# Neural Decision Rule for Constrained Contextual Stochastic Optimization

**Zhangyi Liu*** Tsinghua | **Zhongling Xu*** UT Austin | **Feng Liu** UCAS | **Rui Gao** UT Austin | **Shuang Li** CUHK-SZ

## Abstract

Contextual stochastic optimization is a powerful paradigm for data-driven decision-making under uncertainty, where decisions are tailored to contextual information revealed prior to decision making. Recent advances leverage neural networks to learn expressive decision rules mapping contexts to decisions. However, enforcing feasibility under complex constraints remains a core challenge, as neural architectures do not inherently satisfy constraints in general. Existing approaches often incur significant computational overhead and lack theoretical guarantees. In this paper, we propose a principled framework for training neural decision rules under general constraints via a single-loop algorithm that solves the augmented Lagrangian minimax problem. Our method eliminates the need for iterative projection layers or nested optimization loops, and provides provable guarantees on generalization, constraint violation, and suboptimality. Empirical results on constrained contextual decision tasks demonstrate that our approach outperforms state-of-the-art baselines in both efficiency and solution quality.

## 1 Introduction

Contextual stochastic optimization (CSO) refers to the paradigm of data-driven decision-making under uncertainty, where side information or covariates (i.e., contexts) are available before a decision is made. Unlike classical stochastic programming, where the distribution of uncertainty is assumed to be fixed and known, CSO models leverage covariate-dependent distributions to tailor decisions to the observed context. This approach is highly relevant across domains such as supply chain management [Ban and Rudin, 2019, Cristian et al., 2022], portfolio optimization [Xu and Cohen, 2018, Zhang et al., 2021], and energy systems [Chen et al., 2021, Kong et al., 2022].

A central modeling approach for CSO is to learn a decision rule—a function that maps from contexts to decisions. Early work focused on linear decision rules, which offer tractability but are limited in expressiveness [Ban and Rudin, 2019]. Subsequent developments extended to nonlinear settings through reproducing kernel Hilbert spaces (RKHS) [Bertsimas and Koduri, 2022], and more recently, to neural decision rules parameterized by deep networks that can capture complex, high-dimensional relationships [Qi et al., 2023]. However, one major challenge with neural decision rules is ensuring feasibility under complex constraints, as neural networks do not inherently produce outputs within prescribed feasible sets.

Several methods have been proposed to enforce feasibility for the neural decision rules. Some are application-specific—for example, using softmax parameterizations to ensure simplex constraints in portfolio optimization [Zhang et al., 2021], or incorporating differentiable repair layers that efficiently project infeasible outputs back into the feasible set [Chen et al., 2023]. Other approaches aim for broader generality. One common way is implicit differentiation techniques that enable end-to-end

*Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

training [Donti et al., 2019, 2021] through the KKT conditions of embedded optimization layers [Amos and Kolter, 2017, Agrawal et al., 2019]. An alternative way employs Lagrangian-based methods—including augmented Lagrangian formulations [Kotary and Fioretto, 2024]—to recast constrained CSO as a minimax problem [Park and Hentenryck, 2022]. While this enables a principled treatment of constraints, existing algorithms typically rely on double-loop optimization, which substantially increases training time. Moreover, these methods are often heuristic and lack formal performance guarantees.

In this paper, we propose a new framework for training neural decision rules under general convex constraints. Our approach solves the augmented Lagrangian problem using a single-loop stochastic algorithm that sequentially updates both primal and dual parameters via noisy stochastic gradient with weight decay, avoiding the inefficiencies of double-loop optimization common in functional augmented Lagrangian methods. We establish performance guarantees, including generalization error bounds, regularization bias, explicit bounds on constraint violation and projection error, and the suboptimality gap of the algorithm output. Finally, we validate our approach on constrained contextual decision tasks such as inventory management and energy operations, where it consistently outperforms existing baselines.

## 2 Model Setup

### 2.1 Contextual Stochastic Optimization

The goal of contextual stochastic optimization (CSO) is to make data-driven decisions under uncertainty using observable contextual information. Let $\mathcal{X} \subset \mathbb{R}^{d_\mathcal{X}}$ denote the space of contexts, $\mathcal{Y} \subset \mathbb{R}^{d_\mathcal{Y}}$ the space of feasible decisions, and $\mathcal{Z} \subset \mathbb{R}^{d_\mathcal{Z}}$ the space of random outcomes. Let $\mathcal{D}$ denote the joint distribution over context-outcome pairs $(X, Z) \in \mathcal{X} \times \mathcal{Z}$. In practice, the distribution $\mathcal{D}$ is unknown and only a finite dataset of i.i.d. samples is available. Let $\hat{\mathcal{D}} := \frac{1}{n}\sum_{i=1}^{n} \delta_{(x_i,z_i)}$ denote the empirical distribution induced by $n$ samples $\{(x_i, z_i)\}_{i=1}^{n} \sim \mathcal{D}$.

We seek to learn a decision rule $f : \mathcal{X} \to \mathcal{Y}$ that maps contextual features to feasible actions, minimizing the expected cost. However, directly optimizing over all measurable functions may lead to overfitting the training data and poor generalization. To address this, we restrict attention to a function class $\mathcal{F}$ that captures the inductive bias of the learning algorithm and balances the trade-off between approximation capacity and generalization ability. In our setting, $\mathcal{F}$ typically consists of a class of neural networks.

The learning objective is then

$$\min_{f \in \mathcal{F}} \mathbb{E}_{(X,Z)\sim\hat{\mathcal{D}}}[\Psi(f(X), Z)]. \tag{1}$$

We assume that the cost function $\Psi(y, z)$ is convex in $y$ for every fixed $z \in \mathcal{Z}$. The set of feasible decisions $\mathcal{Y}$ is defined by both convex inequality constraints and affine equality constraints:

$$\mathcal{Y} := \left\{ y \in \mathbb{R}^{d_\mathcal{Y}} : g(y) \leq 0,\ Ay = b \right\},$$

where $g : \mathbb{R}^{d_\mathcal{Y}} \to \mathbb{R}^{d_\leq}$ is a convex function, $A \in \mathbb{R}^{d_= \times d_\mathcal{Y}}$, and $b \in \mathbb{R}^{d_=}$. These constraints encode application-specific structure such as physical, budgetary, or regulatory limits.

### 2.2 Augmented Lagrangian Method on the Function Space

The decision rule $f \in \mathcal{F}$ must map each context $x \in \mathcal{X}$ to a feasible decision $f(x) \in \mathcal{Y}$, satisfying both convex inequality and equality constraints. Enforcing these constraints uniformly across the context space is particularly challenging when $f$ is represented by a neural network, which does not naturally preserve feasibility. To address this, we adopt an augmented Lagrangian formulation that enables principled constraint handling through penalization and dual optimization.

Let $\Lambda(\cdot) = \big(\Lambda_=(\cdot), \Lambda_\leq(\cdot)\big)$ denote the context-dependent Lagrange multipliers for the equality and inequality constraints, respectively, where $\Lambda_= : \mathcal{X} \to \mathbb{R}^{d_=}$ and $\Lambda_\leq : \mathcal{X} \to \mathbb{R}^{d_\leq}$. We consider the following augmented Lagrangian method (ALM) over function spaces:

$$\min_{f \in \mathcal{F}} \max_{\Lambda \in \mathcal{A}} L(f, \Lambda), \tag{2}$$

where $\mathcal{A}$ is a class of admissible dual functions, and $\lambda > 0$ is a penalty parameter. The augmented Lagrangian objective $L(f, \Lambda)$ takes the form:

$$L(f,\Lambda) := \mathbb{E}_{(X,Z)\sim\hat{\mathcal{D}}}\Bigg[\Psi(f(X),Z) + \Lambda_{=}(X)^{\mathsf{T}}(Af(X)-b) + \frac{\lambda}{2}\|Af(X)-b\|_2^2 + \frac{\lambda}{2}\sum_{j=1}^{d_{\leq}}\left(\left[\tfrac{\Lambda_{\leq,j}(X)}{\lambda} + g_j(f(X))\right]_+^2 - \tfrac{\Lambda_{\leq,j}(X)^2}{\lambda^2}\right)\Bigg]. \tag{3}$$

The derivation of this form is given in the supplementary, which extends the classical augmented Lagrangian method (e.g., Nocedal and Wright [1999]) to the functional setting of CSO. The quadratic penalty terms promote feasibility, while the dual variables adaptively adjust to enforce the constraints. The form (18) is similar to those considered in [Park and Hentenryck, 2022, Kotary and Fioretto, 2024], but differ in the design of the penalty term for inequality constraints.

