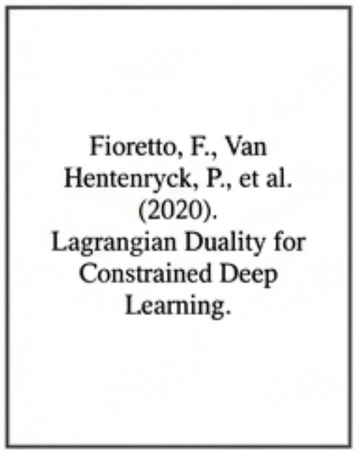
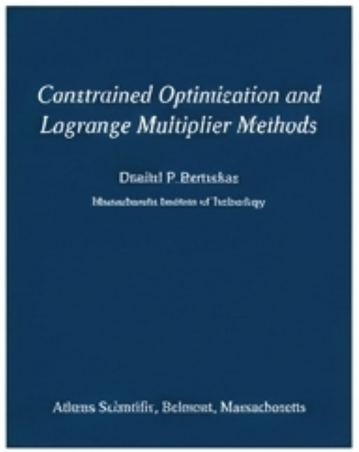
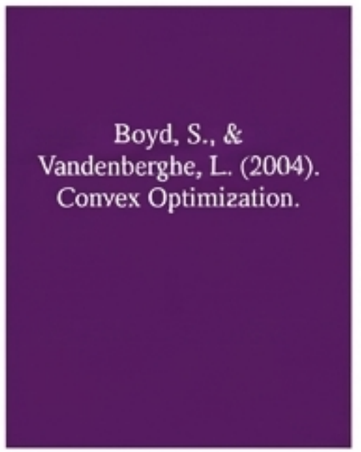
Lagrangian Duality for Constrained Deep Learning

A Tutorial Review of Classic Techniques in a Modern Context

Key References







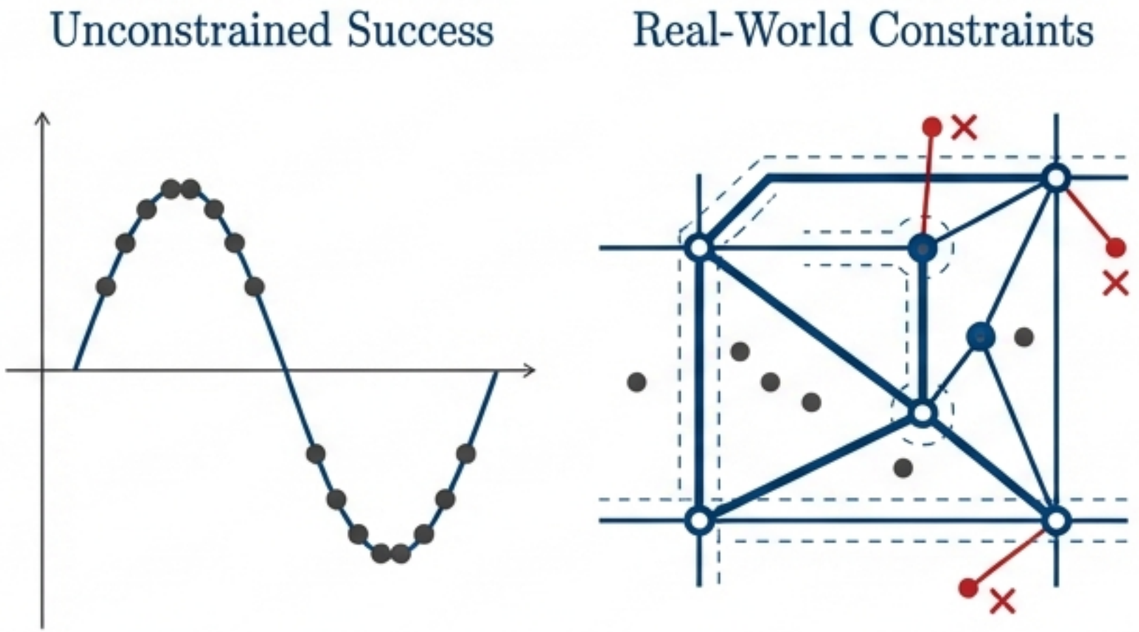
The Challenge: Deep Learning Meets Real-World Constraints

Deep Neural Networks (DNNs) have achieved unprecedented success in unconstrained tasks like classification and regression.

