Learning not to Regret

David Sychrovský[1,2], **Michal Šustr**[2,5], **Elnaz Davoodi**[3], **Michael Bowling**[4], **Marc Lanctot**[3], **Martin Schmid**[1,5]

[1]Department of Applied Mathematics, Charles University
[2]Artificial Intelligence Center, Czech Technical University
[3]Google DeepMind
[4]Department of Computing Science, University of Alberta
[5]EquiLibre Technologies

sychrovsky@kam.mff.cuni.cz, michal.sustr@aic.fel.cvut.cz, schmid@equilibretechnologies.com

Abstract

The literature on game-theoretic equilibrium finding predominantly focuses on single games or their repeated play. Nevertheless, numerous real-world scenarios feature playing a game sampled from a distribution of similar, but not identical games, such as playing poker with different public cards or trading correlated assets on the stock market. As these similar games feature similar equilibra, we investigate a way to accelerate equilibrium finding on such a distribution. We present a novel "learning not to regret" framework, enabling us to meta-learn a regret minimizer tailored to a specific distribution. Our key contribution, Neural Predictive Regret Matching, is uniquely meta-learned to converge rapidly for the chosen distribution of games, while having regret minimization guarantees on any game. We validated our algorithms' faster convergence on a distribution of river poker games. Our experiments show that the meta-learned algorithms outpace their non-meta-learned counterparts, achieving more than tenfold improvements.

1 Introduction

Regret minimization, a fundamental concept in online convex optimization and game theory, plays an important role in decision-making algorithms (Nisan et al. 2007). In games, a common regret minimization framework is to cast each player as an independent online learner. This learner interacts repeatedly with the game, which is represented by a black-box environment and encompasses the strategies of all other players or the game's inherent randomness. When all the learners employ a regret minimizer, their average strategy converges to a coarse correlated equilibrium (Hannan 1957; Hart and Mas-Colell 2000). Furthermore, in two-player zero-sum games, the average strategy converges to a Nash equilibrium (Nisan et al. 2007). Regret minimization has become the key building block of many algorithms for finding Nash equilibria in imperfect-information games (Bowling et al. 2015; Moravcik et al. 2017; Brown and Sandholm 2018; Brown et al. 2020; Brown and Sandholm 2019b; Schmid et al. 2021).

While these algorithms made progress in single game playing, in many real-world scenarios, players engage in more than just one isolated game. For instance, they might play poker with various public cards, solve dynamical routing problems, or trade correlated assets on the stock market. These games, while similar, are not identical and can be thought of as being drawn from a distribution. Despite its relevance, this setting has been largely unexplored, with a few recent exceptions such as (Harris et al. 2022; Zhang et al. 2022).

In this work, we shift focus to this distributional setting. The black-box environment which the learners interact with corresponds to a game *sampled from a distribution*. This perspective aligns with the traditional regret minimization framework, but with an added twist: the game itself is sampled. Our goal is to reduce the *expected* number of interactions needed to closely approximate an equilibrium of the sampled game. This is crucial both for online gameplay and offline equilibrium learning, as fewer steps directly translate to a faster algorithm.

In either the single-game or distributional settings, the worst-case convergence of regret minimizers against a strict adversary cannot occur at a rate faster than $O(T^{-1/2})$ (Nisan et al. 2007). However in practice, algorithms often converge much faster than the worst-case bound suggests. Consider CFR$^+$ (Tammelin 2014), which empirically converges at the rate of $O(T^{-1})$ in poker games[1] despite having the same $O(T^{-1/2})$ worst case guarantees (Burch 2018). Another example of variations in practical performance is discounted CFR with three parameters ($DCFR_{\alpha,\beta,\gamma}$), where the authors reported that they "found the optimal choice of α, β and γ varied depending on the specific game" (Brown and Sandholm 2019a).

These empirical observations are in line with no-free lunch theorems for optimization, which state that no learning algorithm can dominate across all domains (Wolpert and Macready 1997). Thus to improve performance on a domain, it is necessary to use a specialized algorithm, at the expense of deteriorating the performance outside of this domain.

[1]The strong empirical performance of the algorithm was one of the key reasons behind essentially solving Limit Texas Holdem poker, one of the largest imperfect information games to be solved to this day (Bowling et al. 2015). CFR$^+$ required only $1,579$ iterations to produce the final strategy, far less than what the worst-case bound suggests.

A popular approach to find such algorithms is the meta-learning paradigm, namely a variant of "learning to learn" (Andrychowicz et al. 2016). In the meta-learning framework, one learns the optimization algorithm itself. The simplest approach is to directly parametrize the algorithm with a neural network, and train it to minimize regret on the distribution of interest. While the meta-learned network can quickly converge in the domain it has been trained on (e.g. poker games), it can be at the cost of performance (or even lack of convergence) out-of-distribution. This is because the neural network is not necessarily a regret minimizer.

To provide the convergence guarantees, we introduce meta-learning within the predictive regret framework (Farina, Kroer, and Sandholm 2021). Predictive regret minimization has convergence guarantees regardless of the prediction, while a better prediction guarantees lower regret, and a perfect prediction results in zero regret (Farina, Kroer, and Sandholm 2021). This results in an algorithm that combines the best of both worlds – fast convergence in the domain in question while providing general convergence guarantees.

A particularly interesting application of our approach is when the resulting regret minimizer is used in an online search algorithm (Moravcik et al. 2017; Brown and Sandholm 2018; Schmid et al. 2021). When the agent is deployed to face an opponent in chess, poker or other games, it has a limited time to make a decision. The agent needs to minimize regret within its search tree as quickly as possible — that is, with as few iterations as possible. This is because a single iteration evaluates the leaf nodes of a search tree using a value function, which is typically represented by a slow-to-compute neural network. In this context, the critical measure is the speed during the actual deployment time and online search, that is, when facing the opponent. The offline computation is typically used to learn high quality value functions to be used within search and can take even long time. With our method, one can now also use the offline computation to meta-learn the regret minimizer itself, resulting in substantially faster convergence during the play time.

In experiments, we first evaluate our algorithms on a distribution of matrix games to understand what the algorithms learn. Next, we turn our attention to search with value functions in a sequential decision setting. We show that for a distribution over river poker games, our meta-learned algorithms outpace their non-meta-learned counterparts, achieving more than tenfold improvements.

2 Prior Work

Regret minimization is a powerful framework for online convex optimization (Zinkevich 2003), with regret matching as one of the most popular algorithms in game applications (Hart and Mas-Colell 2000). Counterfactual regret minimization allows to use that framework in sequential decision making, by decomposing the full regret to individual states (Zinkevich et al. 2008). A recently introduced extension of regret matching, the predictive regret matching (Farina, Kroer, and Sandholm 2021) was shown to significantly outperform prior regret minimization algorithms in self-play across a large selection of games. The authors also provided a close connection between the prediction and the regret, which offers additional insight into the algorithm and is a clear inspiration for our work.

