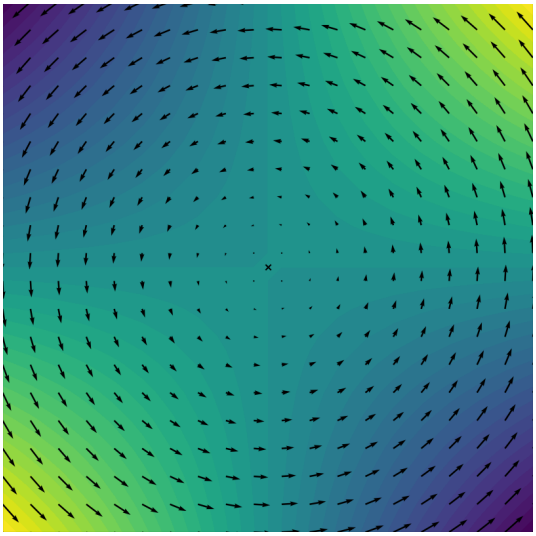
Incremental Learning Report:
Adaptive extra-gradient methods
for min-max optimization and games



Lucas MARANDAT, Yann TREMENBERT

June 3, 2023

1 Introduction

1.1 Context and objectives

This project was done in the context of *Incremental Learning and Game Theory* course for the Master IASD Paris Dauphine - PSL, under the supervision of Pr. Guillaume Vigeral and Pr. Rida Laraki. In this project, each pair of students was given a recent article about one of the subjects tackled in the lectures, and was supposed to read, analyze and implement some parts of it.

The article under study is *Adaptive Extra-Gradient Methods for Min-Max Optimization and Games* [1] by Kimon Antonakopoulos, E. Veronica Belmega and Panayotis Mertikopoulos, published in 2020.

In this article, the authors present a new algorithm, called ***AdaProx (AP)*** for Adaptive Proximal, to tackle min-max optimization problems. The main contributions of Adaprox can be summarized in three principal points:

- It offers an optimal convergence rate interpolation between smooth ($\mathcal{O}(1/T)$) and non-smooth problems ($\mathcal{O}\sqrt{1/T}$), where T is the number of iterations and where smoothness refers to the problem's vector field.
- It can be applied even when the usual domain boundedness and Lipschitz assumptions made in the literature do not hold.
- It requires neither any previous knowledge the problem's parameters nor hyperparameter tuning (e.g., Lipschitz modulus or domain diameter), and can therefore be applied straightforwardly.

Since neither code nor experiments' precise parameters are given in the article, we chose to re-implement the algorithm from the pseudocode, as well as some other comparable algorithms such as the ***Universal Mirror-Prox (UMP)*** algorithm of Bach and Levy [2] and the well-known ***Extra-Gradient (EG)*** algorithm of Korpelevich [3].

All our code is available on GitHub.

1.2 Problem definition

We are considering min-max, or saddle-point, optimization problems and games such as:

$$\min_{\theta\in\Theta}\max_{\phi\in\Phi}\mathcal{L}(\theta,\phi) \tag{1}$$

where Θ and Φ are the action spaces, i.e. convex subsets of some ambient real space and $\mathcal{L}:\Theta\times\Phi\longrightarrow\mathbb{R}$ is the problem's loss function. One can see this problem as a zero-sum two player game where each player wants to minimize their loss function: Player 1 controls θ and wants to minimize $\mathcal{L}(\theta,\phi)$ and Player 2 controls ϕ and wants to minimize $-\mathcal{L}(\theta,\phi)$. Such game will be developed and numerically solved by our different algorithms in further sections with the simple yet very instructive example of $\mathcal{L}(\theta,\phi)=\theta\phi$.

Most of the time, the players' loss functions will be considered individually differentiable, at least on the relative interiors of the action space, i.e. on a subset $\mathcal{X}$ of actions space $\mathcal{K}$ such that $ri(\mathcal{K})\subseteq\mathcal{X}\subseteq\mathcal{K}\subseteq\mathbb{R}^d$.

