Interior Point Solving for LP-based prediction+optimisation

Jayanta Mandi, Tias Guns

VUB DATA ANALYTICS LABORATORY

NEURAL INFORMATION PROCESSING SYSTEMS

Contribution

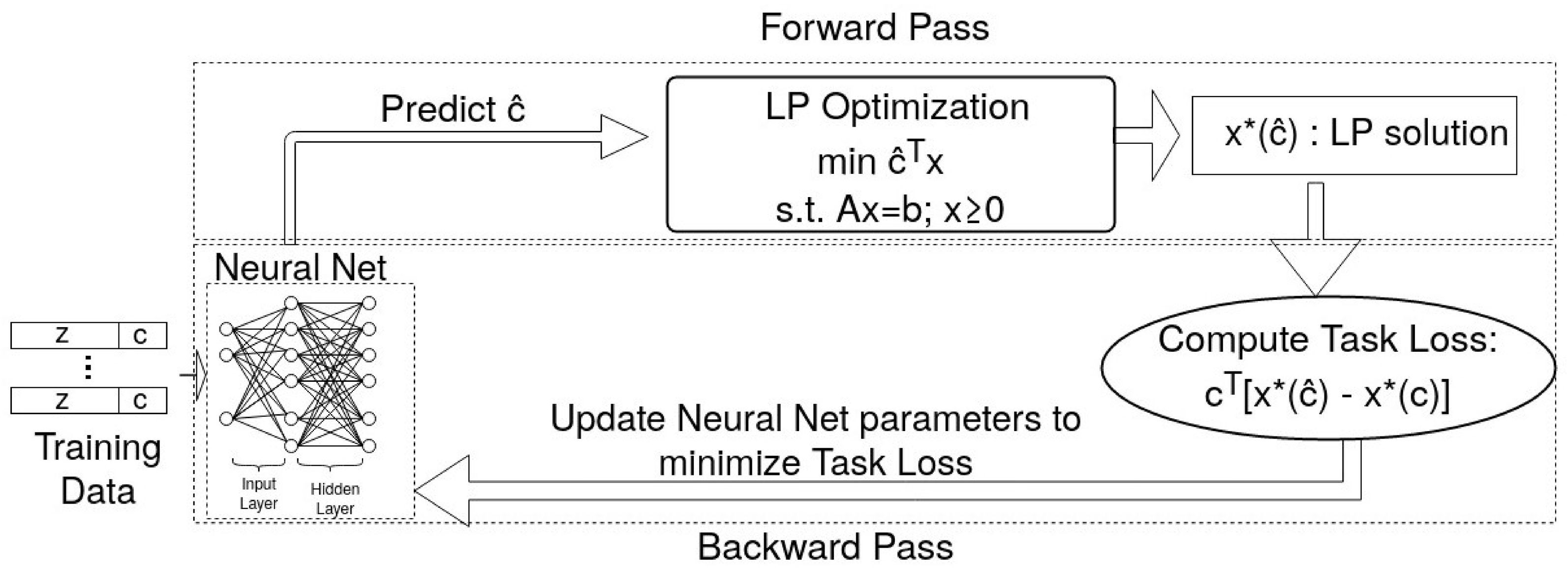
- We propose a *Differentiable Optimization layer* that methodically connects the derivatives used for optimisation and the gradients used for learning
- Specifically for Linear Programs (LP), we differentiate the *Homogeneous Self-dual (HSD)* embedding, the same formulation used for solving LP problems by interior point methods
- On the challenging problems of *prediction+optimization for Mixed Integer Linear Program*, we show that this framework outperforms the state of the art

Prediction+Optimisation Setting

LP Optimization

$$\min c^\top x \quad \text{s.t. } Ax = b; x \geq 0$$

- With c being unknown but historic data $\{(\mathbf{z}, c)\}$ are available to predict c from $\mathbf{z}$
- Neural Net predictions $\hat{c} = m(\mathbf{z})$ will be fed to the optimizer.
- The aim is to generate predictions $x^*(\hat{c})$ to minimize the task loss $c^\top(x^*(\hat{c}) - x^*(c))$