Meta-learning has a long history when used for optimization (Schmidhuber 1992, 1993; Thrun and Thrun 1996; Andrychowicz et al. 2016). This work rather considers meta-learning in the context of regret minimization. Many prior works explored modifications of regret matching to speed-up its empirical performance in games, such as CFR+ (Tammelin 2014), DCFR (Brown and Sandholm 2019a), Lazy-CFR (Zhou et al. 2018), ECFR (Li et al. 2020) or Linear CFR (Brown et al. 2019). However, as the no-free lunch theorems for optimization state, no (learning) algorithm can dominate across all domains (Wolpert and Macready 1997). Therefore, to improve performance on a specific domain, it is necessary to use a specialized algorithm, at the expense of deteriorating the performance outside of this domain.

We thus turn to meta-learning the regret minimizers. It was shown that similar games have similar equilibria, justifying the use of meta-learning in games to accelerate equilibrium finding (Harris et al. 2022). A key difference between our and prior works is that they primarily consider settings where the game utilities come from a distribution, rather than sampling the games themselves. Thus, one of their requirements is that the strategy space itself must be the same. In (Azizi et al. 2022), they consider bandits in Bayesian settings. In (Harris et al. 2022), the authors "warm start" the initial strategies from a previous game, making the convergence provably faster. This approach is "path-dependant", in that it depends on which games were sampled in the past. Both works are fundamentally different from ours, as they use meta-learning online, while we are making meta-learning preparations offline.

To our best knowledge, the most similar to our offline meta-learning setting is AutoCFR (Xu et al. 2022). They are not restricted to the same strategy spaces in games like previous works, as they use evolutionary search for an algorithm that is local to each decision state. They search over a combinatorial space, defined by an algebra which generalizes CFR family of algorithms, to find an algorithm that performs well across many games. Our approach rather learns a neural network via gradient descent to perform the regret minimization, allowing us to learn any function representable by the network architecture. Furthermore, unlike AutoCFR, we provide strong regret minimization guarantees.

We also give a quick overview of the recent work on search with value functions, as we use regret minimization in this context in our experiments. The combination of decision-time search and value functions has been used in the remarkable milestones where computers bested their human counterparts in challenging games — DeepBlue for Chess (Campbell, Hoane Jr, and Hsu 2002) and AlphaGo for Go (Silver et al. 2016). This powerful framework of search with (learned) value functions has been extended to imperfect information games (Schmid 2021), where regret minimization is used within the search tree. Regret minimization has quickly become the underlying equilibrium approximation method for search (Moravcik et al. 2017; Brown and Sandholm 2018; Zarick et al. 2020; Serrino et al. 2019; Brown et al. 2020; Schmid et al. 2021).

3 Background

We begin by describing the regret minimisation framework (Nisan et al. 2007). An **online algorithm** m for the regret minimization task repeatedly interacts with an unknown **environment** g through available actions A, receiving a vector of per-action rewards $\boldsymbol{x}$. The goal of regret minimization algorithm is then to maximize its hindsight performance (i.e. to minimize regret).

Formally, at each step $t \leq T$, the algorithm submits a **strategy** $\boldsymbol{\sigma}^t$ from a probability simplex $\Delta^{|A|}$ and observes the subsequent **reward** $\boldsymbol{x}^t \in \mathbb{R}^{|A|}$ returned from the environment g. The rewards are computed with an unknown function concave in $\boldsymbol{\sigma}$ and are bounded. We denote by Δ_{max} the difference between the highest and lowest reward the environment can produce. The difference in reward obtained under $\boldsymbol{\sigma}^t$ and any fixed action strategy is measured by the instantaneous **regret** $\boldsymbol{r}(\boldsymbol{\sigma}^t, \boldsymbol{x}^t) = \boldsymbol{x}^t - \langle \boldsymbol{\sigma}^t, \boldsymbol{x}^t \rangle \mathbf{1}$. A sequence of strategies and rewards, submitted by algorithm m and returned by environment g, up to a horizon T, is

$$\boldsymbol{x}^0 \to \boldsymbol{\sigma}^1 \to \boldsymbol{x}^1 \to \boldsymbol{\sigma}^2 \to \cdots \to \boldsymbol{x}^{T-1} \to \boldsymbol{\sigma}^T \to \boldsymbol{x}^T, \quad (1)$$

where we set $\boldsymbol{x}^0 = \mathbf{0}$ for notational convenience (see also Figure 1). The **cumulative regret** over the entire sequence is

$$\boldsymbol{R}^T = \sum_{t=1}^{T} \boldsymbol{r}(\boldsymbol{\sigma}^t, \boldsymbol{x}^t).$$

The algorithm m is a regret minimizer, if the **external regret** $R^{\text{ext},T} = \left\| \boldsymbol{R}^T \right\|_\infty$ grows sublinearly in T for an arbitrary sequence of rewards $\{\boldsymbol{x}^t\}_{t=1}^T$. Then the average strategy $\overline{\boldsymbol{\sigma}}^t = \frac{1}{t} \sum_{\tau=1}^{t} \boldsymbol{\sigma}^\tau$ converges to a coarse correlated equilibrium (Nisan et al. 2007).

Finally, we define **exploitability** of a strategy $\boldsymbol{\sigma}$ (i.e. the gap from a Nash equilibrium) as

$$\textit{expl}(\boldsymbol{\sigma}) = \max_{\boldsymbol{\sigma}^*} \min_{\boldsymbol{x}} \langle \boldsymbol{\sigma}^*, \boldsymbol{x}(\boldsymbol{\sigma}^*) \rangle - \min_{\boldsymbol{x}} \langle \boldsymbol{\sigma}, \boldsymbol{x}(\boldsymbol{\sigma}) \rangle,$$

where $\boldsymbol{x}(\boldsymbol{\sigma})$ is the reward vector admissible by the environment as a response to strategy $\boldsymbol{\sigma}$. Note that this exactly corresponds to the standard definition of exploitability of a player's strategy in a two-player zero-sum game when playing with an environment controlled by an adversary.

4 Learning not to Regret

We first describe the meta-learning framework for regret minimization. Then we introduce two variants of meta-learned algorithms, with and without regret minimization guarantees.