### 2.3 Neural Network Parameterization and Training Algorithm

Solving the functional minimax problem in (2) directly is computationally infeasible, as it involves optimization over infinite-dimensional function spaces. To make the problem tractable, we adopt a finite-dimensional approximation by parameterizing both the decision rule $f(\cdot)$ and the dual variables $\Lambda(\cdot)$ using neural networks $f_{\texttt{NN}}(\cdot;\theta)$ and $\Lambda_{\texttt{NN}}(\cdot;\omega)$, respectively. In particular, we consider the following parameterization based on one-hidden-layer neural networks:

$$f_{\texttt{NN}}(x;\theta) = \frac{1}{N}\sum_{i=1}^{N}\phi_1(x;\theta^i), \quad \Lambda_{\texttt{NN}}(x;\omega) = \frac{1}{N}\sum_{i=1}^{N}\phi_2(x;\omega^i), \tag{4}$$

where $\phi_1$ and $\phi_2$ are fixed nonlinear maps, and $\theta = \{\theta^i\}_{i=1}^N$ and $\omega = \{\omega^i\}_{i=1}^N$ are learnable parameters. This scaling regime, where outputs are normalized by $1/N$, is known as the *mean-field regime* [Chizat and Bach, 2018, Mei et al., 2019, Sirignano and Spiliopoulos, 2019]. It has several advantages: (1) It ensures that the network outputs remain bounded as width $N$ increases, allowing controlled approximations in the infinite-width limit; (2) It gives rise to a natural distributional interpretation, where the empirical measure over parameters converges to a probability distribution in parameter space; (3) Crucially, it enables nontrivial feature learning dynamics during training, often leading to superior performance in practice [Fang et al., 2021]. This is in contrast to the Neural Tangent Kernel regime [Du et al., 2019, Jacot et al., 2020], which uses a different scaling and effectively freezes hidden-layer features at initialization.

The parameters $(\theta, \omega)$ are learned from data using first-order stochastic optimization methods. There are many algorithmic choices. We employ a single-loop noisy stochastic gradient descent-ascent (SGDA) algorithm, which performs gradient descent on the primal variable $\theta$ while performing gradient ascent on the dual variable $\omega$.

**Algorithm 1** ALM-SGDA

1: **Input:** Learning rates $\eta_1, \eta_2$, temperature $\tau > 0$
2: **Initialize:** Parameters $\theta_0, \omega_0$, $k = 0$
3: **while** not converge **do**
4: $\quad \theta_{k+1} \leftarrow \theta_k - \eta_1\left(\widehat{\nabla}_\theta L(\theta_k,\omega_k) + \tau\theta_k\right) + \sqrt{2\eta_1\tau}\cdot\xi_k^1$
5: $\quad \omega_{k+1} \leftarrow \omega_k + \eta_2\left(\widehat{\nabla}_\omega L(\theta_{k+1},\omega_k) - \tau\omega_k\right) + \sqrt{2\eta_2\tau}\cdot\xi_k^2$
6: **end while**
7: Return a solution $\bar{\theta}$ and set $\bar{f} := f_{\texttt{NN}}(x;\bar{\theta})$
8: **Output:** $\hat{f}(x) := \arg\min_{y\in\mathcal{Y}} \|y - \bar{f}(x)\|_2$

The algorithm alternates between updating the primal parameters $\theta$ and the dual parameters $\omega$ using stochastic gradients $\widehat{\nabla}_\theta L(\theta,\omega)$ of the augmented Lagrangian objective $L(\theta,\omega)$. The noise terms $\xi_k^1$ and $\xi_k^2$ are independent Gaussian random variables, which help escape local optima and improve convergence properties. The temperature $\tau$ controls the strength of the noise, and also the weight decay on the parameters. In the mean-field limit, this regularization corresponds to penalizing the

Kullback–Leibler divergence from a standard Gaussian $\mathcal{N}(0, I)$, scaled by a factor of $\tau$. Finally, the output decision rule is obtained by projecting the learned neural network output $\bar{f}(x)$ onto the feasible set $\mathcal{Y}$.

Existing algorithms [Park and Hentenryck, 2022, Hentenryck, 2025] for solving the augmented Lagrangian minimax problem typically adopt a double-loop optimization scheme. In each outer-loop iteration, the algorithm first fixes the dual network $\Lambda_{\texttt{NN}}$ and updates the primal network $f_{\texttt{NN}}$ by solving a inner optimization that minimizes the augmented Lagrangian; it then updates the dual network $\Lambda_{\texttt{NN}}$ through another inner optimization, which minimizes the discrepancy from the instance-wise dual update prescribed by the Euclidean augmented Lagrangian method. In contrast, our approach departs from this nested structure by employing a single-loop stochastic gradient descent-ascent algorithm. This eliminates the need for solving two nested subproblems, significantly reducing computational overhead. Moreover, compared with those projection based methods [Donti et al., 2019, 2021], our method only requires a single projection step at the end of the training process, but does not require explicit projection layers or iterative projection steps, which can be computationally expensive.

## 3 Experiments

In this section, we evaluate the empirical performance of Algorithm 1 (ALM-SGDA) on both synthetic and real-world tasks. Specifically, we apply the algorithm to two constrained contextual decision problems: a multi-item, feature-based newsvendor problem with synthetic data, and a real-world battery arbitrage task. The main results for the battery arbitrage task are presented in the main paper. The results for all other experiments, along with comprehensive details on network architectures, dataset configurations, and hyperparameter settings, are provided in the Supplemental Materials.

We evaluate ALM-SGDA on a grid-scale battery arbitrage task, a real-world problem involving optimal control of battery charging and discharging in response to uncertain electricity prices, as studied in Donti et al. [2019]. The goal is to determine hourly charging ($z_{\text{in}} \in \mathbb{R}^{24}$) and discharging ($z_{\text{out}} \in \mathbb{R}^{24}$) levels over a 24-hour horizon to maximize profit from price fluctuations. The objective also incorporates two regularization terms: one that encourages the battery to remain near half its total capacity $B$ (weighted by $\lambda$), and another that penalizes rapid charging or discharging to promote battery longevity (weighted by $\varepsilon < \lambda$).

This task is formulated as a stochastic program subject to various operational constraints, including charging efficiency $\gamma_{\text{eff}}$, rate limits $c_{\text{in}}$ and $c_{\text{out}}$, and state-of-charge bounds $B$ ensuring the battery remains within the safe charge levels. We use the same dataset as Donti et al. [2019], based on six years of historical grid data, with $B = 1$, $\gamma_{\text{eff}} = 0.9$, $c_{\text{in}} = 0.5$ and $c_{\text{out}} = 0.2$. To test the out-of-sample performance and ensure a direct and rigorous comparison, we re-evaluated the RMSE net and Task-based net baselines from Donti et al. [2019] using the same 20/80 train-test split as our ALM-SGDA model. A detailed mathematical formulation is provided in the Supplemental Material.

## 4 Conclusion

This paper has presented a novel single-loop stochastic optimization framework for solving constrained problems through their Lagrangian dual formulations. The proposed method fundamentally differs from conventional augmented Lagrangian approaches by employing sequential gradient-based updates of both primal and dual variables, thereby circumventing the computational bottlenecks and convergence challenges associated with existing nested-loop optimization algorithms. From a theoretical perspective, we have derived key performance guarantees that collectively characterize the algorithm's behavior, these theoretical results are established under standard assumptions and are supported by rigorous mathematical analysis. The practical efficacy of our approach has been thoroughly validated through comprehensive numerical experiments on representative constrained decision-making problems. Several promising research directions emerge from this work. For example, the extension to problems with finite-size neural network instead of mean-field limits is left for future work.

## References


Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pages 136–145. PMLR, 2017.

Gah-Yi Ban and Cynthia Rudin. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67(1):90–108, 2019.

Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020.

Dimitris Bertsimas and Nihal Koduri. Data-driven optimization: A reproducing kernel hilbert space approach. *Operations Research*, 70(1):454–471, 2022.

Bingqing Chen, Priya L Donti, Kyri Baker, J Zico Kolter, and Mario Bergés. Enforcing policy feasibility constraints through differentiable projection for energy optimization. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, pages 199–210, 2021.

Wenbo Chen, Mathieu Tanneau, and Pascal Van Hentenryck. End-to-End Feasible Optimization Proxies for Large-Scale Economic Dispatch, August 2023. URL `http://arxiv.org/abs/2304.11726`. arXiv:2304.11726.

Abhilash Reddy Chenreddy, Nymisha Bandi, and Erick Delage. Data-driven conditional robust optimization. *Advances in Neural Information Processing Systems*, 35:9525–9537, 2022.

Lenaic Chizat and Francis Bach. On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport, October 2018. URL `http://arxiv.org/abs/1805.09545`. arXiv:1805.09545 [cs, math, stat].

Lénaïc Chizat. Mean-Field Langevin Dynamics: Exponential Convergence and Annealing, August 2022. URL `http://arxiv.org/abs/2202.01009`. arXiv:2202.01009 [math].

Rares Cristian, Pavithra Harsha, Georgia Perakis, Brian L Quanz, and Ioannis Spantidakis. End-to-end learning via constraint-enforcing approximators for linear programs with applications to supply chains. In *AI for Decision Optimization Workshop of the AAAI Conference on Artificial Intelligence*, 2022.

Priya Donti, Brandon Amos, and J Zico Kolter. Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems*, 30, 2017.

Priya L. Donti, Brandon Amos, and J. Zico Kolter. Task-based End-to-end Model Learning in Stochastic Optimization, April 2019. URL `http://arxiv.org/abs/1703.04529`. arXiv:1703.04529 [cs].

Priya L. Donti, David Rolnick, and J. Zico Kolter. DC3: A learning method for optimization with hard constraints, April 2021. URL `http://arxiv.org/abs/2104.12225`. arXiv:2104.12225 [cs].

Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient Descent Provably Optimizes Over-parameterized Neural Networks, February 2019. URL `http://arxiv.org/abs/1810.02054`. arXiv:1810.02054 [cs, math, stat].

Othman El Balghiti, Adam N Elmachtoub, Paul Grigas, and Ambuj Tewari. Generalization bounds in the predict-then-optimize framework. *Advances in neural information processing systems*, 32, 2019.

Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68(1): 9–26, 2022.

