Regret Transfer and Parameter Optimization

Noam Brown
Robotics Institute
Carnegie Mellon University
noamb@cs.cmu.edu

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
sandholm@cs.cmu.edu

Abstract

Regret matching is a widely-used algorithm for learning how to act. We begin by proving that regrets on actions in one setting (game) can be transferred to warm start the regrets for solving a different setting with same structure but different payoffs that can be written as a function of parameters. We prove how this can be done by carefully discounting the prior regrets. This provides, to our knowledge, the first principled warm-starting method for no-regret learning. It also extends to warm-starting the widely-adopted *counterfactual regret minimization (CFR)* algorithm for large incomplete-information games; we show this experimentally as well.

We then study optimizing a parameter vector for a player in a two-player zero-sum game (e.g., optimizing bet sizes to use in poker). We propose a custom gradient descent algorithm that provably finds a locally optimal parameter vector while leveraging our warm-start theory to significantly save regret-matching iterations at each step. It optimizes the parameter vector while simultaneously finding an equilibrium. We present experiments in no-limit Leduc Hold'em and no-limit Texas Hold'em to optimize bet sizing. This amounts to the first action abstraction algorithm (algorithm for selecting a small number of discrete actions to use from a continuum of actions—a key preprocessing step for solving large games using current equilibrium-finding algorithms) with convergence guarantees for extensive-form games.

Introduction

Consider an agent that has a number of actions available to choose from. *Regret matching* (Hart and Mas-Colell 2000) is a widely-used, general algorithm for learning, over time, how to act. While regret is a broadly applicable concept, for this paper we will view it through the lens of game theory. The agent is a player in a game and his payoffs can depend on his and the other players' actions. This setup subsumes two important special cases: a game against nature and a two-player zero-sum game. To start, we do not restrict the number of players and we do not assume the game is zero sum. Later we will present certain results that are specific to two-player zero-sum games.

A normal-form (aka. bimatrix) game is defined as follows. The game has a finite set N of players, and for each player $i \in N$ a set A_i of available actions. The game also has:

- For each player $i \in N$ a payoff function $u_i : A_i \times A_{-i} \rightarrow \Re$, where A_{-i} is the space of action vectors of the other agents except i. Define $\Delta_i = \max_{\langle a_i, a_{-i}\rangle} u_i(a_i, a_{-i}) - \min_{\langle a_i, a_{-i}\rangle} u_i(a_i, a_{-i})$ and define $\Delta = \max_i \Delta_i$.

- For each player i, a *strategy* σ_i is a probability distribution over his actions. The vector of strategies of players $N \setminus \{i\}$ is denoted by σ_{-i}. We define $u_i(\sigma_i, \sigma_{-i}) = \sum_{a,a_{-i}} p_{\sigma_i}(a) p_{\sigma_{-i}}(a_{-i}) u_i(a, a_{-i})$. We call the vector of strategies of all players a *strategy profile* and denote it by $\sigma = \langle \sigma_i, \sigma_{-i} \rangle$. Moreover, the *value* of σ to player i is defned as $v_i(\sigma) = u_i(\sigma_i, \sigma_{-i})$.

In regret-minimization algorithms, a strategy is determined through an iterative process. While there are a number of such algorithms (e.g., (Greenwald, Li, and Marks 2006; Gordon 2007)), this paper will focus on a typical one called *regret matching* (specifically, the polynomially weighted average forecaster with polynomial degree 2). We will now review how regret matching works, as well as the necessary tools to analyze it.

Let σ_i^t be the strategy used by player i on iteration t. The *instantaneous regret* of player i on iteration t for action a is

$$r_{t,i}(a) = u_i(a, \sigma_{-i}^t) - u_i(\sigma^t, \sigma_{-i}^t) \tag{1}$$

The *cumulative regret* for player i on iteration T for action a is

$$R_{T,i}^c(a) = \sum_{t=1}^{T} r_{t,i}(a) \tag{2}$$

The *average regret*, or simply *regret*, for player i on iteration T for action a is

$$R_{T,i}(a) = \frac{1}{T} \sum_{t=1}^{T} r_{t,i}(a) \tag{3}$$

Also, $R_{T,i}^c = \max_a \{R_{T,i}^c(a)\}$ and $R_{T,i} = \max_a \{R_{T,i}(a)\}$.

In the regret-matching algorithm, a player simply picks an action in proportion to his positive regret on that action, where positive regret is $R_{t,i}(a)_+ = \max\{R_{t,i}(a), 0\}$. Formally, at each iteration $t+1$, player i selects actions $a \in A_i$ according to probabilities

$$p_{t+1}(a) = \begin{cases} \frac{R_{t,i}(a)_+}{\sum_{a' \in A_i} R_{t,i}(a')_+}, & \text{if } \sum_{a' \in A_i} R_{t,i}(a')_+ > 0 \\ \frac{1}{|A|}, & \text{otherwise} \end{cases} \tag{4}$$

Analysis of regret matching uses a *potential function*

$$\Phi(\vec{R}_{T,i}) = \sum_{a \in A_i} R_{T,i}(a)_+^2 T^2 = \sum_{a \in A_i} R_{T,i}^c(a)_+^2 \tag{5}$$

As shown in (Cesa-Bianchi and Lugosi 2006, p. 10), one can bound the potential function in regret-matching by

$$\Phi(\vec{R}_{T,i}) \leq \sum_{t=1}^{T} \sum_{a \in A_i} r_{t,i}(a)^2 \leq \Delta_i^2 |A_i| T \tag{6}$$

From this, one can also bound regret as

$$R_{T,i} \leq R_{T,i+} \leq \frac{\Delta_i \sqrt{|A_i|}}{\sqrt{T}} \tag{7}$$

Thus, as $T \to \infty$, $R_{T,i+} \to 0$.