4.1 Meta-Learning Framework

On a distribution of regret minimization tasks G, we aim to find an online algorithm m_θ with some parameterization θ that efficiently minimizes the expected external regret after T steps. The expected external regret of m_θ is

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{g \sim G} \left[R^{\text{ext},T} \right] = \mathop{\mathbb{E}}_{g \sim G} \left[\max_{a \in A} \sum_{t=1}^{T} \boldsymbol{r}_a \left(\boldsymbol{\sigma}_\theta^t, \boldsymbol{x}^t \right) \right], \quad (2)$$

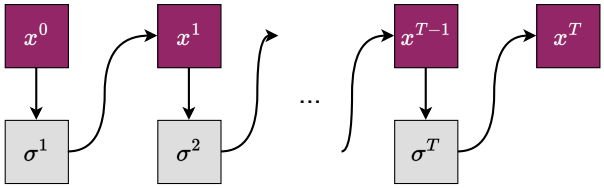


Figure 1: The sequence of strategies $\{\boldsymbol{\sigma}^t\}_{t=1}^T$ submitted by an online algorithm and the rewards $\{\boldsymbol{x}^t\}_{t=1}^T$ received from the environment. The reward $\boldsymbol{x}^0 = \mathbf{0}$ initializes the algorithms to produce the first strategy $\boldsymbol{\sigma}^1$.

where $\boldsymbol{\sigma}_\theta^t$ is the strategy selected at step t by the online algorithm m_θ. We train a recurrent neural network parameterized by θ to minimize (2). By utilizing a recurrent architecture we can also represent algorithms that are history and/or time dependent. This dependence is captured by a hidden state $\boldsymbol{h}$ of the recurrent network. See Section 5 for details.

The choice to minimize external regret in particular is arbitrary. This is because the rewards $\{\boldsymbol{x}^\tau\}_{\tau=1}^T$ that come from the environment are constant[2] w.r.t. θ and the derivative of any element of the cumulative regret vector $\boldsymbol{R}^T$ is thus the same, meaning[3]

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}_\theta^t} = \frac{\partial}{\partial \boldsymbol{\sigma}_\theta^t} \mathop{\mathbb{E}}_{g \sim G} \left[\sum_{\tau=1}^{T} -\langle \boldsymbol{\sigma}_\theta^\tau, \boldsymbol{x}^\tau \rangle \right] = - \mathop{\mathbb{E}}_{g \sim G} \left[\boldsymbol{x}^t \right].$$

Consequently, if the objective (2) is reformulated using other kinds of regrets, it would result in the same meta-learning algorithm. This is because regrets measure the difference between reward accumulated by some fixed strategy, and by the algorithm m_θ. Since the former is a constant at meta-train time, minimizing (2) is equivalent to maximizing the reward $\langle \boldsymbol{\sigma}^t, \boldsymbol{x}^t \rangle$ the algorithm m_θ gets at every $t \leq T$ in a task $g \sim G$, given the previously observed rewards $\{\boldsymbol{x}^\tau\}_{\tau=1}^{t-1}$.

Next, we will show two variants of the algorithm m_θ.

4.2 Neural Online Algorithm

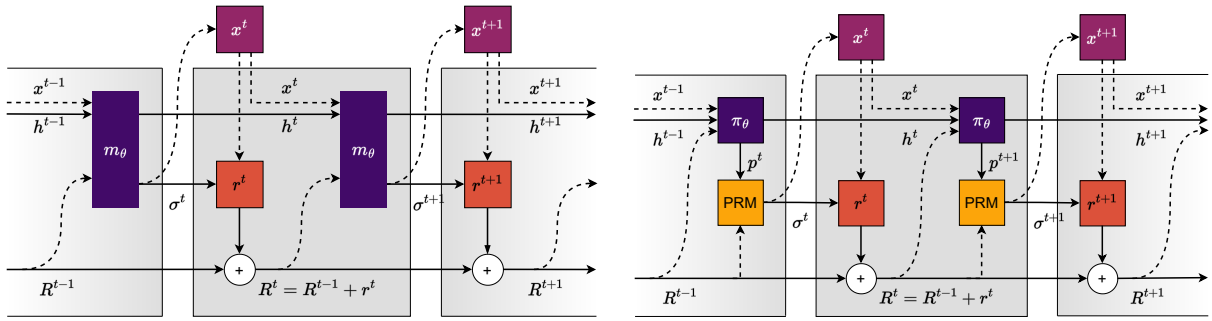
The simplest option is to parameterize the online algorithm m_θ to directly output the strategy $\boldsymbol{\sigma}_\theta^t$. We refer to this setup as neural online algorithm (NOA).

At the step t, the algorithm m_θ receives as input[4] the rewards $\boldsymbol{x}^t$ and cumulative regret $\boldsymbol{R}^t$ and keeps track of its hidden state $\boldsymbol{h}^t$. We estimate the gradient $\partial \mathcal{L} / \partial \theta$ by sampling a batch of tasks and applying backpropagation through the computation graph as shown in Figure 2a. The gradient originates in the final external regret $R^{\text{ext},T}$ and propagates through collection of regrets $\boldsymbol{r}^{1 \dots T}$, the strategies $\boldsymbol{\sigma}^{1 \dots T}$ and hidden states $\boldsymbol{h}^{0 \dots T-1}$. We don't allow the gradient to propagate through the rewards[2] $\boldsymbol{x}^{0 \dots T-1}$ or the cumulative regrets $\boldsymbol{R}^{1 \dots T}$ entering the network. Thus, the only way to influence

[2]This is because the environment is a black-box, i.e. how the reward depends on the chosen strategy is unknown to m_θ.

[3]When recurrent architecture is used, the strategy also depend on strategies used in previous steps, intruding extra terms.

[4]We also input additional contextual information, see Section 5.



(a) Neural online algorithm (NOA).

(b) Neural predictive regret matching (NPRM).

Figure 2: Computational graphs of the proposed algorithms. The gradient flows only along the solid edges. The $\boldsymbol{h}$ denotes the hidden state of the neural network. See also Figure 1 for visual correspondence of the strategy and reward sequence.

the earlier optimization steps is through the hidden states $\boldsymbol{h}^{0\dots T-1}$ of the neural network.[5]

In our experiments, we observe strong empirical performance of NOA. However, NOA is not guaranteed to minimize regret. This is because, similar to policy gradient methods, it is simply maximizing the cumulative reward $\mathbb{E}_{g\sim G}\left[\sum_{t=1}^{T} \boldsymbol{x}^t\right]$, which is not a sufficient condition to be a regret minimizing algorithm (Blackwell et al. 1956).

