Efficient Regret Minimization in Non-Convex Games Project

Evgeny Tayarov 317442309

Zaruhi (Zara) Papyan 316648799

The 18^{th} of September 2020

1 Summary

1.1 Problem

While the classical definition of regret is appropriate for convex optimization problems, it is not appropriate for non-convex problems due to the NP-hardness of non-convex optimization, even in non-online settings. In 2017, the paper of Hazan et al. [2] introduced a novel definition called *local regret*, which generalizes the concept of regret to online non-convex problems. Specifically, local regret is defined as the regular regret function, except it is only computed in a local time window, corresponding to a small subset of all optimization steps. The intuition behind local regret was articulated beautifully in a lecture by the esteemed Prof. Hazan:

> *"Since convexity is important for guessing if we have a low objective, we can deduce that local information can tell us something about the global function."*

The paper of Hazan et al. proposes several efficient algorithms that converge to the optimal local regret. These algorithms rely on averaging the gradients of the loss functions within a local time window. They will be discussed in detail in the following section.

1.2 Motivation

The main motivation behind this article – which in our humble opinion appears way too late in the manuscript – relates to the recent tremendous empirical success of Generative Adversarial Networks (GANs). As was demonstrated in several recent high impact publications, these deep learning systems are capable for example of generating photo-realistic images of faces. However, what is less impressive is the lack of theoretical guarantees behind GANs. The theoretical framework proposed by Hazan et al., and in particular the notion of local regret, attempts to ameliorate the situation. It may be that it is efficient in the field of regret minimization - since we can introduce regret minimization as a technique to reach equilibrium in games and use this to motivate the use of simultaneous GD in GANs.

2 Algorithms and theoretical results

2.1 Brief introduction

Define the time-smoothed objective function $F = \frac{1}{w} \cdot \sum_i f_{t-i}$, where w is the length of the time window and f_t is the loss functions at time t. The paper proposes to minimize the function F, instead of f, using the following three online optimization algorithms:

- ☐ `Algorithm 1`: Projected gradient descent on F.

- ☐ `Algorithm 2`: Stochastic gradient descent on F (no projection, gradients contaminated with Gaussian noise).
- ☐ `Algorithm 3`: Second order (*Newton*) optimization of F.

They also propose Algorithm 4 – a meta-algorithm for cooperative multi-player optimization.

The intuition behind these algorithms is that the time-smoothing of the cost function would create a simpler, less noisy landscape for the loss, which would be easier to optimize, and would benefit from some theoretical guarantees.

2.2 Assumptions and Definitions of theorems

2.2.1 Assumptions

Assumption(`Algorithm 1`). *Let us assume the following is true for each* ***loss function***

- f_t *is bounded:* $\|f_t(x)\| \leq M$
- f_t *is L-Lipschitz:* $\|f_t(x) - f_t(y)\| \leq L\|x - y\|$
- f_t *is* $\beta -$ *smooth (has a* β*-Lipschitz gradient):* $\|\nabla f_t(x) - \nabla f_t(y)\| \leq \beta\|x - y\|$

Assumption(`Algorithm 2`). *We assume that each call to the stochastic gradient oracle yields an i.i.d. random vector* $\tilde{\nabla} f(x)$ *with the following properties:*

- *Unbiased:* $\mathbb{E}\left[\tilde{\nabla} f(x)\right] = \nabla f(x)$
- *Bounded Variance:* $\mathbb{E}\left[\|\tilde{\nabla} f(x) - \nabla f(x)\|^2\right] \leq \sigma^2$

Assumption(`Algorithm 3`). f_t *is twice differentiable and has an* $L_2 -$ *Lipschitz Hessian:*

$$\|\nabla f^2(x) - \nabla f^2(y)\| \leq L_2\|x - y\|$$

2.2.2 Definitions

Definition(s). *The definitions below are used through the paper:*

1. *The sliding-window time average of functions* f*, parametrized by some window size* $1 \leq w \leq T$*:*

$$F_{t,w}(x) := \frac{1}{w}\sum_{i=0}^{w-1}[f_{t-i}(x)]$$

2. *Fix some* $\eta > 0$*. The* w***-local regret*** *of an online algorithm:*

$$\mathfrak{R}_w(T) \stackrel{def}{=} \sum_{t=1}^{T}\|\nabla_{\mathcal{K}\eta} F_{t,w}(x_t)\|^2$$

3. *A stronger approximate-optimality condition:*

$$\Phi_t(x) := Max\left\{\|\nabla F_t(x)\|^2, -\frac{4\beta}{3L_2^2} \cdot \lambda_{min}(\nabla {F_t}^2(x))^3\right\}$$

Using the above assumptions, individually for every case, the article presented theorems regarding the bound of the *local regret* and the total number of gradient steps T taken by each algorithm.

- See Appendix A - regarding the main Assumptions and Definitions.
- See Appendix B - for the main proofs and theorems that support our claims.

2.3 Results

The main theoretical results are summarized in **Table 1** below.

