A Theoretical Analysis of the Learning Dynamics under Class Imbalance

Emanuele Francazi [1 2] **Marco Baity-Jesi** [2] **Aurelien Lucchi** [3]

Abstract

Data imbalance is a common problem in machine learning that can have a critical effect on the performance of a model. Various solutions exist but their impact on the convergence of the learning dynamics is not understood. Here, we elucidate the significant negative impact of data imbalance on learning, showing that the learning curves for minority and majority classes follow sub-optimal trajectories when training with a gradient-based optimizer. This slowdown is related to the imbalance ratio and can be traced back to a competition between the optimization of different classes. Our main contribution is the analysis of the convergence of full-batch (GD) and stochastic gradient descent (SGD), and of variants that renormalize the contribution of each per-class gradient. We find that GD is not guaranteed to decrease the loss for each class but that this problem can be addressed by performing a per-class normalization of the gradient. With SGD, class imbalance has an additional effect on the direction of the gradients: the minority class suffers from a higher directional noise, which reduces the effectiveness of the per-class gradient normalization. Our findings not only allow us to understand the potential and limitations of strategies involving the per-class gradients, but also the reason for the effectiveness of previously used solutions for class imbalance such as oversampling.

1. Introduction

In supervised classification, as well as other problems in machine learning, datasets are often affected by data imbalance. Although the situation can sometimes be solved or mitigated by changing the data collection method, this might induce an undesirable data shift (Quiñonero-Candela et al., 2009; Moreno-Torres et al., 2012). Importantly, many datasets are intrinsically unbalanced (Van Horn and Perona, 2017; Feldman, 2020; D'souza et al., 2021), as in the case of spam identification (Liu et al., 2017), fraud detection (Makki et al., 2019), or biodiversity monitoring (Kyathanahally et al., 2021).

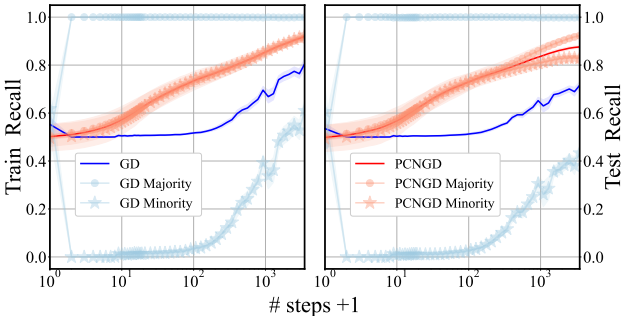

Figure 1. Training (left) and test (right) recall of the *Mod1* on the *Bi60* datasets (see details in Sec. 4.4). Due to the data imbalance, GD (blue curves) first focuses on the majority class only, while the minority class stays at zero accuracy. This effect is suppressed when using the PCNGD algorithm. See Fig. 3 for the related loss function curves.

The problem of data imbalance has already received attention in the literature and we broadly identify three most common ways to handle it. These either act on the data distribution (He and Garcia, 2009; Huang et al., 2016), adjust the objective function (Japkowicz, 2000; Huang et al., 2016; Alshammari et al., 2022) or modify the learning algorithm (Tang et al., 2020; Anand et al., 1993). We provide a detailed discussion of these methods in App. A. The important feature that makes our work depart from prior work is that we focus our study on the *dynamics* of gradient-based learning in the presence of data imbalance. We will for instance analyze the theoretical convergence guarantees of these methods, which, to the best of our knowledge, is a novel contribution.

We start our investigation from the following two empirical observations: (i) The learning dynamics is delayed for imbalanced problems. This is especially a problem during hyperparameter tuning, since it requires very long runs to assess the difference between distinct hyperparameter choices. (ii) While the overall performance improves during

[1]Physics Department, EPFL, Switzerland [2]SIAM Department, Eawag (ETH), Switzerland [3]Department of Mathematics and Computer Science, University of Basel, Switzerland. Correspondence to: Emanuele Francazi <emanuele.francazi@epfl.ch>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

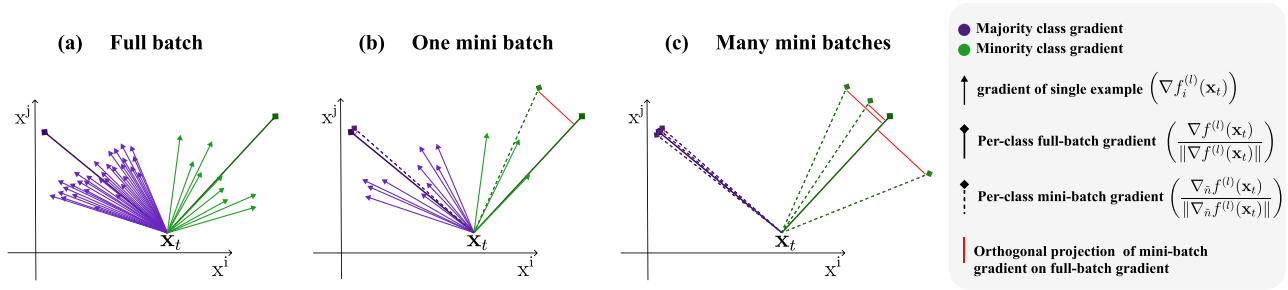

Figure 2. Diagram explaining the directional noise caused by class imbalance on (PCN)SGD in binary classification. We plot two generic components of the parameter vector $\mathbf{x}$ on the axes (x^i, x^j). Starting from a given iterate $\mathbf{x}_t$ at time t, the normalized per-class gradients associated with individual batches, and the entire dataset are represented in the plots. The gradients of the individual examples that make them up are also shown. **(a)** The contribution of each example to the per-class full-batch gradient (FBG). **(b)** Within a mini batch, instead, we consider only a randomly selected subset of dataset elements; the mini-batch gradient will therefore come from a random selection in this subset. **(c)** We show several mini batches and observe that they are more aligned to the FBG of the majority class than the FBG of the minority class. We will see this has negative consequences on the dynamics. For more details, see Sec. 5.1.

the dynamics, that of the minority classes quickly deteriorates at first. This is shown in Fig. 1 (blue curves) for a binary unbalanced classification problem. We call this initial deterioration the ***minority initial drop*** **(MID)**.[1] One can hypothesize why this happens (we will in fact demonstrate this rigorously): at the beginning of learning, the gradient steps are dominated by the majority class, driving the classifier weights towards configurations that correctly classify the majority class, regardless of the minority class. In fact, due to the imbalance, decreasing the loss related to the majority class outweighs the loss increase coming from the minority class. The minority classes will only be learned once the gradient of the majority class examples is small enough. To gain further insights, it therefore seems logical to express the loss as a sum over the examples belonging to each class, and analyze separately the gradient related to each class.

Guided by these observations, we characterize how class imbalance affects the learning dynamics, and study whether it is possible to adapt gradient-based algorithms in order to guarantee the decrease of the loss function of every single class. These adaptations provide us with a deeper insight into how class imbalance affects the training dynamics. The main adaptation that we study is per-class normalization (PCN), *i.e.* we normalize the (S)GD steps in such a way that the magnitude of the signal related to each class is the same. We name *per-class-normalized GD (PCNGD)* the PCN version of GD. For the PCN version of SGD we use the acronym PCNSGD.

We identify our key contributions as follows:

- **Suboptimality of GD and SGD:** Expanding on the observations of (Ye et al., 2021), we provide further evidence for the deceleration caused by class imbalance. We relate this to the non-monotonicity of the minority class losses, providing theorems and analytical arguments explaining the slowdown.
- **Theoretical study of the PCNGD algorithm:** This algorithm is not completely new as it relates to a variant that was empirically motivated in (Anand et al., 1993). We derive a theoretical convergence analysis showing that, for a suitable step-size, PCNGD is guaranteed to decrease the loss of all classes [2]. In addition, we also provide empirical evidence that PCNGD performs better than GD, in terms of both overall and per-class performance indicators (as shown in Fig. 1).
- **Imbalance causes directional noise in SGD:** We show that, while normalizing the per-class gradients is sufficient to counter class imbalance in GD, this is not true for SGD, since the imbalance influences the *direction* of the per-class signal in addition to its norm, and this suppresses the minority class learning (Fig. 2).
- **A framework to understand imbalance:** Our analysis also clarifies the effectiveness of other approaches already used heuristically, such as oversampling, since they can be interpreted in light of whether they help counter the directional noise; and shows how class imbalance should be seen as one of many factors (*e.g.* class difficulty) that influence the per-class gradients.

2. Related work

(Ye et al., 2021) noticed empirically that, in imbalanced SGD dynamics, minority classes are learned later than majority ones. They call *under-fitting phase of minor classes* the initial part of the learning where the minority classes are not learned. Building on this observation, we provide a theory and didactic experiments, explaining this phenomenon

[1] We show in Fig. S11 that the MID occurs also with SGD and multiclass classification.

[2] More specifically, we will see that GD can also decrease the per-class loss but only under a restrictive condition on the angle between the per-class gradients.

formally and showing how it is qualitatively different between GD and SGD. We emphasize that this is linked to an initial deterioration in the minority class recall/loss, so we prefer to focus on avoiding the effect of the MID on the model performance, which leads to an under-fitting phase for the minor classes. Furthermore, (Ye et al., 2021) provide a method to mitigate the MID, while we show what are the theoretical requirements to eliminate it.

Noticing that the gradient magnitude of the minority class is smaller than that of the majority class, (Anand et al., 1993) proposed an algorithm that takes the bisection between the per-class gradients to perform the update rule of the model parameters. The PCN algorithms that we analyze in this work can be seen as an extension of the bisection algorithm, where the optimization steps are taken in the same direction but with a different modulus. Our adaptation allows us to use the algorithm also in the multiclass and SGD settings, and to derive convergence guarantees.

Gradient imbalance is also a problem discussed in multi-task learning. The latter is akin to classification problems; in fact, learning multiple classes can be thought of as learning multiple tasks together if each task is assumed to be just one class *vs* the rest. The approaches proposed in (Chen et al., 2018; Yu et al., 2020), beyond some differences in implementation, are conceptually close to the PCN methods examined here. However, our work, analyzing separately the effects induced on GD and SGD, shows how the two cases, in classification problems, are characterized by substantial differences. In particular, we show how PCN alone is effective in the full-batch case, while in the stochastic case it is necessary to introduce variants to account, not only for the difference in norms, but also for the difference in directional noise.

3. Structure of the article

Sec. 4 focuses on full-batch algorithms. GD is analyzed in Sec. 4.2 where we prove that, while the dynamics converge to a critical point, the angle between the per-class gradients must be small to guarantee a monotonic decrease of the per-class loss functions and avoid the MID. The latter condition on the gradient angle becomes stronger with increasing imbalance [see Eq. (1)]. To solve this problem, in Sec. 4.3 we introduce and study PCNGD, which equalizes the per-class gradient norms, resulting in a relaxed condition on the angles and removing the MID. We also demonstrate that for a specific type of loss function (known as gradient-dominated functions), PCNGD not only ensures a monotonic per-class loss but also converges to the global minimum (Th. 4.3). In Sec. 4.4, we validate our conclusions through experiments. In Sec. 5, we turn our attention to stochastic algorithms. Sec. 5.1 shows that, unless the batch size is large, PCN is not sufficient to avoid the MID because imbalance affects not only the per-class gradient norms but also their direction

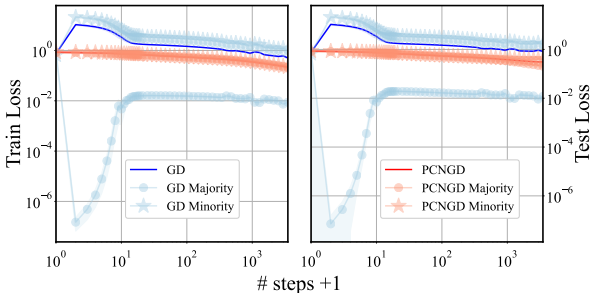

Figure 3. Training (left) and test (right) loss curves for GD and PCNGD algorithms. The corresponding recall is shown for the same setup in 1.

(as illustrated in Fig. 2). Sec. 5.2 reaches the same conclusion by showing that the per-class losses in PCNSGD are not guaranteed to converge monotonically unless the batch size is large (Th. 5.1). To support our results, Secs. 5.3 and 5.4 experimentally show that using algorithms that take into account both effects induced by imbalance (the disproportions in the per-class norm *and* direction) finally avoids the MID also when using stochastic gradient updates.
The notation used in the article is summarized in App. B. The theorems formulated in the main paper are in Apps. C and D. Algorithms are discussed in more detail in App. F. Model and dataset details used for experiments are in App. G and further experiments are in App. H. App. I describes the limitations and ethics of this work.

4. Convergence Guarantees with Full Batch

The loss curves in Fig. 3 are illustrative of the differences between the loss of GD and PCNGD. In particular, the wide gap between per-class performances, in the early stage of GD dynamics, is absent in PCNGD. The absence of the MID phase at the beginning of the dynamics results in accelerated growth in the performances (see also Fig. 1). In this section, we will analyze the differences in the two cases by studying the conditions that guarantee the decrease in per-class loss.

4.1. Setting

Assume we are in a typical supervised setting where we are given a training set of samples $\mathcal{D} = (\xi_i, y_i)_{i=1}^n$ where $\xi_i \in \mathbb{R}^d$ are features, $y_i \in \{0, 1, \dots L-1\}$ are the corresponding labels and L is the number of distinct labels. Our goal is to train a model parametrized by a vector $\mathbf{x} \in \mathbb{R}^m$ that minimizes a given loss function $f : \mathbb{R}^m \to \mathbb{R}$ as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^m} \left[f(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x}) \right].$$

We also define the number of samples per class as n_k, and the fraction of samples per class as $\rho_k = \frac{n_k}{n}$. We will use the following notation to define the per-class losses: $f^{(l)}(\mathbf{x}) := \frac{1}{n} \sum_{i\in C_l} f_i(\mathbf{x}), \quad C_l = \{i \mid y_i = l\}, \quad l \in \{0, 1, \dots L-1\}$.

4.2. Gradient Descent

We start by analyzing the convergence of gradient descent run for T iterations on the loss corresponding to each class $f^{(l)}(\mathbf{x})$. For the sake of clarity, we prove our results for the binary case, where the number of classes $L = 2$ and $y_i \in \{0, 1\}$, but we note that these results are extendable to the case where $L > 2$. For example, by aggregating classes, our results are exact for step imbalance[3] with any number of classes. Also, multiclass classifiers can be expressed as a combination of binary problems. Additionally, in App. E, we provide a multiclass version of Th. 4.1 and 4.2.

To make our results more accessible, in the main paper we give an informal version of our theorems. We invite the reader to check App. C and D for a formal version. In App. C we also mention literature with similar convergence analyses. Since we do not assume that the objective function f is convex, we focus on showing that the gradient norms decrease with the number of iterations (which is the standard metric in the optimization literature, see *e.g.* (Ghadimi and Lan, 2013)). Specifically, our complexity measure will be $\min_{s \in S} \|\nabla f^{(l)}(\mathbf{x}_s)\|^2 \leq \epsilon$ for a given ϵ accuracy and over an interval S.

We first prove the per-class convergence of GD under imbalance, highlighting the requirements to obtain it. We denote the angle between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ by $\angle(\mathbf{x}, \mathbf{y})$. We also define $C_t := \frac{\|\nabla f^{(1)}(\mathbf{x}_t)\|}{\|\nabla f^{(0)}(\mathbf{x}_t)\|}$.

Theorem 4.1 (Informal)**.** *Assume that each $f^{(l)}$ for $l = 0, 1$ is L_1-Lipschitz and L_2-smooth* [a] *and let $\alpha(\mathbf{x}_t) = \angle(\nabla f^{(l)}(\mathbf{x}_t), \nabla f^{(1-l)}(\mathbf{x}_t))$. If $\|\nabla f^{(1-l)}(\mathbf{x}_t)\| \neq 0$ and $\cos(\alpha(\mathbf{x}_t)) > -1/C_t$ for all iterations t, the iterates of gradient descent with step size $\eta_t = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ satisfy*

$$\min_{t \in [0, T-1]} \|\nabla f^{(l)}(\mathbf{x}_t)\|^2 \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right).$$

[a] *i.e.* $\nabla f^{(l)}$ is L_2-Lipschitz continuous.

We stress that the upper bound $\mathcal{O}(\frac{1}{\sqrt{T}})$ depends on the imbalance. We show it explicitly in App. C (constant C_t).

Theorem 4.1 requires a restrictive condition on the angle $\alpha(\mathbf{x}_t)$ in order to guarantee a decrease of the loss of both classes; we find that (App. C), in order to have a strictly decreasing loss for both classes, the angle between the per-class gradients must meet the condition

$$1 + \cos(\alpha(\mathbf{x}_t)) C_t > 0. \tag{1}$$

We therefore see that this condition can not be satisfied when considering worst-case guarantees, especially when C_t is large. While the above result provides an upper bound, we discuss the tightness of this condition in App. C, effectively demonstrating that one can not avoid this problem in the general case: GD can not guarantee monotonic convergence for each class even for convex functions. Interestingly, condition (1) has an intuitive meaning. Indeed, one can see that it depends on the ratio of the norms of the per-class gradients denoted by C_t, which at least in the initial phases of learning is proportional to the imbalance ratio. In the case where the two norms are equal, Eq. (1) is trivially satisfied *almost always* if the classes are equally difficult. On the contrary, if one gradient norm dominates the other, GD will only minimize the gradient of one class, until it gets to a point where the two gradient norms start having comparable values. As we will see in our experimental results, this leads to a suboptimal behavior if one equally cares about the loss of each class. Building on this finding, we will now introduce a variant of GD which compensates the asymmetry in the gradients. Specifically, we will show that the restrictive condition required in Eq. (1) can essentially be removed.

4.3. Per-Class Normalized Gradient Descent

Algorithmic solutions to class imbalance typically rescale the contribution of each gradient (Anand et al., 1993). Inspired by this prior work we present an algorithm named PCNGD that, starting from an initial guess $\mathbf{x}_0$, iteratively updates the parameter $\mathbf{x}_t$ as follows,

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \left(\frac{\nabla f^{(0)}(\mathbf{x}_t)}{\|\nabla f^{(0)}(\mathbf{x}_t)\|} + \frac{\nabla f^{(1)}(\mathbf{x}_t)}{\|\nabla f^{(1)}(\mathbf{x}_t)\|} \right). \tag{PCNGD}$$

This algorithm forces the updates to be independent of the gradient norm related to each class, but it does not act on the angle between the gradients, so the size of the steps depends on the angle between the gradients.

We want to study the convergence behavior of the loss corresponding to each class. We derive such an analysis in the broad setting where the loss function is smooth and non-convex.

Theorem 4.2 (Informal)**.** *Assume that each $f^{(l)}$ for $l = 0, 1$ is L_1-Lipschitz and L_2-smooth. Then if $\cos \alpha(\mathbf{x}_t) \neq -1$ for all iterations t, the iterates of PCNGD with step size $\eta_t = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ satisfy*

$$\min_{t \in [0, T-1]} \|\nabla f^{(l)}(\mathbf{x}_t)\| \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right).$$

[3] Step imbalance: There is a set of majority classes each with n_1 elements, and a set of minority classes, all with n_2 elements.

Proof sketch for $f^{(0)}$. Since $f^{(0)}$ is L_2-smooth, we have

$$\begin{aligned} & f^{(0)}(\mathbf{x}_{t+1}) \\ & \leq f^{(0)}(\mathbf{x}_t) + \langle \nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \tfrac{L_2}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\ & \leq f^{(0)}(\mathbf{x}_t) - \eta_t (1 + \cos\alpha(\mathbf{x}_t)) \|\nabla f^{(0)}(\mathbf{x}_t)\| + 2L_2 \eta_t^2, \end{aligned}$$

where we used the update step of PCNGD in the second equation.

Let $\omega_{\min} := \min_{t \in [0, T-1]} (1 + \cos\alpha(\mathbf{x}_t))$. By rearranging the terms in the equation above, we get

$$\|\nabla f^{(0)}(\mathbf{x}_t)\| \leq \frac{1}{\eta_t \omega_{\min}} [f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + \frac{2L_2 \eta_t}{\omega_{\min}}$$

Taking the minimum over t, we get

$$\min_{s \in [0, T-1]} \|\nabla f^{(0)}(\mathbf{x}_s)\| \leq \tfrac{1}{T} \sum_{t=0}^{T-1} \|\nabla f^{(0)}(\mathbf{x}_t)\|.$$

We conclude the proof by combining the last two equations and by choosing the step size stated in the theorem.

