Position: Adopt Constraints Over Penalties in Deep Learning

Juan Ramirez* **Meraj Hashemizadeh** **Simon Lacoste-Julien**‡
Mila - Quebec AI Institute and DIRO, Université de Montréal, Canada
‡ Canada CIFAR AI Chair

Abstract

Recent efforts to develop trustworthy AI systems with accountability guarantees have led to widespread use of machine learning formulations incorporating external requirements, or *constraints*. These requirements are often enforced via penalization—adding fixed-weight terms to the task loss. We argue this approach is fundamentally ill-suited since there may be no penalty coefficient that simultaneously ensures constraint satisfaction and optimal constrained performance, i.e., that truly solves the constrained problem. Moreover, tuning these coefficients requires costly trial-and-error, incurring significant time and computational overhead. We, therefore, advocate for broader adoption of tailored constrained optimization methods—such as the Lagrangian approach, which jointly optimizes the penalization "coefficients" (the Lagrange multipliers) and the model parameters. Such methods ① truly solve the constrained problem and do so accountably, by clearly defining feasibility and verifying when it is achieved, ② eliminate the need for extensive penalty tuning, and ③ integrate seamlessly with modern deep learning pipelines.

1 Introduction

The rapidly advancing capabilities of AI systems—exemplified by recent models like GPT-4 [53] and Gemini [34]—have enabled their deployment across a wide range of domains, including sensitive applications such as credit risk assessment [7], recidivism prediction [45], and autonomous driving [3]. These developments have raised concerns regarding the reliability, interpretability, fairness, and safety of such tools [24], prompting increasing calls for their regulation and standardization [28].

In response, ensuring that models satisfy desirable or mandated properties has become a central goal in deep learning. A principled approach is to express these requirements as *constraints*, embedding them into the optimization process to guide training toward compliant models. This procedure ensures accountability: compliance is demonstrated by verifying whether the constraints are satisfied. Such *constrained learning* problems have been applied in fairness [16] and safety [19], among others.

To solve these problems, the deep learning community has predominantly adopted the *penalized* approach [11, 12, 14, 25, 39, 44, 48, 59, 64], which incorporates constraints into the objective via fixed-coefficient penalty terms and minimizes the resulting function (see §2.1 or [26, §3]). This approach is applicable when all functions are differentiable and integrates naturally with standard tools for unconstrained deep learning optimization, such as Adam [43] and learning rate schedulers.

However, as detailed in §3, this approach has two major drawbacks: ① it may fail when the constrained problem is non-convex—as is common in deep learning—since there may be no choice of penalty coefficients for which the penalized minimizer optimally solves the constrained problem; and ② tuning coefficients to obtain a desirable solution is often challenging, typically requiring repeated trial-and-error model training runs that waste both computational resources and development time.

[*] Correspondence to: {juan.ramirez, merajhse}@mila.quebec

Preprint. Under review.

Instead, we advocate for a paradigm shift toward broader adoption of tailored constrained optimization methods in deep learning—approaches more likely to truly solve the constrained problem while reducing the burden of hyperparameter tuning. The position of this paper is summarized as follows:

Deep learning researchers and practitioners should ***avoid the penalized approach and instead consider tailored constrained optimization methods****—whenever the problem has explicit targets for some functions, that is, when it naturally fits a constrained learning formulation.*[2]

In §4.2, we contrast a bespoke constrained method—the Lagrangian approach [2], which also reformulates constrained problems via a linear combination of the objective and constraints—with the penalized approach to illustrate the superiority of principled constrained methods over penalization. The key distinction is that the Lagrangian approach is adaptable, as it *optimizes* its coefficients—the multipliers—instead of fixing them in advance. This adaptability underpins its effectiveness and has led to its widespread adoption for (principled) constrained deep learning [19, 27, 31, 38, 41, 58, 61].

Moreover, constrained optimization algorithms integrate easily with deep learning frameworks such as PyTorch [54] and TensorFlow [1], via open-source libraries like Cooper [32] and TFCO [17]. This integration lessens friction that might otherwise lead practitioners to favor penalized approaches. Beyond this practicality, constrained methods offer benefits such as support for non-differentiable constraints [16] and scalability to problems with infinitely many constraints [51] (see §4.3).

Finally, we highlight some limitations of current constrained deep learning algorithms in §5, aiming to encourage further research into methods better tailored to such large-scale and non-convex settings.

Alternative views. While we advocate in favor of constrained deep learning, it may not be ideal in applications where the goal is to optimize all objectives simultaneously or where acceptable constraint levels cannot be clearly defined. In such cases, bespoke methods may be more appropriate—though the constrained approach remains applicable (see Ehrgott [26, §4.1] for multi-objective settings and Hounie et al. [42] for cases where constraint levels are unknown or flexible).

Notably, in convex settings, the penalized and constrained formulations are equivalent (see §2.1), making the penalized approach appealing: it converts the problem into an unconstrained one and avoids the need for specialized constrained optimization techniques. However, we stress the limitations of penalty-based methods in constrained deep learning, where convexity is rare.

Scope. We primarily focus on inequality-constrained problems, as constrained deep learning setups that rely on penalization are typically framed via inequalities. For completeness, however, we present both the penalized (§2.1) and Lagrangian (§4.1) approaches for general constrained problems, including equality constraints. While the penalized approach is not commonly used for equality-constrained problems, we emphasize that it inherits the same drawbacks as in the inequality case—along with the added risk of failing to recover any feasible solution at all (see §3.1)—and should also be avoided.

Related works. Many of the ideas presented in this paper are not new. For example, the inability of penalization to fully explore all optimal trade-offs between non-convex functions is a well-established result [26, Ex. 4.2], and has also been raised in the context of deep learning optimization [21, 30, 31]. Likewise, the difficulty of tuning penalty coefficients is a common critique in constrained deep learning and is often cited as a key motivation for adopting the Lagrangian approach instead. Where prior works typically mention these issues only in passing, this position paper consolidates, formalizes, and expands on them, offering a comprehensive and standalone perspective. We especially acknowledge the blog posts by Degrave and Korshunova [21, 22], which share similar goals; we complement them with greater rigor and a perspective grounded in constrained optimization expertise.

2 Background

Let $f : \mathcal{X} \rightarrow \mathbb{R}$, $\boldsymbol{g} : \mathcal{X} \rightarrow \mathbb{R}^m$ and $\boldsymbol{h} : \mathcal{X} \rightarrow \mathbb{R}^n$ be twice-differentiable functions over $\mathcal{X} \subseteq \mathbb{R}^d$. We consider the constrained optimization problem:

$$\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) \preceq \boldsymbol{\epsilon_g} \quad \text{and} \quad \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{\epsilon_h}, \tag{1}$$

where $\boldsymbol{\epsilon_g}$ and $\boldsymbol{\epsilon_h}$ specify the constraint levels, and $\preceq$ denotes element-wise inequality. We refer to f as the objective function, and to $\boldsymbol{g}$ and $\boldsymbol{h}$ as the inequality and equality constraints, respectively.

[2] Even in multi-objective settings where all functions are minimized, we argue constrained formulations may be preferable to penalized ones for their superior ability to explore optimal trade-offs. See §3.1 for details.

A point $\boldsymbol{x}$ is *feasible* if it satisfies the constraints, and $\boldsymbol{x}^*$ is a *constrained minimizer* if it is feasible and satisfies $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all feasible $\boldsymbol{x}$. An inequality constraint $g_i(\boldsymbol{x}) \leq \epsilon$ is *active* at $\boldsymbol{x}$ if it holds with equality, i.e. $g_i(\boldsymbol{x}) = \epsilon$. Equality constraints are always active at feasible points.

Constrained formulations are ubiquitous in machine learning. Typically, the objective captures the learning task, while the constraints steer optimization toward models satisfying desirable properties such as sparsity [31], fairness [16] or safety [19]. For example, in a sparsity-constrained classification task, f may represent the cross-entropy loss over model parameters $\boldsymbol{x}$, while a constraint such as $\|\boldsymbol{x}\|_0 \leq \epsilon$ enforces a bound on the number of nonzero parameters. In some settings, however, learning is directly embedded in the constraints, as in SVMs and Feasible Learning [57].

A key distinction between the constrained problem in Eq. (1) and a corresponding multi-objective optimization problem—where f, $\boldsymbol{g}$, and $\boldsymbol{h}$ are all minimized—is that here $\boldsymbol{g}$ and $\boldsymbol{h}$ need only be *satisfied*, not optimized. In constrained optimization, only solutions that meet all constraints are admissible; any minimizer of f that violates them is disregarded, regardless of its objective value.[3]

Optimality. In unconstrained differentiable optimization, stationarity of the objective is a necessary condition for optimality. In constrained problems, however, stationary points of f can be infeasible, and optimal solutions may lie on the boundary of the feasible set ($\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{\epsilon_g}$) without being stationary points of the objective. To characterize optimality in constrained problems, we define the Lagrangian:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \triangleq f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \left[\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{\epsilon_g}\right] + \boldsymbol{\mu}^\top \left[\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{\epsilon_h}\right], \tag{2}$$

where $\boldsymbol{\lambda} \succeq \boldsymbol{0}$ and $\boldsymbol{\mu}$ are the Lagrange multipliers associated with the inequality and equality constraints, respectively. We refer to $\boldsymbol{x}$ as the *primal* variables, and to $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ as the *dual* variables.

A necessary condition for constrained optimality of $\boldsymbol{x}^\dagger$ is the existence of multipliers $\boldsymbol{\lambda}^\dagger \succeq \boldsymbol{0}$ and $\boldsymbol{\mu}^\dagger$ such that $(\boldsymbol{x}^\dagger, \boldsymbol{\lambda}^\dagger, \boldsymbol{\mu}^\dagger)$ is a stationary point of the Lagrangian[4]—this is equivalent to satisfying the Karush-Kuhn-Tucker (KKT) first-order conditions [9, §5.5.3]. A sufficient condition is that $\nabla^2_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}^\dagger, \boldsymbol{\lambda}^\dagger, \boldsymbol{\mu}^\dagger)$ is positive definite in all directions within the null space of the active constraints and that strict complementary slackness holds (see Bertsekas [5, Prop. 4.3.2] for details). Under regularity conditions at a constrained minimizer $\boldsymbol{x}^*$—such as Mangasarian-Fromovitz [49]—the existence of corresponding multipliers $\boldsymbol{\lambda}^* \succeq \boldsymbol{0}$ and $\boldsymbol{\mu}^*$ satisfying these conditions is guaranteed.