As underlined by Nemirovski [4], min-max optimization problems or games can be restated with the "vector field formulation" as a *variational inequality* of the form:

$$\text{Find } x^*\in\mathcal{X} \text{ such that } \langle V(x^*),x-x^*\rangle\geq 0,\ \forall x\in\mathcal{X} \tag{2}$$

where V is any vector field verifying the following inequality for each player i with individual loss l_i and taking action x_i while other players take actions x_{-i} such that $x:=(x_i;x_{-i})$ is in the action space $\mathcal{K}$:

$$\ell_i(x_i'; x_{-i}) \geq \ell_i(x_i; x_{-i}) + \langle V_i(x), x_i' - x_i \rangle \text{ for all } x \in \mathcal{X},\ x' \in \mathcal{K} \text{ and all players } i \tag{3}$$

In this case, $V_i(x)$ can be seen as the gradient of ℓ_i with respect to x_i.

For example, for the simple min-max problem (1), the goal would be:

$$\text{Find } x^* = (\theta^*, \phi^*) \in \Theta \times \Phi \text{ such that } \langle V(x^*), x - x^* \rangle \geq 0,\ \forall x \in \Theta \times \Phi \tag{4}$$

$$\text{where } V(\theta, \phi) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta} \\ -\frac{\partial \mathcal{L}}{\partial \phi} \end{bmatrix} = \begin{bmatrix} \phi \\ -\theta \end{bmatrix} \tag{5}$$

The solution x^* extracted from (2) is tantamount to finding a Nash Equilibrium of the game.

One of the only assumption made by the authors is the *monotonicity condition* which is formulated as follows:

$$\langle V(x) - V(x'), x - x' \rangle \geq 0,\ \forall x, x' \in \mathcal{X} \tag{6}$$

Such property translates in having a convex-concave loss function $\mathcal{L}$.

2 Optimization algorithms

2.1 Behaviors and applicability comparison

Even though they make different assumptions and have different properties, the three algorithms under study - EG, UMP and AP - have a similar pseudocode based on gradient descent which can be summarized like so:

Algorithm 1: Gradient descent basis for EG, UMP and AP

Data: problem-related data $\mathcal{S}$
Result: approximation $\widetilde{X_T}$ of the optimal point
$X_t \leftarrow X_0$;
$\eta_t \leftarrow \eta_0$;
$N \leftarrow n$;
for *steps in* N **do**
  $X_{t+1/2} \leftarrow \text{Proj}(X_t, \eta_t, V_t)$;
  $X_{t+1} \leftarrow \text{Proj}(X_t, \eta_t, V_{t+1/2})$;
  $\eta_{t+1} \leftarrow \text{Update\_stepsize}(\eta_t, \mathcal{S})$;
end

The basic idea of such algorithm is to alleviate the rotational momentum of min-max games by getting the gradient $V_t = V(X_t)$ twice on recursive points, but applying the second gradient to the first point in order to converge faster towards the saddle-point.

Whether we use EG, UMP or AP, different assumptions are made and the prior knowledge $\mathcal{S}$ to give to the algorithm, the Proj and η_t update functions and the $\widetilde{X_T}$ formula will differ from one algorithm to the other.

Tables 1 and 2 respectively summarize the different necessary assumptions for each algorithm and the different formulae to use in Algorithm 1.

Table 1: Necessary assumptions (✓: needed, •: not needed)

	EG	UMP	AP
Boundedness of $\mathcal{K}$	✓	✓	•
$\hookrightarrow$ Diameter D of $\mathcal{K}$	•	✓	•
Knowledge of Lipschitz continuity of V	✓	•	•
$\hookrightarrow$ Lipschitz modulus L	✓	•	•
Boundedness modulus M of V	•	✓	•

where:

- Diameter $D = \max_{x\in\mathcal{K}} h(x) - \min_{x\in\mathcal{K}} h(x)$ with $h(x)$: $\mathcal{K} \longrightarrow \mathbb{R}$ a chosen continuously-differentiable strongly convex function such that $\min_{x\in\mathcal{K}}[h(x) + \langle x, x'\rangle]$, $x' \in \mathcal{K}$ is easy to solve.
- Lipschitz, or smoothness, modulus L is such that $\|V(x) - V(x')\| \leq L\|x - x'\|$, $\forall x, x' \in \mathcal{X}$.
- Boundedness modulus M is such that $\exists M > 0$, $\|V(x)\| \leq M$, $\forall x \in \mathcal{X}$.