□

The full proof is given in App. D for different step-size schedules and for randomized iterates (as done in (Ghadimi and Lan, 2013)). An important feature of the above theorem is that it guarantees that each per-class loss function $f^{(l)}$decreases at the same rate. We note that the assumption $\cos\alpha(\mathbf{x}_t) \neq -1$ implies that the two gradients are not allowed to be in completely opposite directions. This condition is a much milder assumption than the one required for GD in Theorem 4.1 (it is as restrictive as in balanced learning, Eq. (1)), and does not become more restrictive when the class imbalance increases.

Note that for a small enough step size, the per-class loss decrease is monotonic [Eq. (25)]. This is desirable since it means that the MID is avoided. This does not however guarantee convergence to a global minimum, but we can ensure a per-class monotonous convergence to a global minimizer under an additional assumption shown next.

Gradient-dominated functions We prove convergence of PCNGD for a class of gradient-dominated functions (Nesterov and Polyak, 2006) which are related to the Polyak-Łojasiewicz (PL) condition (Karimi et al., 2016) that has been shown to hold for overparametrized neural networks (Liu et al., 2022). Instead of requiring this variant of the PL condition to hold for f, we require it to hold for each class separately. Specifically, we say that a function satisfies the *class-GD* inequality if the following holds for each class l,

$$\frac{1}{2} \|\nabla f^{(l)}(\mathbf{x})\| \geq \mu^{(l)} |f^{(l)}(\mathbf{x}) - f^{(l)}(\mathbf{x}^*)|, \quad \forall \mathbf{x} \in \mathbb{R}^m,$$

for some constant $\mu^{(l)} > 0$. For finite-sum objective functions, the class-GD inequality implies the gradient dominated inequality.

Theorem 4.3. *Assume that each $f^{(l)}$ for $l \in \{0, 1\}$ is L_2-smooth and μ-class-GD. Then if $\cos\alpha(\mathbf{x}_t) \neq -1$ for all iterations $t \in [0, T-1]$, the iterates of PCNGD with a decreasing step satisfy*

$$f^{(l)}(\mathbf{x}_T) - f_*^{(l)} \leq \mathcal{O}\left(\tfrac{1}{T}\right), \tag{2}$$

for each $l \in \{0, 1\}$, with $f_*^{(l)} = \min_{\mathbf{x} \in \mathbb{R}^d} f^{(l)}(\mathbf{x})$.

We prove theorem 4.3 in App. D.

4.4. Empirical evidence in favor of PCNGD

We consolidate the findings of our paper with experiments, and provide our code on GitHub (see also App. G).

In Fig. 1 we compare GD and PCNGD with a 60:1 imbalance. In GD, within the first epoch, the recall of the majority class goes to 1, and that of the minority class drops to 0. As shown in Sec. 4.2, the intuition behind this is that with GD in an unbalanced dataset, the gradient points predominantly toward the direction dictated by the majority class. Since the norm of the gradient related to a class scales with its abundance, in order to appreciate the signal of the minority class, the majority class gradient needs to become smaller by an amount that scales with the imbalance ratio ρ.[4] With PCNGD, instead, the MID is suppressed, since both classes are optimized from the beginning of the run. After the first ~ 200 epochs, while the model trained with GD is not yet learning, the PCNGD model already surpasses the peak performance achieved by GD within the entire 3500 epochs of the experiment.

We confirm these observations over several architectures and datasets. We used three different architectures, which we name *Mod1*, *Mod2*, *Mod3*, and five different imbalanced binary and multiclass datasets: *Bi7a*, *Bi7b*, *Bi60*, *Mul10*, *Mul100*. In all cases, while the training set is unbalanced, the test set is balanced. We give full details about the models in App. G.1 and about the datasets in App. G.2. The results of the experiments are reported in App. H.1

4.5. On the better generalization of PCN algorithms

In the experiments shown in Figs 1, 3, and S7, PCNGD not only has a better training performance, but it also has a better test performance. To rationalize the better generalization of PCNGD, we can look at the fixed points of the dynamics. In GD, fixed points satisfy $\nabla f_{n_0}^{(0)} = -\nabla f_{n_1}^{(1)}$. If the training set is imbalanced and the per-class gradients are not exactly zero, this will imply that the minor-

[4] Therefore, we expect that the time required to learn the minority class scales with ρ.

ity gradients are about a factor ρ larger than the majority ones, which is not what we would expect from a balanced dataset. On the contrary, PCNGD allows an infinitely larger quantity of fixed points, since now the condition becomes $\nabla f^{(0)}_{n_0} = -\gamma \nabla f^{(1)}_{n_1}, \forall \gamma > 0$. If on one side this makes us expect a higher variability in the found solutions (which might be beneficial for ensembling (Kyathanahally et al., 2022) or stochastic weighted averaging (Izmailov et al., 2018)), on the other it means that the fixed points are insensitive to the data imbalance.

Another reason for a better generalization can be taken from (Ye et al., 2021), who argue that the MID itself induces overfitting. Therefore, eliminating the MID can on its own be a reason for an improved test performance.

5. SGD: Data Imbalance affects both intensity and direction of the signal

To elucidate the effect of imbalance on SGD, we turn our attention toward a stochastic variant of PCNGD. Perhaps the most obvious way to adapt Eq. (PCNGD) to a stochastic setting is to replace the full gradient by a stochastic estimate computed over a mini-batch, with the condition that at least one element of each class is present in each minibatch.[5] We call the resulting algorithm Per-Class Normalized SGD (PCNSGD), whose update is defined as

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \left(\frac{\nabla_{\tilde{n}} f^{(0)}(\mathbf{x}_t)}{\|\nabla_{\tilde{n}} f^{(0)}(\mathbf{x}_t)\|} + \frac{\nabla_{\tilde{n}} f^{(1)}(\mathbf{x}_t)}{\|\nabla_{\tilde{n}} f^{(1)}(\mathbf{x}_t)\|} \right), \tag{PCNSGD}$$

where $\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$ is the gradient over a random batch of size $\tilde{n}$, taking only examples belonging to the class l.

5.1. Imbalance causes directional noise

In contrast to PCNGD, we empirically observed that PCNSGD still suffers dramatically from the MID, despite the correction on the gradient norms (Figs. 4 and 5–orange curves).

To understand why normalizing the per-class gradient norms does not have the same benefit as with GD, we take a closer look at the computation of the stochastic gradients. To do so, we define $\nabla f^{(l)}_{\infty}(\mathbf{x}_t)$ as the mean gradient direction corresponding to class l on the population loss, and we call $\nabla f^{(l)}_{n_l}(\mathbf{x}_t)$ the full-batch mean gradient relative to class l. More generally, given an arbitrary set S of $\tilde{n}_l$ elements, all belonging to the same generic class l, the notation $\nabla f^{(l)}_{\tilde{n}_l}(\mathbf{x}_t)$ indicates:

$$\nabla f^{(l)}_{\tilde{n}_l}(\mathbf{x}_t) := \frac{1}{\tilde{n}_l} \sum_{i \in S} \nabla f_i(\mathbf{x}_t) \tag{3}$$

where $f_i(\mathbf{x}_t)$ is the loss function corresponding to element $i \in S$. If the underlying gradient distribution has a finite

[5]This can be done in several manners, as long as the batch size is larger than the number of classes. For example, one possible approach is to discard the batches that do not contain both classes and reshuffle the data at every epoch.

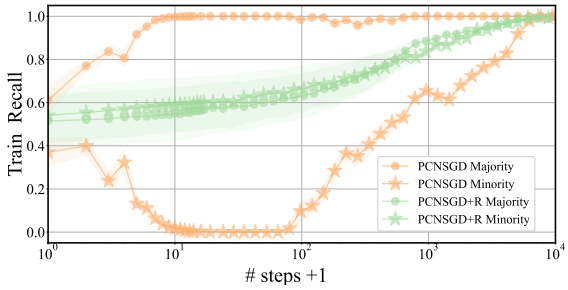

Figure 4. Per-class train recall of PCNSGD and of PCNSGD+R with *Mod1* on the *Bi7a* dataset. With the rescaling proposed in Eq. (8), leading to PCNSGD+R, the MID disappears. The macro-averaged curves are shown in Fig. 5.

variance,[6] by the Central Limit Theorem (CLT) the per-class gradient calculated on $\tilde{n}_l$ examples can be written as fluctuations around the population gradient,

$$\nabla f^{(l)}_{\tilde{n}_l} = \nabla f^{(l)}_{\infty} + \frac{1}{\sqrt{\tilde{n}_l}} \mathbf{Z}^{(l)} + o\left(\frac{1}{\sqrt{\tilde{n}_l}}\right), \tag{4}$$

where $\mathbf{Z}^{(l)}$ is a zero-average multivariate Gaussian random variable whose distribution is fixed by the covariance matrix associated with the gradients of the lth class.

We now consider a single batch; from now on $\tilde{n}_l$ will denote the number of elements, belonging to class l, present in the batch. We want to compare $\nabla f^{(l)}_{\tilde{n}_l}$ with the corresponding full-batch gradient (FBG), in the regime where for both classes the batch size is much smaller than the size of the dataset, *i.e.* $\tilde{n}_l \ll n_{l'}\ \forall l, l' = 0, 1$. In this case, we can use the CLT again, obtaining

$$\nabla f^{(l)}_{\tilde{n}_l} \simeq \nabla f^{(l)}_{n_l} + \frac{1}{\sqrt{\tilde{n}_l}} \mathbf{Z}^{(l)} \equiv G^{(l)}, \tag{5}$$

where we neglected terms of order $1/\sqrt{n_l}$.

Let us now take the updates in Eq. (PCNSGD), and see how normalizing the per-class gradients over the minibatch differs from normalizing them over the full gradient. By using Eq. (5), the PCNSGD steps can be written as

$$\sum_l \frac{\nabla f^{(l)}_{\tilde{n}_l}}{\|\nabla f^{(l)}_{\tilde{n}_l}\|} \simeq \sum_l \left(\frac{\nabla f^{(l)}_{n_l}}{\|G^{(l)}\|} + \frac{\mathbf{Z}^{(l)}}{\sqrt{\tilde{n}_l}\|G^{(l)}\|} \right), \tag{6}$$

where the iterator l goes through the different classes. We want to quantify the projection of the unit vector associated to a generic class l along the corresponding FBG direction, for which Theorem 4.2 shows that we can obtain a monotonous decrease for both classes. The projection of these steps onto the FBG is (see Sec. D.2 for the derivation)

$$\left(\frac{\nabla f^{(l)}_{\tilde{n}_l}}{\|\nabla f^{(l)}_{\tilde{n}_l}\|} \right) \cdot \frac{\nabla f^{(l)}_{n_l}}{\|\nabla f^{(l)}_{n_l}\|} = 1 - \frac{\|\mathbf{Z}^{(l)}\|^2 (\sin(\theta)^2)}{2\tilde{n}_l \|\nabla f^{(l)}_{n_l}\|^2} + o\left(\frac{1}{\tilde{n}_l}\right), \tag{7}$$

[6]This condition can be relaxed by using a generalized CLT (Darling, 1956; Lam et al., 2011).

where θ indicates the angle between $\mathbf{Z}^{(l)}$ and $\nabla f^{(l)}_{n_l}$. Eq. (7) shows that the larger the number of examples of class l in the mini-batch, the closer the steps are to the PCNGD direction. A direct consequence is that, despite the per-class normalization, the two classes do not have the same signal towards the optimal direction: the signal of the minority class is suppressed.

The signal related to each class is attenuated by $\frac{\|\mathbf{Z}^{(l)}\|^2(1-\cos(\theta)^2)}{2\tilde{n}_l\|\nabla f^{(l)}_{n_l}\|^2}$. At the beginning of learning, with random initial conditions, we can expect $\|\nabla f^{(0)}_{\tilde{n}_0}\| \approx \|\nabla f^{(1)}_{\tilde{n}_1}\|$, and we can expect that on average the noise fluctuations have a similar projection onto $\|\nabla f^{(l)}_{n_l}\|$. Consequently, the attenuation of the minority with respect to the majority signal is proportional to the imbalance ratio, $\frac{\tilde{n}_0}{\tilde{n}_1}$. As long as the gradient of the majority class remains large, the minority class updates point far from the direction that would allow the minority class to be optimized. Once the per-class gradient of the majority class converged, by Eq. (5), we have $\frac{\nabla f^{(l)}_{\tilde{n}_l}}{\|\nabla f^{(l)}_{\tilde{n}_l}\|} \simeq \frac{\mathbf{Z}^{(l)}}{\|\mathbf{Z}^{(l)}\|}$, and the signal of the minority class can become relevant. In Secs. 5.3 and 5.4 we will confirm our theory by showing that, if we account for this extra attenuation, the MID disappears.

5.2. Convergence of PCNSGD

We also derive a new convergence analysis for PCNSGD under the common assumption of bounded gradient variance, *i.e.* $\mathbb{E}\|\nabla f^{(l)}(\mathbf{x}_t) - g^{(l)}(\mathbf{x}_t)\|^2 \leq \sigma_l^2$ where we introduced the shorthand notation $g^{(l)}(\mathbf{x}_t) := \nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$ (with $l = 0, 1$) to denote the stochastic gradients, and $\sigma_l > 0$ (the expectation is over the randomness of the algorithm, both in terms of the choice of $\mathbf{x}_0$ and the choice of the mini-batch). We note that for finite-sum objective functions, one can precisely characterize σ_l as a function of the mini-batch size (using standard concentration arguments such as Bernstein's inequality), with $\sigma_l \to 0$ as the batch size approaches the full batch.

Theorem 5.1 (informal)**.** *Assume that each $f^{(l)}$ is L_1-Lipschitz and L_2-smooth and $\mathbb{E}\|\nabla f^{(l)}(\mathbf{x}_t) - g^{(l)}(\mathbf{x}_t)\|^2 \leq \sigma_l^2$ where $\sigma_l > 0$. Then if $\cos\alpha(\mathbf{x}_t) \neq -1$ for all iterations $t \in [0, T-1]$, the iterates of PCNSGD with step size $\eta_t = \mathcal{O}(\frac{1}{\sqrt{T}})$ satisfy*

$$\min_{t\in[0,T-1]} \mathbb{E}\|\nabla f^{(l)}(\mathbf{x}_t)\| \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \mathcal{O}(\sigma_l).$$

Theorem 5.1 shows that the gradient converges at the same rate as in the deterministic case, but only up to a ball of radius σ_l. One way of reducing the size of this ball is to increase the batch size. Another way is to make sure that there is no class l for which σ_l is large. We will later see that one can re-balance the variance σ_l of each class using a simple oversampling technique.

Finally, we note that the monotonicity that characterized PCNGD is now not recovered unless the batch size is large [Eq. (54)]. This is consistent with the analysis of Sec. 5.1.

5.3. Balancing the directional noise

As shown in Secs. 5.1 and 5.2, with SGD, the directional noise induced by class imbalance implies that per-class normalization is not enough to avoid a suppression of the minority-class learning, thus not suppressing the MID. To confirm our theory with experiments, we make use of this knowledge by showing that remedies that compensate for this directional noise induced by imbalance suppress the MID.

Per-class normalization with Rescaling (PCNSGD+R) We want to enforce that both majority and minority mini-batch signals project by the same amount onto the FBGs. This is done by rescaling the minority-class signal by the appropriate amount, calculated from Eq. (7). The rescaling factor (we label with 0 the majority and with 1 the minority class),

$$\alpha = \left(1 - \frac{\|\mathbf{Z}^{(0)}\|^2 \sin(\theta_0)^2}{2\tilde{n}_0\|\nabla f^{(0)}_{n_0}\|^2}\right) \Big/ \left(1 - \frac{\|\mathbf{Z}^{(1)}\|^2 \sin(\theta_1)^2}{2\tilde{n}_1\|\nabla f^{(1)}_{n_1}\|^2}\right), \quad (8)$$

relies on the calculation of the FBGs. To alleviate this burden we only calculate them every 5 steps. The slowness of this algorithm is not a problem, since it is not meant as an alternative to SGD, but rather to provide empirical evidence that the arguments in Sec. 5.1 are correct. Indeed, we show in Fig. 4 (green lines) that, after this further rescaling, the MID is prevented: the minority class recall is monotonically increasing since the beginning of learning, and the majority and minority classes have a similar evolution. In practice, as soon as the signals related to each class are *balanced* (taking into account *all* sources of imbalance, including directional noise), the MID disappears.

Oversampling (O) An alternative way to rebalance the directional noise in a more computationally efficient manner is to impose that all minibatches are composed of the same number of examples from each class. This is similar to minority class oversampling, with the difference that we enforce that every batch is *exactly* split between the classes (see App. F for more details). Contrary to what we saw at the end of Sec. 5.1, in this case, $\frac{\tilde{n}_0}{\tilde{n}_1}$ does not scale with the imbalance ratio, being $\tilde{n}_0 = \tilde{n}_1$.

Per-class normalization with Oversampling (PCNSGD+O) The idea of normalizing per-class gradient contributions can in principle be coupled with other existing methods that are used to deal with class imbalance. Here, we check whether there is a gain in combining per-class normalization with oversampling. In fact, while oversampling equalizes the number of examples per class within each batch, PCN puts an additional constraint on the per-class gradients. For some datasets, this could be influential

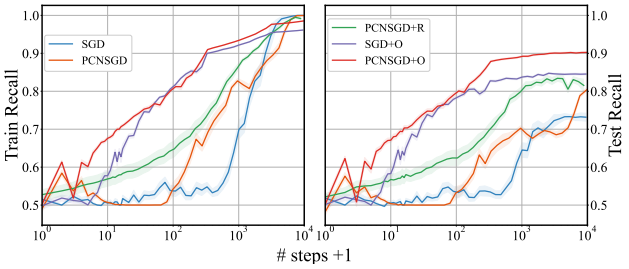

Figure 5. Comparison between different stochastic algorithms with model *Mod1* on the *Bi7a* dataset.

in the initial phases of learning, in the cases in which the gradients related to one class are larger than others. In fact the hardness of different classes may vary (some classes are harder than others, which reflects in the per-class gradients), and this could cause an effect analogous to class imbalance. PCNSGD+O should reduce this difference.

5.4. Experiments with Stochastic Gradients

In Fig. 5 we compare the described algorithms. As per our theory, while SGD and PCNSGD exhibit the MID, PCNSGD+R grows steadily since the beginning of learning. PCNSGD+R is outperformed by SGD+O and PCNSGD+O, which is partly expected, since in PCNSGD+R we only calculate the FBG every 5 steps, and because PCNSGD+R acts on the signal of the minority class, but leaves the signal-to-noise ratio unaltered.[7] From Fig. 5 it seems that PCNSGD+O outperforms SGD+O, since both the recall at short times and the final test recall are higher. This advantage of PCNSGD+O is even more visible with higher imbalance (Fig. S8). However, this is not systematic throughout all the experiments we performed (Tab. 2 and Fig. S9 in App. H). It is instead systematic that the MID is present with SGD dynamics, but it is always avoided when using oversampling. This can be seen from the value of τ in Tab. 2, which is considerably smaller. Extra runs are shown in App. H.

6. Discussion

We presented a new analysis of the learning dynamics of gradient-based algorithms for imbalanced problems, focusing on the learning curves related to each class. Our results highlight the suboptimal behavior of GD, which becomes especially acute for high class imbalance. In particular, at the initial stages of learning, we observe the MID, where the minority classes are classified worse than random. This is particularly problematic when many short runs are needed, such as during hyperparameter tuning. Only in late stages, at times that increase with the imbalance, are the minority classes learned, in agreement with observations by (Ye et al., 2021).

From our analysis, we see that the influence of the imbalance on the dynamics is indirect, and it fades over time. In fact, we find that the learning is not governed by the imbalance ratio itself, but rather by the ratio C_t between per-class gradient norms. This highlights a limitation of methods which renormalize the loss (or the dynamics) in terms of the number of examples per class. While at the beginning of learning C_t is reasonably correlated with the imbalance, it is not so at later stages of learning. This also highlights that it is not class imbalance *per se* that causes suboptimal dynamics, but rather the disparity between per-class gradients. While class imbalance does affect this quantity, there are other sources, *e.g.* related to the difficulty of each class, which could have the same effect. It would therefore be interesting to investigate the relationship between different factors that give rise to a disparity among per-class gradients.

One of our main contributions is a convergence analysis of gradient-based algorithms under class imbalance, with a focus on the *per-class* losses and their gradients. Furthermore, we studied how imbalance differently affects GD and SGD, by analyzing their PCN variants. We found fundamental differences between the deterministic and stochastic settings. While, in the former, class imbalance mostly affects the modulus of the per-class gradients, in the latter it also causes a directional noise, which is stronger with small batch sizes. Consequently, the approaches effective in dealing with these effects need to be different in the two cases. While in the full-batch case, PCN is enough to eliminate the MID, in the stochastic case it is no longer sufficient. Instead, procedures that also balance the effects of the directional noise, as *e.g.* oversampling, allow us to avoid the MID.