A saddle-point $(\boldsymbol{x}^\dagger, \boldsymbol{\lambda}^\dagger, \boldsymbol{\mu}^\dagger)$ of $\mathcal{L}$—where $\boldsymbol{x}^\dagger$ minimizes and $\boldsymbol{\lambda}^\dagger \succeq \boldsymbol{0}$, $\boldsymbol{\mu}^\dagger$ maximize—yields a constrained minimizer $\boldsymbol{x}^\dagger$ of Eq. (1). However, note that being a saddle-point is a stronger condition than satisfying the KKT conditions and is not necessary for optimality: in general, $\boldsymbol{x}^\dagger$ need not minimize $\mathcal{L}(\,\cdot\,, \boldsymbol{\lambda}^\dagger, \boldsymbol{\mu}^\dagger)$ to be constrained-optimal, and $\nabla^2_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}^\dagger, \boldsymbol{\lambda}^\dagger, \boldsymbol{\mu}^\dagger)$ need not be positive definite in all directions. That is, the Lagrangian may be *strictly concave* in some (infeasible) directions—unlike in unconstrained minimization, where strict concavity at a point precludes its optimality.

This distinction is important because saddle-points of the Lagrangian may not exist for general non-convex problems, yet stationary points corresponding to solutions to Eq. (1) can still exist under constraint qualifications (e.g., see Example 1). Moreover, even when constraint qualifications fail, asymptotic KKT conditions still hold at constrained minimizers $\boldsymbol{x}^*$: for every $\epsilon > 0$, ϵ-approximate stationary points of $\mathcal{L}$ exist—even if forming exact stationary points including $\boldsymbol{x}^*$ may require $\|\boldsymbol{\lambda}^*\| \to \infty$ or $\|\boldsymbol{\mu}^*\| \to \infty$ [8, Thm. 3.1].

Constrained deep learning—where $\boldsymbol{x}$ represents the parameters of a neural network and evaluating f, $\boldsymbol{g}$, and $\boldsymbol{h}$ involves computations over data—presents unique challenges: ① it is *large-scale*, often involving billions of parameters; ② *non-convex*, due to network nonlinearities; and ③ *stochastic*, as f, $\boldsymbol{g}$, and $\boldsymbol{h}$ are typically estimated from mini-batches. These properties limit the applicability of many standard constrained optimization techniques; see §5 for details. Hence, the deep learning community largely relies on the penalized approach for handling constraints (§2.1)—which we argue in §3 is not principled. In §4, we present an alternative better suited to constrained deep learning.

[3] In practice, small numerical violations of the constraints are often tolerated.

[4] For convenience, we overload the term "stationary point" to include not only points where the gradient vanishes but also all other candidate optima such as boundary points (where all non-boundary variables have zero gradients). This definition enables us to capture all candidate optima under a unified notion of stationarity. For the Lagrangian, this includes both classical stationary points and points where $\lambda_i = 0$ for some (or several) indices i, provided the usual stationarity conditions hold with respect to $\boldsymbol{x}$, $\boldsymbol{\mu}$, and the remaining λ_j terms.

2.1 The penalized approach

A popular approach to tackling Eq. (1) in deep learning applications is the *penalized* approach—also known as a weighted-sum scalarization of multi-objective problems [26, §3]—which minimizes a linear combination of the objective and the constraints:

$$\min_{\boldsymbol{x}\in\mathcal{X}} \mathcal{P}(\boldsymbol{x}) \triangleq f(\boldsymbol{x}) + \boldsymbol{c}_{\boldsymbol{g}}^\top \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{c}_{\boldsymbol{h}}^\top \boldsymbol{h}(\boldsymbol{x}), \tag{3}$$

where $\boldsymbol{c}_{\boldsymbol{g}} \succeq \mathbf{0}$ and $\boldsymbol{c}_{\boldsymbol{h}}$ are coefficients that encourage constraint satisfaction.

In deep learning applications, the popularity of the penalized approach likely stems from several factors: ① it reformulates constrained problems as *unconstrained* ones, allowing the use of familiar gradient-based optimization tools without requiring specialized knowledge of constrained optimization; ② it integrates smoothly with existing training protocols—such as Adam [43] or learning rate schedulers—which are often highly specialized for deep neural network training [18]; and ③ it can be applied to problems where both the objective and constraint functions are differentiable, without relying on assumptions like convexity, making it well-suited to modern deep learning tasks.

Moreover, the penalized approach is computationally efficient in the context of automatic differentiation. The gradient of $\mathcal{P}$ is a linear combination of the gradients of f, $\boldsymbol{g}$ and $\boldsymbol{h}$, and can be computed without storing each gradient separately. Since constraints often share intermediate computations with the objective—such as forward passes through the model—additional forward calculations and the backward pass can be executed with minimal overhead compared to optimizing a single function.

In the convex setting, the penalized approach is justified in the following sense:

Proposition 1 (Convex penalization)**.** [Proof] *Let f and $\boldsymbol{g}$ be differentiable and convex, and let $\boldsymbol{h}$ be affine. Assume the domain $\mathcal{X}$ is convex.*

Let $\boldsymbol{x}^* \in \operatorname{relint} \mathcal{X}$ be a constrained minimizer of Eq. (1) that satisfies constraint qualifications, such that there exist optimal Lagrange multipliers $\boldsymbol{\lambda}^* \succeq \mathbf{0}$ and $\boldsymbol{\mu}^*$ making $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ a stationary point of the Lagrangian. Then, by choosing $\boldsymbol{c}_{\boldsymbol{g}} = \boldsymbol{\lambda}^*$ and $\boldsymbol{c}_{\boldsymbol{h}} = \boldsymbol{\mu}^*$, we have

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} \mathcal{P}(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{c}_{\boldsymbol{g}}^\top \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{c}_{\boldsymbol{h}}^\top \boldsymbol{h}(\boldsymbol{x}). \tag{4}$$

Consequently, when ① f, $\boldsymbol{g}$, and $\boldsymbol{h}$ are convex; ② the constrained minimizer of Eq. (1) is regular; and ③ reliable estimates of its corresponding optimal multipliers $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ are available, minimizing $\mathcal{P}$ is a principled approach to solving Eq. (1).

Furthermore, under these convexity assumptions, varying the weights $(\boldsymbol{c}_{\boldsymbol{g}}, \boldsymbol{c}_{\boldsymbol{h}})$ allows the penalized formulation to recover every attainable (weakly) Pareto-optimal[5] point $(f^\star, \boldsymbol{\epsilon}_{\boldsymbol{g}}, \boldsymbol{\epsilon}_{\boldsymbol{h}})$ in the image of the map $\boldsymbol{x} \mapsto (f(\boldsymbol{x}), \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}))$ [50, Thm. 3.1.4, §3.1]. Hence, with a suitable choice of coefficients, the penalized approach can recover a constrained minimizer of Eq. (1) for *all* attainable constraint levels. In this sense, the penalized and constrained formulations are equivalent for convex problems.

However, while effective in the convex setting and applicable to any differentiable constrained problem, the penalized approach is ill-suited. First, Prop. 1 no longer holds when f, $\boldsymbol{g}$, or $\boldsymbol{h}$ are non-convex: as discussed in §3.1, certain realizable values of $\boldsymbol{g}$ and $\boldsymbol{h}$ may be unattainable by minimizing the penalized objective $\mathcal{P}$, regardless of the choice of penalty coefficients $(\boldsymbol{c}_{\boldsymbol{g}},\ \boldsymbol{c}_{\boldsymbol{h}})$. Therefore, constrained minimizers for certain attainable constraint levels $(\boldsymbol{\epsilon}_{\boldsymbol{g}}, \boldsymbol{\epsilon}_{\boldsymbol{h}})$ may not be recoverable using the penalized approach—even though proper constrained methods can recover them (see §4.2).

Moreover, the success of the penalized approach relies on having accurate estimates of the optimal Lagrange multipliers, such that $\boldsymbol{c}_{\boldsymbol{g}} \approx \boldsymbol{\lambda}^*$ and $\boldsymbol{c}_{\boldsymbol{h}} \approx \boldsymbol{\mu}^*$. In practice, however—and especially in non-convex problems—these multipliers are unknown. As a result, practitioners often resort to a costly trial-and-error process to tune the penalty coefficients (see §3.2).

3 Limitations of the penalized approach

While pervasive, the penalized approach has attracted substantial criticism [21, 30, 31, 55]. Here, we revisit and formalize these concerns; in §4, we discuss how constrained methods address them.

[5] For vector-valued functions $f_i : \mathbb{R}^d \to \mathbb{R}$, a point $x^\star$ is *weakly Pareto-optimal* if there is no other feasible point x such that $f_i(x) < f_i(x^\star)$ for *every* index i.

3.1 The penalized and constrained problems are not equivalent

In non-convex settings, the guarantees in Prop. 1 no longer hold. The penalized approach may then fail to recover certain achievable constraint levels—regardless of the choice of penalty coefficients—resulting in an ill-posed formulation that cannot recover constrained minimizers of Eq. (1).

Specifically, while (approximately) feasible solutions for inequality-constrained problems can always be recovered by choosing a sufficiently large penalty coefficient, recovering constrained-optimal solutions may be impossible—especially when these involve active constraints $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{\epsilon_g}$, which is often the case.[6] The situation is even more problematic for equality constraints: if the target value $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{\epsilon_h}$ is unattainable, the penalized approach may fail to produce any feasible solution at all.

In extreme cases, the penalized objective can completely disregard certain functions and focus solely on minimizing others. This failure mode is illustrated by the two-dimensional inequality-constrained problem in Example 1, originally presented in an insightful blog post pair by Degrave and Korshunova [21, 22]. We formalize their empirical observation of the penalized method's failure in Prop. 3.

Example 1 (Concave 2D problem)**.** Consider the following constrained optimization problem:

$$\min_{x,y} \; f(x,y) \triangleq (1+y^2)\cos(x) \quad \text{subject to} \quad g(x,y) \triangleq (1+y^2)\sin(x) \leq \epsilon, \tag{5}$$

where $x \in [0, \pi/2]$, $y \in \mathbb{R}$, and $\epsilon \in (0,1)$. This problem violates the premise of Prop. 1 since both the objective and constraint functions are *concave* in x over the interval $[0, \pi/2]$.

For an illustration of Example 1 and an intuitive derivation of its solution, see Fig. 3 in Appendix B.1.