Table 2: Implementation formulae (**bold**: data $\mathcal{S}$ to be given upstream)

	EG	UMP	AP
$\widetilde{X_T}$	$\frac{\sum_{t=1}^T \eta_t X_{t+1/2}}{\sum_{t=1}^T \eta_t}$	$\frac{\sum_{t=1}^T X_{t+1/2}}{T}$	$\frac{\sum_{t=1}^T \eta_t X_{t+1/2}}{\sum_{t=1}^T \eta_t}$
$\text{Proj}(X_t, \eta_t, V_t)$	$\Pi(X_t - \eta_t V_t)$	$\mathcal{P}_{X_t}^{UMP}(\eta_t, V_t)$	$\mathcal{P}_{X_t}^{AP}(-\eta_t V_t)$
η_t	$\propto 1/\sqrt{t}$ (**not-smooth**) or $< 1/\mathbf{L}$ (**smooth**)	$\mathbf{D}/\sqrt{\mathbf{M}^2 + \sum_{s=1}^{t-1} Z_s^2}$	$1/\sqrt{1 + \sum_{s=1}^{t-1} \delta_s^2}$

where $Z_t^2 = \frac{\|X_{t+1} - X_{t+1/2}\|^2 + \|X_{t+1/2} - X_t\|^2}{5\eta_t^2}$ and $\delta_t = \|V_{t+1/2} - V_t\|_{X_{t+1/2},*} = \|V(X_{t+1/2}) - V(X_t)\|_{X_{t+1/2},*}$. The $\|.\|_{X_\cdot,*}$ norm, further defined as the dual Finsler metric, or dual "local" norm, will help us to extend the algorithm out of the common assumptions of Lipschitz continuity or bounded gradient made in the literature. This is the main core contribution of the article, is pretty technical, and will be developed in the next session. For now, you can simply consider it as a local equivalent of the global Euclidean norm, which will help us to locally bound V.

While Π denotes the traditional Euclidean projector onto the set $\mathcal{X}$, new projection operators $\mathcal{P}$ are introduced for UMP and AP. On the one hand, for UMP, Bach and Levy [2] define it as:

$$\mathcal{P}_x^{UMP}(\eta, y) = \underset{x'\in\mathcal{X}}{\text{argmin}}(\langle y, x'\rangle + \frac{1}{\eta}\mathcal{D}_h(x', x)),\ \forall y \in \mathbb{R}^d \tag{7}$$

where $\mathcal{D}_h$ is the Bregman divergence induced by the strongly convex function h defined to infer the diameter D, meaning:

$$\mathcal{D}_h(x', x) = h(x') - h(x) - \langle \nabla h(x), x' - x\rangle,\ \forall x, x' \in \mathcal{X} \tag{8}$$

On the other hand, Antonakopoulos and al [1] describe the operator $\mathcal{P}_x^{AP}$ as its "Bregman equivalent" such that $\Pi(x + y) \rightsquigarrow \mathcal{P}_x^{AP}(y)$.

2.2 Particularity of AdaProx

The main contributions of AdaProx are its abilities to abstract from the action space $\mathcal{K}$ which can be unbounded and also from the values of the vector field V which could either be non-smooth or with singularities, i.e. $\|V(x)\| \longrightarrow +\infty$ for some x in $\mathcal{K}$. AdaProx will eventually converge to a solution of (2) without the

need to give it any prior knowledge about the problem's parameters (e.g. is the problem smooth, D, M, L...).

To achieve such performance, Antonakopoulos and al [1] rely on the Finsler metric defined as follows:

Definition. A Finsler metric on a convex subset $\mathcal{X}$ of $\mathbb{R}^d$ is a continuous function $F : \mathcal{X} \times \mathbb{R}^d \longrightarrow \mathbb{R}_+$ which satisfies the following properties for all $x \in \mathcal{X}$ and all $z, z' \in \mathbb{R}^d$:

1. *Subadditivity*: $F(x; z + z') \leq F(x; z) + F(x; z')$.
2. *Absolute homogeneity*: $F(x; \lambda z) = |\lambda| F(x; z)\ \forall \lambda \in \mathbb{R}$.
3. *Positive-definiteness*: $F(x; z) \geq 0$ with equality if and only if $z = 0$.

