From Logistic Regression to the Perceptron Algorithm: Exploring Gradient Descent with Large Step Sizes

Alexander Tyurin
AIRI, Moscow, Russia
Skoltech, Moscow, Russia
alexandertiurin@gmail.com

Abstract

We focus on the classification problem with a separable dataset, one of the most important and classical problems from machine learning. The standard approach to this task is logistic regression with gradient descent (LR+GD). Recent studies have observed that LR+GD can find a solution with arbitrarily large step sizes, defying conventional optimization theory. Our work investigates this phenomenon and makes three interconnected key observations about LR+GD with large step sizes. First, we find a remarkably simple explanation of why LR+GD with large step sizes solves the classification problem: LR+GD reduces to a batch version of the celebrated perceptron algorithm when the step size tends to infinity. Second, we observe that larger step sizes lead LR+GD to higher logistic losses when it tends to the perceptron algorithm, but larger step sizes also lead to faster convergence to a solution for the classification problem, meaning that logistic loss is an unreliable metric of the proximity to a solution. Surprisingly, high loss values can actually indicate faster convergence. Third, since the convergence rate in terms of loss function values of LR+GD is unreliable, we examine the iteration complexity required by LR+GD with large step sizes to solve the classification problem and prove that this complexity is suboptimal. To address this, we propose a new method, Normalized LR+GD—based on the connection between LR+GD and the perceptron algorithm—with much better theoretical guarantees.

1 Introduction

We consider the classical classification problem from machine learning with a dataset $\{(a_i, y_i)\}_{i=1}^n$ and two classes, where $a_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$ for all $i \in [n] := \{1, \ldots, n\}$. The goal of the classification problem is to

$$\text{find a vector } \theta \in \mathbb{R}^d \text{ such that } y_i a_i^\top \theta > 0 \quad \forall i \in [n]. \quad (1)$$

This is the supervised learning problem that finds a linear model (hyperplane) that separates the dataset. In general, this problem is infeasible, and one can easily find an example when the dataset is not linearly separable. We focus on the setup where the data is separable, which is formalized by the assumption:

Assumption 1.1.

$$\mu := \max_{\|\theta\|=1} \min_{i \in [n]} y_i a_i^\top \theta > 0.$$

This condition ensures that for some $\theta \in \mathbb{R}^d$, the dataset can be perfectly classified. The quantity μ is a *margin* (Novikoff 1962; Duda, Hart, and G.Stork 2001), which characterizes the distance between the two classes. This assumption is practical in modern machine learning problems (Soudry et al. 2018; Ji and Telgarsky 2018). Albeit it is mostly attributed to large-scale nonlinear models (Brown et al. 2020), where the number of parameters is huge, the analysis of the methods in the linear case is equally important as it serves as a foundation for the nonlinear case. Let us define $R := \max_{i \in [n]} \|a_i\|$.

There is a huge number of ways (Bishop and Nasrabadi 2006) how one can solve the problem, including support vector machines (SVMs) (Cortes and Vapnik 1995), logistic regression, and the perceptron algorithm (Novikoff 1962). This work focuses on the latter two, starting with logistic regression, which can be formalized by the following optimization problem:

$$f(\theta) := \frac{1}{n} \sum_{i=1}^n \log\left(1 + \exp(-y_i a_i^\top \theta)\right) \to \min_{\theta \in \mathbb{R}^d}. \quad (2)$$

This optimization problem does not have a finite minimum when the data is separable. Indeed, if θ separates the dataset, then $y_i a_i^\top \theta > 0$ for all $i \in [n]$ and $f(c \cdot \theta) \to 0$, when $c \to \infty$.

Gradient Descent

The logistic regression problem (2) can be solved with gradient descent (GD) (Nesterov 2018), stochastic gradient descent (Robbins and Monro 1951), L-BFGS (Liu and Nocedal 1989), and variance-reduced methods (e.g., SAG, SVRG) (Schmidt, Le Roux, and Bach 2017; Johnson and Zhang 2013). We consider the GD method, arguably one of the simplest and most well-understood methods:[1]

$$\theta_{t+1} = \theta_t - \gamma \nabla f(\theta_t), \quad \text{(LR+GD)}$$

[1] In this paper, we examine five algorithms, referred to as LR+GD, Batch Perceptron, Perceptron, Normalized Batch Perceptron, and Normalized LR+GD.

where θ_0 is a starting point, $\gamma > 0$ is a step size, and $\nabla f(\theta_t)$ is the gradient of (2) at the point θ_t.

What do we know about GD in the context of logistic regression (LR+GD)? Surprisingly, despite the huge popularity of GD and logistic regression, we still lack a comprehensive understanding.

Previous Work

Classical convex and nonconvex optimization theory. Let us recall the classical result for GD: it is well-known that if $\gamma < 2/L$, and a function f is L–smooth and lower bounded, which is true for (2), then[2] $f(\theta_T) - \inf_{\theta \in \mathbb{R}^d} f(\theta) = \widetilde{\mathcal{O}}\left(1/\gamma T\right)$ (Ji and Telgarsky 2018) for convex problems, or $\min_{t \in [T]} \|\nabla f(\theta_t)\|^2 \leq \mathcal{O}\left(1/\gamma T\right)$ for nonconvex problems. At the same time, if $\gamma > 2/L$, then one can find a L–smooth function such that GD diverges (see (Cohen et al. 2021, Sec.2)). Under L–smothness, the value $2/L$ is special because it divides GD into the *convergence* and *divergence* regimes.

The edge of stability (EoS) and large step sizes. Nonetheless, in practice, it was many times observed (e.g., (Lewkowycz et al. 2020; Cohen et al. 2021)) that when a step size is large, $\gamma > 2/L$, GD not only not diverges, but non-monotonically with oscillation converges on the task (2). This phenomenon was coined as *the edge of stability* (Cohen et al. 2021). This means that there is something special about the practical machine learning problems.

