CONSTRAINED OPTIMIZATION OF NONCONVEX PROBLEMS AND ITS APPLICATIONS IN FAIRNESS IN MACHINE LEARNING

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Zhuangjin Du

December 2021

© 2021 Zhuangjin Du
ALL RIGHTS RESERVED

ABSTRACT

Fairness is an important concern in machine learning. One way to characterize the problem is to minimize some objective function, representing a quantity like loss, under constraints that represent fairness conditions like demographic parity. Previous papers show how a two-player approach to optimizing the Lagrangian can be computationally efficient and provably converge to a good, constraint-satisfying solution on convex objective functions. This work analyzes the convergence of the two-player framework in [3] on nonconvex functions, in particular those that satisfy the PL inequality.

BIOGRAPHICAL SKETCH

Zhuangjin is a Master of Science student studying Computer Science at Cornell University. She completed her Bachelor's in Science from Cornell University majoring in Computer Science with a minor in mathematics in 2019.

ACKNOWLEDGEMENTS

My greatest thanks given to those who have helped with this work, in particular Karthik Sridharan, my advisor, and Danny Son, my colleague, who have made critical contributions to this work and have assisted me unfailingly throughout.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Machine learning has become ubiquitous in many modern systems, and as the impact of these systems grows, the problem of quantifying and ensuring fairness becomes more significant. Designing systems that can do these things are increasingly important to settings in which fairness is a critical concern; from evaluating job applicants to recommending products for online shopping. It is not always obvious how to evaluate or enforce fairness, and recent interest in the area has exploded.

One approach to this problem is to use constrained optimization. In general, this technique is useful for managing resources, scheduling [7], and for specific situations like maintaining fairness in machine learning. For fairness, many optimization problems are naturally restricted. It is common to optimize some objective function, such as utility or cost, based on parameters defining the model. These parameters typically operate under restrictions, such as those imposed by money and resources, but can also represent requirements on measures such as statistical parity.

One example of how constrained optimization can be useful is the 80% rule:

$$\begin{aligned} &\min_{\theta\in\Theta} \tfrac{1}{|S|}\textstyle\sum_{x,y\in S} \ell(f(x;\theta),y) \\ &\text{s.t. } \tfrac{1}{|S|}\textstyle\sum_{x\in S_{\min}} \mathbf{1}_{f(x;\theta)>0} \geq \tfrac{0.8}{|S|}\textstyle\sum_{x\in S} \mathbf{1}_{f(x;\theta)>0} \end{aligned}$$

In this application, both the objective and the constraint functions could be non-convex (such as for classifiers f learned by neural networks).

A common technique for approaching constrained optimization is to use the Lagrangian. This game-theoretic approach will be discussed in the Background sec-

tion.

1.1 Background

We introduce and motivate the use of a two-player games approach to this constrained optimization problem in the context of machine learning problems.

1.1.1 Constrained optimization

The following is a general formulation of constrained optimization:

$$\begin{aligned} &\min_{\theta \in \Theta} g_0(\theta) \\ &\text{s.t. } \forall i \in [m] \cdot g_i(\theta) \leq 0 \end{aligned}$$

In general, constrained optimization problems can be rewritten using the Lagrangian, where we minimize over $\theta \in \Theta$ and $\lambda \in \Lambda$:

$$\mathcal{L}(\theta, \lambda) := g_0(\theta) + \sum_{i=1}^{m} \lambda_i g_i(\theta) \tag{1.1}$$

This allows for a game-theoretic approach to the problem, in which the two players are θ and λ oppose each other; θ wants to be set to minimize the value of $\mathcal{L}(\theta, \lambda)$ as much as possible, and λ aims to maximize it. Intuitively, λ distributes itself onto the violated constraints, i where $g_i(\theta) > 0$, and θ aims to mitigate this by searching for $\theta \in \Theta$ such that the constraints are satisfied.