Cong Fang, Hanze Dong, and Tong Zhang. Mathematical Models of Overparameterized Neural Networks. *Proceedings of the IEEE*, 109(5):683–703, May 2021. ISSN 0018-9219, 1558-2256. doi: 10.1109/JPROC.2020.3048020. URL http://arxiv.org/abs/2012.13982. arXiv:2012.13982 [cs].

Pascal Van Hentenryck. Optimization Learning, January 2025. URL http://arxiv.org/abs/2501.03443. arXiv:2501.03443 [math].

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks, February 2020. URL http://arxiv.org/abs/1806.07572. arXiv:1806.07572 [cs, math, stat].

Juno Kim and Taiji Suzuki. Transformers learn nonlinear features in context. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

Juno Kim, Kakei Yamamoto, Kazusato Oko, Zhuoran Yang, and Taiji Suzuki. Symmetric Mean-field Langevin Dynamics for Distributional Minimax Problems, February 2024. URL http://arxiv.org/abs/2312.01127. arXiv:2312.01127 [math].

Lingkai Kong, Jiaming Cui, Yuchen Zhuang, Rui Feng, B Aditya Prakash, and Chao Zhang. End-to-end stochastic optimization with energy-based model. *Advances in Neural Information Processing Systems*, 35:11341–11354, 2022.

James Kotary and Ferdinando Fioretto. Learning Constrained Optimization with Deep Augmented Lagrangian Methods, March 2024. URL http://arxiv.org/abs/2403.03454. arXiv:2403.03454 [cs].

Zhangyi Liu, Feng Liu, Rui Gao, and Shuang Li. Convergence of mean-field langevin stochastic descent-ascent for distributional minimax optimization. 2025. URL https://optimization-online.org/?p=30420.

Liwan H Liyanage and J George Shanthikumar. A practical inventory control policy using operational statistics. *Operations research letters*, 33(4):341–348, 2005.

Yulong Lu. Two-Scale Gradient Descent Ascent Dynamics Finds Mixed Nash Equilibria of Continuous Games: A Mean-Field Perspective, January 2023. URL http://arxiv.org/abs/2212.08791. arXiv:2212.08791 [cs, math, stat].

Andreas Maurer. A vector-contraction inequality for Rademacher complexities, May 2016. URL http://arxiv.org/abs/1605.00251. arXiv:1605.00251.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit, February 2019. URL http://arxiv.org/abs/1902.06015. arXiv:1902.06015 [cond-mat, stat].

Atsushi Nitanda, Denny Wu, and Taiji Suzuki. Convex Analysis of the Mean Field Langevin Dynamics, February 2022. URL http://arxiv.org/abs/2201.10469. arXiv:2201.10469 [cs, math, stat].

Jorge Nocedal and Stephen J. Wright, editors. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, 1999. ISBN 978-0-387-98793-4. doi: 10.1007/b98874. URL http://link.springer.com/10.1007/b98874.

Pascal M Notz and Richard Pibernik. Prescriptive analytics for flexible capacity management. *Management Science*, 68(3):1756–1775, 2022.

Afshin Oroojlooyjadid, Lawrence V Snyder, and Martin Takáč. Applying deep learning to the newsvendor problem. *Iise Transactions*, 52(4):444–463, 2020.

Jianming Pan, Zeqi Ye, Xiao Yang, Xu Yang, Weiqing Liu, Lewen Wang, and Jiang Bian. Bpqp: A differentiable convex optimization framework for efficient end-to-end learning. *Advances in Neural Information Processing Systems*, 37:77468–77493, 2024.

Seonho Park and Pascal Van Hentenryck. Self-Supervised Primal-Dual Learning for Constrained Optimization, November 2022. URL http://arxiv.org/abs/2208.09046. arXiv:2208.09046 [cs].

Meng Qi and Zuo-Jun Shen. Integrating prediction/estimation and optimization with applications in operations management. In *Tutorials in operations research: emerging and impactful topics in operations*, pages 36–58. INFORMS, 2022.

Meng Qi, Yuanyuan Shi, Yongzhi Qi, Chenxin Ma, Rong Yuan, Di Wu, and Zuo-Jun Shen. A practical end-to-end inventory management model with deep learning. *Management Science*, 69 (2):759–773, 2023.

Yves Rychener, Daniel Kuhn, and Tobias Sutter. End-to-end learning for stochastic optimization: A bayesian perspective. In *International Conference on Machine Learning*, pages 29455–29472. PMLR, 2023.

Utsav Sadana, Abhilash Chenreddy, Erick Delage, Alexandre Forel, Emma Frejinger, and Thibaut Vidal. A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, 320(2):271–289, 2025.

Justin Sirignano and Konstantinos Spiliopoulos. Mean Field Analysis of Neural Networks: A Law of Large Numbers, November 2019. URL http://arxiv.org/abs/1805.01053. arXiv:1805.01053 [math].

Chunlin Sun, Linyu Liu, and Xiaocheng Li. Predict-then-calibrate: A new perspective of robust contextual lp. *Advances in Neural Information Processing Systems*, 36:17713–17741, 2023a.

Haixiang Sun, Ye Shi, Jingya Wang, Hoang Duong Tuan, H. Vincent Poor, and Dacheng Tao. Alternating Differentiation for Optimization Layers, April 2023b. URL http://arxiv.org/abs/2210.01802. arXiv:2210.01802 [cs].

Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1658–1665, 2019.

Yumo Xu and Shay B Cohen. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1979, 2018.

Junchi Yang, Negar Kiyavash, and Niao He. Global Convergence and Variance-Reduced Optimization for a Class of Nonconvex-Nonconcave Minimax Problems, February 2020. URL http://arxiv.org/abs/2002.09621. arXiv:2002.09621 [cs, math, stat].

Chao Zhang, Zihao Zhang, Mihai Cucuringu, and Stefan Zohren. A universal end-to-end approach to portfolio optimization via deep learning. *arXiv preprint arXiv:2111.09170*, 2021.

Luhao Zhang, Jincheng Yang, and Rui Gao. Optimal robust policy for feature-based newsvendor. *Management Science*, 70(4):2315–2329, 2024.

# A Related Literature

**Contextual Stochastic Optimization.** The central goal of contextual stochastic optimization is to learn a mapping from observed contexts to optimal actions. Initiated in Liyanage and Shanthikumar [2005] and popularized in Ban and Rudin [2019] for big data environments, decision rule optimization is an end-to-end formulation that directly minimizes the (regularized) empirical loss over a decision rule space. The commonly-used decision rules include linear decision rules [Ban and Rudin, 2019], RKHS based decision rules [Bertsimas and Koduri, 2022, Notz and Pibernik, 2022], nonlinear decision rules such as neural networks [Qi et al., 2023, Rychener et al., 2023, Kim and Suzuki, 2024], and distributionally robust decision rules [Chenreddy et al., 2022, Zhang et al., 2024]. Once trained, the decision rule provides an immediate mapping from any given context to an action.

Two other prominent approaches are sequential learning and optimization (SLO) and integrated learning and optimization (ILO). SLO is a classical two-stage procedure that first estimates the conditional distribution and then optimizes for the conditional expectation [Bertsimas and Kallus, 2020, Sun et al., 2023a]. However, as noted in Ban and Rudin [2019], this method can be problematic in high-dimensional settings, where accurately estimating the conditional distribution is difficult and errors in the first stage may propagate and amplify during optimization. ILO addresses this by directly searching for a predictive model that minimizes decision loss rather than estimation loss [Donti et al., 2017, Wilder et al., 2019, Qi and Shen, 2022]. While ILO retains the interpretability of the SLO paradigm, it typically leads to a nonconvex learning problem that is challenging to solve [El Balghiti et al., 2019, Elmachtoub and Grigas, 2022]. Common training techniques include ensembling, implicit differentiation, and the use of surrogate losses or optimizers; see Sadana et al. [2025] for a comprehensive overview. Notably, both SLO and ILO require estimating a conditional distribution and solving an optimization problem for each context separately. In contrast, decision rule optimization bypasses distribution estimation entirely and marginalizes the conditional problems during training, leading to faster decision-making at inference time.

**Handling Constraints in Neural Network.** A foundational contribution to differentiable optimization is OptNet by Amos and Kolter [2017], which embeds quadratic programs (QPs) as differentiable layers using the KKT conditions and implicit differentiation, enabling constraint-aware end-to-end training Donti et al. [2019]. To generalize beyond QPs, Agrawal et al. [2019] introduce cvxpylayers, which compile convex programs into differentiable layers via conic reformulation and homogeneous self-dual embedding, expanding the scope of differentiable convex optimization. To address the computational burden of differentiating through KKT systems, Sun et al. [2023b] propose Alt-Diff, which uses ADMM for parameterized convex problems with polyhedral constraints, leveraging separability for modular architectures.

Donti et al. [2021] present DC3, a hybrid method that satisfies equality constraints via implicit neural networks and enforces inequality constraints through gradient correction—maintaining feasibility without relying on explicit solvers. Kong et al. [2022] propose an energy-based model to handle constraints in an end-to-end learning framework. Recently, Pan et al. [2024] propose BPQP, which reformulates the backward pass in constrained learning as a standalone QP, enabling efficient gradients via first-order methods without unrolling solvers.

Complementing these methods, Park and Hentenryck [2022] introduce Primal-Dual Learning (PDL), a self-supervised framework that jointly trains primal and dual networks to mimic augmented Lagrangian dynamics, while Kotary and Fioretto [2024] develop a dual ascent approach for solving the augmented Lagrangian.