The mathematical aspects of the large step size regime have attracted significant attention within the research community, which analyzes the phenomenon through the sharpness of loss functions (the largest eigenvalue of Hessians) (Kreisler et al. 2023), small dimension problems (Zhu et al. 2022; Chen and Bruna 2022; Ahn et al. 2024), bifurcation theory (Song and Yun 2023), sharpness behavior in networks with normalization (Lyu, Li, and Arora 2022), 2-layer linear diagonal networks (Even et al. 2023), non-separable data (Ji and Telgarsky 2018; Meng et al. 2024), self-stabilization (Damian, Nichani, and Lee 2023; Ahn, Zhang, and Sra 2022; Ma et al. 2022; Wang, Li, and Li 2022). The papers by Wu, Braverman, and Lee (2024); Wu et al. (2024) are the closest to our research since they also analyze GD and logistic regression (LR+GD). They demonstrate that GD can converge with an arbitrary step size $\gamma > 0$.

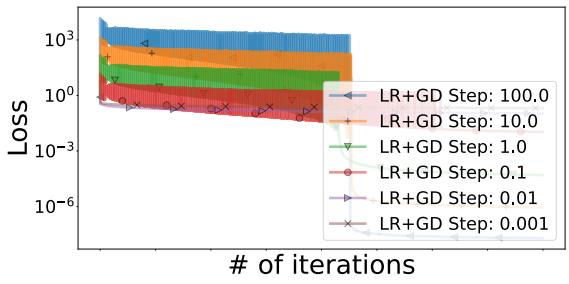
In particular, the results from (Wu, Braverman, and Lee 2024) show that for any fixed $\gamma > 0$, $f(\theta_t)$ is approximately less than or equal to $\text{poly}(e^{\gamma})/t$ (plus additional terms that depend on other parameters). Wu et al. (2024) refined the dependence on γ and demonstrated that GD with a large step size initially operates in *the non-stable regime*, where $f(\theta_t) = \widetilde{\mathcal{O}}((1+\gamma^2)/\gamma t)$. After approximately $\widetilde{\Theta}(\max\{n,\gamma\}/\mu^2)$ iterations, GD transitions to *the stable regime*, where $f(\theta_t) = \widetilde{\mathcal{O}}(1/\gamma t)$. By tuning and taking the fixed step size $\gamma = \Theta(T)$, they get the accelerated rate $f(\theta_T) = \widetilde{\mathcal{O}}(1/T^2)$.

[2]Note that we can not directly apply the results, for instance, from (Nesterov 2018) because (2) does not have a *finite* minimum. We need a minor modification of the classical analysis (Orabona 2024; Ji and Telgarsky 2018).

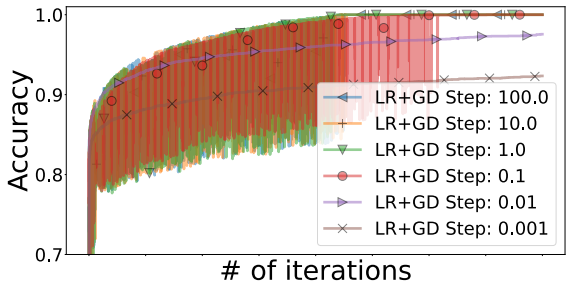
2 Contributions

This paper delves deeper into understanding the dynamics of *the non-stable regime*, where the loss chaotically oscillates due to large step sizes. We explore logistic regression with gradient descent (LR+GD) and find that **1)** LR+GD reduces to a batch version of the perceptron algorithm (Batch Perceptron), **2)** the fastest convergence of LR+GD to a solution of (1) is achieved when step sizes and *loss values* are large, and **3)** LR+GD is a suboptimal method, does not scale with the number of data points, and can be improved. Let us clarify:

1) We begin with a key observation that the iterates of LR+GD, when divided by the step size γ, converge to the iterates of a batch version (Batch Perceptron) of the celebrated perceptron algorithm (Block 1962; Novikoff 1962) when $\gamma \to \infty$. In other words, LR+GD reduces to Batch Perceptron. The proof of this fact is straightforward and occupies less than half a page. This is an advantage of our paper because the typical proofs on this topic are technical and non-intuitive. When combined with the classical convergence results (Novikoff 1962), it offers a clear intuition and explanation for why the method solves (1) with large step sizes in *the non-stable regime*—a detail that, to our knowledge, has been previously overlooked and nonproven in the literature. (see Section 3)



(a)



(b)

Figure 1: Illustration on a subset of *CIFAR-10* dataset (Krizhevsky, Hinton et al. 2009) with 5 000 samples and two classes. We run LR+GD with various step sizes. Note that there is no randomness involved in the process. Oscillation is a natural behavior of LR+GD with separable data and large step sizes.

When we test this fact, Theorem 3.2, with numerical experiments (see Figure 1), we observe that larger step sizes result in higher loss values (Figure 1a) before the moment when LR+GD attains Accuracy=1.0. Additionally, both loss and accuracy oscillate more (Figure 1a and 1b). And despite that, LR+GD solves (1) faster with large step sizes. Indeed, notice that LR+GD has the fastest convergence to Accuracy=1.0 with the step sizes $\gamma \in \{1.0, 10.0, 100.0\}$, but at the same time, it has the highest loss values with these steps (more experiments in the next sections and appendix).

2) We investigate this phenomenon further and show that *the logistic loss and the norm of gradients are unreliable metrics*. We argue that the fact that a loss value $f(\theta_t)$ is small does not necessarily indicate that θ_t is close to solving (1). Surprisingly, the opposite can be true. Our experiments and theorem show that high loss values may indicate fast convergence. (see Section 4)

3) This finding implies that when analyzing and developing methods for solving (1), we have to look at the number of iterations required by methods to solve (1) rather than relying solely on loss and gradient values. Therefore, we looked at the iteration complexity ${}^{nR^2}/_{\mu^2}$ of LR+GD with $\gamma \to \infty$ and noticed that it is suboptimal with respect to n since the iteration complexity ${}^{R^2}/_{\mu^2}$ can be attained by the classical (non-batch) perceptron algorithm (Perceptron). Moreover, we prove a lower bound, showing that the dependence on n cannot be avoided in LR+GD with $\gamma \to \infty$. Provably, LR+GD is a suboptimal method with large step sizes. Finally, we slightly modify LR+GD and develop a new method, Normalized LR+GD, basing on the connection between LR+GD and Batch Perceptron. This new method provably improves the iteration rate of LR+GD to ${}^{R^2}/_{\mu^2}$ when $\gamma \to \infty$. The new iteration rate to solve (1) is n times better.

