Beyond Log-Concavity: Theory and Algorithm for Sum-Log-Concave Optimization

Mastane Achab *mastane.achab@tii.ae*
Technology Innovation Institute, 9639 Masdar City, Abu Dhabi, United Arab Emirates

Abstract

This paper extends the classic theory of convex optimization to the minimization of functions that are equal to the negated logarithm of what we term as a "sum-log-concave" function, i.e., a sum of log-concave functions. In particular, we show that such functions are in general not convex but still satisfy generalized convexity inequalities. These inequalities unveil the key importance of a certain vector that we call the "cross-gradient" and that is, in general, distinct from the usual gradient. Thus, we propose the Cross Gradient Descent (XGD) algorithm moving in the opposite direction of the cross-gradient and derive a convergence analysis. As an application of our sum-log-concave framework, we introduce the so-called "checkered regression" method relying on a sum-log-concave function. This classifier extends (multiclass) logistic regression to non-linearly separable problems since it is capable of tessellating the feature space by using any given number of hyperplanes, creating a checkerboard-like pattern of decision regions.

1 Introduction

In the landscape of machine learning and optimization, the quest to understand the mechanics of gradient descent algorithms, especially in the non-convex scenario, has received much attention. Traditional global convergence analysis of these algorithms has been confined mostly to convex settings (Boyd & Vandenberghe (2004),Bach (2021)). Despite recent advances, our understanding of the performance of these algorithms beyond the convex setting is rather limited, with most results being confined to local convergence analyses (Danilova et al., 2022).

Deep learning methods, such as deep neural networks, have demonstrated remarkable empirical successes in tackling non-linear and high-dimensional problems, ranging from image or speech recognition to natural language processing (LeCun et al. (2015), Goodfellow et al. (2016)). These techniques excel in creating complex decision boundaries and robust feature representations, efficiently capturing intricate patterns in the data. However, while they are empirically proficient, the theoretical underpinnings, particularly a rigorous global convergence analysis of the associated optimization methods, remain an active area of investigation. Most existing theories focus on specific architectures or rely on idealized assumptions that might not hold in practice (Jacot et al. (2018),Chizat & Bach (2018)). Therefore, while deep learning methods continue to dominate the field due to their powerful performance, the lack of comprehensive theoretical guarantees for their convergence behaviors warrants further research.

In this paper, we extend the conventional global convergence analysis of gradient descent, taking it way beyond the classical convex case. We examine functions represented as the negated logarithm of what we call a *sum-log-concave* function, that is a sum of log-concave functions. Our analysis reverts to the traditional convex case when applied to a single log-concave function. In the more general scenario, we are still able to provide an upper bound for the excess risk of order $\tilde{\mathcal{O}}(1/\sqrt{T})$, where T represents the number of steps of our proposed algorithm.

Furthermore, we introduce a new sum-log-concave model that generalizes (multiclass) logistic regression, suitable for non-linearly separable problems. For binary classification, our model is capable of tessellating the feature space with any given number of hyperplanes. The decision regions, outputting label 0 or 1, follow a checkerboard-like pattern, generalizing the half-spaces and the linear prediction rule learned by logistic regression (see e.g. Bishop (2006), Hastie et al. (2009)). For that reason, we call this new method

the *checkered regression model* and we define it in a computationally efficient way through the circular convolution of several SoftArgMax vectors.

In the binary classification case, our checkered regression model can be seen as a single hidden-layer neural network with tanh activations that are multiplied together, instead of being additively combined as is customary in classic deep learning architectures. In particular, when multiplying only two hyperbolic tangents, our model can be interpreted as a smooth XOR with continuous sigmoids in place of Boolean variables.

Key contributions Our main contributions can be summarized as follows:

1. We prove a convergence guarantee of our proposed "cross gradient descent" algorithm holding for any function equal to the negated logarithm of a sum-log-concave function.
2. We introduce a new sum-log-concave model (namely "the checkered regression") that naturally extends multiclass logistic regression beyond linearly separable problems.

Notations The Euclidean norm of any vector $v \in \mathbb{R}^p$ ($p \geq 1$) is denoted $\|v\|$ and, for any $r > 0$, let $\mathcal{B}(v, r) = \{v' \in \mathbb{R}^p : \|v' - v\| \leq r\}$ denote the Euclidean ball centered at v with radius r. Given two vectors $u = (u_0, \dots, u_{p-1}) \in \mathbb{R}^p$ and $v = (v_0, \dots, v_{p-1}) \in \mathbb{R}^p$, their circular convolution $u \circledast v$ is another p-dimensional vector with k-th entry equal to $\sum_{0 \leq i,j \leq p-1 : i+j \equiv k[p]} u_i v_j$, for any $k \in \{0, \dots, p-1\}$. For any integer $S \geq 1$, we denote by $\mathfrak{S}_S$ the set of permutations of $\{1, \dots, S\}$ and by Δ_S the probabilty simplex:

$$\Delta_S = \left\{\mu = (\mu_1, \dots, \mu_S) \in [0,1]^S : \mu_1 + \cdots + \mu_S = 1\right\} .$$

The Kullback-Leibler divergence Kullback & Leibler (1951) will be denoted by "D_{KL}" throughout the paper: for any $\mu, \nu \in \Delta_S$, $D_{\mathrm{KL}}(\mu \| \nu) = \sum_{s=1}^S \mu_s \log\left(\frac{\mu_s}{\nu_s}\right)$. Given $K \geq 2$ vectors $v^{(1)} = (v_1^{(1)}, \dots, v_{d_1}^{(1)}), \dots, v^{(K)} = (v_1^{(K)}, \dots, v_{d_K}^{(K)})$, their outer product, denoted $v^{(1)} \otimes \cdots \otimes v^{(K)}$, is the K-way tensor defined such that $[v^{(1)} \otimes \cdots \otimes v^{(K)}]_{j_1, \dots, j_K} = v_{j_1}^{(1)} \times \cdots \times v_{j_K}^{(K)}$, for all $(j_1, \dots, j_K) \in \{1, \dots, d_1\} \times \cdots \times \{1, \dots, d_K\}$. For $\upsilon = (\upsilon_1, \dots, \upsilon_m) \in \{0, \dots, c-1\}^m$ (with $c \geq 2$), we denote $|\upsilon| = \sum_{k=1}^m \upsilon_k$.

2 Motivation

2.1 Ubiquitous log-concave functions in machine learning

In preamble, let us recall that a function $p : \mathbb{R}^m \to (0, \infty)$ is *log-concave* if $\log(p)$ is concave (or equivalently, if $-\log(p)$ is convex), i.e. if for all $x, y \in \mathbb{R}^m$, for all $\lambda \in [0,1]$,

$$p(\lambda x + (1-\lambda)y) \geq p(x)^\lambda \cdot p(y)^{1-\lambda} . \tag{1}$$

Typical examples of log-concave functions that are used in machine learning include:

- The sigmoid function defined for all $x \in \mathbb{R}$ as $\sigma(x) = (1 + e^{-x})^{-1} \in (0,1)$, that is used in logistic regression.
- Given $c \geq 2$, each component $\sigma_j(z)$ of the vector-valued "SoftArgMax" function

$$\vec{\sigma} : z = (z_0, \dots, z_{c-1}) \in \mathbb{R}^c \mapsto \begin{pmatrix} \sigma_0(z) \\ \vdots \\ \sigma_{c-1}(z) \end{pmatrix} = \begin{pmatrix} \frac{e^{z_0}}{\sum_{k=0}^{c-1} e^{z_k}} \\ \vdots \\ \frac{e^{z_{c-1}}}{\sum_{k=0}^{c-1} e^{z_k}} \end{pmatrix} \in \Delta_c , \tag{2}$$

  that is used to produce a categorical probability distribution in multiclass logistic regression.
- The Gaussian bell curve function $f(x) = e^{-x^2/2}$ ($\forall x \in \mathbb{R}$) is log-concave since the quadratic loss function $-\log(f(x)) = x^2/2$ is convex.