Overall, PCN algorithms were successful in addressing the suboptimal MID behavior of gradient descent under class imbalance. We proved that, under certain conditions on learning rate and batch size, they exhibit a monotonous per-class loss decrease, which results in faster training in the initial stages of learning. While per-class normalization directly acts on the dynamical update rule, this is not the case for most of the typically used methods to counter the class imbalance. Therefore, it can (and should) be used in synergy with other methods. We showed this with PCNSGD+O. In this regard, we also emphasize that our analysis is rather general and not related to a specific loss or architecture. It is possible, therefore, to combine the proposed approaches with loss functions typically employed for long-tailed datasets (*e.g.* (Lin et al., 2017; Cao et al., 2019a; Leng et al., 2022)), or with more complex architectures. However, we note that larger architectures have their own potential pitfalls, as evidenced by (Sagawa et al., 2020) who showed that, in the overparametrized regime, the

[7] After the rescaling of the minority class through Eq. (7), all the per-class gradients project equally onto the respective full-batch direction. However, this rescaling also amplifies the minority-class signal in the direction orthogonal to the FBG.

performance of minority groups deteriorates (despite the performance improving overall).

Finally, there are several extensions of interest regarding the analysis of our algorithm, including a specialized analysis to escape saddle points (Jin et al., 2017; Daneshmand et al., 2018) or analyze the effect of momentum (Nesterov, 2003).

7. Acknowledgements

We thank Emma Chollet Ramampiandra for feedback on the manuscript and valuable tips on how to make it more clear. This work was supported by the Swiss National Foundation, SNF grant # 196902.

References

Joaquin Quiñonero-Candela, Masashi Sugiyama, Neil D Lawrence, and Anton Schwaighofer. Dataset shift in machine learning. Mit Press, 2009.

Jose G. Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. Pattern Recognition, 45(1):521–530, 2012. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2011.06.019. URL https://www.sciencedirect.com/science/article/pii/S0031320311002901.

Grant Van Horn and Pietro Perona. The devil is in the tails: Fine-grained classification in the wild. arXiv preprint arXiv:1709.01450, 2017.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, pages 954–959, 2020.

Daniel D'souza, Zach Nussbaum, Chirag Agarwal, and Sara Hooker. A tale of two long tails. arXiv preprint arXiv:2107.13098, 2021.

Shigang Liu, Yu Wang, Jun Zhang, Chao Chen, and Yang Xiang. Addressing the class imbalance problem in twitter spam detection using ensemble learning. Computers & Security, 69:35–49, 2017.

Sara Makki, Zainab Assaghir, Yehia Taher, Rafiqul Haque, Mohand-Said Hacid, and Hassan Zeineddine. An experimental study with imbalanced classification approaches for credit card fraud detection. IEEE Access, 7:93010–93022, 2019.

Sreenath P Kyathanahally, Thomas Hardeman, Ewa Merz, Thea Bulas, Marta Reyes, Peter Isles, Francesco Pomati, and Marco Baity-Jesi. Deep learning classification of lake zooplankton. Frontiers in microbiology, page 3226, 2021. doi: 10.3389/fmicb.2021.746297.

Haibo He and Edwardo A Garcia. Learning from imbalanced data. IEEE Transactions on knowledge and data engineering, 21(9):1263–1284, 2009.

Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5375–5384, 2016.

Nathalie Japkowicz. The class imbalance problem: Significance and strategies. In Proc. of the Int'l Conf. on Artificial Intelligence, volume 56. Citeseer, 2000.

Shaden Alshammari, Yu-Xiong Wang, Deva Ramanan, and Shu Kong. Long-tailed recognition via weight balancing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6897–6907, 2022.

Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. Long-tailed classification by keeping the good and removing the bad momentum causal effect. Advances in Neural Information Processing Systems, 33:1513–1524, 2020.

R. Anand, K.G. Mehrotra, C.K. Mohan, and S. Ranka. An improved algorithm for neural network classification of imbalanced training sets. IEEE Transactions on Neural Networks, 4(6):962–969, 1993. doi: 10.1109/72.286891.

Han-Jia Ye, De-Chuan Zhan, and Wei-Lun Chao. Procrustean training for imbalanced deep learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 92–102, 2021.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In International conference on machine learning, pages 794–803. PMLR, 2018.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. Advances in Neural Information Processing Systems, 33:5824–5836, 2020.

Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. SIAM Journal on Optimization, 23(4):2341–2368, 2013.

Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. Mathematical Programming, 108(1):177–205, 2006.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In

Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 795–811. Springer, 2016.

Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. Applied and Computational Harmonic Analysis, 59:85–116, 2022.

S Kyathanahally, T Hardeman, M Reyes, E Merz, T Bulas, F Pomati, and M Baity-Jesi. Ensembles of vision transformers as a new paradigm for automated classification in ecology. arXiv preprint arXiv:2203.01726, 2022.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407, 2018.

Donald A Darling. Bv gnedenko and an kolmogorov, limit distributions for sums of independent random variables. Bulletin of the American Mathematical Society, 62(1): 50–52, 1956.

Henry Lam, Jose Blanchet, Damian Burch, and Martin Z Bazant. Corrections to the central limit theorem for heavy-tailed probability densities. Journal of Theoretical Probability, 24(4):895–927, 2011.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision, pages 2980–2988, 2017.

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. Advances in neural information processing systems, 32, 2019a.

Zhaoqi Leng, Mingxing Tan, Chenxi Liu, Ekin Dogus Cubuk, Xiaojie Shi, Shuyang Cheng, and Dragomir Anguelov. Polyloss: A polynomial expansion perspective of classification loss functions. arXiv preprint arXiv:2204.12511, 2022.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In International Conference on Machine Learning, pages 8346–8356. PMLR, 2020.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In International conference on machine learning, pages 1724–1732. PMLR, 2017.

Hadi Daneshmand, Jonas Kohler, Aurelien Lucchi, and Thomas Hofmann. Escaping saddles with stochastic gradients. In International Conference on Machine Learning, pages 1155–1164. PMLR, 2018.

Yurii Nesterov. Introductory lectures on convex optimization: A basic course, volume 87. Springer Science & Business Media, 2003.

Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. ACM SIGKDD explorations newsletter, 6(1):1–6, 2004.

Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. Journal of Big Data, 6(1):1–54, 2019.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16:321–357, 2002.

Inderjeet Mani and I Zhang. knn approach to unbalanced data distributions: a case study involving information extraction. In Proceedings of workshop on learning from imbalanced datasets, volume 126. ICML United States, 2003.

Hsin-Ping Chou, Shih-Chieh Chang, Jia-Yu Pan, Wei Wei, and Da-Cheng Juan. Remix: rebalanced mixup. In European Conference on Computer Vision, pages 95–110. Springer, 2020.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In International Conference on Machine Learning, pages 4334–4343. PMLR, 2018.

Jing An, Lexing Ying, and Yuhua Zhu. Why resampling outperforms reweighting for correcting sampling bias with stochastic gradients. arXiv preprint arXiv:2009.13447, 2020.

Qi Dong, Shaogang Gong, and Xiatian Zhu. Imbalanced deep learning by minority class incremental rectification. IEEE transactions on pattern analysis and machine intelligence, 41(6):1367–1381, 2018.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. arXiv:2007.07314, 2020.

Jianggang Zhu, Zheng Wang, Jingjing Chen, Yi-Ping Phoebe Chen, and Yu-Gang Jiang. Balanced contrastive learning for long-tailed visual recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6908–6917, 2022.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In International Conference on Machine Learning, pages 2712–2721. PMLR, 2019.

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. arXiv:1906.07413, 2019b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

Yuxin Wu and Kaiming He. Group normalization. In Proceedings of the European conference on computer vision (ECCV), pages 3–19, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning, pages 448–456. PMLR, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. URL https://www.cs.toronto.edu/~kriz/cifar.html.

Raman Arora, Sanjeev Arora, Joan Bruna, Nadav Cohen, Simon Du, Rong Ge, Suriya Gunasekar, C Jin, Jason Lee, Tengyu Ma, et al. Theory of deep learning, 2020.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. ACM Computing Surveys (CSUR), 54(6):1–35, 2021.

Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Conference on fairness, accountability and transparency, pages 77–91. PMLR, 2018.

David Danks and Alex John London. Algorithmic bias in autonomous systems. In Ijcai, volume 17, pages 4691–4697, 2017.

Terrance De Vries, Ishan Misra, Changhan Wang, and Laurens Van der Maaten. Does object recognition work for everyone? In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pages 52–59, 2019.

Adrienne Yapo and Joseph Weiss. Ethical implications of bias in machine learning. Proceedings of the 51st Hawaii International Conference on System Sciences, page 5365, 2018.

A. Additional Related Work

We now elaborate more on existing strategies that have been used to mitigate class imbalance. As mentioned in Sec. 1, we identify three broad classes of methods to handle class imbalance.

First, at the data level, resampling is arguably the most known approach where one either under- or over-samples datapoints in order to achieve a class-balanced distribution (He and Garcia, 2009; Huang et al., 2016). These two methods have obvious drawbacks: while undersampling discards training data, oversampling causes longer training time and might lead to overfitting (Chawla et al., 2004; Johnson and Khoshgoftaar, 2019). Various techniques use more advanced sampling techniques, such as rebalancing the dataset through synthetic data (Chawla et al., 2002), relying on a K-NN classifier (Mani and Zhang, 2003) or other distance metrics to select samples near the boundary between classes, or reassigning the labels favoring minority classes (Chou et al., 2020).

A second class of approaches acts on the loss function. Arguably the simplest approach in this category is class reweighting (cost-sensitive training), which assigns a larger weight in the loss function to examples from minority classes. This can be done by rescaling the loss related to each example by the frequency of its class (Japkowicz, 2000), or by making use of an auxiliary balanced validation dataset (Ren et al., 2018). However, in overparameterized regimes, the reweighting approach may be ineffective in improving the performance related to minority groups (Sagawa et al., 2020). In addition, it has been observed that resampling often outperforms reweighting significantly (see (An et al., 2020) and references therein).
Another strategy is to increase the margins of the loss, which was shown to be effective in reducing the imbalance problem (Huang et al., 2016; Dong et al., 2018; Menon et al., 2020). Other recently successful techniques include tailored regularizers (Alshammari et al., 2022) or adapting the loss to allow for supervised contrastive learning (Zhu et al., 2022).

A final way to mitigate class imbalance is to rely on algorithmic solutions. This can be done by: acting on the initial conditions, given empirical evidence that pretraining can be beneficial (Hendrycks et al., 2019); acting on the loss margins, but doing this dynamically according to a learning schedule (Cao et al., 2019b); perturbing the inputs of the majority class (Ye et al., 2021); by using an alternative momentum which properly suits imbalance (Tang et al., 2020); or by adapting the direction of the gradient steps in order to suppress the domination of the majority class (Anand et al., 1993).

As discussed in (Johnson and Khoshgoftaar, 2019), all these methods achieve different levels of performance depending on a multitude of conditions (classifier, performance metric, ...). Importantly, these methods do not have well-understood convergence guarantees (in contrast to the PCN algorithms studied in this paper).

B. Notation

We summarize here some of the notation that is employed in the paper:

- $|\cdot|$: Absolute value of a scalar. When referred to a set, it denotes instead its cardinality, *i.e.* the number of elements that make up the set
- $\|\cdot\|$: L_2 Norm
- $\%$: Modulo operator
- $\mathbf{x}\cdot\mathbf{y} = \langle \mathbf{x}, \mathbf{y}\rangle$: inner product between two vectors $\mathbf{x}, \mathbf{y}$
- $\angle(\mathbf{x}, \mathbf{y})$: angle between two vectors $\mathbf{x}, \mathbf{y}$
- $C_l = \{i \mid y_i = l\}$: subgroup of indices belonging to class l
- $C_t = \frac{\|\nabla f^{(1)}(\mathbf{x}_t)\|}{\|\nabla f^{(0)}(\mathbf{x}_t)\|}$
- $C_{\mu,l} = \min_{t\in[0,T-1]} 2\mu^{(l)}(1+\cos\alpha(\mathbf{x}_t))$
- $\mathcal{D} = (\xi_i, y_i)_{i=1}^n$: dataset
- $\mathcal{D}_l = (\xi_i, y_i)_{i\in C_l}$: Subgroup of $\mathcal{D}$ elements belonging to class l
- $D_0^{(l)} = [f^{(l)}(\mathbf{x}_0) - f_*^{(l)}]$: in the mini-batch case the same quantity is computed as an expectation value, *i.e.* $D_0^{(l)} = \mathbb{E}[f^{(l)}(\mathbf{x}_0) - f_*^{(l)}]$

- $f(\mathbf{x}_t) \equiv \frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}} f_i(\mathbf{x}_t)$: Average loss function calculated over all elements in the dataset.
- $f^{(l)}(\mathbf{x}_t) = \frac{1}{|\mathcal{D}|}\sum_{i\in C_l} f_i(\mathbf{x}_t)$: contribution to $f(\mathbf{x}_t)$ from class l
- $f_*^{(l)} = \min_{\mathbf{x}\in\mathbb{R}^d} f^{(l)}(\mathbf{x})$
- $G^{(l)} = \nabla f_{n_l}^{(l)} + \frac{1}{\sqrt{n_l}}\mathbf{Z}^{(l)}$
- $g^{(l)}(\mathbf{x}_t) = \nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$: see $\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$
- L: total number of classes
- L_1 : Lipschitz constant of the generic per-class loss $f^{(l)}$
- L_2 : Smooth constant of the generic per-class loss $f^{(l)}$
- N_e : total number of simulation epochs
- n : dataset size, *i.e.* the number of elements that makes up the dataset
- n_l : Number of examples in the dataset belonging to class l
- $\tilde{n} = |\gamma_t|$: batch size at step t; note that for full-batch algorithms (*e.g.* GD) $|\gamma_t| = n$
- $\tilde{n}_l$: Number of examples in the batch belonging to class l
- $\mathbf{x}_t$: set of network parameters at time t
- $y_i \in [0, \ldots, L-1]$: label ; by convention, label " 0 " identifies the majority class of the dataset
- $\mathbf{Z}^{(l)}$: zero-average multivariate Gaussian random variable whose distribution is fixed by the covariance matrix associated with the gradients of the lth class
- $\alpha(\mathbf{x}_t) = \angle(\nabla f^{(0)}(\mathbf{x}_t), \nabla f^{(1)}(\mathbf{x}_t))$: angle between the 2 gradients per-class in binary classification problems (in multi-class problems it is necessary to introduce additional indices to identify the classes considered in the angle calculation)
- $\beta_t = \cos(\alpha(\mathbf{x}_t))$
- γ_t: batch selected at step t
- $\{\gamma_t\}_e$: set of batches defined for the epoch e
- η_t : learning rate
- $\mu^{(l)}$: Per-class gradient dominated constant
- $\xi_i \in \mathbb{R}^d$: input vector
- $\rho_k = \frac{n_k}{n}$: fraction of elements in the dataset belonging to class k
- σ_l^2: finite upper bound for $\mathbb{E}\|\nabla f^{(l)}(\mathbf{x}_t) - g^{(l)}(\mathbf{x}_t)\|^2$
- ω_{min} is defined differently in the various theorems; in each of them the definition is reported before the beginning of the proof
- $\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$: is the gradient over a random batch of size $\tilde{n}$, for examples belonging to the class l
- $\nabla f_{\tilde{n}_l}^{(l)}(\mathbf{x}_t) = \frac{1}{\tilde{n}_l}\sum_{\substack{i\in C_l\\ i\in\gamma_t}} \nabla f_i(\mathbf{x}_t)$: Average gradient computed on the elements in the batch belonging to class l. Note that $\nabla f_{\tilde{n}_l}^{(l)}(\mathbf{x}_t)$ and $\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$ differ by only one factor in particular $\nabla f_{\tilde{n}_l}^{(l)}(\mathbf{x}_t) = \frac{\tilde{n}}{\tilde{n}_l}\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$

C. Convergence rate GD

In this section, we derive a worst-case convergence analysis of the GD iterates for the per-class loss. Similar analyses are typically performed on a loss that combines all classes. We refer the reader to (Nesterov, 2003) for convex functions, to (Ghadimi and Lan, 2013) for non-convex (but smooth) functions, and to (Karimi et al., 2016) for PL functions.

We start by stating a more formal version of Theorem 4.1, followed by its proof. For the sake of clarity, we prove our results for the binary case, where the number of classes $L = 2$.

In the following, we denote by $f_*^{(l)} = \min_{\mathbf{x}\in\mathbb{R}^d} f^{(l)}(\mathbf{x})$ the global minimum loss for each class l (assuming each loss $f^{(l)}$ is lower bounded). Note that we do not require the minimum itself for each class to be unique.

Theorem C.1 (Formal version of Theorem 4.1)**.** *Assume that each $f^{(l)}$ is L_1-Lipshitz and L_2-smooth and let $\alpha(\mathbf{x}_t) = \angle(\nabla f^{(l)}(\mathbf{x}_t), \nabla f^{(1-l)}(\mathbf{x}_t))$. If $\|\nabla f^{(1-l)}(\mathbf{x}_t)\| \neq 0$ and $\cos(\alpha(\mathbf{x}_t)) > -\frac{1}{C_t}$ where $C_t := \frac{\|\nabla f^{(1-l)}(\mathbf{x}_t)\|}{\|\nabla f^{(l)}(\mathbf{x}_t)\|}$ for all iterations $t \in [0, T-1]$, the iterates of gradient descent with step size $\eta_t = \min\left(\frac{1+\cos(\alpha(\mathbf{x}_t))C_t}{2(1+C_t^2)L_2}, \frac{c}{\sqrt{T}}\right)$ where $c > 0$ satisfy*

$$\min_{s\in[0,T-1]} \|\nabla f^{(l)}(\mathbf{x}_s)\|^2 \leq \frac{2(1+C_{\max})L_2}{(\omega_{min}^{(l)})^2 T} D_0^{(l)} + \frac{1}{\omega_{min}^{(l)} c\sqrt{T}} D_0^{(l)},$$

for each $l \in \{0, 1\}$, where $D_0^{(l)} = [f^{(l)}(\mathbf{x}_0) - f_*^{(l)}]$, $\omega_{min}^{(l)} = \min_{t\in[0,T-1]} 1 + \cos(\alpha(\mathbf{x}_t))C_t$, and $C_{\max} = \max_{t\in[0,T-1]} C_t^2$.

Proof. We only write the detail for the function $f^{(0)}$ since the proof for $f^{(1)}$ is identical.

Since each function $f^{(0)}$ is L_2-smooth, we have

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) + \langle \nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t\rangle + \frac{L_2}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \qquad (9)\\
&= f^{(0)}(\mathbf{x}_t) - \eta_t\langle \nabla f^{(0)}(\mathbf{x}_t), \nabla f(\mathbf{x}_t)\rangle + \frac{L_2\eta_t^2}{2}\|\nabla f(\mathbf{x}_t)\|^2\\
&\leq f^{(0)}(\mathbf{x}_t) - \eta_t\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta_t\langle \nabla f^{(0)}(\mathbf{x}_t), \nabla f^{(1)}(\mathbf{x}_t)\rangle + L_2\eta_t^2\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 + L_2\eta_t^2\left\|\nabla f^{(1)}(\mathbf{x}_t)\right\|^2\\
&= f^{(0)}(\mathbf{x}_t) - \eta_t(1 - L_2\eta_t)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta_t\langle \nabla f^{(0)}(\mathbf{x}_t), \nabla f^{(1)}(\mathbf{x}_t)\rangle + L_2\eta_t^2\left\|\nabla f^{(1)}(\mathbf{x}_t)\right\|^2\\
&= f^{(0)}(\mathbf{x}_t) - \eta_t(1 - L_2\eta_t)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta_t\cos(\alpha(\mathbf{x}_t))\|\nabla f^{(0)}(\mathbf{x}_t)\|\|\nabla f^{(1)}(\mathbf{x}_t)\| + L_2\eta_t^2\left\|\nabla f^{(1)}(\mathbf{x}_t)\right\|^2,
\end{aligned}$$

where the inequality in the third line is simply due to $\|\mathbf{x} + \mathbf{y}\|_2^2 \leq 2\|\mathbf{x}\|_2^2 + 2\|\mathbf{y}\|_2^2$ for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.