# B Derivation of the Augmented Lagrangian

We first deal with the constraint on the function $f$. Introduce a slack variable (function) $s$ such that the inequality constraints in $\mathcal{Y}$ becomes an equality, *i.e.*,

$$\mathcal{Y} = \{y : \exists s \geq 0, \text{s.t.}, g(y) + s = 0,\ Ay = b\}.$$

The Lagrangian function of problem (1) is given by

$$\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(f(X), Z) + \Lambda_{=}(X)^{\mathsf{T}}(Af(X) - b) + \Lambda_{\leq}(X)^{\mathsf{T}}(g(f(X)) + s(X))], \tag{5}$$

where $s : \mathcal{X} \mapsto \mathbb{R}_{d_\leq}$. Let $\lambda$ be a nonnegative penalty parameter. Then the augmented Lagrangian function of problem (1) is given by

$$\begin{aligned} L(f, \lambda, \Lambda) = \mathbb{E}_{\hat{\mathcal{D}}}\Big[\Psi(f(X), Z) + \Lambda_{=}(X)^{\mathsf{T}}(Af(X) - b) + \Lambda_{\leq}(X)^{\mathsf{T}}(g(f(X)) + s(X)) \\ + \frac{\lambda}{2}\big(\|Af(X) - b\|_2^2 + \|g(f(X)) + s(X)\|_2^2\big)\Big]. \end{aligned} \tag{6}$$

For the augmented Lagrangian method, given the Lagrangian multiplier $\lambda$ and $\Lambda$, we need to solve the following optimization problem to update $f$ and $s$:

$$\min_{f(\cdot),\, s(\cdot)\geq 0} L(f, \lambda, \gamma).$$

Given $f(\cdot)$, the above problem is reduced to solve

$$\begin{aligned} &\min_{s(\cdot)\geq 0} \mathbb{E}_{\hat{\mathcal{D}}}\big[\Lambda_{\leq}(X)^{\mathsf{T}}(g(f(X)) + s(X)) + \frac{\lambda}{2}\|g(f(X)) + s(X)\|_2^2\big] \\ =&\mathbb{E}_{\hat{\mathcal{D}}}\Big[\min_{s\geq 0}\big\{\Lambda_{\leq}(X)^{\mathsf{T}}(g(f(X)) + s(X)) + \frac{\lambda}{2}\|g(f(X)) + s(X)\|_2^2\big\}\Big], \end{aligned}$$

where the equality comes from the interchangeability principle. Hence, the optimal solution is given by

$$s^\star(x) = \max\Big\{-\frac{\Lambda_{\leq}(x)}{\rho} - g(f(x)), 0\Big\},\ \ \forall x.$$

Then the augmented Lagrangian function becomes

$$\begin{aligned} L(f, \Lambda) = \mathbb{E}_{\hat{\mathcal{D}}}\Big[\Psi(f(X), Z) + \Lambda_{=}(X)^{\mathsf{T}}(Af(X) - b) + \frac{\lambda}{2}\|Af(X) - b\|_2^2 \\ + \frac{\lambda}{2}\sum_{i\in[d_\leq]}\Big(\max\Big\{\frac{\Lambda_{\leq,i}(X)}{\lambda} + g_i(f(X)), 0\Big\}^2 - \frac{\Lambda_{\leq,i}^2(X)}{\lambda^2}\Big)\Big]. \end{aligned} \tag{7}$$

Given the simplified augmented Lagrangian function, classical ALM then leverages the KKT condition to perform the updates on $f$, $\lambda$ and $\gamma$. Specifically, it is a double-loop algorithm, which requires an oracle to solve subproblems up to certain accuracy by tuning multiple hyperparameters at each iteration. In the following, we directly tackle the functional minimax optimization problem

$$\min_f \max_\Lambda L(f, \Lambda), \tag{8}$$

and try to develop a single-loop algorithm to solve it efficiently.

# C Performance Guarantees

In this section, we establish theoretical performance guarantees for Algorithm 1 in the mean-field regime, where the number of neurons $N$ in the neural network tends to infinity. In this limit, the empirical parameter distributions converge to probability measures. Accordingly, the function spaces become $\mathcal{F} = \{\mathbb{E}_\mu[\phi_1(x;\cdot)] : \mu \in \mathcal{P}(\mathbb{R}^d)\}$ and $\mathcal{A} = \{\mathbb{E}_\nu[\phi_2(x;\cdot)] : \nu \in \mathcal{P}(\mathbb{R}^d)\}$, where $\mu$ and $\nu$ denote distributions over the parameter spaces of the primal and dual networks, respectively, and we use $d$ to represent the dimension of their parameters for simplicity. Under this representation, the noisy SGDA algorithm for solving (2) converges to the following entropy-regularized problem [Chizat, 2022, Kim et al., 2024, Nitanda et al., 2022, Lu, 2023]

$$\min_{\mu\in\mathcal{P}(\mathbb{R}^d)}\max_{\nu\in\mathcal{P}(\mathbb{R}^d)} E_\tau(\mu, \nu) := L\big(\mathbb{E}_\mu[\phi_1(X;\cdot)], \mathbb{E}_\nu[\phi_2(X;\cdot)]\big) + \tau\mathsf{KL}(\mu\|\mu_0) - \tau\mathsf{KL}(\nu\|\nu_0), \tag{9}$$

where $\mu_0, \nu_0$ are standard Gaussian distributions, $L$ is the ALM objective (18), and $\mathsf{KL}$ denotes the relative entropy.

To facilitate the analysis, we make the following assumptions that are standard in the literature.

**Assumption 1** (Realizibility)**.** The optimal decision rule of (1) can be represented as $f(x) = \mathbb{E}_\mu[\phi_1(x;\theta)]$ for some $\mu \in \mathcal{P}(\mathbb{R}^d)$, and the corresponding Lagrange multiplier as $\Lambda(x) = \mathbb{E}_\nu[\phi_2(x;\omega)]$ for some $\nu \in \mathcal{P}(\mathbb{R}^d)$.

This assumption ensure that the optimal primal-dual pair for the empirical constrained problem (1) can be represented by some neural networks in the space $\mathcal{F} \times \mathcal{A}$.

**Assumption 2.** The cost function $\Psi(\cdot, z)$ is convex and Lipchitz for any $z$. The constraint function $g(\cdot)$ is convex and Lipschitz. The projection at the last step of Algorithm 1 has non-degenerate KKT conditions.

This regularity condition ensures control over both the constraint violation and the suboptimality of the solution.

In what follows, we provide a performance bound on the suboptimality of the learned decision rule produced by Algorithm 1. Let $f_\star$ denote the optimal policy for the true population problem $\min_{f\in\mathcal{F}} \mathbb{E}_{(X,Z)\sim\mathcal{D}}[\Psi(f(X), Z)]$, and let $\hat{f}_\star$ denote the optimal solution of the empirical problem (1). Consider the following decomposition of the suboptimality gap:

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}}[\Psi(\hat{f}(X), Z)] - \mathbb{E}_{\mathcal{D}}[\Psi(f_\star(X), Z)] \\
&= \big(\mathbb{E}_{\mathcal{D}}[\Psi(\hat{f}(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}(X), Z)]\big) + \big(\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\bar{f}(X), Z)]\big) \\
&\quad + \big(\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\bar{f}(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}_\star(X), Z)]\big) + \big(\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}_\star(X), Z)] - \mathbb{E}_{\mathcal{D}}[\Psi(f_\star(X), Z)]\big).
\end{aligned} \tag{10}$$

Here on the right-hand side: the first term represents the generalization error of the algorithm output $\hat{f}$, which arises due to finite-sample approximation of the true distribution $\mathcal{D}$ by the empirical distribution $\hat{\mathcal{D}}$; the second term captures the projection error between $\bar{f}$ and its projection $\hat{f}$, reflecting the cost of enforcing feasibility via post-hoc projection; the third term accounts for the optimization and regularization error, resulting from the relative entropy penalty and approximate convergence of the algorithm to a regularized saddle point; and the last term is the difference between empirical optimal value and true optimal value.

In the following analysis, we bound each of the four terms in (10).

### C.1 Generalization Error Bound

To mitigate overfitting, in Algorithm 1, we inject Gaussian noise and add weight decay during training—or equivalently, apply entropy regularization in the mean-field formulation—which helps improve generalization. Let $\hat{\mathcal{D}}_M = \{(\boldsymbol{x}^m, \boldsymbol{z}^m)\}_{m=1}^M$ denote a dataset of $M$ i.i.d. samples drawn from the underlying distribution $\mathcal{D}$. The empirical cost associated with this dataset is given by $\mathbb{E}_{\hat{\mathcal{D}}_M}[\Psi(f(X), Z)]$, while the generalization gap of a decision rule $f$ is defined as the gap between its population and empirical risks:

$$\mathbb{E}_{\mathcal{D}}[\Psi(f(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}_M}[\Psi(f(X), Z)].$$

We have the following bound on the generalization gap.

**Proposition 1.** *Let $f$ be the output of Algorithm 1, then its generalization error satisfies*

$$\left|\mathbb{E}_{\mathcal{D}}[\Psi(f(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}_M}[\Psi(f(X), Z)]\right| = \mathcal{O}\left(\sqrt{\frac{d_{\mathcal{Y}}}{\tau M}} + \sqrt{\frac{\log(1/\delta)}{M}}\right) \tag{11}$$

*with probability $1-\delta$.*

The inverse relation with $\tau$ is since the KL-divergence of $f$ satisfies $\mathsf{KL}(f\|f_0) = \mathcal{O}(1/\tau)$ for a fixed $f_0$. The proof relies on the vector contraction inequality for Rademacher complexity [Maurer, 2016], along with relative entropy-based control of the hypothesis class.