This approach can improve computational efficiency. Christiano et al. [**?**] showed that the (approximate) max flow problem can be solved in near-linear time using a Laplacian solver. $\tilde{O}\left(mn^{1/3}\epsilon^{-11/3}\right)$, beats traditional algorithms, fastest $O\left(n^{3/2}\right)$

However, unless we can say $\min_{\theta \in \Theta} \max_{\lambda \in \Lambda} \mathcal{L}(\theta, \lambda) = \max_{\lambda \in \Lambda} \min_{\theta \in \Theta} \mathcal{L}(\theta, \lambda)$, it

is possible that an algorithm will not converge to a solution in which neither player moves forward — a pure Nash equilibrium. This is extremely likely with nonconvex problems. Instead, a mixed Nash equilibrium may exist, in which θ and λ are probability distributions, and there does not exist a better *distribution* for eithier player to take such that they can improve the value of Equation 1.1.

In this work, we build off the framework introduced in [3] which offers a theoretical proof of a convergence to a solution that is both low in error and satisfies constraint bounds in the setting with a convex objective, to extend the proof to a broader class of nonconvex objective functions.

1.1.2 Modifying the Lagrangian for machine learning applications

Internal regret

In online learning, wherein an algorithm makes predictions over time and learns from it as it sees new examples, the goal is typically to minimize regret, or the cumulative loss generated over time.

The standard notion of regret is external regret, which represents the total loss compared to some fixed optimal predictor in the hypothesis class $\mathcal{H}$.

Definition 1. *Given an optimal predictor h^* such that*

$$h^* := \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{T} \sum_{t=1}^{T} \ell\left(h, (\mathbf{x}_t, y_t)\right)$$

define external regret to be

$$R(h) := \frac{1}{T}\sum_{t=1}^{T} \ell\left(h, (\mathbf{x}_t, y_t)\right) - \frac{1}{T}\sum_{t=1}^{T} \ell\left(h^*, (\mathbf{x}_t, y_t)\right) \tag{1.2}$$

Another type of regret, swap regret (also called internal regret) introduced in 1998 by Vohra and described in detail in [1], is used in the framework in [3] to show that their constrained optimization framework, Algorithm 2, leads to a feasible and optimal solution. We will define it here and sketch the result in a later section $\Phi-$correlated equilibrium below.

Definition 2. *Swap regret is defined as how much better a classifier h could have done, compared to some better h_s that does exactly what h did, but is allowed to switch every occurence of decision i for some different decision j made by h.*

$$R_s(h) := \frac{1}{T}\sum_{t=1}^{T} \ell\left(h, (\mathbf{x}_t, y_t)\right) - \frac{1}{T}\sum_{t=1}^{T} \ell\left(h_s, (\mathbf{x}_t, y_t)\right) \tag{1.3}$$

Another way to think of internal regret is to think of the online learning problem of predicting stocks; if every choice to purchase some stock A had instead been switched to the choice of purchasing a different stock B, how much could the total revenue have been? The swap regret is the difference between the actual history and the optimal history under these conditions.

Online optimization of internal regret can be achieved using the exponential weights algorithm [4]. However, this is an impractical step to play for the θ player, which is potentially as large as the number of parameters in a neural network, and is data-based.

Φ-correlated equilibrium

One critical change we make to the algorithm is to then look not for a mixed Nash equilibrium, but a Φ-correlated equilibrium.

Definition 3. Φ*-correlated equilibrium. equilibrium in which each player can remap their choices in the history to achieve minimum regret.*

We will show that aiming for a Φ-correlated equilibrium in which the θ-player minimizes its external regret and the λ-player minimizes its internal regret leads to a solution to the original optimization problem that is very close to satisfying

So far, what we have is an algorithm that repeats these two steps over each iteration:

1. *Take one step to update* $\theta^{(t)}$ *to minimize external regret.* Assuming convex g_0, θ can be updated with one (stochastic) gradient step.
2. *Take one step to update* $\lambda^{(t)}$ *to minimize internal regret.* λ can be updated with a an algorithm using exponential weights [4].

We can show that this algorithm that alternates online external regret minimization and internal regret minimization can achieve *near-feasibility* and *near-optimality* in expectation, as defined below:

Definition 4. *For the optimal feasible solution* θ^**, some small* $\epsilon > 0$*, and* R *a maximum 1-norm for the Lagrangian multipliers* λ_i*'s, we know that a solution* θ, λ *is* ***nearly-optimal*** *in expectation if:*

$$\mathbb{E}_\theta \left[g_0(\theta) \right] \leq g_0 \left(\theta^* \right) + \epsilon$$

Definition 5. *For the optimal feasible solution θ^*, some small $\epsilon > 0$, and R a maximum 1-norm for the Lagrangian multipliers λ_i's, we know that a solution θ, λ is* ***nearly-feasible*** *in expectation if:*

$$\max_{i \in [m]} E_\theta \left[g_i(\theta) \right] \leq \frac{\epsilon}{R - \|\lambda\|_1}$$

We want to show that an approximate Nash equilibrium does correspond to a nearly-optimal solution.