Let $C_t := \frac{\|\nabla f^{(1)}(\mathbf{x}_t)\|}{\|\nabla f^{(0)}(\mathbf{x}_t)\|}$. We get

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) - \eta_t\left(1 + \cos(\alpha(\mathbf{x}_t))C_t - L_2\eta_t - L_2\eta_t C_t^2\right)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2\\
\implies \eta_t\left(1 + \cos(\alpha(\mathbf{x}_t))C_t - (1 + C_t^2)L_2\eta_t\right)&\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 \leq f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1}). \qquad (10)
\end{aligned}$$

At this point, we see that we need the following condition on the angle $\alpha(\mathbf{x}_t)$:

$$\cos(\alpha(\mathbf{x}_t)) > -\frac{1}{C_t}. \qquad (11)$$

Taking $\eta_t = \min\left(\frac{1+\cos(\alpha(\mathbf{x}_t))C_t}{2(1+C_t^2)L_2}, \frac{c}{\sqrt{T}}\right)$ (where $c > 0$), we have

$$\eta_t(1 + \cos(\alpha(\mathbf{x}_t))C_t) - \eta_t^2(1 + C_t^2)L_2 \geq \frac{\eta_t}{2}(1 + \cos(\alpha(\mathbf{x}_t))C_t),$$

therefore

$$\frac{\eta_t}{2}(1+\cos(\alpha(\mathbf{x}_t))C_t)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 \leq f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1}). \tag{12}$$

Let $\omega_t := 1+\cos(\alpha(\mathbf{x}_t))C_t$, then

$$\begin{aligned}\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 &\leq \frac{2}{\omega_t\eta_t}(f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})) \\ &\leq \left(\max_t \frac{1}{\omega_t}\right)\frac{2}{\eta_t}(f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})).\end{aligned} \tag{13}$$

Let $\omega_{min}^{(0)} = \min_{t\in[0,T-1]}\omega_t$ and $C_{\max} := \max_{t\in[0,T-1]} C_t^2$. By summing from $t=0$ to $T-1$,

$$\begin{aligned}\min_{s\in[0,T-1]}\|\nabla f^{(0)}(\mathbf{x}_s)\|^2 &\leq \frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f^{(0)}(\mathbf{x}_t)\|^2 \\ &\leq \frac{1}{\omega_{min}^{(0)}T}\max\left(\frac{2(1+C_{\max})L_2}{\omega_{min}^{(0)}}, \frac{\sqrt{T}}{c}\right)[f^{(0)}(\mathbf{x}_0) - f^{(0)}(\mathbf{x}_T)] \\ &\leq \frac{1}{\omega_{min}^{(0)}T}\left(\frac{2(1+C_{\max})L_2}{\omega_{min}^{(0)}} + \frac{\sqrt{T}}{c}\right)[f^{(0)}(\mathbf{x}_0) - f_*^{(0)}] \\ &\leq \frac{2(1+C_{\max})L_2}{(\omega_{min}^{(0)})^2 T}[f^{(0)}(\mathbf{x}_0) - f_*^{(0)}] + \frac{1}{\omega_{min}^{(0)}c\sqrt{T}}[f^{(0)}(\mathbf{x}_0) - f_*^{(0)}],\end{aligned} \tag{14}$$

where $f_*^{(0)}$ denotes the global minimum of $f^{(0)}(\mathbf{x})$. Note that Eq. (14) is related to the imbalance, since both c and $\omega_{min}^{(0)}$ depend on C_t, which at least at the beginning of the dynamics depends on the imbalance.

□

We note that the condition required in the theorem, $\cos(\alpha(\mathbf{x}_t)) > -\frac{\|\nabla f^{(0)}(\mathbf{x}_t)\|}{\|\nabla f^{(1)}(\mathbf{x}_t)\|}$ is quite restrictive and might not be satisfied in practice. We discuss this in more detail next.

Tightness of the upper bound Consider a quadratic function with a constant diagonal Hessian where all eigenvalues are equal to L, and for which

$$\begin{aligned}f^{(0)}(\mathbf{x}_{t+1}) &= f^{(0)}(\mathbf{x}_t) + \langle\nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t\rangle + \frac{L}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\ &= f^{(0)}(\mathbf{x}_t) - \eta_t\langle\nabla f^{(0)}(\mathbf{x}_t), \nabla f(\mathbf{x}_t)\rangle + \frac{L\eta_t^2}{2}\|\nabla f(\mathbf{x}_t)\|^2 \\ &= f^{(0)}(\mathbf{x}_t) - \eta_t\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta_t\langle\nabla f^{(0)}(\mathbf{x}_t), \nabla f^{(1)}(\mathbf{x}_t)\rangle + \frac{L\eta_t^2}{2}\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 + \frac{L\eta_t^2}{2}\left\|\nabla f^{(1)}(\mathbf{x}_t)\right\|^2 \\ &\quad + L\eta_t^2\langle\nabla f^{(0)}(\mathbf{x}_t), \nabla f^{(1)}(\mathbf{x}_t)\rangle \\ &= f^{(0)}(\mathbf{x}_t) - \eta_t\left(1 - \frac{L\eta_t}{2}\right)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta_t(1 - L\eta_t)\langle\nabla f^{(0)}(\mathbf{x}_t), \nabla f^{(1)}(\mathbf{x}_t)\rangle + \frac{L\eta_t^2}{2}\left\|\nabla f^{(1)}(\mathbf{x}_t)\right\|^2.\end{aligned}$$

Let $C_t := \frac{\|\nabla f^{(1)}(\mathbf{x}_t)\|}{\|\nabla f^{(0)}(\mathbf{x}_t)\|}$. We get

$$f^{(0)}(\mathbf{x}_{t+1}) = f^{(0)}(\mathbf{x}_t) - \eta_t\left(1 - L\eta_t/2 - L\eta_t C_t^2/2 + (1 - L\eta_t)\cos(\alpha(\mathbf{x}_t))C_t\right)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2. \tag{15}$$

We see that in order to decrease $f^{(0)}$, we require the following condition to hold:

$$1 - (1 + C_t^2)L\eta_t/2 + (1 - L\eta_t)\cos(\alpha(\mathbf{x}_t))C_t > 0.$$

Taking $\eta_t = \frac{c}{\sqrt{T}}$, we see that we require the following condition on the step-size:

$$c \leq \frac{(1+\cos(\alpha(\mathbf{x}_t))C_t)\sqrt{T}}{(1+C_t^2)L/2 + L\cos(\alpha(\mathbf{x}_t))C_t},$$

which in turns require $1+\cos(\alpha(\mathbf{x}_t))C_t > 0$. We therefore see that this condition can not be avoided when considering worst-case guarantees.

Alternate step size We also derive a convergence result for the case where the step size does not depend on the total number of iterations T. This allows us to obtain a faster rate of convergence, but we still observe a severe restriction on the angle between the gradient of the two classes.

Theorem C.2. *Assume that each $f^{(l)}$ is L_1-Lipshitz and L_2-smooth. There is a choice of step size that such under **restricted conditions** on the angle $\alpha(\mathbf{x}_t)$, we get*

$$\min_{t\leq T} \|\nabla f^{(l)}(\mathbf{x}_t)\|^2 \leq \frac{f^{(l)}(\mathbf{x}_0) - f_*^{(l)}}{\omega(T+1)}, \tag{16}$$

where $\omega = \min_t \eta(1 \pm |\beta_t|C_t - L_2\eta(1+C_t^2))$.

Proof. Since each function $f^{(0)}$ is L_2-smooth, we have

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) + \langle \nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t\rangle + \frac{L_2}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \qquad (17)\\
&= f^{(0)}(\mathbf{x}_t) - \eta\langle \nabla f^{(0)}(\mathbf{x}_t), \nabla f(\mathbf{x}_t)\rangle + \frac{L_2\eta^2}{2}\|\nabla f(\mathbf{x}_t)\|^2\\
&\leq f^{(0)}(\mathbf{x}_t) - \eta\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta\langle \nabla f^{(0)}(\mathbf{x}_t), \nabla f^{(1)}(\mathbf{x}_t)\rangle + L_2\eta^2\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 + L_2\eta^2\left\|\nabla f^{(1)}(\mathbf{x}_t)\right\|^2\\
&= f^{(0)}(\mathbf{x}_t) - \eta(1-L_2\eta)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta\langle \nabla f^{(0)}(\mathbf{x}_t), \nabla f^{(1)}(\mathbf{x}_t)\rangle + L_2\eta^2\left\|\nabla f^{(1)}(\mathbf{x}_t)\right\|^2\\
&= f^{(0)}(\mathbf{x}_t) - \eta(1-L_2\eta)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta\cos(\alpha(\mathbf{x}_t))\|\nabla f^{(0)}(\mathbf{x}_t)\|\|\nabla f^{(1)}(\mathbf{x}_t)\| + L_2\eta^2\left\|\nabla f^{(1)}(\mathbf{x}_t)\right\|^2.
\end{aligned}$$

Let $\|\nabla f^{(1)}(\mathbf{x}_t)\| = C_t\|\nabla f^{(0)}(\mathbf{x}_t)\|$ for $C_t \geq 0$ and $\beta_t = \cos(\alpha(\mathbf{x}_t))$. Note that C_t depends on time.

Case 1: $\beta_t < 0$

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) - \eta(1-L_2\eta)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 + \eta|\beta_t|C_t\|\nabla f^{(0)}(\mathbf{x}_t)\|^2 + L_2\eta^2C_t^2\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2\\
&= f^{(0)}(\mathbf{x}_t) - \eta(1 - |\beta_t|C_t - L_2\eta(1+C_t^2))\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2. \qquad (18)
\end{aligned}$$

We need

$$(1 - |\beta_t|C_t - L_2\eta(1+C_t^2)) > 0 \implies \eta < \frac{1-|\beta_t|C_t}{L_2(1+C_t^2)}. \tag{19}$$

We also need $C_t < \frac{1}{|\beta_t|}$ for the function to decrease.

Case 2: $\beta_t \geq 0$

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) - \eta(1-L_2\eta)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta|\beta_t|C_t\|\nabla f^{(0)}(\mathbf{x}_t)\|^2 + L_2\eta^2C_t^2\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2\\
&= f^{(0)}(\mathbf{x}_t) - \eta(1 + |\beta_t|C_t - L_2\eta(1+C_t^2))\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2. \qquad (20)
\end{aligned}$$

We need

$$(1 + |\beta_t|C_t - L_2\eta(1 + C_t^2)) > 0 \implies \eta < \frac{1 + |\beta_t|C_t}{L_2(1 + C_t^2)}. \tag{21}$$

Function decrease In both cases, we have an equation of the form

$$f^{(0)}(\mathbf{x}_{t+1}) \leq f^{(0)}(\mathbf{x}_t) - \omega_t \left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2, \tag{22}$$

where $\omega_t = \eta(1 \pm |\beta_t|C_t - L_2\eta(1 + C_t^2))$.

Let $\omega = \min_t \omega_t$. We then sum up the above inequality:

$$\omega \sum_{t=0}^{T} \|\nabla f^{(0)}(\mathbf{x}_t)\|^2 \leq f^{(0)}(\mathbf{x}_0) - f^{(0)}(\mathbf{x}_{T+1}) \leq f^{(0)}(\mathbf{x}_0) - f_*^{(0)}, \tag{23}$$

where $f_*^{(0)}$ is the global minimum loss of class 0.

Finally, we lower bound $\|\nabla f^{(0)}(\mathbf{x}_t)\|$ by its minimum,

$$\min_{t \leq T} \|\nabla f^{(0)}(\mathbf{x}_t)\|^2 \leq \frac{f^{(0)}(\mathbf{x}_0) - f_*^{(0)}}{\omega(T+1)} \tag{24}$$

We see that we are only guaranteed a function decrease if $\omega_t > 0$, *i.e.* $\beta_t \geq 0$ or $\beta_t \leq 0$ and $C_t < \frac{1}{|\beta_t|}$.

□

D. Convergence rate PCNGD

This section contains the formal version of Theorem 4.2, with a detailed proof.

Theorem D.1 (Formal version of Theorem 4.2)**.** *Assume that each $f^{(l)}$ is L_1-Lipschitz and L_2-smooth. Then if $\cos\alpha(\mathbf{x}_t) \neq -1$ for all iterations $t \in [0, T-1]$, the iterates of PCNGD with step size $\eta_t = \frac{c}{\sqrt{T}}$ where $c > 0$ satisfy*

$$\min_{s \in [0, T-1]} \|\nabla f^{(l)}(\mathbf{x}_s)\| \leq \frac{1}{\omega_{\min}\sqrt{T}} \left(\frac{D_0^{(l)}}{c} + 2L_2 c \right),$$

for each $l \in \{0, 1\}$, where $D_0^{(l)} = [f^{(l)}(\mathbf{x}_0) - f_*^{(l)}]$ and $\omega_{\min} := \min_{t \in [0, T-1]}(1 + \cos\alpha(\mathbf{x}_t))$.

Proof. We only write the detail for the function $f^{(0)}$ since the proof for $f^{(1)}$ is identical.

First, since $f^{(0)}$ is L_2-smooth, we have

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) + \langle \nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{L_2}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\
&\leq f^{(0)}(\mathbf{x}_t) - \eta_t \|\nabla f^{(0)}(\mathbf{x}_t)\| - \eta_t \langle \nabla f^{(0)}(\mathbf{x}_t), \frac{\nabla f^{(1)}(\mathbf{x}_t)}{\|\nabla f^{(1)}(\mathbf{x}_t)\|} \rangle \\
&\quad + L_2 \eta_t^2 \left\| \frac{\nabla f^{(0)}(\mathbf{x}_t)}{\|\nabla f^{(0)}(\mathbf{x}_t)\|} \right\|^2 + L_2 \eta_t^2 \left\| \frac{\nabla f^{(1)}(\mathbf{x}_t)}{\|\nabla f^{(1)}(\mathbf{x}_t)\|} \right\|^2 \\
&= f^{(0)}(\mathbf{x}_t) - \eta_t \|\nabla f^{(0)}(\mathbf{x}_t)\| - \eta_t \|\nabla f^{(0)}(\mathbf{x}_t)\| \cos\alpha(\mathbf{x}_t) + 2L_2 \eta_t^2 \\
&= f^{(0)}(\mathbf{x}_t) - \eta_t (1 + \cos\alpha(\mathbf{x}_t)) \|\nabla f^{(0)}(\mathbf{x}_t)\| + 2L_2 \eta_t^2.
\end{aligned} \tag{25}$$

Since, at least at the beginning of training, both $(1+\cos\alpha(\mathbf{x}_t))$ and $\|\nabla f^{(l)}(\mathbf{x}_t)\|$ are strictly positive, Eq. 25 implies that, for a small enough learning rate, the per-class loss at time $t+1$ is smaller than that at time t, so the MID is avoided.

Let $\omega_t := (1+\cos\alpha(\mathbf{x}_t))$ and $\omega_{\min} := \min_{t\in[0,T-1]}(1+\cos\alpha(\mathbf{x}_t))$. By rearranging the terms in the equation above, we get

$$\begin{aligned}
\|\nabla f^{(0)}(\mathbf{x}_t)\| &\leq \frac{1}{\eta_t\omega_t}[f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + \frac{2L_2\eta_t}{\omega_t} \\
&\leq \frac{1}{\eta_t}\left(\max_{t\in[0,T-1]}\frac{1}{\omega_t}\right)[f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + 2L_2\eta_t\left(\max_{t\in[0,T-1]}\frac{1}{\omega_t}\right) \\
&\leq \frac{1}{\eta_t\omega_{\min}}[f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + \frac{2L_2\eta_t}{\omega_{\min}}. \qquad (26)
\end{aligned}$$

Taking the minimum over t, we get

$$\begin{aligned}
\min_{s\in[0,T-1]}\|\nabla f^{(0)}(\mathbf{x}_s)\| &\leq \frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f^{(0)}(\mathbf{x}_t)\| \\
&\leq \frac{1}{c\,\omega_{\min}\sqrt{T}}[f^{(0)}(\mathbf{x}_0) - f^{(0)}(\mathbf{x}_T)] + \frac{2L_2c}{\omega_{\min}\sqrt{T}} \\
&\leq \frac{1}{c\,\omega_{\min}\sqrt{T}}[f^{(0)}(\mathbf{x}_0) - f_*^{(0)}] + \frac{2L_2c}{\omega_{\min}\sqrt{T}} \\
&\leq \frac{1}{\omega_{\min}\sqrt{T}}\left(\frac{1}{c}[f^{(0)}(\mathbf{x}_0) - f_*^{(0)}] + 2L_2c\right). \qquad (27)
\end{aligned}$$

□

Alternate choice of step size We also present a second version of Theorem 4.2 with an alternate choice of step size.

Theorem D.2 (Second formal version of Theorem 4.2)**.** *Assume that each $f^{(l)}$ is L_1-Lipschitz and L_2-smooth. Then if $\cos\alpha(\mathbf{x}_t) \neq -1$ for all iterations $t \in [0, T-1]$, the iterates of PCNGD with step size $\eta_t = \frac{c}{(1+\cos\alpha(\mathbf{x}_t))\sqrt{T}}$ where $c > 0$ satisfy*

$$\min_{s\in[0,T-1]}\|\nabla f^{(l)}(\mathbf{x}_s)\| \leq \frac{1}{\sqrt{T}}\left(\frac{D_0^{(l)}}{c} + \frac{2L_2c}{\omega_{\min}}\right),$$

for each $l \in \{0,1\}$, where $D_0^{(l)} = [f^{(l)}(\mathbf{x}_0) - f_*^{(l)}]$ and $\omega_{\min} := \min_{t\in[0,T-1]}(1+\cos\alpha(\mathbf{x}_t))^2$.

Proof. We only write the detail for the function $f^{(0)}$ since the proof for $f^{(1)}$ is identical.

First, since $f^{(0)}$ is L_2-smooth, we have

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) + \langle \nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{L_2}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\
&\leq f^{(0)}(\mathbf{x}_t) - \eta_t \|\nabla f^{(0)}(\mathbf{x}_t)\| - \eta_t \langle \nabla f^{(0)}(\mathbf{x}_t), \frac{\nabla f^{(1)}(\mathbf{x}_t)}{\|\nabla f^{(1)}(\mathbf{x}_t)\|} \rangle \\
&\quad + L_2 \eta_t^2 \left\| \frac{\nabla f^{(0)}(\mathbf{x}_t)}{\|\nabla f^{(0)}(\mathbf{x}_t)\|} \right\|^2 + L_2 \eta_t^2 \left\| \frac{\nabla f^{(1)}(\mathbf{x}_t)}{\|\nabla f^{(1)}(\mathbf{x}_t)\|} \right\|^2 \\
&= f^{(0)}(\mathbf{x}_t) - \eta_t \|\nabla f^{(0)}(\mathbf{x}_t)\| - \eta_t \|\nabla f^{(0)}(\mathbf{x}_t)\| \cos\alpha(\mathbf{x}_t) + 2L_2 \eta_t^2 \\
&= f^{(0)}(\mathbf{x}_t) - \eta_t (1 + \cos\alpha(\mathbf{x}_t)) \|\nabla f^{(0)}(\mathbf{x}_t)\| + 2L_2 \eta_t^2 \\
&= f^{(0)}(\mathbf{x}_t) - \frac{c}{\sqrt{T}} \|\nabla f^{(0)}(\mathbf{x}_t)\| + 2L_2 \frac{c^2}{T(1 + \cos\alpha(\mathbf{x}_t))^2}.
\end{aligned} \tag{28}$$

Let $\omega_{\min} := \min_{t \in [0, T-1]} (1 + \cos\alpha(\mathbf{x}_t))^2$. By rearranging the terms in the equation above, we get

$$\begin{aligned}
\|\nabla f^{(0)}(\mathbf{x}_t)\| &\leq \frac{\sqrt{T}}{c} [f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + \max_{t \in [0, T-1]} \frac{2L_2 c}{\sqrt{T}(1 + \cos\alpha(\mathbf{x}_t))^2} \\
&\leq \frac{\sqrt{T}}{c} [f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + \frac{2L_2 c}{\sqrt{T}\omega_{\min}}.
\end{aligned} \tag{29}$$

Taking the minimum over t, we get

$$\begin{aligned}
\min_{s \in [0, T-1]} \|\nabla f^{(0)}(\mathbf{x}_s)\| &\leq \frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f^{(0)}(\mathbf{x}_t)\| \\
&\leq \frac{1}{c\sqrt{T}} [f^{(0)}(\mathbf{x}_0) - f^{(0)}(\mathbf{x}_T)] + \frac{2L_2 c}{\sqrt{T}\omega_{\min}} \\
&\leq \frac{1}{c\sqrt{T}} [f^{(0)}(\mathbf{x}_0) - f_*^{(0)}] + \frac{2L_2 c}{\sqrt{T}\omega_{\min}} \\
&\leq \frac{1}{\sqrt{T}} \left(\frac{1}{c} [f^{(0)}(\mathbf{x}_0) - f_*^{(0)}] + \frac{2L_2 c}{\omega_{\min}} \right).
\end{aligned} \tag{30}$$

□

Randomized iterate Next, we use the same randomization technique as in (Ghadimi and Lan, 2013) where we choose an iterate of SGD at random according to a particular probability distribution.