4.3 Neural Predictive Regret Matching

In order to get convergence guarantees, we turn to the recently introduced predictive regret matching (PRM) (Farina, Kroer, and Sandholm 2021), see also Algorithm 1. The PRM is an extension of regret matching (RM) (Hart and Mas-Colell 2000) which uses an additional predictor $\pi : (\bullet) \to \mathbb{R}^{|A|}$. The algorithm has two functions, NEXTSTRATEGY and OBSERVEREWARD, which alternate over the sequence (1). The predictor makes a prediction $\boldsymbol{p}^{t+1}$ of the next anticipated regret[6] $\boldsymbol{r}^{t+1}$. The PRM algorithm incorporates $\boldsymbol{p}^{t+1}$ to compute the next strategy[7] $\boldsymbol{\sigma}^{t+1}$. The RM algorithm can be instantiated as PRM with $\pi = \mathbf{0}$. Unless stated otherwise, we use PRM with a simple predictor $\pi : (\boldsymbol{\sigma}^t, \boldsymbol{x}^t) \to \boldsymbol{p}^{t+1} = \boldsymbol{r}(\boldsymbol{\sigma}^t, \boldsymbol{x}^t)$, i.e. it predicts the next observed rewards will be the same as the current ones.[8]

We introduce neural predictive regret matching (NPRM), a variant of PRM which uses a predictor π_θ parameterized by a recurrent neural network θ. The predictor π_θ receives as input[4] the rewards $\boldsymbol{x}^t$, cumulative regret $\boldsymbol{R}^t$ and hidden state $\boldsymbol{h}^t$. We train π_θ to minimize Eq. (2), just like NOA. The computational graph is shown in Figure 2b. The output of the network $\boldsymbol{p}^{t+1}$ is used in NEXTSTRATEGY to obtain the strategy $\boldsymbol{\sigma}^{t+1}$. Similar to NOA, the gradient $\partial\mathcal{L}/\partial\theta$ originates in the final external regret $R^{\text{ext},T}$ and propagates through the collection of regrets $\boldsymbol{r}^{1\dots T}$, the strategies $\boldsymbol{\sigma}^{1\dots T}$, the predictions $\boldsymbol{p}^{1\dots T}$, and hidden states $\boldsymbol{h}^{0\dots T-1}$. Again, we do not propagate the gradient through the rewards[2] $\boldsymbol{x}^{0\dots T-1}$ or through the cumulative regrets $\boldsymbol{R}^{1\dots T}$ entering the network.[5] Any time-dependence comes only through the hidden states $\boldsymbol{h}^{0\dots T-1}$. It is interesting to note NPRM can learn to recover both RM and PRM as it receives all the information needed, i.e. $\boldsymbol{x}$ and $\boldsymbol{R}$.

Algorithm 1: Predictive regret matching (Farina, Kroer, and Sandholm 2021)

$\boldsymbol{R}^0 \leftarrow \mathbf{0} \in \mathbb{R}^{|A|}$, $\boldsymbol{x}^0 \leftarrow \mathbf{0} \in \mathbb{R}^{|A|}$

function NEXTSTRATEGY()
  $\boldsymbol{\xi}^t \leftarrow [\boldsymbol{R}^{t-1} + \boldsymbol{p}^t]^+$
  if $\|\boldsymbol{\xi}^t\|_1 > 0$ **return** $\boldsymbol{\sigma}^t \leftarrow \boldsymbol{\xi}^t \,/\, \|\boldsymbol{\xi}^t\|_1$
  else **return** $\boldsymbol{\sigma}^t \leftarrow$ arbitrary point in $\Delta^{|A|}$
function OBSERVEREWARD($\boldsymbol{x}^t$)
  $\boldsymbol{R}^t \leftarrow \boldsymbol{R}^{t-1} + \boldsymbol{r}(\boldsymbol{\sigma}^t, \boldsymbol{x}^t)$
  $\boldsymbol{p}^{t+1} \leftarrow \pi(\boldsymbol{x}^t)$

Importantly, we show that the cumulative regret of NPRM grows sub-linearly, making it a regret minimizer.

Theorem 1 (Correctness of Neural-Predicting)**.** *Let $\alpha \geq 0$, and π_θ be a regret predictor with outputs bounded in $[-\alpha, \alpha]^{|A|}$. Then PRM which uses π_θ is a regret minimizer.*

Proof. Since the reward $\boldsymbol{x}$ for any action is bounded by the maximum utility difference $\Delta_{\max}$, the regret $\boldsymbol{r}$ for any action is bounded by $2\Delta_{\max}$. Thus, for an arbitrary prediction $\boldsymbol{p}$ it holds

$$\|\boldsymbol{r}(\boldsymbol{\sigma}, \boldsymbol{x}) - \boldsymbol{p}\|_2 \leq (2\Delta_{\max} + \alpha)|A|.$$

Using the PRM regret bound (Farina, Kroer, and Sandholm 2021, Thm 3), we obtain

$$\begin{aligned} R^{\text{ext},T} &\leq \sqrt{2}\left(\sum_{t=1}^{T} \left\|\boldsymbol{r}(\boldsymbol{\sigma}^t, \boldsymbol{x}^t) - \boldsymbol{p}^t\right\|_2\right)^{\frac{1}{2}} \\ &\leq \sqrt{2}\left((2\Delta_{\max} + \alpha)|A|T\right)^{\frac{1}{2}} \in O\left(\sqrt{T}\right). \end{aligned}$$

□

[5] This is similar to the "learning to learn" setup (Andrychowicz et al. 2016)

[6] Originally, the predictive regret was formulated in terms of next reward. However, for our application predicting next regret proved more stable as the network outputs don't mix.

[7] Note the prediction can change the actual observed $\boldsymbol{x}^{t+1}$, unless we are at a fixed point.

[8] This predictor was used in the original work.

As NPRM is regret minimizing regardless of the prediction $\boldsymbol{p}$, our network outputs $\boldsymbol{p}$ rather than strategy $\boldsymbol{\sigma}$ as for NOA. This allows us to achieve the best of both words – adaptive learning algorithm with a small cumulative regret in our domain, while keeping the $O(T^{-1/2})$ worst case average regret guarantees. Note that $O(T^{-1/2})$ is the best achievable bound in terms of T against a black-box (Nisan et al. 2007).

5 Experiments

We focus on application of regret minimization in games, see Appendix A for their detailed description. Specifically, we apply regret minimization to one-step lookahead search with (approximate) mini-max subgame value functions. See also Section 1 for motivation of this approach.

For both NOA and NPRM, the neural network architecture we use is a two layer LSTM. For NOA, these two layers are followed by a fully-connected layer with the softmax activation. For NPRM, we additionally scale all outputs by $\alpha \geq 2\Delta_{\max}$, ensuring any regret vector can be represented by the network. In addition to the last observed reward and the cumulative regret, the networks also receive contextual information corresponding to the player's observations.