However, many critical applications require outputs that adhere to hard constraints:

- **Physical Laws**: Energy grids (Kirchhoff's Laws), fluid dynamics (Navier-Stokes).
- **Operational Limits**: Voltage bounds, thermal limits, engineering tolerances.
- **Model Properties**: Monotonicity, convexity, Lipschitz continuity.
- **Societal Rules**: Algorithmic fairness, disparate impact avoidance.

A naive DNN prediction may be accurate on average, but catastrophic if it violates these fundamental rules. This research addresses the need to provide deep learning architectures with the capability to capture constraints directly.



A Tale of Two Constraints: Sample-Wise vs. Predictor-Wide

We can classify constrained learning problems into two distinct types:

1. Constrained Optimization Problems

Constraints apply to the output for **each individual sample**.

- The model M_w predicts an output $\hat{y} = M_w(d)$ based on input features d.
- The constraints are a function of the prediction and the input: $g_i(\hat{y}, d) \leq 0$.

Example: Optimal Power Flow. For a given energy demand d, the predicted generator dispatch $\hat{y}$ must satisfy the physical laws of the grid for that specific demand.

2. Constrained Predictor Problems

Constraints enforce **a global property on the predictor itself**, relating multiple samples.

- The constraint involves a set of predictions $\{M_w(d_l)\}_{l \in S_i}$.
- The constraint is of the form $h_i(\{M_w(d_l)\}_{l \in S_i}) \leq 0$.
- Examples:
  - **Monotonicity:** For any two inputs d_1, d_2 where $d_1 \succ d_2$, we must have $M_w(d_1) \geq M_w(d_2)$.
  - **Fairness:** The positive prediction rate must be equal across two demographic groups S_0 and S_1.

The Toolkit: A Refresher on Lagrangian Duality

Let's revisit the standard constrained optimization problem:

The Primal Problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \end{aligned}$$

The Lagrangian Relaxation
We relax the hard constraints into a 'soft' penalty in the objective using non-negative Lagrange multipliers $\lambda_i \geq 0$:

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

Key Property: For any feasible point x^* (a point satisfying all constraints) and any $\lambda \geq 0$, the Lagrangian provides a lower bound on the optimal value $f(x^*)$:

$$\min_x \mathcal{L}(x, \lambda) \leq \mathcal{L}(x^*, \lambda) = f(x^*) + \sum \lambda_i g_i(x^*) \leq f(x^*)$$

This holds because $g_i(x^*) \leq 0$ and $\lambda_i \geq 0$, making the sum non-positive.

Finding the Best Lower Bound: The Dual Problem

The Lagrange Dual Function

The dual function $q(\lambda)$ is the minimum value of the Lagrangian over the primal variable x. It is a function of the multipliers λ.

$$q(\lambda) = \inf_x \mathcal{L}(x, \lambda) = \inf_x \left(f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \right)$$

The Dual Problem

The dual problem is to find the multipliers λ that provide the tightest possible lower bound on the primal problem. This is always a convex optimization problem.

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & q(\lambda) \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned}$$

Solving the Dual: Subgradient Ascent

We can solve the dual problem iteratively. If x_k minimizes $\mathcal{L}(x, \lambda_k)$, then the constraint violations $g_i(x_k)$ form a subgradient of $q(\lambda)$ at λ_k. We can ascend in the direction of violation:

$$\lambda_{k+1} = [\lambda_k + s_k \nabla_\lambda \mathcal{L}(x_k, \lambda_k)]^+ = [\lambda_k + s_k \, g(x_k)]^+$$

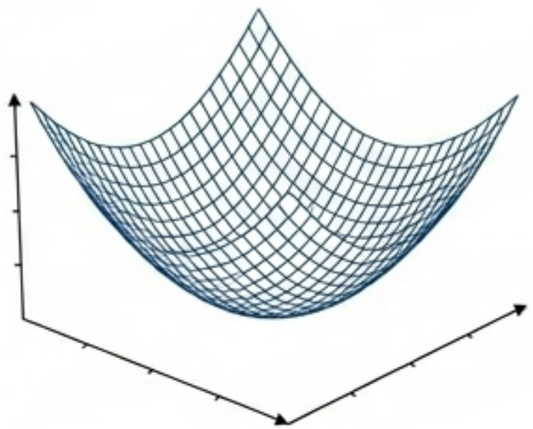
where s_k is a step size and $[\cdot]^+$ denotes projection onto the non-negative orthant.