3 Reduction to the Batch Perceptron Algorithm

Before we state our first result, let us recall a batch version of the perceptron algorithm (Novikoff 1962; Duda, Hart, and G.Stork 2001):

$$\text{Take the first step } \hat{\theta}_1 = \hat{\theta}_0 + \frac{1}{2n}\sum_{i=1}^{n} y_i a_i.$$

$$\text{For all } t \geq 1, \text{ find the set } S_t := \{i \in [n] \,:\, y_i\, {a_i}^\top \hat{\theta}_t \leq 0\,\},$$

$$\text{and take the step } \hat{\theta}_{t+1} = \hat{\theta}_t + \frac{1}{n}\sum_{i\in S_t} y_i a_i \text{ while } |S_t| \neq 0, \tag{Batch Perceptron}$$

where $\hat{\theta}_0$ is a starting point. This method finds all misclassified samples and uses them to find the next iterate $\hat{\theta}_{t+1}$ of the algorithm. Note that the classical version of the perceptron algorithm does the step only with one misclassified sample, as presented in (Perceptron).

We will require the following technical assumption in Theorem 3.2:

Assumption 3.1. (Non-Degenerate Dataset) For all $j \in [n]$, the hyperplane $\{x \in \mathbb{R}^n : \sum_{i=1}^n x_i \langle y_i a_i, y_j a_j\rangle = 0\}$ *does not* intersect the point $(0.5 + k_1, \ldots, 0.5 + k_n)$ for all $k_1, ..., k_n \in \mathbb{N}_0$.

This is a very weak assumption that cuts off pathological datasets since the chances that any hyperplane will intersect the *countable* set are zero in practice. Indeed, assume that $\{x \in \mathbb{R}^n : \sum_{i=1}^n x_i \langle y_i a_i, y_j a_j\rangle = 0\}$ intersects some point $(0.5+k_1, \ldots, 0.5+k_n)$. For any arbitrarily small $\sigma > 0$, let us take i.i.d. normal noises $\xi_1, \ldots, \xi_n \sim \mathcal{N}(0, \sigma)$. Then the probability that a slightly perturbed hyperplane $\{x \in \mathbb{R}^n : \sum_{i=1}^n x_i(\langle y_i a_i, y_j a_j\rangle + \xi_i) = 0\}$ intersects any point $(0.5 + k_1, \ldots, 0.5 + k_n)$ is zero. We are ready to state and prove the first result:

Theorem 3.2. *Let Assumption 1.1 hold. For $\gamma \to \infty$ and[3] $\theta_0 = 0$, the logistic regression with gradient descent (LR+GD) reduces to the batch perceptron algorithm (Batch Perceptron), i.e., $\theta_t/\gamma \to \hat{\theta}_t$ for all $t \geq 0$, with $\hat{\theta}_0 = 0$ if the dataset satisfies Assumption 3.1 (almost all datasets).*

Proof. Clearly, we have

$$\nabla f(\theta) = -\frac{1}{n}\sum_{i=1}^{n}(1 + \exp(y_i\, a_i^\top \theta))^{-1} y_i a_i \tag{3}$$

and $\theta_1 = \theta_0 - \gamma \nabla f(\theta_0)$. Thus $\theta_1/\gamma \to \hat{\theta}_1 = \frac{1}{2n}\sum_{i=1}^n y_i a_i$ and $\theta_0/\gamma \to \hat{\theta}_0 = 0$ when $\gamma \to \infty$. We now use mathematical induction, and assume that θ_t/γ converges to $\hat{\theta}_t$ when $\gamma \to \infty$. Using simple algebra, we get

$$\begin{aligned}\frac{\theta_{t+1}}{\gamma} &= \frac{\theta_t + \gamma \frac{1}{n}\sum_{i=1}^n \frac{1}{1+\exp(y_i\, a_i^\top \theta_t)} y_i a_i}{\gamma} \\ &= \frac{\theta_t}{\gamma} + \frac{1}{n}\sum_{i=1}^{n} \frac{1}{1 + \exp(\gamma \cdot y_i\, a_i^\top \frac{\theta_t}{\gamma})} y_i a_i.\end{aligned}$$

For all $t \geq 1$, notice that $\hat{\theta}_t = \frac{1}{n}\sum_{i=1}^n (0.5 + k_i) y_i a_i$ in Batch Perceptron for some $k_1, \ldots, k_n \in \mathbb{N}_0$. Using Assumption 3.1, we have[4] $y_i a_i^\top \hat{\theta}_t \neq 0$ for all $i \in [n]$. Since $\theta_t/\gamma \to \hat{\theta}_t$ when $\gamma \to \infty$, we get $\text{sign}(y_i a_i^\top \theta_t/\gamma) = \text{sign}(y_i a_i^\top \hat{\theta}_t) \neq 0$ for all $i \in [n]$ and γ large enough. Therefore $(1 + \exp(\gamma \cdot y_i\, a_i^\top \frac{\theta_t}{\gamma}))^{-1} \to 1$ if $y_i\, a_i^\top \hat{\theta}_t < 0$, and $(1 + \exp(\gamma \cdot y_i\, a_i^\top \frac{\theta_t}{\gamma}))^{-1} \to 0$ if $y_i\, a_i^\top \hat{\theta}_t > 0$ when $\gamma \to \infty$ for all $i \in [n]$, meaning $\frac{\theta_{t+1}}{\gamma} \overset{\gamma\to\infty}{=} \hat{\theta}_t + \frac{1}{n}\sum_{i\in S_t} y_i a_i$. We have showed that $\theta_{t+1}/\gamma \to \hat{\theta}_{t+1}$. □

Thus, indeed, LR+GD reduces to Batch Perceptron when $\gamma \to \infty$. It is left to recall the following classical result, which we prove in Section B for completeness.