We minimize objective (2) for $T = 64$ iterations over 512 epochs using the Adam optimizer.[9] Other hyperparameters[10] were found via a grid search. For evaluation, we compute exploitability of the strategies up to $2T = 128$ iterations to see whether the algorithms can generalize outside of the horizon T they were trained on and whether they keep reducing the exploitability. We train and evaluate both NOA and NPRM and compare our methods against (P)RM. Our results are presented in Figure 3.

The section is structured as follows. First, we illustrate how our algorithms behave using a simple distribution of matrix games. Next, we show how their performance extends to the sequential setting, where we evaluate on river poker. To illustrate viability of our approach, we study the computational time reduction achieved by our algorithms. Next, we demonstrate our algorithms are tailored to the training domain, and thus their performance can deteriorate out-of-distribution. Finally, we discuss several possible modifications of our approach.

5.1 Matrix Games

In the case of matrix games, a value function corresponds to playing against a best responding opponent.[11] We use a modification of the standard `rock_paper_scissors` game and perturb two elements of the utility matrix to generate a distribution G, see Appendix A.1.

Our results are presented in Figure 3. First, we consider the distribution to have probability 1 for a single game, i.e. the game is fixed. In this setting, our algorithms can simply overfit and output a strategy close to a Nash equilibrium.

[9] We use cosine learning rate decay from 10^{-3} to $3 \cdot 10^{-4}$.

[10] Specifically, the size of the LSTM layer, the number of games in each batch gradient update, and the regret prediction bound α.

[11] A simultaneous-move matrix game can be formulated as a strategy-equivalent two step sequential game. The value function assumes optimal play by the opponent, i.e. a best response.

Target	$4 \cdot 10^{-1}$	10^{-1}	$6 \cdot 10^{-2}$	$2 \cdot 10^{-2}$
RM	20	128	212	615
PRM	36	158	261	793
NOA	**1**	18	41	157
NPRM	**1**	**16**	**26**	**118**

Table 1: Number of steps each algorithm requires to reach target exploitability on `river_poker` (sampled) in expectation.

Their convergence is very fast compared to (P)RM. Notice that NOA outperforms NPRM in this setting. We hypothesize there are two main reasons for this difference. First, NPRM is more restricted in its functional dependence. Second, the gradient of NPRM vanishes, resp. explodes when the cumulative regret is large, resp. small, making overfitting more challenging.

Next, we sample games in the perturbed setting. Our methods keep outperforming (P)RM – even after the horizon T on which they were trained. To further illustrate the differences between the meta-learned algorithms and (P)RM, we plot the current and average strategies selected by each algorithm in Figure 4. Both NOA and NPRM are initially close to the equilibrium and converge relatively smoothly. In contrast, (P)RM visit large portion of the policy space even in later steps, making the convergence slower.

Notice that RM exhibits similar performance to PRM both in matrix and sequential games. This may seem surprising, as PRM was shown to be stronger than RM in self-play settings (Farina, Kroer, and Sandholm 2021). However, the reason is that we minimize regret against an adversary rather than the self-play opponent. PRM performs well when the last-observed reward is a good prediction of the next one. This is true in self-play, as the opponent does not radically change their strategy between iterations. However, it is no longer the case when the values are coming from a value function where arbitrarily small modification of the input can lead to large changes of the output (Schmid 2021).

5.2 Sequential Games

To evaluate our methods for sequential games, we use the public root state of `river_poker` – a subgame of no-limit Texas Hold'em Poker with approximately 62 thousand states. The distribution G is generated by sampling public cards uniformly, while the player beliefs are sampled in the same way as in (Moravcik et al. 2017). For the value function, we used 1,000 iterations of CFR$^+$. See Appendix A.2 for more details.

Our setup allows NOA and NPRM to learn to minimize regret in a contextualized manner, specific to each decision state. This is achieved by augmenting the input of the network by features corresponding to the player's observation at each state. In this case, the input is the the beliefs for both players and an encoding of private and public cards.

Our results are presented in Figure 3. We show that both NOA and NPRM are able to approximate an equilibrium

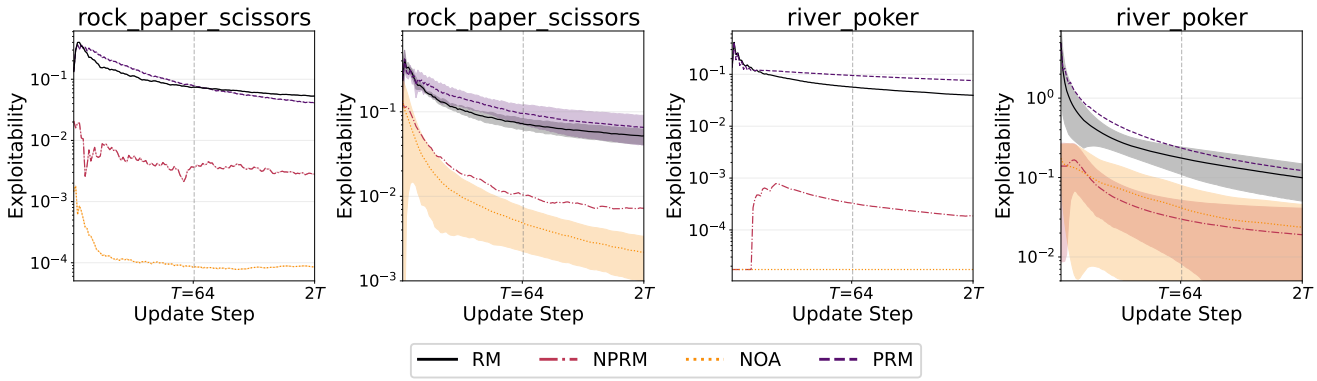


Figure 3: Comparison of non-meta-learned algorithms (RM, PRM) with meta-learned algorithms (NOA, NPRM), on a small matrix game and a large sequential game and for a single fixed game versus a whole distribution over games. The figures show exploitability of the average strategy $\overline{\boldsymbol{\sigma}}^t$. The y-axis uses a logarithmic scale. Vertical dashed lines separate two regimes: training (up to T steps) and generalization (from T to $2T$ steps). Colored areas show standard error for the sampled settings.