Proposition 2 (Solution to Example 1)**.** [Proof] *The constrained minimizer of Example 1 is* $(x^*, y^*) = (\arcsin(\epsilon), 0)$, *with corresponding Lagrange multiplier* $\lambda^* = \epsilon/\sqrt{1-\epsilon^2}$. *At this point, we have* $f\left(\arcsin(\epsilon), 0\right) = \sqrt{1-\epsilon^2}$ *and* $g\left(\arcsin(\epsilon), 0\right) = \epsilon$, *indicating that the constraint is active.*

Proposition 3 (The penalized approach's failure)**.** [Proof] *Consider the penalized formulation of Example 1:*

$$\min_{x \in [0,\pi/2], y} \; \mathcal{P}_c(x,y) \triangleq (1+y^2)\cos(x) + c\,(1+y^2)\sin(x), \tag{6}$$

where $c \geq 0$ *is a penalty coefficient.*

For any $c \in [0,1)$, *the unique solution to Eq. (6) is* $(x^*, y^*) = (\pi/2, 0)$, *with* $f(\pi/2, 0) = 0$ *and* $g(\pi/2, 0) = 1 > \epsilon$. *For* $c > 1$, *the unique solution becomes* $(x^*, y^*) = (0,0)$, *yielding* $f(0,0) = 1$ *and* $g(0,0) = 0$. *When* $c = 1$, *the penalized objective has two minimizers:* $(0,0)$ *and* $(\pi/2, 0)$.

In fact, the penalized objective $\mathcal{P}_c(x,y)$ *is concave in* $x \in [0, \pi/2]$ *for all* $c \geq 0$.

Prop. 3 shows that for any choice of penalty coefficient, the penalized formulation of Example 1 either recovers the unconstrained minimizer of f by effectively ignoring the constraint g—resulting in an infeasible solution—or prioritizes minimizing g, yielding a feasible point with $g(x^*, y^*) = 0 < \epsilon$, but at the cost of significantly suboptimal performance on f. Notably, even when the penalty coefficient is set to the optimal Lagrange multiplier of the constrained problem ($c = \epsilon/\sqrt{1-\epsilon^2}$), the penalized formulation still fails to recover the constrained minimizer at $(x^*, y^*) = (\arcsin(\epsilon), 0)$.

The penalized approach fails to solve Example 1 because its objective $\mathcal{P}_c$ is concave in x for any coefficient choice, pushing minimizers to the domain boundaries. This issue extends to general non-convex problems: if a (local) constrained optimum lies within a non-convex region of the penalized function $\mathcal{P}$, it becomes unreachable, as it does not minimize $\mathcal{P}$. Moreover, verifying whether $\mathcal{P}$ is convex around a solution often requires knowing the solution—i.e., solving the problem first.

A note on multi-objective problems. The exploration failure of the penalized approach carries over to multi-objective settings, where the goal is to minimize several objectives simultaneously. Its inability to capture the full range of optimal trade-offs can limit exploration of the Pareto front [26, §4]. As with constrained problems, we emphasize the importance of using specialized techniques for non-convex multi-objective optimization rather than relying on the penalized approach.[7]

[6] Under inequality constraints, optima often lie on the boundary: reducing $\boldsymbol{g}$ below $\boldsymbol{\epsilon_g}$ typically worsens f.

[7] The constrained approach is a valid method for multi-objective optimization; see Ehrgott [26, §4.1].

3.2 Penalty coefficients are costly to tune

Even when the penalized formulation is principled—i.e., when Prop. 1 holds—selecting appropriate penalty coefficients remains a costly trial-and-error process. The penalized problem often must be solved repeatedly with different coefficient values before arriving at a solution that satisfies the constraints and delivers good performance. This iterative re-training wastes both computational resources and development time, placing a significant burden on the community.

The unnecessarily lengthy process of tuning penalty coefficients is illustrated in Fig. 1, where the goal is to train a neural network under a constraint enforcing at least 50% sparsity. Using bisection search—which leverages the monotonic relationship between the coefficient and the resulting constraint value—the penalized problem must be solved five times before finding a satisfactory solution: one that is feasible but not overly sparse, as excessive sparsity would degrade performance. Even with this simple tuning method, the example demonstrates the process's inefficiency. In contrast, a proper constrained optimization method can solve the problem *in one shot*, requiring only a single model training run [31].

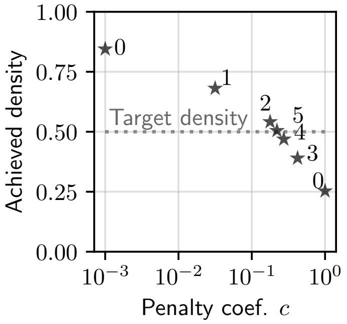

Figure 1: Solving a sparsity-constrained training problem. Starting from coefficients yielding low (25%) and high (85%) model density, **five bisection search steps are needed to reach 50% sparsity**. Annotations show iterations; endpoints are labeled as 0. Reproduces Fig. 5(b) from App. E of [31] (Details in App. B.2).

Several factors make coefficient tuning challenging: ① The relationship between a given coefficient and the resulting values of f, $\boldsymbol{g}$, and $\boldsymbol{h}$ is often highly non-linear, requiring specialized hyperparameter tuning techniques or even manual intervention; ② Penalty coefficients lack intuitive semantics, making them difficult to interpret; and ③ There is no general principled strategy for initializing them.

In contrast, constraint levels $(\boldsymbol{\epsilon_g}, \boldsymbol{\epsilon_h})$ in constrained formulations ① map directly to constraint values and ② are interpretable, since their units match those of the constraints. In constrained approaches that learn the "coefficients"—e.g., the multipliers in the Lagrangian approach—③ initialization is less critical, as they adapt during optimization and are thus typically set to zero for simplicity.

Moreover, tuning penalty coefficients becomes increasingly impractical in the presence of multiple constraints, and is further complicated by the fact that coefficient choices rarely transfer across tasks.

Multiple constraints. Tuning penalty coefficients becomes substantially more difficult in the presence of multiple constraints. Unlike the single-constraint case, multiple coefficients must be tuned jointly, as each one influences the satisfaction of all constraints. Moreover, while the monotonic relationship between coefficients and constraint values is straightforward to exploit in the single-constraint setting (e.g., as demonstrated in Fig. 1), extending this to the multi-constraint setting is challenging. This interdependence significantly increases the complexity of the tuning process, often rendering it impractical and potentially leading practitioners to accept suboptimal—or even infeasible—solutions.

Penalty coefficients rarely transfer across settings. The same coefficient can yield different outcomes across architectures and datasets, due to changes in the landscape of the penalized objective. Moreover, adding new constraints to an existing problem typically requires re-tuning all coefficients, as feasibility may depend on rebalancing them to account for both existing and new constraints.

In contrast, the hyperparameters of the constrained approach—the constraint levels $\boldsymbol{\epsilon_g}$ and $\boldsymbol{\epsilon_h}$—are agnostic to the number of constraints, their parameterization, or their relationship to model inputs. While some constrained methods may still require hyperparameter tuning when tasks change or new constraints are added, these typically transfer more reliably than penalty coefficients (see §4.2).

4 Getting started with constrained learning: the Lagrangian approach

We adopt the Lagrangian approach [2] to illustrate how a principled non-convex constrained optimization method addresses the limitations of the penalized approach. We choose this approach because of its broad applicability (requiring only differentiability), its similarity to penalized methods (facilitating comparison), its widespread use for (principled) constrained deep learning [30], and the limited repertoire of methods for large-scale non-convex constrained optimization (see §5 for details).

4.1 The Lagrangian approach

The Lagrangian approach [2] amounts to finding stationary points of the Lagrangian, which may correspond to solutions of the original constrained optimization problem (see §2):

$$\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^* \in \arg \min_{\boldsymbol{x} \in \mathcal{X}} \max_{\boldsymbol{\lambda} \succeq \mathbf{0}, \boldsymbol{\mu}} \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*). \tag{7}$$

This reformulates the constrained problem as an effectively unconstrained[8] optimization problem. Notably, the Lagrangian is always linear—and therefore concave—in the multipliers.

A simple algorithm for finding stationary points of $\mathcal{L}$ is alternating gradient descent–ascent (GDA):

$$\boldsymbol{\mu}_{t+1} \leftarrow \boldsymbol{\mu}_t + \eta_{\text{dual}} \nabla_{\boldsymbol{\mu}} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{\lambda}_t, \boldsymbol{\mu}_t) = \boldsymbol{\mu}_t + \eta_{\text{dual}} \big[\boldsymbol{h}(\boldsymbol{x}_t) - \boldsymbol{\epsilon}_{\boldsymbol{h}}\big], \tag{8}$$

$$\boldsymbol{\lambda}_{t+1} \leftarrow \Big[\boldsymbol{\lambda}_t + \eta_{\text{dual}} \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{\lambda}_t, \boldsymbol{\mu}_t)\Big]_+ = \Big[\boldsymbol{\lambda}_t + \eta_{\text{dual}} \big[\boldsymbol{g}(\boldsymbol{x}_t) - \boldsymbol{\epsilon}_{\boldsymbol{g}}\big]\Big]_+, \tag{9}$$

$$\boldsymbol{x}_{t+1} \leftarrow \left[\boldsymbol{x}_t - \eta_{\text{primal}} \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{\lambda}_{t+1}, \boldsymbol{\mu}_{t+1})\right]_{\mathcal{X}}, \tag{10}$$

where $[\,\cdot\,]_+$ denotes a projection onto $\mathbb{R}^m_{\succeq \mathbf{0}}$ to enforce $\boldsymbol{\lambda} \succeq \mathbf{0}$, $[\,\cdot\,]_{\mathcal{X}}$ denotes a projection onto $\mathcal{X}$, and $\eta_{\{\text{primal,dual}\}}$ are step-sizes. The multipliers are typically initialized as $\boldsymbol{\lambda}_0 = \mathbf{0}$ and $\boldsymbol{\mu}_0 = \mathbf{0}$. We favor alternating updates over simultaneous ones due to their enhanced convergence guarantees under a strongly convex $\mathcal{L}$ [63], without incurring additional computational cost on Lagrangian games [60].

Note that the update with respect to $\boldsymbol{x}$ in Eq. (8) matches that of (projected) gradient descent on the penalized objective, with $\boldsymbol{c_g} = \boldsymbol{\lambda}_{t+1}$ and $\boldsymbol{c_h} = \boldsymbol{\mu}_{t+1}$. However, unlike penalty coefficients, which require manual tuning, the multipliers are instead *optimized* to balance feasibility and optimality.

The Lagrangian approach has been successfully applied to a wide range of constrained deep learning problems [16, 19, 27, 31, 38, 40, 41, 57, 58, 61]. Beyond this effectiveness, it is also viable in practice, as discussed below.