For all three types of log-concave functions described above, taking the negative logarithm (a.k.a. "log-loss") yields the loss function of a popular machine learning method, respectively: logistic regression, multi-class logistic regression, and least squares linear regression.

Next, we provide motivating examples for the sum-log-concave assumption studied in this paper. One shall notice that this assumption is frequently encountered in the machine learning literature through the Gaussian mixture model (GMM). Indeed, the probability density of a GMM is a sum of log-concave Bell curve functions.

2.2 Motivating examples of sum-log-concave methods

Example 2.1 (SoftMin Regression)**.** Given an input vector $x \in \mathbb{R}^d$, and for each $1 \leq s \leq S$, $K_s \geq 1$ targets $y_{s,1}, \ldots, y_{s,K_s} \in \mathbb{R}$, and parameters $\omega_{s,1}, \ldots, \omega_{s,K_s} \in \mathbb{R}^d$ (all collected in "ω"), we define the "SoftMin regression" loss function as follows:

$$F(\omega) = -\log\left(\sum_{s=1}^{S} e^{-\frac{1}{2}\sum_{k=1}^{K_s}(y_{s,k}-x^\top w_{s,k})^2}\right) ,$$

which is equal to the log-loss of a sum of log-concave functions. If $S = 1$ and $K_1 = 1$, this corresponds to a least squares linear regression. In the general case, this objective smoothly mimicks the minimum of the square losses:

$$F(\omega) \lessapprox \min_{1\leq s\leq S} \frac{1}{2}\sum_{k=1}^{K_s}(y_{s,k} - x^\top w_{s,k})^2 .$$

Example 2.2 (Smooth XOR)**.** Given two Boolean variables $A, B \in \{0, 1\}$, we recall that the XOR logical gate is given by: $\mathrm{XOR}(A, B) = A(1 - B) + (1 - A)B$. By replacing these Boolean variables by continuous sigmoids, namely $A = \sigma(a), B = \sigma(b)$ with $a, b \in \mathbb{R}$ and $\sigma(x) = (1 + e^{-x})^{-1} = 1 - \sigma(-x)$, we obtain a sum-log-concave smooth XOR function:

$$\chi(a, b) = \sigma(a)\sigma(-b) + \sigma(-a)\sigma(b) = \frac{e^{-a} + e^{-b}}{1 + e^{-a} + e^{-b} + e^{-a-b}} .$$

In fact, this function naturally arises as the posterior probability of binary labelled data sampled from mixtures of distributions belonging to some exponential family. For instance in the Gaussian case, consider a random pair (X, Y) valued in $\mathbb{R}^d \times \{-1, +1\}$ such that $\mathbb{P}(Y = 1) = \alpha$ and

$$\begin{cases} [X|Y = +1] \sim \frac{1}{2}\mathcal{N}(\mu_+, \Sigma) + \frac{1}{2}\mathcal{N}(\nu_+, \Sigma) \\ [X|Y = -1] \sim \frac{1}{2}\mathcal{N}(\mu_-, \Sigma) + \frac{1}{2}\mathcal{N}(\nu_-, \Sigma) , \end{cases}$$

such that $\mu_+ + \nu_+ = \mu_- + \nu_-$ (which includes the classic "XOR Gaussian mixture model" where $\mu_+ = -\nu_+$ and $\mu_- = -\nu_-$). Then, a direct application of Bayes' rule shows that the posterior class probability writes as a smooth XOR with affine arguments with respect to any data point x:

$$\mathbb{P}(Y = 1|X = x) = \chi(w_1^\top x + b_1, w_2^\top x + b_2) ,$$

where $w_1, w_2 \in \mathbb{R}^d$ and $b_1, b_2 \in \mathbb{R}$ are constants depending on α and the Gaussian mean parameters. This smooth XOR model will be further generalized in section 5.

As seen in Example 2.2, the smooth XOR function χ is a sum of two log-concave functions, which makes it fundamentally distinct from the well-studied class of log-concave functions. For instance, any log-concave function p (see Eq. 1) is also quasi-concave (i.e. for all x, y, for all $\lambda \in [0, 1]$, $p(\lambda x + (1 - \lambda y)) \geq \min(p(x), p(y))$). Nevertheless, the smooth XOR function is not quasi-concave: indeed,

$$\chi\left(\frac{1+0}{2}, \frac{0+1}{2}\right) \approx 0.47 < \chi(1, 0) = \chi(0, 1) = 0.5 .$$

2.3 Learning rock-paper-scissors

We now illustrate on a simple example that sum-log-concave models allows to learn richer patterns than standard log-concave methods. More specifically, we consider the smooth XOR classifier that we define through the smooth XOR function χ introduced in Example 2.2 together with parameters vectors $\omega_1, \omega_2 \in \mathbb{R}^d$

(with no bias parameters). Given an input vector $x \in \mathbb{R}^d$, our proposed classifier outputs the probability: $\chi(\omega_1^\top x, \omega_2^\top x) \in (0,1)$. We seek to minimize (with respect to ω_1, ω_2) the corresponding log-loss over the dataset consisting of the three possible duels in the rock-paper-scissors game. The rules of this game form a cycle as illustrated in Figure 1. Formally, rock-paper-scissors can be represented as an intransitive distribution P over $\mathfrak{S}_3$ with pairwise marginals $p_{ab} = \mathbb{P}_{\Sigma\sim P}(\Sigma(a) < \Sigma(b))$ (for $a \neq b \in \{\text{rock}, \text{paper}, \text{scissors}\}$) that violate the standard stochastic transitivity assumption: $p_{ab} \geq \frac{1}{2}$ and $p_{bc} \geq \frac{1}{2} \not\Longrightarrow p_{ac} \geq \frac{1}{2}$, as shown in Figure 2. Our smooth XOR method is compared against the classic Bradley-Terry model (Bradley & Terry (1952)) ; both methods are learned via gradient descent[1]. Figure 3 shows that the smooth XOR approach successfully learns the cyclic pairwise probabilities with a log-loss tending towards zero, while the Bradley-Terry method cannot do better than predicting equal probability $\frac{1}{2}$ for each duel and has a log-loss that converges to $\log(2)$.

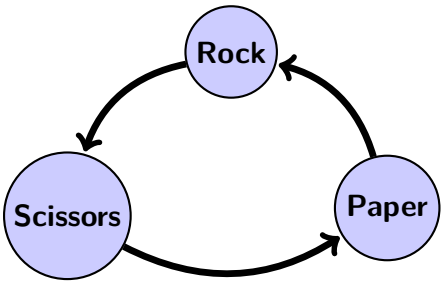

Figure 1: Cyclic representation of the rules of rock-paper-scissors. In every possible duel, the arrow starts from the winner and points towards the loser.

	Rock	Paper	Scissors
Rock	\	$p_{rp} = 0$	$p_{rs} = 1$
Paper	$p_{pr} = 1$	\	$p_{ps} = 0$
Scissors	$p_{sr} = 0$	$p_{sp} = 1$	\

Figure 2: Pairwise marginal probabilities for rock-paper-scissors game.

Although our proposed smooth XOR classifier allows to learn cyclic rankings as shown in Figure 3, we stress the importance of the utilized random initialization of the weights. Indeed, since the log-loss of this classifier has a saddle point at zero (i.e., $-\nabla \log(\chi)(0,0) = 0$), it cannot be learned successfully by gradient descent if we initialize its weights with all-zeros vectors. On the other hand, since the Bradley-Terry model has a convex log-loss, it can be learned by gradient descent independently of the initial weights (even though its representational power remains limited to acyclic rankings). This observation raises an algorithmic challenge, namely that of designing appropriate optimization algorithms for sum-log-concave methods that can escape from saddle-points and that can be learned for any initialization. Before tackling this challenge in section 4, we first introduce in the next section the class of functions that we propose to minimize and discuss some of their properties.

3 Main assumption

In this section, we introduce our proposed generalization of the notion of log-concavity, termed "sum-log-concavity", that is obtained by summing several log-concave functions.

3.1 Sum-log-concavity

We now introduce the main assumption about the class of functions that we seek to minimize in the present study, namely any function that is equal to the log-loss of a sum-log-concave model.