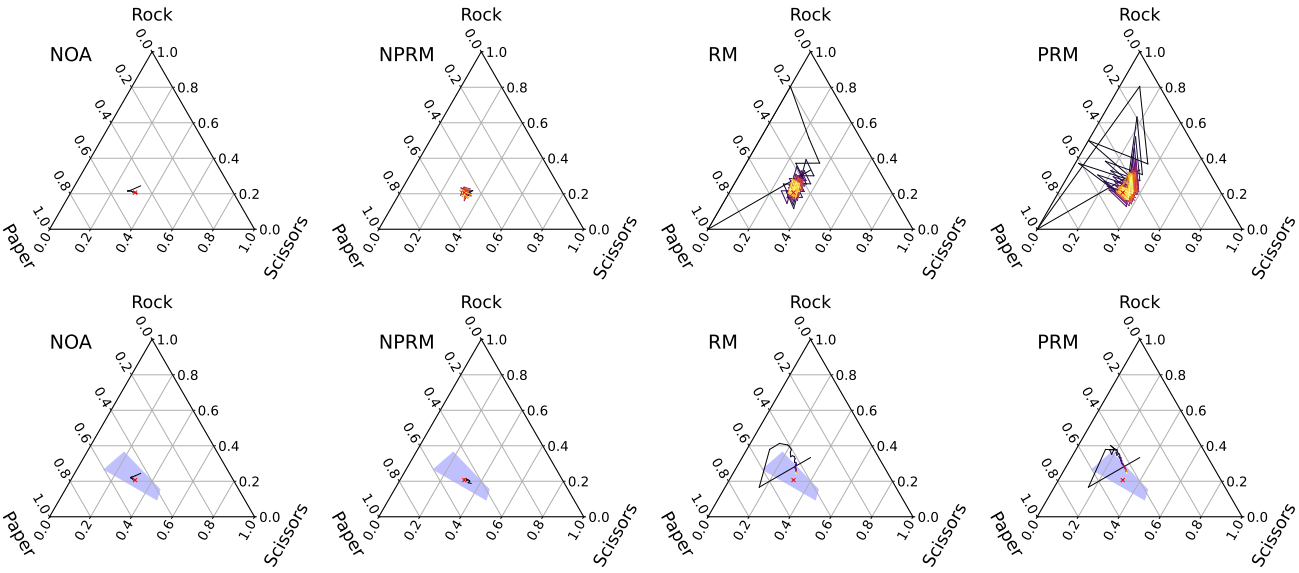


Figure 4: For each algorithm, we show the trajectories of current strategies $\boldsymbol{\sigma}^t$ (top row) and average strategies $\overline{\boldsymbol{\sigma}}^t$ (bottom row) on `rock_paper_scissors` (sampled) for $2T = 128$ steps. The red cross shows the equilibrium of the sampled game. The trajectories start in dark colors and get brighter for later steps. The blue polygon is the set of all equilibria in the distribution `rock_paper_scissors` (sampled), computed according to (Bok and Hladík 2015). Notice how the strategies of our meta-learned algorithms begin in the polygon and refine their strategy to reach the current equilibrium. In contrast, (P)RM are initialized with the uniform strategy and visit a large portion of the policy space.

of a fixed game very closely, often to higher precision than the solver. This manifests seemingly as a lower bound on exploitability for `river_poker` (fixed), see Appendix B for details. Importantly, even in the sampled setting, our algorithms greatly outperform (P)RM, reducing the exploitability roughly ten-times faster. Just like in the matrix setting, PRM shows similar performance to RM, see previous section.

To further evaluate the improvements, we tracked how many steps it takes to reach a solution of specified target quality, see Table 1. Both NOA and NPRM outperform (P)RM for all target exploitabilities, with better solutions requiring an order of magnitude less steps.

5.3 Computational Time Reduction

The reduction of the number of interactions with the environment may come at the expense of increasing the computational time, due to the overhead associated with calling the neural network. This time also depends on other factors, such as the selected domain, the available hardware, or the size of the network. To assess the computational savings, we plot our results as a function of wall time in Figure 5.

On `rock_paper_scissors`, the network overhead is

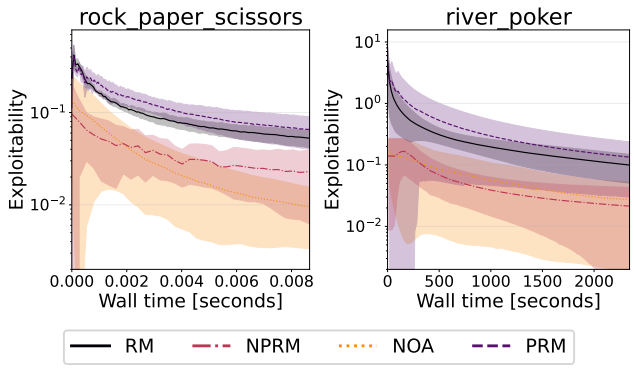


Figure 5: Comparison of regret minimization algorithms as a function of wall time, rather than number of steps shown in Figure 3.

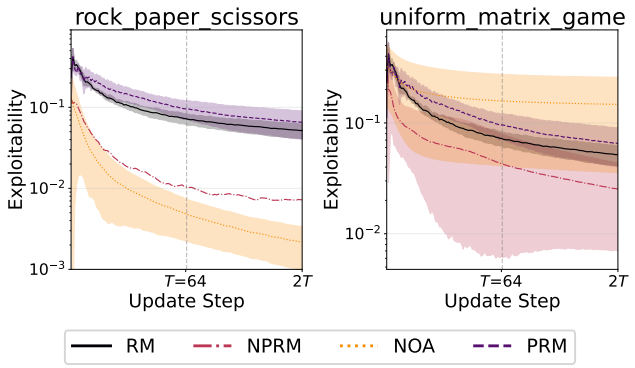


Figure 6: Comparison of the converge guarantees of NOA and NRPM. Both were trained on `rock_paper_scissors` (sampled). Left figure shows NOA and NPRM can out outperform (P)RM on the distribution it was trained on. However, right figure shows that when evaluated on `uniform_matrix_game` (sampled), the performance of NOA deteriorates significantly.

noticeable, making each step of our methods about $4\times$ slower than (P)RM. Despite this, our methods keep outperforming (P)RM even after accounting for this extra cost. The offline meta-training was performed in about ten minutes.

On `river_poker`, interacting with the environment is very expensive. Each interaction requires approximating optimal strategy[12] in the subgame i.e. 1,000 iterations of CFR$^+$. Here, we observed the reduction in the number of steps translates well to the reduction of computational time. For example, exploitability reached by NPRM after one minute would take RM, resp. PRM approximately 26, resp. 34 minutes to reach. The meta-training on `river_poker` took two days.

We ran these experiments using a single CPU thread. As neural networks greatly benefit from using parallel processing, in some sense this can be seen as the worst-case hardware choice. Furthermore, for larger games than the ones considered here, each interaction is typically even more expensive.