Challenges

- For the backward pass the derivative of task loss: $c^\top(x^*(\hat{c}) - x^*(c))$ must be computed
- Computing the derivative of $x^*(\hat{c})$ w.r.t. $\hat{c}$, i.e. $\frac{\partial}{\partial \hat{c}} x^*(\hat{c}) \Rightarrow$ *argmin differentiation*

Differentiating the KKT condition

For an optimization problem: $\min \mathbf{f}(\mathbf{c}, \mathbf{x}) s.t. \mathbf{A}\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$, the Lagrangian relaxation:

$$\mathbb{L}(x, y; c) = f(c, x) + y^\top(b - Ax) \; ; \; \textbf{dual variable: } y \tag{1}$$

And the KKT conditions are:

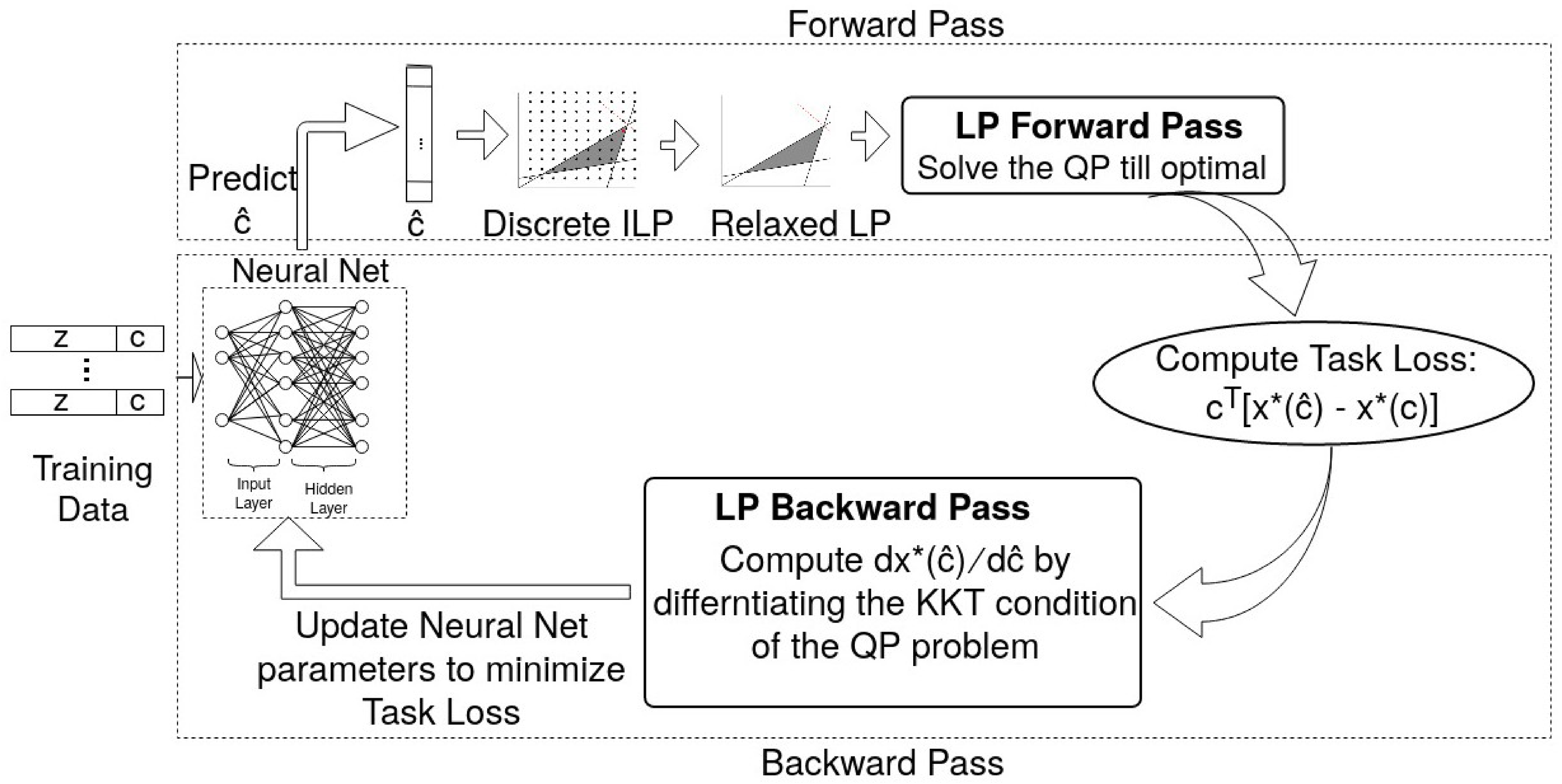
$$\begin{aligned} f_x(c, x) - A^\top y &= 0 \\ Ax - b &= 0 \end{aligned} \tag{2}$$