Algorithm 1	Achieves $\mathfrak{R}_w \leq \mathrm{O}(T/w^2)$ by gradient descent, with $\mathrm{O}(Tw^2)$ gradient calls, where T is the number of optimization steps. If the cost functions f_t are equal for all t, the problem reduces to the off-line setting, and the theoretical result matches the best known rate for the convergence of gradient descent.
Algorithm 2	Achieves $\mathbb{E}[\mathfrak{R}_w] = \mathrm{O}(T/w)$. Furthermore, the algorithm makes a total of $\mathrm{O}(Tw)$ calls to the stochastic gradient oracle. And by choosing the following parameters: $w = \frac{12M\beta+2\sigma^2}{\epsilon}, T = 2w, \eta = \frac{1}{\beta}$ we can conclude that the algorithm makes $\mathrm{O}(\sigma^4/\epsilon^2)$ stochastic gradient oracle calls in total.
Algorithm 3	The iteration x_t satisfies $\sum_{t=1}^{T} \phi_t(x_t) = \mathrm{O}(T/w^2)$, where ϕ_t is defined in the appendix and intuitively measures the positive-definiteness of the Hessian and the magnitude of the gradient. The total number of iterations of the inner loop is $\mathrm{O}(Tw^2)$ and the inner loop uses the Newton method
Algorithm 4	For some $w \leq t \leq T$, the joint strategy is an ϵ-approximation (η, w)-smoothed local equilibrium.

Table 1: Main theoretical results

We may conclude that Algorithm 3 will likely converge the fastest, but it requires 2^{nd}-order information, which might not be readily available.

3 Thoughts

3.1 Thought 1

Since we are both graduating B.Sc. in mathematics - from out point of view – the paper is very basic. It relies on simple linear algebra and first course in probability that we have learned in our first and second year.

The main novelty of the paper is the use of *time smoothing (by t)* of the objective function. However, previous works, for example on the topic of multi-grid, already proposed using smoothing on the optimization variable (instead of on time). And so, the contribution seems somewhat limited. Also, the results derived seem not very surprising.

3.2 Thought 2

The paper of Hazan et al. [2] introduced a new notion of *local regret* for online non-convex problems. However, since the presented regret definition of local regret assumes a static best model, e.g. the main assumptions, it seemed to us a bit problematic.

That said, from a quick search under papers that used Hazan as a reference, on 2019, the paper of Aydore et al. [1] presented in a novel regret framework both on non-convexity and dynamic environment for online forecasting problems. They also claimed that their approach is more computationally efficient than the algorithm proposed by Hazan et al.

3.3 Thought 3

Let us propose the idea of *Quasi-Newton*, instead of the regular Newton method to accelerate the process in Algorithm 3. Here are the properties of each method:

Newton's Method	Quasi-Newton Method
1. Computationally expensive.	1. Computationally cheap.
2. Slow computation.	2. Fast(er) computation.
3. Calculation of 2^{nd} derivative is needed.	3. No need for 2^{nd} derivative.
4. Solving linear system of equations is needed.	4. No need to solve linear system of equations.
5. Less convergence steps.	5. More convergence steps.
6. More precise convergence path.	6. Less precise convergence path.

The disadvantages: The main disadvantage of QNM is the need to store the inverse Hessian approximation. This could require a large amount of memory and could thus be detrimental in the cases of large complicated systems. In general, QNMs work by replacing the Hessian with a linear operator A, where A is positive definite and solving linear systems $Ax = b$ is fast. Now, the trade off is that if A is a poor approximation of the Hessian, the quasi-Newton method may converge slowly. Furthermore, it is a known fact that coming up with an effective approximation of the Hessian is not easy.

Having said that - The advantages: Although, 5 and 6 look like serious weak points of QNMs, property 2 changes the whole picture. In conclusion - we still believe that the QNMs method can induce a better result in Algorithm 3 - at least with smooth enough functions.

3.4 Thought 4

In the last part of the paper, we are introduced with the idea of *Generative Adversarial Networks*, GANs. In a nutshell, the key idea of GANs is to learn both the generative model and the loss function at the same time. The resulting training dynamics is a game between a generator (the generative model) and a discriminator (the loss function). The goal of the generator is to produce realistic samples that fool the discriminator, while the discriminator is trained to distinguish between the true training data and samples from the generator. In GAN training, the time-smoothed algorithms introduced in this paper are better known as experience replay, a known technique for promoting stability. Let us notice that there are few main known problems with this method:

1. **Non-convergence:** the model parameters oscillate, destabilize and never converge
2. **Collapse:** the generator collapses which produces limited varieties of samples
3. **Diminished gradient:** the discriminator gets too successful that the generator gradient vanishes and learns nothing
4. **Unbalance between the generator and discriminator:** causing over-fitting, and high sensitivity to the hyper parameter selections.

These problems seems to indicate that convergence in GANs is highly problematic. It therefore puts into question the theoretical results proved in the paper and their applicability to practical scenarios.

The use of 2^{nd} order optimization methods for training GANs is highly strange since no one does that in practice and it would completely fail, as any practitioner in deep learning would know.