5.4 Out of Distribution Convergence

As stated before, NOA is not guaranteed to minimize regret. However, NPRM is a regret minimizer even for games $g' \sim G' \neq G$. Figure 6 shows both methods trained to minimize regret on `rock_paper_scissors` (sampled) and evaluated on `uniform_matrix_game` (sampled). The results show that the performance of NOA deteriorates significantly. This is expected, as it aligns with the no-free-lunch theorems for optimization. However, NPRM is able to keep minimizing regret even outside the domain it was trained on. In this case, it even outperforms (P)RM.

5.5 Additional Experiments

While the previous experiments made use of the common regret-matching-like setup, our meta-learning approach is more general. We investigated two modification based on previous methods. First, instead of aggregating the instantaneous regrets directly, we summed only the positive parts of said regrets, similar to (P)RM$^+$ (Tammelin 2014). Second, we used Hedge (Freund and Schapire 1997) instead of regret matching to produce the strategy $\boldsymbol{\sigma}$. We present results for both of these approaches in Appendix C, and Figure 7. Both meta-learned algorithms keep outperforming corresponding equivalents of (P)RM.

[12]We wrote a costume solver for `river_poker` which outperforms other publicly available solvers. We made the solver available on https://github.com/DavidSych/RivPy.

6 Conclusion

We introduced two new meta-learning algorithms for regret minimization in a new "learning not to regret" framework. Our algorithms are meta-learned to minimize regret fast against a distribution of potentially adversary environments. We evaluated our methods in games, where we minimize regret against an (approximate) value function and measure the exploitability of the resulting strategy. Our experiments show that our meta-learned algorithms attain low exploitability approximately an order of magnitude faster than prior regret minimization algorithms.

In the future, we plan to extend our results to the self-play settings. We also plan to apply our methods with hindsight rationality (Morrill et al. 2021) for games which change over time. This is also an opportunity to combine our offline meta-learning with the online meta-learning of (Harris et al. 2022).

Acknowledgements The authors would like to thank Martin Loebl, Matej Moravčík, Viliam Lisý, and Milan Hladík for their insightful comments. This work was supported by the Czech Science Foundation grant no. GA22-26655S and CoSP Project grant no. 823748. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

References

Andrychowicz, M.; Denil, M.; Gomez, S.; Hoffman, M. W.; Pfau, D.; Schaul, T.; Shillingford, B.; and De Freitas, N. 2016. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29.

Azizi, M.; Kveton, B.; Ghavamzadeh, M.; and Katariya, S. 2022. Meta-Learning for Simple Regret Minimization. *arXiv preprint arXiv:2202.12888*.

Blackwell, D.; et al. 1956. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1): 1–8.

Bok, J.; and Hladík, M. 2015. Selection-based approach to cooperative interval games. In Vitoriano, B.; and Parlier, G. H., eds., *Proceedings of the International Conference on Operations Research and Enterprise Systems*, 34–41. Lisbon, Portugal: SciTePress. ISBN 978-989-758-075-8.

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit hold'em poker is solved. *Science*, 347(6218): 145–149.

Brown, N.; Bakhtin, A.; Lerer, A.; and Gong, Q. 2020. Combining deep reinforcement learning and search for imperfect-information games. *Advances in Neural Information Processing Systems*, 33: 17057–17069.

Brown, N.; Lerer, A.; Gross, S.; and Sandholm, T. 2019. Deep counterfactual regret minimization. In *International conference on machine learning*, 793–802. PMLR.

Brown, N.; and Sandholm, T. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374): 418–424.

Brown, N.; and Sandholm, T. 2019a. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1829–1836.

Brown, N.; and Sandholm, T. 2019b. Superhuman AI for multiplayer poker. *Science*, 365(6456): 885–890.

Burch, N. 2018. Time and space: Why imperfect information games are hard.

Campbell, M.; Hoane Jr, A. J.; and Hsu, F.-h. 2002. Deep blue. *Artificial intelligence*, 134(1-2): 57–83.

Farina, G.; Kroer, C.; and Sandholm, T. 2021. Faster game solving via predictive blackwell approachability: Connecting regret matching and mirror descent. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 6, 5363–5371.

Freund, Y.; and Schapire, R. E. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1): 119–139.

Hannan, J. 1957. Approximation to Bayes risk in repeated play. *Contributions to the Theory of Games*, 3: 97–139.

Harris, K.; Anagnostides, I.; Farina, G.; Khodak, M.; Wu, Z. S.; and Sandholm, T. 2022. Meta-Learning in Games. *arXiv preprint arXiv:2209.14110*.

Hart, S.; and Mas-Colell, A. 2000. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5): 1127–1150.

Li, H.; Wang, X.; Qi, S.; Zhang, J.; Liu, Y.; Wu, Y.; and Jia, F. 2020. Solving imperfect-information games via exponential counterfactual regret minimization. *arXiv preprint arXiv:2008.02679*.

Moravcik, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337): 508–513.

Morrill, D.; D'Orazio, R.; Sarfati, R.; Lanctot, M.; Wright, J. R.; Greenwald, A. R.; and Bowling, M. 2021. Hindsight and sequential rationality of correlated play. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 6, 5584–5594.

Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V. V. 2007. Algorithmic Game Theory. *Google Scholar Google Scholar Digital Library Digital Library*.

Schmid, M. 2021. Search in Imperfect Information Games. *arXiv preprint arXiv:2111.05884*.

Schmid, M.; Moravcik, M.; Burch, N.; Kadlec, R.; Davidson, J.; Waugh, K.; Bard, N.; Timbers, F.; Lanctot, M.; Holland, Z.; et al. 2021. Player of games. *arXiv preprint arXiv:2112.03178*.

Schmidhuber, J. 1992. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1): 131–139.

Schmidhuber, J. 1993. A neural network that embeds its own meta-levels. In *IEEE International Conference on Neural Networks*, 407–412. IEEE.

Serrino, J.; Kleiman-Weiner, M.; Parkes, D. C.; and Tenenbaum, J. 2019. Finding friend and foe in multi-agent games. *Advances in Neural Information Processing Systems*, 32.

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

Tammelin, O. 2014. Solving large imperfect information games using CFR+. *arXiv preprint arXiv:1407.5042*.

Thrun, S.; and Thrun, S. 1996. *Explanation-based neural network learning*. Springer.

Wolpert, D. H.; and Macready, W. G. 1997. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1): 67–82.

Xu, H.; Li, K.; Fu, H.; Fu, Q.; and Xing, J. 2022. AutoCFR: Learning to Design Counterfactual Regret Minimization Algorithms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 5, 5244–5251.

