

A Roadmap for Enforcing Hard Constraints in AI Models

A Tutorial on Lagrangian and Repair Methods

The Critical Need for Hard Constraints

Standard AI/ML models are trained to minimize a loss function, but they offer no guarantees that their outputs will satisfy critical operational constraints. This is a major barrier to deployment in high-stakes domains:

- **Robotics:** A robot arm's trajectory must remain within its joint limits and avoid collisions.
- **Power Grids:** Power generation schedules must exactly match demand to prevent blackouts.
- **Logistics:** A delivery route must respect vehicle capacity and driver work-hour limits.

In these applications, constraint violations are not just suboptimal; they are failures. Our goal is to bridge the gap between AI-driven decisions and the strict requirements of the physical world.



Two Paths to Guaranteed Constraints

We can frame the problem of enforcing constraints as a choice between two fundamental strategies. Both leverage classical optimization theory as a foundation.

AI Model Output

Path 1: Pricing Constraints (Lagrangian Methods)

The Idea: Relax hard constraints and incorporate them as tunable penalties in the training objective. We find the right “price” for each constraint violation.

The Story: An economic interpretation of constraints.

Path 2: Projecting Solutions (Repair Methods)

The Idea: Allow a model to produce an unconstrained output, then use an explicit “repair” step to project it onto the feasible set.

The Story: A geometric interpretation of feasibility.

The Standard Form and Our Foundation

We consider a general optimization problem where the decision variable x is the output of an AI model, $x = M_\theta(\text{input})$:

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\ & && h_i(x) = 0, \quad i = 1, \dots, p \end{aligned}$$

- $f_0(x)$ is the objective function we wish to minimize (e.g., tracking error, cost).
- $f_i(x)$ are the inequality constraints.
- $h_i(x)$ are the equality constraints.

The feasible set is $C = \{x \mid f_i(x) \leq 0, h_i(x) = 0\}$. Throughout this tutorial, we will use the notation and concepts from Boyd & Vandenberghe's Convex Optimization, which we assume as a shared foundation.

Path 1: The Lagrangian “Pricing” Strategy

The core idea of Lagrangian relaxation is to transform a constrained problem into an unconstrained one by associating a “price” or penalty with each constraint. Let’s revisit the Lagrangian from B&V, Chapter 5:

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x).$$

- $\lambda_i \geq 0$ and ν_i are the Lagrange multipliers or dual variables.
- We can interpret λ_i as the price per unit violation of the constraint $f_i(x) \leq 0$.
- The goal is to find prices (λ^*, ν^*) so that the minimizer of $L(x, \lambda^*, \nu^*)$ is also the solution to our original constrained problem.

The Lagrange dual function gives the optimal value of this “priced” problem for a fixed set of prices:

$$g(\lambda, \nu) = \inf_x L(x, \lambda, \nu)$$

For any $\lambda \geq 0$, $g(\lambda, \nu)$ provides a lower bound on the optimal value p^* of the original problem (Weak Duality).

Application: Integrating the Lagrangian into the Learning Objective

To apply this to an AI model M_θ that outputs x , we can define the training loss using the Lagrangian. The key question is how to handle the dual variables (λ, ν) .

Strategy 1: Fixed Hyperparameter Prices

The simplest approach is to treat λ and ν as fixed, user-defined hyperparameters.

$$\mathcal{L}(\theta) = f_0(M_\theta(z)) + \lambda^T f(M_\theta(z)) + \nu^T h(M_\theta(z))$$

- **Pros:** Easy to implement.
- **Cons:** Extremely difficult to tune the prices λ, ν to achieve feasibility. No guarantee of satisfaction.

Strategy 2: Learning the Prices via Dual Ascent (Fioretto et al.)

A more powerful approach is to learn the prices simultaneously with the model parameters. This mimics the saddle-point formulation of the dual problem. The training process becomes a two-player game:

1. **Primal Step:** Update model parameters θ to minimize \mathcal{L} for fixed (λ, ν) :

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(M_\theta(z), \lambda, \nu)$$

2. **Dual Step:** Update dual variables λ, ν to maximize \mathcal{L} (i.e., raise the price on violated constraints):

$$\begin{aligned}\lambda_i &\leftarrow [\lambda_i + \beta f_i(M_\theta(z))]_+ \\ \nu_i &\leftarrow \nu_i + \beta h_i(M_\theta(z))\end{aligned}$$