Once the user has found a suitable Finsler metric on $\mathcal{X}$, one can define the *primal/dual local norms* associated to their problem on $\mathcal{X}$, respectively:

$$\|z\|_x = F(x; z) \text{ and } \|v\|_{x,*} = \max\{\langle v, z\rangle : F(x; z) = 1\} \tag{9}$$

Furthermore, to convey their result, Antonakopoulos and al [1] need this Finsler metric to be *regular*, i.e. $\|v\|_{x',*}/\|v\|_{x,*} = 1 + \mathcal{O}(\|x' - x\|_x)\ \forall x, x' \in \mathcal{X}, v \in \mathbb{R}^d$. In such case, we will say that $\mathcal{X}$ equipped with a regular Finsler metric is a *Finsler space*, in which we can "metrically" bound the vector field V, even in cases where $\|V\| \longrightarrow +\infty$ or where V is not Lipschitz continuous.

Indeed, this enables us to say that a vector field $V : \mathcal{X} \longrightarrow \mathbb{R}^d$ is

- *Metrically bounded* if $\exists M > 0$ such that

$$\|V(x)\|_{x,*} \leq M\ \forall x \in \mathcal{X} \tag{10}$$

- *Metrically smooth* if $\exists L > 0$ such that

$$\|V(x') - V(x)\|_{x,*} \leq L\|x' - x\|\ \forall x', x \in \mathcal{X} \tag{11}$$

By defining such properties, the authors extend the casual assumptions on V for the convergence of optimization algorithms.

They also define their own projection operator, which is in fact identical to the one defined by Bach and Levy [2], except for the strongly convex function h which has now to be a *Bregman-Finsler function* (since this definition is fairly technical and not very meaningful for our analysis, we'll let the readers see it for themselves in [1]) :

$$\mathcal{P}_x^{AP}(y) = \underset{x' \in \mathcal{X}}{\operatorname{argmin}}(\langle y, x - x'\rangle + \mathcal{D}_h(x', x)),\ \forall y \in \mathbb{R}^d \tag{12}$$

Therefore,

$$\begin{aligned}
\mathcal{P}_{X_t}^{AP}(-\eta_t V_t) &= \underset{x' \in \mathcal{X}}{\operatorname{argmin}}(\langle -\eta_t V_t, X_t - x'\rangle + \mathcal{D}_h(x', X_t)) \\
&= \underset{x' \in \mathcal{X}}{\operatorname{argmin}}(\langle \eta_t V_t, x'\rangle + \mathcal{D}_h(x', X_t) + \text{constant w.r.t. } x') \\
&= \underset{x' \in \mathcal{X}}{\operatorname{argmin}}(\langle V_t, x'\rangle + \frac{1}{\eta_t}\mathcal{D}_h(x', X_t)) \\
&= \mathcal{P}_{X_t}^{UMP}(\eta_t, V_t)
\end{aligned} \tag{13}$$

Applying Algorithm 1 with this projection operator and the η_t update function given in Table 2 will eventually yield the wanted convergence rate in both smooth and not-smooth cases, i.e. $\mathcal{O}(1/T)$ and $\mathcal{O}(1/\sqrt{T})$

respectively, without having to manually tune the algorithm or giving it the problem's parameters, even in unbounded domains and in the case of singularities.

To better understand this result, one can build the following intuition:

- In the non-smooth case, to metrically-bound the vector field V guarantees that the difference sequence $\delta_t = \|V_{t+1/2} - V_s\|_{X_{t+1/2},*}$ is bounded (which could not be the case without the introduction of the Finsler metric) so that η_t will vanish at a $\mathbf{\Theta}(1/\sqrt{T})$ rate, which is the necessary rate for EG to converge in the non-smooth case.
- In the smooth case, $\eta_t \not\longrightarrow 0$ when $t \longrightarrow +\infty$ but tends towards a positive limit η_∞ because the infinite series $\sum_t \delta_t^2$ will be summable. Remember that in this case EG converges only if $\eta < 1/L$. Since the difference δ_t captures the variation of the gradient, $\sum_{t=1}^{T} \delta_t^2$ sums up all the "local" Lipschitz constants of V up to time T and is therefore expected to be bigger than L^2, which ensures convergence.