Contributions and Outline of This Paper

In the next section we prove that regrets on actions in one setting (game) can be transferred to warm start the regrets for solving a different setting with same structure but different payoffs that can be written as a function of parameters. We prove how this can be done by carefully discounting the prior regrets. This provides, to our knowledge, the first principled warm-starting method for no-regret learning. It also extends to warm-starting the widely-adopted *counterfactual regret minimization (CFR)* algorithm (Zinkevich et al. 2007) for large incomplete-information games; we show this experimentally as well. CFR is one of the two leading algorithms for finding near-equilibrium in large imperfect-information games (the other algorithm is based on different principles (Hoda et al. 2010)); CFR is much more commonly used.

In the section after that, we study optimizing a parameter vector for a player in a two-player zero-sum game (e.g., optimizing bet sizes to use in poker). We propose a custom gradient descent algorithm that provably finds a locally optimal parameter vector while leveraging our warm-start theory to significantly save regret-matching iterations at each step. It optimizes the parameter vector while simultaneously finding an equilibrium. We present experiments in no-limit Leduc Hold'em and no-limit Texas Hold'em to optimize bet sizing. This amounts to the first *action abstraction algorithm* (algorithm for selecting a small number of discrete actions to use from a continuum of actions—a key preprocessing step for solving large games using current equilibrium-finding algorithms (Sandholm 2010)) with convergence guarantees for extensive-form games. Prior action abstraction algorithms (bet sizing algorithms) have either had no convergence guarantees (Hawkin, Holte, and Szafron 2011; 2012) or have been defined only for a much narrower game class, stochastic games (Sandholm and Singh 2012).

Regret Transfer: Initializing Regrets of Actions Based on Regrets Computed for Related Settings

We consider a space of games with identical structure but potentially differing rewards. Suppose that we ran T iterations of regret matching on a game Γ_1 and then made a very small change to the rewards so that we are now in a game Γ_2. Intuitively, if the change was small, then most of what we learned from Γ_1 should still apply to Γ_2, though perhaps it will not be quite as relevant as before. Indeed, in this section we build on that intuition and prove that we can transfer most of what we learned, and therefore avoid restarting regret matching from scratch. We will show that this can be done by appropriately discounting the former regrets.

The fundamental idea is that since we make only a small change to the payoffs, if the same exact sequence of strategies that were played in the T iterations of regret matching in game Γ_1 were repeated in Γ_2 (ignoring the regrets when choosing actions in each iteration), then we can still bound the regret in Γ_2. If we then carefully scale those iterations down so that they "weigh less", we can fit them into a bound that is equivalent to having played $T' \leq T$ iterations of regret matching from scratch in Γ_2. We then prove that we can continue regret matching for some number of iterations T_2 in Γ_2, and our resulting bound is equivalent to having played $T' + T_2$ iterations of regret matching from scratch in Γ_2.

Key to this algorithm is the assumption that we can replay the exact sequence of strategies from the T iterations of regret matching from Γ_1 in Γ_2. In general, this is not worthwhile to implement, as it would be better to simply play T iterations of regret matching in Γ_2 from scratch. However, we can "replay" the T iterations in $O(1)$ in the case we mentioned previously, where payoffs are all functions of a parameter vector $\vec{\theta}$, and we change only the value of this vector. In this case, we can store the regret we accumulate in Γ_1 as a function of this vector. For example, consider the case of $\vec{\theta}$ being just a scalar θ. If all payoffs are of the form $u_{i,\langle a_i,a_{-i}\rangle} = \alpha_{i,\langle a_i,a_{-i}\rangle}\theta + \beta_{i,\langle a_i,a_{-i}\rangle}$, then we can store regret for every action with separate values $R_\alpha(a)$ and $R_\beta(a)$. When we access the regret to determine the action for the next iteration, we can simply calculate $R(a) = R_\alpha(a)\theta_1 + R_\beta(a)$, where θ_1 is the value of θ in Γ_1. This way, if we move to Γ_2 where θ is some different value θ_2, then we can simply evaluate $R_\alpha(a)\theta_2 + R_\beta(a)$ to get regret.

We will now proceed to proving that regrets can be transferred across settings by appropriately discounting the iterations conducted at prior settings. First, we will present a result that will be useful in our proof. The bound (6) on the potential function requires us to play according to regret matching at each iteration. But suppose, instead, we played a sequence of arbitrary strategies first and only then start playing according to regret matching. We will now show that we can still bound the potential function.

Lemma 1. *Suppose player i is given an initial potential $\Phi(\vec{R}_{T_0,i}) \leq wT_0|A_i|\Delta_i^2$ for a game where T_0 iterations have been played and w is an arbitrary scalar. If regret matching is used in all future iterations, then at iteration $T_0 + T$, we can bound the potential:*

$$\Phi(\vec{R}_{T_0,i} + \vec{R}_{T,i}) \leq (wT_0 + T)|A_i|\Delta_i^2 \tag{8}$$

All proofs are presented in an extended version of this paper.

We now show specifically how to weigh the previous iterations. Suppose after T iterations we have some regret R_T, and then we modify the game by changing the payoffs slightly. We will analyze how one can initialize regret matching in this new game.