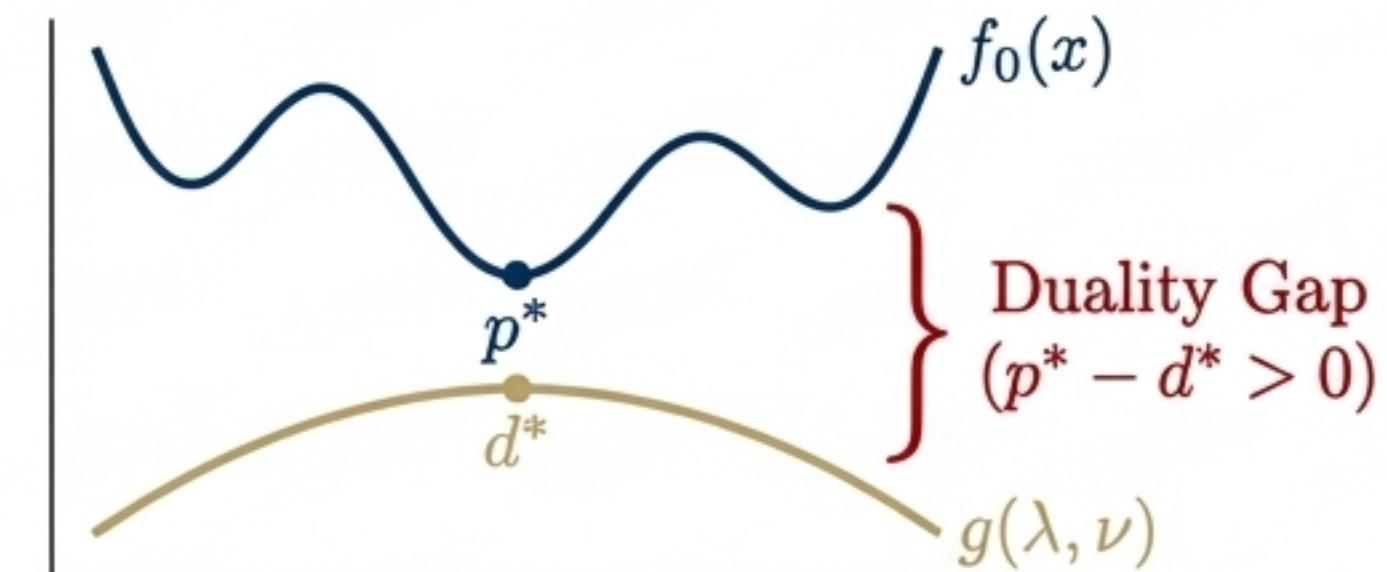
Lagrangian Methods: Success and Failure

When it Succeeds

- **Convex Problems with Strong Duality:** If the original problem is convex and satisfies a constraint qualification (e.g., Slater's condition), then strong duality holds ($p^* = d^*$). In this case, finding the optimal dual variables (λ^*, ν^*) and then minimizing $L(x, \lambda^*, \nu^*)$ solves the primal problem. The method is powerful as $\inf_x L(x, \dots)$ may be an easy unconstrained problem.
- **Provides a Bound:** Weak duality always holds, meaning $g(\lambda, \nu)$ is always a valid lower bound on the true optimal value p^* .

When it Fails (Guarantees are Lost)

- **Non-Convexity and the Duality Gap:** For non-convex problems, such as training a deep neural network, a non-zero duality gap ($p^* > d^*$) is common. This means that even for the optimal dual variables (λ^*, ν^*) , the minimizer of the Lagrangian, $x_L = \operatorname{argmin}_x L(x, \lambda^*, \nu^*)$, is **not guaranteed to be feasible** for the original problem.
- **Result:** The method devolves into a sophisticated penalty method. It will push solutions *towards* feasibility, but it does not provide a **hard guarantee**. Violations, though often small, can persist.