Dynamics of GDA. Consider a single inequality constraint $g(\boldsymbol{x}) \leq 0$. The corresponding dual variable λ is updated based on the constraint violation: when $g(\boldsymbol{x}) > 0$, λ increases by $\eta_{\text{dual}}\, g(\boldsymbol{x})$. Repeated violations drive λ larger, which shifts the primal update toward reducing $g(\boldsymbol{x})$ and promoting feasibility. Conversely, when $g(\boldsymbol{x}) < 0$, λ decreases. In turn, a smaller multiplier shifts focus away from constraint satisfaction and toward minimizing the objective. When $\lambda = 0$, the constraint is ignored. For instance, consider a classification problem with a sparsity constraint. When the number of non-zero parameters exceeds the constraint level, the multiplier increases, pushing the model to be sparser. For an equality constraint $h(\boldsymbol{x}) = \epsilon$, the multiplier μ increases when $h(\boldsymbol{x}) > \epsilon$. Conversely, it decreases—becoming negative—when $h(\boldsymbol{x}) < \epsilon$. These dynamics encourage h to approach ϵ.

Computational cost. Applying (mini-batch) GDA to the Lagrangian is as efficient as applying (mini-batch) GD to the penalized objective $\mathcal{P}$, since the primal updates in both cases follow a linear combination of the objective and constraint gradients. The only overhead is updating the multipliers, which is negligible when the number of constraints is much smaller than dim$(\mathcal{X})$.

Tuning $\boldsymbol{\eta_{\text{dual}}}$. A larger dual step-size makes the multipliers respond more aggressively to constraint violations: if η_{dual} is too small, constraints may remain unsatisfied within the optimization budget; if too large, rapidly growing multipliers can overshoot their targets $\boldsymbol{\epsilon_g}$ and $\boldsymbol{\epsilon_h}$. In practice, however, tuning η_{dual} is significantly easier than tuning penalty coefficients—a point we elaborate on in §4.2.

Implementation. The Lagrangian approach is straightforward to implement in popular deep learning frameworks such as PyTorch [54] and JAX [10]. Open-source libraries like Cooper [32] for PyTorch and TFCO [17] for TensorFlow [1] offer ready-to-use implementations, along with additional features such as the Augmented Lagrangian [4], Proxy Constraints [16], Multiplier Models [51], and νPI [60].

Convergence. If $\mathcal{L}$ is strongly convex in $\boldsymbol{x}$, Eq. (7) becomes a strongly convex–concave min–max problem, for which alternating GDA enjoys local linear convergence [63]. Moreover, simultaneous GDA converges to local min–max points in non-convex–concave settings, provided the maximization domain is compact [47]. While the domains of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are unbounded, if the optimal multipliers are finite, restricting optimization to a bounded region containing them allows these guarantees to hold.

Generalization. Chamon and Ribeiro [13] develop a PAC learning framework for constrained learning. They argue that hypothesis classes that are PAC-learnable in the unconstrained setting may remain learnable under statistical constraints, probably approximately satisfying them on unseen data.

[8] Given that $\boldsymbol{\lambda} \succeq \mathbf{0}$ can be easily handled via projections.

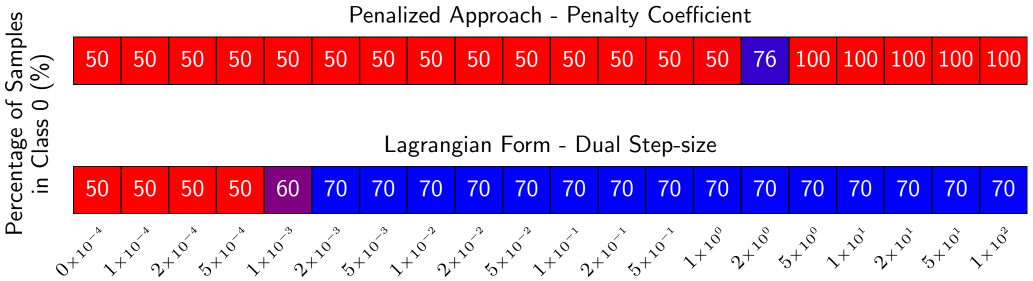

Figure 2: Solving a binary classification task under a constraint that at least 70% of predictions belong to one class. **The penalized approach satisfies the constraint with reasonable performance only for a specific coefficient, while the Lagrangian approach recovers approximate constrained minimizers across a broad range of dual step-sizes.** Task based on Fig. 2 of [16] (details in App. B.3).

4.2 Contrasting the Lagrangian and penalized approaches

This section elaborates on why the simple Lagrangian approach is, in many respects, preferable to the penalized formulation. In short, transitioning to the Lagrangian approach—which can be seen as a principled alternative to manually tuning penalty coefficients by instead increasing them when constraints are violated and decreasing them when satisfied—yields an algorithm that: ① is explicitly designed to solve the constrained problem, and is therefore expected to do so; and ② requires substantially less manual hyperparameter tuning. As discussed in §4.3, it also offers several additional benefits, many of which remain relatively underexplored in the deep learning literature.

The Lagrangian approach is constrained by design. Lagrangian optimization continues until a feasible solution is found. In contrast, the penalized approach may converge while still producing an infeasible solution. This constrained-by-design philosophy also introduces a notion of *accountability*: by explicitly specifying acceptable values for $\boldsymbol{g}$ and $\boldsymbol{h}$ in the formulation, we adopt a more principled framework for both solving the problem and evaluating success—where infeasibility is not acceptable.

The Lagrangian approach can solve non-convex problems. As shown in Table 1 (App. C.1), the Lagrangian approach successfully recovers the correct solution to Example 1. Note that this requires either a specialized dual optimizer such as νPI [60], or the Augmented Lagrangian method; standard gradient descent-ascent fails in this setting due to the concavity of the Lagrangian with respect to x. νPI overcomes this by dampening the oscillations that arise with GDA, while the Augmented Lagrangian method convexifies the problem—both enabling convergence to the constrained optimum (see App. D for details). Hyperparameter choices for Table 1 are provided in App. B.1.

Hyperparameter tuning. The Lagrangian approach eliminates the need to tune penalty coefficients, but introduces a complementary challenge: selecting the dual step-size η_{dual}. However, this tuning is generally more forgiving—whereas typically only a single choice of penalty coefficient yields a satisfactory solution, a broad range of dual step-sizes often leads to feasibility, sometimes spanning several orders of magnitude. This is because the Lagrangian approach always pursues constraint satisfaction—regardless of how aggressively, as determined by the step-size—unlike the penalized approach, which is agnostic to the constraint levels. For this reason, we argue that this relative ease of tuning likely extends to other constrained optimization methods beyond the Lagrangian approach.

Figure 2 illustrates this on a rate-constrained binary classification task [16], where 70% of predictions are required to belong to class "0". Since the data is balanced, the optimal constrained solution should assign 70% of predictions to class 0—this is achieved with an optimal Lagrange multiplier of $\lambda^* = 1.9975$, as found experimentally via the Lagrangian approach. The penalized formulation is infeasible for all $c < 2$; for $c > 2$, it satisfies the constraint but collapses to predicting a single class, degrading performance. Only at $c = 2$ does it yield a reasonable prediction rate of 76%. In contrast, the Lagrangian approach with proxy constraints [16] consistently recovers the correct solution over a wide range of dual step-sizes by automatically learning the appropriate multiplier (see App. B.3).

4.3 Further benefits of the Lagrangian approach

Recent advances in constrained learning have equipped the Lagrangian approach with new capabilities that further strengthen its appeal in deep learning applications.

Non-differentiable constraints. Cotter et al. [16] show that the Lagrangian approach can be extended to satisfy non-differentiable constraints by using a differentiable surrogate during the primal update—where constraint gradients are required—while relying on the original, non-differentiable constraint when updating the multipliers. This "proxy constraints" approach ensures that constraint enforcement is driven by the true quantity of interest, independent of the choice of surrogate.

This capability is particularly relevant in machine learning, where optimization often relies on differentiable surrogates that do not match the true objective. While unconstrained empirical risk minimization offers no direct way to optimize for accuracy—e.g., optimizing cross-entropy instead—accuracy-based constraints can be enforced via the Lagrangian formulation with proxy constraints [16, 38].

Scaling the number of constraints. Each additional constraint introduces one more Lagrange multiplier—a single scalar. While this already scales efficiently, especially given the Lagrangian approach's greater robustness to hyperparameters compared to the penalized formulation (§4.2), scalability can be further improved using parametric Lagrange multiplier models [51]. These models use a fixed set of parameters to represent all multipliers, regardless of the number of constraints, and can thus handle even problems with infinitely many constraints—unlike the penalized approach.

Sensitivity analysis via the multipliers. A standard result states that the optimal Lagrange multipliers $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ capture how much the objective would improve if a constraint were relaxed [9, §5.6]. This can be exploited to design algorithms that trade off constraint tightness for performance [42], making the Lagrangian approach well suited to settings with flexible or uncertain constraint levels.

Efficient dual variance reduction. Variance reduction techniques [36], popular for accelerating convergence in unconstrained optimization, have seen limited adoption in deep learning due to poor scalability—e.g., SAGA [20] requires storing gradients for each datapoint. In contrast, they are inexpensive to apply to Lagrange multiplier updates, as the dual gradient is simply the constraint value, and have been shown to enhance convergence for problems involving stochastic constraints [38].

Hyperparameter scaling. We hypothesize that, for a given task, the choice of dual step-sizes transfers well across model architectures and datasets. This would make hyperparameter scaling in the constrained approach significantly easier—allowing practitioners to tune parameters on smaller models and reuse them as effective starting points for larger-scale training, saving time and resources.

5 Call for research: enhancing constrained deep learning

Constrained learning still has notable limitations. This section reviews the key challenges and seeks to spur further research on tailored constrained deep learning algorithms.

Choosing the constraint levels ϵ_g and ϵ_h. Domain knowledge often informs acceptable constraint levels; however these are sometimes unclear. If set too tightly, the problem may become infeasible; if too loose, the desired properties may be weakly enforced. In non-convex settings, it is often unclear whether the levels are overly strict or too lax. This can be mitigated by introducing slack variables that adjust constraint levels to balance the objective and constraints [42]. However, when no meaningful levels can be specified, tailored multi-objective optimization methods may be more appropriate.

Limited repertoire of methods for constrained deep learning. The non-convexity of the feasible set in Eq. (1) rules out standard methods such as projected gradient descent [6, 35, 46] and feasible direction methods [29, 65], as computing projections or feasible directions over non-convex sets is often intractable. Methods that require all iterates to remain feasible, such as barrier methods [23], also break down in practice when constraints are estimated from mini-batches, due to variability in feasibility assessment across batches. The scale of deep learning further limits practical options: for example, vanilla SQPs [37, 56, 62] incur a per-iteration cost of $\mathcal{O}(d^3)$, which is prohibitive for large models. Penalty methods such as the quadratic penalty method [52, §17.1] remain applicable, but often require excessively large penalty coefficients to succeed—leading to numerical instability.