The Bridge: Reframing the Learning Task as Constrained Optimization

Let's view the training of a neural network through a constrained optimization lens.

Classical View

Variables: Vector $x \in \mathbb{R}^d$
Objective: Function $f(x)$
Constraints: A set of functions $g_i(x) \leq 0$



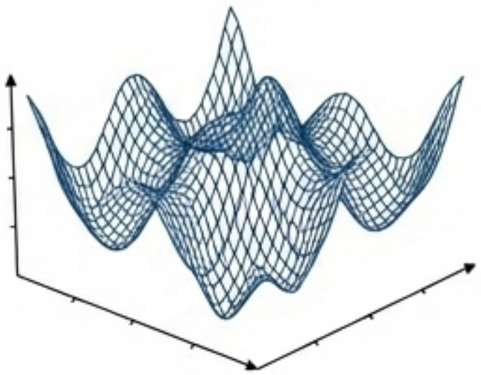
Learning Task View

Variables: Network weights $w \in \mathbb{R}^W$
Objective: Empirical Loss over dataset $\mathcal{D}$:

$$\mathcal{L}_{emp}(w) = \sum_{(d_l, y_l) \in \mathcal{D}} L(M_w(d_l), y_l)$$

Constraints: The model's predictions must be feasible for all inputs in the training set:

$$g_i(M_w(d_l), d_l) \leq 0, \quad \forall l \in \{1, \ldots, n\}, \forall i \in \{1, \ldots, m\}$$



The learning task becomes a massive, high-dimensional constrained optimization problem over the weights w. A direct solution is intractable. We need a more subtle approach.

The Core Idea: An Iterative Game in the Training Loop

Instead of solving for w and λ separately, we interleave their updates during training. This transforms the static optimization problem into a dynamic two-player game.

The Lagrangian Loss Function

We augment the standard loss function L with a penalty for constraint violations, weighted by the *learned* multipliers λ:

$$\mathcal{L}_\lambda(w) = \underbrace{L(M_w(d_l), y_l)}_{\text{Accuracy Term}} + \underbrace{\sum_{i=1}^{m} \lambda_i \cdot \nu(g_i(M_w(d_l), d_l))}_{\text{Constraint Violation Term}}$$

where $\nu(\cdot) = \max(0, \cdot)$ measures the degree of violation.

The Two-Player Game

1. **Primal Player (The Network):** Minimizes the Lagrangian loss for a fixed set of penalties λ.

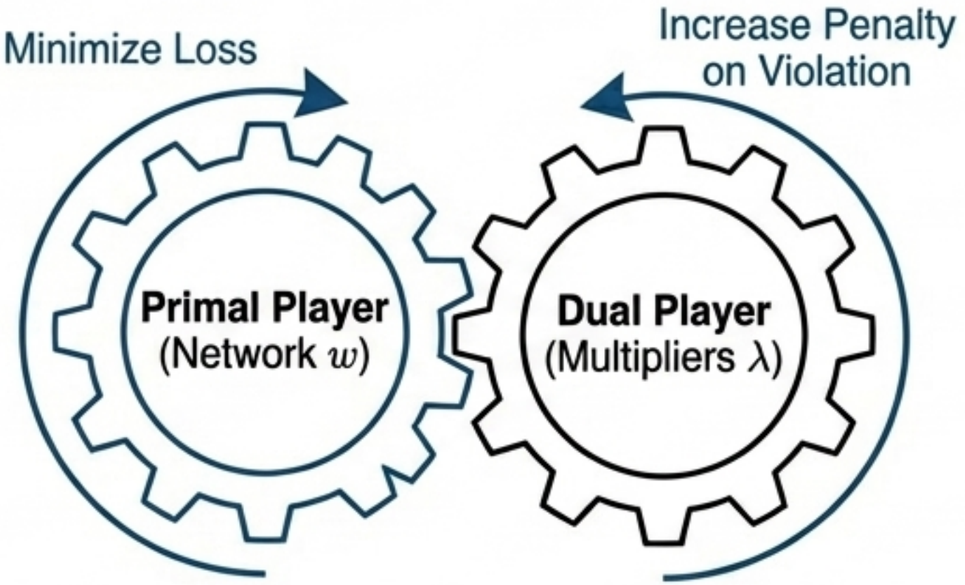
   $$w \leftarrow w - \alpha \nabla_w \mathcal{L}_\lambda(w)$$

