponents. Moreover, we show that this transfer is able to cover the space of mixed strategies with no additional computation.

Next, we tested our algorithm's robustness on a sequential social dilemma. In this environment, we show the benefit of introducing an opponent policy classifier, which uses the agent's observations to update its belief about the opponent's policy. This updated belief is then used to revise the weighting of the respective BR Q-values.

Finally, we address the concern that a Q-Mixing policy may become too large or computationally expensive to use. To ease this concern we demonstrate that policy distillation can be used to compress a Q-Mixing policy into a much smaller parameter space.

References

Ammar, H. B., Eaton, E., Luna, J. M., and Ruvolo, P. Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In *24th International Conference on Artificial Intelligence*, IJCAI, pp. 3345–3349, 2015.

Banerjee, B. and Stone, P. General game learning using knowledge transfer. In *20th International Joint Conference on Artificial Intelligence*, IJCAI, pp. 672–677, 2007.

Bard, N., Johanson, M., Burch, N., and Bowling, M. Online implicit agent modelling. In *12th International Conference on Autonomous Agents and Multiagent Systems*, pp. 255–262, 2013.

Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., Parisotto, E., Dumoulin, V., Moitra, S., Hughes, E., Dunning, I., Mourad, S., Larochelle, H., Bellemare, M. G., and Bowling, M. The Hanabi challenge: A new frontier for AI research. *Artificial Intelligence*, 280, 2020.

Barrett, S. and Stone, P. Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In *29th AAAI Conference on Artificial Intelligence*, AAAI, pp. 2010–2016, 2015.

Caruana, R. Multitask learning. *Machine Learing*, 28(1): 41–75, 1997.

Claus, C. and Boutilier, C. The dynamics of reinforcement learning in cooperative multiagent systems. In *15th National Conference on Artificial Intelligence*, AAAI, pp. 746–752, 1998.

Czarnecki, W., Jayakumar, S., Jaderberg, M., Hasenclever, L., Teh, Y. W., Heess, N., Osindero, S., and Pascanu, R. Mix & match agent curricula for reinforcement learning. In *35th International Conference on Machine Learning*, ICML, pp. 1087–1095, 2018.

Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H. S., Kohli, P., and Whiteson, S. Stabilising experience replay for deep multi-agent reinforcement learning. In *34th International Conference on Machine Learning*, ICML, pp. 1146–1155, 2017.

Foerster, J., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. Learning with opponent-learning awareness. In *17th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pp. 122–130, 2018a.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *32nd AAAI Conference on Artificial Intelligence*, AAAI, pp. 2974–2982, 2018b.

Foster, D. and Dayan, P. Structure in the space of value functions. *Machine Learning*, 49(2):325–346, 2002.

Greenwald, A. and Hall, K. Correlated-Q learning. In *20th International Conference on Machine Learning*, ICML, pp. 242–249, 2003.

Guestrin, C., Koller, D., and Parr, R. Multiagent planning with factored MDPs. In *14th International Conference on Neural Information Processing Systems*, NeurIPS, pp. 1523–1530, 2001.

Gupta, J. K., Egorov, M., and Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In Sukthankar, G. and Rodriguez-Aguilar, J. A. (eds.), *16th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pp. 66–83, 2017.

He, H., Boyd-Graber, J., Kwok, K., and III, H. D. Opponent modeling in deep reinforcement learning. In *33rd International Conference on Machine Learning*, ICML, pp. 1804–1813, 2016.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NeurIPS: Deep Learning and Representation Learning Workshop*, 2014.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural Computing*, 3 (1):79–87, March 1991.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castañeda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., Sonnerat, N., Green, T., Deason, L., Leibo, J. Z., Silver, D., Hassabis, D., Kavukcuoglu, K., and Graepel, T. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Johanson, M., Zinkevich, M., and Bowling, M. Computing robust counter-strategies. In *Advances in Neural Information Processing Systems 20*, 2007.

Jong, N. K. and Stone, P. State abstraction discovery from irrelevant state variables. In *19th International Joint Conference on Artificial Intelligence*, IJCAI, pp. 752–757, 2005.