3.5 Thought 5

In deep learning optimization it is commonly known that training converges to the global optimum, despite the non-convexity of the problem. Often, this phenomenon is attributed to the grose overparameterization of the network. The main idea of this paper is to prove convergence to a local equilibrium or a local minima, which seems strange in light of what is known in deep learning theory. It would therefore be more interesting to investigate under which conditions does one converge to the global optimum.

References

[1] Sergul Aydore, Tianhao Zhu, and Dean P Foster. "Dynamic Local Regret for Non-convex Online Forecasting". In: *Advances in Neural Information Processing Systems*. 2019, pp. 7982–7991.

[2] Elad Hazan, Karan Singh, and Cyril Zhang. "Efficient regret minimization in non-convex games". In: *arXiv preprint arXiv:1708.00075* (2017).

Appendices

Appendix A :Main Assumptions and Definitions

Let us first introduce several definitions that play an important role in understating the main algorithms. Starting with *online non-convex optimization*, whose settings are modeled as a game between a learner and an adversary. During each iteration t,

- The learner commits to a decision $x_t \in \mathcal{K}$, where $\mathcal{K} \subseteq \mathbb{R}^n$ is a convex decision set $\mathcal{K}$
- At the same time, the adversary chooses a loss function $f : \mathcal{K} \to \mathbb{R}$, such that for each x_t in the decision set $\mathcal{K}$ there exists a loss function f_t
- The learner then observes $f_t(x)$, known for every x, and suffers a loss of $f_t(x_t) \in \mathbb{R}$

That is to say that the usual goal from online convex optimization is achieving a low regret. Similarly, the performance of the learner is measured through its regret, which is defined as a function of the loss sequence $\{f_i\}_{i=1}^T$ and the sequence of online decisions $\{x_i\}_{i=1}^T$ made by the learner.

In mathematical optimization, there is a *constrained* and *unconstrained* optimization. Roughly speaking, in the first one we deal with *gradient descent* and in the second we deal with *projected gradient descent*. See *Figure 1.* to understand the main difference. Since constrained non-convex optimization can be computationally hard, notably we can handle the constraints with *projected gradient descent*.

(a) Finding a small ∇f is intractable

(b) Finding a small $\nabla_{\mathcal{K}\eta} f$ can be done with PGD

Figure 1: Projected Gradients for Constraint Sets

Definition(2.2). *Let $f : \mathcal{K} \to \mathbb{R}$ be a differentiable function on a closed (but not necessarily bounded) convex set $\mathcal{K} \subseteq \mathbb{R}^n$. Let $\eta > 0$. We define $\nabla_{\mathcal{K}\eta} f : \mathcal{K} \to \mathbb{R}^n$, the $(\mathcal{K}, \eta)$ -* ***projected gradient*** *of f, by:*

$$\nabla_{\mathcal{K}\eta} f(x) = \frac{1}{\eta}\Big(x - \Pi_{\mathcal{K}}[x - \eta \nabla f(x)]\Big) \tag{1}$$

where $\Pi_{\mathcal{K}}[.]$ denotes the orthogonal projection onto $\mathcal{K}$.

To give an illustration, this can be viewed as a surrogate for the gradient which can ensure that the gradient descent step will always lie within $\mathcal{K}$, by transforming it into a projected gradient descent step. Indeed, one can verify by definition - as:

$$x - \eta \nabla_{\mathcal{K}\eta} f(x) = \overbrace{\Pi_{\mathcal{K}} \underbrace{[x - \eta \nabla f(x)]}_{\text{gradient step}}}^{\text{projected gradient descent step}}$$

as: And in particular, when $\mathcal{K} = \mathbb{R}^n$

$$\nabla_{\mathcal{K}\eta} f(x) = \frac{1}{\eta}\Big(x - x + \eta\nabla f(x)\Big) = \nabla f(x)$$

meaning we retrieve to the usual gradient at all x. Let us note that there always exists a point with vanishing projected gradient. In particular - See *Figure 2.* for an illustration that shows how it looks.

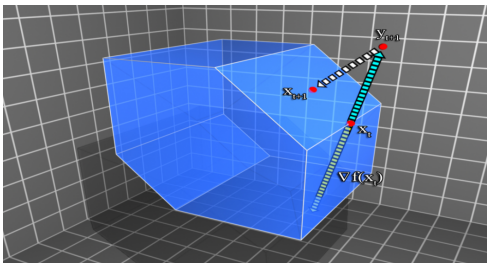

Figure 2: [Visualisation of Orthogonal projection onto $\mathcal{K}$]. It is simple to optimize over $\mathbb{R}^n$. However, if we have constrains, for instance we want to minimize the function from a subset of an euclidean space, like the blue shape above. Then moving in this direction might make us step outside - in which case we have to return. That can be done easily with *projection*: We take the closest point inside x_{t+1} to the point outside $y_{t+1} = x_t - \eta\nabla f(x_t)$

Now, The next proposition introduces us kind of 'triangle inequality' for the gradient -