### C.2 Regularization Bias

While entropic regularization enhances optimization stability and generalization, it also introduces bias into the optimal solution. To quantify this regularization-induced bias, we first recall the Lyapunov functions that are commonly used in the analysis of minimax problems [Yang et al., 2020, Lu, 2023, Kim et al., 2024, Liu et al., 2025], defined as

$$L_1(\mu) := \max_{\nu'\in\mathcal{P}(\mathbb{R}^d)} E_\tau(\mu, \nu') - \min_{\mu'\in\mathcal{P}(\mathbb{R}^d)} \max_{\nu'\in\mathcal{P}(\mathbb{R}^d)} E_\tau(\mu', \nu'). \tag{12}$$

$$L_2(\mu, \nu) := \max_{\nu'\in\mathcal{P}(\mathbb{R}^d)} E_\tau(\mu, \nu') - E_\tau(\mu, \nu). \tag{13}$$

The function $L_1(\mu)$ measures the optimality gap of solution $\mu$ relative to the best possible regularized solution, while $L_2(\mu, \nu)$ quantifies the dual suboptimality at a given pair $(\mu, \nu)$ with respect to the regularized problem. The convergence of mean-field SGDA has been established in prior work. Building on the results therein [Lu, 2023, Liu et al., 2025], we assume that Algorithm 1 returns a solution $\bar{f}$ whose associated Lyapunov function satisfies $L_1(\bar{\mu}) \leq \epsilon$. A detailed verification of this assumption is provided in the supplemental material. We remark that our analysis is not limited to the SGDA algorithm in Algorithm 1; it applies to any optimization procedure that yields a distribution $\mu$ satisfying $L_1(\mu) \leq \epsilon$, although our Algorithm 1 demonstrates nice empirical convergence.

To isolate the effect of entropic regularization, we define the unregularized counterparts of the Lyapunov functionals by setting $\tau = 0$, denoted $L_1^0$ and $L_2^0$. The functional $L_2^0(\mu, \nu)$ serves as a certificate of feasibility: it vanishes for all $\nu$ if and only if the corresponding decision rule $f$ satisfies the constraints exactly. Meanwhile, $L_1^0(\mu)$ quantifies the optimality gap in the original, unregularized problem. Since regularization perturbs the objective, it introduces a bias between the regularized and unregularized solutions. The following key lemma quantifies this discrepancy by bounding the difference between the regularized and unregularized Lyapunov values.

**Lemma 1.** *It holds uniformly for every $(\mu, \nu)$ that*

$$L_1^0(\mu) - L_1(\mu) \leq 2\tau d \log(2\pi),$$
$$L_2^0(\mu, \nu) - L_2(\mu, \nu) \leq \tau d \log(2\pi).$$

This result shows that that any convergence guarantee for $L_1$ directly translates into a bound on the unregularized Lyapunov function $L_1^0$, up to an additive $\mathcal{O}(\tau)$ bias. Using this lemma, we obtain a bound on the regularization bias for the empirical problem.

**Proposition 2.** *Suppose the Lyapunov function associated with $\bar{f}$ satisfies $L_1(\bar{\mu}) \leq \epsilon$. Then*

$$\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\bar{f}(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}_\star(X), Z)] \leq \epsilon + 2\tau d \log(2\pi).$$

### C.3 Projection Error Bound

While the regularized objective improves optimization and generalization, the solution returned by the algorithm may not exactly satisfy the hard constraints due to both the entropic regularization and the use of a finite number of optimization steps. As a result, we apply a post-hoc projection step to ensure feasibility in the last step of Algorithm 1. This projection introduces an additional error, which we now quantify.

**Proposition 3.** *Under the setup of Proposition 2, we further have that*

$$\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\bar{f}(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}(X), Z)] = \mathcal{O}\left(\sqrt{\epsilon + 2d\tau \log(2\pi)}\right).$$

The proof follows a similar strategy to Proposition 2, where we first bound the magnitude of constraint violation via the Lyapunov functional $L_1^0$, and then quantify the projection error induced by enforcing feasibility.

### C.4 Optimality Gap between the Unregularized Empirical and True Problems

To bound the last term in (10), we exploit the Rademacher complexity bound in Proposition 1 as well as the gap between the regularized and the unregularized problems.

**Proposition 4.** *With probability $1 - \delta$, it holds that*

$$\left|\mathbb{E}_{\mathcal{D}}[\Psi(f_\star(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}_M}[\Psi(\hat{f}_\star(X), Z)]\right| = \mathcal{O}\left(\sqrt{\frac{d_{\mathcal{Y}}}{\tau M}} + \sqrt{\frac{\log(1/\delta)}{M}}\right) + \mathcal{O}(\epsilon + d\tau).$$

### C.5 Adding All Together

Finally, combining the bounds above and using (10), we obtain the out-of-sample performance gap for the output $\hat{f}$ of our algorithm:

$$\mathbb{E}_{\mathcal{D}}[\Psi(\hat{f}(X), Z)] - \mathbb{E}_{\mathcal{D}}[\Psi(f_\star(X), Z)] \leq \mathcal{O}\left(\sqrt{\frac{d_{\mathcal{Y}}}{\tau M}}\right) + \mathcal{O}\left(\sqrt{\frac{\log(1/\delta)}{M}}\right) + \mathcal{O}(\sqrt{\epsilon + 2d\tau}).$$

Thus, we establish a high-probability performance guarantee for the output $\hat{f}$ of Algorithm 1, showing that it achieves a suboptimality gap that scales with the inverse square root of the sample size $M$, and is further controlled by the regularization parameter $\tau$ and the optimization error $\epsilon$, as measured by the Lyapunov potential.

# D Experiments

In this section, we evaluate the empirical performance of Algorithm 1 (ALM-SGDA) on both synthetic and real-world tasks. Specifically, we apply the algorithm to two constrained contextual decision problems: a multi-item, feature-based newsvendor problem with synthetic data, and a real-world battery arbitrage task. Additional experimental results, along with detailed network architecture design, dataset configuration, and hyperparameter settings, are provided in the Supplemental Materials.

## D.1 Multi-item Feature-based Newsvendor with Constraints

We first consider a constrained multi-item feature-based newsvendor problem, adapted from Ban and Rudin [2019], which extends the classical inventory model to a data-driven setting involving multiple products and observable contextual features. In this formulation, let $J$ denote the number of products. The objective is to determine the optimal order quantity $Y_j = f_j(X)$ for each product $j = 1, \ldots, J$, in order to minimize the expected cost under stochastic demand $Z$, which depends on an observed context $X$. The order quantities must satisfy a linear resource constraint of the form $Af(X) \leq C$ for all feature realizations $X \in \mathcal{X}$, where $A$ is a resource consumption matrix and $C$ is a capacity vector. To define the objective, let $b_j$ and $h_j$ be the back-order and holding penalty coefficients, respectively. We adopt a squared penalty function—following Donti et al. [2019], Oroojlooyjadid et al. [2020]—which promotes smoothness and facilitates efficient optimization. Denote $(\cdot)_+ = \max\{\cdot, 0\}$. The constrained multi-item newsvendor problem is then given by:

$$\begin{aligned} \min_{f(\cdot):\mathcal{X}\to\mathbb{R}^J} \quad & \Psi(f) := \mathbb{E}_{X,Z}\left[\frac{1}{J}\sum_{j=1}^{J}\left(b_j(Z_j - f_j(X))_+ + h_j(f_j(X) - Z_j)_+\right)^2\right] \\ \text{subject to} \quad & Af(X) \leq C, \quad \forall X \in \mathcal{X}, \end{aligned} \tag{14}$$

We assume that the feature vector $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$, and the conditional distribution of $Z$ for any feature $X$ follows

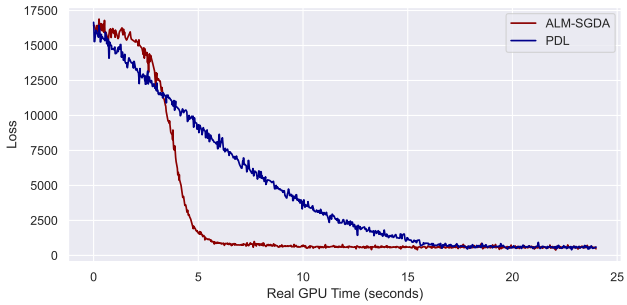
$$Z|X \sim \beta^\mathsf{T} X + \varepsilon,$$

where $\beta$ is the weight parameter and $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is the noise term independent of the feature $X$ [Ban and Rudin, 2019]. The specific parameter values are detailed in the Supplemental Materials.

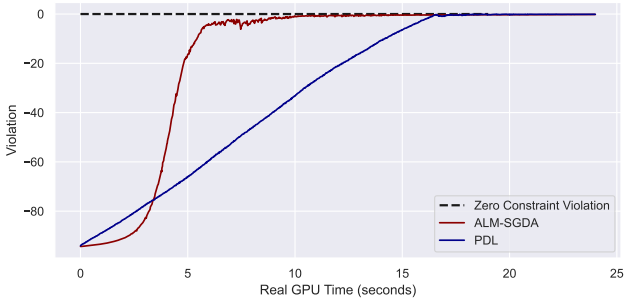
Given the structural similarities between PDL and our proposed ALM-SGDA—both of which solve augmented Lagrangian problems using joint training of primal and dual networks (in double-loop and single-loop formulations, respectively)—we compare their empirical performance in terms of convergence speed and solution quality. Note that only in this experiement, we do not perform a projection at Step 8 of Algorithm 1, so as to make a direct comparison with PDL.