   This is just standard backpropagation on a modified loss.

2. **Dual Player (The Multipliers):** Increases the penalty λ_i for constraints that the network is currently violating.

   $$\lambda_i \leftarrow \lambda_i + s \cdot \frac{1}{n} \sum_{l=1}^{n} \nu(g_i(M_w(d_l), d_l))$$

   This is a dual ascent step.



Algorithm 1: The Lagrangian Dual Framework (LDF)

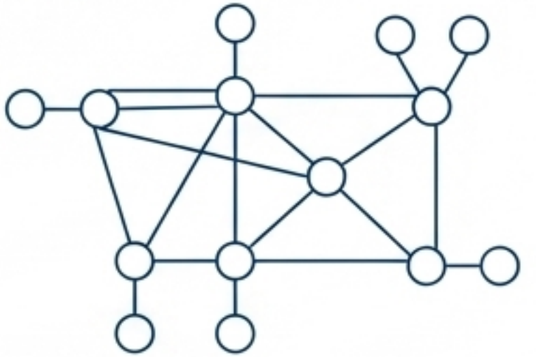
\begin{algorithmic}[1]
\STATE **Input:** Training data $\mathcal{D} = \{(d_l, y_l)\}_{l=1}^{n}$, step sizes α, s_k
\STATE **Initialize** multipliers $\lambda_i^{(0)} \leftarrow 0$ for all constraints $i \in \{1, \ldots, m\}$
FOR epoch $k = 0, 1, \ldots\}$
FOReach sample $(d_l, y_l) \in \mathcal{D}$
\STATE Compute prediction $\hat{y}_l \leftarrow M_{w^{(k)}}(d_l)$
\STATE Compute Lagrangian Loss $\mathcal{L}_{\lambda^{(k)}}(w^{(k)})$
\STATE Update weights: $w \leftarrow w - \alpha \nabla_w \mathcal{L}_{\lambda^{(k)}}$ \COMMENT *Primal Step: SGD*
ENDFOR
\STATE Update multipliers:

$$\lambda_i^{(k+1)} \leftarrow \lambda_i^{(k)} + s_k \left(\frac{1}{n} \sum_{l=1}^{n} \nu(g_i(M_w(d_l), d_l)) \right)$$

\COMMENT *Dual Step: Ascent*
ENDFOR
\end{algorithmic}

The multipliers λ adaptively learn which constraints are “hard” for the network, dynamically adjusting the loss landscape to guide the weights w toward feasible solutions.

Payoff 1: Physically-Sound Predictions for Optimal Power Flow



Problem

Predict generator settings ($\hat{y}$) for a power grid given demand (d). Predictions must satisfy AC power flow equations, voltage limits, etc. ($g(\hat{y}, d) \leq 0$). A standard DNN (M) produces physically impossible predictions.

Results: Constraint Violation Distance (%)

(The minimal adjustment needed to make a prediction feasible. Lower is better.)

Test Case	Variable	M (Naive)	MC (Fixed λ)	**MDc (LDF)**
30_ieee	Voltage (v)	83.14%	0.094% (880x gain)	**0.0037% (22,469x gain)**
300_ieee	Voltage (v)	31.70%	0.238% (133x gain)	**0.1994% (159x gain)**

*Source: Fioretto et al. (2020), Table 3

Takeaway

The LDF model, which learns the multipliers, dramatically reduces constraint violations by several orders of magnitude, producing physically realistic outputs.

Payoff 2: Enforcing Model Properties like Monotonicity

Problem

In transprecision computing, predict the error $(\hat{y})$ of a low-precision configuration (d). Higher precision should never result in higher error. This is a **Constrained Predictor** problem.