3 Experiments

Before diving into the experiments, we wanted to underline the crucial lack of implementation guidelines in the original article [1]. The authors tackle many welcomed examples but do not give many indications about the associated chosen norms or the chosen hyperparameters, such as the initial step size η_0. This would not have harmed if they had provided code through a GitHub link or in the Appendix, but none of them are to be found. Since one major contribution of this article is the application of AdaProx in a Finsler space as described in Section 2.2, we initially wanted to reproduce the *resource allocation problem* but encountered different analytical results for $V_i(x)$ along with implementation issues which prevented us from reiterating such experiment.

Because of this lack of code and implementation parameters, we initially focused on simpler and widely known problems, such as *"Prisoner's Dilemma"*, *"Battle of the Sexes"*, *"Matching Pennies"* and *"Rock-Paper-Scissors"*. Then, to validate our implementation, we tackled the original example problem (1) and compared our results with theirs. Finally, we aimed to reproduce the "bilinear min-max games" proposed in the original paper [1] to evaluate our implementation.

3.1 Simple games

These simple games have the advantages to be linked to our classes, to have a known optimal solution and to be drawable, which makes the implementation checking easier. These games can be summarized using the payoff matrix notation:

$$\min_{x \in \Delta(I)} \max_{y \in \Delta(J)} x^T A^{(i)} y$$

Where $A^{(i)}$ is either the payoff matrix of the zero-sum game or the payoff matrix of the interested player $i \in N$. $\Delta(I)$ and $\Delta(J)$ respectively denote the set of mixed strategies of players 1 and 2.

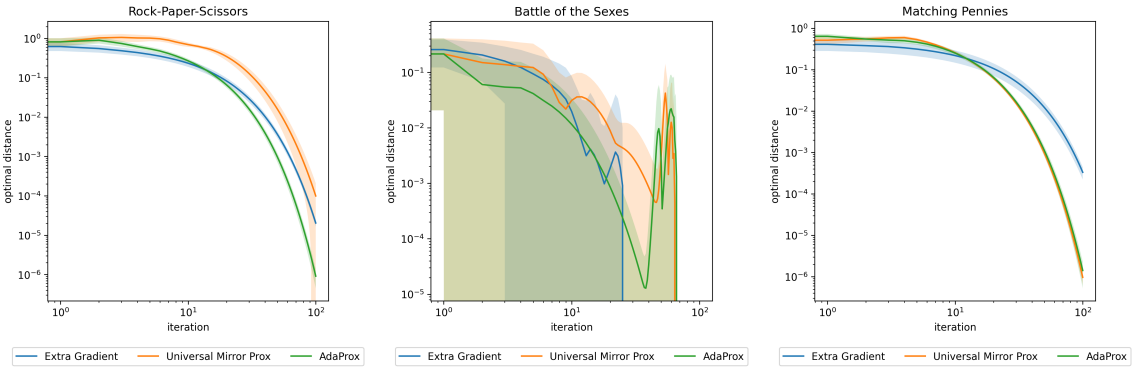


Figure 1: **(lower is better)** Convergence of algorithms on different simple games.
By tuning the Extra-Gradient method with a learning rate of $\eta = .1$ and the Universal Mirror Prox method with a diameter $D = .5$ and an initial $G_0 = 2$, we were able to achieve convergence to one of the Nash Equilibria for three simple games. AdaProx, which does not have any hyperparameters - hence does not require any tuning - converged at the same rate as the other tuned methods.

Notice that for the *"Battle of the sexes"* game, we are observing a spike in the convergence rate that may be attributed to the presence of two pure Nash Equilibria and a mixed one. The mixed equilibrium $(\frac{2}{3}, \frac{1}{3})$ has a lower payoff than the pure equilibria, which could cause the algorithm to initially converge to the mixed equilibrium and then diverge from it in order to get closer to a better equilibrium.
All algorithms were able to solve the *"Prisoner's Dilemma"* almost instantly, so there is no need for a plot to show their convergence.

3.2 The basic saddle-point problem

The simple saddle-point problem, as described in Section 1.2, has the advantage of being set in the Euclidean space. As a consequence, even if we were to consider the problem on $\mathbb{R}^2$ where $\lim_{\theta,\phi\to+\infty} V(\theta,\phi) = +\infty$, we could use the basic Euclidean norm as our Finsler metric because one can notice that the Euclidean norm satisfies Definition 2.2.