Figure 1 presents a side-by-side comparison of ALM-SGDA (red) and PDL (blue) across two key metrics: training loss (Figure 1(a)) and mean constraint violation (Figure 1(b)), both measured against real GPU time (in seconds). In Figure 1(a), PDL shows a steeper decline in training loss during the initial phase (for $t < 5$ sec), but ALM-SGDA quickly overcomes its burn-in period and converges more rapidly thereafter. Figure 1(b) evaluates constraint satisfaction by plotting the mean constraint violation relative to a zero threshold (indicated by the black dashed line), illustrating how each method maintains feasibility over time. A similar pattern is observed in Figure 1(b) for constraint violation: ALM-SGDA, after a short burn-in phase, decreases constraint violations faster. These illustrate the stability and effectiveness of our proposed algorithm.

Table 1 reports the average out-of-sample relative optimality gap of the solutions from ALM-SGDA and PDL onto the feasible region. We compute the optimality gap based on the optimal solution given the instance. The true optimal solution is computed by the Gurobi solver, individually for each context. The out-of-sample relative optimality gap in percentage is the average value of the optimality gaps over the test instances, i.e. $\frac{|\Psi(y^*)-\Psi(y)|}{|\Psi(y^*)|}$ where $y^*$ is the global optimal solution from




(a) Training Loss vs Real GPU Time




(b) Mean constraint violation vs Real GPU Time

Figure 1: The training loss $\Psi(f)$ and mean constraint violation

Table 1: Optimality gaps of ALM-SGDA (Ours) and PDL

| Noise scale | Method | Opt. Gap (%) |
|---|---|---|
| $\sigma = 0.1$ | ALM-SGDA | **1.25** (0.000) |
| | PDL | 2.03 (0.000) |
| $\sigma = 0.5$ | ALM-SGDA | **7.52** (0.001) |
| | PDL | 8.30 (0.000) |
| $\sigma = 0.8$ | ALM-SGDA | **17.46** (0.001) |
| | PDL | 18.06 (0.001) |

the solver and $y$ is the decision output predicted by the model. Since we do not report constraint violation as a separate metric, for PDL, we perform the same projection as in Step 8 of Algorithm 1 to enforce feasibility as well. We vary the noise scale in $\{0.1, 0.5, 0.8\}$. We observe that ALM-SGDA consistently outperforms PDL across varying noise scales. In addition, we compare ALM-SGDA against several other benchmarks—including OptNet [Amos and Kolter, 2017], cvxpylayers [Agrawal et al., 2019], Alt-diff [Sun et al., 2023b] and BPQP [Pan et al., 2024]—in the Supplemental Material. These results show that ALM-SGDA significantly reduces computational overhead while providing strong guarantees on convergence and generalization.

Table 2: Task loss under different configurations

| Configuration $\lambda$ | $\epsilon$ | RMSE net | Task-based net | ALM-SGDA (Ours) |
|---|---|---|---|---|
| 35 | 15 | 208.57(43.1) | 199.69(14.5) | **167.13**(9.54) |
| 10 | 5 | 153.23(175.8) | 121.54(16.2) | **52.3**(8.15) |
| 1 | 0.5 | 26.17(19.5) | 18.56(11.9) | **2.15**(4.26) |
| 0.1 | 0.05 | 5.94(7.8) | $-0.81$(4.9) | **-2.91**(2.52) |

Table 2 illustrates the performance of ALM-SGDA under four distinct hyperparameter configurations, alongside another two baseline models–RMSE net and Task-based net–considered in Donti et al. [2019]. In all settings, ALM-SGDA consistently achieves lower task loss, indicating superior average performance compared to both benchmarks. Notably, ALM-SGDA is computationally more efficient thanks to its projection-free iterations (expect for its final step). Comparison of computational time with other projection-based approaches are supported in the Supplemental Material.

# E Omitted Proofs

We first rewrite Assumption 2 with explicit constants.

**Assumption 2.** The cost function $\Psi(\cdot, z)$ is convex and $\ell$-Lipchitz for any $z$. The constraint function $g(\cdot)$ is convex and $G$-Lipschitz. The projection at the last step of Algorithm 1 has non-degenerate KKT conditions, which means $\nabla g_i$ is invertible and $\|\nabla g_i(\cdot)\|_2 > m > 0$ on a neighborhood of $y$ with $g_i(y) = 0$.

### E.1 Proofs for Section C.1

The following vector contraction lemma [Maurer, 2016] is useful in our anlaysis.

**Lemma 2.** *Let $\mathcal{Z}$ be any set, $(z_1, \ldots, z_n) \in \mathcal{Z}^n$. Let $\mathcal{F}$ be a class of functions $f : \mathcal{Z} \to \mathbb{R}^\kappa$ and $h_i : \mathbb{R}^\kappa \to \mathbb{R}$ be L-Lipschitz continuous, $i = 1, \ldots, n$. Then*

$$\mathbb{E}_\epsilon \left[ \sup_{f \in \mathcal{F}} \sum_{i=1}^n \epsilon_i h_i(f(z_i)) \right] \leq \sqrt{2} L \mathbb{E}_\epsilon \left[ \sup_{f \in \mathcal{F}} \sum_{i=1}^n \sum_{k=1}^\kappa \epsilon_{ik} f_k(z_i) \right],$$

*where $\{\epsilon_i\}_{i=1,\ldots,n}$ and $\{\epsilon_{ik}\}_{i=1,\ldots,n;k=1,\ldots,\kappa}$ are two independent Rademacher sequences.*

Motivated by the bound on the right-hand side, we define for a function class $\mathcal{F}$ mapping from $\mathcal{X}$ to $\mathcal{Y}$:

$$\mathfrak{R}_{\hat{\mathcal{D}}_M}(\mathcal{F}) := \mathbb{E}_\epsilon \left[ \sup_{f \in \mathcal{F}} \frac{1}{M} \sum_{i=1}^M \sum_{s=1}^{d_\mathcal{Y}} \epsilon_{ms} f^s(x^m) \right], \tag{15}$$

where $\epsilon_{ms}$ be independent Rademacher random variables. Then we have the following result.

**Lemma 3.** *Let $\mathcal{F} = \{x \mapsto \mathbb{E}_\rho[\sigma(x; \cdot)] : \rho \in \mathcal{P}(\mathbb{R}^d),\ \mathsf{KL}(\rho \| \rho_0) \leq B\}$, where $\rho_0$ is a fixed distribution, and $\sigma : \mathcal{X} \to \mathcal{Y}$ with $\|\sigma(x; \cdot)\|_\infty \leq C_0$, then*

$$\mathfrak{R}_{\hat{\mathcal{D}}_M}(\mathcal{F}) \leq \sqrt{\frac{2 d_\mathcal{Y} C_0^2 B}{M}}.$$

*Proof of Lemma 3.* We bound the offset Rademacher complexity

$$\mathfrak{R}^\lambda(\mathcal{F}) := \mathbb{E}_\epsilon \left[ \sup_{\rho \in \mathcal{P}(\mathbb{R}^d)} \frac{1}{M} \sum_{i=1}^M \sum_{s=1}^{d_\mathcal{Y}} \epsilon_{ms} \mathbb{E}_\rho[\sigma_s(x^m; \omega)] - \lambda \mathsf{KL}(\rho \| \rho_0) \right] \tag{16}$$

as follows:

$$\begin{aligned}
\mathfrak{R}^\lambda(\mathcal{F}) &= \lambda \mathbb{E}_\epsilon \left[ \log \mathbb{E}_{\rho_0} \left[ \exp \left( \frac{1}{\lambda M} \sum_{i=1}^M \sum_{s=1}^{d_\mathcal{Y}} \epsilon_{ms} \sigma_s(x^m; \omega) \right) \right] \right] \\
&\leq \lambda \log \mathbb{E}_\epsilon \left[ \mathbb{E}_{\rho_0} \left[ \exp \left( \frac{1}{\lambda M} \sum_{i=1}^M \sum_{s=1}^{d_\mathcal{Y}} \epsilon_{ms} \sigma_s(x^m; \omega) \right) \right] \right] \\
&= \lambda \log \mathbb{E}_{\rho_0} \left[ \prod_{m=1}^M \prod_{s=1}^{d_\mathcal{Y}} \mathbb{E}_\epsilon \left[ \exp \left( \frac{1}{\lambda M} \epsilon_{ms} \sigma_s(x^m; \omega) \right) \right] \right] \\
&\leq \lambda \log \mathbb{E}_{\rho_0} \left[ \prod_{m=1}^M \prod_{s=1}^{d_\mathcal{Y}} \exp \left( \frac{1}{2\lambda^2 M^2} \sigma_s(x^m; \cdot)^2 \right) \right] \\
&\leq \frac{C_0^2 d_\mathcal{Y}}{2\lambda M}.
\end{aligned}$$

Hence we have

$$\begin{aligned}
\mathfrak{R}_{\hat{\mathcal{D}}_M}(\mathcal{F}) &= \mathbb{E}_\epsilon \left[ \sup_{f \in \mathcal{F}} \frac{1}{M} \sum_{i=1}^M \sum_{s=1}^{d_\mathcal{Y}} \epsilon_{ms} f^s(x^m) - \lambda \mathsf{KL}(\rho \| \rho_0) + \lambda \mathsf{KL}(\rho \| \rho_0) \right] \\
&\leq \mathfrak{R}^\lambda(\mathcal{F}) + \sup_{\rho : \mathsf{KL}(\rho \| \rho_0) \leq B} \lambda \mathsf{KL}(\rho \| \rho_0) \\
&\leq \frac{d_\mathcal{Y} C_0^2}{2\lambda M} + \lambda B.
\end{aligned}$$

Minimizing over $\lambda$ yields

$$\mathfrak{R}_{\hat{\mathcal{D}}_M}(\mathcal{F}) \leq \sqrt{\frac{2d_{\mathcal{Y}}C_0^2 B}{M}}.$$

□

The following lemma will be used in the proof of Proposition 1.

