Characterizing the Effect of Class Imbalance on the Learning Dynamics

Emanuele Francazi  
Physics Department  
EPFL  
CH-1015 Lausanne, Switzerland  
emanuele.francazi@epfl.ch

Marco Baity-Jesi  
SIAM Department  
Eawag (ETH)  
8600 Dübendorf, Switzerland  
marco.baityjesi@eawag.ch

Aurelien Lucchi  
Department of Mathematics and Computer Science  
University of Basel  
4051 Basel, Switzerland  
aurelien.lucchi@unibas.ch

Abstract

Data imbalance is a common problem in the machine learning literature that can have a critical effect on the performance of a model. Various solutions exist – such as the ones that focus on resampling or data generation – but their impact on the convergence of gradient-based optimizers used in deep learning is not understood. We here 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. The reason is not only that the gradient signal neglects the minority classes, but also that the minority classes are subject to a larger directional noise, which slows their learning by an amount related to the imbalance ratio. To address this problem, we propose a new algorithmic solution, for which we provide a detailed analysis of its convergence behavior. We show both theoretically and empirically that this new algorithm exhibits a better behavior with more stable learning curves for each class, as well as a better generalization performance.

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]].

The problem of data imbalance has already received some attention in the literature and we broadly identify three most common ways to handle it. These either act on the data distribution, adjust the objective function or modify the learning algorithm. We provide a detailed discussion of these methods in Sec. 2. 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 discuss the theoretical convergence guarantees of these methods, which, to the best of our knowledge, is a novel contribution.

Preprint. Under review.
We start our investigation from the following two empirical observations: i) the learning dynamics is delayed for imbalanced problems and, ii) while the overall performance improves during 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. One can argue why this happens: 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. In fact, due to the imbalance, decreasing the loss related to the majority class outweighs the loss increase coming from the minority. The minority classes will only be learned once the gradient calculated on 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 have a monotonic decrease in the loss function of each class. In section 3, we first prove that vanilla GD is indeed not guaranteed to monotonically decrease the per-class loss. We then demonstrate that a simple modification of GD, that we name Per-Class Normalized GD (PCNGD), can correct for this deficiency (Fig. 1, red curves). In Sec. 4, we show that the situation is more complicated for SGD, because the imbalance also influences the direction of the updates, in addition to their norm. In this case, we will see that a solution is to combine class-normalized updates with oversampling.

We identify our key contributions as:

1. Analytical and numerical evidence that in unbalanced problems the minority classes do not have a monotonous loss function decrease. This is due to the signal of minority classes being negligible with respect to that of majority classes until later stages of learning. This minority initial drop implies that at initial stages of the dynamics, the minority classes are identified with worse than random recall: as far as what concerns the minority class, no training is better than short training.

2. The introduction of Per-Class Normalized Gradient Descent (PCNGD): this algorithm combines the normalized loss gradient restricted to each separate class. In this way the intensity of the signal related to each class is equalized and the minority initial drop is avoided.

---

1We show in Fig. S14 that the minority initial drop occurs also with SGD and multiclass classification.
• An analysis of the theoretical convergence of PCNGD showing that, for a suitable step-size, PCNGD is guaranteed to have a monotonic decrease in the loss of all classes, and empirical evidence that PCNGD performs better than GD, in terms of both overall and per-class indicators.

• Analytical evidence showing that, while normalizing the per-class gradients is sufficient to counter class imbalance in GD, this is not true for SGD, due to the fact that data imbalance not only influences the intensity of the signal related to each class, but also its direction.

• A stochastic variant of PCNGD, that takes into account the effect of data imbalance on both the intensity and the direction of the gradient updates; with empirical evidence that it outperforms SGD on unbalanced datasets. The analysis underlying the formulation of these algorithms also clarifies the effectiveness of some approaches already used heuristically, such as oversampling methods.

2 Related work

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 some 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.

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]. 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].

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., 2019]; perturbing the inputs of the majority class [Ye et al., 2021]; or by adapting the direction of the gradient steps in order to suppress the domination of the majority class [Anand et al., 1993].

This last solution, noticing that the gradient magnitude of the minority class is smaller than that of the majority class, takes the bisection between the per-class gradients. The optimization steps are taken in the same direction as in our PCNGD, but with a different modulus.

As discussed in [Johnson and Khoshgoftaar, 2019], all these methods achieve different levels of performance depending on a multitude of conditions (classifier, performance metric, etc). Importantly, these methods do not have well-understood convergence guarantees.

Finally, [Ye et al., 2021] notices that in imbalanced problems with SGD dynamics the minority classes are learned later than the majority classes. We extend this observation to gradient descent and highlight that this is due to an undesired effect at the beginning of learning, which makes the identification of the minority classes worse than random. Also, although minority classes can be learned at late times, our algorithmic solutions (and those of the works we cite in this section) aim at learning them in the initial stages of the dynamics, and at avoiding learning time scales that grow with the class imbalance.
3 Monotonic optimization through per-class normalization

3.1 Setting

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

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

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:

\[
f^{(l)}(x) := \frac{1}{n} \sum_{i \in C_l} f_i(x), \quad C_l = \{i \mid y_i = l\}, \quad l \in [L].
\]

3.2 Gradient Descent

In the following theorem, we start by analyzing the convergence of gradient descent run for \( T \) iterations on the loss corresponding to each class \( f^{(l)}(x) \). 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 straightforwardly extendable to the case where \( L > 2 \).

To make our results more accessible, we give an informal version of our theorems in this section. We invite the reader to check App. A and B for a formal version. Since we do not assume that the objective function \( f \) is convex, we focus on showing that the gradient norm decreases with the number of iterations (which is the standard metric in the optimization literature, see e.g. Ghadimi and Lan [2013]). Note that we use the standard notation \( \angle(x, y) \) to denote the angle between two vectors \( x, y \in \mathbb{R}^m \).

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

\[
\min_{s \in [0,T-1]} \|\nabla f^{(1)}(x_s)\|^2 \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right).
\]

Theorem 1 requires a potentially restrictive condition on the angle \( \alpha(x_t) \) in order to guarantee a decrease of the loss of both classes. While the above result provides an upper bound, we next discuss the tightness of this condition, effectively demonstrating that one can not avoid this problem in the general case.