Theorem D.3 (R-PCNGD). *Let the probability mass function $P_R(\cdot)$ be defined as*

$$P_R(t) = \mathit{Prob}\{R = t\} = \frac{\omega_t}{\sum_{t=0}^{T-1} \omega_t}, \tag{31}$$

where $\omega_t := (1 + \cos\alpha(\mathbf{x}_t))$.
Then, for any step size $\eta_t = \frac{c}{\sqrt{T}}$ and $\cos\alpha(\mathbf{x}_t) \neq -1$ for all iterations t, we have for $l = \{0, 1\}$,

$$\mathbb{E}\|\nabla f^{(l)}(\mathbf{x}_R)\| \leq \frac{1}{\sqrt{T}\bar{\omega}} \left[c^{-1}(f^{(l)}(\mathbf{x}_0) - f^{(l)}(\mathbf{x}^*)) + 2L_2 c \right], \tag{32}$$

where $\bar{\omega} = \frac{1}{T} \sum_{t=0}^{T-1} \omega_t$.

Proof. We only write the detail for the function $f^{(0)}$ since the proof for $f^{(1)}$ is identical.

First, since $f^{(0)}$ is L_2-smooth, we have

$$\begin{aligned} f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) + \langle \nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{L_2}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\ &\leq f^{(0)}(\mathbf{x}_t) - \eta_t \|\nabla f^{(0)}(\mathbf{x}_t)\| - \eta_t \langle \nabla f^{(0)}(\mathbf{x}_t), \frac{\nabla f^{(1)}(\mathbf{x}_t)}{\|\nabla f^{(1)}(\mathbf{x}_t)\|} \rangle \\ &\quad + L_2 \eta_t^2 \left\| \frac{\nabla f^{(0)}(\mathbf{x}_t)}{\|\nabla f^{(0)}(\mathbf{x}_t)\|} \right\|^2 + L_2 \eta_t^2 \left\| \frac{\nabla f^{(1)}(\mathbf{x}_t)}{\|\nabla f^{(1)}(\mathbf{x}_t)\|} \right\|^2 \\ &= f^{(0)}(\mathbf{x}_t) - \eta_t \|\nabla f^{(0)}(\mathbf{x}_t)\| - \eta_t \|\nabla f^{(0)}(\mathbf{x}_t)\| \cos\alpha(\mathbf{x}_t) + 2L_2 \eta_t^2 \\ &= f^{(0)}(\mathbf{x}_t) - \eta_t (1 + \cos\alpha(\mathbf{x}_t)) \|\nabla f^{(0)}(\mathbf{x}_t)\| + 2L_2 \eta_t^2. \end{aligned} \tag{33}$$

Let $\omega_t := (1 + \cos\alpha(\mathbf{x}_t))$. By rearranging the terms in the equation above, we get

$$\omega_t \|\nabla f^{(0)}(\mathbf{x}_t)\| \leq \frac{1}{\eta_t} [f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + 2L_2 \eta_t \tag{34}$$

By summing from $t = 0$ to $T - 1$ and dividing by $\sum_{t=0}^{T-1} \omega_t$,

$$\begin{aligned} \frac{\sum_{t=0}^{T-1} \omega_t \|\nabla f^{(0)}(\mathbf{x}_t)\|}{\sum_{t=0}^{T-1} \omega_t} &\leq \frac{(\sum_{t=0}^{T-1} \eta_t)^{-1}}{\sum_{t=0}^{T-1} \omega_t} (f^{(0)}(\mathbf{x}_0) - f_*^{(0)}) + \frac{2L_2 \sum_{t=0}^{T-1} \eta_t}{\sum_{t=0}^{T-1} \omega_t} \\ &= \frac{(\sum_{t=0}^{T-1} \eta_t)^{-1}}{T\bar{\omega}} (f^{(0)}(\mathbf{x}_0) - f_*^{(0)}) + \frac{2L_2 \sum_{t=0}^{T-1} \eta_t}{T\bar{\omega}}, \end{aligned} \tag{35}$$

where $\bar{\omega}$ is the average of the sequence $\{\omega_t\}_{t=0}^{T-1}$.

Taking $\eta_t = \eta := \frac{c}{\sqrt{T}}$, we obtain

$$\begin{aligned} \frac{\sum_{t=0}^{T-1} \omega_t \|\nabla f^{(0)}(\mathbf{x}_t)\|}{\sum_{t=0}^{T-1} \omega_t} &\leq \frac{1}{c\bar{\omega}\sqrt{T}} (f^{(0)}(\mathbf{x}_0) - f_*^{(0)}) + \frac{2L_2 \eta}{\bar{\omega}} \\ &\leq \frac{1}{c\bar{\omega}\sqrt{T}} (f^{(0)}(\mathbf{x}_0) - f_*^{(0)}) + \frac{2L_2 c}{\sqrt{T}\bar{\omega}} \end{aligned} \tag{36}$$

Finally, observe that the RHS is an expectation $\mathbb{E}\|\nabla f^{(0)}(\mathbf{x}_r)\|$ for an appropriately chosen random variable $\mathbf{x}_R$ according to the distribution $P_R(\cdot)$, therefore

$$\mathbb{E}\|\nabla f^{(0)}(\mathbf{x}_R)\| \leq \frac{1}{\sqrt{T}\bar{\omega}} \left[c^{-1}(f^{(0)}(\mathbf{x}_0) - f_*^{(0)}) + 2L_2 c \right] \tag{37}$$

□

Convergence under Gradient dominance condition We prove convergence of PCNGD under a gradient-dominated condition which is a variant of the Polyak-Łojasiewicz (PL) condition that has been shown to hold for overparametrized neural networks (Liu et al., 2022). Instead of requiring this condition to hold for f, we require it to hold for each class separately. Specifically, we say that a function satisfies the *class-GD* inequality if the following holds for each class l,

$$\frac{1}{2} \|\nabla f^{(l)}(\mathbf{x})\| \geq \mu^{(l)} |f^{(l)}(\mathbf{x}) - f^{(l)}(\mathbf{x}^*)|, \quad \forall \mathbf{x} \in \mathbb{R}^m, \tag{38}$$

for some constant $\mu^{(l)} > 0$. For finite-sum objective functions, the class-GD inequality implies the gradient-dominated inequality.

We are now ready to state the main convergence result for smooth and class-GD functions.

Theorem D.4 (Formal version of Theorem 4.3)**.** *Assume that each $f^{(l)}$ for $l \in \{0,1\}$ is L_2-smooth and μ-class-GD. Then if $\cos\alpha(\mathbf{x}_t) \neq -1$ for all iterations $t \in [0, T-1]$, the iterates of PCNGD with step size $\eta_t = \frac{2t+1}{C_{\mu,l}(t+1)^2}$, we have*

$$f^{(l)}(\mathbf{x}_T) - f_*^{(l)} \leq \frac{8L_2}{C_{\mu,l}^2 T}, \tag{39}$$

for each $l \in \{0,1\}$, where $C_{\mu,l} := \max_{t\in[0,T-1]} 2\mu^{(l)}(1+\cos\alpha(\mathbf{x}_t))$. Furthermore, we obtain a linear rate of convergence up to a ball with a constant step size $\eta_t = \eta = \frac{c}{C_{\mu,l}}$ where $c \in (0,1)$:

$$f^{(l)}(\mathbf{x}_T) - f_*^{(l)} \leq (1-c)^{T-1}(f^{(l)}(\mathbf{x}_0) - f_*^{(l)}) + \frac{2L_2 c}{C_{\mu,l}^2}. \tag{40}$$

Proof. We again provide a proof for $f^{(0)}$ as the proof for $f^{(1)}$ is identical.

First, since $f^{(0)}$ is L_2-smooth, we have

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) + \langle \nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t\rangle + \frac{L_2}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\
&\leq f^{(0)}(\mathbf{x}_t) - \eta_t\|\nabla f^{(0)}(\mathbf{x}_t)\| - \eta_t\langle \nabla f^{(0)}(\mathbf{x}_t), \frac{\nabla f^{(1)}(\mathbf{x}_t)}{\|\nabla f^{(1)}(\mathbf{x}_t)\|}\rangle \\
&\quad + L_2\eta_t^2\left\|\frac{\nabla f^{(0)}(\mathbf{x}_t)}{\|\nabla f^{(0)}(\mathbf{x}_t)\|}\right\|^2 + L_2\eta_t^2\left\|\frac{\nabla f^{(1)}(\mathbf{x}_t)}{\|\nabla f^{(1)}(\mathbf{x}_t)\|}\right\|^2 \\
&= f^{(0)}(\mathbf{x}_t) - \eta_t\|\nabla f^{(0)}(\mathbf{x}_t)\| - \eta_t\|\nabla f^{(0)}(\mathbf{x}_t)\|\cos\alpha(\mathbf{x}_t) + 2L_2\eta_t^2 \\
&= f^{(0)}(\mathbf{x}_t) - \eta_t(1+\cos\alpha(\mathbf{x}_t))\|\nabla f^{(0)}(\mathbf{x}_t)\| + 2L_2\eta_t^2.
\end{aligned} \tag{41}$$

Using the gradient-dominated condition, and subtracting $f_*^{(0)}$ from both sides,

$$f^{(0)}(\mathbf{x}_{t+1}) - f_*^{(0)} \leq (1 - 2\eta_t\mu^{(l)}(1+\cos\alpha(\mathbf{x}_t)))(f^{(0)}(\mathbf{x}_t) - f_*^{(0)}) + 2L_2\eta_t^2. \tag{42}$$

Note that we need $0 < (1 - 2\eta_t\mu^{(l)}(1+\cos\alpha(\mathbf{x}_t))) < 1$ for all t, i.e.

$$\eta_t < \frac{1}{2\mu^{(l)}(1+\cos\alpha(\mathbf{x}_t))} < \frac{1}{C_{\mu,l}},$$

where $C_{\mu,l} := \max_{t\in[0,T-1]} 2\mu^{(l)}(1+\cos\alpha(\mathbf{x}_t))$.

<u>Decreasing step size</u> Choose $\eta_t = \frac{2t+1}{C_{\mu,l}(t+1)^2}$, then

$$f^{(0)}(\mathbf{x}_{t+1}) - f_*^{(0)} \leq \left(\frac{t^2}{(t+1)^2}\right)(f^{(0)}(\mathbf{x}_t) - f_*^{(0)}) + \frac{2L_2(2t+1)^2}{C_{\mu,l}^2(t+1)^4}. \tag{43}$$

Multiplying both sides by $(t+1)^2$, and letting $\delta_f(t) = t^2(f^{(0)}(\mathbf{x}_t) - f_*^{(0)})$, we obtain

$$\begin{aligned}
\delta_f(k+1) &\leq \delta_f(k) + \frac{2L_2(2t+1)^2}{C_{\mu,l}^2(t+1)^2} \\
&\leq \delta_f(k) + \frac{8L_2}{C_{\mu,l}^2},
\end{aligned} \tag{44}$$

where the last inequality is due to $\frac{2t+1}{t+1} \leq 2$

Summing up this inequality from $t = 0$ to $T - 1$, we conclude

$$\delta_f(T) \leq \delta_f(0) + \frac{8L_2 T}{C_{\mu,l}^2} \tag{45}$$

$$\implies T^2(f^{(0)}(\mathbf{x}_T) - f_*^{(0)}) \leq \frac{8L_2 T}{C_{\mu,l}^2} \tag{46}$$

$$\implies f^{(0)}(\mathbf{x}_T) - f_*^{(0)} \leq \frac{8L_2}{C_{\mu,l}^2 T}. \tag{47}$$

Constant step size: $\eta_t = \eta > 0$ Choose $\eta = \frac{c}{C_{\mu,l}}$ for $c \in (0, 1)$, then

$$\begin{aligned} f^{(0)}(\mathbf{x}_t) - f_*^{(0)} &\leq (1-c)^{t-1}(f^{(0)}(\mathbf{x}_0) - f_*^{(0)}) + 2L_2\eta^2 \sum_{i=0}^{t-1}(1-c)^i \\ &\leq (1-c)^{t-1}(f^{(0)}(\mathbf{x}_0) - f_*^{(0)}) + 2L_2\eta^2 \sum_{i=0}^{\infty}(1-c)^i \\ &\leq (1-c)^{t-1}(f^{(0)}(\mathbf{x}_0) - f_*^{(0)}) + \frac{2L_2\eta^2}{c} \qquad (48) \\ &= (1-c)^{t-1}(f^{(0)}(\mathbf{x}_0) - f_*^{(0)}) + \frac{2L_2 c}{C_{\mu,l}^2}, \qquad (49) \end{aligned}$$

where we used the fact that the last term in the second line is a geometric series in the last equation.

□

D.1. Stochastic algorithms

We will now analyze the convergence property of PCNSGD whose update rule is given by

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \left(\frac{\nabla_{\tilde{n}} f^{(0)}(\mathbf{x}_t)}{\|\nabla_{\tilde{n}} f^{(0)}(\mathbf{x}_t)\|} + \frac{\nabla_{\tilde{n}} f^{(1)}(\mathbf{x}_t)}{\|\nabla_{\tilde{n}} f^{(1)}(\mathbf{x}_t)\|} \right), \tag{50}$$

where the subscript $\tilde{n}$ indicates gradients that are taken over batches of size $\tilde{n}$. In the following, we will use the shorthand notation $g^{(l)}(\mathbf{x}_t) := \nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$ (with $l = 0, 1$) to denote the stochastic gradients.

Theorem D.5 (Formal version of Theorem 5.1)**.** *Assume that each $f^{(l)}$ is L_1-Lipschitz and L_2-smooth and $\mathbb{E}\|\nabla f^{(l)}(\mathbf{x}_t) - g^{(l)}(\mathbf{x}_t)\|^2 \leq \sigma_l^2$ (where $\sigma_l > 0$). Then if $\cos\alpha(\mathbf{x}_t) \neq -1$ for all iterations $t \in [0, T-1]$, the iterates of PCNSGD with step size $\eta_t = \frac{c}{\sqrt{T}}$ (where $c > 0$) satisfy*

$$\min_{s \in [0,T-1]} \mathbb{E}\|\nabla f^{(l)}(\mathbf{x}_s)\| \leq \frac{1}{\omega_{\max}\sqrt{T}} \left(\frac{D_0^{(l)}}{c} + 2L_2 c \right) + \sigma_l \left(1 + \frac{2}{\omega_{\max}} \right),$$

for each $l \in \{0, 1\}$, where $D_0^{(l)} = \mathbb{E}[f^{(l)}(\mathbf{x}_0) - f_*^{(l)}]$ and $0 < \max_{t \in [0,T-1]}(1 + \cos\alpha(\mathbf{x}_t)) < \omega_{\max}$.

Proof. We only write the detail for the function $f^{(0)}$ since the proof for $f^{(1)}$ is identical. We also use the shorthand notation $\sigma := \sigma_0$.

First, since $f^{(0)}$ is L_2-smooth, we have

$$\begin{aligned} f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) + \langle \nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{L_2}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\ &\leq f^{(0)}(\mathbf{x}_t) - \eta_t \langle \nabla f^{(0)}(\mathbf{x}_t), \frac{g^{(0)}(\mathbf{x}_t)}{\|g^{(0)}(\mathbf{x}_t)\|} \rangle - \eta_t \langle \nabla f^{(0)}(\mathbf{x}_t), \frac{g^{(1)}(\mathbf{x}_t)}{\|g^{(1)}(\mathbf{x}_t)\|} \rangle \\ &\quad + L_2 \eta_t^2 \left\| \frac{g^{(0)}(\mathbf{x}_t)}{\|g^{(0)}(\mathbf{x}_t)\|} \right\|^2 + L_2 \eta_t^2 \left\| \frac{g^{(1)}(\mathbf{x}_t)}{\|g^{(1)}(\mathbf{x}_t)\|} \right\|^2 \\ &= f^{(0)}(\mathbf{x}_t) \underbrace{-\eta_t \langle \nabla f^{(0)}(\mathbf{x}_t), \frac{g^{(0)}(\mathbf{x}_t)}{\|g^{(0)}(\mathbf{x}_t)\|} \rangle}_{:=(A)} \underbrace{-\eta_t \langle \nabla f^{(0)}(\mathbf{x}_t), \frac{g^{(1)}(\mathbf{x}_t)}{\|g^{(1)}(\mathbf{x}_t)\|} \rangle}_{:=(B)} + 2L_2 \eta_t^2. \end{aligned} \tag{51}$$

By simple manipulations, term (A) can be bounded as follows,

$$\begin{aligned} (A) &= -\eta_t \langle \nabla f^{(0)}(\mathbf{x}_t) - g^{(0)}(\mathbf{x}_t), \frac{g^{(0)}(\mathbf{x}_t)}{\|g^{(0)}(\mathbf{x}_t)\|} \rangle - \eta_t \frac{\langle g^{(0)}(\mathbf{x}_t), g^{(0)}(\mathbf{x}_t) \rangle}{\|g^{(0)}(\mathbf{x}_t)\|} \\ &= -\eta_t \langle \nabla f^{(0)}(\mathbf{x}_t) - g^{(0)}(\mathbf{x}_t), \frac{g^{(0)}(\mathbf{x}_t)}{\|g^{(0)}(\mathbf{x}_t)\|} \rangle - \eta_t \|g^{(0)}(\mathbf{x}_t)\| \\ &\leq \eta_t \|\nabla f^{(0)}(\mathbf{x}_t) - g^{(0)}(\mathbf{x}_t)\| - \eta_t \|g^{(0)}(\mathbf{x}_t)\|, \end{aligned} \tag{52}$$

where we used Cauchy-Schwarz in the last inequality.

Similarly for term (B),

$$\begin{aligned} (B) &= -\eta_t \langle \nabla f^{(0)}(\mathbf{x}_t), \frac{g^{(1)}(\mathbf{x}_t)}{\|g^{(1)}(\mathbf{x}_t)\|} \rangle \\ &\leq \eta_t \|\nabla f^{(0)}(\mathbf{x}_t) - g^{(0)}(\mathbf{x}_t)\| - \eta_t \|g^{(0)}(\mathbf{x}_t)\| \cos \alpha(\mathbf{x}_t). \end{aligned} \tag{53}$$

Let $\omega_{\max} > \max_{t \in [0, T-1]} (1 + \cos \alpha(\mathbf{x}_t))$. Combining the last three equations and taking the expectation (over initial conditions and batches) yields

$$\begin{aligned} \mathbb{E}[f^{(0)}(\mathbf{x}_{t+1}) - f^{(0)}(\mathbf{x}_t)] &\leq 2\eta_t \mathbb{E}\|\nabla f^{(0)}(\mathbf{x}_t) - g^{(0)}(\mathbf{x}_t)\| - \eta_t \mathbb{E}[\|g^{(0)}(\mathbf{x}_t)\|(1 + \cos \alpha(\mathbf{x}_t))] + 2L_2 \eta_t^2 \\ &\leq 2\eta_t \sqrt{\mathbb{E}\|\nabla f^{(0)}(\mathbf{x}_t) - g^{(0)}(\mathbf{x}_t)\|^2} - \eta_t \omega_{\max} \mathbb{E}[\|g^{(0)}(\mathbf{x}_t)\|] + 2L_2 \eta_t^2 \\ &\leq 2\eta_t \sigma - \eta_t \omega_{\max} \mathbb{E}[\|g^{(0)}(\mathbf{x}_t)\|] + 2L_2 \eta_t^2. \end{aligned} \tag{54}$$

From Eq. (54) we see that if σ is finite the loss function is not monotonic on average. In order to have a per-class loss function which is monotonic on average and avoid the MID, σ needs to be small, which corresponds to a large batch size.