**Lemma 4.** *[Lu, 2023] For any $\mu \in \mathcal{P}(\mathbb{R}^d)$ , the following inequality hold:*

$$L_1(\mu) \leq \tau \mathsf{KL}(\mu \| \mu^*).$$

*where $\mu^*$ is the optimal solution of problem* (9)*.*

*Proof of Proposition 1.* By Lemma 2, the generalization bound for $\hat{f}$ is upper bounded by $\sqrt{2}\ell$ multiplied by the Rademacher complexity of $\bar{f}$, as defined in Step 7 of Algorithm 1, thanks to the Lipschitz property of $\Psi$ and the Euclidean projection. Consider the output function $\bar{f}$, since $L_1(\bar{f}) \leq \epsilon$, then by Lemma 4 we have

$$\mathsf{KL}(\bar{\mu} \| \mu^*) \leq \frac{\epsilon}{\tau}.$$

Then using Lemma 3, we can obtain

$$\mathfrak{R}_{\hat{\mathcal{D}}_M}(\mathcal{F}) \leq \sqrt{\frac{2d_{\mathcal{Y}}C_0^2 \epsilon}{M\tau}}.$$

Hence, using McDiarmid inequality, we obtain

$$\left| \mathbb{E}_{\mathcal{D}}[\Psi(\hat{f}(X), Z)] - \mathbb{E}_{\hat{D}_M}[\Psi(\hat{f}(X), Z)] \right| \leq 2\sqrt{2}\ell \sqrt{\frac{2d_{\mathcal{Y}}C_0^2 \epsilon}{M\tau}} + \mathcal{O}\left( \sqrt{\frac{\log(1/\delta)}{M}} \right).$$

□

### E.2 Proofs for Section C.2

*Proof of Lemma 1.* We first consider the inner maximization

$$\max_{\nu'} E_\tau(\mu, \nu') = \max_{\nu'} \left( E(\mu, \nu') - \tau \mathsf{KL}(\nu' \| \nu_0) \right) + \tau \mathsf{KL}(\mu \| \mu_0). \tag{17}$$

By first-order optimality condition, the solution $\nu_\tau^*$ of problem (17) satisfies

$$\frac{\delta E}{\delta \nu}(\mu, \nu_\tau^*)(\omega) - \tau \log \nu_\tau^*(\omega) + \tau \frac{\|\omega\|_2^2}{2} = 0.$$

It follows that

$$\begin{aligned} &\max_{\nu} \left( E(\mu, \nu') - \tau \mathsf{KL}(\nu' \| \nu_0) \right) + \tau \mathsf{KL}(\mu \| \mu_0) \\ =& - d\tau \log(2\pi) + \tau \mathsf{KL}(\mu \| \mu_0) + E(\mu, \nu_\tau^*) - \tau \int \log \nu_\tau^* \mathrm{d}\nu_\tau^* - \frac{\tau}{2} \mathbb{E}_{\nu_\tau^*} \left[ \|\omega\|_2^2 \right]. \end{aligned}$$

Define $g(\tau) = E(\mu, \nu_\tau^*) - \tau \int \log \nu_\tau^* \mathrm{d}\mu_\tau^* - \frac{\tau}{2} \mathbb{E}_{\nu_\tau^*}[\|\omega\|_2^2]$, then we have

$$g(0^+) = E(\mu, \nu^*) - \lim_{\tau \to 0} \tau \int \log \nu_\tau^* \mathrm{d}\nu_\tau^* - 0 \cdot \mathbb{E}_{\nu^*} \left[ \|\omega\|_2^2 \right] = E(\mu, \nu^*),$$

and by envelope principle,

$$g'(0^+) = - \lim_{\tau \to 0} \left( \int \log \nu^* \mathrm{d}\nu^* - \frac{1}{2} \mathbb{E}_{\nu^*}[\|\omega\|_2^2] \right) = \lim_{\tau \to 0} \frac{\frac{\delta E}{\delta \nu}(\mu, \nu_\tau^*)}{\tau} = 0.$$

Now we consider the second derivative,

$$g''(\tau) = -\frac{\mathrm{d}}{\mathrm{d}\tau} g'(\tau) = -\frac{\mathrm{d}}{\mathrm{d}\tau} \mathsf{KL}(\nu_\tau^* \| \nu_0).$$

Since

$$\frac{\mathrm{d}\nu_\tau^*}{\mathrm{d}\nu_0}(\omega) = \frac{1}{Z(\tau)}\exp(\frac{1}{\tau}\eta(\omega)).$$

where $\eta(\omega) = \frac{\delta E}{\delta \nu}(\mu, \nu^*)(\omega)$, and $Z(\tau) = \int \exp(\eta(\omega)/\tau)\nu_0(\mathrm{d}\omega)$. Combine these expressions we can obtain

$$g''(\tau) = \mathbb{V}\mathrm{ar}_{\nu_\tau^*}\left(\log\frac{\mathrm{d}\nu_\tau^*}{\mathrm{d}\nu_0}\right) \geq 0.$$

Then by Taylor's expansion with Lagrangian remainder we obtain

$$g(\tau) = g(0^+) + g'(0^+)\tau + \frac{1}{2}g''(\xi)\tau^2 \geq g(0^+) = E(\mu, \nu^*).$$

Then

$$\begin{aligned} L_2^0(\mu,\nu) - L_2(\mu,\nu) &= (\max_\nu E(\mu,\nu) - \max_\nu E_\tau(\mu,\nu)) - (E(\mu,\nu) - (E(\mu,\nu) - \tau\mathsf{KL}(\nu\|\nu_0))) \\ &= E(\mu,\nu^*) - f(\tau) - \tau\mathsf{KL}(\nu\|\nu_0) \\ &\leq E(\mu,\nu^*) - (-d\tau\log(2\pi) + E(\mu,\nu^*)) \\ &\leq d\tau\log(2\pi). \end{aligned}$$

Similarly,

$$\begin{aligned} &\min_{\mu'}\max_{\nu'} E(\mu',\nu') - \min_{\mu'}\max_{\nu'} E_\tau(\mu',\nu') \\ =&\min_{\mu'}\max_{\nu'} E(\mu',\nu') - \min_{\mu'}\max_{\nu'}(E(\mu',\nu') - \tau\mathsf{KL}(\nu'\|\nu_0) + \tau\mathsf{KL}(\mu'\|\mu_0)) \\ \geq&\min_{\mu'}\max_{\nu'} E(\mu',\nu') - \min_{\mu'}(\max_{\nu'} E(\mu',\nu') + \tau\mathsf{KL}(\mu'\|\mu_0)) \\ \geq& - d\tau\log(2\pi). \end{aligned}$$

The last inequality is the similar to the bound $\max_\nu E(\mu,\nu) - \max_\nu E_\tau(\mu,\nu)$ by replacing $E(\mu,\nu)$ to $\max_\nu E(\mu,\nu)$. Hence

$$\begin{aligned} L_1^0(\mu) - L_1(\mu) &= \left(\max_\nu E(\mu,\nu) - \max_\nu E_\tau(\mu,\nu)\right) - \left(\min_\mu\max_\nu E(\mu,\nu) - \min_\mu\max_\nu E_\tau(\mu,\nu)\right) \\ &\leq d\tau\log(2\pi) - \tau\mathsf{KL}(\mu\|\mu_0) + d\tau\log(2\pi) \\ &\leq 2d\tau\log(2\pi). \end{aligned}$$

□

### E.3 Proofs for Section C.3

*Proof of Proposition 2.* Since the output of neural network $\bar{\mu}$ satisfies $L_1(\bar{\mu}) \leq \epsilon$. By Lemma 1, we can obtain

$$L_1^0(\bar{\mu}) \leq \epsilon + 2\tau d\log(2\pi).$$

We claim

$$\max_\nu E(\bar{\mu},\nu) \geq \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\bar{f}(X), Z)],$$
$$\min_\mu\max_\nu E(\mu,\nu) = \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}_\star(X), Z)].$$

Indeed, the first inequality follows because the duality function $\max_\nu E(\bar{\mu},\nu)$ must have positive Lagrangian penalty term, which implies

$$\mathbb{E}_{(X,Z)\sim\hat{\mathcal{D}}}\left[\Lambda_=(X)^\mathsf{T}(Af(X)-b) + \frac{\lambda}{2}\|Af(X)-b\|_2^2 + \frac{\lambda}{2}\sum_{j=1}^{d_\leq}\left(\left[\tfrac{\Lambda_{\leq,j}(X)}{\lambda} + g_j(f(X))\right]_+^2 - \tfrac{\Lambda_{\leq,j}(X)^2}{\lambda^2}\right)\right] \tag{18}$$

is positive. And the second equality holds since the saddle point of $E(\mu,\nu)$ satisfies strong duality, hence the same Lagrangian penalty term is zero. By these two inequalities, we can obtain

$$\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\bar{f}(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}_\star(X), Z)] \leq L_1^0(\bar{\mu}) \leq \epsilon + 2\tau d\log(2\pi).$$

□

Next, we verify that the assumption $L_1(\bar{\mu}) \leq \epsilon$ made in Proposition 2 can be achieved using SGDA algorithm. We need the following additional assumption.