Path 2: The Repair ‘Projection’ Strategy

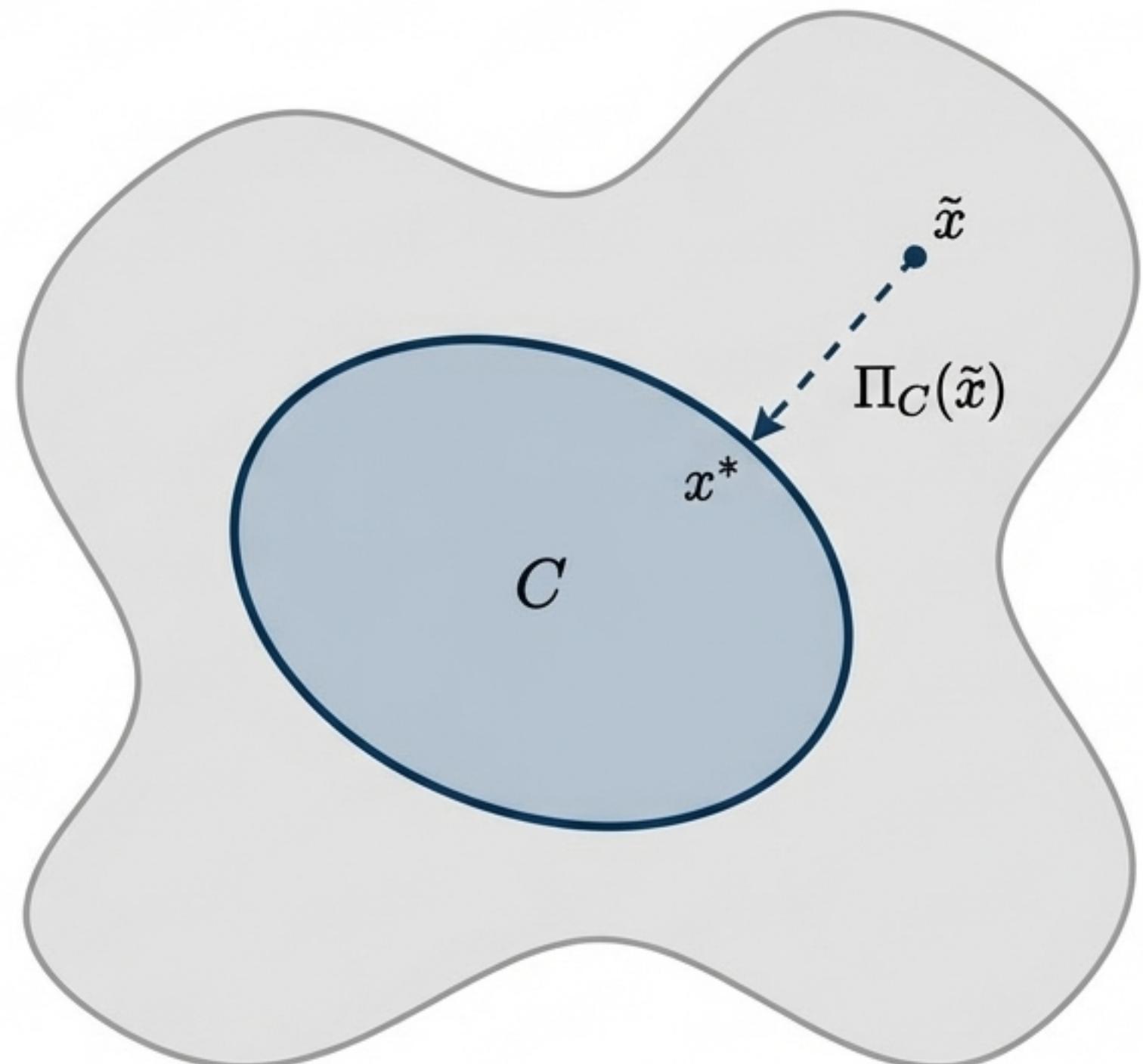
The repair method takes a different philosophical approach:

1. Train a model M_θ to produce a ‘best guess’ solution $\tilde{x} = M_\theta(z)$, ignoring the constraints.
2. In a separate, deterministic step, ‘repair’ \tilde{x} by finding the closest point to it that lies within the feasible set C .

This repair step is a **projection**. The final, guaranteed-feasible output x^* is:

$$x^* = \Pi_C(\tilde{x}) = \operatorname{argmin}_{x \in C} \|x - \tilde{x}\|_2.$$

- $C = \{x \mid f_i(x) \leq 0, h_i(x) = 0\}$ is the feasible set.
- This approach guarantees feasibility by construction.
- The key challenge is how to train the model M_θ end-to-end *through* this projection operator.



Application: The Differentiable Repair Layer

To enable end-to-end training, the projection operator $\Pi_C(\cdot)$ must be differentiable. The model's loss depends on the final, projected output:

$$\mathcal{L}(\theta) = f_0(x^*) = f_0(\Pi_C(M_\theta(z)))$$

To compute the gradient $\partial \mathcal{L}/\partial \theta$, the chain rule requires the Jacobian of the projection, $\partial \Pi_C(\tilde{x})/\partial \tilde{x}$.

How can we differentiate through `argmin`?

- The projection $x^* = \Pi_C(\tilde{x})$ is the solution to a convex optimization problem.
- We can use the KKT conditions that define x^* . For a convex set C , these conditions implicitly define x^* as a function of \tilde{x} .
- Using the **Implicit Function Theorem** on the KKT conditions, we can compute the Jacobian $\partial x^*/\partial \tilde{x}$. This is particularly straightforward when the projection is a Quadratic Program (QP).

This technique allows gradients to flow back from the final objective $f_0(x^*)$ through the repair layer to the underlying model M_θ , as demonstrated in frameworks like HardNet.

Repair Methods: Success and Failure

When it Succeeds

- **Hard Feasibility Guarantee:** This is the primary advantage. The output is *always* feasible by construction, which is non-negotiable for high-stakes applications.
- **Efficient for Simple Sets:** If the feasible set C allows for an efficient projection (e.g., affine sets, boxes, balls, probability simplex), the repair step is fast and its derivative is easy to compute.