Proposition(2.4). *Let x be any point in $\mathcal{K} \subseteq \mathbb{R}^n$, and let f, g be differentiable functions $\mathcal{K} \to \mathbb{R}$. Then, for any $\eta > 0$,*

$$\|\nabla_{\mathcal{K}\eta} f + g(x)\| \leq \|\nabla_{\mathcal{K}\eta} f(x)\| + \|\nabla g(x)\| \tag{2}$$

Proof.

$$[a]\ u = x + \eta\nabla f(x) \qquad [b]\ v = u + \eta\nabla g(x)$$

Define their respective projections: $u' = \Pi_{\mathcal{K}}[u]$, $v' = \Pi_{\mathcal{K}}[v]$ so that $u' = x - \eta\nabla_{\mathcal{K}\eta} f(x)$ and $v' = x - \eta\nabla_{\mathcal{K}\eta} f + g(x)$. By the generalized Pythagorean theorem for convex sets:

$$[1]\ \langle u' - v', v - v'\rangle \leq 0 \qquad [2]\ \langle v' - u', u - u'\rangle \leq 0 \qquad [3]\ \langle u' - v', u' - v' - (u - v)\rangle \leq 0$$

when from inner product property [1], [2] yields [3]. Now, by using Cauchy Schwartz inequality, we get:

$$[4]\ \|u' - v'\|^2 \leq \langle u' - v', u - v\rangle \leq \|u' - v'\|\|u - v\|$$

Finally, by the triangle inequality, we have

$$\|\nabla_{\mathcal{K}\eta}[f + g][x]\| - \|\nabla_{\mathcal{K}\eta} f[x]\| \leq \|\nabla_{\mathcal{K}\eta}[f + g][x] - \nabla_{\mathcal{K}\eta} f[x]\| = \frac{1}{\eta}\|u' - v'\| \underset{[a][b][4]}{\leq} \|\nabla g(x)\|$$

as required. □

As previously stated, the projected gradient is a natural analogue for the constrained case. Unfortunately, even in the offline case, it is too ambitious to converge towards a global minimizer in hindsight - so referring to it as ϵ-approximate will be quite enough. Also, for convenience, we will use the following notation to denote the *sliding-window time average* of functions f, parametrized by window size $1 \leq w \leq T$ and for simplicity of notation, define $f_t(x)$ to be identically zero for all $t \leq 0$.

$$F_{t,w}(x) := \frac{1}{w}\sum_{i=0}^{w-1}[f_{t-i}(x)] \tag{3}$$

In light of the computational intractability of direct analogues of convex regret, we define the *local regret* as used in the article:

Definition(2.5). *Fix some $\eta > 0$. Define w-**local regret** of an online algorithm as:*

$$\mathfrak{R}_w(T) \stackrel{def}{=} \sum_{t=1}^{T} \|\nabla_{\mathcal{K}\eta} F_{t,w}(x_t)\|^2 \tag{4}$$

notice - future use we won't write w in the equation - since it is trivial

Theorem(2.7). *Define $\mathcal{K} = (-1, 1)$. For any $T \geq 1$, $1 \leq w \leq T$, and $\eta \leq 1$, there exists a distribution D on 0-smooth, 1-bounded cost functions $f_1, \ldots, f_T$ on K such that for any online algorithm, when run on this sequence of functions,*

$$\mathop{\mathbb{E}}_{D}[\mathfrak{R}_w(T)] \geq \frac{1}{4w}\left\lfloor \frac{T}{2w} \right\rfloor \tag{5}$$

Proof. Let us partition the T rounds of play into $\left\lfloor \frac{T}{2w} \right\rfloor$ repeated segments, each of $2w$.
For the first half of the first segment, $t = 1, \ldots, w$, the adversary declares that

- **For odd** t, select $f_t(x)$ i.i.d. as:

$$f_t(x) := \begin{cases} -x & \text{with prob.} \frac{1}{2} \\ x & \text{with prob.} \frac{1}{2} \end{cases}$$

- **For even** t, select

$$f_t(x) := -f_{t-1}(x)$$

During the second half, $t = w + 1, \ldots, 2w$, the adversary sets all $f_t(x) = 0$. This construction is repeated $\left\lfloor \frac{T}{2w} \right\rfloor$ times, padding the finale $T \ mod \ 2w$ costs arbitrarily with $f_t(x) = 0$. Now, by this construction, at each round t at which $f_t(x) = 0$ is drawn randomly, we get

$$F_{t,w}(x) = \frac{1}{w}\sum_{i=0}^{w-1} f_{t-i}(x) \underset{even}{=} \frac{f_t(x)}{w}$$