We will find it useful to define *weighted average regret*:

$$R^w_{T,T_2,i}(a) = \frac{w\sum_{t=1}^{T} r_{t,i}(a) + \sum_{t=1}^{T_2} r_{t,i}(a)}{wT + T_2} \tag{9}$$

Theorem 1. *Say we have played T iterations of regret matching in a game. Assume all players play the same sequence of strategies over T iterations in some other game, where the structure of the game is identical but the payoffs may differ. We denote regret in this new game by R'. Consider any weight w_i for agent i such that*

$$0 \leq w_i \leq \frac{\Delta_i^2 |A_i| T}{\Phi(\vec{R}'_{T,i})} \tag{10}$$

If we scale the payoffs of the first T iterations by w_i and then play according to regret matching for T_2 iterations, then weighted average regret in the new game is

$$R'^{w}_{T,T_2,i} \leq \frac{\Delta_i \sqrt{|A_i|}}{\sqrt{(w_i T + T_2)}} \tag{11}$$

This is the same bound achieved from the player playing $w_i T + T_2$ iterations of regret-matching from scratch.

Later in this paper we will find it useful to define w_i in terms of the maximum change in regret. We therefore present the following proposition.

Proposition 1. *Let*

$$\delta_i = \max\left\{ \max_a [R'_{T,i}(a) - R_{T,i}(a)],\ 0 \right\} \tag{12}$$

If we choose

$$0 \leq w_i \leq \frac{1}{1 + \frac{2\delta_i \sqrt{|A_i|}\sqrt{T}}{\Delta_i} + \frac{\delta_i^2 T}{\Delta_i^2}} \tag{13}$$

then Theorem 1 still holds.

Warm Start Toward Nash Equilibrium in Zero-Sum Games

In general-sum games, regret matching converges to a coarse correlated equilibrium (Gordon, Greenwald, and Marks 2008). For the case of two-player zero-sum games, we now strengthen Theorem 1 to show that we get a warm start toward an approximate Nash equilibrium. A *Nash equilibrium* (Nash 1950) is a strategy profile σ such that $\forall i,\ u_i(\sigma_i, \sigma_{-i}) = \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$. An *$\epsilon$-Nash equilibrium* is a strategy profile σ such that $\forall i,\ u_i(\sigma_i, \sigma_{-i}) + \epsilon \geq \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$.

It is well known that if both players' regrets in a two-player zero-sum game are bounded by ϵ, then the average of their strategies across all iterations form a 2ϵ-Nash equilibrium (see, e.g., (Waugh 2009, p. 11)). We now prove we can obtain a similar result after regret transfer.

We first define a *weighted average strategy* as the average of a series of strategies where the first T iterations are weighted by w and the latter T_2 iterations by 1. Formally, we define $\sigma^w_{T,T_2,i}$ such that

$$p_{\sigma^w_{T,T_2,i}}(a) = \frac{w \sum_{t=1}^{T} p_{\sigma_{t,i}}(a) + \sum_{t=1}^{T_2} p_{\sigma_{t,i}}(a)}{wT + T_2} \tag{14}$$

Corollary 1. *Say both players have played T iterations of regret matching in a two-player game Γ. Let us transfer regret for both players to a new two-player zero-sum game with identical structure Γ' according to Theorem 1, and let $w = \min_i\{w_i\}$. If both players play an additional T_2 iterations of regret matching, then their weighted average strategies constitute a 2ϵ-Nash equilibrium where*

$$\epsilon = \max_i \left\{ \frac{\Delta_i \sqrt{|A_i|}}{\sqrt{(wT + T_2)}} \right\}$$

From (10), we see that the algorithm allows a range of valid values for the weight w_i. At first glance it may seem always better to use the largest valid w_i so as to get the most aggressive warm start via discounting the prior iterations by as little as possible. However, this is usually not the case in practice. Because regret matching in practice converges significantly faster than $\frac{\Delta_i \sqrt{|A_i|}}{\sqrt{T}}$, it may be possible to get a faster practical convergence by choosing a smaller w_i, even if this results in a theoretically worse convergence rate. One option—consistent with our theory—is to use $w_i = \frac{\Phi(\vec{R}_{T,i})}{\Phi(\vec{R}'_{T,i})}$; this performed well in our preliminary experiments, but has the slight downside of requiring repeated calculations of the potentials. Another option is to calculate w_i by replacing Δ_i with average payoff in (13). This performed well in practice and maintains the theoretical guarantees because w_i is guaranteed to be within the correct range. An additional benefit is that this way we express w_i as a function of the largest change in regret, which is typically easy to bound—an aspect we will leverage in the next section. Therefore, in the experiments in the rest of this paper we calculate w_i according to (13) with estimated average payoff instead of Δ_i.[1]

Generalization to Extensive-Form Games

We now present a corollary that the same algorithm can be applied to extensive-form games when solved using the *Counterfactual Regret Minimization (CFR)* algorithm described in (Zinkevich et al. 2007). CFR has emerged as the most widely used algorithm for solving large imperfect information games, for example poker (Sandholm 2010). Before presenting our result, we review extensive-form games and the CFR algorithm.

Definition of an Extensive-Form Game An extensive form game is defined as having the following features. This presentation follows from Osborne and Rubinstein (1994).

- A finite set N of players.
- A finite set H of sequences, the possible histories of actions, such that the empty sequence is in H and every prefix of a sequence in H is also in H. $Z \subseteq H$ are the terminal histories (those which are not a prefix of any other sequences). $A(h) = a : (h, a) \in H$ are the actions available after a nonterminal history $h \in H$.

[1] As a computational detail, we scale future iterations by $\frac{1}{w}$ rather than scaling previous iterations by w. Both yield identical results, but the former seems easier to implement.