Zarick, R.; Pellegrino, B.; Brown, N.; and Banister, C. 2020. Unlocking the potential of deep counterfactual value networks. *arXiv preprint arXiv:2007.10442*.

Zhang, M.; Zhao, P.; Luo, H.; and Zhou, Z.-H. 2022. No-regret learning in time-varying zero-sum games. In *International Conference on Machine Learning*, 26772–26808. PMLR.

Zhou, Y.; Ren, T.; Li, J.; Yan, D.; and Zhu, J. 2018. Lazy-CFR: fast and near optimal regret minimization for extensive games with imperfect information. *arXiv preprint arXiv:1810.04433*.

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, 928–936.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, 1729–1736.

A Games

A.1 Matrix Games

The `rock_paper_scissors` is a matrix game given by

$$u_1 = -u_2 = \begin{pmatrix} 0 & -1 & 3+X \\ 1 & Y & -1 \\ -1 & 1 & 0 \end{pmatrix},$$

where the parameters X, Y are set to zero for the `rock_paper_scissors` (fixed), and $X, Y \sim \mathcal{U}(-1, 1)$ for `rock_paper_scissors` (sampled). Note that the fixed variant is a biased version of the original game. We opted for this option to make the equilibrium strategy non-uniform, as in the original game (P)RM are initialized with the equilibrium policy.

The `uniform_matrix_game` (sampled) is a 3×3 matrix game with elements generated i.i.d. from $\mathcal{U}(-1, 1)$.

A.2 Sequential Game

For `river_poker`, we use the endgame of no-limit Texas Hold'em Poker with all public cards revealed. The currency used is normalized such that the initial pot of each player is one. The total budget of each player is set to one-hundred times that amount, which implies there are 61,617 information states in total. To create a distribution, we sample the five public cards, and the beliefs for both players in the root of the subgame. The public cards are sampled uniformly, while the beliefs are sampled in the same way as (Moravcik et al. 2017). The algorithms presented in the main text are used only in the public root state, and the optimal strategy in the rest of the game is approximated via 1,000 iterations of self-play CFR$^+$. The corresponding approximate counterfactual values were used as rewards $\boldsymbol{x}^t$. The exploitability of the optimal extension was obtained by approximating the game value (again via CFR$^+$), and subtracting the value in the root given the average strategy of each algorithm. We opted to use CFR$^+$ due to its strong empirical performance on poker games (Bowling et al. 2015). Compared to value functions represented for example by a neural network, this approach offers strong guarantees and replicability.

B Approximate Value Function Error

To approximate the exploitability, one needs to approximate both the value function, and the game value of the `river_poker`. As stated above, we used CFR$^+$ in self-play to approximate both. During evaluation of `river_poker` (fixed), we used $10\times$ more CFR$^+$ iterations than in training, to improve the approximation of both game value and the value function. In this case, when used to approximate the game value, it yields a solution with a two-player Nash gap of approximately $1.6 \cdot 10^{-5}$. The error is of similar magnitude as the exploitability observed in Figure 3 for `river_poker` (fixed) and it explains the apparent lower bound.

During evaluation, we observed rapid changes of performance of NPRM when we changed the number of CFR$^+$ iterations. Since it was trained using 1,000 iterations, the evaluation is effectively out-of-distribution. We hypothesise that the reason why NPRM struggles more than NOA is that PRM is very sensitive to small changes in the prediction when the cumulative regret is small. This makes not only the training challenging, but may also explain the sudden degradation in performance observed on `river_poker` (fixed).

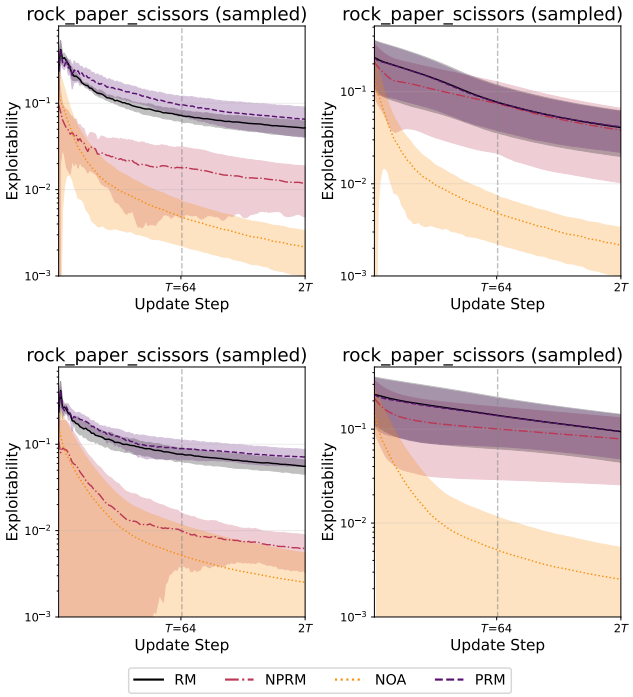


Figure 7: Comparison of regret minimization algorithms against best responding opponent. The figures show exploitability of the average strategy $\overline{\boldsymbol{\sigma}}$. Vertical dashed line separates the training (up to T steps) and generalization (from T to $2T$ steps) regimes. The bottom row aggregates only positive regret, while the right column uses Hedge.

C Alternative Setups

In this section, we show our methods can be combined with two popular methods for regret minimization. First, similar to CFR$^+$ (Tammelin 2014), we aggregate only positive parts of the regret $\boldsymbol{R}^t = [\boldsymbol{R}^{t-1} + \boldsymbol{r}^t]^+$. Second, we use Hedge (Freund and Schapire 1997) to produce the strategy

$$\boldsymbol{\sigma}^t = \frac{e^{\beta(R^t + p^t)}}{\sum_{a \in A} e^{\beta(R^t + p^t)}}, \qquad \beta = \sqrt{\frac{2\log(|A|)}{T}},$$

for NPRM, RM and PRM. We train and evaluate all algorithms on `rock_paper_scissors` (sampled). Our results are presented in Figure 7.

Aggregating only positive regret seems to improve the performance of NPRM, and hinder NOA. Since RM$^+$ was observed to outperform RM on similar games (Tammelin 2014), it may be the case this helps NPRM as well. In contrast, NOA receives less information through $\boldsymbol{R}$. Hedge exhibits slower convergence in general, and severely decreases the performance of NPRM. Interestingly, higher values of α perform better with Hedge. This corresponds to the fact that in order to get a strategy far from uniform, the cumulative

regret needs to be large compared to when regret matching is used.

Note that it is known the temperature β can be tuned, leading to improved performance (Burch 2018). This can also be done within our meta-learning approach, by allowing the network to output β directly.