When it Fails (or is Impractical)

- **Computational Cost:** If C is defined by complex constraints, solving the projection problem at every forward pass of the network can be prohibitively expensive.
- **Information Loss:** The projection can be a “violent” operation. It might move \tilde{x} a large distance, potentially discarding useful information learned by the model and hurting the primary objective f_0 . The model might learn to rely on the projection as a crutch.
- **Requires Convexity:** The projection $\Pi_C(\tilde{x})$ is a well-defined and tractable convex optimization problem only if the feasible set C is itself convex. This limits applicability for problems with non-convex constraints.

Example Showdown: Projection onto the Probability Simplex

Problem: An AI model outputs a logit vector $\tilde{\mathbf{x}} \in \mathbb{R}^n$. We want to produce a probability distribution \mathbf{x} that minimizes $\|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2$ subject to the simplex constraints $C = \{\mathbf{x} \mid \mathbf{x} \geq 0, \mathbf{1}^T \mathbf{x} = 1\}$.

Path 1: Lagrangian Approach

1. Form the Lagrangian:

$$L(\mathbf{x}, \lambda, \nu) = \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2 - \lambda^T \mathbf{x} + \nu(1^T \mathbf{x} - 1).$$

2. Minimize L with respect to \mathbf{x} by setting $\nabla_{\mathbf{x}} L = 0$:

$$2(\mathbf{x} - \tilde{\mathbf{x}}) - \lambda + \nu \mathbf{1} = 0 \Rightarrow x_i = \tilde{x}_i + \frac{\lambda_i - \nu}{2}.$$

3. The training process must then learn $\lambda \geq 0$ and ν that satisfy the KKT conditions (stationarity, primal/dual feasibility, complementarity). The final \mathbf{x} is not guaranteed to be exactly on the simplex.

Path 2: Repair Approach

1. The model produces $\tilde{\mathbf{x}}$.

2. The repair layer solves the projection problem:

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x} \in C} \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2.$$

3. This is a standard Quadratic Program (QP) with a known, efficient solution algorithm.

4. The final output \mathbf{x}^* is guaranteed to be a valid probability distribution. Gradients are computed by differentiating through the KKT solution of the QP.

A Traveler's Guide: Choosing Your Path

The choice between Lagrangian and Repair methods depends on the specific problem's requirements.

Feature	Lagrangian Methods	Repair Methods
Feasibility Guarantee	Soft. Prone to small violations in non-convex cases.	Hard. Guaranteed by construction.
Core Idea	Economic: ‘Pricing’ constraints via dual variables.	Geometric: ‘Projecting’ solutions onto the feasible set.
Computational Cost	Solving $\inf_x L(x, \lambda, \nu) +$ dual updates during training.	Solving the projection problem $\Pi_C(\cdot)$ in every forward and backward pass.
Primary Requirement	Differentiable objective f_0 and Lagrangian.	A convex feasible set C with a tractable and differentiable projection.
Best Suited For...	Problems where small violations are tolerable and constraints might be non-differentiable.	High-stakes problems requiring strict feasibility where C is a convex set amenable to fast projection.
B&V Foundation	Chapter 5: Duality	Chapter 8: Projection on a Set

The Frontier: A Synthesis of Learning and Optimization

Enforcing hard constraints is not just about retrofitting classical optimization onto modern AI. It is about creating a true synthesis. The frontier of this field lies in pushing beyond these two fundamental paths:

- **Hybrid Methods:** Can we use a Lagrangian penalty to guide a model's output \tilde{x} to be closer to the feasible set, thereby making the final repair projection less “violent” and more efficient?
- **Amortized Optimization:** Instead of solving a projection problem from scratch every time, can we train a separate neural network to *approximate* the projection operator $\Pi_C(\cdot)$ for faster inference?
- **Non-Convex Constraints:** Developing methods that can provide guarantees for structured non-convex feasible sets remains a major open challenge.



The principles laid out in texts like Boyd & Vandenberghe are not relics; they are the essential grammar for building the next generation of robust, reliable, and deployable AI systems.

Key References

Foundation

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Roadmap

Van Hentenryck, P. (2020). AI and optimization: A roadmap for constraint satisfaction. *AI Magazine*.

Methodology Examples

- **Lagrangian Methods:** Fioretto, F., et al. (2020). Lagrangian Duality for Constrained Deep Learning. *AAAI Conference on Artificial Intelligence*.
- **Repair Methods:** Vlastelica, M., et al. (2019). Differentiation of Blackbox Combinatorial Solvers. *International Conference on Learning Representations* (Introduced concepts for differentiating through solvers, relevant to HardNet).

Note: Full citations for Fioretto and HardNet papers are representative examples. Replace with the exact papers if they differ.