Theorem 3.3. *[(Novikoff 1962; Duda, Hart, and G.Stork 2001)] Let Assumption 1.1 hold. The batch perceptron algorithm (Batch Perceptron) solves* (1) *after at most*

$$\frac{nR^2}{\mu^2} \tag{4}$$

[3]The theorem is true for $\theta_0 \neq 0$, but we have to modify Assumption 3.1, and change 0.5 to $(1 + \exp(y_i\, a_i^\top \theta_0))^{-1}$.

[4]This statement is the only and the main reason why we need Assumption 3.1. This assumption helps to avoid the corner case when the samples lie on the hyperplane.

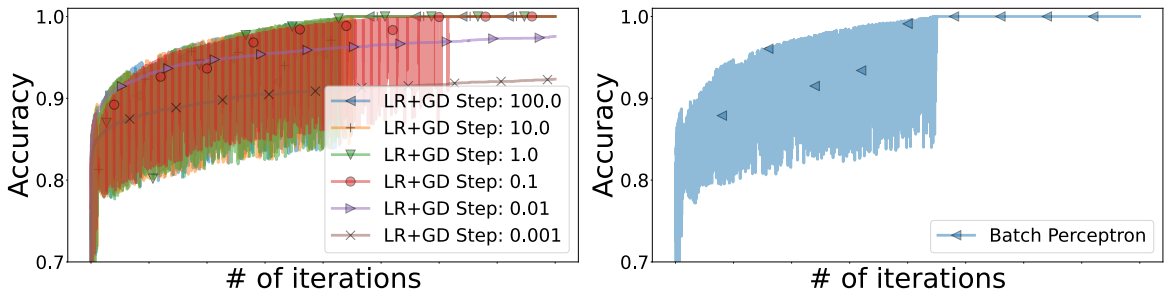


Figure 2: We show that LR+GD with large step sizes aligns with Batch Perceptron on *CIFAR-10*.

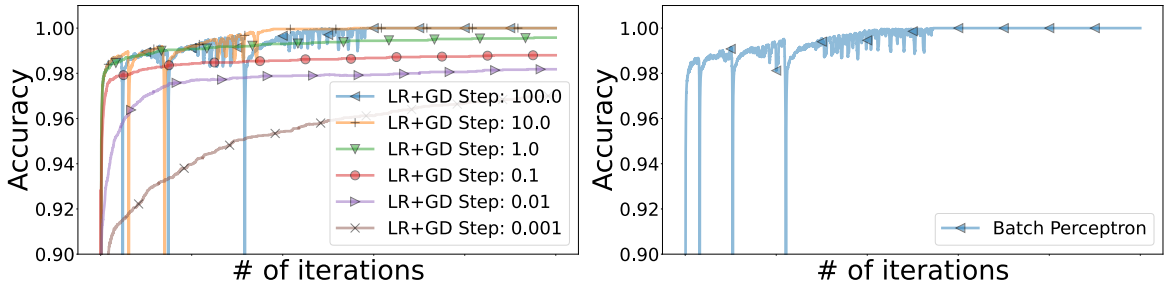


Figure 3: We show that LR+GD with large step sizes aligns with Batch Perceptron on *FashionMNIST*.

iterations if $\hat{\theta}_0 = 0$.

Theorem 3.3 and Theorem 3.2 explain why LR+GD solves (1) with $\gamma \to \infty$. Notice that the convergence rate (4) does not degenerate when $\gamma \to \infty$. Crucially, we provide the convergence guarantees for the task (1), not for the task (2). The latter is merely *a proxy problem*. In practice, what matters is how fast we find a separator rather than how fast the loss converges to zero, and in fact, we will see in Section 4 that the logistic loss is an unreliable metric.

Remark: A scaled version of (2), $\frac{1}{t \times n} \sum_{i=1}^{n} \log\left(1 + \exp(-t \times y_i a_i^\top \theta)\right)$, reduces to the perceptron loss when $t \to \infty$. Thus, there can potential connection between LR+GD with large step sizes and this fact.

Numerical experiments. We now numerically verify the obtained results by comparing the performance of two algorithms that solve (1): logistic regression with gradient descent (LR+GD) and the perceptron algorithm (Batch Perceptron). Batch Perceptron has no hyperparameters, while LR+GD requires the step size γ. We evaluate these algorithms on four datasets: *CIFAR-10* (Krizhevsky, Hinton et al. 2009), *FashionMNIST* (Xiao, Rasul, and Vollgraf 2017), *EuroSAT* (Helber et al. 2019), and *MNIST* (LeCun, Cortes, and Burges 2010), selecting two classes and 5 000 samples from each dataset (see details in Section E). For LR+GD, we vary the step size from 0.001 to 100. Figures 2, 3, 8, and 10 present the results side by side. The results indicate that LR+GD with a small step size has monotonic and stable convergence curves. However, as the step size increases, the plots become unstable and chaotic. The behavior of LR+GD with large step sizes aligns closely with that of Batch Perceptron across all datasets almost exactly, which supports our theory. And in the limit of $\gamma \to \infty$, converges to Batch Perceptron. We also run experiments with 1 000 and 10 000 samples in Section E for additional support.

4 Logistic Loss and the Norm of Gradients are Unreliable Metrics

Looking closer at the results of the experiments on datasets (Figures 4, 5, 9, and 11), we notice that the large step size not only leads to faster convergence rates but also to larger function values (before the moment when Accuracy $= 1.0$). Is this a coincidence, or is there some pattern? We can prove the following simple theorem that explains the phenomenon:

Theorem 4.1. *Assume that* $\theta_1 = 0$. *There exists a separable dataset (Assumption 1.1) such that*