Kaelbling, L. P. Learning to achieve goals. In *13th International Joint Conference on Artificial Intelligence*, IJCAI, pp. 1094–1099, 1993.

Konidaris, G. and Barto, A. Autonomous shaping: Knowledge transfer in reinforcement learning. In *23rd International Conference on Machine Learning*, ICML, pp. 489–496, 2006.

Kraemer, L. and Banerjee, B. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82 – 94, 2016.

Lampinen, A. K. and Ganguli, S. An analytic theory of generalization dynamics and transfer learning in deep linear networks. In *7th International Conference on Learning Representations*, ICLR, 2019.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. In *31st International Conference on Neural Information Processing Systems*, NuerIPS, pp. 4193–4206, 2017.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. Multi-agent reinforcement learning in sequential social dilemmas. In *16th International Conference on Autonomous Agents and Multiagent Systems*, 2017.

Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. In *11th International Conference on International Conference on Machine Learning*, ICML, pp. 157–163, 1994.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *31st International Conference on Neural Information Processing Systems*, NeurIPS, pp. 6382–6393, 2017.

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. Maven: Multi-agent variational exploration. In *33rd Conference on Neural Information Processing Systems*, NeurIPS, 2019.

McMahan, H. B., Gordon, G. J., and Blum, A. Planning in the presence of cost functions controlled by an adversary. In *20th International Conference on International Conference on Machine Learning*, ICML, pp. 536–543, 2003.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., and Vian, J. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *34th International Conference on Machine Learning*, ICML, 2017.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22 (10):1345–1359, Oct 2010.

Perolat, J., Z. Leibo, J., Zambaldi, V., Beattle, C., Tuyls, K., and Graepel, T. A multi-agent reinforcement learning model of common-pool resource appropriation. In *31st Conference on Neural Information Processing Systems*, 2017.

Rashid, T., Samvelyan, M., Witt, C. S. d., Farquhar, G., Foerster, J., and Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *35th International Conference on Machine Learning*, ICML, pp. 4295–4304, 2018.

Ruder, S. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. Policy distillation. In *International Conference on Learning Representations*, ICLR, 2015.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & compress: A scalable framework for contiual learning. In *35th International Conference on Machine Learning*, ICML, 2018.

Silva, F. and Costa, A. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64, 03 2019.

Smith, M. O., Anthony, T., and Wellman, M. P. Iterative empirical game solving via single policy best response. In *9th International Conference on Learning Representations*, 2021.

Snel, M. and Whiteson, S. Learning potential functions and their representations for multi-task reinforcement learning. In *13th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pp. 637–681, 2014.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. Value-decomposition networks for cooperative multi-agent learning. In *17th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, 2018.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, second edition, 2018.

Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *10th International Conference on Machine Learning*, ICML, pp. 330–337, 1993.

Taylor, M. E., Stone, P., and Liu, Y. Value functions for RL-based behavior transfer: A comparative study. In *20th National Conference on Artificial Intelligence*, AAAI, pp. 880–885, 2005.

Tesauro, G. Extending Q-learning to general adaptive multi-agent systems. In *16th International Conference on Neural Information Processing Systems*, NeurIPS, pp. 871–878, 2003.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double Q-learning. In *30th AAAI Conference on Artificial Intelligence*, AAAI, pp. 2094–2100, 2016.

Walsh, T. J., Li, L., and Littman, M. L. Transferring state abstractions between MDPs. In *ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

Wang, X. and Sandholm, T. Learning near-Pareto-optimal conventions in polynomial time. In *16th International Conference on Neural Information Processing Systems*, NeurIPS, pp. 863–870, 2003.

Yang, T., Hao, J., Meng, Z., Zhang, C., Zheng, Y., and Zheng, Z. Efficient detection and optimal response against sophisticated opponents. In *International Joint Conference on Artificial Intelligence*, 2019.

Zheng, Y., Meng, Z., Hao, J., Zhang, Z., Yang, T., and Fan, C. A deep bayesian policy reuse approach against non-stationary agents. In *32nd Conference on Neural Information Processing Systems*, 2018.