Proof. We examine a θ that satisfies an ϵ-Nash Equilibrium. In other words, θ satisfies the following:

$$\max_{\lambda^* \in \Lambda} \mathbb{E}_\theta \left[\mathcal{L} \left(\theta, \lambda^* \right) \right] - \inf_{\theta^* \in \Theta} \mathbb{E}_\lambda \left[\mathcal{L} \left(\theta^*, \lambda \right) \right] \leq \epsilon \tag{1.4}$$

Since $\mathcal{L}$ is linear in λ, we let

$$\max_{\lambda^* \in \Lambda} \mathbb{E}_\theta \left[\mathcal{L} \left(\theta, \lambda^* \right) \right] - \inf_{\theta^* \in \Theta} \mathcal{L} \left(\theta^*, \bar{\lambda} \right) \leq \epsilon$$

for $\bar{\lambda} = \mathbb{E}_\lambda[\lambda]$.

$$\max_{\lambda^* \in \Lambda} \mathbb{E}_\theta \left[\mathcal{L} \left(\theta, \lambda^* \right) \right] - \inf_{\theta^* \in \Theta} \mathcal{L} \left(\theta^*, \bar{\lambda} \right) \leq \epsilon \tag{1.5}$$

θ is nearly-optimal.

□

Likewise, for near-feasibility, we have:

Proof. By setting $\lambda^* = 0$, we see that

$$\mathbb{E}_\theta \left[g_0(\theta) \right] - g_0 \left(\theta^* \right) \leq \epsilon$$

θ is nearly-feasible. We let $\theta^* = \theta$, and rearrange:

$$\max_{\lambda^* \in \Lambda} \sum_{i=1}^{m} \lambda_i^* \mathbb{E}_\theta \left[g_i(\theta) \right] - \sum_{i=1}^{m} \bar{\lambda}_i \mathbb{E}_\theta \left[g_i(\theta) \right] \leq \epsilon$$

$$R \left\| \left(\mathbb{E}_\theta \left[g_:(\theta) \right] \right)_+ \right\|_q - \|\bar{\lambda}\|_p \left\| \left(\mathbb{E}_\theta \left[g_:(\theta) \right] \right)_+ \right\|_q \leq \epsilon$$

□

Algorithm 1 Two-player constrained optimization of non-convex Lagrangian

procedure PHICORRELATEDOPTIMIZATION($\mathcal{L} : \Theta \times \Delta^{m+1} \to \mathbb{R}, T \in \mathbb{N}$) ▷ $\mathcal{L}$ is the objective function, T is the number of time steps to run the algorithm.
  Initialize $\theta^{(1)} = 0$, and $M^{(1)} \in \mathbb{R}^{(m+1)\times(m+1)}$ with $M_{i,j} = 1/(m+1)$
  for $t \in T$ **do**
    Let $\theta^{(t)} =$ fix $M^{(t)}$ (*)
    Let $\Delta_\theta^{(t)}$ be the gradient of $\mathcal{L}(\theta^{(t)}, \lambda^{(t)})$ w.r.t. θ
    Let $\Delta_\lambda^{(t)}$ be the gradient of $\mathcal{L}(\theta^{(t)}, \lambda^{(t)})$ w.r.t. λ
    $\theta^{(t+1)} \leftarrow \Pi_\Theta \left(\theta^{(t)} - \eta_\theta \check{\Delta}_\theta^{(t)} \right)$ (**)
    Update $\tilde{M}^{(t+1)} = M^{(t)} \odot \cdot \exp\left(\eta_\lambda \Delta_\lambda^{(t)} \left(\lambda^{(t)} \right)^T \right)$
    Update $M_{:,i}^{(t+1)} = \tilde{M}_{:,i}^{(t+1)} / \left\| \tilde{M}_{:,i}^{(t+1)} \right\|$ for $i \in [m+1]$
  return $\theta^{(1)}, \cdots, \theta^{(T)}$ and $\lambda^{(1)}, \cdots, \lambda^{(T)}$
  ▷ (*) This fix operation retrieves the eigenvalue associated with the top eigenvector of M.
  ▷ (**) Π_Θ projects onto Θ w.r.t. the Euclidian norm.