1. $f(\theta_1) \to \infty$ *and* $f(\theta_2) \to 0$ *when* $\gamma \to \infty$,
2. θ_2/γ *is a solution of* (1) *when* $\gamma \to \infty$,
3. $\|\nabla f(\theta_1)\| \to \sqrt{2}/2$ *and* $\|\nabla f(\theta_2)\| \to 0$ *when* $\gamma \to \infty$,

where θ_1 *and* θ_2 *are the first and second iterates of LR+GD.*

Proof. We take the dataset with one sample $(1, -1)^\top$ assigned to the class 1 and one sample $(-1, -4)^\top$ assigned to the class -1. Using (LR+GD) and (3), we have $\theta_1 = \gamma(\frac{2}{4}, \frac{3}{4})^\top$. Thus $f(\theta_1) = \frac{1}{2}\left(\log\left(1 + \exp(\frac{\gamma}{4})\right) + \log\left(1 + \exp(-\frac{7\gamma}{2})\right)\right)$, meaning $f(\theta_1) \to \infty$ when $\gamma \to \infty$. On the other hand, a direct calculation yield

$$\frac{\theta_2}{\gamma} = (\tfrac{2}{4}, \tfrac{3}{4})^\top + \tfrac{1}{2}\left(\left(1 + \exp(-\tfrac{\gamma}{4})\right)^{-1}(1, -1)^\top + \left(1 + \exp(\tfrac{7\gamma}{2})\right)^{-1}(1, 4)^\top\right) \to (1, \tfrac{1}{4})^\top$$

when $\gamma \to \infty$. The point $(1, \frac{1}{4})^\top$ is a solution of (1), and $f(\theta_2) \to 0$ when $\gamma \to \infty$. The last statement of the theorem can be verified using (3). □

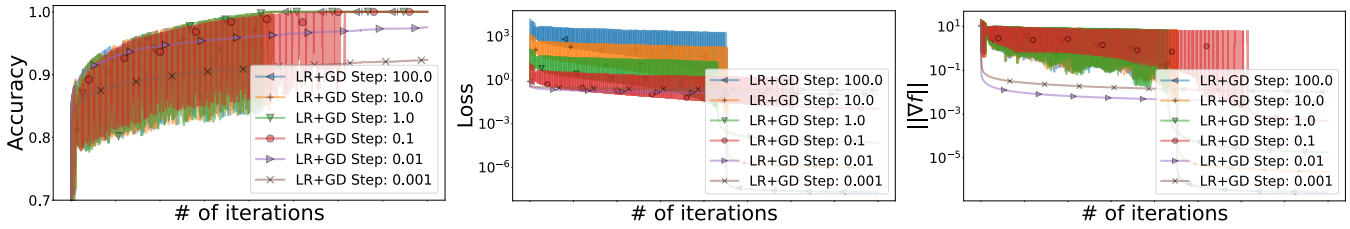


Figure 4: Accuracy, function values, and the norm of gradients of the logistic loss (2) on *CIFAR-10* during the runs of LR+GD.

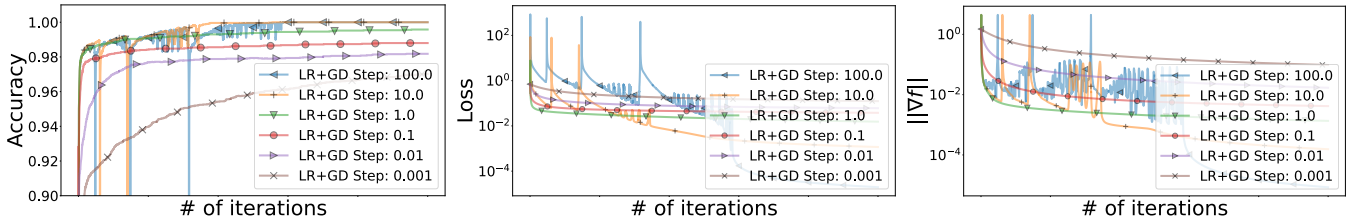


Figure 5: Accuracy, function values, and the norm of gradients on *FashionMNIST*. The larger the step size in LR+GD, the faster LR+GD solves (1) and gets Accuracy$= 1.0$. At the same time, the larger the step size, the higher the loss values.

Even though the first value $f(\theta_1)$ of the loss indicates divergence, the algorithm solves (1) after two steps when $\gamma \to \infty$! In this example, it is clear that the high value of $f(\theta_1)$ does not reflect the fact that the algorithm will solve the problem in the next step. The experiments from Figures 4 and 5 (see also Section E) support this theorem. That also applies to the norm of gradients. The ratio between $\|\nabla f(\theta_1)\|$ and $\|\nabla f(\theta_2)\|$ can be arbitrarily large for large γ. In Figures 4 and 5 (see also Section E), the norm of gradients are chaotic and large until the moment when LR+GD finds a solution of (1).

5 LR+GD is a Suboptimal Method

In the previous section, we explain that logistic loss and the norm of gradient do not provide sufficient information about our proximity to solving (1). Recall that GD is a method of choice because, for instance, it is an optimal method in the nonconvex setting (Carmon et al. 2020) and has the optimal convergence rate by the norm of gradients. In the case of the task (1) and LR+GD, this is no longer true. Indeed, let us now consider the iteration rate ${}^{nR^2}/_{\mu^2}$ from Theorem 3.3 by LR+GD when $\gamma \to \infty$. The iteration rate is suboptimal since it linearly depends on n, and can be improved by the classical (non-batch) perceptron algorithm (Novikoff 1962). The following lower bound proves that the dependence is unavoidable for Batch Perceptron.

Theorem 5.1. *There exists a separable dataset (Assumption 1.1) with $\mu = \Theta(1)$ and $R = \Theta(1)$ such that Batch Perceptron (LR+GD when $\gamma \to \infty$) requires at least $\Omega(n)$ iterations to solve* (1) *if $\hat{\theta}_0 = 0$ and $n \geq 10$.*