Assumption 1 (Negated Log of Sum-Log-Concave)**.** There exists $S \geq 1$ such that the function $F : \mathbb{R}^m \mapsto \mathbb{R}$ is given by:

$$\forall \theta \in \mathbb{R}^m \ , \quad F(\theta) = -\log\left(\sum_{s=1}^{S} p_s(\theta)\right) \ ,$$

[1]The code can be found here: https://gist.github.com/mastane/rock-paper-scissors.ipynb

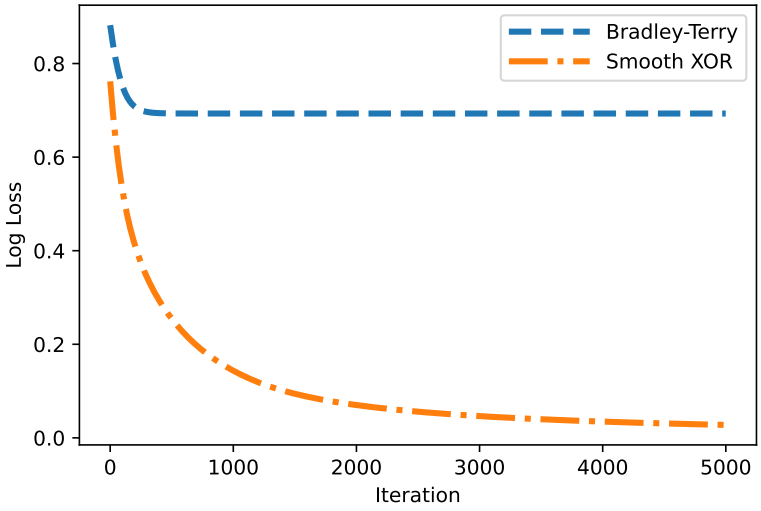

Figure 3: Learning dynamics of the Bradley-Terry model (only allowing stochastically transitive ranking distributions) versus our proposed sum-log-concave smooth XOR method at learning rock-paper-scissors rules. Each of the three items (rock, paper, scissors) is represented by a one-hot encoders of size 3. The dataset is composed of $\binom{3}{2} = 3$ pairwise comparisons corresponding to all possible duels. Both models are learned via gradient descent over 5000 epochs with learning rate set to 0.01 for the log-loss. The curves are averaged over 100 seeds with every parameter independently initialized from the standard normal distribution.

where each $p_s : \mathbb{R}^m \mapsto (0, \infty)$ is a differentiable log-concave function. Given $m \geq 1$, we denote by $\mathfrak{X}_m$ the family of such functions F.

We point out that most of the notions and results presented in this paper for functions satisfying Assumption 1 naturally extend to functions belonging to the conical hull of $\mathfrak{X}_m$. For notational convenience and clarity of exposition, we restrict our attention to $\mathfrak{X}_m$.

3.2 Closure properties

As for the class of convex functions, the set $\mathfrak{X}_m$ (resp. $\mathfrak{X} := \bigcup_{m=1}^{\infty} \mathfrak{X}_m$) is closed under summation (resp. affine reparameterization). These closure properties are highly desirable in machine learning, where one typically aims at minimizing an objective function that writes as a sum of functions: $\mathcal{L}(A, b) = \sum_{i=1}^{n} f_i(Az_i + b)$ with n the size of the dataset $(z_1, \ldots, z_n)$, and where the argument of each f_i is an affine transformation of the input z_i.

Proposition 3.1 (Summation/Affine Closure)**.** *The Assumption 1 is closed under summation and affine reparameterization.*

(i) Summation. If $F_1, F_2 \in \mathfrak{X}_m$, then $F_1 + F_2 \in \mathfrak{X}_m$.

(ii) Tensor representation. Let $K \geq 1$ and, for each $k \in \{1, \ldots, K\}$, $F_k = -\log(\sum_{s=1}^{S_K} p_s^{(k)}) \in \mathfrak{X}_m$ with $S_k \geq 1$. Then,

$$F_1 + \cdots + F_K = -\log \left(\sum_{1 \leq s_1 \leq S_1, \ldots, 1 \leq s_K \leq S_K} \tilde{p}_{s_1, \ldots, s_K} \right) \quad \text{with} \quad \tilde{p}_{s_1, \ldots, s_K} = p_{s_1}^{(1)} \times \cdots \times p_{s_K}^{(K)} \ .$$

(iii) Affine reparameterization. If $F \in \mathfrak{X}_m$, $A \in \mathbb{R}^{m \times p}, b \in \mathbb{R}^m$, then $\tilde{F} : z \in \mathbb{R}^p \mapsto F(Az + b)$ is in $\mathfrak{X}_p$.

Proof. **(i) Summation.** The summation closedness property of $\mathfrak{X}_m$ is a simple consequence of the fact that the product of two log-concave functions is still log-concave. Indeed, if $F_1 = -\log(\sum_{s=1}^{S_1} p_s)$ and $F_2 = -\log(\sum_{s=1}^{S_2} q_s)$, then $F_1 + F_2 = -\log(\sum_{s=1}^{S_1}\sum_{s'=1}^{S_2} p_s q_{s'})$ involves a sum of $S_1 \times S_2$ log-concave functions.

(ii) Tensor representation. We have:

$$F_1 + \cdots + F_K = -\log\left(\prod_{k=1}^{K}\left(\sum_{s=1}^{S_k} p_s^{(k)}\right)\right) = -\log\left(\sum_{1\le s_1\le S_1,\ldots,1\le s_K\le S_K} p_{s_1}^{(1)} \times \cdots \times p_{s_K}^{(K)}\right) .$$

(iii) Affine reparameterization. Follows from the invariance of the notion of concavity under affine reparameterization: indeed, the concavity of $\log(p_s)$ implies that $z \in \mathbb{R}^p \mapsto \log p_s(Az+b)$ is concave. Hence, $z \in \mathbb{R}^p \mapsto p_s(Az+b)$ is log-concave and thus $\tilde{F} \in \mathfrak{X}_p$.

□

The summation closure property Proposition 3.1-(i) generalizes the fact that a sum of convex functions is still convex. For instance, this is not true for the class of quasi-convex functions that is studied in Hazan et al. (2015).

4 The cross gradient descent algorithm

In this section, we introduces a novel algorithm tailored to sum-log-concave optimization. We start by extending the standard notion of convexity to our sum-log-concave framework. In particular, we will see that a specific vector, that we call "cross gradient", plays a role similar to the classic gradient in convex optimization. This will motivate our new "Cross Gradient Descent" algorithm (shortened "XGD") for which we provide a convergence analysis.

4.1 Cross gradients

Let us introduce the notion of cross gradient that will appear (in place of the standard gradient) in our generalized convexity inequalities.

Definition 1 (Cross gradient)**.** Let $F : \mathbb{R}^m \mapsto \mathbb{R}$ be a function satisfying Assumption 1.

(i) For any $\theta, \eta \in \mathbb{R}^m$, we define the "cross-gradient of F at θ seen from η" as follows:

$$\nabla^\eta F(\theta) = -\sum_{s=1}^{S} \frac{p_s(\eta)}{\sum_{s'} p_{s'}(\eta)} \frac{\nabla p_s(\theta)}{p_s(\theta)} .$$

(ii) Similarly for any probability law $\mu = (\mu_1, \ldots, \mu_S) \in \Delta_S$, we overload the cross gradient notation by defining:

$$\nabla^\mu F(\theta) = -\sum_{s=1}^{S} \mu_s \frac{\nabla p_s(\theta)}{p_s(\theta)} .$$

In particular for $\eta = \theta$, the cross gradient is simply equal to the usual gradient: $\nabla^\theta F(\theta) = \nabla F(\theta)$. In the "trivial" case $S = 1$, the function F is convex and its cross gradient is always equal to the classic gradient: $\nabla^\eta F(\theta) = \nabla F(\theta)$ for all $\theta, \eta \in \mathbb{R}^m$. In the general case, notice that $\nabla^\eta F : \theta \mapsto \nabla^\eta F(\theta)$ can be viewed as a vector field parameterized by η. We call this new operation "cross gradient" because it blends the influence of both points θ and η, and, as will be shown in the proof of Lemma 1, it naturally derives from a cross entropy term. Furthermore, the cross gradient corresponds to an average of the gradients of the "partial losses" $\ell_s(\theta) = -\log(p_s(\theta))$: indeed, $\nabla \ell_s(\theta) = -\frac{\nabla p_s(\theta)}{p_s(\theta)}$. The cross gradient operator verifies an important linearity property as explained below. The proof is also provided here as it involves insightful manipulation of functions belonging to the class $\mathfrak{X}_m$.

