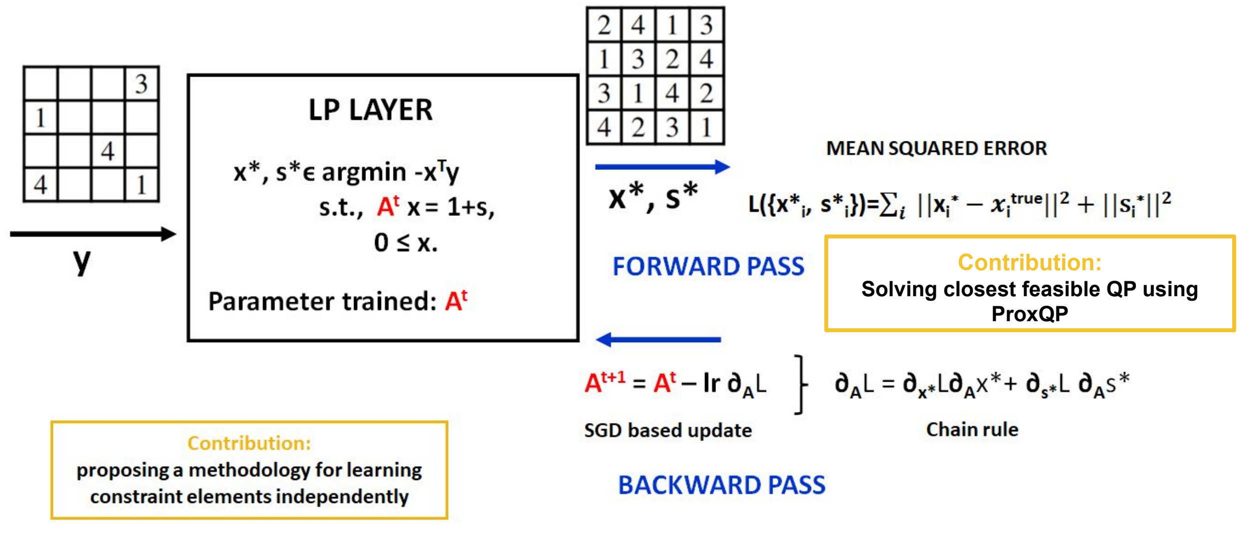
Leveraging augmented-Lagrangian techniques for differentiating over infeasible quadratic programs in machine learning

Leveraging augmented-Lagrangian techniques for differentiating over infeasible quadratic programs in machine learning

A. Bambade[1,2,3] F. Schramm[1,2] A. Taylor[1,2] J. Carpentier[1,2]

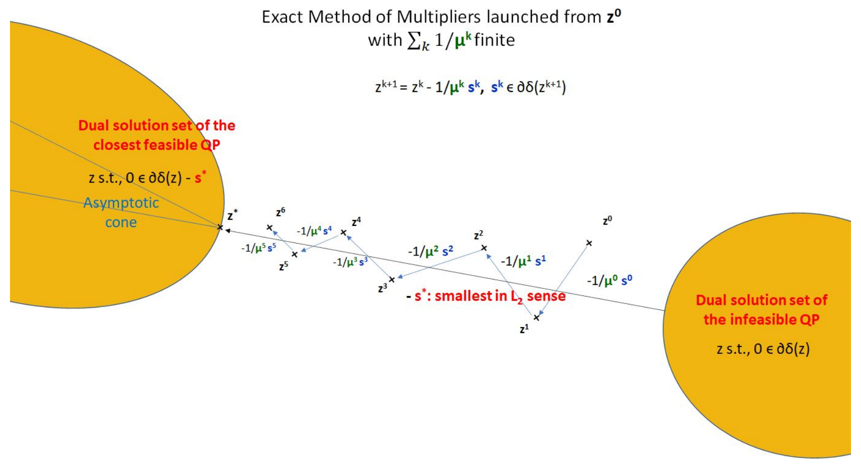
[1]Inria Paris, France [2]Département d'informatique de l'ENS, PSL Research University, Paris, France [3]École des Ponts, Marne-la-Vallée, France

Motivation and solution pipeline



A. Chiche, J-C. Gilbert (2016)

The forward pass: Augmented Lagrangian methods in a nutshell



The backward pass: differentiating closest feasible QP solutions

$$
\begin{aligned}
s^\star(\theta) = & \arg\min_{s\in\mathbb{R}^{n_i}} \tfrac{1}{2}\|s\|_2^2 \\
& \text{s.t. } x^\star(\theta), z^\star(\theta) \in \arg\min_{x\in\mathbb{R}^n}\max_{z\in\mathbb{R}^{n_i}_+} L(x,z,s;\ \theta),
\end{aligned}
$$

with $L(x,z,s;\ \theta) \triangleq f(x;\ \theta) + z^\top (C(\theta)x - u(\theta) - s)$.

Contribution: QPLayer: A full differentiable pipeline in C++ connected with PyTorch.

A classical technique: **the Implicit Function Theorem.**

$$
G(x,z,t;\ \theta) \triangleq \begin{bmatrix} \nabla_x f(x;\ \theta) + C(\theta)^\top z \\ C(\theta)x - u(\theta) - t \\ [[t]_- + z]_+ - z \\ C(\theta)^\top [t]_+ \end{bmatrix}
$$

Contribution: Extend the technique for the closest feasible QP solutions.