Proof. We take the dataset with one sample $(0.5, -1)^\top$ assigned to the class 1 and $n-1$ samples $(-0.5, -1)^\top$ assigned to the class -1. We start at the point $\hat{\theta}_0 = (0, 0)^\top$. Then $\hat{\theta}_1 = \hat{\theta}_0 + \frac{1}{2n}\sum_{i=1}^n y_i a_i = (0.25, \frac{n-2}{2n})^\top$. Only the sample from the class -1 is misclassified at $\hat{\theta}_1$ and belongs to S_1. Therefore $\hat{\theta}_2 = (0.25, \frac{n-2}{2n})^\top + \frac{1}{n}(0.5, -1)^\top = (0.25(1+\frac{2}{n}), \frac{n-4}{2n})^\top$. Again, only the sample from the class -1 belongs to S_2. Thus $\hat{\theta}_3 = (0.25(1+\frac{2}{n}), \frac{n-4}{2n})^\top + \frac{1}{n}(0.5, -1)^\top = (0.25(1+\frac{4}{n}), \frac{n-6}{2n})^\top$. This will happen further with $\hat{\theta}_4, \ldots, \hat{\theta}_k$ until either the last coordinate becomes negative (the samples from the class 1 will be misclassified), i.e., $\frac{n-2k}{2n} < 0$, or the sample from the class -1 stops being misclassified, i.e., $0.5 \times 0.25(1+\frac{2k-2}{n}) + -1 \times \frac{n-2k}{2n} > 0$. Both conditions require k to be greater or equal to $\Omega(n)$. □

We have proved that LR+GD with $\gamma \to \infty$ is a suboptimal method. At the same time, it is well-known that we can improve the rate using the classical versions of the perceptron algorithm that yield better iteration rates:

Theorem 5.2 ((Duda, Hart, and G.Stork 2001),Theorem 5.1)**.** *The classical perceptron algorithm (Novikoff 1962), defined as*

$$\text{For all } t \geq 0, \text{ find the set } S_t := \{i \in [n] : y_i \, a_i{}^\top \hat{\theta}_t \leq 0\},$$
$$\text{choose } j \in S_t \text{ and take the step } \hat{\theta}_{t+1} = \hat{\theta}_t + y_j a_j, \qquad \text{(Perceptron)}$$

solves (1) *after at most $\frac{R^2}{\mu^2}$ iterations if $\hat{\theta}_0 = 0$.*

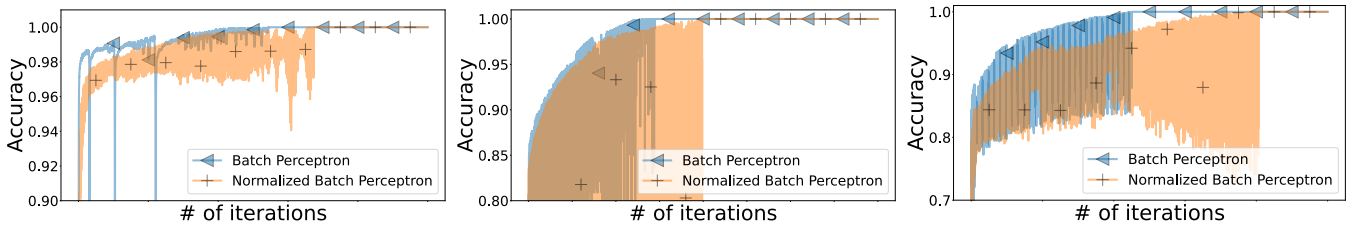
We can also consider a practical variant with proper normalization. Using a different normalization factor, ${}^1/_{|S_t|}$ instead of ${}^1/_n$, we can provide better guarantees:

Theorem 5.3. *[Proof in Section D] The batch perceptron algorithm with a proper normalization, defined as*

$$\text{For all } t \geq 0, \text{ find the set } S_t := \{i \in [n] : y_i \, a_i{}^\top \hat{\theta}_t \leq 0\},$$
$$\text{and take the step } \hat{\theta}_{t+1} = \hat{\theta}_t + \frac{1}{|S_t|}\sum_{i \in S_t} y_i a_i \text{ while } |S_t| \neq 0, \qquad \text{(Normalized Batch Perceptron)}$$

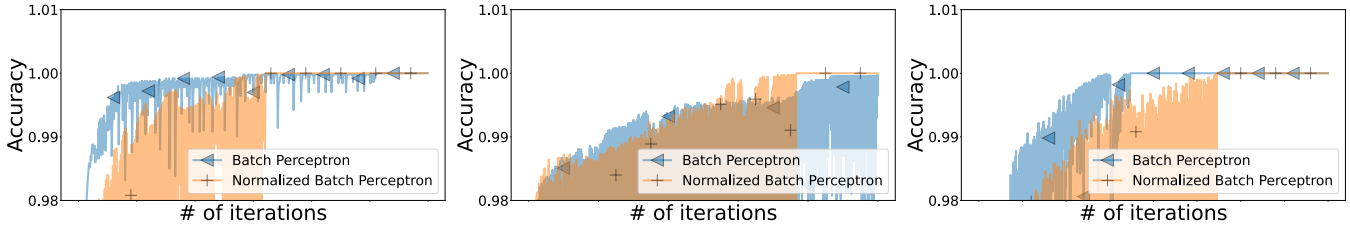
solves (1) *after at most*

$$\frac{R^2}{\mu^2}$$



(a) *Fashion MNIST* (b) *EuroSAT* (c) *CIFAR-10*

Figure 6: Comparison of perceptron algorithms.



(a) *Fashion MNIST* (b) *EuroSAT* (c) *CIFAR-10*

Figure 7: Comparison of perceptron algorithms on **imbalanced** data (see details in Section E). On two of three datasets Normalized Batch Perceptron converges to Accuracy=1.0 faster than Batch Perceptron.

iterations if $\hat{\theta}_0 = 0$.

One can see that Perceptron and Normalized Batch Perceptron have n times better convergence rates than Batch Perceptron. The only difference between Normalized Batch Perceptron and Batch Perceptron is the proper normalization, which is crucial to get a better iteration rate.

Numerical experiments. In Figure 6, we compare Normalized Batch Perceptron and Batch Perceptron numerically and observe that Batch Perceptron converges slightly better in practice despite worse theoretical guarantees. However, in a setup with imbalanced data, described in Section E, we observe that Normalized Batch Perceptron finds a solution faster than Batch Perceptron (Figure 7). One research question is to uncover the reasons behind this. A potential high-level explanation is that Batch Perceptron performs well "on average", but not robust to inbalanced data.