**Tightness of the upper bound** Consider the case where the loss function \( f \) is a quadratic function with a constant diagonal Hessian \( H \) where all eigenvalues of \( H \) are equal to \( L \). By a Taylor expansion
We see that in order to decrease \( f \), we have

\[
\begin{align*}
    f^{(0)}(x_{t+1}) &= f^{(0)}(x_t) + \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \| x_{t+1} - x_t \|^2 \\
    &= f^{(0)}(x_t) - \eta_t \langle \nabla f^{(0)}(x_t), \nabla f(x_t) \rangle + \frac{L\eta_t^2}{2} \| \nabla f(x_t) \|^2 \\
    &= f^{(0)}(x_t) - \eta_t \left( \frac{\eta_t}{2} \| \nabla f^{(0)}(x_t) \|^2 - \eta_t \langle \nabla f^{(0)}(x_t), \nabla f^{(1)}(x_t) \rangle + \frac{L\eta_t^2}{2} \| \nabla f^{(0)}(x_t) \|^2 \right) \\
    &\quad + \frac{L\eta_t^2}{2} \| \nabla f^{(1)}(x_t) \|^2 + L\eta_t^2 \langle \nabla f^{(0)}(x_t), \nabla f^{(1)}(x_t) \rangle \\
    &= f^{(0)}(x_t) - \eta_t \left( 1 - \frac{L\eta_t}{2} \right) \| \nabla f^{(0)}(x_t) \|^2 - \eta_t (1 - L\eta_t) \langle \nabla f^{(0)}(x_t), \nabla f^{(1)}(x_t) \rangle \\
    &\quad + \frac{L\eta_t^2}{2} \| \nabla f^{(1)}(x_t) \|^2.
\end{align*}
\]

Let \( C_t := \frac{\| \nabla f^{(1)}(x_t) \|}{\| \nabla f^{(0)}(x_t) \|} \). We get

\[
f^{(0)}(x_{t+1}) = f^{(0)}(x_t) - \eta_t \left( 1 - \frac{L\eta_t}{2} - L\eta_t C_t^2 / 2 + (1 - L\eta_t) \cos(\alpha(x_t)) C_t \right) \| \nabla f^{(0)}(x_t) \|^2.
\]

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

\[
1 - (1 + C_t^2) L\eta_t / 2 + (1 - L\eta_t) \cos(\alpha(x_t)) C_t > 0.
\]

Taking \( \eta_t = \frac{c}{\sqrt{T}} \) for \( c > 0 \), we see that we require the following condition on the step-size:

\[
c \leq \frac{(1 + \cos(\alpha(x_t)) C_t) \sqrt{T}}{(1 + C_t^2) L/2 + L \cos(\alpha(x_t)) C_t},
\]

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

Interestingly, the condition under consideration has a rather intuitive meaning. Indeed, one can see that it depends on the imbalance ratio of the norms of the per-class gradients denoted by \( C_t \). In the case where the two norms are equal, the condition is trivially satisfied. 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 rather suboptimal behavior if one equally cares about the loss of each class.

### 3.3 Per-Class Normalized Gradient Descent

We present a new algorithm, named PCNGD, that, starting from an initial guess \( x_0 \), iteratively updates the parameter \( x_t \) as follows,

\[
x_{t+1} = x_t - \eta_t \left( \frac{\nabla f^{(0)}(x_t)}{\| \nabla f^{(0)}(x_t) \|} + \frac{\nabla f^{(1)}(x_t)}{\| \nabla f^{(1)}(x_t) \|} \right).
\]

(\text{PCNGD})

Our interest is to demonstrate that PCNGD allows for a monotonic convergence 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. In the following and in App. \text{[B]} we provide three theorems that prove convergence of PCNGD under different conditions.

\begin{itemize}
    \item \textbf{Theorem 2} (informal). Assume that each \( f^{(l)} \) for \( l = 0, 1 \) is \( L_1 \)-Lipschitz and \( L_2 \)-smooth. Then if \( \cos(\alpha(x_t)) \neq -1 \) for all iterations \( t \), the iterates of gradient descent with step size \( \eta_t = \mathcal{O} \left( \frac{1}{\sqrt{T}} \right) \) satisfy

\[
\min_{s \in [0,T-1]} \| \nabla f^{(l)}(x_s) \| \leq \mathcal{O} \left( \frac{1}{\sqrt{T}} \right).
\]
\end{itemize}
**Proof sketch.** We only give a sketch of the proof for \( f^{(0)} \), the details are given in App. B.

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

\[
 tf^{(0)}(x_{t+1}) \leq f^{(0)}(x_t) + \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2.
\]

Using the update step of PCNGD, we can show that

\[
 f^{(0)}(x_{t+1}) \leq f^{(0)}(x_t) - \eta_t (1 + \cos \alpha(x_t)) \|\nabla f^{(0)}(x_t)\| + 2L_2\eta_t^2.
\]

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

\[
 \|\nabla f^{(0)}(x_t)\| \leq \frac{1}{\eta_t \omega_{\min}} [f^{(0)}(x_t) - f^{(0)}(x_{t+1})] + \frac{2L_2\eta_t^2}{\omega_{\min}}.
\]

Taking the minimum over \( t \), we get

\[
 \min_{s \in [0, T-1]} \|\nabla f^{(0)}(x_s)\| \leq \frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f^{(0)}(x_t)\|.
\]

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

\[\Box\]

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 \( \cos \alpha(x_t) \neq -1 \) implies that the two gradients are not allowed to be in completely opposite directions, which is a much milder assumption than the one required for GD in Theorem 1.

We also give a different version of the Theorem for different step sizes and for randomized iterates (as done in Ghadimi and Lan [2013]) in App. B.

### 3.4 Empirical evidence in favor of PCNGD

We provide numerical experiments comparing PCNGD to GD. We used two different architectures,

- **Mod1.** A three-layer CNN without dropout,
- **Mod2.** A VGG with dropout and group normalization,

and the following imbalanced binary and multiclass datasets:

- **Bi7a.** We take the truck and car classes from cifar10, with a 7:1 imbalance, with the truck being the majority class.
- **Bi7b.** Same as Bi7a, but with the horse (majority) and deer (minority) classes.
- **Bi60.** We superclass cifar10 into animals and vehicles. This allows us to create a more imbalanced dataset (60:1 ratio), with animals as the majority class.
- **Mul10.** We take the 10 classes from cifar10, with an exponentially decreasing of examples per class. The \((i+1)\)th class has \( \frac{2}{3} \) of the examples of the \(i\)th. This way, the majority class has 5000 examples, and the minority has 50.

In all cases, while the training set is unbalanced, the test set is balanced. We give full details about the models in App. C.1 and about the data in App. C.2. Throughout the main paper, we only show the recall curves, since it is a clearly interpretable metric of interest for practical applications. Unless specified otherwise, we are reporting the macro-averaged recall, which is the average of the per-class recalls. All shown curves are averaged over between 4 and 30 different choices of the initial conditions (standard error bars are shown as a shading), each with a different random number seed.

\[\text{Anonymous GitHub}\]
Figure 2: Per-class train recall of PCNSGD and of PCNSGD+R with Mod1 on the Bi7a dataset. The rescaling proposed in Eq. (6), leading to PCNSGD+R, solves the problem of minority initial drop. The macro-averaged curves are shown in Fig. 3.

for the dynamics. In the main text, we provide a single example of model and data for each section, but we provide more empirical evidence for various data and models in App. D along with the loss curves.

We show a comparison between GD and PCNGD with a 60:1 imbalance in Fig. 1. In GD, within the first epoch, the recall of the majority class goes to 1, and that of the minority class drops to 0. The intuitive reason behind this is that with GD in an unbalanced dataset, the gradient points predominantly towards 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 gradient of the majority class needs to become smaller by an amount that scales with the imbalance ratio $R$.

With PCNGD, instead, the minority initial drop is mitigated, since both classes are optimized from the beginning of the run. After the first $\sim 200$ 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 time interval of the experiment (3500 epochs).