Using SGD

This algorithm introduced in [3] uses stochastic first-order methods like SGD on the Lagrangian directly, while further analysis in this work will use GD. In [3], they show that re-framing as a 2-player game allows using SGD during the θ player's step, which converges to a nearly-optimal nearly-feasible solution. the λ-player is updated using exponential weights, an external regret-minimizing algorithm. SGD is computationally much more efficient than evaluating a gradient in GD, but we will not be discussing the use of SGD with the framework in section 2.2.

CHAPTER 2

CONSTRAINED OPTIMIZATION ON A PL OBJECTIVE

It may be more productive to work with a narrower range of objective functions than all nonconvex functions to achieve good rates for the progression of the two-player algorithm. One such set of functions is the PL inequality.

2.1 PL Inequality

By assuming stricter properties of the Lagrangian, we can prescribe an algorithm that finds a near-optimal θ per iteration. If the iteration of the λ player — which still maintains the feasibility of the constraints — manages to allow the Lagrangian to preserve this property, the iterative proof applies.

Definition 6. *A function $f : \mathbb{R}^d \to \mathbb{R}^+$ satisfies the Polyak-Łojasiewicz (PL) inequality if it satisfies*

$$f(w) - f(w^*) \leq \mu \|\nabla f(w)^2\| \tag{2.1}$$

for some real $\mu > 0$.

This definition was established in 1963 [9], and some functions that also satisfy this condition include one-point convexity, star convexity, τ-star convexity.

We also know that all critical points are global by this inequality.

2.2 Regret Minimization

2.3 Intuition

We can then make the assumptions that

1. The Lagrangian satisfies the proximal-PL condition, and that
2. The minimizer for f_{t-1} is *not far* from the minimizer for f_t (to be characterized below).

The intuition is that, while we know the λ player changes the Lagrangian each turn, that the minimizer θ for a given time step is close to the minimizer for the next time step as well. This means the minimzer at time t is a good first guess for the minimzer at step $t+1$. With a warm start for f_1, we see that not many iterations are required to reach the objective. Knowing this, Algorithm 2 can minimize a nonconvex objective function while dealing with the Lagrangian changing after the λ-player update.

2.3.1 Algorithm and Proof

Algorithm 2 Regret Minimization for θ player

function RegretMinimizer(T, θ^0, ϵ)
  Set $M \geq f_1(\theta^{(0)}) - f_1(\theta_1^*)$
  $\tau_1 = \frac{\beta}{\mu} \cdot \log \frac{M}{\epsilon}$
  $\theta^{(1)} = OGD\left(f_1, \theta^{(0)}, \tau_1\right)$
  for $1 \leq t \leq T$ **do**
    Set $\tau_t = \frac{\beta}{\mu} \log \frac{2C_t + \epsilon}{\epsilon}$
    Set $\theta^{(t)} = OGD\left(f_t, \theta^{(t-1)}, \tau_t\right)$
  return $\theta^1, \cdots, \theta^T$

Algorithm 3 Online Gradient Descent

function OGD($f, \theta^{(0)}, K$)
  for $1 \leq k \leq K$ **do**
    Set $\theta^{(t)} = \prod_{\Theta}\left(\theta^{(t-1)} - \eta \nabla f\left(\theta^{(t-1)}\right)\right)$ ▷ Projected GD update
  return $\theta^1, \cdots, \theta^T$

Formally, we will show the main result in Theorem 2.3.1.