From the *law of total expectation*, we have $\mathbb{E}(X) = \sum_j \mathbb{E}(X|A_j) \cdot p(A_j)$. Furthermore, for any x_t played by the algorithm, $\|\nabla_{\mathcal{K}\eta} f_t(x_t)\| = 1$ with probability at least $\frac{1}{2}$. So that

$$\begin{aligned} \mathbb{E}\left[\|\nabla_{\mathcal{K}\eta} F(x)\|^2\right] &= \mathbb{E}\left[\|\nabla_{\mathcal{K}\eta} F(x)\|^2 \;\middle|\; \|\nabla_{\mathcal{K}\eta} f(x)\| = 1\right]\frac{1}{2} + \overbrace{\mathbb{E}\left[\|\nabla_{\mathcal{K}\eta} F(x)\|^2 \;\middle|\; \|\nabla_{\mathcal{K}\eta} f(x)\| \neq 1\right]}^{\geq 0}\frac{1}{2} \\ &\geq \mathbb{E}\left[\|\nabla_{\mathcal{K}\eta} F(x)\|^2 \;\middle|\; \|\nabla_{\mathcal{K}\eta} f(x)\| = 1\right]\frac{1}{2} = \mathbb{E}\left[\frac{1}{w^2}\right]\frac{1}{2} = \frac{1}{2w^2} \end{aligned}$$

The claim now follows from the fact that there are at least $\frac{w}{2}$ of these rounds per segment, and exactly $\left\lfloor \frac{T}{2w} \right\rfloor$ segments in total. □

Let us further note that the notion of time-smoothing captures non-convex online optimization under limited concept drift: in online learning problems where $F_{t,w} \approx f_t(x)$, a bound on local regret truly captures a guarantee of *playing points* with small gradients. Notably, it means that there exists a distribution for any Online Algorithm. Up next we will see that for any distribution there exists an Online Algorithm with bounded local regret.

Appendix B :Main Proofs of the Theorems

B.1 Algorithm 1:

In this paragraph we will come to the conclusion that there exists an algorithm for every distribution that achieves the local regret bound $O(\frac{T}{w^2})$, by taking $O(Tw)$ iterations of the inner loop - as shown in the following theorem:

Theorem(3.1). *Let $f_1, \ldots, f_T$ be the sequence of loss functions presented to Algorithm 1, satisfying Assumption 2.1. Then:*

(i) The w-local regret incurred satisfies

$$\mathfrak{R}_w(T) \leq (\delta + 2L)^2 \frac{T}{w^2} \tag{6}$$

(ii) The total number of gradient steps τ taken by Algorithm 1 satisfies

$$\tau \leq \frac{M}{\delta^2(\eta - \frac{\beta\eta^2}{2})}(2Tw + w^2) \tag{7}$$

We will present next the complete proof of this theorem. When part(i) is direct from the assumption but for part(ii) we wont get away without a couple of helping lemmas.

Proof. (i) Note that Algorithm 1 will only play an iterate x_t if $\|\nabla_{\mathcal{K}\eta} F_{t-1,w}\| \leq \frac{\delta}{w}$ and that at $t = 1$, *$F_{t-1,w}$ is zero.* Now, let us denote

$$h_t(x) = \frac{1}{w}\Big(f_t(x) - f_{t-w}(x)\Big)$$

and since

$$\begin{aligned} h_t(x) - h_t(y) &= \frac{1}{w}\Big(f_t(x) - f_{t-w}(x)\Big) - \frac{1}{w}\Big(f_t(y) - f_{t-w}(y)\Big) \\ &= \frac{1}{w}\Big(f_t(x) - f_t(y) + f_{t-w}(y) - f_{t-w}(x)\Big) \underset{\substack{Asump. \\ 2.1(ii)}}{\leq} \frac{1}{w} \cdot 2L(x - y) \end{aligned}$$

h it is $\frac{2L}{w}$-Lipschitz. Then, for each $1 \leq t \leq T$ we have a bound on each cost

$$\begin{aligned} \|\nabla_{\mathcal{K}\eta} F_{t,w}(x_t)\|^2 &= \|\nabla_{\mathcal{K}\eta}\Big[F_{t-1,w} + h_t(x)\Big](x_t)\|^2 \\ &\underset{\substack{Prop. \\ 2.4}}{\leq} (\|\nabla_{\mathcal{K}\eta} F_{t-1,w}\| + \|\nabla h_t(x_t)\|)^2 \leq \left(\frac{\delta}{w} + \frac{2L}{w}\right)^2 = \frac{(\delta + 2L)^2}{w^2} \end{aligned}$$

Summing over all t gives the desired result. □

Now, let us briefly introduce the lemmas for the second part: Lemma(3.2) is actually like the known Cauchy–Schwarz inequality - showing us that the length of the projection is bigger than 1. Then we use this conclusion in the proof of Lemma(3.3). The algorithm only takes projected gradient steps when Summing across all t consecutive iterations in the epoch yields the claim.