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

4.1 Imbalance causes directional noise

We now turn our attention towards designing 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. We call the resulting algorithm Per-Class Normalized SGD (PCNSGD), whose update is simply defined as

$$x_{t+1} = x_t - \eta_t \left( \frac{\nabla_{\hat{n}} f^{(0)}(x_t)}{\|\nabla_{\hat{n}} f^{(0)}(x_t)\|} + \frac{\nabla_{\hat{n}} f^{(1)}(x_t)}{\|\nabla_{\hat{n}} f^{(1)}(x_t)\|} \right),$$

(PCNSGD)

where $\nabla_{\hat{n}} f^{(l)}(x_t)$ is the gradient over a random batch of size $\hat{n}$, for points belonging to the class $l$. In contrast to PCNGD, we empirically observed that PCNSGD1 still suffers dramatically from the minority initial drop, despite the correction on the gradient norms (Figs. 2 and 3 (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 $\nabla f^{(l)}(x_t)$ as the mean gradient direction corresponding to class $l$ on the population loss, and we call $\nabla f^{(l)}(x_t)$ the full-batch mean gradient relative to class $l$. More generally, given an arbitrary set $S$ of

\footnote{Therefore, we expect that the time required to learn the minority class scales with $R$.}

\footnote{Since Eq. (PCNSGD) requires each mini-batch to contain at least one element of each class, we discard the batches that do not contain both classes and reshuffle the data at every epoch.}
\( n \) elements, all belonging to the same generic class \( l \), the notation \( \nabla f_{\hat{n}}^{(l)}(x_t) \) indicates:

\[
\nabla f_{\hat{n}}^{(l)}(x_t) := \frac{1}{\hat{n}} \sum_{i \in S} \nabla f_i(x_t)
\]

where \( f_i(x_t) \) is the loss function corresponding to element \( i \in S \). If the underlying gradient distribution has a finite variance by the Central Limit Theorem (CLT) the per-class gradient calculated on \( \hat{n} \) examples can be written as fluctuations around the population gradient,

\[
\nabla f_{\hat{n}}^{(l)} = \nabla f_{\infty}^{(l)} + \frac{1}{\sqrt{\hat{n}}} Z^{(l)},
\]

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

We want to compare \( \nabla f_{\hat{n}}^{(l)} \) with the full-batch gradient, in the regime where for both classes the batch size is much smaller than the size of the dataset, i.e. \( \hat{n}_l \ll n' \forall l, l' = 0, 1 \). In this case, we can use the CLT again, obtaining

\[
\nabla f_{\hat{n}}^{(l)} \approx \nabla f_{\hat{n}}^{(l)} + \frac{1}{\sqrt{\hat{n}_l}} Z^{(l)},
\]

where we neglected terms of order \( 1/\sqrt{\hat{n}_l} \).

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. (3), the PCNSGD steps can be written as

\[
\sum_{l} \frac{\nabla f_{\hat{n}}^{(l)}}{\|\nabla f_{\hat{n}}^{(l)}\|} \simeq \sum_{l} \frac{\nabla f_{\hat{n}}^{(l)}}{\|\nabla f_{\hat{n}}^{(l)}\|} + \sum_{l} \frac{Z^{(l)}}{\sqrt{\hat{n}_l} \|\nabla f_{\hat{n}}^{(l)}\|} + \sum_{l} \frac{Z^{(l)}}{\sqrt{\hat{n}_l} \|\nabla f_{\hat{n}}^{(l)}\|}.
\]

We want to see how these steps project along the direction of the full-batch gradient, for which Theorem 2 shows that we can obtain a monotonous decrease for both classes. The projection of these steps onto the full-batch gradient is

\[
\left( \frac{\nabla f_{\hat{n}}^{(l)}}{\|\nabla f_{\hat{n}}^{(l)}\|} \right) \cdot \nabla f_{\infty}^{(l)} = \left( \frac{\nabla f_{\hat{n}}^{(l)}}{\|\nabla f_{\hat{n}}^{(l)}\|} \right) - \left( \frac{\nabla f_{\hat{n}}^{(l)} \cdot Z^{(l)}}{\sqrt{\hat{n}_l} \|\nabla f_{\hat{n}}^{(l)}\|^3} \right) \frac{\nabla f_{\hat{n}}^{(l)} \|Z^{(l)}\|^2}{2 \hat{n}_l \|\nabla f_{\hat{n}}^{(l)}\|^3} + \frac{3 \nabla f_{\hat{n}}^{(l)} \cdot Z^{(l)} \|Z^{(l)}\|^2}{2 \hat{n}_l \|\nabla f_{\hat{n}}^{(l)}\|^4} + \frac{Z^{(l)} \left( \nabla f_{\hat{n}}^{(l)} \cdot Z^{(l)} \right)}{\hat{n}_l \|\nabla f_{\hat{n}}^{(l)}\|^3} + o \left( \frac{1}{\hat{n}_l} \right) \right) \cdot \nabla f_{\infty}^{(l)}
\]

\[
= 1 - \frac{\|Z^{(l)}\|^2 (1 - \cos(\theta)^2)}{2 \hat{n}_l \|\nabla f_{\hat{n}}^{(l)}\|^2} + o \left( \frac{1}{\hat{n}_l} \right),
\]

where \( \theta \) indicates the angle between \( Z^{(l)} \) and \( \nabla f_{\hat{n}}^{(l)} \).

Eq. (5) shows that the larger the number of examples of class \( l \) in the mini-batch, the closer the steps are to the PCNGD direction. 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. 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 \( \|Z^{(l)}\|^2 (1 - \cos(\theta)^2) \). At the beginning of learning, with random initial conditions, we can expect \( \|\nabla f_{\hat{n}_0}^{(l)}\| = \|\nabla f_{\hat{n}_1}^{(l)}\| \), and we can expect that on average the noise fluctuations have a similar projection onto \( \|\nabla f_{\hat{n}}^{(l)}\| \). Consequently, the attenuation of the minority with respect to the majority signal is proportional to the imbalance ratio, \( \frac{n_0}{\hat{n}_0} \). Once the per-class gradient of the majority class has converged, by Eq. (3), we have \( \frac{\nabla f_{\hat{n}}^{(l)}}{\|\nabla f_{\hat{n}}^{(l)}\|} \simeq \frac{Z^{(l)}}{\|Z^{(l)}\|} \), and the signal of the minority class can become relevant.

---

This condition can be relaxed by using a generalized CLT [Darling 1956].
4.2 Balancing the directional noise

Here, we discuss and empirically test different ways of dealing with the problem we raised in Sec. 4.1.

**Per-class normalization with Rescaling (PCNSGD+R)**  The first option is to enforce that both majority and minority mini-batch gradients project by the same amount onto the full-batch signals. This is done by rescaling the minority-class signal by the appropriate amount, calculated from Eq. (5).

The rescaling factor (we label with 0 the minority and with 1 the majority class),

\[
\alpha = \left(1 - \frac{\|Z^{(1)}\|^2 \sin^2(\theta_1)}{2n_1\|\nabla f^{(1)}_{n_1}\|^2}\right) / \left(1 - \frac{\|Z^{(0)}\|^2 \sin^2(\theta_0)}{2n_0\|\nabla f^{(0)}_{n_0}\|^2}\right),
\]

relies on the calculation of the full-batch gradients. To alleviate this burden we only calculate them every 5 steps. This algorithm is not meant as an alternative to SGD, but rather to provide empirical evidence that the arguments in Sec. 4.1 are correct. Indeed, we show in Fig. 2 that, after this further rescaling, the *minority initial drop* is prevented: the minority class recall is monotonically increasing since the beginning of learning, and the majority and minority classes have a similar evolution.