Since $\cos(\alpha_t) \neq -1$, we can rearrange the terms in Eq (54), getting

$$\mathbb{E}\|g^{(0)}(\mathbf{x}_t)\| \leq \frac{1}{\omega_{\max}} \left(2\sigma + \frac{1}{\eta_t} \mathbb{E}[f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + 2L_2 \eta_t \right), \tag{55}$$

therefore

$$\begin{aligned} \mathbb{E}\|\nabla f^{(0)}(\mathbf{x}_t)\| &\leq \mathbb{E}\|g^{(0)}(\mathbf{x}_t)\| + \mathbb{E}\|\nabla f^{(0)}(\mathbf{x}_t) - g^{(0)}(\mathbf{x}_t)\| \\ &\leq \frac{2\sigma}{\omega_{\max}} + \frac{1}{\omega_{\max} \eta_t} \mathbb{E}[f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + \frac{2L_2 \eta_t}{\omega_{\max}} + \sqrt{\mathbb{E}\|\nabla f^{(0)}(\mathbf{x}_t) - g^{(0)}(\mathbf{x}_t)\|^2} \\ &\leq \frac{2\sigma}{\omega_{\max}} + \frac{1}{\omega_{\max} \eta_t} \mathbb{E}[f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + \frac{2L_2 \eta_t}{\omega_{\max}} + \sigma. \end{aligned} \tag{56}$$

Taking the minimum over t, and using $\eta_t = \frac{c}{\sqrt{T}}$, we get

$$
\begin{aligned}
\min_{s\in[0,T-1]} \mathbb{E}\|\nabla f^{(0)}(\mathbf{x}_s)\| &\leq \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f^{(0)}(\mathbf{x}_t)\| \\
&\leq \frac{1}{c\,\omega_{\max}\sqrt{T}}\mathbb{E}[f^{(0)}(\mathbf{x}_0)-f^{(0)}(\mathbf{x}_T)] + \frac{2L_2 c}{\omega_{\max}\sqrt{T}} + \frac{2\sigma}{\omega_{\max}} + \sigma \\
&\leq \frac{1}{c\,\omega_{\max}\sqrt{T}}\mathbb{E}[f^{(0)}(\mathbf{x}_0)-f_*^{(0)}] + \frac{2L_2 c}{\omega_{\max}\sqrt{T}} + \frac{2\sigma}{\omega_{\max}} + \sigma \\
&\leq \frac{1}{\omega_{\max}\sqrt{T}}\left(\frac{1}{c}\mathbb{E}[f^{(0)}(\mathbf{x}_0)-f_*^{(0)}] + 2L_2 c\right) + \frac{2\sigma}{\omega_{\max}} + \sigma, \qquad (57)
\end{aligned}
$$

where we used $f_*^{(0)} \leq f^{(0)}(\mathbf{x}_T)$ by definition of $f_*^{(0)}$.

□

D.2. Projection of the PCNSGD steps onto the full-batch gradient

Here, we elicit more explicitly how Eq. (7) is obtained. On the left hand side we write the projection of the per-batch gradients on the full batch. Then we use Eq. 6 for the per-batch gradients, and keep the leading orders:

$$
\begin{aligned}
&\left(\frac{\nabla f_{\tilde{n}_l}^{(l)}}{\|\nabla f_{\tilde{n}_l}^{(l)}\|}\right)\cdot\frac{\nabla f_{n_l}^{(l)}}{\|\nabla f_{n_l}^{(l)}\|} = \left(\frac{\nabla f_{n_l}^{(l)}}{\|\nabla f_{n_l}^{(l)}\|} - \frac{\nabla f_{n_l}^{(l)}\left(\nabla f_{n_l}^{(l)}\cdot\mathbf{Z}^{(l)}\right)}{\sqrt{\tilde{n}_l}\|\nabla f_{n_l}^{(l)}\|^3} - \frac{\nabla f_{n_l}^{(l)}\|\mathbf{Z}^{(l)}\|^2}{2\tilde{n}_l\|\nabla f_{n_l}^{(l)}\|^3} + \right. \\
&\left. + \frac{3\nabla f_{n_l}^{(l)}\left(\nabla f_{n_l}^{(l)}\cdot\mathbf{Z}^{(l)}\right)^2}{2\tilde{n}_l\|\nabla f_{n_l}^{(l)}\|^5} + \frac{\mathbf{Z}^{(l)}}{\sqrt{\tilde{n}_l}\|\nabla f_{n_l}^{(l)}\|} - \frac{\mathbf{Z}^{(l)}\left(\nabla f_{n_l}^{(l)}\cdot\mathbf{Z}^{(l)}\right)}{\tilde{n}_l\|\nabla f_{n_l}^{(l)}\|^3} + o\left(\frac{1}{\tilde{n}_l}\right)\right)\cdot\frac{\nabla f_{n_l}^{(l)}}{\|\nabla f_{n_l}^{(l)}\|} \qquad (58) \\
&= 1 - \frac{\|\mathbf{Z}^{(l)}\|^2(1-\cos(\theta)^2)}{2\tilde{n}_l\|\nabla f_{n_l}^{(l)}\|^2} + o\left(\frac{1}{\tilde{n}_l}\right).
\end{aligned}
$$

Here, θ indicates the angle between $\mathbf{Z}^{(l)}$ and $\nabla f_{n_l}^{(l)}$.

E. Multi-class analysis

In the following, we demonstrate how the analysis derived in previous sections can be adapted to multi-class problems. We only derive the proof for the deterministic case (full-batch) with one choice of step-size, but the analysis can be adapted similarly to other settings.

Gradient descent analysis The update is

$$
\begin{aligned}
\mathbf{x}_{t+1} &= \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t) \qquad (59) \\
&= \mathbf{x}_t - \eta_t \sum_{i=0}^{L-1}\nabla f^{(i)}(\mathbf{x}_t). \qquad (60)
\end{aligned}
$$

Theorem E.1 (Formal, multiclass, version of Theorem 4.1 - multi-class)**.** *Assume that each $f^{(l)}$ is L_1-Lipshitz and L_2-smooth and let $\alpha^{(l)}(\mathbf{x}_t) = \angle(\nabla f^{(l)}(\mathbf{x}_t), \sum_{i\neq l} \nabla f^{(i)}(\mathbf{x}_t))$. If $\|\sum_{i\neq l} \nabla f^{(i)}(\mathbf{x}_t))\| \neq 0$ and $\cos(\alpha^{(l)}(\mathbf{x}_t)) > -\frac{1}{C_t^{(l)}}$ where $C_t^{(l)} := \frac{\|\sum_{i\neq l} \nabla f^{(i)}(\mathbf{x}_t)\|}{\|\nabla f^{(l)}(\mathbf{x}_t)\|}$ for all iterations $t \in [0, T-1]$, the iterates of gradient descent with step size $\eta_t = \min\left(\frac{1+\cos(\alpha^{(l)}(\mathbf{x}_t))C_t^{(l)}}{2\left(1+\left(C_t^{(l)}\right)^2\right)L_2}, \frac{c}{\sqrt{T}}\right)$ where $c > 0$ satisfy*

$$\min_{s\in[0,T-1]} \|\nabla f^{(l)}(\mathbf{x}_s)\|^2 \leq \frac{2(1+C_{\max}^{(l)})L_2}{(\omega_{min}^{(l)})^2 T} D_0^{(l)} + \frac{1}{\omega_{min}^{(l)} c\sqrt{T}} D_0^{(l)},$$

for each $l \in \{0, 1\}$, where $D_0^{(l)} = [f^{(l)}(\mathbf{x}_0) - f_*^{(l)}]$, $\omega_{min}^{(l)} = \min_{t\in[0,T-1]} 1 + \cos(\alpha^{(l)}(\mathbf{x}_t))C_t^{(l)}$, and $C_{\max}^{(l)} = \max_{t\in[0,T-1]} \left(C_t^{(l)}\right)^2$.

Proof. We only write the detail for the function $f^{(0)}$ since the proof for the generic $f^{(l)}$ is identical. We use the shorthand notation $\alpha := \alpha^{(0)}$, $C_t := C_t^{(0)}$.

Since each function $f^{(0)}$ is L_2-smooth, we have

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) + \langle \nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t\rangle + \frac{L_2}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \qquad (61)\\
&= f^{(0)}(\mathbf{x}_t) - \eta_t\langle \nabla f^{(0)}(\mathbf{x}_t), \nabla f(\mathbf{x}_t)\rangle + \frac{L_2\eta_t^2}{2}\|\nabla f(\mathbf{x}_t)\|^2\\
&\leq f^{(0)}(\mathbf{x}_t) - \eta_t\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta_t\langle \nabla f^{(0)}(\mathbf{x}_t), \sum_{i\neq 0}\nabla f^{(i)}(\mathbf{x}_t)\rangle + L_2\eta_t^2\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 + L_2\eta_t^2\left\|\sum_{i\neq 0}\nabla f^{(i)}(\mathbf{x}_t)\right\|^2\\
&= f^{(0)}(\mathbf{x}_t) - \eta_t(1 - L_2\eta_t)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta_t\langle \nabla f^{(0)}(\mathbf{x}_t), \sum_{i\neq 0}\nabla f^{(i)}(\mathbf{x}_t)\rangle + L_2\eta_t^2\left\|\sum_{i\neq 0}\nabla f^{(i)}(\mathbf{x}_t)\right\|^2\\
&= f^{(0)}(\mathbf{x}_t) - \eta_t(1 - L_2\eta_t)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 - \eta_t\cos(\alpha(\mathbf{x}_t))\|\nabla f^{(0)}(\mathbf{x}_t)\|\|\sum_{i\neq 0}\nabla f^{(i)}(\mathbf{x}_t)\| + L_2\eta_t^2\left\|\sum_{i\neq 0}\nabla f^{(i)}(\mathbf{x}_t)\right\|^2,
\end{aligned}$$

where the inequality in the third line is simply due to $\|\mathbf{x}+\mathbf{y}\|_2^2 \leq 2\|\mathbf{x}\|_2^2 + 2\|\mathbf{y}\|_2^2$ for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.

Let $C_t := \frac{\|\sum_{i\neq 0}\nabla f^{(i)}(\mathbf{x}_t)\|}{\|\nabla f^{(0)}(\mathbf{x}_t)\|}$. Note that, in the presence of only two classes, this definition reduces to the same as the one used in other sections. We get

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) - \eta_t\left(1 + \cos(\alpha(\mathbf{x}_t))C_t - L_2\eta_t - L_2\eta_t C_t^2\right)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2\\
\implies \eta_t(1 + \cos(\alpha(\mathbf{x}_t))C_t - (1 + C_t^2)L_2\eta_t)&\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 \leq f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1}). \qquad (62)
\end{aligned}$$

At this point, we see that, in order to have a monotonic decrease of the loss related to class 0, we need the following condition on the angle $\alpha(\mathbf{x}_t)$:

$$\cos(\alpha(\mathbf{x}_t)) > -\frac{1}{C_t}. \qquad (63)$$

Taking $\eta_t = \min\left(\frac{1+\cos(\alpha(\mathbf{x}_t))C_t}{2(1+C_t^2)L_2}, \frac{c}{\sqrt{T}}\right)$, we have

$$\eta_t(1+\cos(\alpha(\mathbf{x}_t))C_t) - \eta_t^2(1+C_t^2)L_2 \geq \frac{\eta_t}{2}(1+\cos(\alpha(\mathbf{x}_t))C_t),$$

therefore

$$\frac{\eta_t}{2}(1+\cos(\alpha(\mathbf{x}_t))C_t)\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 \leq f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1}). \tag{64}$$

Let $\omega_t := 1+\cos(\alpha(\mathbf{x}_t))C_t$, then

$$\left\|\nabla f^{(0)}(\mathbf{x}_t)\right\|^2 \leq \frac{2}{\omega_t\eta_t}(f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})). \tag{65}$$

Let $\omega_{min}^{(0)} = \min_{t\in[0,T-1]}\omega_t$ and $C_{\max} := \max_{t\in[0,T-1]} C_t^2$. By summing from $t=0$ to $T-1$,

$$\begin{aligned}
\min_{s\in[0,T-1]}\|\nabla f^{(0)}(\mathbf{x}_s)\|^2 &\leq \frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f^{(0)}(\mathbf{x}_t)\|^2 \\
&\leq \frac{1}{\omega_{min}^{(0)}T}\max\left(\frac{2(1+C_{\max})L_2}{\omega_{min}^{(0)}}, \frac{\sqrt{T}}{c}\right)[f^{(0)}(\mathbf{x}_0) - f^{(0)}(\mathbf{x}_T)] \\
&\leq \frac{1}{\omega_{min}^{(0)}T}\left(\frac{2(1+C_{\max})L_2}{\omega_{min}^{(0)}} + \frac{\sqrt{T}}{c}\right)[f^{(0)}(\mathbf{x}_0) - f_*^{(0)}] \\
&\leq \frac{2(1+C_{\max})L_2}{(\omega_{min}^{(0)})^2T}[f^{(0)}(\mathbf{x}_0) - f_*^{(0)}] + \frac{1}{\omega_{min}^{(0)}c\sqrt{T}}[f^{(0)}(\mathbf{x}_0) - f_*^{(0)}],
\end{aligned} \tag{66}$$

where $f_*^{(0)}$ denotes the global minimum of $f^{(0)}(\mathbf{x})$. Note that Eq. (66) is related to the imbalance, since both c and $\omega_{min}^{(0)}$ depend on C_t, which at least at the beginning of the dynamics depends on the imbalance.

□

We now want to get an intuition on the meaning of Th. E.1 and how condition (63) varies with imbalance and number of classes. Since the numerator of C_t is the norm of a sum of vectors, the value of C_t depends on the mutual angles between the per-class gradients. This these are *a priori* not known, we will use the worst-case scenario to elucidate the role of condition (63). The worse-case scenario for the optimization of class 0 is when all the $(L-1)$ gradient vectors $\left\{\nabla f^{(i)}(\mathbf{x}_t)\right\}_{i\neq 0}$ are collinear and pointing in the same direction (and, in the case of imbalance, class 0 is the minority class). Using the extensivity of the gradient, we can write $\|\nabla f^{(i)}(\mathbf{x}_t)\| \sim n_i M_i$ where n_i indicates the number of elements inside the class i, and M_i the typical gradient norm (we are assuming that fluctuations have a finite variance). In the case of a balanced dataset, $n_i = \hat{n}\ \forall i$. If all classes are equivalent, we can write $\|\nabla f^{(i)}(\mathbf{x}_t)\| \sim \hat{n}M$. Under these assumptions, we will have, in the worst-case scenario for a balanced dataset

$$C_t = (L-1)\,, \tag{67}$$

where we see that condition (63) is increasingly restrictive with the number of classes.

Let us now relax the hypothesis of balanced dataset to consider imbalance. If, again, different classes have similar gradient norms, the difference between per-class gradients reduce to the imbalance between classes. For the worst-case scenario

$$C_t = \frac{\sum_{i\neq 0} n_i}{n_0}. \tag{68}$$

Comparing the condition in case of imbalance with the balanced case (Eq. (67)) we can see how, if class 0 is a minority class (which is the case we are interested in), the worst-case scenario become more restrictive; the imbalance condition implies, in fact, $\frac{\sum_{i\neq 0} n_i}{n_0} > (L-1)$.

Example We now show a simple example where the worst-case scenario value derived above represent a good estimation of the general case, *i.e.* the limit where the value of C_t is not influenced by the angles between the set of vectors $\left\{\nabla f^{(i)}(\mathbf{x}_t)\right\}_{i\neq 0}$. Let us consider a problem with L classes with a big gap between the population of the majority one and all the others. We say there are N examples in total, the majority class, L, has n_L examples, and minority classes have $\epsilon \ll \frac{n_L}{L}$ examples. In this case, there is no significant difference induced by the interference between the $(L-1)$ vectors and Eq. (68) becomes

$$C_t \sim \frac{N}{\epsilon} \gg 1\,, \tag{69}$$

where we remind the reader that the least restrictive value for C_t is $C_t = 1$, and the bigger C_t, the more stringent condition (63) becomes.

PCNGD We now turn to the analysis of PCNGD in the multi-class scenario. We study the following variant for the multi-class case:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \sum_{i=0}^{L-1} \frac{\nabla f^{(i)}(\mathbf{x}_t)}{\|\nabla f^{(i)}(\mathbf{x}_t)\|}, \tag{70}$$

which recovers the binary update for $L = 2$:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \left(\frac{\nabla f^{(0)}(\mathbf{x}_t)}{\|\nabla f^{(0)}(\mathbf{x}_t)\|} + \frac{\nabla f^{(1)}(\mathbf{x}_t)}{\|\nabla f^{(1)}(\mathbf{x}_t)\|} \right). \tag{71}$$

Theorem E.2 (Formal version of Theorem 4.2)**.** *Assume that each $f^{(l)}$ is L_1-Lipschitz and L_2-smooth and let $\alpha^{(l)}(\mathbf{x}_t) = \angle(\nabla f^{(l)}(\mathbf{x}_t), \sum_{i\neq l} \frac{\nabla f^{(i)}(\mathbf{x}_t))}{\|\nabla f^{(i)}(\mathbf{x}_t))\|})$. Then if $\cos\alpha^{(l)}(\mathbf{x}_t) \neq -1$ for all iterations $t \in [0, T-1]$, the iterates of PCNGD with step size $\eta_t = \frac{c}{\sqrt{T}}$ where $c > 0$ satisfy*

$$\min_{s\in[0,T-1]} \|\nabla f^{(l)}(\mathbf{x}_s)\| \leq \frac{1}{\omega^{(l)}_{\min}\sqrt{T}} \left(\frac{D_0^{(l)}}{c(K-1)} + L_2 c \right),$$

for each $l \in \{0, \dots L-1\}$, where $D_0^{(l)} = [f^{(l)}(\mathbf{x}_0) - f_*^{(l)}]$ and $\omega^{(l)}_{\min} := \min_{t\in[0,T-1]}(1 + \cos\alpha^{(l)}(\mathbf{x}_t))$.

Proof. We only write the detail for the function $f^{(0)}$ since the proof for any $f^{(l)}$ is identical. We use the shorthand notation $\alpha := \alpha^{(0)}$.

First, since $f^{(0)}$ is L_2-smooth, we have

$$\begin{aligned}
f^{(0)}(\mathbf{x}_{t+1}) &\leq f^{(0)}(\mathbf{x}_t) + \langle \nabla f^{(0)}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t\rangle + \frac{L_2}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\
&\leq f^{(0)}(\mathbf{x}_t) - \eta_t\|\nabla f^{(0)}(\mathbf{x}_t)\| - \eta_t\langle \nabla f^{(0)}(\mathbf{x}_t), \sum_{i\neq 0} \frac{\nabla f^{(i)}(\mathbf{x}_t)}{\|\nabla f^{(i)}(\mathbf{x}_t)\|}\rangle + \frac{L_2}{2}\eta_t^2 \left\| \sum_{i=0}^{K-1} \frac{\nabla f^{(i)}(\mathbf{x}_t)}{\|\nabla f^{(i)}(\mathbf{x}_t)\|} \right\|^2 \\
&= f^{(0)}(\mathbf{x}_t) - \eta_t\|\nabla f^{(0)}(\mathbf{x}_t)\| - \eta_t \left\| \sum_{i\neq 0} \frac{\nabla f^{(i)}(\mathbf{x}_t)}{\|\nabla f^{(i)}(\mathbf{x}_t)\|} \right\| \|\nabla f^{(0)}(\mathbf{x}_t)\| \cos\alpha(\mathbf{x}_t) + \frac{L_2}{2} \left\| \sum_{i=0}^{L-1} \frac{\nabla f^{(i)}(\mathbf{x}_t)}{\|\nabla f^{(i)}(\mathbf{x}_t)\|} \right\|^2 \eta_t^2 \\
&= f^{(0)}(\mathbf{x}_t) - \eta_t \left(1 + \left\| \sum_{i\neq 0} \frac{\nabla f^{(i)}(\mathbf{x}_t)}{\|\nabla f^{(i)}(\mathbf{x}_t)\|} \right\| \cos\alpha(\mathbf{x}_t) \right) \|\nabla f^{(0)}(\mathbf{x}_t)\| + \frac{L_2}{2} \left\| \sum_{i=0}^{L-1} \frac{\nabla f^{(i)}(\mathbf{x}_t)}{\|\nabla f^{(i)}(\mathbf{x}_t)\|} \right\|^2 \eta_t^2.
\end{aligned} \tag{72}$$

Taking a sufficiently small η_t the monotonicity condition for the per-class loss function (considering only first-order term in η_t) is:

$$\cos\alpha^{(l)}(\mathbf{x}_t) > -\frac{1}{\tilde{C}_t} \tag{73}$$

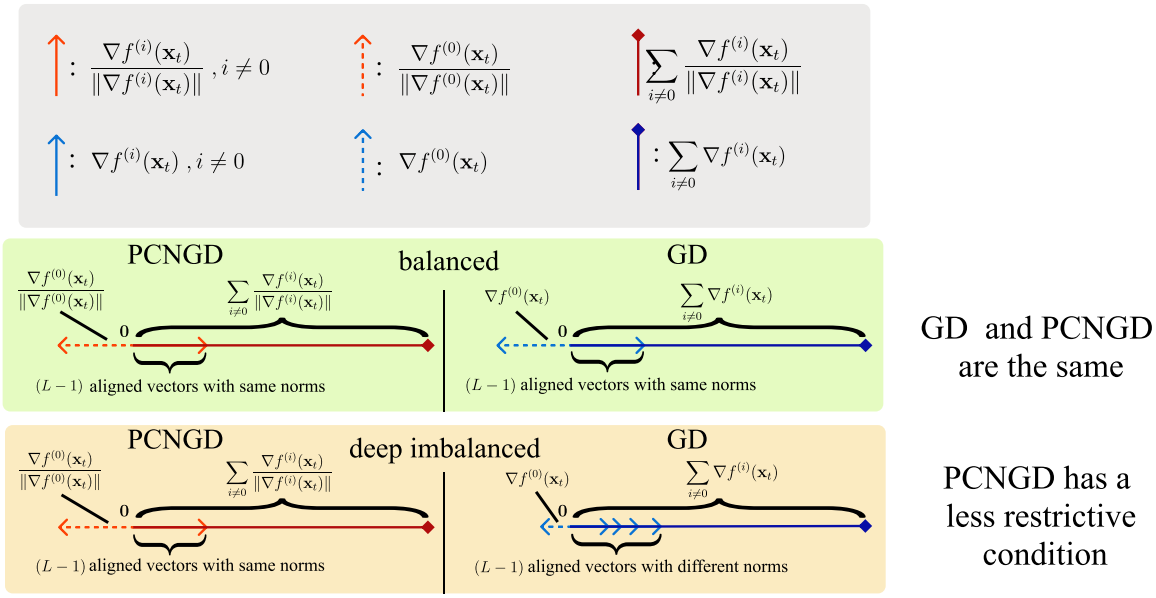

Figure S6. Comparison between GD (left schemes) and PCNGD (right schemes) for both balcanced (upper schemes) and imbalanced cases (bottom scheme). All diagrams represent the L per-class gradient vectors (normalized vectors in the PCNGD cases); all vectors are centered on the same point, 0. From Eq. (73) and Eq. (63), we know that the convergence condition depends on C_t (and $\tilde{C}_t$ respectively) which, in turns, depends on the norms of the gradient contribution coming from classes different from 0 (red vectors in the diagram). In particular, fixed $\|\nabla f^{(0)}(\mathbf{x}_t)\|$, the bigger the norm of the red vector is (with respect to the norm of the purple vector), the more restrictive the condition becomes. Here we report a visual low-dimensional representation of the worst-case scenario.

with $\tilde{C}_t = \tilde{C}_t^{(0)} \equiv \left\| \sum_{i\neq 0} \frac{\nabla f^{(i)}(\mathbf{x}_t)}{\|\nabla f^{(i)}(\mathbf{x}_t)\|} \right\|$. Similarly as done for GD we can, also in this case, starting from the set of per-class norms, get $\tilde{C}_t$ in the worst-case scenario, *i.e.*

$$C_t = (L-1) \tag{74}$$

Note that this is identical to the one derived for GD in the balanced case. Now, however the interval does not change with imbalance, *i.e.* for PCNGD in the imbalanced case, we get the same condition for GD in the balanced case (see Fig. S6). Finally, we note that although the condition on the angle does not depend on imbalance, is still does depend on the number of classes, in PCNGD as for GD.