Lemma(3.2). *Let $\mathcal{K} \in \mathbb{R}^n$ be a closed convex set, and let $\eta > 0$. Suppose $f : \mathcal{K} \to \mathbb{R}$ is differentiable. Then, for any $x \in \mathbb{R}$:*

$$\langle \nabla f(x), \nabla_{\mathcal{K}\eta} f(x) \rangle \geq \|\nabla_{\mathcal{K}\eta} f(x)\|^2 \tag{8}$$

Proof. Let $u = x - \eta \nabla f(x)$ and $u' = \Pi_{\mathcal{K}}[u]$. Then,

$$\frac{\langle \nabla f(x), \nabla_{\mathcal{K}\eta} f(x) \rangle}{\eta^2} = \frac{\|\nabla_{\mathcal{K}\eta} f(x)\|^2}{\eta^2} = \langle u - x, u' - x \rangle - \langle u' - x, u' - x \rangle = \overbrace{\langle u - x, u' - x \rangle}^{\geq 0}$$

The last inequality follows from the angular definition of the inner product

$$cos(\theta) \cdot \|u - x\| \cdot \|u' - x\| = \langle u - x, u' - x \rangle, \quad \theta \in \left[0, \frac{\pi}{2}\right]$$

□

For $2 \leq t \leq T$, let τ_t be the number of gradient steps taken in the outer loop at iteration $t-1$, in order to compute the iterate x_t. For convenience, define $\tau_1 = 0$. We establish a progress lemma during each gradient descent epoch:

Lemma(3.3). *For any* $2 \leq t \leq T$

$$F_{t-1}(x_t) - F_{t-1}(x_{t-1}) \leq -\tau_t \left(\eta - \frac{\beta \eta^2}{2}\right) \frac{\delta^2}{w^2} \tag{9}$$

Proof. Consider a single iterate z of the inner loop, and the next iterate

$$[1] \; z' := z - \eta \nabla_{\mathcal{K}\eta} F_{t-1}(z)$$

We have, by β-smoothness of F_{t-1},

$$F_{t-1}(z') - F_{t-1}(z) \underset{Descent-lemma}{\overset{\beta-smoothness}{\leq}} \langle \nabla F_{t-1}(z), z' - z \rangle + \frac{\beta}{2} \|z' - z\|^2$$

$$\underset{[1]}{=} -\eta \langle \nabla F_{t-1}(z), \nabla_{\mathcal{K}\eta} F_{t-1}(z) \rangle + \frac{\beta \eta^2}{2} \|\nabla_{\mathcal{K}\eta} F_{t-1}(z)\|^2 \underset{\substack{Lemma \\ (3.2)}}{\leq} -\left(\eta - \frac{\beta \eta^2}{2}\right) \|\nabla_{\mathcal{K}\eta} F_{t-1}(z)\|^2$$

The algorithm only takes projected gradient steps when $\|\nabla_{\mathcal{K}\eta} F_{t-1}(z)\| \geq \frac{\delta}{w}$.
Summing across all τ_t consecutive iterations in the epoch yields the claim. □

Now, let us go back to the second part of $3.1(ii)$ To complete the proof of the theorem, we write the telescopic sum* understanding $F_0(x_0) = 0$.

Proof. (3.1)(ii)

$$F_T(x_T) \underbrace{=}_{*} \sum_{t=1}^{T} \Big(F_t(x_t) - F_{t-1}(x_{t-1}) \Big) = \sum_{t=1}^{T} \Big(\underbrace{F_{t-1}(x_t) - F_{t-1}(x_{t-1})}_{F_{t-1}} + \underbrace{f_{t-1}(x_t) - f_{t-w}(x_t)}_{h_t} \Big)$$

$$\underset{\substack{Asamp. \\ (2.1)(i)}}{\leq} \sum_{t=2}^{T} \Big(F_{t-1}(x_t) - F_{t-1}(x_{t-1}) \Big) + \frac{2MT}{w} \underset{\substack{Lemma \\ (3.3)}}{\leq} \frac{2MT}{w} - \left(\eta - \frac{\beta \eta^2}{2}\right) \frac{\delta^2}{w^2} \cdot \underbrace{\sum_{t=1}^{T} \tau_t}_{\tau}$$

Then, from changing the sides and isolating τ we get exactly what we wanted. □

Setting $\eta = \frac{1}{\beta}$ and $\delta = L$ gives the asymptotically optimal local regret bound, with $O(Tw)$ time-averaged gradient steps and thus $O(Tw^2)$ individual gradient oracle calls. $[O(Tw + w^2) \underset{T \geq w}{=} O(Tw)]$

B.2 Algorithm 2:

In this section let us introduce the implications for offline and stochastic non-convex optimization and its correlation to online algorithms that incur small local regret in expectation. For convenience, for $1 \leq t \leq t' \leq T$, we denote by $D[t, t']$ *the uniform distribution on time steps* t *through* t' *inclusive.*

The next corollary is pretty similar to Prop.(2.6), but with a small adjustment - since all the small f's are similar, we use the next notation

$$F = \frac{1}{w} \cdot \sum_{i=0}^{w-1} f_{t-i} = f$$

and conclude the next:

Corollary(4.1). *Let* $f : \mathcal{K} \to \mathbb{R}$ *satisfy Assumption 2.1. When online algorithm* $\mathcal{A}$ *is run on a sequence of* T *identical loss functions* $f(x)$*, it holds that for any* $1 \leq w < T$

$$\mathop{\mathbb{E}}_{t \sim \mathcal{D}[w,T]} \left[\left\| \nabla_{\mathcal{K}\eta} f(x_t) \right\|^2 \right] \leq \frac{\mathfrak{R}_w(\mathcal{A})}{T - w} \tag{10}$$