While the Lagrangian [2] and Augmented Lagrangian [4] approaches remain applicable—both reformulating constrained problems as min-max games—they still face challenges:

Non-existence of optimal Lagrange multipliers. A necessary and sufficient condition for the existence of finite optimal Lagrange multipliers satisfying the KKT conditions at $\boldsymbol{x}^*$ in differentiable problems is the Mangasarian–Fromovitz constraint qualification (MFCQ) [33, 49]. However, when MFCQ fails, the Lagrangian and Augmented Lagrangian approaches do not necessarily break down.

Convergence to $\boldsymbol{x}^*$ can still be achieved approximately by allowing $\|\boldsymbol{\lambda}^*\| \to \infty$ or $\|\boldsymbol{\mu}^*\| \to \infty$ (see §2 for details). Although large multipliers may lead to numerical instability when optimizing for $\boldsymbol{x}$, many problems still admit approximate constrained minimizers that are sufficiently close to $\boldsymbol{x}^*$ without requiring excessively large multipliers.

Thus, the failure of MFCQ does not necessarily make the Lagrangian approach ill-posed. However, it does limit the effectiveness of dual updates via gradient ascent. As primal iterates approach feasibility and constraint violations decrease, the dual updates tend to slow and may eventually stall—causing the multipliers to plateau before they have fully closed the feasibility gap.

Optimization. The min-max structure of the Lagrangian formulation can induce oscillations in the multipliers and their associated constraints, slowing convergence—an issue further exacerbated by dual momentum [60]. These oscillations can be mitigated using PI controllers on the multipliers [60, 61] or by the Augmented Lagrangian approach [55]. However, both methods introduce additional hyperparameters: the damping coefficient in the former and the penalty coefficient in the latter.

While helpful in practice, tuning these additional hyperparameters remains challenging. Even setting the dual step-size—though much easier than tuning penalty coefficients—remains a burden. Ideally, we would develop self-tuning strategies for updating dual variables, akin to Adam [43] in non-convex unconstrained minimization, that work reliably out of the box across a broad range of problems.

Generalization. When constraints depend on data, satisfying them on the training set does not guarantee feasibility on unseen samples. Despite PAC guarantees [13], constraint satisfaction often fails to generalize in practice [15, 38]. Cotter et al. [15] propose a regularization strategy that uses a held-out set to train the multipliers, while updating the model parameters using the main training set. This decouples the signal enforcing constraint satisfaction (the multipliers) from the signal responsible for achieving it (the model), helping prevent the model from overfitting to the constraints. Nonetheless, exploring additional regularization strategies remains an important research direction.

6 Conclusion

Recent efforts to build trustworthy and safe AI systems with accountability guarantees have led to a growing focus on constrained optimization in deep learning. We argue against penalized approaches for solving such problems, due to their unreliability and the difficulty of tuning them—especially in settings where domain knowledge prescribes desirable constraint levels, making the constrained formulation both natural and principled. We also identify open research directions for constrained deep learning methods which—building on their success in large-scale, non-convex problems—offer opportunities to further enhance their optimization adaptability and generalization performance.

Broader Impact Statement

This paper is pertinent to nearly all machine learning practitioners, not just a niche sub-community. This is because researchers and practitioners very frequently encounter tasks requiring the enforcement of multiple properties. Through this work, we aim to enhance the entire community's toolkit by providing more robust methods for addressing these problems, beyond the flawed penalized approach. Moving beyond the penalized approach offers significant advantages, such as:

- **Achieving better objective-constraint trade-offs**: This would enable practitioners to reach trade-offs that were previously out of reach. As a result, techniques or models that were once discarded for failing to meet desired requirements could become viable.

- **Reducing tuning overhead**: It would lessen the time and resources spent tuning penalty coefficients, which is a major burden on the machine learning community. This also lowers the computational footprint linked to repeated model training during hyperparameter searches.

- **Spurring further research**: Broader adoption of constrained methods would motivate more research into constrained learning. This would lead to the development of more effective and efficient algorithms, which would, in turn, further benefit the community.

Acknowledgements and disclosure of funding

This work was supported by the Canada CIFAR AI Chair program (Mila), the NSERC Discovery Grant RGPIN-2025-05123, by an unrestricted gift from Google, and by Samsung Electronics Co., Ldt. Simon Lacoste-Julien is a CIFAR Associate Fellow in the Learning in Machines & Brains program.

We thank Jose Gallego-Posada and Lucas Maes for their helpful discussions during the conceptualization of this work.

Many of the ideas presented in this work arose from discussions during the development of various research papers on constrained deep learning. This is especially true for Gallego-Posada et al. [31]. We would therefore like to thank our collaborators Yoshua Bengio, Juan Elenter, Akram Erraqabi, Jose Gallego-Posada, Golnoosh Farnadi, Ignacio Hounie, Alejandro Ribeiro, Rohan Sukumaran, Motahareh Sohrabi, and Tianyue (Helen) Zhang.

References

[1] M. Abadi et al. TensorFlow: A system for Large-Scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016. (Cit. on p. 2, 7)

[2] K. Arrow, L. Hurwicz, and H. Uzawa. *Studies in Linear and Non-linear Programming*. Stanford University Press, 1958. (Cit. on p. 2, 6, 7, 9)

[3] M. R. Bachute and J. M. Subhedar. Autonomous Driving Architectures. *Machine Learning with Applications*, 2021. (Cit. on p. 1)

[4] D. Bertsekas. On the Method of Multipliers for Convex Programming. *IEEE Transactions on Automatic Control*, 1975. (Cit. on p. 7, 9, 22)

[5] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2016. (Cit. on p. 3, 16, 22)

[6] D. P. Bertsekas. On the Goldstein-Levitin-Polyak Gradient Projection Method. *IEEE Transactions on automatic control*, 1976. (Cit. on p. 9)

[7] S. Bhatore et al. Machine learning techniques for credit risk evaluation. *Journal of Banking and Financial Technology*, 2020. (Cit. on p. 1)

[8] E. G. Birgin and J. M. Martínez. *Practical Augmented Lagrangian Methods for Constrained Optimization*. The SIAM series on Fundamentals of Algorithms, 2014. (Cit. on p. 3)

[9] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. (Cit. on p. 3, 9)

[10] J. Bradbury et al. JAX: composable transformations of Python+NumPy programs. `http://github.com/google/jax`, 2018. (Cit. on p. 7)

[11] P. Brouillard, S. Lachapelle, A. Lacoste, S. Lacoste-Julien, and A. Drouin. Differentiable Causal Discovery from Interventional Data. In *NeurIPS*, 2020. (Cit. on p. 1)

[12] D. Chakraborty, Y. LeCun, T. G. Rudner, and E. Learned-Miller. Improving Pre-trained Self-Supervised Embeddings Through Effective Entropy Maximization. In *AISTATS*, 2025. (Cit. on p. 1)

[13] L. Chamon and A. Ribeiro. Probably Approximately Correct Constrained Learning. In *NeurIPS*, 2020. (Cit. on p. 7, 10)

[14] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *NeurIPS*, 2016. (Cit. on p. 1)

[15] A. Cotter, M. Gupta, H. Jiang, N. Srebro, K. Sridharan, S. Wang, B. Woodworth, and S. You. Training Well-Generalizing Classifiers for Fairness Metrics and Other Data-Dependent Constraints. In *ICML*, 2019. (Cit. on p. 10)

[16] A. Cotter, H. Jiang, M. R. Gupta, S. Wang, T. Narayan, S. You, and K. Sridharan. Optimization with Non-Differentiable Constraints with Applications to Fairness, Recall, Churn, and Other Goals. *JMLR*, 2019. (Cit. on p. 1, 2, 3, 7, 8, 9, 19, 20)

[17] A. Cotter et al. TensorFlow Constrained Optimization (TFCO). `https://github.com/google-research/tensorflow_constrained_optimization`, 2019. (Cit. on p. 2, 7)

[18] G. E. Dahl, F. Schneider, Z. Nado, N. Agarwal, C. S. Sastry, P. Hennig, S. Medapati, R. Eschenhagen, P. Kasimbeg, D. Suo, J. Bae, J. Gilmer, A. L. Peirson, B. Khan, R. Anil, M. Rabbat, S. Krishnan, D. Snider, E. Amid, K. Chen, C. J. Maddison, R. Vasudev, M. Badura, A. Garg, and P. Mattson. Benchmarking Neural Network Training Algorithms. *arXiv preprint arXiv:2306.07179*, 2023. (Cit. on p. 4)

[19] J. Dai, X. Pan, R. Sun, J. Ji, X. Xu, M. Liu, Y. Wang, and Y. Yang. Safe RLHF: Safe Reinforcement Learning from Human Feedback. In *ICLR*, 2024. (Cit. on p. 1, 2, 3, 7)

[20] A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. In *NeurIPS*, 2014. (Cit. on p. 9)

[21] J. Degrave and I. Korshunova. Why machine learning algorithms are hard to tune and how to fix it. Engraved, blog: www.engraved.blog/why-machine-learning-algorithms-are-hard-to-tune/, 2021. (Cit. on p. 2, 4, 5, 18, 22)

[22] J. Degrave and I. Korshunova. How we can make machine learning algorithms tunable. Engraved, blog: www.engraved.blog/how-we-can-make-machine-learning-algorithms-tunable/, 2021. (Cit. on p. 2, 5, 18, 22)

[23] I. I. Dikin. Iterative Solution of Problems of Linear and Quadratic Programming. In *Doklady Akademii Nauk*. Russian Academy of Sciences, 1967. (Cit. on p. 9)

[24] M.-A. Dilhac et al. Montréal Declaration for a Responsible Development of Artificial Intelligence, 2018. (Cit. on p. 1)

[25] M. Dunion, T. McInroe, K. Luck, J. Hanna, and S. Albrecht. Conditional Mutual Information for Disentangled Representations in Reinforcement Learning. In *NeurIPS*, 2023. (Cit. on p. 1)

[26] M. Ehrgott. *Multicriteria Optimization*. Springer Science & Business Media, 2005. (Cit. on p. 1, 2, 4, 5)

[27] J. Elenter, N. NaderiAlizadeh, and A. Ribeiro. A Lagrangian Duality Approach to Active Learning. In *NeurIPS*, 2022. (Cit. on p. 2, 7)

[28] European Parliament. Artificial Intelligence Act. https://artificialintelligenceact.eu, 2024. (Cit. on p. 1)

[29] M. Frank and P. Wolfe. An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly*, 1956. (Cit. on p. 9)

[30] J. Gallego-Posada. Constrained Optimization for Machine Learning: Algorithms and Applications. *PhD Thesis, University of Montreal*, 2024. (Cit. on p. 2, 4, 6)