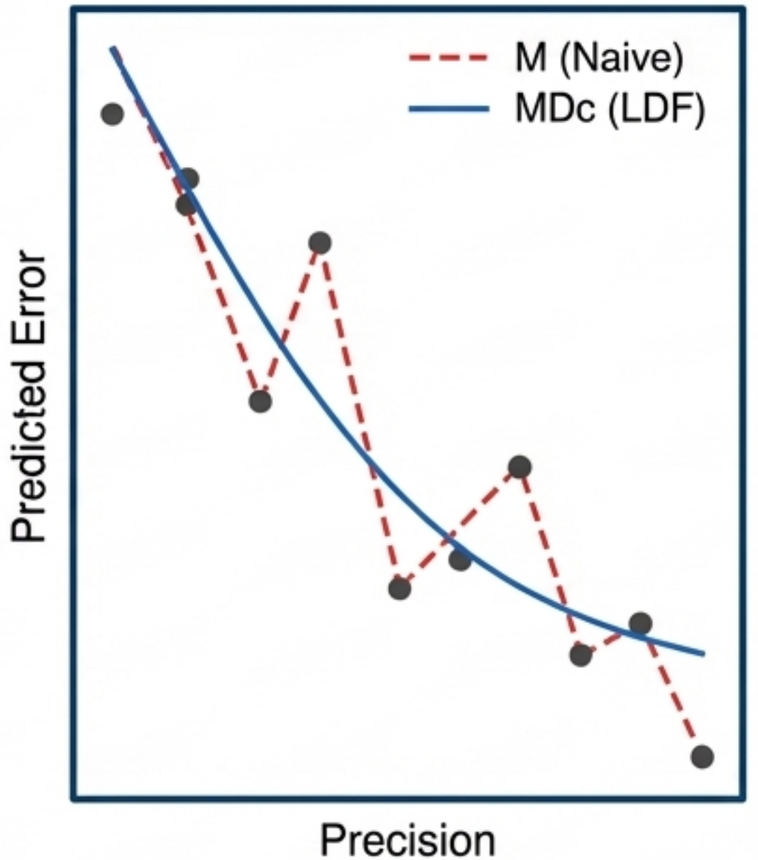
$$\forall (d_i, d_j), \text{ if } d_i \succ d_j \text{ (is more precise), then } M_w(d_i) \leq M_w(d_j)$$

Results: Violation Count (VC) on Test Set

(Average number of pairs that violate the monotonicity constraint)

Train Size	**M** (Naive)	MC (Fixed λ)	**MDc (LDF)**
200 samples	9.6	35.8	**7.4**
600 samples	2.5	9.1	**1.7**
1000 samples	0.4	5.7	**0.5**

*Source: Fioretto et al. (2020), Table 4



Takeaway

LDF effectively enforces monotonicity without sacrificing prediction accuracy. The constraints act as a powerful regularizer, especially in low-data regimes. The fixed-penalty approach (MC) performs poorly, highlighting the need for adaptive multipliers.

Payoff 3: A Principled Approach to Algorithmic Fairness

Problem

A classifier must not have a disparate impact on different demographic groups. For a protected attribute (e.g., race), the positive prediction rates should be similar. This is a **Constrained Predictor** problem, where the constraint is on the statistical properties of the model over subsets of the data.

$$|\mathbb{E}[\hat{y}|s=0] - \mathbb{E}[\hat{y}|s=1]| \leq \epsilon$$

M (Naive)

MDc (LDF)

Results: Accuracy (Acc) vs. Disparate Treatment (DT) Score

(Lower is Better)

Dataset	**M** (Naive)		MDc (LDF)	
	Acc.	DT	Acc.	**DT**
Adult	0.842	0.185	0.834	**0.055**
Bank	0.826	0.447	0.814	**0.122**