Proposition 4.1 (Linearity)**.** *(i) Given some reference point $\eta \in \mathbb{R}^m$, the cross gradient operator ∇^η is linear in the sense that if $F, G \in \mathfrak{X}_m$, then $\nabla^\eta(F+G) = \nabla^\eta F + \nabla^\eta G$.*

(ii) Let $K \geq 1$ and, for each $k \in \{1, \dots, K\}$, $F_k = -\log(\sum_{s=1}^{S_K} p_s^{(k)}) \in \mathfrak{X}_m$ with $S_k \geq 1$, and $\mu^{(k)} \in \Delta_{S_k}$. Then,

$$\nabla^{\mu^{(1)} \otimes \cdots \otimes \mu^{(K)}}(F_1 + \cdots + F_K) = \nabla^{\mu^{(1)}} F_1 + \cdots + \nabla^{\mu^{(K)}} F_K \ ,$$

where we use the tensor representation from Proposition 3.1-(ii) for the sum $\sum_k F_k$, and the outer product $\mu^{(1)} \otimes \cdots \otimes \mu^{(K)}$ can be interpreted as an element of the probability simplex $\Delta_{S_1 \times \cdots \times S_K}$.

Proof. **(i) Given η.** As $F, G \in \mathfrak{X}_m$, they can be written as follows:

$$\begin{cases} F(\theta) = -\log\left(\sum_{s=1}^{S} p_s(\theta)\right) \\ G(\theta) = -\log\left(\sum_{r=1}^{R} q_r(\theta)\right) \end{cases} ,$$

where p_s, q_r are log-concave functions. Then, the sum also belongs to $\mathfrak{X}_m$ and involves a sum of $S \times R$ log-concave functions:

$$F(\theta) + G(\theta) = -\log\left(\left(\sum_{s=1}^{S} p_s(\theta)\right)\left(\sum_{r=1}^{R} q_r(\theta)\right)\right) = -\log\left(\sum_{s=1}^{S}\sum_{r=1}^{R} p_s(\theta) q_r(\theta)\right) \ .$$

Finally, we deduce that the cross gradient of $F+G$ is equal to

$$\begin{aligned} \nabla^\eta(F+G)(\theta) &= -\sum_{s=1}^{S}\sum_{r=1}^{R} \frac{p_s(\eta) q_r(\eta)}{\left(\sum_{s'} p_{s'}(\eta)\right)\left(\sum_{r'} q_{r'}(\eta)\right)} \left[\frac{\nabla p_s(\theta)}{p_s(\theta)} + \frac{\nabla q_r(\theta)}{q_r(\theta)}\right] \\ &= -\underbrace{\sum_{r=1}^{R} \frac{q_r(\eta)}{\sum_{r'} q_{r'}(\eta)}}_{1} \sum_{s=1}^{S} \frac{p_s(\eta)}{\sum_{s'} p_{s'}(\eta)} \frac{\nabla p_s(\theta)}{p_s(\theta)} - \underbrace{\sum_{s=1}^{S} \frac{p_s(\eta)}{\sum_{s'} p_{s'}(\eta)}}_{1} \sum_{r=1}^{R} \frac{q_r(\eta)}{\sum_{r'} q_{r'}(\eta)} \frac{\nabla q_r(\theta)}{q_r(\theta)} = \nabla^\eta F(\theta) + \nabla^\eta G(\theta) \ , \end{aligned}$$

where we have used the fact that $\nabla \log(p_s q_r) = \nabla \log(p_s) + \nabla \log(q_r) = \frac{\nabla p_s}{p_s} + \frac{\nabla q_r}{q_r}$.

(ii) Given μ's. By the tensor representation from Proposition 3.1-(ii),

$$F_1 + \cdots + F_K = -\log\left(\sum_{1 \leq s_1 \leq S_1, \dots, 1 \leq s_K \leq S_K} \tilde{p}_{s_1, \dots, s_K}\right) \quad \text{with} \quad \tilde{p}_{s_1, \dots, s_K} = p_{s_1}^{(1)} \times \cdots \times p_{s_K}^{(K)} \ .$$

Hence,

$$\begin{aligned} \nabla^{\mu^{(1)} \otimes \cdots \otimes \mu^{(K)}}(F_1 + \cdots + F_K)(\theta) &= -\sum_{1 \leq s_1 \leq S_1, \dots, 1 \leq s_K \leq S_K} \mu_{s_1}^{(1)} \times \cdots \times \mu_{s_K}^{(K)} \left[\frac{\nabla p_{s_1}^{(1)}(\theta)}{p_{s_1}^{(1)}(\theta)} + \cdots + \frac{\nabla p_{s_K}^{(K)}(\theta)}{p_{s_K}^{(K)}(\theta)}\right] \\ &= -\sum_{k=1}^{K} \prod_{j \neq k} \underbrace{\sum_{1 \leq s_j \leq S_j} \mu_{s_j}^{(j)}}_{1} \sum_{1 \leq s_k \leq S_k} \mu_{s_k}^{(k)} \frac{\nabla p_{s_k}^{(k)}(\theta)}{p_{s_k}^{(k)}(\theta)} = \nabla^{\mu^{(1)}} F_1(\theta) + \cdots + \nabla^{\mu^{(K)}} F_K(\theta) \ , \end{aligned}$$

where we used the fact that $\frac{\nabla \tilde{p}_{s_1, \dots, s_K}}{\tilde{p}_{s_1, \dots, s_K}} = \frac{\nabla p_{s_1}^{(1)}}{p_{s_1}^{(1)}} + \cdots + \frac{\nabla p_{s_K}^{(K)}}{p_{s_K}^{(K)}}$.

□

Cross Gradient Descent (XGD) The new notion of cross gradient that we have introduced in Definition 1 motivates our proposed XGD algorithm that we now describe formally. Given a function F satisfying Assumption 1, a hyperparameter probability distribution $\mu \in \Delta_S$ (from which all cross gradients will be "seen"), and an initial point θ_0, the XGD iterates are defined as follows: $\forall t \geq 1$,

$$\boxed{\theta_t \leftarrow \theta_{t-1} - \gamma_t \nabla^\mu F(\theta_{t-1}) = \theta_{t-1} + \gamma_t \sum_{s=1}^{S} \mu_s \frac{\nabla p_s(\theta_{t-1})}{p_s(\theta_{t-1})}} \qquad \text{(XGD update)}$$

4.2 Cross convexity

We are now ready to state one of our main contribution, namely a generalization of the notion of convexity to the class of functions given by the log-loss of any sum-log-concave function.

Lemma 1 (Cross convexity)**.** Consider a function $F : \mathbb{R}^m \mapsto \mathbb{R}$ satisfying Assumption 1.