6 A New Method, Normalized LR+GD, Yields a Faster Iteration Rate

Since Normalized Batch Perceptron converges faster than Batch Perceptron by n times, it raises the question of whether it is possible to modify LR+GD and obtain a better rate. The answer is affirmative. We propose the following method:

$$\theta_{t+1} = \theta_t - \gamma \beta_t \nabla f(\theta_t), \qquad \text{(Normalized LR+GD)}$$

where $\nabla f(\theta_t)$ is the gradient of (2) at the point θ_t, and

$$\beta_t = \left(\tfrac{1}{n}\sum_{i=1}^n (1+\exp(y_i a_i^\top \theta_t))^{-1}\right)^{-1}.$$

We reverse-engineered this method from Normalized Batch Perceptron, observing that $\nabla f(\theta_t) \to \frac{1}{|S_t|}\sum_{i\in S_t} y_i a_i$ (see Theorem 3.2) and $\beta_t \to |S_t|$ when $\gamma \to \infty$. There are many ways how one can interpret it. For instance, it can be seen as LR+GD but with adaptive step sizes. Note that this method is specialized for the problem (2) because β_t requires the features $\{a_i\}_{i=1}^n$ and labels $\{y_i\}_{i=1}^n$. We can prove that this method solves (1) faster than LR+GD:

Theorem 6.1. *Let Assumption 1.1 hold. Normalized LR+GD solves* (1) *after at most*

$$\frac{R^2}{\mu^2} + \frac{2\log(2n-1)}{\gamma\mu^2} \qquad (5)$$

iterations if $\theta_0 = 0$.

The theorem suggests that we should increase the step size and let $\gamma \to \infty$.

Numerical experiments. This theoretical insight is supported by the experiments from Table 1, where the best convergence rate is achieved with large step sizes. In Table 1, we compare the methods and observe that Normalized LR+GD is more robust to imbalanced datasets.

7 Conclusion

In this work, we analyze the classification problem (1) through the logistic regression (2). Our key takeaways are

1. Logistic regression and GD with large step sizes reduces to the celebrated (batch) perceptron algorithm, which can explain why LR+GD solves (1) even when $\gamma \to \infty$.

2. We can not fully trust function and gradient values when optimizing logistic regression problems. The same caution applies to theoretical works. If a theoretical method has a good convergence rate based on function value residuals or gradient norms ($f(\theta_t) - f^* \le \dots$ or $\|\nabla f(\theta_t)\|^2 \le \dots$), it does not necessarily mean that this method will perform well in practical machine learning tasks. In fact, the opposite may be true.

Dataset	Method	# of Iterations
"Worst-case" Dataset from Theorem 5.1	Normalized LR+GD Step: 100.0	**2**
	Normalized LR+GD Step: 10.0	**2**
	Normalized LR+GD Step: 1.0	16
	Normalized LR+GD Step: 0.1	157
	LR+GD Step: 100.0	308
Imbalanced *Fashion MNIST* (Sec. E)	Normalized LR+GD Step: 100.0	**311**
	Normalized LR+GD Step: 10.0	349
	Normalized LR+GD Step: 1.0	353
	Normalized LR+GD Step: 0.1	504
	LR+GD Step: 100.0	538
Balanced *Fashion MNIST* (Sec. E)	LR+GD Step: 10.0	**2778**
	LR+GD Step: 100.0	2928
	Normalized LR+GD Step: 100.0	3160
	Normalized LR+GD Step: 1.0	3351
	Normalized LR+GD Step: 10.0	3359

Table 1: The # of iterations required by LR+GD and Normalized LR+GD to solve (1) with various step sizes. Notice that Normalized LR+GD solves (1) *150 times* faster than LR+GD on the first dataset, 1.7 times faster on the second dataset, and only 1.1 times slower on the third dataset.

3. While it is well-known that GD is an *optimal* method for nonconvex problems when measuring convergence by *the norm of gradients* (Carmon et al. 2020), and (almost[5]) optimal for convex problems when measuring convergence by *function values* (Nesterov 2018), the iteration rate of GD to solve (1) with logistic regression is suboptimal and does not scale with # of of data points n in the worst case. This can be improved with the new Normalized LR+GD method.

8 Future Work

Nonlinear models and neural networks. The natural question is whether extending this work to nonlinear models and neural networks is possible. In the general case, we have to

$$\text{find a vector } \theta \in \mathbb{R}^m \text{ such that } y_i g(a_i;\theta) > 0 \quad \forall i \in [n],$$

where $g : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ is a nonlinear mapping. The mathematical aspects in this section are not strict but rather serve as a foundation for future research since the general case with nonlinear models is very challenging and unexplored. All theorems from the previous sections heavily utilize the fact that g is a linear model. If g is a neural network, then it is well-known that logistic regression will diverge for large step sizes. Let us look at the gradient step with the logistic loss:

$$\theta_{t+1} = \theta_t + \frac{1}{n}\sum_{i=1}^{n}(1+\exp(y_i g(a_i;\theta_t)))^{-1}\nabla_\theta g(a_i;\theta_t).$$

For the linear model, in the proof of Theorem 3.2, we show that $(1+\exp(y_i g(a_i;\theta_t)))^{-1} \to \mathbf{1}[y_i g(a_i;\theta_t) < 0]$ when $\gamma \to \infty$. For the nonlinear models, it is not clear if it is true. Nevertheless, if we assume that $(1+\exp(y_i g(a_i;\theta_t)))^{-1} \approx \mathbf{1}[y_i g(a_i;\theta_t) < 0]$, then

$$\theta_{t+1} \approx \theta_t + \frac{1}{n}\sum_{i\in S_t}\nabla_\theta g(a_i;\theta_t),$$
$$S_t := \{i \in [n] \,:\, y_i\, g(a_i;\theta_t) \le 0\,\}$$

which can be seen as a *generalized perceptron algorithm*. As far as we know, the analysis and convergence rates for this method have not been explored well. We believe these questions are important research endeavors.

Implicit bias. One of the main features of LR+GD is *implicit bias*. For small step sizes, $\gamma < 2/L$, Soudry et al. (2018); Ji and Telgarsky (2018) showed that the iterates of LR+GD not only solve (1), but also have a stronger property: $\theta_t \to \theta_*$ when $t \to \infty$, where $\theta_* = \arg\max_{\|\theta\|=1} \min_{i\in[n]} y_i a_i^\top \theta$ is the max-margin/SVM solution. From our observation, for $\gamma \to \infty$, LR+GD reduces to Batch Perceptron, which generally does not return the max-margin solution. Therefore, to ensure the implicit bias property, one has to choose $\gamma < \infty$, and intuitively, the larger γ, the slower the convergence to the max-margin solution, but faster convergence to a solution of (1) according to the experiments and Theorem 6.1.