**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 oversampling the minority class, with the difference that we enforce that every batch is *exactly* split between the two classes.

**Per-class normalization with Oversampling (PCNSGD+O)**  Finally, we check whether there is a gain in combining both per-class normalization and oversampling.

4.3 Empirical evidence in favor of PCNSGD

In Fig. 3 we compare the described algorithmic solutions. PCNSGD+R has a steady growth since the beginning of learning, as expected from our theory. While it performs better than SGD and PCNSGD, it is outperformed by SGD+O and PCNSGD+O both in terms of convergence speed and final test recall. However, PCNSGD+O outperforms SGD+O, because both the recall at short times and the final test recall are higher. The advantages of PCNSGD+O are even more visible with higher imbalance (Fig. S11).

To rationalize the better generalization of PCN algorithms, we can look at the fixed points of the dynamics. In GD, fixed points satisfy \(\nabla f^{(0)}_{n_0} = -\nabla f^{(1)}_{n_1}\). If the training set is imbalanced with an imbalance ratio \(R\) and the per-class gradients are not exactly zero, this will imply that the minority gradients are about a factor \(R\) 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]), on the other it means that the fixed points are insensitive to the data imbalance.

5 Discussion

We presented a new analysis of the learning dynamics of gradient-based algorithms for imbalanced problems. Our results clearly highlight the suboptimal behavior of Gradient Descent (GD), which becomes especially acute for high class imbalance. In particular, at the initial stages of learning, the minority classes are classified worse than random, and only in late stages, at times that increase with the data imbalance, are the minority classes learned, in agreement with the observations from [Ye et al., 2021].

In order to address the suboptimal behavior of gradient descent, we proposed a new algorithm with convergence guarantees for general smooth (non-convex) functions. We demonstrated that this algorithm provides better trade-offs to decrease the loss function of each class.

We extended our analysis to SGD, finding that in this case the suppression of the minority class is further amplified. Not only are the gradient updates favoring the majority class in terms of intensity of the signal, but the signal itself is more noisy, due to the presence of fewer minority examples in each
batch. This entails a further slow-down of the learning of the minority class, which is proportional to the class-imbalance ratio. We analyzed several ways to deal with this problem, finding that the best performances, both in terms of speed and test recall, are obtained by combining PCNSGD with oversampling.

Finally, there are several extensions of interest regarding the analysis of our algorithm, including a specialized analysis to the case of convex functions or other non-convex functions [Karimi et al., 2016], as well as the use of clipping [Zhang et al., 2019] to improve the theoretical rate of convergence.
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](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, Rafiquil Haque, Mohand-Said Hacid, and Hassan Zeinedine. 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 Pometti, 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 Xiaouo Tang. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5375–5384, 2016.

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. k-nn 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.

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

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.

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.
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*, 2019.

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.

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.

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

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

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.

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.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv:1905.11881*, 2019.

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

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.

A. Krizhevsky, V. Nair, and G. Hinton, 2013. URL https://www.cs.toronto.edu/~kriz/cifar.html
A Convergence rate GD

We start by stating a more formal version of Theorem 1, followed by its proof. For the sake of clarity and without loss of generality, we prove our results for the binary case, where the number of classes \( L = 2 \).

**Theorem 3** (Formal version of Theorem 1). Assume that each \( f^{(l)} \) is \( L_1 \)-Lipschitz and \( L_2 \)-smooth and let \( \alpha(x_t) = \angle (\nabla f^{(l)}(x_t), \nabla f^{(1-l)}(x_t)) \). If \( \| \nabla f^{(1-l)}(x_t) \| \neq 0 \) and \( \cos(\alpha(x_t)) > -\frac{\| \nabla f^{(l)}(x_t) \|}{\| \nabla f^{(1-l)}(x_t) \|} \) for all iterations \( t \in [0, T-1] \), the iterates of gradient descent with step size \( \eta_t = \frac{c}{\sqrt{T}} \) where \( c > 0 \) satisfy

\[
\min_{s \in [0, T-1]} \| \nabla f^{(l)}(x_s) \|^2 \leq \frac{1}{\sqrt{T}} \left( \frac{D_0^{(l)}}{\omega_{\min}^{(l)}} + L_2 L_1^2 c \right),
\]

for each \( l \in [L] \), where \( D_0^{(l)} = [f^{(l)}(x_0) - f^{(l)}(x^*)] \) and \( \omega_{\min}^{(l)} = \min_{t \in [0, T-1]} \eta_t (1 + \cos(\alpha(x_t)) C - L \eta_t) \).

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

Since each function \( f^{(l)} \) is \( L \)-smooth, we have

\[
f^{(0)}(x_{t+1}) \leq f^{(0)}(x_t) + \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \| x_{t+1} - x_t \|^2 \tag{7}
\]

\[
= f^{(0)}(x_t) - \eta_t \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle + \frac{L \eta^2_t}{2} \| \nabla f(x_t) \|^2
\]

\[
\leq f^{(0)}(x_t) - \eta_t \| \nabla f^{(0)}(x_t) \|^2 - \eta_t \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle + L \eta^2_t \| \nabla f^{(0)}(x_t) \|^2 + L \eta^2_t \| f^{(1)}(x_t) \|^2
\]

\[
= f^{(0)}(x_t) - \eta_t (1 - L \eta) \| \nabla f^{(0)}(x_t) \|^2 - \eta_t \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle + L \eta^2_t \| f^{(1)}(x_t) \|^2
\]

\[
= f^{(0)}(x_t) - \eta_t (1 - L \eta) \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle \| \nabla f^{(0)}(x_t) \| \| f^{(1)}(x_t) \| + L \eta^2_t \| f^{(1)}(x_t) \|^2
\]

Let \( C_t := \frac{\| \nabla f^{(l)}(x_t) \|}{\| \nabla f^{(0)}(x_t) \|} \). We get

\[
f^{(0)}(x_{t+1}) \leq f^{(0)}(x_t) - \eta_t (1 + \cos(\alpha(x_t)) C_t - L \eta_t - L \eta_t C_t^2) \| \nabla f^{(0)}(x_t) \|^2
\]

\[
\implies \eta_t (1 + \cos(\alpha(x_t)) C_t - (1 + C_t^2) L \eta_t) \| \nabla f^{(0)}(x_t) \|^2 \leq f^{(0)}(x_t) - f^{(0)}(x_{t+1}). \tag{8}
\]

Taking \( \eta_t = \frac{c}{\sqrt{T}} \),

\[
\omega_t := \eta_t (1 + \cos(\alpha(x_t)) C_t - (1 + C_t^2) L \eta_t) \| \nabla f^{(0)}(x_t) \|^2 = \frac{c}{\sqrt{T}} \left( 1 + \cos(\alpha(x_t)) C_t - (1 + C_t^2) L \frac{c}{\sqrt{T}} \right) > 0
\]

\[
\implies c < \frac{\sqrt{T}(1 + \cos(\alpha(x_t)) C_t)}{(1 + C_t^2)L} \text{ and } 1 + \cos(\alpha(x_t)) C_t > 0. \tag{9}
\]