[31] J. Gallego-Posada, J. Ramirez, A. Erraqabi, Y. Bengio, and S. Lacoste-Julien. Controlled Sparsity via Constrained Optimization or: *How I Learned to Stop Tuning Penalties and Love Constraints*. In *NeurIPS*, 2022. (Cit. on p. 2, 3, 4, 6, 7, 11, 19)

[32] J. Gallego-Posada, J. Ramirez, M. Hashemizadeh, and S. Lacoste-Julien. Cooper: A Library for Constrained Optimization in Deep Learning. *arXiv preprint arXiv:2504.01212*, 2025. (Cit. on p. 2, 7, 18)

[33] J. Gauvin. A Necessary and Sufficient Regularity Condition to Have Bounded Multipliers in Nonconvex Programming. *Mathematical Programming*, 1977. (Cit. on p. 9)

[34] Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. (Cit. on p. 1)

[35] A. A. Goldstein. Convex Programming in Hilbert Space. *University of Washington*, 1964. (Cit. on p. 9)

[36] R. M. Gower, M. Schmidt, F. Bach, and P. Richtárik. Variance-Reduced Methods for Machine Learning. *Proceedings of the IEEE*, 2020. (Cit. on p. 9)

[37] S.-P. Han. A globally convergent method for nonlinear programming. *Journal of optimization theory and applications*, 1977. (Cit. on p. 9)

[38] M. Hashemizadeh, J. Ramirez, R. Sukumaran, G. Farnadi, S. Lacoste-Julien, and J. Gallego-Posada. Balancing Act: Constraining Disparate Impact in Sparse Models. In *ICLR*, 2024. (Cit. on p. 2, 7, 9, 10)

[39] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework . In *ICLR*, 2017. (Cit. on p. 1)

[40] I. Hounie, L. F. Chamon, and A. Ribeiro. Automatic Data Augmentation via Invariance-Constrained Learning. In *ICML*, 2023. (Cit. on p. 7)

[41] I. Hounie, J. Elenter, and A. Ribeiro. Neural Networks with Quantization Constraints. In *ICASSP*, 2023. (Cit. on p. 2, 7)

[42] I. Hounie, A. Ribeiro, and L. F. Chamon. Resilient Constrained Learning. In *NeurIPS*, 2024. (Cit. on p. 2, 9)

[43] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. (Cit. on p. 1, 4, 10, 19)

[44] A. Kumar, P. Sattigeri, and A. Balakrishnan. Variational Inference of Disentangled Latent Concepts from Unlabeled Observations. In *ICLR*, 2018. (Cit. on p. 1)

[45] J. Larson et al. Data and analysis for "Machine Bias". https://github.com/propublica/compas-analysis, 2016. (Cit. on p. 1)

[46] E. S. Levitin and B. T. Polyak. Constrained Minimization Methods. *USSR Computational mathematics and mathematical physics*, 1966. (Cit. on p. 9)

[47] T. Lin, C. Jin, and M. I. Jordan. Two-Timescale Gradient Descent Ascent Algorithms for Nonconvex Minimax Optimization. *JMLR*, 2025. (Cit. on p. 7)

[48] C. Louizos, M. Welling, and D. P. Kingma. Learning Sparse Neural Networks through L_0 Regularization. In *ICLR*, 2018. (Cit. on p. 1, 19)

[49] O. L. Mangasarian and S. Fromovitz. The Fritz John Necessary Optimality Conditions in the Presence of Equality and Inequality Constraints. *Journal of Mathematical Analysis and Applications*, 1967. (Cit. on p. 3, 9)

[50] K. Miettinen. *Nonlinear Multiobjective Optimization*. Springer Science & Business Media, 1999. (Cit. on p. 4)

[51] H. Narasimhan, A. Cotter, Y. Zhou, S. Wang, and W. Guo. Approximate Heavily-Constrained Learning with Lagrange Multiplier Models. In *NeurIPS*, 2020. (Cit. on p. 2, 7, 9)

[52] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006. (Cit. on p. 9, 22)

[53] OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023. (Cit. on p. 1)

[54] A. Paszke et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. (Cit. on p. 2, 7, 18)

[55] J. C. Platt and A. H. Barr. Constrained Differential Optimization. In *NeurIPS*, 1987. (Cit. on p. 4, 10, 22)

[56] M. J. Powell. The convergence of variable metric methods for nonlinearly constrained optimization calculations. In *Nonlinear programming 3*. Elsevier, 1978. (Cit. on p. 9)

[57] J. Ramirez, I. Hounie, J. Elenter, J. Gallego-Posada, M. Hashemizadeh, A. Ribeiro, and S. Lacoste-Julien. Feasible Learning. In *AISTATS*, 2025. (Cit. on p. 3, 7)

[58] A. Robey, L. Chamon, G. J. Pappas, H. Hassani, and A. Ribeiro. Adversarial Robustness with Semi-Infinite Constrained Learning. In *NeurIPS*, 2021. (Cit. on p. 2, 7)

[59] E. Shi, L. Kong, and B. Jiang. Deep Fair Learning: A Unified Framework for Fine-tuning Representations with Sufficient Networks. *arXiv preprint arXiv:2504.06470*, 2025. (Cit. on p. 1)

[60] M. Sohrabi, J. Ramirez, T. H. Zhang, S. Lacoste-Julien, and J. Gallego-Posada. On PI Controllers for Updating Lagrange Multipliers in Constrained Optimization. In *ICML*, 2024. (Cit. on p. 7, 8, 10, 18, 22)

[61] A. Stooke, J. Achiam, and P. Abbeel. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. In *ICML*, 2020. (Cit. on p. 2, 7, 10, 22)

[62] R. B. Wilson. A simplicial algorithm for concave programming. *PhD Thesis, Graduate School of Bussiness Administration*, 1963. (Cit. on p. 9)

[63] G. Zhang, Y. Wang, L. Lessard, and R. B. Grosse. Near-optimal Local Convergence of Alternating Gradient Descent-Ascent for Minimax Optimization. In *AISTATS*, 2022. (Cit. on p. 7)

[64] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *ICCV*, 2017. (Cit. on p. 1)

[65] G. Zoutendijk. *Methods of Feasible Directions: A Study in Linear and Non-linear Programming*. Elsevier Publishing Company, 1960. (Cit. on p. 9)

Appendix

A Proofs

Proof of Proposition 1. Since $\boldsymbol{x}^*$ is a regular constrained minimizer with corresponding optimal Lagrange multipliers $\boldsymbol{\lambda}^* \succeq \mathbf{0}$ and $\boldsymbol{\mu}^*$, we can invoke the KKT conditions, which state:

$$\nabla f(\boldsymbol{x}^*) + \boldsymbol{\lambda}^{*\top} \nabla \boldsymbol{g}(\boldsymbol{x}^*) + \boldsymbol{\mu}^{*\top} \nabla \boldsymbol{h}(\boldsymbol{x}^*) = \mathbf{0} \quad \text{(Stationarity)}. \tag{11}$$

Choose $\boldsymbol{c_g} = \boldsymbol{\lambda}^*$ and $\boldsymbol{c_h} = \boldsymbol{\mu}^*$. The stationarity condition then implies that $\boldsymbol{x}^*$ is a critical point of the penalized objective

$$\mathcal{P}(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{c_g}^\top \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{c_h}^\top \boldsymbol{h}(\boldsymbol{x}).$$

The objective $\mathcal{P}$ is convex in $\boldsymbol{x}$, as it is the sum of the convex function f, the convex function $\boldsymbol{c_g}^\top \boldsymbol{g}$, and the affine function $\boldsymbol{c_h}^\top \boldsymbol{h}$. For convex functions, any critical point is a global minimizer. Hence,

$$\boldsymbol{x}^* \in \operatorname*{argmin}_{\boldsymbol{x} \in \mathcal{X}} \mathcal{P}(\boldsymbol{x}).$$

□

Proof of Proposition 2. We first show that the point $(x^*, y^*, \lambda^*) = \left(\arcsin(\epsilon),\, 0,\, \epsilon/\sqrt{1-\epsilon^2}\right)$ satisfies the first-order necessary conditions for constrained optimality.

The Lagrangian of Example 1 is given by

$$\mathcal{L}(x, y, \lambda) = (1 + y^2)\cos(x) + \lambda\left[(1 + y^2)\sin(x) - \epsilon\right], \tag{12}$$

where $\lambda \geq 0$ is the Lagrange multiplier.

We verify the KKT conditions:

1. *Primal feasibility*. We have $g(x^*, y^*) = \sin(\arcsin(\epsilon)) = \epsilon \leq \epsilon$.
2. *Stationarity*. The gradients of $\mathcal{L}$ with respect to the primal variables vanish at (x^*, y^*, λ^*):
$$\frac{\partial \mathcal{L}}{\partial x} = -(1 + y^2)\sin(x) + \lambda(1 + y^2)\cos(x)\Big|_{(x^*, y^*, \lambda^*)} = -\epsilon + \lambda^* \sqrt{1 - \epsilon^2} = 0, \tag{13}$$
$$\frac{\partial \mathcal{L}}{\partial y} = 2y\cos(x) + 2\lambda y \sin(x)\Big|_{(x^*, y^*, \lambda^*)} = 0. \tag{14}$$
3. *Complementary slackness*. We have $\lambda^*\left[g(x^*, y^*) - \epsilon\right] = 0$. Since $\lambda^* > 0$, *strict* complementary slackness holds.

We now verify that (x^*, y^*, λ^*) satisfies the second-order sufficient conditions for optimality [5, Prop. 4.3.2]. Consider the Hessian of $\mathcal{L}$ with respect to (x, y):

$$\nabla^2_{x,y}\mathcal{L} = \begin{pmatrix} -(1 + y^2)\left(\cos(x) + \lambda \sin(x)\right) & -2y\sin(x) + 2\lambda y \cos(x) \\ -2y\sin(x) + 2\lambda y\cos(x) & 2\cos(x) + 2\lambda \sin(x) \end{pmatrix}. \tag{15}$$

At (x^*, y^*, λ^*), this becomes:

$$\nabla^2_{x,y}\mathcal{L}(x^*, y^*, \lambda^*) = \frac{1}{\sqrt{1-\epsilon^2}} \begin{pmatrix} -1 & 0 \\ 0 & 2 \end{pmatrix}. \tag{16}$$

To satisfy the second-order sufficient conditions, we must ensure that the Hessian is positive definite on the null space of constraints at (x^*, y^*)—that is, the set of feasible first-order directions that preserve the set of active constraints.