Since, at least at the beginning of training, both $(1+\cos\alpha(\mathbf{x}_t))$ and $\|\nabla f^{(l)}(\mathbf{x}_t)\|$ are strictly positive, Eq. 72 implies that, for a small enough learning rate, the per-class loss at time $t+1$ is smaller than that at time t, so the MID is avoided.

Let $\omega_{\min} := \min_{t\in[0,T-1]}(1+\cos\alpha(\mathbf{x}_t))$. By rearranging the terms in the equation above, we get

$$\|\nabla f^{(0)}(\mathbf{x}_t)\| \leq \frac{1}{\eta_t(L-1)\omega_{\min}}[f^{(0)}(\mathbf{x}_t) - f^{(0)}(\mathbf{x}_{t+1})] + \frac{K^2 L_2 \eta_t}{2(L-1)\omega_{\min}}. \tag{75}$$

Taking the minimum over t, we get

$$\begin{aligned}
\min_{s\in[0,T-1]} \|\nabla f^{(0)}(\mathbf{x}_s)\| &\leq \frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f^{(0)}(\mathbf{x}_t)\| \\
&\leq \frac{1}{c\,\omega_{\min}(L-1)\sqrt{T}}[f^{(0)}(\mathbf{x}_0) - f^{(0)}(\mathbf{x}_T)] + \frac{K^2 L_2 c}{2(L-1)\omega_{\min}\sqrt{T}} \\
&\leq \frac{1}{c\,\omega_{\min}(L-1)\sqrt{T}}[f^{(0)}(\mathbf{x}_0) - f_*^{(0)}] + \frac{K^2 L_2 c}{2(L-1)\omega_{\min}\sqrt{T}} \\
&\leq \frac{1}{\omega_{\min}(L-1)\sqrt{T}}\left(\frac{1}{c}[f^{(0)}(\mathbf{x}_0) - f_*^{(0)}] + \frac{L_2}{2}L^2 c\right).
\end{aligned} \tag{76}$$

□

F. Algorithms

In this section we give a summary presentation (with pseudo-codes) of the various variants of (S)GD introduced in the study.

F.1. PCNGD

Algorithm 1 PCNGD

1: Initialize $\mathbf{x}_0$
2: Split $\mathcal{D}$ into subgroups $\{\mathcal{D}_l\}$
3: **for** epoch $e \in [1, \dots, N_e]$ **do**
4: **for** $l \in [0, \dots, L-1]$ **do**
5: Calculate $\nabla f^{(l)}(\mathbf{x}_t)$ and $\|\nabla f^{(l)}(\mathbf{x}_t)\|$
6: **end for**
7: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \left(\sum_l \frac{\nabla f^{(l)}(\mathbf{x}_t)}{\|\nabla f^{(l)}(\mathbf{x}_t)\|} \right)$ {Update the set of parameters}
8: **end for**

After initializing the network weights, $\mathbf{x}_0$, (line 1 in Algorithm 1) we split the examples of the dataset according to their class (line 2 in Algorithm 1). For each epoch we calculate the gradient associated with the individual classes and the corresponding norm (line 4 in Algorithm 1). Finally, we use the calculated quantities to perform the update rule (line 5 in Algorithm 1).

F.2. PCNSGD

Algorithm 2 PCNSGD

1: Initialize $\mathbf{x}_0$
2: Split $\mathcal{D}$ into subgroups $\{\mathcal{D}_l\}$
3: **for** epoch $e \in [1, \dots, N_e]$ **do**
4: Shuffle $\{\mathcal{D}_l\}$
5: Group $\{\mathcal{D}_l\}$ into $\{\gamma_t^{(l)}\}_e$
6: **for** $i \in [1, \dots, N_b]$ **do**
7: **for** $l \in [0, \dots, L-1]$ **do**
8: Select $\gamma_t^{(l)}$
9: Calculate $\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$ and $\|\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)\|$
10: **end for**
11: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \left(\sum_l \frac{\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)}{\|\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)\|} \right)$ {Update the set of parameters}
12: **end for**
13: **end for**

After initializing the network weights, $\mathbf{x}_0$, (line 1 in Algorithm 2) we split the examples of the dataset according to their class (line 2 in Algorithm 2). At the beginning of each epoch we shuffle the elements of each subgroup (line 4 in Algorithm 2) and group them into per-class batches (line 5 in Algorithm 2). Note that the per-class batch sizes are set by the imbalance ratio; consequently the number of per-class batches is the same $\forall l$, *i.e.* $|\{\gamma_t^{(l)}\}| = N_b^{(l)} = N_b$.
We then begin to iterate over the batch index (line 6 in Algorithm 2); at each step we then select a per-class batch (line 8 in Algorithm 2) and calculate the gradient associated with it and its norm (line 9 in Algorithm 2). Finally, we use the calculated quantities to apply the update rule on the network weights (line 11 in Algorithm 2).

F.3. PCNSGD+O

After initializing the network weights, $\mathbf{x}_0$, (line 1 in Algorithm 3) we split the examples of the dataset according to their class (line 2 in Algorithm 3). At the beginning of each epoch we shuffle the elements of each subgroup (line 4 in Algorithm 3) and group them into per-class batches using the same per-class batch size, $|\gamma_t^{(l)}| = |\gamma_t| \; \forall l$ (line 5 in Algorithm 3). We

Algorithm 3 PCNSGD+O

1: Initialize $\mathbf{x}_0$
2: Split $\mathcal{D}$ into subgroups $\{\mathcal{D}_l\}$
3: **for** epoch $e \in [1, \dots, N_e]$ **do**
4: Shuffle $\{\mathcal{D}_l\}$
5: Group $\{\mathcal{D}_l\}$ into $\{\gamma_t^{(l)}\}_e$ {" 0 " is the label of the majority class}
6: **for** $i \in [1, \dots, N_b^{(0)}]$ **do**
7: **for** $l \in [0, \dots, L-1]$ **do**
8: **if** $i \% N_b^{(l)} = 0$ **then**
9: Regroup $\mathcal{D}_l$ into per-class batches
10: **end if**
11: Select $\gamma_t^{(l)}$
12: Calculate $\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$ and $\|\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)\|$
13: **end for**
14: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \left(\sum_l \frac{\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)}{\|\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)\|} \right)$ {Update the set of parameters}
15: **end for**
16: **end for**

then begin to iterate over the majority class batch index (line 6 in Algorithm 3). Note that since different classes have a different number of elements, $|\mathcal{D}_l|$, we will get a different number of batches for each of them: $|\{\gamma_t^{(l)}\}| = N_b^{(l)}$, with $N_b^{(0)} = \max_l N_b^{(l)}$ (" 0 " is the label of the majority class).
At each step we iterate along the classes. For each of them we check that per-classes batches are still available to use as input; if the per-class batches associated with a class l are finished we shuffle the elements into the subgroup $\mathcal{D}_l$ and define a new set of per-class-batches $\{\gamma_l\}$ (line 9 in Algorithm 3) as done (for each class) at the beginning of the epoch. After this check we then select a per-class batch γ_l (line 11 in Algorithm 3). We calculate the gradient associated with it and its norm (line 12 in Algorithm 3). Finally, we use the calculated quantities to apply the update rule on the network weights (line 14 in Algorithm 3).

F.4. SGD+O

Algorithm 4 SGD+O

1: Initialize $\mathbf{x}_0$
2: Split $\mathcal{D}$ into subgroups $\{\mathcal{D}_l\}$
3: **for** epoch $e \in [1, \dots, N_e]$ **do**
4: Shuffle $\{\mathcal{D}_l\}$
5: Group $\{\mathcal{D}_l\}$ into $\{\gamma_t^{(l)}\}_e$ {" 0 " is the label of the majority class}
6: **for** $i \in [1, \dots, N_b^{(0)}]$ **do**
7: **for** $l \in [0, \dots, L-1]$ **do**
8: **if** $i \% N_b^{(l)} = 0$ **then**
9: Regroup $\mathcal{D}_l$ into per-class batches
10: **end if**
11: Select $\gamma_t^{(l)}$
12: Calculate $\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$
13: **end for**
14: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \left(\sum_l \nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t) \right)$ {Update the set of parameters}
15: **end for**
16: **end for**

After initializing the network weights, $\mathbf{x}_0$, (line 1 in Algorithm 4) we split the examples of the dataset according to their class (line 2 in Algorithm 4). At the beginning of each epoch we shuffle the elements of each subgroup (line 4 in Algorithm 4) and group them into per-class batches using the same per-class batch size, $|\gamma_t^{(l)}| = |\gamma_t| \; \forall l$ (line 5 in Algorithm 4). We

then begin to iterate over the majority class batch index (line 6 in Algorithm 4). Note that since different classes have a different number of elements, $|\mathcal{D}_l|$, we will get a different number of batches for each of them: $|\{\gamma_t^{(l)}\}| = N_b^{(l)}$, with $N_b^{(0)} = \max_l N_b^{(l)}$ (" 0 " is the label of the majority class).
At each step we iterate along the classes. For each of them we check that per-classes batches are still available to use as input; if the per-class batches associated with a class l are finished we shuffle the elements into the subgroup $\mathcal{D}_l$ and define a new set of per-class-batches $\{\gamma_l\}$ (line 9 in Algorithm 4) as done (for each class) at the beginning of the epoch. After this check we then select a per-class batch γ_l (line 11 in Algorithm 4). We calculate the gradient associated with it (line 12 in Algorithm 4). Finally, we use the calculated quantities to apply the update rule on the network weights (line 14 in Algorithm 4).

F.5. PCNSGD+R

Algorithm 5 PCNSGD+R

1: Initialize $\mathbf{x}_0$
2: Split $\mathcal{D}$ into subgroups $\{\mathcal{D}_l\}$
3: **for** epoch $e \in [1, \ldots, N_e]$ **do**
4: Shuffle $\{\mathcal{D}_l\}$
5: Group $\{\mathcal{D}_l\}$ into $\{\gamma_t^{(l)}\}_e$.
6: **for** $i \in [1, \ldots, N_b]$ **do**
7: **for** $l \in [0, \ldots, L-1]$ **do**
8: Select $\gamma_t^{(l)}$
9: Calculate $\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)$ and $\|\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)\|$
10: Calculate $\nabla f^{(l)}(\mathbf{x}_t)$ and $\|\nabla f^{(l)}(\mathbf{x}_t)\|$
11: Compute $p_l = \left(\frac{\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)}{\|\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)\|}\right) \cdot \left(\frac{\nabla f^{(l)}(\mathbf{x}_t)}{\|\nabla f^{(l)}(\mathbf{x}_t)\|}\right)$
12: **end for**
13: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \left(\sum_l \frac{\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)}{p_l \|\nabla_{\tilde{n}} f^{(l)}(\mathbf{x}_t)\|}\right)$ {Update the set of parameters}
14: **end for**
15: **end for**

After initializing the network weights, $\mathbf{x}_0$, (line 1 in Algorithm 5) we split the examples of the dataset according to their class (line 2 in Algorithm 5). At the beginning of each epoch we shuffle the elements of each subgroup (line 4 in Algorithm 5) and group them into per-class batches (line 5 in Algorithm 5). Note that Per-class batch sizes are set by the imbalance ratio; consequently the number of per-class batches $|\{\gamma_t^{(l)}\}| = N_b^{(l)} = N_b$ is the same $\forall l$.
We then begin to iterate over the batch index (line 6 in Algorithm 5); at each step, for each class, we then select a per-class batch (line 8 in Algorithm 5) and calculate the gradient associated with it and its norm (line 9 in Algorithm 5). Next, we calculate the per-class gradient associated with the entire dataset and its norm (line 10 in Algorithm 5). We then calculate the projections of the normalized mini-batch gradients along the corresponding full-batch directions (line 11 in Algorithm 5). Finally, we use the calculated quantities to apply the update rule on the network weights (line 13 in Algorithm 5).

G. Models and data

The codes needed to reproduce the experiments presented are available in the following link: https://github.com/EmanueleFrancazi/PCNGD-Algorithms file, included in the repository, includes some details regarding the structure and operation of the scripts (how to set parameters, how seed initialization takes place, and so on).

G.1. Network architecture

We provide here more extended information about the network architectures. Before doing that, for the sake of clarity, we introduce some notation. To define a convolutional layer (as well as a pooling one) it is necessary to specify some parameters:

- **in channels**: Number of channels in the input originating from the preceding layer
- **out channels**: Number of channels produced by the convolution
- **stride**: Stride of the convolution
- **padding**: Padding added to all four sides of the input
- **kernel** : Size of the convolving kernel

We used three architectures for the simulations. In particular, we propose a simple network prototype (*Mod1*), and a two deeper ones as a prototype for more articulated models (*Mod2* and *Mod3*).

- *Simple-CNN* (*Mod1*): As first architecture we chose a convolutional neural network, whose architecture was fixed as follows:
  - Input data
  - Convolutional layer: **in channels**=3, **out channels** =16, **kernel** =5, **stride**=1, **padding**=2
  - activation function: ReLU
  - Max Pooling: **kernel** =2, **stride**=2
  - Convolutional layer: **in channels**=16, **out channels** =32, **kernel** =5, **stride**=1, **padding**=2
  - activation function: hyperbolic tangent
  - Average pooling: **kernel** =2, **stride**=2
  - Output: Fully connected linear layer
  - Loss function: Cross Entropy

  Before starting the network training, learning rate (*LR*) and batch size (*BS*)[8] values need to be fixed through hyperparameter tuning.

- *ResNet* (*Mod2*): as a second architecture we adopted *ResNet18* (He et al., 2016). The output of each convolutional layer is regularized by means of group normalization (Wu and He, 2018). The latter procedure, as opposed to the canonical batch normalization (Ioffe and Szegedy, 2015) is independent of the batch size. This aspect on the one hand poses no performance limitation for particular batch size choices (as pointed out in (Wu and He, 2018)). On the other it does not place an explicit dependence on the batch size parameter and thus allows, for example, gradient accumulation to be used. Indeed, such a technique is useful if one wants to forward a batch whose overall size saturates the memory of the machine, or even if one wants to split the batch for reasons of efficiency. In our case, the separation concerned elements belonging to different classes, so as to efficiently collect the gradient at associated with each of them. Group normalization needs an additional parameter to fix the number of blocks within which to group features (*GF*).

- *VGG* (*Mod3*): as a third architecture we adopted *VGG16* (Simonyan and Zisserman, 2014). Here, each convolutional layer is followed by a dropout layer. before passing through the dropout (DO) layer, the output of each convolutional layer is regularized by means of group normalization (Wu and He, 2018). Compared to *Mod1* we thus have in *Mod3* two additional HPs to be fixed through the HP validation process (DO rate and GF).

[8]The batch size HO is not present in GD runs

The HP tuning involved the optimization of batch size and learning rate, by exhaustive grid search. The optimal hyperparameters were chosen based on the macro-averaged recall.

G.2. Dataset and Hyper-parameters

The runs in this work were performed using data from the CIFAR10 and CIFAR100 datasets (Krizhevsky et al., 2009). Working only with CIFAR images allows us to attribute possible discrepancies between results to the imbalance choice, instead of the nature of the dataset. In order to create different imbalanced classification settings, we preprocessed the dataset in several ways:

- *Simple classes* (*Bi7a* , *Bi7b*): We selected two classes and discarded the others. We chose pairs of classes representing similar target items. The training dataset was composed of 5000 images belonging to the majority class and 714 to the minority one, so it had a 7:1 imbalance ratio. The test set was balanced, with 150 images per class. In particular, 2 different pairs of classes were analyzed:
  - *Bi7a* dataset: `truck` (majority class) and `car` (minority class),
  - *Bi7b* dataset: `horse` (majority class) and `deer` (minority class).
- *Cifar10 Super-classes (Bi60)*: In order to have a dataset of larger size and imbalance, we created two super classes, `animals` and `vehicles`. The first comprises the `bird`, `cat`, `deer`, `dog`, `frog` and `horse` labels (with an equal distribution), and the second encompasses `airplane`, `automobile`, `ship` and `truck` (with an equal distribution). The `animals` class contained 30000 elements, and the `vehicles` class contained 500 elements (60:1 imbalance). The testing data consisted of 600 elements per superclass. We called this the *Bi60* dataset.
- *Cifar10 Multi-class (Mul10)*: We use the 10 classes of the CIFAR10 dataset. The number of images associated with each class, N_i, is set by the relation:

$$N_i = N_{max} \left(\frac{3}{5}\right)^i \tag{77}$$

  where $N_{max} = 5000$ and i is the label associated with the class, *i.e.* a number between 0 and 9 that identifies the classes in alphabetical order (0: `airplane`, 1: `car`, 2: `bird`, 3: `cat`, 4: `deer`, 5: `dog`, 6: `frog`, 7: `horse`, 8: `ship`, 9: `truck`). We called this the *Mul10* dataset. The value of the base ($\frac{3}{5}$) was chosen so as to have a $\rho = 100$ imbalance ($N_0 \sim 100 N_9$), without excessively reducing the number of images in the least represented classes.
- *Cifar100 Multi-class (Mul100)*: Analogously to *Mul10* the number of images associated with each class, N_i, is set by the relation:

$$N_i = N_{max} \left(0.955\right)^i \tag{78}$$

  with $N_{max} = 500$ and $i \in [0 \,..\, 99]$ in this case. We called this the *Mul100* dataset. Also for *Mul100* the imbalance ratio is $\rho = 100$ ($N_0 \sim 100 N_{99}$).

For each of the datasets described above, the validation set was constructed similarly to the test set (same criteria for the composition and same size) but using a different subset of images.

In choosing the combination of datasets and architectures for the experiments, we used simple (complex) models for simple (complex) datasets to limit the number of experiments to a reasonable number.