A. Q-Mixing

Theorem 1 (Single-State Q-Mixing)**.** *Let $Q_i^*(\cdot \mid \pi_{-i})$, $\pi_{-i} \in \Pi_{-i}$, denote the optimal strategic response Q-value against opponent policy π_{-i}. Then for any opponent mixture $\sigma_{-i} \in \Delta(\Pi_{-i})$, the optimal strategic response Q-value is given by*

$$Q_i^*(a_i \mid \sigma_{-i}) = \sum_{\pi_{-i} \in \Pi_{-i}} \sigma_{-i}(\pi_{-i}) Q_i^*(a_i \mid \pi_{-i}).$$

Proof. The definition of Q-value is as follows (Sutton & Barto, 2018):

$$Q_i^*(a_i) = \sum_{r_i} p(r_i \mid a_i) \cdot r_i.$$

In a multiagent system, the dynamics model p suppresses the complexity introduced by the other agents. We can unpack the dynamics model to account for the other agents as follows:

$$p(r_i \mid a_i) = \sum_{\pi_{-i}} \sum_{a_{-i}} \pi_{-i}(a_{-i}) \cdot p(r_i \mid \boldsymbol{a}).$$

We can then unpack the strategic response value as follows:

$$Q_i^*(a_i \mid \pi_{-i}) = \sum_{a_{-i}} \pi_{-i}(a_{-i}) \sum_{r_i} p(r_i \mid \boldsymbol{a}) \cdot r_i.$$

Now we can rearrange the expanded Q-value to explicitly account for the opponent's strategy. The independence assumption enables the following re-writing by letting us treat the opponent's mixed strategy as a constant condition.

$$\begin{aligned} &Q_i^*(a_i \mid \sigma_{-i}) \\ &\quad = \sum_{r_i} \sum_{\pi_{-i}} \sigma_{-i}(\pi_{-i}) \sum_{a_{-i}} \pi_{-i}(a_{-i}) p(r_i \mid \boldsymbol{a}) \cdot r_i \\ &\quad = \sum_{\pi_{-i}} \sigma_{-i}(\pi_{-i}) \sum_{a_{-i}} \pi_{-i}(a_{-i}) \sum_{r_i} p(r_i \mid \boldsymbol{a}) \cdot r_i \\ &\quad = \sum_{\pi_{-i}} \sigma_{-i}(\pi_{-i}) Q_i^*(a_i \mid \pi_{-i}). \end{aligned}$$

□

B. Experimental Details

Double DQN was used for all experiments (van Hasselt et al., 2016). Determining the correct hyperparameters to utilize is a non-trivial problem, because the learning dynamics may vary given different opponent policies. To this end, we sought to find a method for selecting hyperparameters that performs well against a diversity in opponents, while also being computationally tractable to run. We choose hyperparameters that performed best against a uniform-mixture of a fixed set of opponents (five for the soccer environment, and three for cyber-security environment). The opponent policies were generated through PSRO, and were sampled from the resulting strategy sets. For both experiments we also included a random opponent to each strategy set, because we expect this opponent to be one of the more challenging opponents to learn against.

However, there's a chicken-and-egg problem present. In order to utilize PSRO we would also need the aforementioned hyperparameters. As a stop-gap, we choose initial hyperparameters, by evaluating their performance against a random opponent. In summary, for both environments we select hyperparameters by:

1. Sample 200 possible hyperparameter settings, and choose the one that best-performs against a random opponent.
2. Run PSRO until it exceeds a three day walltime.
3. Sample a fixed set of policies from the strategy set generated from PSRO.
4. Sample 200 hyperparameter settings, and evaluate them against uniform mixed-strategy of the four PSRO policies and the random opponent.

We chose to evaluate our hyperparameters against the mixed-strategy opponent, because we believed it offered the most benefit to the baseline method. Future work could look at the interplay of the hyperparameter selection method and the respective performance of both Q-Mixing, and learning a BR directly against a mixed-strategy.

The non-standard hyperparameters listed throughout the appendix are defined as follows:

Timesteps Total number of experiences collected during training.

Exploration Fraction Fraction of the training timesteps used for exploration. The exploration policy is ϵ-greedy, and starts with $\epsilon = 1.0$, and linearly decays to *Exploration Final* hyperparameter.

Exploration Final ϵ The final ϵ value.