(i) For all $\theta, \eta \in \mathbb{R}^m$,

$$F(\eta) - F(\theta) \geq \nabla^\eta F(\theta)^\top (\eta - \theta) + D_{\mathrm{KL}} \left(\left[\frac{p_s(\eta)}{\sum_{s'} p_{s'}(\eta)} \right]_s \middle\| \left[\frac{p_s(\theta)}{\sum_{s'} p_{s'}(\theta)} \right]_s \right) .$$

(ii) More generally, for any distribution $\mu \in \Delta_S$: $\forall \theta, \eta \in \mathbb{R}^m$,

$$F(\eta) - F(\theta) \geq \nabla^\mu F(\theta)^\top (\eta - \theta) + D_{\mathrm{KL}} \left(\mu \middle\| \left[\frac{p_s(\theta)}{\sum_{s'} p_{s'}(\theta)} \right]_s \right) - D_{\mathrm{KL}} \left(\mu \middle\| \left[\frac{p_s(\eta)}{\sum_{s'} p_{s'}(\eta)} \right]_s \right) .$$

Proof. Let us lower bound the following quantity:

$$\begin{aligned} &F(\eta) - F(\theta) - D_{\mathrm{KL}} \left(\mu \middle\| \left[\frac{p_s(\theta)}{\sum_{s'} p_{s'}(\theta)} \right]_s \right) + D_{\mathrm{KL}} \left(\mu \middle\| \left[\frac{p_s(\eta)}{\sum_{s'} p_{s'}(\eta)} \right]_s \right) \\ &\qquad = -\log \left(\frac{\sum_{s=1}^S p_s(\eta)}{\sum_{s=1}^S p_s(\theta)} \right) - \sum_{s=1}^{S} \mu_s \log \left(\frac{p_s(\eta) \Big/ \sum_{s'} p_{s'}(\eta)}{p_s(\theta) \Big/ \sum_{s'} p_{s'}(\theta)} \right) = -\sum_{s=1}^{S} \mu_s \log \left(\frac{p_s(\eta)}{p_s(\theta)} \right) . \end{aligned} \tag{3}$$

Then, by applying S convexity inequalities,

$$-\sum_{s=1}^{S} \mu_s \log \left(\frac{p_s(\eta)}{p_s(\theta)} \right) \geq -\sum_{s=1}^{S} \mu_s \frac{\nabla p_s(\theta)^\top (\eta - \theta)}{p_s(\theta)} = \nabla^\mu F(\theta)^\top (\eta - \theta) . \tag{4}$$

□

In particular in the case $S = 1$, Lemma 1 reduces to a classic convexity inequality since the KL terms are equal to zero and $\nabla^\mu F(\theta) = \nabla F(\theta)$. In the general case $S \geq 2$, we say that the function F is *cross convex* since it satisfies a variant of convexity involving the cross gradient (instead of the gradient in standard convexity) and a "penalization" equal to the difference of two KL divergences. In fact, this "penalty" term may be either positive or negative. In particular for the first inequality (i), the positive sign in front of the KL strengthens the linear lower bound instead of relaxing it: this will allow us to prove the convergence of our new algorithm leveraging the cross convexity of F.

4.3 Convergence analysis

Given $\mu = (\mu_1, \ldots, \mu_S) \in \Delta_S$, let us define

$$\mathcal{E}(\mu) = \{\eta \quad \text{s.t.} \quad \nabla^\eta F = \nabla^\mu F\} .$$

This subset of $\mathbb{R}^m$ contains all points η that share the same cross gradient vector field as the one produced by μ, i.e. for all $\eta \in \mathcal{E}(\mu)$, $\nabla^\eta F = \nabla^\mu F$. In particular, if the probability law generated by η is equal to μ, then η belongs to $\mathcal{E}(\mu)$:

$$\left\{\eta \quad \text{s.t.} \quad \forall s,\ \frac{p_s(\eta)}{\sum_{s'} p_{s'}(\eta)} = \mu_s\right\} \subseteq \mathcal{E}(\mu)\ .$$

Theorem 4.2 (Convergence of XGD)**.** *Let $\mu \in \Delta_S$ ($S \geq 1$). Consider a function $F : \mathbb{R}^m \mapsto \mathbb{R}$ satisfying Assumption 1, such that every partial loss function $\ell_s = -\log p_s$ is B-Lipschitz continuous. Then, for $T \geq 1$ iterations of XGD given by Eq. (XGD update) with learning rates $\gamma_t > 0$, it holds:*

$$\forall \eta \in \mathcal{E}(\mu)\ , \qquad \frac{1}{\sum_{t=1}^T \gamma_t} \sum_{t=1}^T \gamma_t \left(F(\theta_{t-1}) - F(\eta)\right) \leq \frac{\|\theta_0 - \eta\|^2}{2\sum_{t=1}^T \gamma_t} + B^2 \frac{\sum_{t=1}^T \gamma_t^2}{2\sum_{t=1}^T \gamma_t}\ .$$

Proof. Let us consider $\eta \in \mathcal{E}(\mu)$. For $t \geq 1$, we have:

$$\|\theta_t - \eta\|^2 = \|\theta_{t-1} - \gamma_t \nabla^\mu F(\theta_{t-1}) - \eta\|^2 = \|\theta_{t-1} - \eta\|^2 - 2\gamma_t \langle \nabla^\mu F(\theta_{t-1}), \theta_{t-1} - \eta\rangle + \gamma_t^2 \|\nabla^\mu F(\theta_{t-1})\|^2\ .$$

Then, the cross convexity inequality from Lemma 1-(i) tells us that:

$$F(\eta) - F(\theta_{t-1}) \geq \nabla^\mu F(\theta_{t-1})^\top (\eta - \theta_{t-1}) + D_{\mathrm{KL}}\left(\left[\frac{p_s(\eta)}{\sum_{s'} p_{s'}(\eta)}\right]_s \middle\| \left[\frac{p_s(\theta_{t-1})}{\sum_{s'} p_{s'}(\theta_{t-1})}\right]_s\right)\ ,$$

which implies that

$$F(\theta_{t-1}) - F(\eta) \leq \nabla^\mu F(\theta_{t-1})^\top (\theta_{t-1} - \eta).$$

By using the boundedness of the gradient of each ℓ_s, we deduce that

$$\gamma_t \left(F(\theta_{t-1}) - F(\eta)\right) \leq \frac{1}{2}\left(\|\theta_{t-1} - \eta\|^2 - \|\theta_t - \eta\|^2\right) + \frac{1}{2}\gamma_t^2 B^2\ .$$

Then, by summing these inequalities, we obtain:

$$\frac{1}{\sum_{t=1}^T \gamma_t} \sum_{t=1}^T \gamma_t \left(F(\theta_{t-1}) - F(\eta)\right) \leq \frac{\|\theta_0 - \eta\|^2}{2\sum_{t=1}^T \gamma_t} + B^2 \frac{\sum_{t=1}^T \gamma_t^2}{2\sum_{t=1}^T \gamma_t}\ .$$

□

Remark 4.1. In the case $S = 1$, the function F is convex and XGD coincides with gradient descent (GD). Theorem 4.2 then corresponds to the classic analysis of GD for a convex function, and the upper bound is of order $\mathcal{O}(\log T/\sqrt{T})$ for decaying learning rates γ_t of order $1/\sqrt{t}$ (see e.g. Bach (2021)). In our more general setting, the same guarantee can be obtained over any bounded subset $\mathcal{E}(\mu) \cap \mathcal{B}(\theta_0, D)$ by choosing $\gamma_t = \frac{D}{B\sqrt{t}}$.

5 Application: the checkered regression classifier

This section formally introduces the checkered regression method along with a few elementary properties. As will be seen, this sum-log-concave model allows to generalize multi-class logistic regression to non-linearly separable problems. We start by introducing the binary version before moving to the multiclass setting.

5.1 Definition and immediate properties

Let $m \geq 1$. We start with the definition below.

Definition 5.1 (Checkoid function)**.** The checkoid function Ξ_m is defined for all $z = (z_1, \ldots, z_m) \in \mathbb{R}^m$ by[2]

$$\Xi_m(z) = \frac{1}{2}\left(1 + \prod_{k=1}^m \tanh\left(\frac{z_k}{2}\right)\right)\ .$$

[2]We recall that the hyperbolic tangent function is defined on $\mathbb{R}$ as $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x}) \in (-1, 1)$.

Obviously, the checkoid function is permutation-invariant: for any permutation τ of $\{1,\dots,m\}$, $\Xi_m(z_{\tau(1)},\dots,z_{\tau(m)}) = \Xi_m(z_1,\dots,z_m)$. We now describe an important "anti-symmetry" property of our model that characterizes its checkered structure.