- A function P that assigns to each nonterminal history (each member of $H \setminus Z$) a member of $N \cup c$. P is the player function. $P(h)$ is the player who takes an action after the history h. If $P(h) = c$ then chance determines the action taken after history h.
- A function f_c that associates with every history h for which $P(h) = c$ a probability measure $f_c(\cdot|h)$ on $A(h)$ ($f_c(a|h)$ is the probability that a occurs given h), where each such probability measure is independent of every other such measure.
- For each player $i \in N$ a partition $\mathcal{I}_i$ of $\{h \in H : P(h) = i\}$ with the property that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. For $I_i \in \mathcal{I}_i$ we denote by $A(I_i)$ the set $A(h)$ and by $P(I_i)$ the player $P(h)$ for any $h \in I_i$. We define $|A_i| = \max_{I_i} |A(I_i)|$ and $|A| = \max_i |A_i|$. $\mathcal{I}_i$ is the information partition of player i; a set $I_i \in \mathcal{I}_i$ is an information set of player i. We denote by $|\mathcal{I}_i|$ the number of information sets belonging to player i in the game and $|\mathcal{I}| = \max_i |\mathcal{I}_i|$.
- For each player $i \in N$ a payoff function u_i from Z to the reals. If $N = 1, 2$ and $u_1 = -u_2$, it is a zero-sum extensive game. Define $\Delta_i = \max_z u_i(z) - \min_z u_i(z)$ to be the range of payoffs to player i.

Counterfactual Regret Minimization In the CFR algorithm (Zinkevich et al. 2007), $u_i(\sigma, h)$ is defined as the expected utility to player i given that history h has occurred, and that all players then play according to σ. Next, *counterfactual utility* is defined as the expected utility given that information set I is reached, and all players play according to σ except that player i plays to reach I. Formally, if $\pi^\sigma(h)$ is the probability of reaching history h according to σ, and $\pi^\sigma(h, h')$ is the probability of going from history h to history h', then

$$u_i(\sigma, I) = \frac{\Sigma_{h \in I, h' \in Z} \pi^\sigma_{-i}(h) \pi^\sigma(h, h') u_i(h')}{\pi^\sigma_{-i}(I)}$$

Further, for all $a \in A(I)$, $\sigma|_{I \to a}$ is defined to be a strategy profile identical to σ except that player i always chooses action a when in information set I. *Immediate counterfactual regret* for an action is defined as

$$R^T_{i,imm}(I, a) = \frac{1}{T} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I)(u_i(\sigma^t|_{I \to a}, I) - u_i(\sigma^t, I))$$

and for an information set as

$$R^T_{i,imm}(I) = \max_{a \in A(I)} R^T_{i,imm}(I, a)$$

In CFR, on iteration $T + 1$ a player at an information set selects among actions $a \in A(I)$ by

$$p_{T+1}(a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a \in A(I)} R_i^{T,+}(I,a)}, & \text{if } \sum_{a \in A(I)} R_i^{T,+}(I, a) > 0 \\ \frac{1}{|A(I)|}, & \text{otherwise} \end{cases}$$

A key result is that the regret of an extensive-form game can be bounded by the game's immediate regret. Specifically:

$$R^T_i \leq \sum_{I \in \mathcal{I}_i} R^{T,+}_{i,imm}(I)$$

Moreover, one can bound immediate regret by

$$R^T_{i,imm}(I) \leq \frac{\Delta_i \sqrt{|A_i|}}{\sqrt{T}}$$

which gives

$$R^T_i \leq \frac{\Delta_i |\mathcal{I}_i| \sqrt{|A_i|}}{\sqrt{T}}$$

In CFR, the entire game tree is traversed each iteration, and each information set is updated accordingly. There are also many variants of CFR that take advantage of sampling to improve efficiency (see, e.g., (Lanctot et al. 2009)). In our experiments, we use *chance-sampled CFR*, where actions in chance nodes are sampled rather than being fully traversed. This variant has also been proven to converge.

Warm Starting for CFR We are now ready to present our result for extensive-form games.

Corollary 2. *Let Γ be an extensive-form game. Suppose player i has played T CFR iterations in Γ. Assume that all players play the exact same sequence of strategies in some other game Γ' with identical structure but potentially different payoffs. We define for each information set $I_i \in \mathcal{I}_i$*

$$\delta_{I_i} = \max \left\{ \max_a [R_{T,imm}(I_i, a) - R'_{T,imm}(I_i, a)],\ 0 \right\}$$

We also define w_{I_i} using δ_{I_i} according to Theorem 1. Let $w_i = min_{I_i}\{w_{I_i}\}$. If we scale the payoffs of the T iterations by w_i and then play according to CFR for T_2 iterations, then weighted average regret for player i is bounded by

$$R^w_{T,T_2,i} \leq \frac{\Delta_i |\mathcal{I}_i| \sqrt{|A_i|}}{\sqrt{w_i T + T_2}} \tag{15}$$

If Γ' is a two-player zero-sum game, then the weighted average strategies form an ϵ-Nash equilibrium, with $\epsilon = 2 \max_i R^w_{T,T_2,i}$.

Recall that in regret transfer we can replay the iterations in the new game in $O(1)$ by storing regrets as a function of $\vec{\theta}$. For example, in the case where $\vec{\theta}$ is one-dimensional, we would need to store two values for regret instead of one, and therefore require twice as much memory. However, in extensive-form games, not every information set may be affected by such a change in $\vec{\theta}$. If an information set's possible payoffs are all constant (independent of $\vec{\theta}$), even though there is a variable payoff somewhere else in the game, then there is no need to use extra memory to store the coefficients on $\vec{\theta}$ at that information set. The exact amount of memory used thus depends on the structure of the game.

Regret Transfer Experiments

We now show experimental results on regret transfer. We use Leduc hold'em poker (Southey et al. 2005) as the test problem here. It has become a common testbed because the game tree is rather large, but small enough that exploitability of a strategy can be computed, and thereby solution quality can be measured.