For simplicity and fair comparison with the original article, we decided to solve it for $\theta, \phi \in [-1, 1]$ and ran the three different algorithms to obtain the following results:

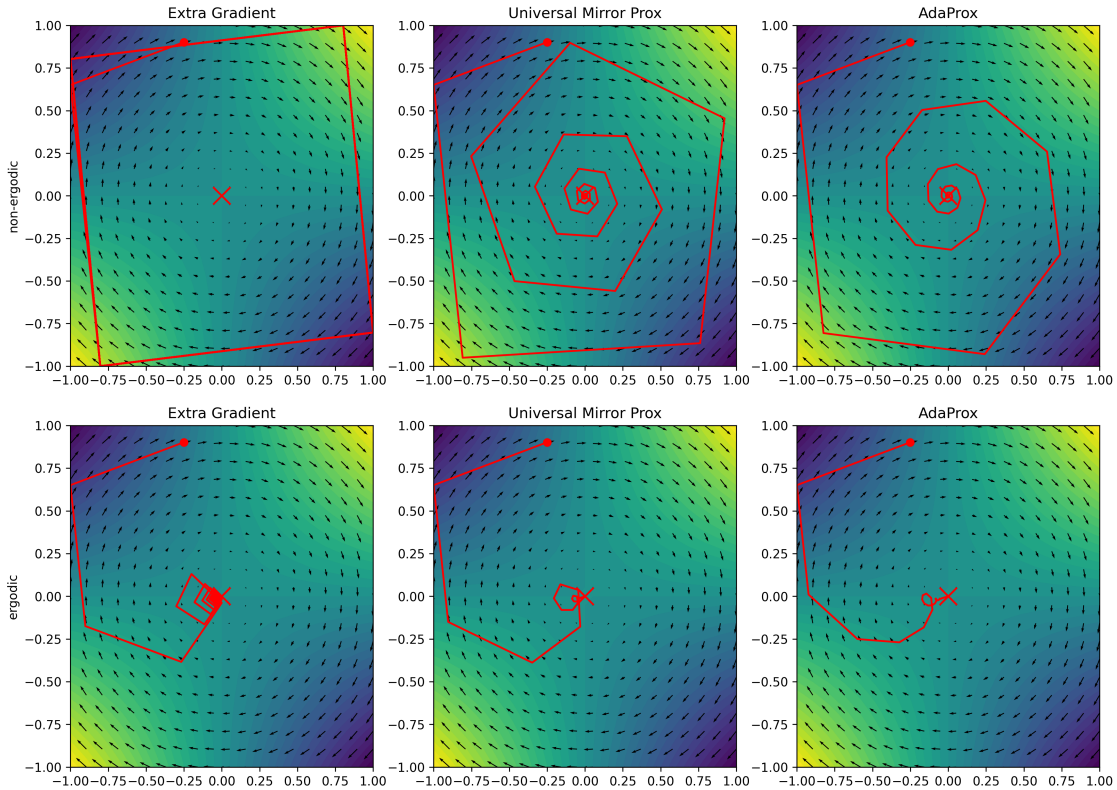


Figure 2: Convergence path of different algorithm (EG, UMP and AP) with the same starting point and using both ergodic and non-ergodic returns.

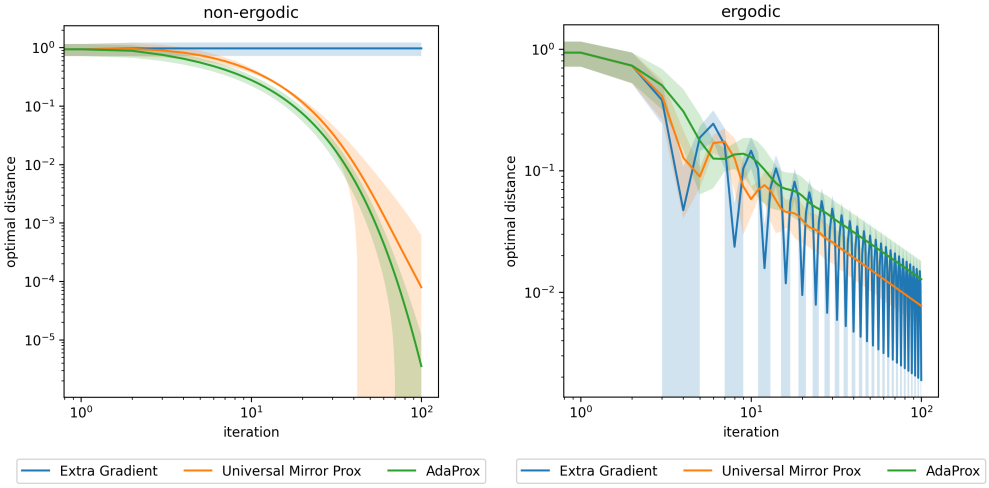


Figure 3: **(lower is better)** Convergence rate of different algorithm (EG, UMP and AdaProx) with the same starting point and using both ergodic and non-ergodic aggregated over 100 runs.