Property 5.1 (Anti-symmetry)**.** *Given $z = (z_1,\dots,z_m) \in \mathbb{R}^m$, let $k \in \{1,\dots,m\}$ and $z' = (z'_1,\dots,z'_m)$ with $z'_k = -z_k$ and $z'_j = z_j$ for $j \neq k$. Then,*

$$\Xi_m(z') = 1 - \Xi_m(z).$$

Proof. (i) is trivial while (ii) follows from the oddness of the hyperbolic tangent. □

Property 5.1 generalizes the standard identity $\sigma(-x) = 1 - \sigma(x)$ with $\sigma(x) = 1/(1+e^{-x})$ the sigmoid function. Indeed in the case $m = 1$, the checkoid coincides with the sigmoid.

Example 5.2 (Case $m = 1$: Sigmoid)**.** For $m = 1$, $\Xi_1(z) = (1/2)(1 + \tanh(z/2)) = \sigma(z)$. In this case, the checkered regression (defined from the checkoid equipped with the log-loss) boils down to a logistic regression.

Example 5.3 (Case $m = 2$: Smooth XOR)**.** If $m = 2$, one can easily show that the checkoid function is equal to

$$\Xi_2(z_1,z_2) = \frac{1}{2}\left(1 + \tanh\left(\frac{z_1}{2}\right)\tanh\left(\frac{z_2}{2}\right)\right) = \frac{\sigma(z_1)\sigma(z_2)}{\sigma(z_1+z_2)} \ .$$

Furthermore, Ξ_2 can be interpreted as a smooth XOR gate[3], where the Boolean variables are replaced by continuous sigmoids. Indeed,

$$\Xi_2(z_1,z_2) = \sigma(z_1)\sigma(z_2) + (1-\sigma(z_1))(1-\sigma(z_2)) = \mathrm{XOR}(\sigma(z_1), 1-\sigma(z_2)) \ ,$$

where we extended the XOR formula to continuous inputs in the interval $(0,1)$.

Figure 4 illustrates the checkered pattern produced by the bivariate checkoid function ($m = 2$). Indeed, the decision regions are given by the checkerboard formed by the two coordinate axes:

$$\mathrm{Sign}\left(\Xi_2(z_1,z_2) - \frac{1}{2}\right) = \mathrm{Sign}(z_1 \cdot z_2) \ . \tag{5}$$

Moreover, $\Xi_2(z_1,z_2)$ is equal to $1/2$ if and only if $z_1 = 0$ or $z_2 = 0$.

The checkered regression Equipped with the checkoid function introduced earlier, we can now define a new non-linear model for binary classification.

Definition 5.4 (Checkered regression)**.** Let $\mathcal{X} \subseteq \mathbb{R}^d$ ($d \geq 1$). The checkered regression model with parameters $\omega = (\omega_1,\dots,\omega_m) \in \mathbb{R}^{dm}$ is given by the posterior probabilities

$$p_\omega(0|x) = 1 - p_\omega(1|x) = \Xi_m(\omega_1^\intercal x,\dots,\omega_m^\intercal x) \quad \text{for all } x \in \mathcal{X} \ .$$

In particular, the "anti-symmetry" Property 5.1 implies that the decision boundary of our model showcases a checkered structure as explained in the next proposition.

Proposition 5.2 (Hamming distance parity)**.** *Consider a checkered regression (CR) model with parameters $\omega = (\omega_1,\dots,\omega_m)$. Let x and x' be two points in $\mathbb{R}^d$ both outside the m hyperplanes of the CR model, i.e. such that $\omega_k^\intercal x \neq 0$ and $\omega_k^\intercal x' \neq 0$ for all $1 \leq k \leq m$. Then, the two following conditions are equivalent.*

(i) The CR model predicts that x and x' share the same label:

$$sign\left(p_\omega(0|x) - \frac{1}{2}\right) = sign\left(p_\omega(0|x') - \frac{1}{2}\right) \ .$$

(ii) The Hamming distance $\sum_{k=1}^m \mathbb{I}\{sign(\omega_k^\intercal x) \neq sign(\omega_k^\intercal x')\}$ is even.

[3]For Boolean variables $A, B \in \{0,1\}$, $\mathrm{XOR}(A,B) = A(1-B) + (1-A)B$.

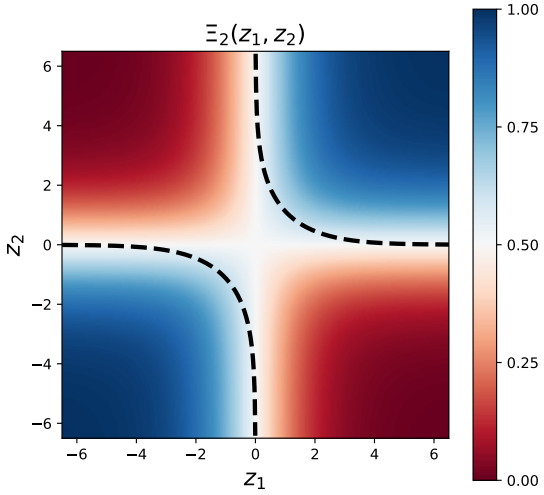

Figure 4: Heatmap of the bivariate checkoid function $\Xi_2(z_1, z_2)$ taking values in the interval $(0, 1)$ and equal to $\frac{1}{2}$ over the union of the coordinate axes. The dotted lines correspond to the frontier "$\det(H(z_1, z_2)) = 0$" (with H the Hessian matrix of $-\log \Xi_2$) that forms a quartic algebraic curve in $X = e^{z_1}$, $Y = e^{z_2}$, defined by the roots of the bivariate polynomial $P(X, Y) = X^2Y^2 - X^2Y - XY^2 - 2XY - X - Y + 1 = X^2Y^2P(1/X, 1/Y)$. In fact, this loss function is convex in the top-right (resp. bottom-left) corner delimited by the top-right (resp. bottom-left) dotted line.

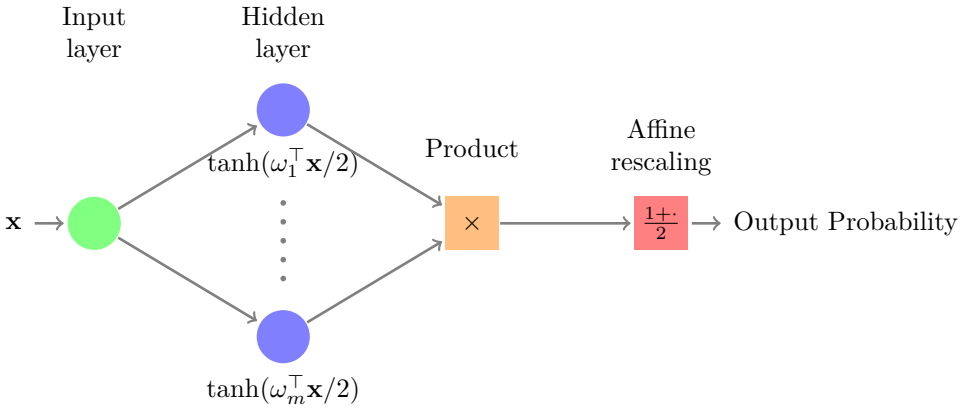

Figure 5: Diagram of the checkered regression neural network architecture for binary classification. It can be seen as a 1-hidden layer neural network with *tanh* activations that are multiplied together. Lastly, the product is rescaled to the interval $(0, 1)$. In particular, the case $m = 1$ simply outputs a sigmoid.

Proof. By successive applications of Property 5.1. □

Proposition 5.2 shows that the decision regions of a CR model form a hyperplane tessellation of the feature space $\mathcal{X}$, where the tiles are binary labeled in a checkered fashion.