Theoretical guarantees of optimization methods. Typically, when researchers develop new methods, they compare them with previous methods using convergence rates by (loss) function values or by the norm of gradients. We believe that this work raises an important concern about this methodology in the context of machine learning tasks. While this work only analyzes the logistic loss with a linear model, the problem can be even more dramatic with more complex losses and models.

[5]There exists the accelerated gradient method by Nesterov (1983, 2018)

References

Ahn, K.; Bubeck, S.; Chewi, S.; Lee, Y. T.; Suarez, F.; and Zhang, Y. 2024. Learning threshold neurons via edge of sta-

bility. *Advances in Neural Information Processing Systems*, 36.

Ahn, K.; Zhang, J.; and Sra, S. 2022. Understanding the unstable convergence of gradient descent. In *International Conference on Machine Learning*, 247–257. PMLR.

Bishop, C. M.; and Nasrabadi, N. M. 2006. *Pattern recognition and machine learning*, volume 4. Springer.

Block, H.-D. 1962. The perceptron: A model for brain functioning. i. *Reviews of Modern Physics*, 34(1): 123.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Carmon, Y.; Duchi, J. C.; Hinder, O.; and Sidford, A. 2020. Lower bounds for finding stationary points I. *Mathematical Programming*, 184(1): 71–120.

Chen, L.; and Bruna, J. 2022. On gradient descent convergence beyond the edge of stability. *arXiv preprint arXiv:2206.04172*, 3.

Cohen, J.; Kaur, S.; Li, Y.; Kolter, J. Z.; and Talwalkar, A. 2021. Gradient Descent on Neural Networks Typically Occurs at the Edge of Stability. In *International Conference on Learning Representations*.

Cortes, C.; and Vapnik, V. 1995. Support-vector networks. *Machine learning*, 20: 273–297.

Damian, A.; Nichani, E.; and Lee, J. D. 2023. Self-Stabilization: The Implicit Bias of Gradient Descent at the Edge of Stability. In *The Eleventh International Conference on Learning Representations*.

Duda, R.; Hart, P.; and G.Stork, D. 2001. *Pattern Classification*. Wiley Interscience.

Even, M.; Pesme, S.; Gunasekar, S.; and Flammarion, N. 2023. (S)GD over Diagonal Linear Networks: Implicit bias, Large Stepsizes and Edge of Stability. *Advances in Neural Information Processing Systems*, 36: 29406–29448.

Helber, P.; Bischke, B.; Dengel, A.; and Borth, D. 2019. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7): 2217–2226.

Ji, Z.; and Telgarsky, M. 2018. Risk and parameter convergence of logistic regression. *arXiv preprint arXiv:1803.07300*.

Johnson, R.; and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems*, 26.

Kreisler, I.; Nacson, M. S.; Soudry, D.; and Carmon, Y. 2023. Gradient descent monotonically decreases the sharpness of gradient flow solutions in scalar networks and beyond. In *International Conference on Machine Learning*, 17684–17744. PMLR.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto.

LeCun, Y.; Cortes, C.; and Burges, C. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Lewkowycz, A.; Bahri, Y.; Dyer, E.; Sohl-Dickstein, J.; and Gur-Ari, G. 2020. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*.

Liu, D. C.; and Nocedal, J. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3): 503–528.

Lyu, K.; Li, Z.; and Arora, S. 2022. Understanding the generalization benefit of normalization layers: Sharpness reduction. *Advances in Neural Information Processing Systems*, 35: 34689–34708.

Ma, C.; Kunin, D.; Wu, L.; and Ying, L. 2022. Beyond the quadratic approximation: the multiscale structure of neural network loss landscapes. *arXiv preprint arXiv:2204.11326*.

Meng, S. Y.; Orvieto, A.; Cao, D. Y.; and De Sa, C. 2024. Gradient Descent on Logistic Regression with Non-Separable Data and Large Step Sizes. *arXiv preprint arXiv:2406.05033*.

Nesterov, Y. 1983. A Method for Solving a Convex Programming Problem with Convergence Rate O (1/k**2). In *Soviet Mathematics. Doklady*, volume 27, 367–372.

Nesterov, Y. 2018. *Lectures on convex optimization*, volume 137. Springer.

Novikoff, A. B. 1962. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, 615–622. New York, NY.

Orabona, F. 2024. A Minimizer Far, Far Away. https://parameterfree.com/2024/02/14/a-minimizer-far-far-away/.

Robbins, H.; and Monro, S. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics*, 400–407.

Schmidt, M.; Le Roux, N.; and Bach, F. 2017. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162: 83–112.

Song, M.; and Yun, C. 2023. Trajectory alignment: understanding the edge of stability phenomenon via bifurcation theory. *arXiv preprint arXiv:2307.04204*.

Soudry, D.; Hoffer, E.; Nacson, M. S.; Gunasekar, S.; and Srebro, N. 2018. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1): 2822–2878.

Wang, Z.; Li, Z.; and Li, J. 2022. Analyzing sharpness along GD trajectory: Progressive sharpening and edge of stability. *Advances in Neural Information Processing Systems*, 35: 9983–9994.

Wu, J.; Bartlett, P. L.; Telgarsky, M.; and Yu, B. 2024. Large Stepsize Gradient Descent for Logistic Loss: Non-Monotonicity of the Loss Improves Optimization Efficiency. In *Conference on Learning Theory*.

Wu, J.; Braverman, V.; and Lee, J. D. 2024. Implicit bias of gradient descent for logistic regression at the edge of stability. *Advances in Neural Information Processing Systems*, 36.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Zhu, X.; Wang, Z.; Wang, X.; Zhou, M.; and Ge, R. 2022. Understanding edge-of-stability training dynamics with a minimalist example. *arXiv preprint arXiv:2210.03294*.