In particular, Alg.1, with parameter choices $T = 2w,\ \eta = \frac{1}{\beta},\ \delta = L,\ w = (\delta + 2L)\sqrt{\frac{2}{\epsilon}}$ *yields*

$$\mathop{\mathbb{E}}_{t \sim \mathcal{D}[w,T]} \left[\left\| \nabla_{\mathcal{K}\eta} f(x_t) \right\|^2 \right] \leq \epsilon$$

Furthermore, the algorithm makes $O(\frac{1}{\epsilon})$ *calls to the gradient oracle in total.*

Now let us examine the offline stochastic case. From here on we will assume that we have a *noisy* stochastic oracle $\tilde{\nabla} f(x)$ and only focus on the constrained case where $\mathcal{K} = \mathbb{R}^n$ because it becomes challenging to work with projected gradient with *noisy* information. Also, the authors tried but in a futile attempt to show that this black-box reduction recovers an optimal convergence rate in terms of ϵ, but not σ^2. (Since $(\frac{\sigma^4}{\epsilon^2})$).

Proposition(4.3). *Let* $1 \leq w < T$ *. Suppose that online algorithm* $\mathcal{A}$ *is run on a sequence of* T *identical loss functions* $f(x)$ *satisfying Assumption 2.1, with identical stochastic gradient oracles satisfying Assumption 4.2. Sample* $t \sim \mathcal{D}[w, T]$ *Then, over the randomness of* t *and the oracles*

$$\mathbb{E} \left[\left\| \nabla f(x_t) \right\|^2 \right] \leq \frac{\mathbb{E} \left[\mathfrak{R}_w(\mathcal{A}) \right]}{T - w} \tag{11}$$

Proof.

$$\mathop{\mathbb{E}}_{t \sim \mathcal{D}[w,T]} \left[\left\| \nabla f(x_t) \right\|^2 \right] \leq \frac{1}{T - w} \sum_{t=1}^{T} \left\| \nabla f(x_t) \right\|^2 \leq \frac{\mathfrak{R}_w(\mathcal{A})}{T - w}$$

The claim follows by taking the expectation of both sides, over the randomness of the oracles. □

For a concrete online-to-stochastic reduction, we consider Algorithm 2, which exhibits such a bound on expected local regret. Now, Let us remind ourselves that the main goal was to do an *epsilon*-approximation to the gradient descent. And indeed, by using Proposition 4.3 together with Theorem 4.4 we induce exactly this conclusion in Corollary 4.5.

B.3 Algorithm 3:

An efficient algorithm with second-order guarantees. In this section we will begin by modifying Algorithm .1 to exploit to second order information, meaning the Hessian $\nabla^2 f_t(x)$. This will help us in better approximation of the first order of the critical points. We will do that by replacing the GDM with *cubic-regularized Newton*, CRN. Moreover, we will assume that we have an access to the values of f_t up to the second order derivative.

Let **MinEig**(A) be the minimum (eigenvalue, eigenvector) pair for matrix A . As is standard for offline second-order algorithms, we must add the following additional smoothness restriction:

Exactly as before, we show the convergence and oracle complexity properties on Theorem 5.2 - when the first can be proven from the previous assumptions, just as theorem 3.1(i). For the second part we will again show two helping lemmas.

Theorem(5.2). *Let Let $f_1, \ldots, f_T$ be the sequence of loss functions presented to Algorithm 3, satisfying Assumptions 2.1 and 5.1. Choose $\delta = \beta$. Then, for some constants C_1, C_2 in terms of M, L, β, L_2:*

(i) The iterates $\{x_t\}$ produced by Algorithm 3 satisfy

$$\sum_{t=1}^{T} \Phi_t(x_t) \leq C_1 \cdot \frac{T}{w^2} \tag{12}$$

(ii) The total number of iterations τ of produced by Algorithm 3 satisfy

$$\tau \leq C_2 \cdot \frac{T}{w^2} \tag{13}$$

Proof. The proof of the first part is quiet similar to $3.1(i)$ - the only difference is with changing the while statement condition to the new function Φ. Likewise, the proof of the second part is pretty similar to (3.1)(ii) using the lemmas below. □

Lemma(5.3). *Let z, z' be two consecutive iterates of the inner loop in Algorithm 3 during round t. Then,*

$$F_t(z') - F_t(z) \leq -\frac{\Phi_t(t)}{2\beta} \tag{14}$$

Proof. The idea is using the 2^{nd} order smoothness of F_t when we do inner product between $g := \nabla F_t(z)$ and $u = -\frac{g}{\beta}$. Then by the 2^{nd} order step so that $u = \pm\frac{2\lambda}{L_2}v$ (taking whichever sign makes $\langle g, u\rangle \leq 0$) and then by using the 3^{rd}-order smoothness of F_t - it implies exactly what we need; We notice that the lemma follows due to the fact that the algorithm takes the step that gives a smaller value of $F_t(z')$. □