Remark 5.5 (DC loss)**.** We point out that the log-loss of our model is not convex. In fact, it is equal to the difference of two convex functions (a.k.a. "DC"). In particular for $m = 2$,

$$-\log(\Xi_2(z_1, z_2)) = \underbrace{[-\log(\sigma(z_1)\sigma(z_2))]}_{\text{convex}} - \underbrace{[-\log(\sigma(z_1 + z_2))]}_{\text{convex}}. \tag{6}$$

For general $m \geq 2$ (and also in the multiclass case defined later), the log-loss of the checkered regression model is equal to the difference of two convex "LogSumExp" functions, and thus is also DC.

Remark 5.6 (XGD vs constrained GD)**.** We highlight that XGD is intimately related to a constrained GD procedure. For simplicity, let us focus on the minimization of the function $\ell(z_1, z_2) = -\log(\Xi_2(z_1, z_2)) =$

$-\log(\sigma(z_1)\sigma(z_2)) + \log(\sigma(z_1 + z_2))$. For any real constant κ, the function ℓ is convex over the region "$z_1 + z_2 = \kappa$". Hence, it can be minimized by gradient descent with respect to a single variable, say z_1, through the substitution $z_2 = \kappa - z_1$. By selecting $\mu = (\mu_1, 1 - \mu_1)$ with $\mu_1 = \sigma(\kappa)$, and *starting from any initial point* (even outside the "κ-region"), XGD allows to minimize ℓ over that same region. Although the benefit compared to constrained GD seems limited in this simple example (since the substitution $z_2 = \kappa - z_1$ is trivial), XGD becomes much more convenient in more complex scenarios involving large and high dimensional datasets. Indeed, XGD circumvents the double need of finding an initial point belonging to $\mathcal{E}(\mu)$, and of performing intricate projections onto this same set, which can be very challenging in general (especially for checkered regression models with $m > 2$).

5.2 Sum-log-concavity via circular convolution

The purpose of this subsection is to demonstrate that the checkered regression is a sum-log-concave model.

Proposition 5.3 (Circular convolution)**.** *For all $z = (z_1, \dots, z_m) \in \mathbb{R}^m$,*

$$\begin{pmatrix} \Xi_m(z) \\ 1 - \Xi_m(z) \end{pmatrix} = \begin{pmatrix} \sigma(z_1) \\ 1 - \sigma(z_1) \end{pmatrix} \circledast \cdots \circledast \begin{pmatrix} \sigma(z_m) \\ 1 - \sigma(z_m) \end{pmatrix} \ ,$$

where $\circledast$ denotes the circular convolution operator.

Proof. Let $m \geq 1$ and $z = (z_1, \dots, z_m) \in \mathbb{R}^m$. For all $1 \leq k \leq m$, let us denote the following binary probability laws:

$$b_k = \begin{pmatrix} \sigma(z_k) \\ 1 - \sigma(z_k) \end{pmatrix} \ , \tag{7}$$

with σ the sigmoid function. Then, the *Discrete Fourier Transform* (DFT in short) B_k of each b_k is given by:

$$B_k = \mathrm{DFT}(b_k) = \begin{pmatrix} \sigma(z_k) + 1 - \sigma(z_k) \\ \sigma(z_k) - (1 - \sigma(z_k)) \end{pmatrix} = \begin{pmatrix} 1 \\ 2\sigma(z_k) - 1 \end{pmatrix} = \begin{pmatrix} 1 \\ \tanh(\frac{z_k}{2}) \end{pmatrix} \ . \tag{8}$$

By the circular convolution theorem, we know that the circular convolution can be computed through the product of the discrete Fourier transforms. Thus, we consider the componentwise product of the B_k's:

$$\prod_{k=1}^{m} B_k = \begin{pmatrix} 1 \\ \prod_{k=1}^{m} \tanh(\frac{z_k}{2}) \end{pmatrix} \ . \tag{9}$$

And finally, we take the *Inverse Discrete Fourier Transform* (IDFT):

$$\mathrm{IDFT}(\prod_{k=1}^{m} B_k) = \frac{1}{2} \begin{pmatrix} 1 + \prod_{k=1}^{m} \tanh(\frac{z_k}{2}) \\ 1 - \prod_{k=1}^{m} \tanh(\frac{z_k}{2}) \end{pmatrix} = \begin{pmatrix} \Xi_m(z) \\ 1 - \Xi_m(z) \end{pmatrix} \ , \tag{10}$$

which is the binary law produced by the checkoid function Ξ_m.

□

In the next section, we generalize our checkered regression model to the multiclass setting. More precisely, we extend Proposition 5.3 by replacing sigmoids with SoftArgMax vectors.

Probabilistic interpretation The next result describes a probabilistic interpretation of the checkoid function in terms of the parity of the number of successes in Bernoulli trials. Although this is a corollary of Proposition 5.3, we provide an additional more intuitive proof.

Let $X_1, \dots, X_m$ be independent Bernoulli random variables valued in $\{-1, 1\}$ with success probabilities:

$$\mathbb{P}(X_k = 1) = \sigma(z_k). \tag{11}$$

For any $1 \leq k \leq m$, we refer to the event $(X_k = -1)$ as a "failure".

Proposition 5.4. *Let us flip the m independent Bernoulli coins defined in Eq. 11. Then,*

$$\mathbb{P}(\textit{total number of failures is even}) = \Xi_m(z_1, \dots, z_m) \ .$$

Proof.

$$\begin{aligned}
\mathbb{P}(\text{total number of failures is even}) &= \mathbb{P}(X_1 \times \cdots \times X_m = 1) \\
&= \frac{1}{2}\left(1 + \mathbb{E}[X_1 \times \cdots \times X_m]\right) \quad (\text{since } X_1 \times \cdots \times X_m \text{ is valued in } \{-1, 1\}) \\
&= \frac{1}{2}\left(1 + \mathbb{E}[X_1] \times \cdots \times \mathbb{E}[X_m]\right) \quad (\text{by independence}) \\
&= \frac{1}{2}\left(1 + (2\sigma(z_1) - 1) \times \cdots \times (2\sigma(z_m) - 1)\right) = \frac{1}{2}\left(1 + \prod_{k=1}^{m} \tanh(\frac{z_k}{2})\right) . \quad (12)
\end{aligned}$$

□

Proposition 5.4 shows that the quantity $\Xi_m(z)$ is a very natural one that results from combining several sigmoids.

GMM class posterior probability Furthermore, the checkoid function can be derived as the class posterior probability of a Gaussian mixture model (GMM).

Proposition 5.5 (Checkered GMM posterior)**.** *Let $d, m \geq 1$, $\mu_0, \mu_1, \dots, \mu_m \in \mathbb{R}^d$ and denote the matrix $\mu = (\mu_1, \dots, \mu_m) \in \mathbb{R}^{d \times m}$. Consider a random pair (X, Y) valued in $\mathbb{R}^d \times \{0, 1\}$ such that $\mathbb{P}(Y = 1) = \alpha$ and, given Y,*

$$X \sim \frac{1}{2^{m-1}} \sum_{\upsilon \in \{0,1\}^m : |\upsilon| \equiv Y [2]} \mathcal{N}(\mu_0 + \mu\upsilon, \Sigma) \ .$$

Then for all $x \in \mathbb{R}^d$,

$$\mathbb{P}(Y = +1 | X = x) = \Xi_m(w_1^\top x + b_1, \dots, w_m^\top x + b_m) \ ,$$

where $w_1, \dots, w_m, b_1, \dots, b_m$ are constants depending on α and the Gaussian parameters $\mu_1, \dots, \mu_m$.

Proof. The proof easily follows from:

$$\Xi_m(z) = \frac{\sum_{\upsilon \in \{0,1\}^m : |\upsilon| \equiv 0 [2]} e^{-\upsilon^\top z}}{(1 + e^{-z_1}) \times \cdots \times (1 + e^{-z_m})} \quad \text{and} \quad 1 - \Xi_m(z) = \frac{\sum_{\upsilon \in \{0,1\}^m : |\upsilon| \equiv 1 [2]} e^{-\upsilon^\top z}}{(1 + e^{-z_1}) \times \cdots \times (1 + e^{-z_m})} \ .$$

□

Of course, a more general version of Proposition 5.5 can be proved in the same way for mixtures of distributions belonging to any exponential family. In the multiclass case, one shall replace the set $\{0, 1\}^m$ by $\{0, \dots, c-1\}^m$ and the "modulo 2" by "modulo c". Note that the case $m = 1$ corresponds to the well-understood linearly separable case with a single Gaussian per class. The more complex case $m = 2$ with two Gaussian distributions per class is often called "XOR GMM" (for instance in Ben Arous et al. (2022)): see Figure 6 for an illustration.