Training Frequency Timestep frequency for performing updates.

Training Starts Number of timesteps experienced before training begins.

Number of Simulations The number of simulated episodes performed for evaluation.

Algorithm 1: Value Iteration: Q-Mixing

Input: $\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \epsilon, \gamma$
$V_0(s \mid \sigma_{-i}) \leftarrow \sum_{\pi_{-i}} \sigma_{-i}(\pi_{-i}) Q(s, a \mid \pi_{-i})$
do
  $Q_t(s, a \mid \sigma_{-i}) \leftarrow \sum_{\pi_{-i}} \psi(\pi_{-i} \mid s, \sigma_{-i}) \sum_{s',\, r} \mathcal{T}(s', r \mid s, a, \pi_{-i}) \left[r + \gamma V_{t-1}(s' \mid \sigma_{-i})\right]$
  $V_t(s \mid \sigma_{-i}) \leftarrow \max_a Q_t(s, a \mid \sigma_{-i})$
  $\pi_t(s \mid \sigma_{-i}) \leftarrow \arg\max_a Q_t(s, a \mid \sigma_{-i})$
while $\exists_{s \in \mathcal{S}} \, |V_t(s) - V_{t-1}(s)| > \epsilon$
Output: $V_t,\ Q_t,\ \pi_t$

B.1. Soccer

The soccer environment is a gridworld comprised of two players, one ball, and two goals. The soccer field is a 5×4 matrix where the goals are off-field on the left and right sides of the field. The ball can spawn in one two positions in the middle of the field, and the players spawn on either side of the spawn points. A graphical representation of the soccer environment can be see in Figure 7. The players are rewarded for moving the ball into the opponent's goal (the one they spawned furthest from).

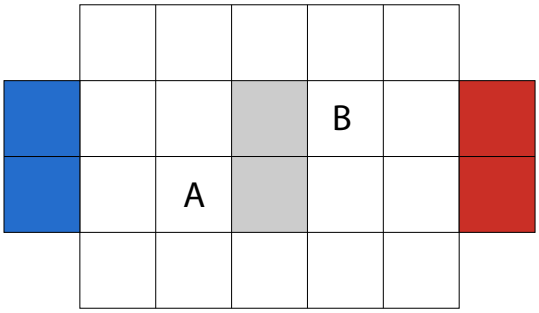

Figure 7. Grid-world soccer environment. The letters represent the respective players. The ball may spawn in either middle highlighted tile, and the player's goal is to score in the opposite net.

In this experiment we select five opponent policies, and hold them fixed throughout the experiments. We consider the hyperparameters in Table 2 as candidates, and found the hyperparameters listed in Table 1 to perform best.

To ensure that each method receives the same simulation budget, we allow each pure-strategy BR 60000 timesteps. An interesting future direction is investigating the trade-off of simulation budget and performance that exists between these methods.

The trained DQN was approximated using a 2 hidden layer neural network. The hidden layers each had 50 units and were fully connected with a ReLU activation. The possible actions are moving in any of the four cardinal directions, or staying in place. The input is a vector of length 120, to represent each of the 20 positions on the board having one of the following states:

- Player 0 is on this square and does not have the ball.
- Player 0 is on this square and is holding the ball.
- Player 1 is on this square and does not have the ball.
- Player 1 is on this square and is holding the ball.
- The ball is on the ground on this square.
- Unoccupied space.

Table 1. Soccer environment hyperparameters against a mixed strategy.

Hyperparameter	Value
Optimizer	Adam
Learning Rate	0.0003
Buffer Size	3000
Gamma	0.99
Timesteps	300000
Batch Size	64
Exploration Fraction	0.33
Exploration Final ϵ	0.01
Training Frequency	1
Training Starts	300

Table 2. Soccer environment considered hyperparameters.

Hyperparameter	Value
Batch Size	32, 64
Buffer Size	300, 1000, 3000, 10000
Learning Rate	1e-3, 3e-3, 1e-4, 3e-4
Timesteps	10000, 30000, 100000, 300000
Exploration Fraction	0.1, 0.3, 0.4, 0.7
Training Starts	100, 300, 1000

B.1.1. Opponent Policy Classifier

The hyperparameters selected for training the opponent classifier are listed in Table 3. The replay buffers gathered

from training best-responses against each opponent were merged into one dataset. The classifier was trained to predict the opponent for each observation in the dataset. This resulted in 15000 data points, which were randomly split 90-10 between training and validation.