The gradient of the active constraint is:

$$\nabla g(x^*, y^*) = \left((1 + y^2)\cos(x),\, 2y\sin(x)\right)\Big|_{(x^*, y^*)} = \left(\sqrt{1-\epsilon^2},\, 0\right). \tag{17}$$

Thus, the null space is:

$$\mathcal{N} = \left\{(0, \delta y) \,\middle|\, \delta y \in \mathbb{R}\right\}. \tag{18}$$

For any $\mathbf{v} = (0, \delta y) \in \mathcal{N}$:

$$\mathbf{v}^\top \nabla^2_{x,y} \mathcal{L}\, \mathbf{v} = \frac{2(\delta y)^2}{\sqrt{1-\epsilon^2}} > 0 \quad \text{for all } \delta y \neq 0. \tag{19}$$

Since the Hessian is (strictly) positive definite over $\mathcal{N}$, and (x^*, y^*) satisfies the first-order conditions along with strict complementary slackness, it follows that (x^*, y^*) is a strict local constrained minimizer of Example 1.

Substituting the solution, we find:

$$f\left(\arcsin(\epsilon), 0\right) = \sqrt{1-\epsilon^2}, \quad g\left(\arcsin(\epsilon), 0\right) = \epsilon.$$

To verify global optimality, we evaluate the objective and constraint at the boundaries of the domain:

- At $x = \pi/2$, the constraint evaluates to $g(\pi/2, y) = 1 + y^2 > \epsilon$, which is infeasible.
- At $x = 0$, we have $g(0, y) = 0 \leq \epsilon$, and the objective becomes $f(0, y) = 1 + y^2$, which is minimized at $y = 0$, giving $f(0, 0) = 1$.

However, $f\left(\arcsin(\epsilon), 0\right) = \sqrt{1-\epsilon^2} < 1 = f(0, 0)$, so we conclude that $(x^*, y^*) = (\arcsin(\epsilon), 0)$ is the global constrained minimizer of Example 1.

□

Proof of Proposition 3. We begin by factorizing the penalized objective as:

$$\mathcal{P}_c(x, y) = (1 + y^2)(\cos(x) + c\sin(x)).$$

Since $\cos(x) + c\sin(x) \geq 0$ for $x \in [0, \pi/2]$ and $c > 0$, the minimum with respect to y is attained at $y^* = 0$. Let $h(x) = \mathcal{P}_c(x, 0) = \cos(x) + c\sin(x)$, and compute its derivatives:

$$h'(x) = -\sin(x) + c\cos(x), \tag{20}$$

$$h''(x) = -\cos(x) - c\sin(x) < 0, \quad \forall x \in [0, \pi/2],\ c \geq 0. \tag{21}$$

Thus, $h(x)$ is strictly concave on the interval. While its maximum occurs at the critical point $x = \arctan(c)$, we are minimizing $\mathcal{P}_c$, so the minimizer must lie at one of the boundaries.

- For $c < 1$, we have $h(\pi/2) = c < 1 = h(0)$, so the minimum is at $x^* = \pi/2$.
- For $c > 1$, we have $h(0) = 1 < c = h(\pi/2)$, so the minimum is at $x^* = 0$.
- When $c = 1$, both endpoints yield the same value: $h(0) = h(\pi/2) = 1$.

Evaluating the constraint and objective at the endpoints, we find: $g(0, 0) = 0 \leq \epsilon$, $f(0, 0) = 1$; and $g(\pi/2, 0) = 1 > \epsilon$, $f(\pi/2, 0) = 0$. □

B Experimental Details

Our implementations use PyTorch [54] and the Cooper library for constrained deep learning [32]. Our code is available at: `https://github.com/merajhashemi/constraints-vs-penalties`.

B.1 Example 1

Figure 3 illustrates the Pareto front of non-dominated solutions to Example 1. Varying x changes the angle (position) of a point along the Pareto front, while varying y controls its distance from the front. Intuitively, this shows that all optimal solutions occur at $y = 0$. Moreover, imposing a constraint of the form $g(x, y) \leq \epsilon$ restricts the optimization of f to the region below the line $g = \epsilon$; since minimizing f with respect to x requires increasing f, this constraint selects a specific solution on the front where $g = \epsilon$. The choice of ϵ therefore determines the corresponding optimal value of x, that is, the position along the front.

Intuitively, the concavity of this Pareto front—which is tied to the concavity of the penalized function for this problem—causes gradient descent on the penalized objective to fall to the corners of the front. In contrast, a convex Pareto front—which would arise under the convexity assumptions outlined in Prop. 1—would allow gradient descent to reach intermediate points, enabling proper exploration of the trade-offs between f and g by varying ϵ.

See Degrave and Korshunova [21, 22] for animations illustrating the optimization dynamics of ① gradient descent on the penalized formulation, which falls to the corners; ② gradient descent-ascent on the Lagrangian, which oscillates without converging; ③ and gradient descent-ascent on the Augmented Lagrangian, which does converge.

Hyper-parameters. We use the Lagrangian approach to solve Example 1, applying gradient descent with a step-size of 0.01 as the primal optimizer and νPI [60]—as implemented in Cooper [32]—as the dual optimizer, with a step-size of 0.3, damping coefficient $\kappa_p = 40$, and $\nu = 0$. To enforce $x \in [0, \pi/2]$, we apply a projection after every primal update. Table 1 in Appendix C.1 presents results after 10,000 training iterations. We elaborate on this choice of dual optimizer in Appendix D—it is necessary for convergence, as standard gradient ascent would otherwise lead to undamped oscillations.

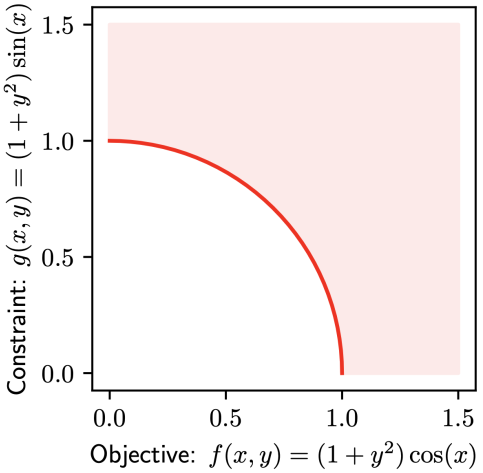

Figure 3: The Pareto front of non-dominated solutions between the objective f and constraint g in Example 1. The shaded region corresponds to the image of the map $(x, y) \mapsto (f, g)$, cropped to $[0, 1.5]^2$. The top-left corner of the Pareto front is achieved at $x = \pi/2$, $y = 0$; the bottom-right at $x = 0$, $y = 0$. More generally, x determines the angle (position) along the Pareto front, while y controls the distance from it.

B.2 Sparsity Constraints

Louizos et al. [48] propose a model reparameterization that enables applying L_0-norm regularization to the weights via stochastic gates $\boldsymbol{z} \in [0, 1]$ that follow a Hard Concrete distribution parameterized by $\boldsymbol{\phi}$. This is formulated in a penalized fashion as follows:

$$\min_{\boldsymbol{w}, \boldsymbol{\phi} \in \mathbb{R}^d} \mathbb{E}_{\boldsymbol{z}|\boldsymbol{\phi}} \left[L(\boldsymbol{w} \odot \boldsymbol{z} \,|\, \mathcal{D}) \right] + c \cdot \mathbb{E}_{\boldsymbol{z}|\boldsymbol{\phi}}[\|\boldsymbol{z}\|_0], \tag{22}$$

where $\boldsymbol{w}$ are the model parameters, L is the task loss, $\mathcal{D}$ is the dataset, $\| \cdot \|_0$ denotes the L_0 norm, and $c > 0$ is a penalty coefficient.

To illustrate the tunability advantages of constrained approaches over the penalized approach, we replicate the bisection search experiment from Figure 5(b), Appendix E of Gallego-Posada et al. [31]. We use their MNIST classification setup with an MLP (300–100 hidden units), as shown in Fig. 1. To achieve 50% global sparsity, we perform a log-scale bisection search over penalty coefficients. The corresponding results, including model training accuracy, are reported in Table 2 in Appendix C.2.

Akin to Gallego-Posada et al. [31], we set the temperature of the stochastic gate distribution to $2/3$ and use a stretching interval of $[-0.1, 1.1]$. The droprate init is set to 0.01, resulting in a fully dense model at the start of training. We use Adam [43] with a step-size of 0.001 for optimizing both the model and gate parameters. Training is performed for 150 epochs with a batch size of 256.

B.3 Rate Constraints

We consider a linear binary classification problem , where the model is constrained to predict class 0 for at least 70% of the training examples. The resulting optimization problem is:

$$\min_{w,b} \; \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\ell(w^\top x + b, y) \right] \quad \text{s.t.} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathbf{1}_{w^\top x + b < 0} \right] \geq 0.7, \tag{23}$$

where ℓ denotes the cross-entropy loss, w and b are the weights and bias of the linear model, (x, y) are input-label pairs drawn from the data distribution $\mathcal{D}$, and σ is the sigmoid function.

This rate-constrained setup matches the experiment in Figure 2 of Cotter et al. [16], originally designed to showcase the effectiveness of their method for handling non-differentiable constraints. Here, we repurpose it to highlight the tunability advantages of the Lagrangian approach over the penalized one.

Since the constraint is not differentiable with respect to the model parameters, we follow Cotter et al. [16] and use a differentiable surrogate to update the model parameters, while still using the true non-differentiable constraint to update the multipliers.[9] As a surrogate, we use the hinge loss:

$$P(\hat{y} = 0) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[1 - \sigma(w^\top x + b) \right] \geq 0.7, \tag{24}$$

which represents the expected proportion of inputs predicted as class 0.

To construct the penalized formulation of Eq. (23), we penalize the objective with the surrogate term:

$$\min_{w,b} \; \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\ell(w^\top x + b, y) \right] + c \cdot P(\hat{y} = 0), \tag{25}$$

where $c > 0$ is a penalty coefficient. Note that it is not possible to use the non-differentiable constraint with the penalized formulation, as gradient-based optimization requires a differentiable objective.

In Figure 2, we present an ablation over the dual step-size when solving Eq. (23) using the Lagrangian approach with proxy constraints [16], and over the penalty coefficient when solving the corresponding penalized formulation, Eq. (25). The results illustrate that tuning dual step-sizes in the Lagrangian approach is significantly easier than tuning penalty coefficients. Corresponding tables with the same results, including accuracy measurements, are provided in Tables 3 and 4 in Appendix C.3.

Hyper-parameters. For the constrained approach, we use gradient descent–ascent with a primal step-size of 2×10^{-2}, training for 10,000 iterations. The dual step-size is ablated over several orders of magnitude. For the penalized approach, we use the same primal optimization pipeline—gradient descent with a step-size of 2×10^{-2}—and ablate over penalty coefficients using the same set of values as for the dual step-size. The dataset is a 2-dimensional, linearly separable binary mixture of Gaussians with 100 datapoints per class. Training is done using full-batch optimization.