5.3 Multiclass checkered regression

We now define a muticlass version of our model that can be used in classification problems having more than two classes. Let $\mathcal{Y} = \{0, \dots, c-1\}$ be the set of classes with $c \geq 2$.

Definition 5.7 (Multiclass checkered regression)**.** The multiclass checkered regression model with parameters $\Omega = (\Omega_1, \dots, \Omega_m) \in \mathbb{R}^{m \times c \times d}$ (with each $\Omega_k \in \mathbb{R}^{c \times d}$) is given, for any input vector $x \in \mathbb{R}^d$, by the posterior probabilities

$$\forall y \in \mathcal{Y} \ , \quad p_\Omega(y|x) = \Xi_{m,y}(\Omega_1 x, \dots, \Omega_m x) \ ,$$

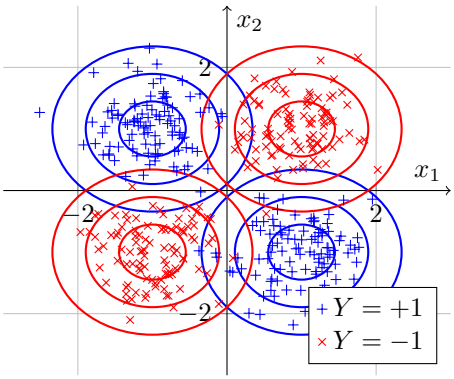

Figure 6: Cloud of points $X = (x_1, x_2)$ from two classes, each generated from a mixture of Gaussian distributions with identical covariance matrix equal to 0.2 times the 2-dimensional identity matrix. The positive class (blue +) is a balanced mixture of two Gaussians with respective means at $(1, -1)$ and $(-1, 1)$. The negative class (red $\times$) is also a balanced mixture of two Gaussians with respective means at $(1, 1)$ and $(-1, -1)$.

where the multiclass checkoid function $\Xi_{m,y}$ is defined for all $Z = (Z_1^\top, \dots, Z_m^\top)^\top \in \mathbb{R}^{m \times c}$ as

$$\vec{\Xi}_m(Z) = \begin{pmatrix} \Xi_{m,0}(Z) \\ \vdots \\ \Xi_{m,c-1}(Z) \end{pmatrix} = \vec{\sigma}(Z_1) \circledast \cdots \circledast \vec{\sigma}(Z_m) \ ,$$

where $\circledast$ denotes the circular convolution operator and we recall that the SoftArgMax function is defined at equation 2.

The multiclass checkered regression model is a simple way of obtaining a sum-log-concave probability distribution by taking the circular convolution of several SoftArgMax vectors. As in the binary case discussed before, it allows to represent richer structures than a standard multiclass logistic regression that only uses a single SoftArgMax distribution (particular case $m = 1$).

(Cross) Gradient Formulas We here provide the expressions for the (cross) gradients of the log-loss of the checkered regression model.

Proposition 5.6 (Gradient). *(i) Consider the logarithmic loss $\ell(z) = -\log(\Xi_m(z))$ for all $z = (z_1, \dots, z_m) \in \mathbb{R}^m$. Then for any $1 \le k \le m$, the k-th partial derivative of ℓ is*

$$\frac{\partial \ell}{\partial z_k}(z) = \sigma(z_k) \cdot \left(1 - \frac{\Xi_{m-1}(z_{-k})}{\Xi_m(z)}\right) \ ,$$

where $z_{-k} = (z_j)_{j \neq k}$.

(ii) For multiclass,

$$-\nabla_{Z_k} \log \Xi_{m,y}(Z) = \left[\sigma_l(Z_k)\left(1 - \frac{\Xi_{m-1,y-l}(Z_{-k})}{\Xi_{m,y}(Z)}\right)\right]_{0 \le l \le c-1} \ ,$$

where $Z_{-k} = (Z_j)_{j \neq k}$ and $y - l$ must be understood "modulo c".

In particular, these partial derivatives are all bounded in the interval $(-1, 1)$ since

$$\sum_{0 \le l \le c-1} \sigma_l(Z_k) \Xi_{m-1,y-l}(Z_{-k}) = \Xi_{m,y}(Z).$$

Proposition 5.7 (Cross-Gradient). *For $Z = (Z_1^\top, \dots, Z_m^\top)^\top, Z' = (Z_1'^\top, \dots, Z_m'^\top)^\top \in \mathbb{R}^{m \times c}$,*

$$-\nabla_{Z_k}^{Z'} \log \Xi_{m,y}(Z) = \sum_{j=0}^{c-1} \frac{\sigma_j(Z_k') \Xi_{m-1,y-j}(Z'_{-k})}{\Xi_{m,y}(Z')} \begin{bmatrix} \sigma_0(Z_k) \\ \vdots \\ \sigma_j(Z_k) - 1 \\ \vdots \\ \sigma_{c-1}(Z_k) \end{bmatrix} \ ,$$

which is equal to the gradient in Proposition 5.6-(ii) for $Z' = Z$.

Proof. We have

$$-\nabla_{Z_k}^{Z'} \log \Xi_{m,y}(Z) = \frac{1}{\Xi_{m,y}(Z')} \sum_{v \in \{0,\dots,c-1\}^m : |v| \equiv y[c]} \prod_{l=1}^{m} \sigma_{v_l}(Z'_l) \begin{bmatrix} \sigma_0(Z_k) \\ \vdots \\ \sigma_{v_k}(Z_k) - 1 \\ \vdots \\ \sigma_{c-1}(Z_k) \end{bmatrix}$$

$$= \frac{1}{\Xi_{m,y}(Z')} \sum_{j=0}^{c-1} \sigma_j(Z'_k) \underbrace{\sum_{\tilde{v} \in \{0,\dots,c-1\}^{m-1} : |\tilde{v}| \equiv y-j[c]} \prod_{l \neq k} \sigma_{\tilde{v}_l}(Z'_l)}_{\Xi_{m-1,y-j}(Z'_{-k})} \begin{bmatrix} \sigma_0(Z_k) \\ \vdots \\ \sigma_j(Z_k) - 1 \\ \vdots \\ \sigma_{c-1}(Z_k) \end{bmatrix}.$$

□

6 Conclusion

In this paper, we proposed an extension of the standard convex optimization framework for the class of functions given by the log-loss of a sum-log-concave function. We also proposed a sum-log-concave generalization of logistic regression. Future directions of research include:

- Exploring adaptive strategies for choosing the hyperparameter distribution μ in XGD, potentially through Dirichlet distribution random sampling.
- Combining XGD with GD, leveraging XGD's ability to avoid saddle points with a suitable μ, and GD's exploration of better minimizers beyond the μ-domain.
- Extending Fenchel's duality and biconjugate theorems to accommodate the sum-log-concave scenario.

Acknowledgments

The author thanks Nidham Gazagnadou for precious insights on convex optimization.

References

Mastane Achab. Checkered regression. 2022.

Francis Bach. Learning theory from first principles. *Draft of a book, version of Sept*, 6:2021, 2021.

Gerard Ben Arous, Reza Gheissari, and Aukosh Jagannath. High-dimensional limit theorems for sgd: Effective dynamics and critical scaling. *Advances in Neural Information Processing Systems*, 35:25349–25362, 2022.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

Marina Danilova, Pavel Dvurechensky, Alexander Gasnikov, Eduard Gorbunov, Sergey Guminov, Dmitry Kamzolov, and Innokentiy Shibaev. Recent theoretical advances in non-convex optimization. In *High-Dimensional Optimization and Probability: With a View Towards Data Science*, pp. 79–163. Springer, 2022.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *Advances in neural information processing systems*, 28, 2015.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.