Three different observations can be made from the results of our experiment:

- First, we observe that the non-ergodic return, i.e. the last iterate X_T, provides better accuracy for both metrics - the distance to the optimal point $(0, 0)$ and the gradient norm. This observation corroborates

the statement of Antonakopoulos et al [1] conveying the idea that it is more common to harvest the last iterate than the ergodic average defined in Table 2.

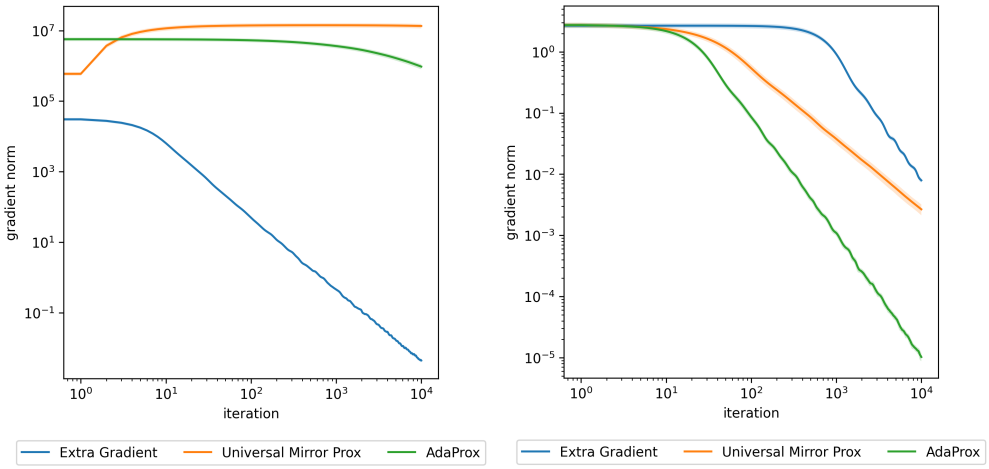
- Even though the UMP offers better results for the ergodic average, we still remind that this algorithm requires some input data $\mathcal{S}$, e. g. an initial guess of the boundedness modulus M and the domain's diameter D, which makes AP more convenient to use in practice.
- Finally, we actually obtained slightly different results from the original article [1]. Indeed, it looks like the gradients are not taken in the exact same points as theirs, which results in a different trajectory towards the optimal point. This difference may come from the fact that we had to guess the initial step size η_0. But even by tuning it, we still get some small differences.

3.3 Bilinear min-max games

Consider the min-max problem from the original paper [1] of the form:

$$\min_{\theta \in \mathbb{R}^d} \max_{\phi \in \mathbb{R}^d} (\theta - \theta^*)^T A(\phi - \phi^*)$$

For an unknown $\theta^*, \phi^* \in \mathbb{R}^d$ and $A \in \mathbb{R}^{d \times d}$ drawn i.i.d. component-wise from standard Gaussian. The original paper does not mention which standard deviation they used, using the unit one give us similar curves but with higher order for the gradient squared norm. After investigating, we found that normalizing A such that $\|A\| = 1$ gives a similar output with comparable order.