Implicit differentiation of Eq. (2) w.r.t. c yields:

$$\begin{bmatrix} f_{cx}(c, x) \\ 0 \end{bmatrix} + \begin{bmatrix} f_{xx}(c, x) & -A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial c} x \\ \frac{\partial}{\partial c} y \end{bmatrix} = 0 \tag{3}$$

Previous Work (Wilder et al., 2019)[1]

- To solve Eq. (3) $f_{xx}(c, x) \neq 0 \Rightarrow f(c, x)$ strictly convex
- Wilder et al. proposed to add *an external quadratic regularization term* to turn the LP into a QP, and differentiate Eq. (3)



Adding a log-barrier function

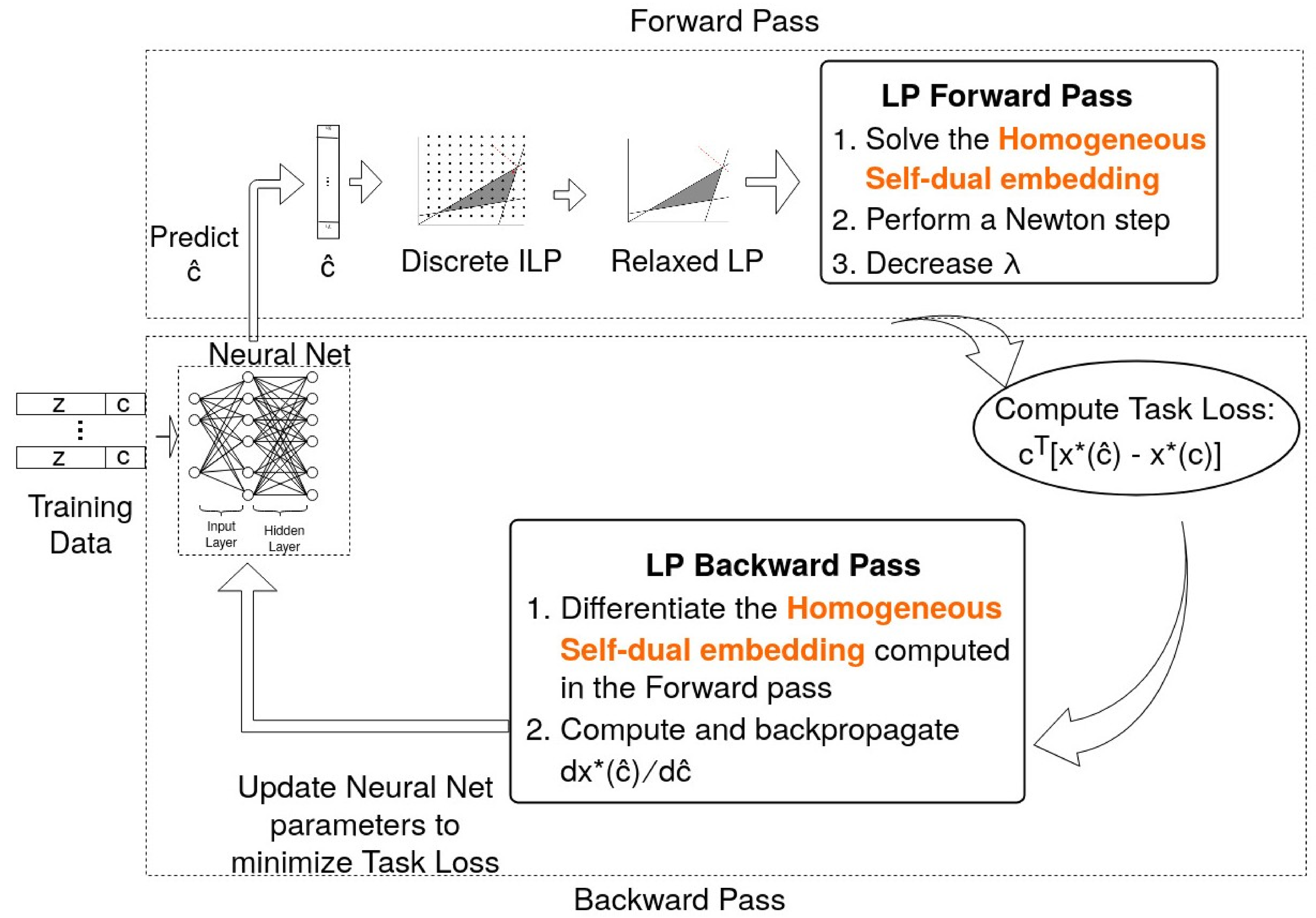
- A common tool in LP programming is the log barrier function: $\lambda\left(\sum_{i=1}^{k} ln(x_i)\right)$, which entails the constraint $x \geq 0$.
- We propose to differentiate the KKT conditions of the LP after adding the log-barrier function; i.e. $f(c, x) := c^\top x - \lambda\left(\sum_{i=1}^{k} ln(x_i)\right)$
- We do not differentiate at the optimal point (as at the optimal point $\lambda \simeq \mathbf{0}$) but rather at the neighborhood of the optimal point