Theorem 2.3.1 (Main Theorem)**.** *Suppose $f_1, \cdots f_T : \Theta \to \mathbb{R}$ satisfy the PL condition and are $\beta-$smooth. Let θ_*^T be the minimizer of f_t. Furthermore, assume that $\|f_t(\theta) - f_{t-1}(\theta)\| \leq C_t = O(T^{-1/2})$. Then, running Algorithm 2 produces iterates $\theta^1, \cdots, \theta^T$ that satisfy*

$$\frac{1}{T}\sum_{t=1}^{T} f_t\left(\theta^t\right) - f_t\left(\theta_*^t\right) \leq \epsilon$$

where the number of calls to the subprocess OGD is at most

$$\frac{\beta}{\mu}\left\{\log\frac{M}{\epsilon} + T\log\left(\frac{2C}{\epsilon\sqrt{T}} + 1\right)\right\}$$

where C is the constant that satisfies $\frac{C}{\sqrt{T}} = \max_{t\in[1..T]} C_t$. Now, by setting $\epsilon = T^{-1/2}$, we can bound the number of OGD updates to at most

$$O\left(\frac{\beta}{\mu}\{\log((f_1(\theta^0) - f_1(\theta_1^*)\sqrt{T}) + T\log C\}\right)$$

We first establish a progress lemma on the θ-player. We can do this by exploiting the fact that running Gradient Descent (GD) on the objective will always reduce it. For reference, we will use the update in (2.2) to denote a GD update for some step size η,

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla f(\theta^{(t)}) \tag{2.2}$$

and the update in (2.3) to denote a projected GD update for step size η. We define Π_Θ to project its argument onto some parameter space Θ with respect to the Euclidean norm.

$$\theta^{(t+1)} \leftarrow \Pi_\Theta\left(\theta^{(t)} - \eta \nabla f(\theta^{(t)})\right) \tag{2.3}$$

Lemma 2.3.2 (Progress Lemma)**.** *Suppose f satisfies the PL condition and is $\beta-$smooth. Let θ^* bet the minimzer of f. Then, performing one GD update gets us*

$$f(\theta^{(t+1)}) - f(\theta^*) \leq -(\eta - \beta/2 \cdot \eta^2)\|\nabla f(\theta^{(t)})\|^2$$

Setting step size $\eta = \frac{1}{\beta}$ gets

$$f(\theta^{(t+1)}) - f(\theta^*) \leq -\frac{1}{2\beta}\|\nabla f(\theta^{(t)})\|^2 \tag{2.4}$$

Proof. This follows from the β-smoothness of f.

$$\begin{aligned} f(\theta^{(t+1)}) - f(\theta^*) &\leq \langle \nabla f(\theta^{(t)}), \theta^{(t+1)} - \theta^{(t)} \rangle + \frac{\beta}{2}\|\nabla\theta^{(t+1)} - \theta^{(t)}\|^2 \\ &\leq -\eta\|\nabla f(\theta^{(t)})\|^2 + \frac{\beta}{2}\eta^2\|\nabla f(\theta^{(t)})\|^2 \\ &= -(\eta - \beta/2 \cdot \eta^2)\|\nabla f(\theta^{(t)})\|^2 \end{aligned}$$

□

Lemma 2.3.3 (Progress Lemma — constrained)**.** *Suppose $F(\theta) = f(\theta) + g(\theta)$ where f is differentiable and β-smooth, and g is the indicator function that takes the value 0 on some parameter space θ, and ∞ elsewhere. We can say that a function F satisfies the Proximal-PL condition ([6]) if there exists $\mu > 0$ such that*

$$\frac{1}{2}D_g(\theta, \beta) \geq \mu(F(\theta) - F(\theta^*)) \tag{2.5}$$

where

$$D_g(\theta, \beta) = -2\beta \cdot \min_y \left\{ \langle \nabla f(\theta), y - \theta \rangle + \frac{\beta}{2}\|y - \theta\|^2 + g(y) - g(\theta) \right\} \tag{2.6}$$

If Θ is closed and convex, and F satisfies the Proximal-PL condition, then performing one GD update gets us

$$f(\theta^{(t+1)}) - f(\theta^*) \leq \frac{1}{2\beta} D_g(\theta, \beta) \tag{2.7}$$

when we set $\eta = \frac{1}{\beta}$.