We see that the second constraint implies the following condition on the angle \( \alpha(x_t) \):

\[
\cos(\alpha(x_t)) > -\frac{1}{C_t}. \tag{10}
\]
Let $\omega_{\min}^{(0)} = \min_{s \in [0, T \wedge 1]} \omega_s$. By summing from $t = 0$ to $T - 1$,

$$
\min_{s \in [0, T - 1]} \|\nabla f^{(0)}(x_s)\|^2 \leq \frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f^{(0)}(x_t)\|^2
$$

$$
\leq \frac{1}{\omega_{\min}^{(0)} T} f^{(0)}(x_0) - f^{(0)}(x_T) + L_2 L_1^2 \eta t
$$

$$
\leq \frac{1}{\omega_{\min}^{(0)} T} [f^{(0)}(x_0) - f^{(0)}(x^*)] + L_2 L_1^2 \eta t
$$

$$
\leq \frac{1}{\sqrt{T}} \left( \frac{f^{(0)}(x_0) - f^{(0)}(x^*)}{\omega_{\min}^{(0)}} + L_2 L_1^2 c \right).
$$

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

**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 4.** Assume that each $f^{(t)}$ is $L_1$-Lipschitz and $L_2$-smooth. There is a choice of step size that such under restricted conditions on the angle $\alpha(x_t)$, we get

$$
\inf_{t \leq T} \|\nabla f^{(0)}(x_t)\|^2 \leq \frac{f^{(0)}(x_0) - f^*}{\omega(T + 1)},
$$

where $\omega = \min_t \eta (1 + |\beta_t| C - L\eta (1 + C^2))$.

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

$$
f^{(0)}(x_{t+1}) \leq f^{(0)}(x_t) + \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2
$$

$$
= f^{(0)}(x_t) - \eta \langle \nabla f^{(0)}(x_t), \nabla f(x_t) \rangle + \frac{L\eta^2}{2} \|\nabla f(x_t)\|^2
$$

$$
\leq f^{(0)}(x_t) - \eta \|\nabla f^{(0)}(x_t)\|^2 - \eta \|\nabla f^{(0)}(x_t), \nabla f^{(1)}(x_t)\| + L\eta^2 \|\nabla f^{(0)}(x_t)\|^2 + L\eta^2 \|\nabla f^{(1)}(x_t)\|^2
$$

$$
= f^{(0)}(x_t) - \eta (1 - L\eta) \|\nabla f^{(0)}(x_t)\|^2 - \eta \|\nabla f^{(0)}(x_t), \nabla f^{(1)}(x_t)\| + L\eta^2 \|\nabla f^{(1)}(x_t)\|^2
$$

$$
= f^{(0)}(x_t) - \eta (1 - L\eta) \|\nabla f^{(0)}(x_t)\|^2 - \eta \cos(\alpha(x_t)) \|\nabla f^{(0)}(x_t)\| \|\nabla f^{(1)}(x_t)\| + L\eta^2 \|\nabla f^{(1)}(x_t)\|^2.
$$

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

**Case 1: $\beta_t < 0$**

$$
f^{(0)}(x_{t+1}) \leq f^{(0)}(x_t) - \eta (1 - L\eta) \|\nabla f^{(0)}(x_t)\|^2 - \eta |\beta_t| C \|\nabla f^{(0)}(x_t)\|^2 + L\eta^2 C^2 \|\nabla f^{(0)}(x_t)\|^2
$$

$$
= f^{(0)}(x_t) - \eta (1 - |\beta_t| C - L\eta (1 + C^2)) \|\nabla f^{(0)}(x_t)\|^2.
$$

We need

$$
(1 - |\beta_t| C - L\eta (1 + C^2)) > 0 \implies \eta < \frac{1 - |\beta_t| C}{L(1 + C^2)}.
$$

We also need $C < \frac{1}{|\beta_t|}$ for the function to decrease.
Case 2: \( \beta_t \geq 0 \)

\[
\begin{align*}
  f^{(0)}(x_{t+1}) &\leq f^{(0)}(x_t) - \eta(1 - L\eta) \left\| \nabla f^{(0)}(x_t) \right\|^2 - \eta\beta_t C \| \nabla f^{(0)}(x_t) \|^2 + L\eta^2 C^2 \left\| \nabla f^{(0)}(x_t) \right\|^2 \\
  &= f^{(0)}(x_t) - \eta(1 + \beta_t C - L\eta(1 + C^2)) \left\| \nabla f^{(0)}(x_t) \right\|^2.
\end{align*}
\]

(16)

We need

\[
(1 + \beta_t C - L\eta(1 + C^2)) > 0 \quad \implies \quad \eta < \frac{1 + \beta_t C}{L(1 + C^2)}.
\]

(17)

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

\[
\begin{align*}
  f^{(0)}(x_{t+1}) &\leq f^{(0)}(x_t) - \omega_t \left\| \nabla f^{(0)}(x_t) \right\|^2,
\end{align*}
\]

(18)

where \( \omega_t = \eta(1 + \beta_t C - L\eta(1 + C^2)) \).

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

\[
\omega \sum_{t=0}^T \left\| \nabla f^{(0)}(x_t) \right\|^2 \leq f^{(0)}(x_0) - f^{(0)}(x_{T+1}) \leq f^{(0)}(x_0) - f^*,
\]

(19)

where \( f^* \) is the loss at the global minimum.

Finally, we lower bound \( \| \nabla f^{(0)}(x_t) \| \) by its infimum,

\[
\inf_{t \leq T} \left\| \nabla f^{(0)}(x_t) \right\|^2 \leq \frac{f^{(0)}(x_0) - f^*}{\omega(T+1)}
\]

(20)

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 \( C < \frac{1}{|\beta_t|} \).

\[ \square \]

**B  Convergence rate PCNGD**

This section contains the formal version of Theorem 2 with a detailed proof.

**Theorem 5** (Formal version of Theorem 2). Assume that each \( f^{(l)} \) is \( L_1 \)-Lipschitz and \( L_2 \)-smooth. Then if \( \cos \alpha(x_t) \neq -1 \) for all iterations \( t \in [0, T-1] \), the iterates of gradient descent with step size \( \eta_t = \frac{c}{\sqrt{T}} \) where \( c > 0 \) satisfy

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

for each \( l \in [L] \), where \( D_0^{(l)} = [f^{(l)}(x_0) - f^{(l)}(x^*)] \) and \( \omega_{\min} := \min_{t \in [0, T-1]} (1 + \cos \alpha(x_t)) \).