Following the technique from Theorem 3.1, for $2 \leq t \leq T$, let τ_t be the number of iterations of the inner loop during the execution of Algorithm 3 during round $t-1$ (in order to generate the iterate x_t). Then, we have the following lemma:

Lemma(5.4). *For any $2 \leq t \leq T$,*

$$F_{t-1}(x_t) - F_{t-1}(x_{t-1}) \leq -\tau_t \cdot \frac{\delta^3}{2\beta w^3} \tag{15}$$

Proof. Similar to the proof of 3.3 □

B.4 Algorithm 4:

A solution concept for non-convex games. Finally, we will refer to discuss an application of our regret minimization framework to learning in k-player T-round iterated games with smooth, non-convex payoff functions. Now, suppose that each player $i \in [k]$ has a fixed decision set $\mathcal{K}_i \subseteq \mathbb{R}^n$, and a fixed payoff function $f_i : \mathcal{K} \to \mathbb{R}$ satisfies Assumption 2.1 as before. $\mathcal{K} = \mathcal{K}_1 \times \ldots \times \mathcal{K}_k$, the Cartesian product each payoff function is defined in terms of the choices made by every player.

Using the idea of time-smoothing, we formulate a tractable relaxed notion of local equilibrium, defined over some time window w. Intuitively, this definition captures a state of an iterated game in which each player examines the past w actions played, and no player can make small deviations to improve the average performance of her play against her opponents' historical play. Now, let us introduce the formal definition for *Smoothed local equilibrium*:

Definition(6.1). *Fix some $\eta > 0, w \geq 1$. Let $\left\{f_i(x^1, \ldots, x^k) : \mathcal{K} \to \mathbb{R}\right\}_{i=1}^{k}$ be the payoff functions for a k-player iterated game. A joint strategy $(x_t^1, \ldots, x_t^k)$ is an ϵ-approximate (η, w)* ***smoothed local equilibrium*** *with respect to past iterates $\left\{(x_{t-j}^1, \ldots, x_{t-j}^k)\right\}_{j=0}^{w-1}$ if, for every player $i \in [k]$,*

$$\left\| \nabla_{\mathcal{K}\eta} \left[\frac{\sum_{j=0}^{w-1} \tilde{f}_{i,t-j}}{w} \right] (x_t^i) \right\| \leq \epsilon \tag{16}$$

where

$$\tilde{f}_{i,t'}(x) \stackrel{def}{=} f_i(x_{t'}^1, \ldots, x_{t'}^{i-1}, x, x_{t'}^{i+1}, \ldots, x_{t'}^k) \tag{17}$$

To achieve such an equilibrium efficiently, we use Algorithm 4, which runs k copies of any online algorithm that achieves a w-local regret bound for some $\eta > 0$. In other words show that this meta-algorithm yields a sub-sequence of iterates that satisfy our solution concept, with error parameter dependent on the local regret guarantees of each player.

Let us introduce the next theorem (Similarly to 4.1)

Theorem(6.2). *For some t such that $w \leq t \leq T$, the joint strategy $(x_t^1, \ldots, x_t^k)$ produced by Algorithm 4 is an ϵ-approximate (η, w)-smoothed local equilibrium with respect to $\left\{(x_{t-j}^1, \ldots, x_{t-j}^k)\right\}_{j=0}^{t-1}$ where*

$$\epsilon = \sqrt{\sum_{i=1}^{k} \frac{\mathfrak{R}_{w,\mathcal{A}_i}}{T-w}} \tag{18}$$

Proof. Summing up the definitions of w-regret bounds achieved by each $\mathcal{A}$, and truncating the first $w-1$ terms, we get

$$\sum_{i=1}^{k} \sum_{t=w}^{T} \|\nabla_{\mathcal{K}\eta} F_t^i(x_t^i)\|^2 \leq \sum_{i=1}^{k} \mathfrak{R}_{w,\mathcal{A}_i}(T)$$

Now, from Col.(4.1) and since it 'works' for the expectation - we can conclude that there exists at least one such t between w and T inclusive, that holds:

$$\sum_{i=1}^{k} \left\| \nabla_{\mathcal{K}\eta} \left[\frac{\sum_{j=0}^{w-1} \tilde{f}_{i,t-j}}{w} \right] (x_t^i) \right\|^2 = \sum_{i=1}^{k} \|\nabla_{\mathcal{K}\eta} F_t^i(x_t^i)\|^2 \leq \sum_{i=1}^{k} \frac{\mathfrak{R}_{w,\mathcal{A}_i}(T)}{T-w} \tag{19}$$

Thus, for the same t we have:

$$\max_{1 \in [k]} \left\| \nabla_{\mathcal{K}\eta} \left[\frac{\sum_{j=0}^{w-1} \tilde{f}_{i,t-j}}{w} \right] (x_t^i) \right\| \underset{(*)+(18)}{\leq} \sqrt{\sum_{i=1}^{k} \frac{\mathfrak{R}_{w,\mathcal{A}_i}(T)}{T-w}}$$

as claimed. $(*)$: Let us notice that from metric spaces: $\|.\|_\infty \leq \|.\|_2$ □