These experiments will show average exploitability as a function of the number of iterations run. In a two-player

zero-sum game, if v_i^* is the value of a Nash equilibrium solution for player i, then *exploitability* of player i is

$$e_i(\sigma_i) = v_i^* - \min_{\sigma'_{-i} \in \Sigma_{-i}} u_i(\sigma_i, \sigma'_{-i}) \tag{16}$$

Since $u_1 = -u_2$ in a two-player zero-sum game, we can define *average exploitability* of a strategy profile σ as

$$\begin{aligned} &\frac{e_i(\sigma_i) + e_{-i}(\sigma_{-i})}{2} \\ &= \frac{v_1^* - \min_{\sigma'_2 \in \Sigma_2} u_1(\sigma_1, \sigma'_2) + v_2^* - \min_{\sigma'_1 \in \Sigma_1} u_2(\sigma_2, \sigma'_1)}{2} \\ &= \frac{\max_{\sigma'_1 \in \Sigma_1} u_1(\sigma'_1, \sigma_2) - \min_{\sigma'_2 \in \Sigma_2} u_1(\sigma_1, \sigma'_2)}{2} \end{aligned} \tag{17}$$

In the experiments in this section, we consider warm starting after the allowed bet sizes in our game model have changed. We first estimated a solution to Leduc with 1 million CFR iterations. We then considered a modified form of Leduc where the bet sizes for the first and second round were slightly different. Specifically, we changed the first-round bet from 2 to 2.1 and the second-round bet from 4 to 4.1. In other words, $\vec{\theta}$ changed from $\langle 2, 4 \rangle$ to $\langle 2.1, 4.1 \rangle$. We tested three approaches to solving this new game. In the first, we simply solved the game from scratch using CFR. In the second, we transferred the regret and average strategy from the original Leduc game, but did not de-weight them. Finally, in the third, we transferred regret and average strategy by de-weighting by $w = 0.125$, a value chosen by the method described in the previous section. Figure 1 shows the results.

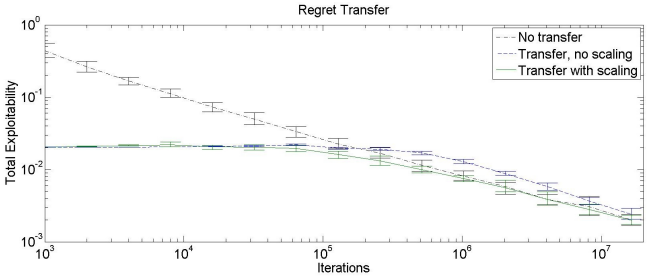

Figure 1: Regret transfer after increasing the bet size in both rounds of Leduc Hold'em by 0.1. The average over 20 runs is shown with 95% confidence intervals. The warm start provides a benefit that is equivalent to about 125,000 iterations. In the long run, that benefit becomes visually almost imperceptible on the log scale. Unlike transferring regret without scaling, our method does not cause long-term harm.

It is clear that while transferring without scaling provides a short-term improvement, in the long-run it is actually detrimental (worse than starting from scratch). In contrast, when the transferred regret is properly weighed using our method, we see an equal improvement in the short-term without long-term detriment. Since the transfer only gives us a head-start of a fixed number of iterations, in the long run this gain, of course, becomes negligible (on a logarithmic plot). Nevertheless, if the change is small then this fixed number of iterations can be a substantial portion of the run.

Parameter Optimization

We now incorporate the result from the previous section into a custom gradient descent algorithm for two-player zero-sum games. Our objective is to find the value of a parameter vector $\vec{\theta}$ that results in a locally maximal value for a Nash equilibrium for a family of games with payoffs that are Lipschitz functions of $\vec{\theta}$. For example, we could be optimizing the bet sizes for a player in no-limit poker. Without loss of generality, in this section we present everything as if we are maximizing the parameter vector for Player 1.[2]

To do this, we will simultaneously solve the parameterized game using regret matching (CFR in the case of extensive-form games), storing regret in terms of $\vec{\theta}$, while running our gradient descent algorithm. Specifically, at each step s we will run some number of iterations, t_s, of regret matching. Then, for each $\theta_d \in \vec{\theta}$ we will calculate an estimated derivative $\hat{g}_{d,s}$, as we will detail later. We will then update θ_d as

$$\theta_{d,s+1} = \theta_{d,s} + \hat{g}_{d,s} \frac{\alpha}{\ell_s} \tag{18}$$

where ℓ_s is a learning rate (analyzed later) and α is any positive constant. We then multiply the weights of all prior steps $s' \leq s$ by w_s, where w_s is determined according to (10). So, the previous step ends up being discounted by w_s, the one before that by $w_s \cdot w_{s-1}$ (because it was multiplied by that previous w_s before), etc.

The number of regret matching iterations we conduct at step s is

$$t_s = \lceil Ks - w_s K(s-1) \rceil \tag{19}$$

for some constant K.[3] Thus, at the end of step s we will have run an equivalent of Ks regret matching iterations, and by Theorem 1 we have weighted average regret $R^{w_s}_{K(s-1),t_s,i} \leq \frac{\Delta_i \sqrt{|A_i|}}{\sqrt{Ks}}$.

We compute the estimated derivative $\hat{g}_{d,s}$ (for each dimension d of $\vec{\theta}$ separately) as follows. Define $\vec{\xi}_d$ to be the unit vector in the vector space of $\vec{\theta}$ along dimension $\theta_d \in \vec{\theta}$. In concept, we will estimate the value of the game at two points $\vec{\theta}_s \pm s^{-\frac{1}{4}} \vec{\xi}_d$. We do this by running the equivalent of Ks iterations of regret matching at each of the two points. Then we use the slope between them as the estimated derivative. We later prove that this converges to the true derivative.