[9] While possible, using the Lagrangian formulation with a constraint on the surrogate—$P(\hat{y} = 0) \geq 0.7$—does not, as noted by Cotter et al. [16], yield solutions that satisfy the original, non-differentiable constraint. This highlights the strength of their proxy constraints approach.

C Comprehensive Experimental Results

This appendix section provides comprehensive table results to compliment the experiments of the paper: on Example 1, and Figures 1 and 2 in §3.1 and §3.2, respectively.

C.1 Concave 2D Problem

Table 1 reports results for solving Example 1 using the Lagrangian approach (see Appendix B.1 for experimental details). Across all constraint levels, the method recovers the true constrained minimizer $(\arcsin(\epsilon), 0)$ (see Prop. 2), up to negligible numerical error. This is in contrast with the penalized approach, whose minimizers lie at either $x^* = 0$ or $x^* = \pi/2$—which are, respectively, suboptimal or infeasible for all choices of ϵ (Prop. 3).

Table 1: Solutions to Example 1 recovered using the Lagrangian approach. Across all constraint levels ϵ, the method consistently recovers the constrained optimizer $(x^*, y^*) = (\arcsin(\epsilon), 0)$.

ϵ	$x^* = \arcsin(\epsilon)$	x at convergence	y at convergence
0.1	1.00×10^{-1}	1.00×10^{-1}	5.89×10^{-10}
0.2	2.01×10^{-1}	2.01×10^{-1}	4.47×10^{-10}
0.3	3.05×10^{-1}	3.05×10^{-1}	2.70×10^{-10}
0.4	4.12×10^{-1}	4.12×10^{-1}	1.22×10^{-10}
0.5	5.24×10^{-1}	5.24×10^{-1}	3.70×10^{-11}
0.6	6.44×10^{-1}	6.44×10^{-1}	6.04×10^{-12}
0.7	7.75×10^{-1}	7.75×10^{-1}	3.32×10^{-13}
0.8	9.27×10^{-1}	9.27×10^{-1}	1.62×10^{-15}

C.2 Sparsity Constraints

Table 2 reports the same results as the bisection search experiment in Figure 1, including the training accuracy of each model at convergence. Task and experimental details are provided in Appendix B.2. As shown, for example, by the result with $c = 1$, which yields a feasible solution with 25.3% model density (74.7% sparsity), overshooting the constraint reduces performance due to unnecessary loss of model capacity. The solution should instead lie closer to the constraint boundary, as with the final coefficient choice of 2.21×10^{-1}.

Table 2: Sparsity-constrained neural network training using the penalized approach, targeting approximately 50% model density. The penalty coefficient is selected via a log-scale bisection search. This table complements Figure 1.

Iteration #	Penalty coef. c	Model density (%)	Acc. (%)
0	1.00×10^{-3}	84.5	99.97
0	1	25.3	99.97
1	3.16×10^{-2}	68.1	99.96
2	1.78×10^{-1}	54.2	100.00
3	4.22×10^{-1}	39.0	100.00
4	2.74×10^{-1}	46.9	99.99
5	2.21×10^{-1}	50.5	99.99

C.3 Rate Constraints

Tables 3 and 4 report the same results as the rate-constrained classification experiment in Figure 2, now including the training accuracy of each model at convergence. The task and experimental choices are described in Appendix B.3. Most penalty coefficients result in collapsed solutions—either optimizing only for accuracy and yielding a 50% classification rate, or focusing entirely on the penalty and classifying all inputs as class 0. As in the sparsity constrained experiment (Table 2), overshooting the constraint degrades performance due to excessive emphasis on the penalty, which conflicts with the objective. In contrast, the Lagrangian approach with proxy constraints [16] recovers the desired solution for most of the considered dual step-sizes.

Table 3: Rate-constrained linear classification using the penalized approach, targeting 70% in class 0. Only $c = 2.15$ achieves a non-collapsed solution (76% class 0), while other coefficients either ignore the constraint (50% class 0 for $c < 2.15$) or over-satisfy it (100% class 0 for $c > 2.15$), sacrificing accuracy.

Penalty coef.	Class 0 Percentage (%)	Accuracy (%)
0	49.75	99.75
1.00×10^{-4}	49.75	99.75
2.15×10^{-4}	49.75	99.75
4.60×10^{-4}	49.75	99.75
1.00×10^{-3}	49.75	99.75
2.15×10^{-3}	49.75	99.75
4.60×10^{-3}	49.75	99.75
1.00×10^{-2}	49.75	99.75
2.15×10^{-2}	49.75	99.75
4.60×10^{-2}	49.75	99.75
1.00×10^{-1}	49.75	99.75
2.15×10^{-1}	49.75	99.75
4.60×10^{-1}	49.75	99.75
1.00×10^{0}	50.50	99.50
2.15×10^{0}	76.00	74.00
4.60×10^{0}	99.75	50.25
1.00×10^{1}	100.00	50.00
2.15×10^{1}	100.00	50.00
4.60×10^{1}	100.00	50.00
1.00×10^{2}	100.00	50.00

Table 4: Rate-constrained linear classification using the Lagrangian approach, targeting 70% in class 0. The constrained approach achieves the target class prediction rate across a wide range of dual step-sizes. Training accuracy stabilizes at 80% once the rate constraint is met, demonstrating robust feasibility-performance trade-offs.

Dual Step-size	Class 0 Percentage (%)	Accuracy (%)
0	49.75	99.75
1.00×10^{-4}	49.75	99.75
2.15×10^{-4}	49.75	99.75
4.60×10^{-4}	50.00	100.00
1.00×10^{-3}	60.00	90.00
2.15×10^{-3}	70.00	80.00
4.60×10^{-3}	70.00	80.00
1.00×10^{-2}	70.00	80.00
2.15×10^{-2}	70.00	80.00
4.60×10^{-2}	70.00	80.00
1.00×10^{-1}	70.00	80.00
2.15×10^{-1}	70.00	80.00
4.60×10^{-1}	70.00	80.00
1.00×10^{0}	70.00	80.00
2.15×10^{0}	69.75	80.25
4.60×10^{0}	70.00	80.00
1.00×10^{1}	70.00	80.00
2.15×10^{1}	70.00	80.00
4.60×10^{1}	69.75	80.25
1.00×10^{2}	69.75	80.25

D On the Augmented Lagrangian Method

As discussed in Degrave and Korshunova [21, 22]—and formally analyzed by Platt and Barr [55] from a dynamical-systems perspective—standard gradient descent–ascent on the Lagrangian fails to solve Example 1. The updates exhibit undamped oscillations around the optimal solution and its associated Lagrange multiplier, driven by the concavity of the Lagrangian with respect to x (just as the penalized objective is concave in x for any penalty coefficient $c > 0$). Intuitively, these dynamics reflect a fundamental tension: minimizing the Lagrangian with respect to x pushes toward the domain boundaries—mimicking the behavior of the penalized approach—while updates to the multiplier seek to enforce the constraint. As these two forces act out of phase, their interplay results in persistent oscillations, as also noted in Degrave and Korshunova [21, 22].

Degrave and Korshunova [22], Platt and Barr [55] instead propose optimizing the Augmented Lagrangian [4]; we briefly explain in this section why it resolves the issue. However, to keep the presentation of the main paper streamlined and focused on the vanilla Lagrangian approach, we adopt a PI controller to update the dual variables [60, 61], which still recovers the correct solution.

The Augmented Lagrangian function includes a quadratic penalty for constraint violations, in addition to the linear penalty used in the standard Lagrangian:

$$\mathcal{L}_c(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}) \triangleq f(x) + \frac{1}{2c}\left[\|\boldsymbol{\mu}+c\,\boldsymbol{h}(\boldsymbol{x})\|_2^2 - \|\boldsymbol{\mu}\|_2^2 + \|\max\{0,\boldsymbol{\lambda}+c\,\boldsymbol{g}(\boldsymbol{x})\}\|_2^2 - \|\boldsymbol{\lambda}\|_2^2\right] \tag{26}$$

$$= f(x) + \boldsymbol{\mu}^\top \boldsymbol{h}(\boldsymbol{x}) + \frac{c}{2}\,\|\boldsymbol{h}(\boldsymbol{x})\|_2^2 + \sum_{i=1}^{m} \begin{cases} \lambda_i g_i(\boldsymbol{x}) + \frac{c}{2}\,g_i^2(\boldsymbol{x}), & \text{if } \lambda_i + c\,g_i(\boldsymbol{x}) \geq 0 \\ -\lambda_i^2/2c, & \text{otherwise} \end{cases}, \tag{27}$$

where $c > 0$ is a penalty coefficient. Violations of equality constraints are penalized both linearly, through the term involving the multiplier $\boldsymbol{\mu}$, and quadratically, via a penalty term with coefficient c. Inequality violations are similarly penalized linearly and quadratically, but only when $\lambda_i + c\,g_i(\boldsymbol{x}) \geq 0$; otherwise, no penalty is applied.

As in the standard Lagrangian approach, the Augmented Lagrangian method seeks a stationary point of the Augmented Lagrangian function:

$$\min_{\boldsymbol{x}\in\mathcal{X}} \max_{\boldsymbol{\lambda}\succeq\boldsymbol{0},\boldsymbol{\mu}} \mathcal{L}_c(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}). \tag{28}$$

It is straightforward to show that the Augmented Lagrangian shares the same set of stationary points as the original Lagrangian.[10] Thus, any stationary point of the $\mathcal{L}_c$ satisfying the second-order necessary conditions [5, Prop. 4.3.2] corresponds to a solution of the constrained problem. Therefore, finding stationary points of Eq. (26) remains a principled approach to constrained optimization.

Crucially, for a local solution $\boldsymbol{x}^*$ satisfying the second-order sufficient conditions and the Linear Independence Constraint Qualification [5, Prop. 4.3.8], there exists a sufficiently large c such that the Augmented Lagrangian becomes *strictly convex* at $\boldsymbol{x}^*$, regardless of whether $\mathcal{L}$ is convex at $\boldsymbol{x}^*$ [52, Thm. 17.5]. In practice, this convexification ensures that gradient descent–ascent locally converges on the Augmented Lagrangian, even when it oscillates or fails to converge on the standard Lagrangian.

[10] This follows because dual stationarity implies feasibility and complementary slackness, causing the augmented terms to vanish when evaluating the primal gradient—ultimately yielding the same stationarity condition as the standard Lagrangian; namely, that the objective gradient is a linear combination of the constraint gradients weighted by the multipliers.