Homogeneous Self-dual embedding

- Interior Point (IP) Solvers does not directly solve the KKT conditions. Instead a homogeneous self-dual formulation is formed which has two additional parameters.
- The advantage of homogeneous self-dual formulation is it always has a strictly complementary solution and based which parameter is 0, it can detect whether the LP problem is infeasible.
- IP solvers solve the LP using a "homotopy strategy" where a sequence of "easier" problems are solved for a decreasing sequence of the barrier parameter λ.
- The method can start from any point even from a point infeasible to the original LP.

Differentiating Homogeneous Self-dual embedding

- Instead of the KKT conditions, differentiate the HSD embedding
- We stop solving in the forward pass when λ goes below a threshold value λ-cut-off



KKT vs HSD

	KKT, log barrier			HSD, log barrier		
λ / λ-cut-off	10^{-1}	10^{-3}	10^{-10}	10^{-1}	10^{-3}	10^{-10}
Regret	*14365*	14958	21258	**10774**	14620	21594

Table 1: Differentiating the HSD formulation is more efficient than differentiating the KKT condition

Comparison with the state of the art

	Two-stage		QPTL[1]		SPO[2,3]		HSD,log barrier	
	0-layer	1-layer	0-layer	1-layer	0-layer	1-layer	0-layer	1-layer
MSE-loss	**745 (7)**	796 (5)	3516 (56)	2×10^9 (4×10^7)	3327 (485)	3955 (300)	2975 (620)	1.6×10^7 (1×10^7)
Regret	13322 (1458)	13590 (2021)	13652 (325)	13590 (288)	11073 (895)	12342 (1335)	**10774 (1715)**	11406 (1238)

Table 2: Our approach is able to outperform the state of the art

References

References

[1] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *AAAI-19*.

[2] Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize". *arXiv preprint arXiv:1710.08005*, 2017.

[3] Jayanta Mandi, Tias Guns, Emir Demirovi, and Peter. J Stuckey. Smart predict-and-optimize for hard combinatorial optimization problems. In *AAAI 2020 : The Thirty-Fourth AAAI Conference on Artificial Intelligence*, volume 34, pages 1603–1610, 2020.