There is a problem, however, because $\vec{\theta}$ changes at each step and running Ks iterations from scratch becomes increasing expensive as s grows. To address this, we could transfer regret from a maintained strategy at $\vec{\theta}_s$ using Theorem 1, but even this would be too expensive because $s^{-\frac{1}{4}}$

[2]The function used to maximize the parameter vector can be independent of the zero-sum game. It can be maximized for either player, or parts of it could be maximized for one player or the other. It could even be maximized to the preferences of some third party.

[3]Theoretically, any positive constant K works, but in practice K could be chosen to ensure the overhead of stepping (conducting the re-weighting, calculating the gradient, etc.) is small compared to the cost of conducting the regret matching iterations.

shrinks very slowly. As a better solution to address the problem, we do the following. For each dimension d we maintain a strategy profile σ_{d-} for the game at $\vec{\theta}_s - s^{-\frac{1}{4}}\vec{\xi}_d$ and a strategy profile σ_{d+} for the game at $\vec{\theta}_s + s^{-\frac{1}{4}}\vec{\xi}_d$, and we estimate the derivative using those points, as well as transfer regret from them. At each step s, we run t_s iterations of regret matching at each of these $2N$ points. As $\vec{\theta}_s$ moves, these points move with it, so they are always at $\pm s^{-\frac{1}{4}}\vec{\xi}_d$ from it along each dimension d, respectively.

The pseudocode of the full algorithm is shown as Algorithm 1. The following theorem proves that Algorithm 1 is

Algorithm 1 Parameter optimization in two-player zero-sum games

Choose K, ℓ_s
Choose $U \leq \Delta_i$ // In the experiments we used average payoff of the player.
Initialize $s \leftarrow 1, t \leftarrow 0$
Initialize N-dimensional parameter vector $\vec{\theta}$
Initialize $\Theta \leftarrow \{\vec{\theta}_{d-} = \vec{\theta} - \vec{\xi}_d, \vec{\theta}_{d+} = \vec{\theta} + \vec{\xi}_d : d \in \{1...N\}\}$
Initialize avg regret tables: $\forall p \in \Theta\ \forall I\ \forall a, r_p(I,a) \leftarrow 0$
Initialize avg strategy tables: $\forall p \in \Theta\ \forall I\ \forall a, \sigma_p(I,a) \leftarrow 0$
loop
  while $t < sK$ **do**
    for all $p \in \Theta$ **do**
      $r_p, \sigma_p \leftarrow$ *One-Iteration-of-Regret-Matching*(r_p)
    $t \leftarrow t + 1$
  for all d in 1 to N **do** // Loop over the parameters
    $\hat{g}_d \leftarrow \frac{v_i(\sigma_{\vec{\theta}_{d+}}) - v_i(\sigma_{\vec{\theta}_{d-}})}{2s^{-\frac{1}{4}}}$ // Estimate derivative wrt. θ_d
    $\theta'_d \leftarrow \theta_d + \hat{g}_d \frac{\alpha}{\ell(s)}$ // Revise value of parameter θ_d
  for all d in 1 to N **do** // Narrow the interval around each θ_d
    $\vec{\theta}_{d-} \leftarrow \vec{\theta}' - \vec{\xi}_d(s+1)^{-\frac{1}{4}}$
    $\vec{\theta}_{d+} \leftarrow \vec{\theta}' + \vec{\xi}_d(s+1)^{-\frac{1}{4}}$
  $\delta \leftarrow \max\left\{\max_{p\in\Theta, I\in\mathcal{I}, a\in A}\left\{r'_p(I,a) - r_p(I,a)\right\}, 0\right\}$
  $w \leftarrow \frac{1}{1 + \frac{2\delta\sqrt{|A|t}}{U} + \frac{\delta^2 t}{U^2}}$ // Calculate weight based on Theorem 1
  for all $p \in \Theta, I \in \mathcal{I}, a \in A$ **do**
    $r_p(I,a) \leftarrow w r_p(I,a)$ // De-weight old regret
    $\sigma_p(I,a) \leftarrow w\sigma_p(I,a)$ // De-weight old average strategy
  $\vec{\theta} \leftarrow \vec{\theta}', \quad t \leftarrow wt, \quad s \leftarrow s+1$

correct and shows its speed advantage.

Theorem 2. *Let $\Gamma_{\vec{\theta}}$ be a family of two-player zero-sum games with identical structure. Assume each payoff is, across games, an identical function of some N-dimensional parameter vector $\vec{\theta}$ that is bounded so that $\forall d$, $\theta_d \in [a_d, b_d]$. Assume also that $\forall d$, these functions are Lipschitz continuous in θ_d. Let the learning rate ℓ_s be such that $\ell_s = \Omega(\sqrt{s})$ and $\frac{1}{\ell_s}$ diverges as $s \to \infty$. Define $v_i^*(\vec{\theta})$ to be the Nash equilibrium value of $\Gamma_{\vec{\theta}}$. As $s \to \infty$, Algorithm 1 converges to a locally optimal $\vec{\theta}$ with respect to $v_i^*(\vec{\theta})$, and to a Nash equilibrium strategy profile at that $\vec{\theta}$.*

Let

$$\hat{g}_{d,s} = \frac{v_i(\sigma_{d+}) - v_i(\sigma_{d-})}{2s^{-\frac{1}{4}}} \tag{20}$$

and let $\hat{g}_s = \max_d \hat{g}_{d,s}$. At each step s, Algorithm 1 conducts $O(s^{\frac{3}{2}}\hat{g}_s\frac{1}{\ell_s})$ iterations of regret matching.

This represents a substantial improvement over naïve gradient descent. If we were to do gradient descent without regret transfer, we would require $\Theta(s)$ iterations at each step, and thus take $\Theta(s^2)$ time. With regret transfer, however, if we use a learning rate $\ell_s = s$, and even if the gradient did not converge at any significant rate (although in reality it does), we only do $O(\sqrt{s})$ iterations at each step, thus taking $O(s\sqrt{s})$ time overall.