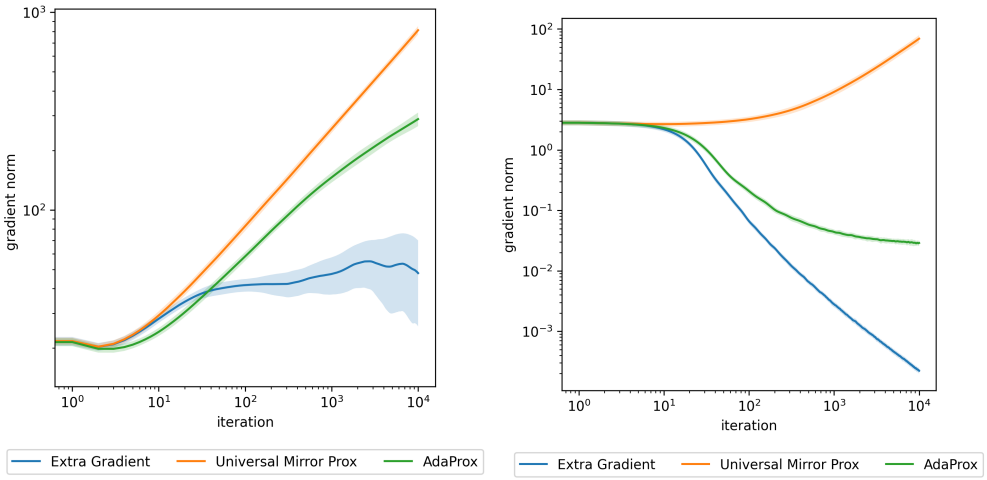
(a) Bilinear min-max games as the original paper described it with its tuned parameters.

(b) Bilinear min-max games with normalized A and tuned parameters $\eta = 1$ for EG and $D = \sqrt{2}$, $G_0 = 2$ for UMP.

Figure 4: **(lower is better)** Convergence of $\|V(\overline{X}_t)\|_2^2$ over 100 runs initialized with the same seed.

We are currently experiencing challenges in providing a definitive explanation for the observed differences. To gain a better understanding of the underlying reasons for this divergence, we aim to compare our implementation of both the problem and the algorithm with that of the authors. Unfortunately, they are not given.

In addition, the authors aimed to enhance the "full-gradient" framework by introducing noise to the true gradient through the equation $V_t = V(X_t) + U_t$, where U_t was drawn from a centered Gaussian distribution with a unit covariance matrix. Nevertheless, when we implemented this noisy framework, we found that the gradient was too noisy for our purposes. To address this problem, we adjusted our implementation by drawing U_t from a Gaussian distribution with a smaller variance of $\sigma = 10^{-1}$.



(a) Bilinear min-max games with the same noising process as described the original paper.

(b) Bilinear min-max games with $\sigma = 10^{-1}$.

Figure 5: **(lower is better)** Convergence of $\|V(\overline{X}_t)\|_2^2$ over 100 runs initialized with the same seed in the **noisy framework** with again the normalized A and the same hyperparameters as 4.
Our results significantly deviated from those reported in the paper within this framework. Despite dedicating a considerable amount of time to address this discrepancy, we were unable to achieve any meaningful progress.

4 Conclusion

All our implementations ended up providing convincing results compared to the theoretical ones. As a matter of fact, the AdaProx algorithm is a very suitable and recommendable algorithm to solve games and min-max optimization problems. Even though it sometimes shows worse convergence speed than Universal Mirror-Prox or Extra-Gradient algorithms, it offers a reliable rate of interpolation between smooth and non-smooth problems without feeding it any prior knowledge about the problem's parameters. Therefore, this algorithm is very suited to solve new unknown problems in practice.

This article proved to be very challenging, whether it be from a mathematical or implementation point of view. Indeed, it dived in the notions of Finsler space and primal/dual local norms, which could be quite intricate at first sight. Furthermore, as highlighted in Section 3, this article lacks crucial information in order to enable the reader to reproduce their results.

Nevertheless, it was an opportunity to discover the complexity of Incremental Learning and Game Theory, to make connections with the classes' material and to numerically solve famous Game Theory problems from scratch with our own environment.

References

[1] K. Antonakopoulos, E. V. Belmega, and P. Mertikopoulos, "Adaptive extra-gradient methods for min-max optimization and games," *arXiv preprint arXiv:2010.12100*, 2020.

[2] F. Bach and K. Y. Levy, "A universal algorithm for variational inequalities adaptive to smoothness and noise," in *Conference on learning theory*. PMLR, 2019, pp. 164–194.

[3] G. M. Korpelevich, "The extragradient method for finding saddle points and other problems," *Matecon*, vol. 12, pp. 747–756, 1976.

[4] A. Nemirovski, "Prox-method with rate of convergence o (1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems," *SIAM Journal on Optimization*, vol. 15, no. 1, pp. 229–251, 2004.