A note on the meaning of the imbalance ratio We note that although the ratio $\rho = 100$ between number of examples in the majority and minority classes is the same as in the *Mul10* dataset and in the *Mul100* , in the latter contiguous classes have a very similar number of examples. This renders the *Mul100* dataset effectively less imbalanced. The learning problem remains hard because of the very small number of images per class, but this is unrelated to the dynamical effects induced by class imbalance that we are investigating in this paper. In general, the same value of ρ indicates a smaller imbalance when the number of classes is large.

Table 1. Summary of experiments with (PCN)GD dynamics. We report test recall and characteristic time τ (see main text in App. H.1), as well as the imbalance ratio ρ, and the threshold recall R^* used to calculate τ. We highlight in bold the best values of recall and τ. Details on the runs are in Sec. G, and the full learning curves for these runs are shown in Sec. H.

Models+Dataset	GD		PCNGD		# iterations	ρ	R^*
	Recall [%]	τ [steps]	Recall [%]	τ [steps]			
Mod1 + Bi7a	80.4 ± 0.4	272 ± 6	$\mathbf{82.8 \pm 0.4}$	$\mathbf{16.8 \pm 0.3}$	3000	7	0.7
Mod1 + Bi7b	79.8 ± 0.7	280 ± 10	$\mathbf{83.8 \pm 0.4}$	$\mathbf{12.5 \pm 0.2}$	3000	7	0.7
Mod1 + Bi60	73.3 ± 0.1	2200 ± 100	$\mathbf{87.6 \pm 0.5}$	$\mathbf{122 \pm 4}$	3500	60	0.7
Mod3 + Bi7a	81.0 ± 0.5	580 ± 20	$\mathbf{88.8 \pm 0.5}$	$\mathbf{200 \pm 5}$	1500	7	0.7
Mod1 + Mul10	41.6 ± 0.5	170 ± 6	$\mathbf{50.4 \pm 0.4}$	$\mathbf{15.5 \pm 0.2}$	1500	100	0.3
Mod2 + Mul100	13.5 ± 0.2	640 ± 60	$\mathbf{14.4 \pm 0.2}$	$\mathbf{310 \pm 50}$	1500	100	0.3

G.3. Execution times

Simulations were run on different servers. The code we provide allows the choice of using either the CPU or the GPU; for the experiments presented, either one or the other was used, depending on availability. The models of the GPUs mounted on the servers used are:

- GeForce- RTX 2080 Ti
- GP104GL Quadro P4000
- GM 200 GeForce GTX TitanX

Run times varied depending on the specific device model used and the level of occupancy (if multiple simulations are run in parallel on the same device). For all simulations run on GPUs, a single GPU was always used for each run. Other factors may cause the execution time to vary, such as the used batch size. A rigorous comparison of algorithm execution times should be made by setting same conditions for the various algorithms analyzed. This type of comparison is outside the scope of our analysis.
Instead, we report below a rough estimate of the order of magnitude of the execution times of the simulations performed. We try in this way to give an idea of the differences between the various algorithms. The values below refer to the average run time (expressed in seconds) normalized by the number of iterations ($\frac{sec}{iteration}$ units) .
In the case of *Mod1* the times are lower. For the deterministic case GD and PCNGD proceed with similar speeds when run on the same devices. In particular we have an execution time that varies between $\sim 5\frac{sec}{iteration}$ for *Bi7b* to $\sim 30\frac{sec}{iteration}$ for *Mul10* .
In the stochastic case we have a large gap between execution times of PCNSGD+R and the rest of the algorithms. PCNSGD+R runs with a speed of $\sim 4\frac{sec}{iteration}$. The remaining algorithms, on the other hand, with a speed of $\sim 0.2\frac{sec}{iteration}$.
Execution time increases considerably by moving to a more complex architecture. For example, for the *Mod3* architecture we have an execution time of $\sim 60\frac{sec}{iteration}$ for the deterministic algorithms and $\sim 1.5\frac{sec}{iteration}$ for the stochastic ones. For future use, the code records the execution times of the simulations on file.

H. Additional experiments

This section reports the results of additional experiments performed on the data sets described in App. G.2. The loss curves are reported together with the corresponding recall, since it is a clearly interpretable metric of interest for practical applications. Unless specified otherwise, we report the macro-averaged curves, which is the average of the per-class ones. All shown curves are averaged over between 4 and 30 random seeds (standard error bars are shown as a shading), depending on how demanding the run was. For the dynamics we used a further different random seed.

H.1. Deterministic optimization

Here, we show more plots on the comparison between GD and PCNGD.

Fig. S7 shows the recall and loss for several combinations of the models and datasets presented in G.2. The careful reader can remark that the learning curves for PCNGD are not strictly monotonic as one could expect from Theorem 4.2. The

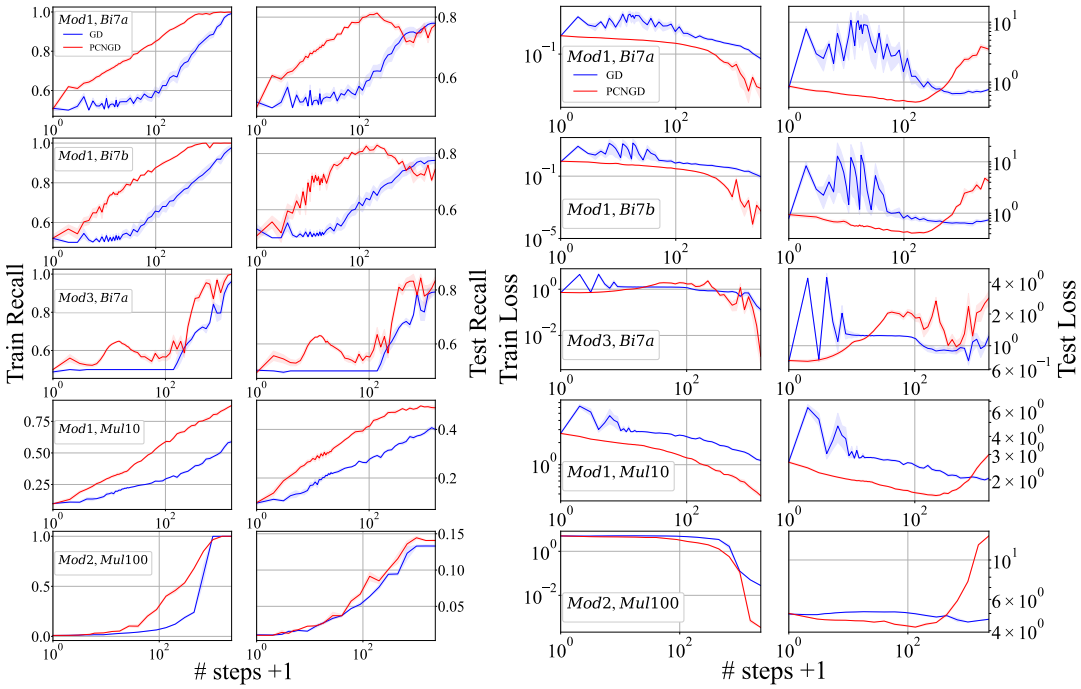

Figure S7. Recall and Loss function for GD and PCNGD algorithms.

reason is that the monotonous decrease of the loss is only guaranteed for sufficiently small learning rates. For some of our runs, the HP tuning yielded larger learning rates, for which the per-class loss is non-monotonic. This reveals a trade-off between a strictly monotonic decrease of the loss and a larger step size.

In Fig. S10 we show an example of the per-class curves in a multiclass problem. As in the binary setting, the GD dynamics exhibit the MID, which is instead not there with PCNGD.

We summarize the results of these experiments in Tab. 1, where we show the peak test recall after a fixed number of iterations, and the number of time steps, τ[9], that the model requires before the test recall reaches a value R^*, also reported on the table. While the test recall is an indicator of the performance at convergence, τ is an indicator of the speed of convergence, which is particularly relevant when one needs to assess performance over a limited number of iterations, as in hyperparameter tuning or when training is especially expensive. We also indicate the total number of iterations performed in each run, and the imbalance ratio, ρ, defined as the ratio between the number of examples of majority and minority classes.[10] From Tab. 1, we see that PCNGD systematically outperforms GD in terms of convergence speed, in agreement with our theory. In addition, also the test recall is consistently higher: we elaborate on this in Sec. 4.5.

H.2. Stochastic optimization

Here we show additional experiments regarding stochastic optimization algorithms, as described in the main part of the paper. In Fig. S8, the recall and loss curves are shown for the Model: *Mod1*, dataset: *Bi60*. In Fig. S9 we show macro averaged recalls and loss functions for stochastic algorithms. We do not always show the curves for PCNSGD+R for all the experiments because they are computationally expensive. In most of the cases presented, PCNSGD+O outperforms all the other solutions. The initial spike in the loss for the SGD+O algorithm can be reduced by decreasing the learning rate, but at the expense of a worse final performance. The improvement of PCNSGD+O with respect to the other algorithms is more marked when the data imbalance is increased, as we show in Fig. S8, where there is a significant gain with respect to SGD+O along the entire dynamics. Similar to the deterministic case, the results of the various experiments are summarized

[9]Since measurements during the dynamics are not taken at every time step, errors on the quantity τ are the $\min$ between the standard error and the interval between measurements.

[10]We use this definition because it is the one usually adopted in the literature. However, with this definition, the same value of ρ indicates a larger imbalance when the number of classes is small, than when it is large.

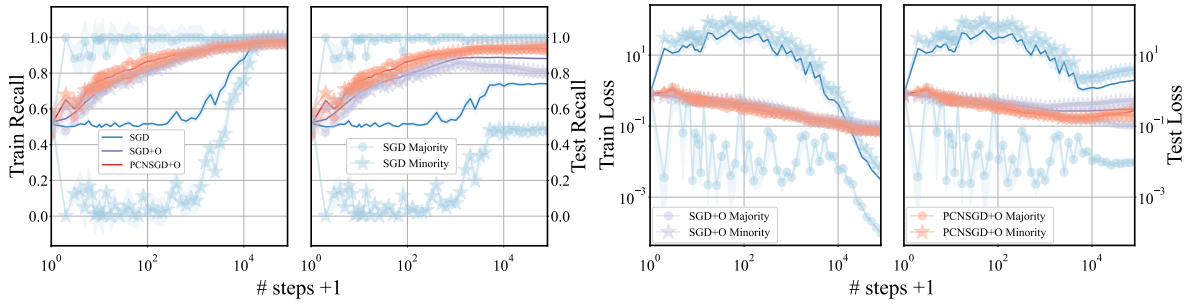

Figure S8. Recall and Loss function for GD and PCNGD algorithms. Model: *Mod1*, dataset: *Bi60*. The SGD per-class curves clearly display the MID, while those related to the two oversampled algorithms follow closely the macro-averaged trends.

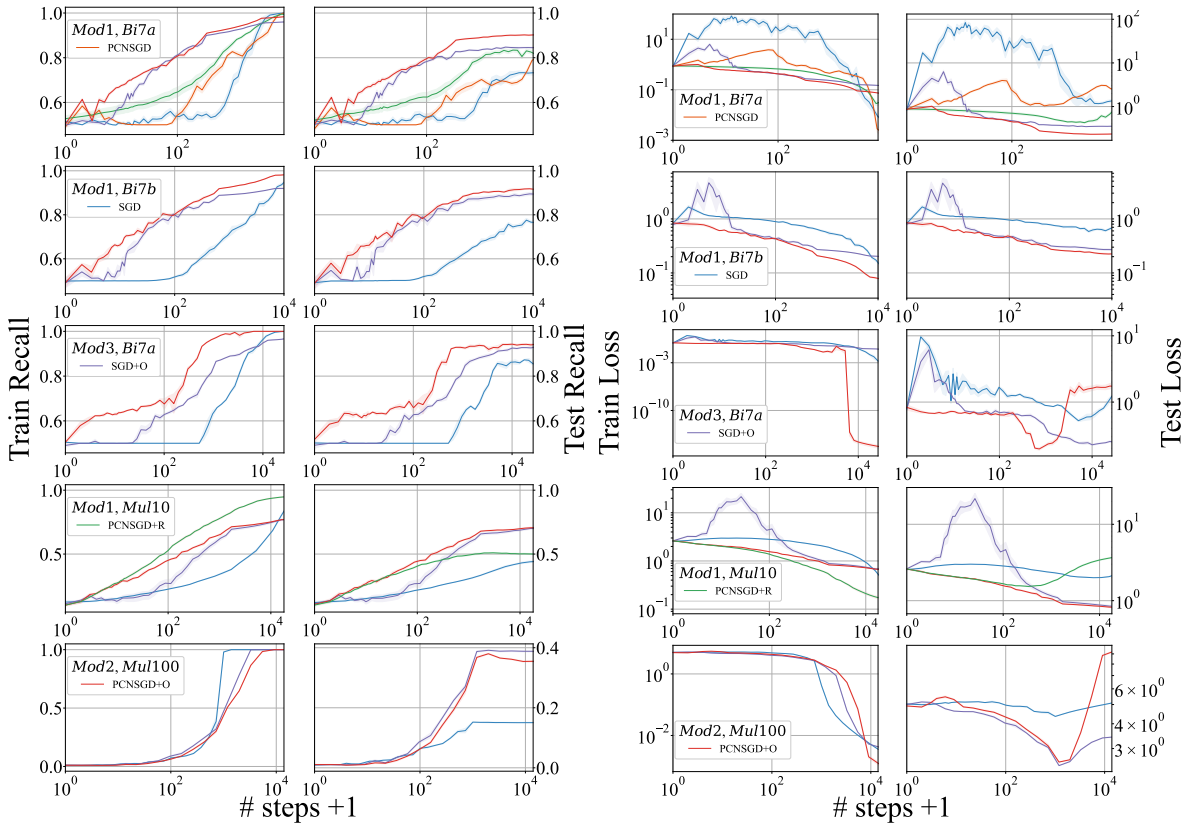

Figure S9. Recall and Loss function for SGD and PCNSGD algorithms.

in Tab. 2. We do not report PCNSGD+R on Tab. 2 because of its high computational cost, but we still show some extra PCNSGD+R runs in Fig S9.

The effect of PCNSGD+R on the single classes In Fig. S11 we show a comparison of the per-class learning curves of SGD and PCNSGD+R. The improvement in class-averaged performance reflects, as in the deterministic case, a more balanced growth in single-class performance. In SGD, the beginning of the dynamics is characterized by a sudden growth of the majority classes and a degrowth of the minority ones, analogously to the *minority initial drop* we showed in Fig. 1 in the case of binary classification. After the initial boost, the growth of the majority classes' recall slows down and the remaining classes gradually begin to grow following the order set by the imbalance of the data. In the case of PCNSGD+R, on the contrary, a more regular growth is observed among the various classes. Note that, while in SGD the fastest classes are the majority ones, with PCNSGD+R the fastest are the minority, but the majority class is not the slowest.

Table 2. Same as Tab. 1, but for (PCN)SGD dynamics.

Models+Dataset	SGD		SGD+O		PCNSGD+O		# iterations	ρ	R^*
	Recall	τ	Recall	τ	Recall	τ			
Mod1 + Bi7a	77.0 ± 0.8	1400 ± 300	84.9 ± 0.2	20.3 ± 0.4	$\mathbf{90.9 \pm 0.2}$	$\mathbf{14.5 \pm 0.2}$	10000	7	0.7
Mod1 + Bi7b	78.4 ± 0.5	2200 ± 100	90.3 ± 0.5	24 ± 1	$\mathbf{92.3 \pm 0.2}$	$\mathbf{14.9 \pm 0.6}$	10000	7	0.7
Mod1 + Bi60	81 ± 1	1540 ± 60	89.3 ± 0.1	10.8 ± 0.2	$\mathbf{94.2 \pm 0.1}$	$\mathbf{5.4 \pm 0.1}$	80000	60	0.7
Mod3 + Bi7a	88.3 ± 0.3	2400 ± 100	93.5 ± 0.2	410 ± 20	$\mathbf{95.1 \pm 0.3}$	$\mathbf{111 \pm 6}$	25000	7	0.7
Mod1 + Mul10	44.6 ± 0.5	5300 ± 500	71.1 ± 0.7	250 ± 10	$\mathbf{71.8 \pm 0.4}$	$\mathbf{62 \pm 4}$	18000	100	0.4
Mod2 + Mul100	15.2 ± 0.1	420 ± 30	$\mathbf{39.2 \pm 0.2}$	$\mathbf{140 \pm 20}$	38.0 ± 0.2	210 ± 30	18000	100	0.1

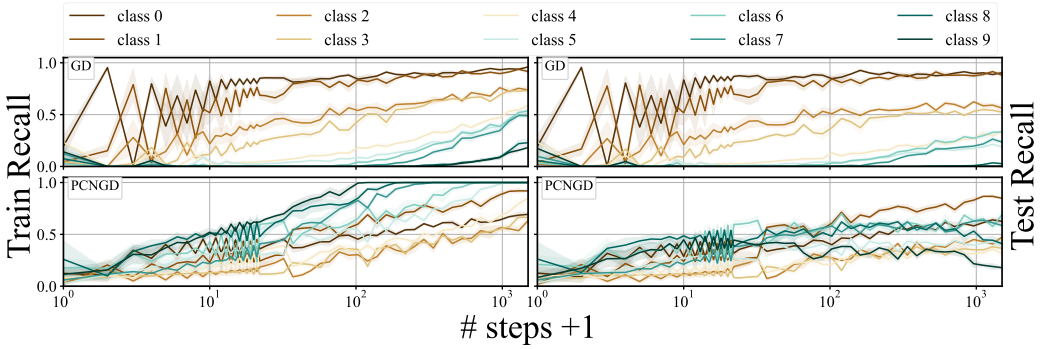

Figure S10. Comparison GD (top) and with PCNGD (bottom) Model: *Mod1*, dataset: *Mul10*. The majority class is class 0, the minority class is class 9.

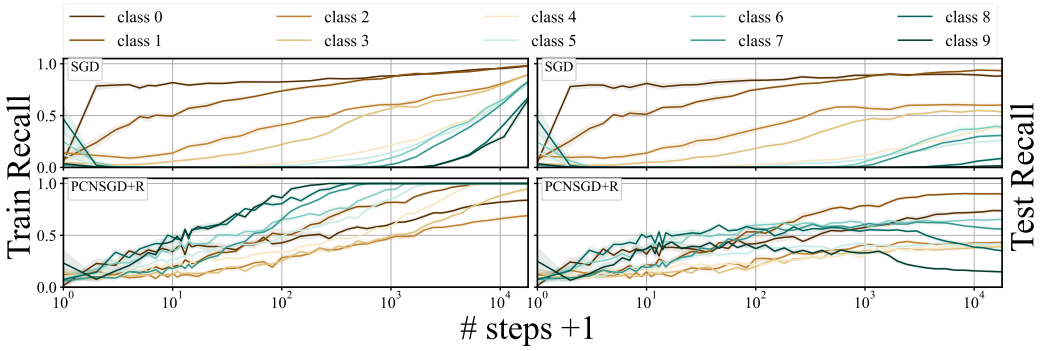

Figure S11. Comparison SGD (top) and with PCNSGD+R (bottom) Model: *Mod1*, dataset: *Mul10*. The majority class is class 0, the minority class is class 9.

I. Limitations and Ethics

Limitations We can identify the following limitations to our work:

- Our theory does not cover dropout, but it could probably be extended by trying to include the averaging effect of Dropout. Alternatively, one could use the formulation of (Arora et al., 2020) that shows that the Dropout objective is a regularized version of the original objective. In addition, we did run experiments on a VGG network (which has dropout), the *Mod3* model, and found results that are consistent with the rest of the architectures.

- Our theory does not cover non-differentiable activations: There are various standard ways to deal with non-differentiable functions in optimization, *e.g.* subgradients, smoothing techniques, etc. We expect that all these techniques are applicable to PCN(S)GD, and that the convergence theorems can be adapted accordingly. Furthermore, note that the models *Mod1* and *Mod2* have ReLU activations (which is non-differentiable at a single point), so at least empirically we see that our results also hold with ReLUs.

Ethics Class imbalance, often present in real datasets used to train neural networks, not only impacts performance but also brings with it a number of ethical questions related to representative discrimination and fairness (Mehrabi et al., 2021; Buolamwini and Gebru, 2018; Danks and London, 2017; De Vries et al., 2019; Yapo and Weiss, 2018).

The CIFAR datasets, are subsets of the 80 million tiny images, which is formally withdrawn since it contains some derogatory terms as categories and offensive images (http://groups.csail.mit.edu/vision/TinyImages/). However, note that none of the experiments described in the paper was performed on the overall tiny images dataset: the said derogatory images are not present in CIFAR10 and CIFAR100.