Parameter Optimization Experiments

Leduc Hold'em As we mentioned in the regret transfer experiments, one can view Leduc as having two parameters: the first-round bet size θ_1 and the second-round bet size θ_2.

In the first experiment here, we held θ_1 fixed at 2.0 (the standard in Leduc), and ran Algorithm 1 to optimize θ_2. We used a learning rate $l_s = s^{\frac{3}{4}}$, $\alpha = 50$, and $K = 100$. We conducted three runs of the algorithm, starting from three different initial values for θ_2, respectively. Each run converged to $\theta_2 = 9.75 \pm 0.01$ within 10^8 iterations (Figure 2).

As a sanity check, we ran CFR on 41 models of the game where we fixed $\theta_2 \in \{3.0, 3.25, ..., 12.75, 13.0\}$. Indeed, $\theta_2 = 9.75$ maximized Player 1's payoff.

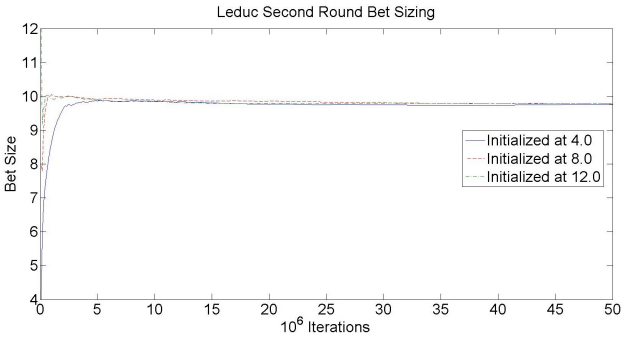

Figure 2: Parameter optimization where θ is the second-round bet size in Leduc Hold'em.

In the next experiment, we ran Algorithm 1 to simultaneously optimize θ_1 and θ_2. The same learning rate, α, and K were used as in the previous experiment. Three initial points were chosen: $(\theta_1, \theta_2) = (2.0, 4.0)$, $(4.0, 2.0)$, and $(4.0, 8.0)$. Within $5 \cdot 10^8$ iterations, all runs converged to $\theta_1 = 1.69 \pm 0.01$ and $\theta_2 = 8.56 \pm 0.01$. The results of these experiments are shown in Figure 3, with θ_1 on the bottom and θ_2 on the top. The value of the game at the converged $(\theta_1, \theta_2) = (1.69, 8.56)$ was 0.1645 ± 0.002. This was an improvement over the 1-dimensional optimization in the previous experiment, which fixed $\theta_1 = 2.0$ and converged to $\theta_2 = 9.75$. The value of the game there was 0.1611 ± 0.002.

No Limit Texas Hold'em We also provide results demonstrating that our algorithm even scales to two-player no-limit Texas Hold'em poker. The rules of the game can be found, for example, in Ganzfried and Sandholm (2013).

The following experiments were done with the betting abstraction used by Tartanian5 in the 2012 Annual Computer

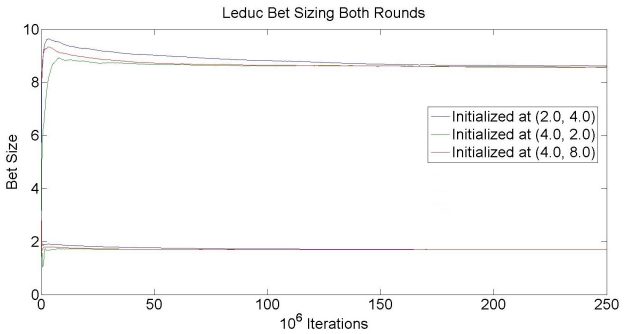

Figure 3: Parameter optimization where θ_1 is the first-round bet size in Leduc, and θ_2 is the second-round bet size.

Poker Competition. For the card abstraction, we used the leading algorithm (Johanson et al. 2013) for generating an imperfect-recall abstraction. We used no abstraction on the first round of the game (aka "preflop"), i.e., 169 buckets, and 200 buckets for each of the remaining three rounds. This resulted in an abstracted game with roughly 1.4 million information sets.

The 2012 betting abstraction allows the first player to act to either fold, call, raise, or go all-in. The raise amount is set to 1x the pot. This choice of bet size was based on human intuition, and may not be optimal. In these experiments we ran Algorithm 1 to find the raise amount that would be optimal for the first action of the game (in this abstraction).[4]

No-limit Texas Hold'em with an imperfect-recall card abstraction posed a challenge in that evaluating $v(\sigma)$ is difficult in large imperfect-recall games. To get around this, we estimated $v(\sigma)$ while running CFR at each step using the same public card samples. This resulted in some noise in the steps.[5] Nevertheless, after 10^9 iterations of chance-sampled CFR, all 4 runs converged to 0.77 ± 0.01 (Figure 4).

As a sanity check, we ran CFR on models of the game with a fixed bet size in $\{0.5, 0.6, ..., 1.0\}$. Indeed, 0.8 resulted in the highest expected payoff for Player 1, followed by 0.7.

[4]As the initial bet size changed, it was necessary to adhere to the chipstack limitations in No-limit Texas Hold'em. We dealt with this by maintaining the structure of the game, but limiting all payoffs to the size of the chipstack. When the initial bet size increased, all actions remained valid. However, some actions essentially had no consequence, as the chipstack limit had already been reached. When the initial bet size decreased, the all-in actions would nevertheless result in all chips being bet, up to the pre-determined chipstack limit. In this way we were able to maintain the structure of the game across parameter settings. However, when transferring regret, we did not explicitly consider the constraints of chip stacks. That is, if regret from an iteration with a parameter of 1.0 was transferred to a parameter of 1.1, then we would implicitly assume that the new chipstack was 1.1x the original. This was only the case for iterations that were transferred, and not for iterations being played at the current bet size. Moreover, as the parameter converged, this became an increasingly insignificant issue.