Table 3. Markov-Soccer opponent policy classifier hyperparameters.

Hyperparameter	Value
Optimizer	Adam
Learning Rate	$5 \cdot 10^{-5}$
Loss	Cross Entropy
Batch Size	64

The classifier was a neural network with the same architecture as a single policy; however, the last layer is modified to choose opponents rather than actions. We did not perform a hyperparameter search on this network or learning algorithm.

B.2. Gathering

The Gathering environment is a gridworld tragedy-of-the-commons game. For motivation and background on the environment please refer to Perolat et al. (2017); Leibo et al. (2017). The observation space in the Gathering environment is the rectangular area in front of the agent stretching 20 cells forward with a width of 10. The agents simultaneous take actions of either moving in the four cardinal directions, rotating left or right, tagging the other agent with a time-out beam, or taking no action. A visual depiction of the environment is provided in Figure 8.

The hyperparameters that were searched over are provided in Table 5 and the final hyperparameters are in Table 4.

Table 4. Gathering environment hyperparameters.

Hyperparameter	Value
Optimizer	Adam
Learning Rate	$3 \cdot 10^{-4}$
Gradient Norm Clip	None
Buffer Size	30000
Gamma	0.99
Batch Size	64
Exploration Fraction	0.3
Exploration Final ϵ	0.03
Training Starts	1000

B.3. Policy Distillation

In the policy distillation framework, a larger neural network referred to as the "teacher" is used as a training signal for a smaller neural network called the "student". In our experiment the Q-Mixing policy is the teacher to a student neural network that is the size of a single BR policy. The student is trained via supervised learning, reusing the pure-strategy BRs' replay buffers as a dataset. A batch of data is sampled from the replay-buffer and the student predicts Q^S the teacher's response Q^T. The student is trained to imitate the softmax policy of the teacher. The full policy distillation loss is

$$\mathcal{L}_{\text{Distill}} = \sum_{i}^{|D|} \text{softmax}(\frac{Q^T}{\tau}) \ln \frac{\text{softmax}(\frac{Q^T}{\tau})}{\text{softmax}(\frac{Q^S}{\tau})},$$

where D is the dataset of concatenated replay buffers.

The hyperparameters used in policy distillation are listed in Table 6. The student policy is the same neural network that's used in computing the best-responses to individual policies; it is described in Section **??**. We did not perform a hyperparameter search on this network or learning algorithm.

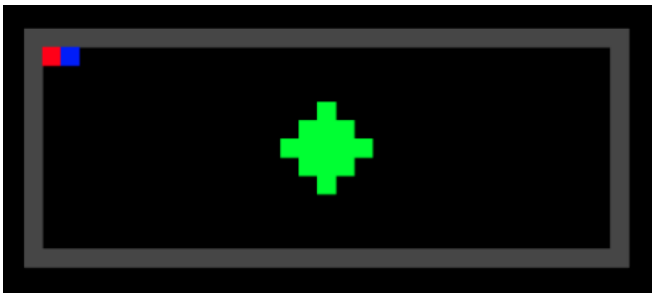

Figure 8. Gathering game map. Players spawn in either the blue or red tile, and apples on the green tiles.

Table 5. Gathering environment considered hyperparameters.

Hyperparameter	Value
Learning Rate	3e-3, 1e-4, 3e-4, 1e-5, 3e-5
Gradient Norm Clip	None, 0.1, 1, 10
Buffer Size	3e4, 5e4, 8e5, 1e5
Batch Size	32, 64
Timesteps	3e5, 5e5, 7e5, 1e6, 1.2e6, 1.5e6, 1.7e6, 2e6
Exploration Timesteps	1e5, 2e5, 3e5, 4e5, 5e5, 7e5

Table 6. Policy distillation hyperparameters.

Hyperparameter	Value
Optimizer	Adam
Learning Rate	0.003
Batch Size	64