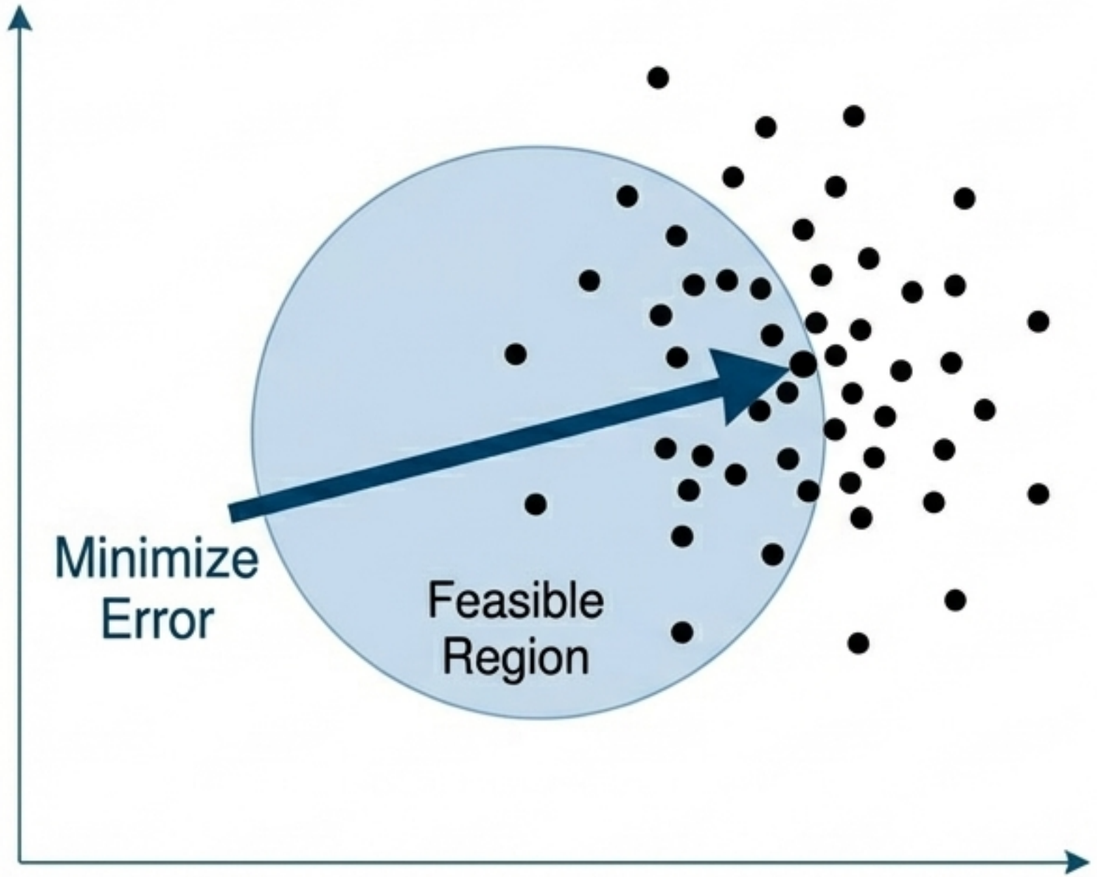
*Source: Fioretto et al. (2020), Table 5

Takeaway

LDF provides a principled way to enforce fairness, achieving a significantly lower (fairer) disparity score with only a minor trade-off in overall accuracy.