Proof. First, we observe that

$$\begin{aligned}
\theta^{(t+1)} &= \Pi_\Theta \left(\theta^{(t)} - \eta \cdot \nabla f(\theta)\right) \\
&= \operatorname*{argmin}_{y \in \Theta} \|y - \theta^{(t+1)}\|^2 \\
&= \operatorname*{argmin}_{y} \|y - (\theta^{(t)} - \eta \cdot \nabla f(\theta^{(t)}))\|^2 + g(y) \\
&= \operatorname*{argmin}_{y} \left\{ \langle f(\theta^{(t)}), y - \theta^{(t)} \rangle + \frac{1}{2\eta} \|y - \theta^{(t)}\|^2 + g(y) - g(\theta^{(t)}) \right\}
\end{aligned}$$

We then use the β-smoothness of f again and set $\eta = \frac{1}{\beta}$, and use the definition of Proximal-PL from Equation (2.6):

$$\begin{aligned}
f(\theta^{(t+1)}) - f(\theta^*) &\leq \langle f(\theta^{(t)}), \theta^{(t+1)} - \theta^{(t)} \rangle + \frac{\beta}{2} \|\theta^{(t+1)} - \theta^{(t)}\|^2 \\
&= \min_{y} \left\{ \langle f(\theta^{(t)}), y - \theta^{(t)} \rangle + \frac{\beta}{2} \|y - \theta^{(t)}\|^2 + g(y) - g(\theta) \right\} \\
&= -\frac{1}{2\beta} D_g(\theta, \beta)
\end{aligned}$$

□

We now look at the sequence of functions that the θ-player aims to minimize regret over.

Theorem 2.3.4. *Suppose* $f_1, f_2, \cdots, f_T : \Theta \to \mathbb{R}$ *satisfy the PL condition and are* β*-smooth. Let* $\theta_*^{(t)}$ *be the minimizer of* f_t*. Furthermore, assume that* $\|f_t(\theta) - f_{t-1}(\theta)\| \leq C_t = O(T^{-1/2})$.

Then, running Algorithm 2 produces iterates $\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(T)}$ that satisfy equation 2.8 for some small $\epsilon \geq 0$:

$$\sum_{t=1}^{T} f_t(\theta^{(t)}) - f_t(\theta_*^{(t)}) \leq \epsilon \tag{2.8}$$

Lemma 2.3.5. *Suppose $\theta^{(t-1)}$ satisfies*

$$f_{(t-1)}(\theta^{(t-1)}) - f_{(t-1)}(\theta_*^{(t-1)}) \leq \epsilon \tag{2.9}$$

Then, we have

$$f_{(t-1)}(\theta^{(t-1)}) - f_{(t-1)}(\theta_*^{(t)}) \leq 2C_t\epsilon \tag{2.10}$$

In other words, we know $\theta^{(t-1)}$ serves as a good initial guess for the minimizer of f_t.

Proof. We make use of these three inequalities:

$$f_t(\theta^{(t-1)}) - f_{t-1}(\theta^{(t-1)}) \leq C_t \tag{2.11}$$

$$f_{t-1}(\theta^{(t-1)}) - f_{t-1}(\theta_*^{(t-1)}) \leq \epsilon \tag{2.12}$$

$$f_{t-1}(\theta_*^{(t-1)}) - f_t(\theta_*^{(t)}) \leq C_t \tag{2.13}$$

Equations (2.11) and (2.12) follow from the assumptions we have made. (2.13) is a result of $\theta_*^{(t-1)}$ being the minimizer of f_{t-1}, which means $f_{t-1}(\theta^{(t-1)}) \leq f_{t-1}(\theta_*^{(t)})$. Then, we know

$$f_{t-1}(\theta_*^{(t-1)}) - f_t(\theta_*^{(t)}) \leq f_{t-1}(\theta_*^{(t)}) - f_t(\theta_*^{(t)}) \leq C_t$$

We add equations (2.11) through (2.13) to get the desired result. □

This lemma is useful because it implies that the suboptimality of our initial guess at time t (which is the optimizer from time $t-1$) is not far away from the optimum at time t.

Lemma 2.3.6. *Suppose we run Algorithm 3 on $f, \theta^{(0)}, K$ where f satisfies the PL condition for a constant $\mu > 0$, $\theta^{(0)}$ is some initial guess, and K is the number of iterations. Define $\Delta_t = f(\theta^{(t)}) - f(\theta^*)$. Let M be any upper bound on Δ_0. In other words, M is an upper bound on the initial suboptimality. Then, if $K \geq \frac{\beta}{\mu}$, we have*

$$f(\theta^{(K)} - f(\theta^*) \leq \epsilon \tag{2.14}$$

for some small ϵ, where $\theta^{(K)}$ is the output of Algorithm 3.