[5]Due to the challenge of evaluating $v_i(\sigma)$ in large imperfect-recall games, K was set to a minimum of 5,000 and a maximum of 200,000. We stepped when we had statistical significance that the derivative was of the right sign; otherwise we stepped at 200,000.

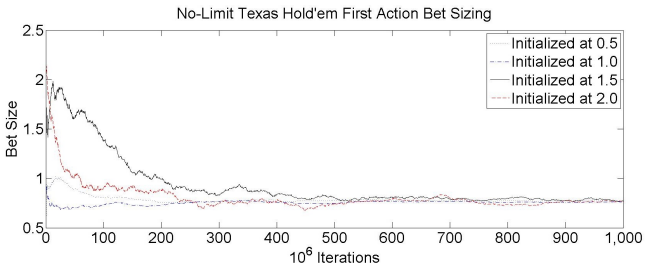

Figure 4: Parameter optimization where θ is the first action bet size in no-limit Texas Hold'em. Runs with four different initializations are shown. The learning rate was $s^{\frac{3}{4}}$. For initializations at 0.5 and 1, $\alpha = 0.3$. For initializations at 1.5 and 2.0, $\alpha = 1.0$.

Conclusions

Regret matching is a widely-used algorithm for learning how to act. We began by proving that regrets on actions in one setting (game) can be transferred to warm start the regrets for solving a different setting with same structure but different payoffs that can be written as a function of parameters. We proved this can be done by carefully discounting the prior regrets. This provides, to our knowledge, the first principled warm-starting method for no-regret learning. It also extends to warm-starting the widely-adopted counterfactual regret minimization (CFR) algorithm for large extensive-form incomplete-information games. Experiments on Leduc Hold'em poker verified the benefits of the technique. It provided a significant head start over running CFR from scratch. Furthermore, if one were to warm start without our careful discounting of prior iterations, one would get a warm start that is no better in the short run than ours, and is worse in the long run than starting from scratch. Our technique has no such detriment.

We then studied optimizing a parameter vector for a player in a two-player zero-sum game (e.g., optimizing bet sizes to use in poker). We proposed a custom gradient descent algorithm that provably finds a locally optimal parameter vector while leveraging our warm-start theory to significantly save regret-matching iterations at each step. It optimizes the parameter vector while simultaneously finding an equilibrium. To conduct s steps, it takes $O(s\sqrt{s})$ time (and significantly less than that in practice) while straightforward gradient descent takes $\Theta(s^2)$ to conduct those same steps.

We ran experiments in no-limit Leduc Hold'em and no-limit Texas Hold'em to optimize bet sizing. This amounts to the first action abstraction algorithm (algorithm for selecting a small number of discrete actions to use from a continuum of actions—a key preprocessing step for solving large games using current equilibrium-finding algorithms) with convergence guarantees for extensive-form games. Very few action abstraction algorithms have been presented to date, and the prior ones either have no convergence guarantees (Hawkin, Holte, and Szafron 2011; 2012) or are for a much narrower game class, stochastic games (Sandholm and Singh 2012).

Acknowledgements

This material is based on work supported by the National Science Foundation under grants IIS-1320620, CCF-1101668, and IIS-0964579, as well as XSEDE computing resources provided by the Pittsburgh Supercomputing Center.

References

Cesa-Bianchi, N., and Lugosi, G. 2006. *Prediction, learning, and games*. Cambridge University Press.

Ganzfried, S., and Sandholm, T. 2013. Action translation in extensive-form games with large action spaces: Axioms, paradoxes, and the pseudo-harmonic mapping. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Gordon, G. J.; Greenwald, A.; and Marks, C. 2008. No-regret learning in convex games. In *Proceedings of the 25th international conference on Machine learning*, 360–367. ACM.

Gordon, G. J. 2007. No-regret algorithms for online convex programs. *Advances in Neural Information Processing Systems* 19:489.

Greenwald, A.; Li, Z.; and Marks, C. 2006. Bounds for regret-matching algorithms. In *ISAIM*.

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

Hawkin, J.; Holte, R.; and Szafron, D. 2011. Automated action abstraction of imperfect information extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Hawkin, J.; Holte, R.; and Szafron, D. 2012. Using sliding windows to generate action abstractions in extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Hoda, S.; Gilpin, A.; Peña, J.; and Sandholm, T. 2010. Smoothing techniques for computing Nash equilibria of sequential games. *Mathematics of Operations Research* 35(2):494–512. Conference version appeared in WINE-07.

Johanson, M.; Burch, N.; Valenzano, R.; and Bowling, M. 2013. Evaluating state-space abstractions in extensive-form games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte Carlo sampling for regret minimization in extensive games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 1078–1086.

Nash, J. 1950. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences* 36:48–49.

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. MIT Press.

Sandholm, T., and Singh, S. 2012. Lossy stochastic game abstraction with bounds. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*.

Sandholm, T. 2010. The state of solving large incomplete-information games, and application to poker. *AI Magazine* 13–32. Special issue on Algorithmic Game Theory.

Southey, F.; Bowling, M.; Larson, B.; Piccione, C.; Burch, N.; Billings, D.; and Rayner, C. 2005. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 550–558.

Waugh, K. 2009. Abstraction in large extensive games. Master's thesis, University of Alberta.

Zinkevich, M.; Bowling, M.; Johanson, M.; and Piccione, C. 2007. Regret minimization in games with incomplete information. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.