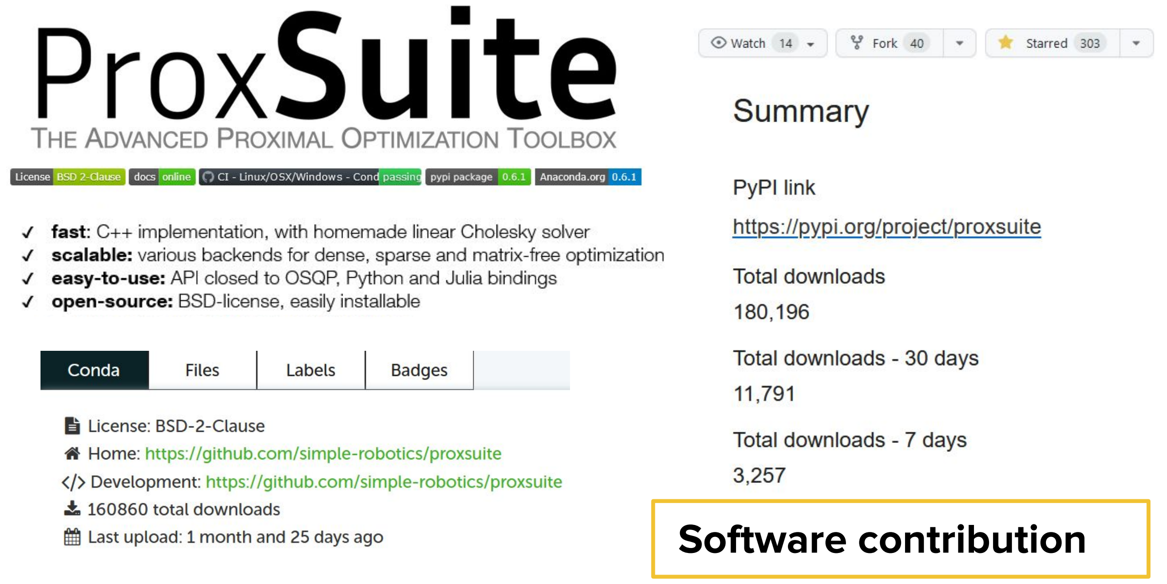
The map is **path-differentiable**.

$$
\left(\frac{\partial x^\star}{\partial \theta}, \frac{\partial z^\star}{\partial \theta}\right) \in \arg\min_w \left\| \frac{\partial G(x^\star, z^\star;\ \theta)}{\partial v^\star} w + \frac{\partial G(x^\star, z^\star;\ \theta)}{\partial \theta} \right\|_2^2,
$$

$$
\Pi \frac{\partial t^\star}{\partial \theta} \in \frac{\partial s^\star}{\partial \theta}, \text{ with } \Pi \in \partial([.]_+)(t^\star).
$$

Contribution: Efficient algorithms to solve these problems.

Augmented Lagrangian methods in practice with ProxSuite library



Software contribution

Benchmarks

	OptNet	QPLayer
Forward pass (ms)	615.84 ± 16.15	55.2 ± 6.93
Backward pass (ms)	61.26 ± 2.84	39.27 ± 2.17
Final Loss	0.02604	0.02556

Table: Average computational times (over 800 epochs) for solving a cart-pole example with friction when using OptNet or QPLayer. Randomized smoothing is used for obtaining informative gradients [1].

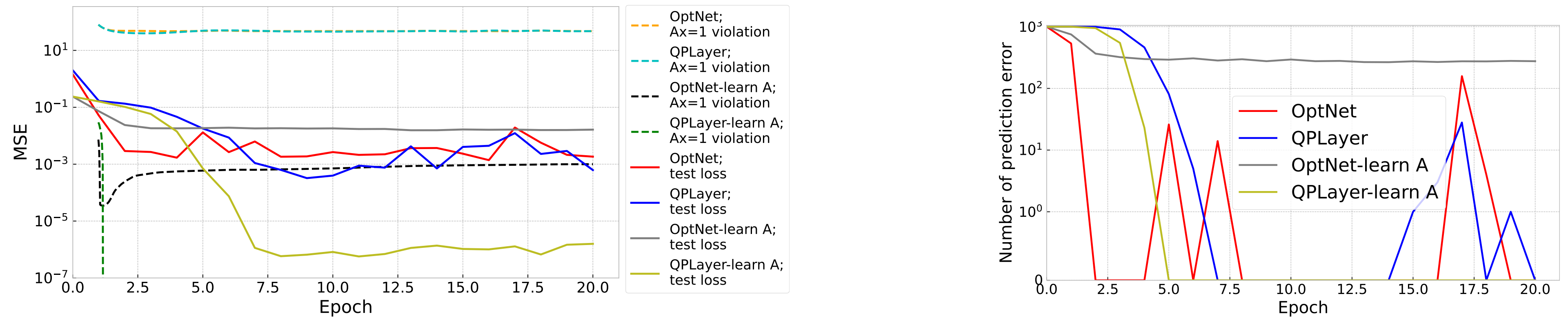


Figure: Sudoku training and test plots using QPLayer and OptNet layers. QPLayer can learn LPs, whereas OptNet is restricted to strictly convex QPs, which limits its representational power. QPLayer can also be specialized to learn models satisfying specific linear constraints.

Future work

- Extension to GPUs,
- Beyond LPs and QPs: SOCPs, SDPs, etc..

References

1. Lidec, Q. L., Montaut, L., Schmid, C., Laptev, I. & Carpentier, J. *Leveraging Randomized Smoothing for Optimal Control of Nonsmooth Dynamical Systems.* 2022.