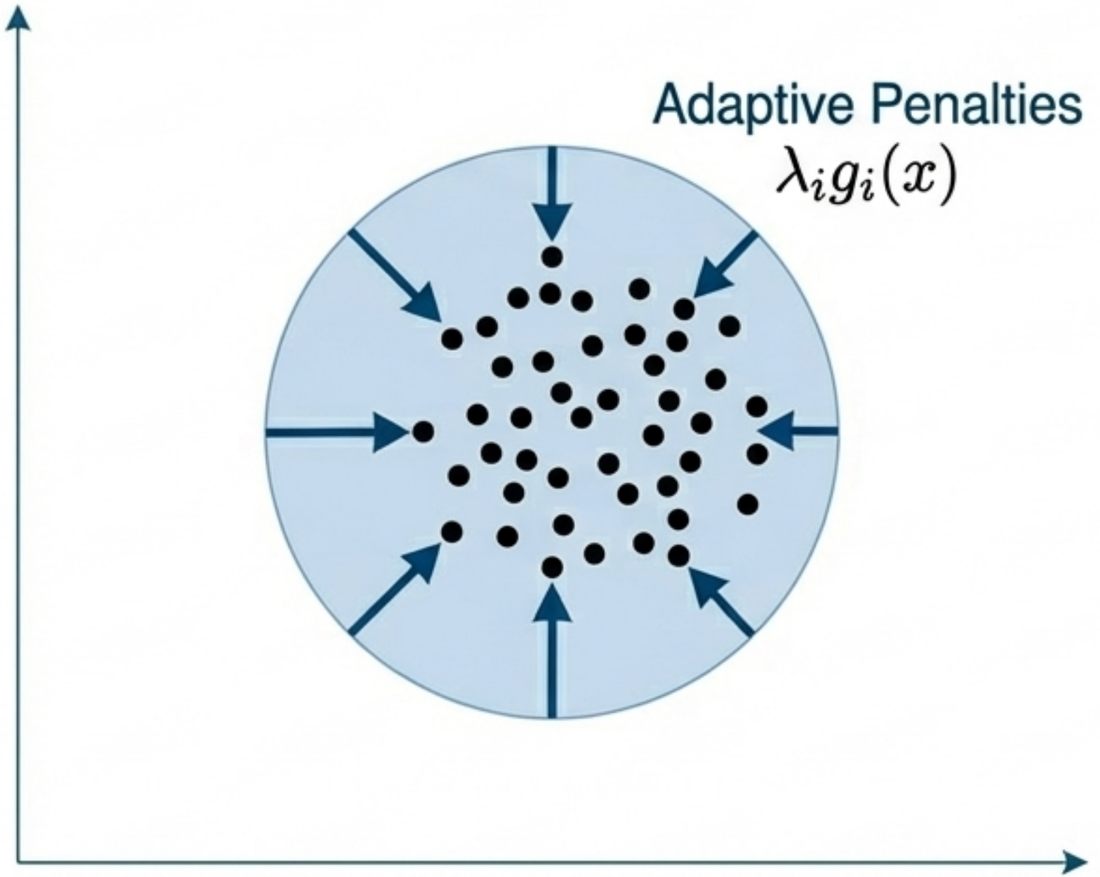
How LDF Shapes the Solution Space During Training

Unconstrained Learning (Standard NN)



Predictions are driven solely by minimizing error, often landing in the infeasible region.

Constrained Learning (LDF)



The adaptive multipliers create “walls” that guide predictions into the feasible region during training.

Why This Works So Well: Key Benefits of the LDF

1. Principled Constraint Handling

Converts a hard, constrained learning problem into a sequence of familiar unconstrained problems that can be solved with standard tools (e.g., SGD, Adam).

2. Adaptive Penalties

The Lagrange multipliers λ are not fixed hyperparameters that need to be tuned. They are *learned variables* that automatically adjust throughout training to focus on the most violated constraints.

3. Domain Knowledge as Regularization

Enforcing known structural properties (like physical laws or monotonicity) acts as a powerful regularizer, often leading to better generalization and requiring less data.

4. Flexibility and Generality

The framework can handle a wide variety of differentiable (or sub-differentiable) constraints, from sample-wise physical limits to dataset-wide statistical properties.

Conclusion: Old Tools, Powerful New Tricks

Lagrangian duality, a cornerstone of classical optimization, provides a powerful and elegant framework for enforcing complex constraints in deep learning.

The Big Idea: By embedding the iterative process of dual ascent directly into the training loop, we create an adaptive penalty mechanism that guides models towards solutions that are not only **accurate** but also **feasible**, **robust**, and **fair**.

Food for Thought for Researchers

- What other constraints from your domain could be integrated using this framework?
  - Safety constraints in Reinforcement Learning?
  - Causality constraints in graphical models?
  - Stability constraints for learning dynamical systems?
  - Conservation laws in scientific machine learning (SciML)?

Key References

1. Primary Application Paper

Fioretto, F., Van Hentenryck, P., et al. (2020). 'Lagrangian Duality for Constrained Deep Learning." *Preprint arXiv:*2001.09394.

2. Classical Foundation: Duality and Multiplier Methods

Bertsekas, D. P. (1999). "Constrained Optimization and Lagrange Multiplier Methods." *Athena Scientific.*

3. Classical Foundation: Convex Optimization

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