**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

\[
\begin{align*}
  f^{(0)}(x_{t+1}) &\leq f^{(0)}(x_t) + \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle + \frac{L_2}{2} \| x_{t+1} - x_t \|^2 \\
  &\leq f^{(0)}(x_t) - \eta_t \| \nabla f^{(0)}(x_t) \| - \eta_t \langle \nabla f^{(0)}(x_t), \nabla f^{(1)}(x_t) \rangle \\
  &\quad + L_2 \eta_t^2 \left\| \nabla f^{(0)}(x_t) \right\|^2 + L_2 \eta_t^2 \left\| \nabla f^{(1)}(x_t) \right\|^2 \\
  &= f^{(0)}(x_t) - \eta_t \| \nabla f^{(0)}(x_t) \| - \eta_t \| \nabla f^{(0)}(x_t) \| \cos \alpha(x_t) + 2L_2 \eta_t^2 \\
  &= f^{(0)}(x_t) - \eta_t (1 + \cos \alpha(x_t)) \| \nabla f^{(0)}(x_t) \| + 2L_2 \eta_t^2.
\end{align*}
\]

(21)
Let $\omega_{\min} := \min_{t \in [0, T-1]} (1 + \cos \alpha(x_t))$. By rearranging the terms in the equation above, we get
\[
\|\nabla f^{(0)}(x_t)\| \leq \frac{1}{\eta_t \omega_{\min}} \left[ f^{(0)}(x_t) - f^{(0)}(x_{t+1}) \right] + \frac{2L_2 \eta_t}{\omega_{\min}}. \tag{22}
\]

Taking the minimum over $t$, we get
\[
\min_{s \in [0, T-1]} \|\nabla f^{(0)}(x_s)\| \leq \frac{1}{\sqrt{T}} \sum_{t=0}^{T-1} \|\nabla f^{(0)}(x_t)\|
\leq \frac{1}{\eta_t \omega_{\min} T} [f^{(0)}(x_0) - f^{(0)}(x_T)] + \frac{2L_2 \eta_t}{\omega_{\min}}
\leq \frac{1}{\eta_t \omega_{\min}} [f^{(0)}(x_0) - f^{(0)}(x^*)] + \frac{2L_2 \eta_t}{\omega_{\min}}
\leq \frac{1}{\omega_{\min} \sqrt{T}} \left( \frac{1}{c} [f^{(0)}(x_0) - f^{(0)}(x^*)] + 2L_2 \right). \tag{23}
\]

\[\square\]

**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 6 (R-PCNGD).** Let the probability mass function $P_R(\cdot)$ be defined as
\[P_R(t) = \text{Prob}\{ R = t \} = \frac{\omega_t}{\sum_{t=0}^{T-1} \omega_t}, \tag{24}\]
where $\omega_t := (1 + \cos \alpha(x_t))$. Then, for any step size $\eta_t = \frac{1}{\sqrt{T}}$ and $\cos \alpha(x_t) \neq -1$ for all iterations $t$, we have for $k = 1, 2$,
\[
\mathbb{E}\|\nabla f^{(k)}(x_R)\| \leq \frac{1}{\sqrt{T} \bar{\omega}} \left[ c^{-1} (f^{(k)}(x_0) - f^{(k)}(x^*)) + 2L_2c \right], \tag{25}\]
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
\[
f^{(0)}(x_{t+1}) \leq f^{(0)}(x_t) + \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle + \frac{L_2}{2} \| x_{t+1} - x_t \|^2
\leq f^{(0)}(x_t) - \eta_t \|\nabla f^{(0)}(x_t)\| - \eta_t \langle \nabla f^{(0)}(x_t), \nabla f^{(1)}(x_t) \rangle
+ L_2 \eta_t^2 \left\| \frac{\nabla f^{(0)}(x_t)}{\|\nabla f^{(0)}(x_t)\|} \right\|^2 + L_2 \eta_t^2 \left\| \frac{\nabla f^{(1)}(x_t)}{\|\nabla f^{(1)}(x_t)\|} \right\|^2
= f^{(0)}(x_t) - \eta_t \|\nabla f^{(0)}(x_t)\| - \eta_t \|\nabla f^{(0)}(x_t)\| \cos \alpha(x_t) + 2L_2 \eta_t^2
= f^{(0)}(x_t) - \eta_t (1 + \cos \alpha(x_t)) \|\nabla f^{(0)}(x_t)\| + 2L_2 \eta_t^2. \tag{26}\]

Let $\omega_t := (1 + \cos \alpha(x_t))$. By rearranging the terms in the equation above, we get
\[
\omega_t \|\nabla f^{(0)}(x_t)\| \leq \frac{1}{\eta_t} [f^{(0)}(x_t) - f^{(0)}(x_{t+1})] + 2L_2 \eta_t \tag{27}\]
By summing from $t = 0$ to $T - 1$ and dividing by $\sum_{t=0}^{T-1} \omega_t$,
\[
\frac{\sum_{t=0}^{T-1} \omega_t \| \nabla f^{(0)}(x_t) \|}{\sum_{t=0}^{T-1} \omega_t} \leq \frac{\sum_{t=0}^{T-1} \eta_t^{-1} (f^{(0)}(x_0) - f^{(0)}(x^*))}{\sum_{t=0}^{T-1} \omega_t} + \frac{2L_2 \sum_{t=0}^{T-1} \eta_t}{\sum_{t=0}^{T-1} \omega_t}, \tag{28}
\]
where $\bar{\omega}$ is the average of the sequence $\{\omega_t\}_{t=0}^{T-1}$.

Taking $\eta_t = \frac{\alpha}{\sqrt{T}}$, we obtain
\[
\sum_{t=0}^{T-1} \omega_t \| \nabla f^{(0)}(x_t) \| \leq \frac{1}{c\sqrt{T}} (f^{(0)}(x_0) - f^{(0)}(x^*)) + \frac{2L_2 c}{\sqrt{T}} \tag{29}
\]

Finally, observe that the RHS is an expectation $E \| \nabla f^{(0)}(x_R) \|$ for an appropriately chosen random variable $x_R$ according to the distribution $P_R(\cdot)$, therefore
\[
E \| \nabla f^{(0)}(x_R) \| \leq \frac{1}{\sqrt{T\bar{\omega}}} \left[ c^{-1} (f^{(0)}(x_0) - f^{(0)}(x^*)) + 2L_2 c \right] \tag{30}
\]

\[\square\]

**Alternate choice of step-size**  We also derive a convergence result for the case where the step size does not depend on the total number of iterations $T$.

**Theorem 7.** Assume that each $f^{(l)}$ for $l = 0, 1$ is $L_1$-Lipschitz and $L_2$-smooth. Then if $\cos \alpha(x_t) \neq -1$ for all iterations $t$, the iterates of gradient descent with step size $\eta_t = \frac{\| \nabla f^{(0)}(x_t) \|}{4L_1 L_2}$ for all $t$, $0 \leq t \leq T - 1$, we have

\[
\min_{s \in [0, T-1]} \| \nabla f^{(0)}(x_s) \|^2 \leq \frac{1}{\omega_{\min}^{(1)} T} \left[ f^{(0)}(x_0) - f^{(0)}(x^*) \right] \tag{31}
\]

and

\[
\min_{s \in [0, T-1]} \| \nabla f^{(1)}(x_s) \|^2 \leq \frac{1}{\omega_{\min}^{(0)} T} \left[ f^{(1)}(x_0) - f^{(1)}(x^*) \right] \tag{32}
\]

where

\[
\omega_{\min}^{(l)} = \min_{t \in [0, T-1]} \frac{\| \nabla f^{(l)}(x_t) \|}{4L_1 L_2} \left( 1 + \cos \alpha(x_t) \right) \left( 1 - \frac{\| \nabla f^{(l)}(x_t) \|}{2L_1} \right), \tag{33}
\]

for $l = 0, 1$.