Proof. We will make use of the inequality below.

$$\Delta^j \leq (1 - \frac{\mu}{\beta})\Delta^{j-1} \tag{2.15}$$

A proof for equation (2.15) can be found in the proof of Theorem 1 in [6]; we use the recursive form of the inequality. By applying 2.15 K times and using the fact that $1 - x \leq e^{-x}$, we can get

$$\Delta^K \leq \left(1 - \frac{\mu}{\beta}\right)^K \Delta^0 \leq e^{-K(\mu/\beta)}\Delta^0 \leq e^{-K(\mu/\beta)}M \tag{2.16}$$

Now, we want to solve for the K that satisfies

$$e^{-K(\mu/\beta)}M \leq \epsilon \tag{2.17}$$

Rearranging this inequality, we can see that

$$K \geq \frac{\beta}{\mu} \log \frac{M}{\epsilon} \tag{2.18}$$

□

Proof. Now, we return to the main theorem. In Algorithm 2, we set τ_t to be the number of OGD updates (calls to Algorithm 3) we perform to produce $\theta^{(t)}$.

Recall that we set $\tau_1 = \frac{\beta}{\mu} \log \frac{M}{\epsilon}$, where M is the upper bound on $f_1\left(\theta^{(0)}\right) - f_1\left(\theta_*\right)$. Then, by Lemma 2.3.6, we know

$$f_1\left(\theta^{(1)}\right) - f_1\left(\theta_*^{(1)}\right) \leq \epsilon \tag{2.19}$$

for some small ϵ.

Now for $2 \leq t \leq T$, we recall that in Algorithm 2, we set $\tau_t = \frac{\beta}{\mu} \frac{2C_t + \epsilon}{\epsilon}$. By Lemma 2.3.5, we know that

$$f_t\left(\theta^{(t-1)}\right) - f_t\left(\theta_*^{(t)}\right) \leq 2C_t + \epsilon \tag{2.20}$$

Then, in every iteration for $t \geq 2$ of Algorithm 2, we can use the update $\theta^{(t)} = OGD(f_t, \theta^{(t-1)}, \tau_t$. Therefore, by Lemma 2.3.6, we have

$$f_t(\theta^{(t)}) - f_t(\theta_*^{(t)}) \leq \epsilon \tag{2.21}$$

for all $t \in [T]$. Therefore,

$$\frac{1}{T} \sum_{t=1}^{T} f_t(\theta^{(t)}) - f_t(\theta_*^{(t)}) \leq \epsilon' \tag{2.22}$$

for some small ϵ'.

Now, it remains to find the upper bound to the number of OGD updates (i.e. $\sum_{t \in [T]} \tau_t$).

We have the following:

$$\sum_{t=1}^{T} \tau_t = \frac{\beta}{\mu} \log \frac{M}{\epsilon} + \sum_{t=2}^{T} \frac{\beta}{\mu} \log \frac{2C_t + \epsilon}{\epsilon} \tag{2.23}$$

We examine the second term on the RHS. Recall that $C_t = O(T^{-1/2})$, and that C

is the constant such that $\max_{t\in[T]} C_t \leq CT^{-1/2}$. Then,

$$\begin{aligned}
\sum_{t=2}^{T} \frac{\beta}{\mu} \log \frac{2C_t + \epsilon}{\epsilon} &= \frac{\beta}{\mu} \sum_{t=2}^{T} \log \frac{2C_t + \epsilon}{\epsilon} \\
&= \frac{\beta}{\mu} \log \Pi_{t=2}^{T} \frac{2C_t + \epsilon}{\epsilon} \\
&\leq \frac{\beta}{\mu} \log \Pi_{t=2}^{T} \frac{2CT^{-1/2} + \epsilon}{\epsilon} \\
&=\leq \frac{\beta}{\mu} \log \Pi_{t=2}^{T} \frac{2C}{\epsilon\sqrt{T}} + 1 \\
&\leq \frac{\beta}{\mu} \log (\frac{2C}{\epsilon\sqrt{T}} + 1)^T \\
&\leq \frac{\beta}{\mu} T \log (\frac{2C}{\epsilon\sqrt{T}} + 1)
\end{aligned}$$

By setting $\epsilon = T^{-1/2}$, we have that the number of OGD updates is at most

$$\frac{\beta}{\mu} \left\{ \log \frac{M}{\epsilon} + T \log \left(\frac{2C}{\epsilon\sqrt{T}} + 1 \right) \right\}$$

□

Additional notes

It is important for the computational time of the algorithm to bound the number of calls to Algorithm 3, or OGD, because it can be costly to evaluate the gradient of a data-driven function.

2.4 Applications

In general, the PL inequality is useful as a condition that does not require convexity, and is a tool usable to show linear convergence rates [9], [2]. In particular, [2] shows

that it can be used to analyze several different machine learning situations. For example, strongly convex functions composed with piecewise linear functions, like (leaky) ReLU, satisfy the condition.

It is also potentially interesting to look at the class of "definable functions" which can be shown to include problems of interest for this method.

Definition 7. *An o-minimal structure is a collection* $\mathcal{S} = \{\mathcal{S}_n\}_{n=1}^{\infty}$*, where* $\mathcal{S}_n$ *is a set of subsets of* $\mathbb{R}^n$ *which includes all algebraic sets and is closed under ifnite union/intersection and complement, Cartesian product, and projection, and* $\mathcal{S}_1$ *consists of finite unions of open intervals and points.*

Definition 8. *A function* $f : D \to \mathbb{R}^m$ *with* $D \subset \mathbb{R}^n$ *is definable if its graph is in* $S_n + m$*.*

We also note from [5] that for any definable functions $f, g : D \to R$, any linear combination $\alpha f + \beta g$ and fg is definable.

Recall from our definition of the Lagrangian formulation in Equation 1.1 that we are interested in the sum of an objective function and several constraint functions.

Assuming that constraint functions $[g_i]$ for $1 \leq i \leq m$ are convex and definable, we can see that given a definable objective function we will have a definable Lagrangian. Indeed, [5] analyzes arbitrarily deep neural networks (which are highly convex) with common types of layers like linear, convolution, ReLU, and max-pooling satisfy definability.

Kurdyka shows that functions that satisfy the Łojasiewicz inequality are also definable in some o-minimal structure [8]. Paraphrased, the main result of the paper is that functions f that are differentiable and on an open and bounded subset of

$\mathbb{R}^n$ such that $f(x) > 0$, must satisfy

$$\| \operatorname{grad}(\Psi \circ f)(x)\| \geq c \tag{2.24}$$

for some increasing positive function Ψ and $c > 0, \rho > 0$.

CHAPTER 3

CONCLUSION

We show a proof of convergence to an optimal and feasible solution to a constrained optimization problem under specific conditions. A natural direction following this would be to explore a broader class of problems and to weaken the conditions required for the algorithm to apply.

Additionally, among other things, [3] presents a method to find a distribution over at most $m + 1$ distinct values of θ, where m is the number of constraints, and a single deterministic λ. This requires a decoupling of the two players, in which they aim to optimize two different Lagrangian functions $\mathcal{L}_\theta$ and $\mathcal{L}_\lambda$.

BIBLIOGRAPHY

[1] A. Blum and Y. Mansour. From external to internal regret. *J. Mach. Learn. Res.*, 8:1307–1324, 2007.

[2] Zachary Charles and Dimitris Papailiopoulos. Stability and generalization of learning algorithms that converge to global optima. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 745–754. PMLR, 10–15 Jul 2018.

[3] Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. Two-player games for efficient non-convex constrained optimization. In Aurélien Garivier and Satyen Kale, editors, *Proceedings of the 30th International Conference on Algorithmic Learning Theory*, volume 98 of *Proceedings of Machine Learning Research*, pages 300–332, Chicago, Illinois, 22–24 Mar 2019. PMLR.

[4] Geoffrey J. Gordon, Amy Greenwald, and Casey Marks. No-regret learning in convex games. ICML '08, page 360–367, New York, NY, USA, 2008. Association for Computing Machinery.

[5] Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning, 2020.

[6] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. *CoRR*, abs/1608.04636, 2016.

[7] James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. *CoRR*, abs/2103.16378, 2021.

[8] K. Kurdyka. On gradients of functions definable in o-minimal structures. *Annales de l'Institut Fourier*, 48:769–783, 1998.

[9] B. Polyak. Gradient methods for the minimisation of functionals. *Ussr Computational Mathematics and Mathematical Physics*, 3:864–878, 1963.