**Proof.** **Convergence $f^{(0)}$**

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

\[
f^{(0)}(x_{t+1}) \leq f^{(0)}(x_t) + \langle \nabla f^{(0)}(x_t), x_{t+1} - x_t \rangle + \frac{L_2}{2} \| x_{t+1} - x_t \|^2
\]

\[
\leq f^{(0)}(x_t) - \eta_t \| \nabla f^{(0)}(x_t) \| - \eta_t \langle \nabla f^{(0)}(x_t), \frac{\nabla f^{(1)}(x_t)}{\| \nabla f^{(1)}(x_t) \|} \rangle + L_2 \eta_t^2 \| \nabla f^{(0)}(x_t) \|^2 + L_2 \eta_t^2 \| \nabla f^{(1)}(x_t) \|^2
\]

\[
= f^{(0)}(x_t) - \eta_t \| \nabla f^{(0)}(x_t) \| - \eta_t \| \nabla f^{(0)}(x_t) \| \cos \alpha(x_t) + 2L_2 \eta_t^2
\]

\[
= f^{(0)}(x_t) - \| \nabla f^{(0)}(x_t) \| \eta_t \left( 1 + \cos \alpha(x_t) - \frac{2L_2 \eta_t}{\| \nabla f^{(0)}(x_t) \|} \right). \tag{34}
\]
We have to choose an appropriate step size. We need \( \eta_t < \frac{2L_2\eta_t}{\|\nabla f(0)(x_t)\|} \cdot (1 + \cos \alpha(x_t)) > 0 \), i.e.

\[
\eta_t < \frac{\|\nabla f(0)(x_t)\| (1 + \cos \alpha(x_t))}{2L_2}.
\]

We simply take
\[
\eta_t = \frac{\|\nabla f(0)(x_t)\| \|\nabla f(1)(x_t)\| (1 + \cos \alpha(x_t))}{4L_1 L_2}.
\]

The coefficient in front of \( \|\nabla f(0)(x_t)\| \) becomes
\[
-\eta_t \left( 1 + \cos \alpha(x_t) - \frac{2L_2\eta_t}{\|\nabla f(0)(x_t)\|} \right) = -\frac{\|\nabla f(0)(x_t)\| \|\nabla f(1)(x_t)\| (1 + \cos \alpha(x_t))}{4L_1 L_2} \left( 1 + \cos \alpha(x_t) - \frac{\|\nabla f(1)(x_t)\| (1 + \cos \alpha(x_t))}{2L_1} \right)
\]
\[
\leq -\frac{\|\nabla f(0)(x_t)\| \|\nabla f(1)(x_t)\| (1 + \cos \alpha(x_t))}{4L_1 L_2} \left( 1 + \cos \alpha(x_t) \left( 1 - \frac{\|\nabla f(1)(x_t)\|}{2L_1} \right) \right).
\]

Let \( \omega_{\min}^{(1)} = \min_t \frac{\|\nabla f(1)(x_t)\| (1 + \cos \alpha(x_t))}{4L_1 L_2} \left( 1 + \cos \alpha(x_t) \left( 1 - \frac{\|\nabla f(1)(x_t)\|}{2L_1} \right) \right), \) then
\[
\omega_{\min}^{(1)} \|\nabla f(0)(x_t)\|^2 \leq f^{(0)}(x_t) - f^{(0)}(x_{t+1}).
\]

Taking the minimum over \( t \), we get
\[
\min_{s \in [0, T-1]} \frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f(0)(x_s)\|^2 \leq \frac{1}{\omega_{\min}^{(1)} T} \left[ f^{(0)}(x_0) - f^{(0)}(x_T) \right]
\]
\[
\leq \frac{1}{\omega_{\min}^{(1)} T} \left[ f^{(0)}(x_0) - f^{(0)}(x^*) \right].
\]

**Convergence \( f^{(1)} \)**

Since \( f^{(1)} \) is \( L_2 \)-smooth, we have
\[
f^{(1)}(x_{t+1}) \leq f^{(1)}(x_t) + \langle \nabla f^{(1)}(x_t), x_{t+1} - x_t \rangle + \frac{L_2}{2} \| x_{t+1} - x_t \|^2
\]
\[
\leq f^{(1)}(x_t) - \eta_t \|\nabla f^{(1)}(x_t)\| - \eta_t \langle \nabla f^{(1)}(x_t), \nabla f^{(0)}(x_t) \rangle
\]
\[
+ L_2 \eta_t^2 \left( \frac{\nabla f^{(0)}(x_t)}{\|\nabla f^{(0)}(x_t)\|} \right)^2 + L_2 \eta_t^2 \left( \frac{\nabla f^{(1)}(x_t)}{\|\nabla f^{(1)}(x_t)\|} \right)^2
\]
\[
= f^{(1)}(x_t) - \eta_t \|\nabla f^{(1)}(x_t)\| - \eta_t \|\nabla f^{(1)}(x_t)\| \cos \alpha(x_t) + 2L_2 \eta_t^2
\]
\[
= f^{(1)}(x_t) - \|\nabla f^{(1)}(x_t)\| \eta_t \left( 1 + \cos \alpha(x_t) - \frac{2L_2 \eta_t}{\|\nabla f^{(1)}(x_t)\|} \right).
\]

We have to choose an appropriate step size. We need \( \left( 1 + \cos \alpha(x_t) - \frac{2L_2 \eta_t}{\|\nabla f^{(1)}(x_t)\|} \right) > 0 \), i.e.
\[
\eta_t < \frac{\|\nabla f^{(1)}(x_t)\| (1 + \cos \alpha(x_t))}{2L_2}.
\]

We simply take
\[
\eta_t = \frac{\|\nabla f^{(0)}(x_t)\| \|\nabla f^{(1)}(x_t)\| (1 + \cos \alpha(x_t))}{4L_1 L_2}.
\]
The coefficient in front of $\|\nabla f^{(1)}(x_t)\|$ becomes
\[-\eta_t \left( 1 + \cos \alpha(x_t) - \frac{2L_2\eta_t}{\|\nabla f^{(1)}(x_t)\|} \right) = -\frac{\|\nabla f^{(0)}(x_t)\|}{4L_1L_2} \|\nabla f^{(1)}(x_t)\| (1 + \cos \alpha(x_t)) \left( 1 + \cos \alpha(x_t) - \frac{\|\nabla f^{(0)}(x_t)\|}{2L_1} \right) \leq -\frac{\|\nabla f^{(0)}(x_t)\|}{4L_1L_2} \|\nabla f^{(1)}(x_t)\| (1 + \cos \alpha(x_t)) \left( 1 - \frac{\|\nabla f^{(0)}(x_t)\|}{2L_1} \right).
\]

Let $\omega_{\min}^{(0)} = \min \frac{\|\nabla f^{(0)}(x_t)\| (1 + \cos \alpha(x_t))}{4L_1L_2} \left( 1 + \cos \alpha(x_t) \right) \left( 1 - \frac{\|\nabla f^{(0)}(x_t)\|}{2L_1} \right)$, then
\[\omega_{\min}^{(0)} \|\nabla f^{(1)}(x_t)\|^2 \leq f^{(1)}(x_t) - f^{(1)}(x_{t+1}). \quad (42)\]

Taking the minimum over $t$, we get
\[
\min_{s \in [0,T-1]} \|\nabla f^{(1)}(x_s)\|^2 \leq \frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f^{(1)}(x_t)\|^2 \leq \frac{1}{\omega_{\min}^{(0)}} \frac{T}{T} \left[ f^{(1)}(x_0) - f^{(1)}(x_T) \right] \leq \frac{1}{\omega_{\min}^{(0)}} \frac{T}{T} \left[ f^{(1)}(x_0) - f^{(1)}(x^*) \right]. \quad (43)\]

\[\square\]

C Models and data

The codes needed to reproduce the experiments presented are available in the following link: Anonymous GitHub. The README.md 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...).

C.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. A convolutional layer (as well as a pooling one) the following parameters:

- **in channels**: Number of channels in the input image
- **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 two architectures for the simulations. In particular, we propose a simple network prototype (Mod1), and a second deeper one as a prototype for more articulated models (Mod2).

- **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: Tanh
Average pooling: kernel $= 2$, stride $= 2$

Fully connected linear layer

Output: The cross entropy had been chosen as loss function for the training.

The only two hyper-parameters (HPs) to be tuned, in this case, are the learning rate (LR) and batch size ($BS$).

• **Deep-CNN** ($Mod2$): as a second 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]. 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 ($GF$). Compared to $Mod1$ we thus have in $Mod2$ two additional HPs to be fixed through the HP validation process (DO rate and GF).

The HPs 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.

**C.2 Dataset and Hyper-parameters**

The runs in this work were performed using data from the CIFAR10 dataset [Krizhevsky et al. 2013]. In order to work in a binary classification setting, we preprocessed the dataset in several ways:

• **Simple classes**: 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).

• **Super-classes**: In order to have a dataset of larger size, 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.

• **Multi-class**: All 10 classes in the CIFAR10 dataset were used. The number of images associated with each class, $N_i$, is set by the relation:

$$N_i = N_{max} \left( \frac{3}{5} \right)^i$$  \hspace{1cm} (44)

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: automobile, 2: bird, 3: cat, 4: deer, 5: dog, 6: frog, 7: horse, 8: ship, 9: truck). We called this the **Mul10** dataset.

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.

**C.3 Execution times**

Simulations were run on different servers. The code allows a 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

\footnote{the latter present only in the stochastic case}
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; in other words, the simulations run were never parallelized on clusters of GPUs. 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 $(\frac{\text{sec}}{\text{iteration}})$ units. The two architectures have different execution times.

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 $\sim 5\frac{\text{sec}}{\text{iteration}}$ for Bi7b to $\sim 30\frac{\text{sec}}{\text{iteration}}$ 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 $\sim 4\frac{\text{sec}}{\text{iteration}}$. The remaining algorithms, on the other hand, with a speed of $\sim 0.2\frac{\text{sec}}{\text{iteration}}$.

The times increase considerably by moving to the Mod2 architecture. We have in this case an execution time of $\sim 60\frac{\text{sec}}{\text{iteration}}$ for the deterministic algorithms and $\sim 1.5\frac{\text{sec}}{\text{iteration}}$ for the stochastic ones. For future use, the code records the execution times of the simulations on file.

D Additional experiments

D.1 Deterministic optimization

Here, we show more plots on the comparison between GD and PCNGD. Fig. S4 shows the loss function related to the curves shown in Fig. 1.

![Figure S4: Loss function for GD and PCNGD algorithms. Model: Mod1, dataset: Bi60. The related recall curves are shown in Fig. 1](image)

Figs. S5 and S6 show the recall and loss of the Mod1 model, respectively for the Bi7a and Bi7b datasets. The picture is similar to the one shown in Fig. 1. The reader could remark that the learning curves for the PCNGD are not strictly monotonic as one could expect from Theorem 2. The 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 learning rates for which the per-class loss is non-monotonic. This reveals a trade-off between a strictly monotonous decrease of the loss and a larger step size.

In Fig. S7 we show the recall and loss of the Mod2 model for the Bi7a dataset. Also here, the macro-averaged recall of PCNGD outperforms GD. The curves are however noisier, due to the presence of dropout in the model. In fact Theorem 1 and 2 does not apply if dropout is used.
Figure S5: Recall (top) and Loss (bottom) for GD and PCNGD algorithms. Model: Mod1, dataset: Bi7a.

In Fig. S8 we show the macro-averaged recall and loss in the multiclass case. Also here the improvement obtained by PCNGD is evident.

D.2 Stochastic optimization

Here we show additional experiments regarding stochastic optimization algorithms, as described in the main part of the paper.

In Fig. S9 shows the loss functions related to Fig. 3. Interestingly, an increase in the recall is not strictly connected with a decrease in the loss function.

In Fig. S10 we show macro averaged recalls and loss functions for stochastic algorithms, for the Mod1 model on the Bi7b dataset. We do not show the curves for PCNSGD+R because they are computationally expensive and this algorithm is not one of the algorithmic solutions that we propose. Also here, PCNSGD+O outperforms all the other solutions. The initial spike in the loss for the SGD+O algorithm can be reduced, 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. S11, where there is a significant gain with respect to SGD+O along the entire dynamics.

In Fig. S12 we show that the scenario does not change when changing architecture: PCNSGD+O is again reaching the highest test recalls, and it has a higher test recall also at short times.
In Fig. S13 we show that the situation remains unchanged also in the multiclass setting.

The effect of PCNSGD+R on the single classes  In Fig. S14 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.

As similar behavior can be seen in the full-batch dynamics. In Fig. S15 we show again that the GD dynamics exhibits a minority initial drop, which is instead not observed with the PCNGD dynamics.
Figure S7: Recall (top) and Loss (bottom) for GD and PCNGD algorithms. Model: Mod2, dataset: Bi7a.
Figure S8: Macro-averaged recall (top) and loss (bottom) for GD and PCNGD algorithms. Model: Mod1, dataset: Mul10.

Figure S9: Macro-averaged recall (top) and loss (bottom) for several stochastic algorithms. Model: Mod1, dataset: Bi7a.
Figure S10: Macro-averaged recall (top) and loss (bottom) for several stochastic algorithms. Model: Mod1, dataset: Bi7b.
Figure S11: Macro-averaged recall (top) and loss (bottom) for several stochastic algorithms. Model: Mod1, dataset: Bi60.
Figure S12: Macro-averaged recall (top) and loss (bottom) for several stochastic algorithms. Model: Mod2, dataset: Bi7a.
Figure S13: Macro-averaged recall (top) and loss (bottom) for several stochastic algorithms. Model: Mod1, dataset: Mul10.
Figure S14: Comparison SGD (top) and with PCNSGD+R (bottom) Model: Mod1, dataset: Mul10. The majority class is class 0, the minority class is class 9.
Figure S15: Comparison GD (top) and with PCNGD (bottom) Model: *Mod1*, dataset: *Mul10*. The majority class is class 0, the minority class is class 9.