**Assumption 3.** $\phi_i(x;\cdot)$ all have bounded gradient up to fourth order, i.e. $\|\nabla^j \phi_i(x;\cdot)\|_F \leq C_j, j = 0,1,2,3,4$ for $i = 1,2,3$.

Following Lu [2023], Liu et al. [2025], which establishes the $\tilde{O}(1/\epsilon)$ complexity to obtain a solution satisfying $L_1(\bar{\mu}) \leq \epsilon$, we will verify these terms required in Liu et al. [2025]:

1. The first variation of $E$ have bounded derivatives up to the fourth order: $\|\nabla^i \frac{\delta E}{\delta \mu}\|_F, \|\nabla^i \frac{\delta E}{\delta \nu}\|_F \leq M_i, i = 1,\ldots,4$, where $\|\cdot\|_F$ is Frobenius norm.
2. $E$ has a bounded cross second-order variation: $\|\frac{\delta^2 J}{\delta \mu \delta \nu}\|_\infty \leq C_0$.
3. The Hessian of second variations are bounded: $\|\nabla_\theta \nabla_{\theta'}^\mathsf{T} \frac{\delta^2 E}{\delta \mu^2}\|_F, \|\nabla_\theta \nabla_\omega^\mathsf{T} \frac{\delta^2 J}{\delta \mu \delta \nu}\|_F, \|\nabla_\omega \nabla_{\omega'}^\mathsf{T} \frac{\delta^2 E}{\delta \nu^2}\|_F \leq C_1$, $\|\nabla_\theta \frac{\delta^2 E}{\delta \mu \delta \nu}\|_\infty, \|\nabla_\omega \frac{\delta^2 E}{\delta \mu \delta \nu}\|_\infty \leq C_2$.

We first calculate the first variation term:

$$\begin{aligned}
&\frac{\delta E}{\delta \mu}(\mu,\nu) = \mathbb{E}_{\mathcal{D}}\Big[\Psi'(\mathbb{E}_\mu[\phi_1(X;\theta)], Z)\phi_1(X;\theta) + \mathbb{E}_\nu[\phi_2(X;\omega)]^\mathsf{T} A \nabla_\theta \phi_1(X;\theta) \\
&+ \sigma(A\mathbb{E}_\mu[\phi_1(X;\theta)] - b)^\mathsf{T} \phi_1(X;\theta) + \\
&\sigma \sum_{i\in\mathcal{O}} \left(\frac{\mathbb{E}_\nu[\phi_{3i}(X;\zeta)]}{\sigma} + g_i(\mathbb{E}_\mu[\phi_1(X;\theta)])\right) g_i'(\mathbb{E}_\mu[\phi_1(X;\theta)])\phi_1(X;\theta)\Big]
\end{aligned} \tag{19}$$

$$\begin{aligned}
&\frac{\delta E}{\delta \nu}(\mu,\nu) = \mathbb{E}_{\mathcal{D}}\Big[\phi_2(X;\omega)^\mathsf{T}(A\mathbb{E}_\mu[\phi_1(X;\theta)] - b). \\
&+ \sum_{i\in\mathcal{O}} \left[\frac{\mathbb{E}_\nu[\phi_{3i}(X;\zeta)]}{\sigma} + g_i(\mathbb{E}_\mu[\phi_1(X;\theta)])\right] \phi_{3i}(X;\zeta) + \frac{1}{\sigma} \sum_{i\in[d_\mathcal{C}]} \mathbb{E}_\nu[\phi_{3i}(X;\zeta)]\phi_{3i}(X;\zeta)\Big].
\end{aligned} \tag{20}$$

where $\frac{\delta E}{\delta \mu}(\mu,\nu)$, $\frac{\delta E}{\delta \nu}(\mu,\nu)$ can be bounded by

$$\left\|\frac{\delta E}{\delta \mu}(\mu,\nu)\right\| \leq \ell C_0 + C_0^2\|A\| + \sigma(\|A\|C_0 + \|b\|)C_0 + \sigma d_\mathcal{C}(C_0/\sigma + G_0)G_1 C_0.$$

$$\left\|\frac{\delta E}{\delta \nu}(\mu,\nu)\right\| \leq C_0(\|A\|C_0 + \|b\|) + d_\mathcal{C}\left((\frac{C_0}{\sigma} + G_0)C_0 + \frac{1}{\sigma}C_0^2\right).$$

The correspoonding Wasserstein gradient is

$$\begin{aligned}
&\nabla_\theta^i \frac{\delta E}{\delta \mu}(\mu,\nu) = \mathbb{E}_{\mathcal{D}}\Big[\Psi'(\mathbb{E}_\mu[\phi_1(X;\theta)], Z)\nabla_\theta^i \phi_1(X;\theta) + \mathbb{E}_\nu[\phi_2(X;\omega)]^\mathsf{T} A \nabla_\theta^i \phi_1(X;\theta) \\
&+ \sigma(A\mathbb{E}_\mu[\phi_1(X;\theta)] - b)^\mathsf{T} \nabla_\theta^i \phi_1(X;\theta) + \\
&\sigma \sum_{i\in\mathcal{O}} \left(\frac{\mathbb{E}_\nu[\phi_{3i}(X;\zeta)]}{\sigma} + g_i(\mathbb{E}_\mu[\phi_1(X;\theta)])\right) g_i'(\mathbb{E}_\mu[\phi_1(X;\theta)])\nabla_\theta^i \phi_1(X;\theta)\Big]
\end{aligned} \tag{21}$$

$$\begin{aligned}
&\nabla_{\omega,\zeta}^i \frac{\delta E}{\delta \nu}(\mu,\nu) = \mathbb{E}_{\mathcal{D}}\Big[\nabla_{\omega,\zeta}^i \phi_2(X;\omega)^\mathsf{T}(A\mathbb{E}_\mu[\phi_1(X;\theta)] - b). \\
&+ \sum_{i\in\mathcal{O}} \left[\frac{\mathbb{E}_\nu[\phi_{3i}(X;\zeta)]}{\sigma} + g_i(\mathbb{E}_\mu[\phi_1(X;\theta)])\right] \nabla_{\omega,\zeta}^i \phi_{3i}(X;\zeta) + \frac{1}{\sigma} \sum_{i\in[d_\mathcal{C}]} \mathbb{E}_\nu[\phi_{3i}(X;\zeta)]\nabla_{\omega,\zeta}^i \phi_{3i}(X;\zeta)\Big].
\end{aligned} \tag{22}$$

Hence the norm an also be bounded by

$$\left\|\nabla_\theta^i \frac{\delta E}{\delta \mu}(\mu,\nu)\right\| \leq \ell C_i + C_0 C_i\|A\| + \sigma(\|A\|C_0 + \|b\|)C_i + \sigma d_\mathcal{C}(C_0/\sigma + G_0)G_1 C_i.$$

$$\left\|\nabla^i_{\omega,\zeta}\frac{\delta E}{\delta \nu}(\mu,\nu)\right\| \le C_i(\|A\|C_0+\|b\|)+d_{\mathcal{C}}\left((\frac{C_0}{\sigma}+G_0)C_i+\frac{1}{\sigma}C_0C_i\right).$$

For the second order variation,

$$\frac{\delta^2 E}{\delta \mu^2}(\mu,\nu)=\mathbb{E}_{\mathcal{D}}\Big[\Psi'(\mathbb{E}_\mu[\phi_1(X;\theta)],Z)\phi_1(X;\theta)\phi_1(X;\theta')+\sigma(A\mathbb{E}_\mu[\phi_1(X;\theta)]-b)^{\mathsf{T}}\phi_1(X;\theta)\phi_1(X;\theta')$$
$$+\sigma\sum_{i\in\mathcal{O}}\left(\frac{\mathbb{E}_\nu[\phi_{3i}(X;\zeta)]}{\sigma}+g_i(\mathbb{E}_\mu[\phi_1(X;\theta)])\right)g_i''(\mathbb{E}_\mu[\phi_1(X;\theta)])\phi_1(X;\theta)\phi_1(X;\theta')\Big]. \tag{23}$$

$$\frac{\delta^2 E}{\delta \nu^2}(\mu,\nu)=\mathbb{E}_{\mathcal{D}}\Big[\sum_{i\in\mathcal{O}}\left[\frac{\mathbb{E}_\nu[\phi_{3i}(X;\zeta)]}{\sigma}+g_i(\mathbb{E}_\mu[\phi_1(X;\theta)])\right]\phi_{3i}(X;\zeta)\phi_{3i}(X;\zeta')+\frac{1}{\sigma}\sum_{i\in[d_{\mathcal{C}}]}\phi_{3i}(X;\zeta)\phi_{3i}(X;\zeta')\Big]. \tag{24}$$

which can also be bounded similarly by the Assumptions 2, 3. Finally, we remark that Liu et al. [2025] requires the differentiability of $\Psi$, but their results remain to hold when $\Psi$ has weak derivatives, which is satisfied due to the convexity of $\Psi$.

### E.4 Proofs for Section C.4

*Proof of Proposition 3.* Recall that the projection of $\bar{f}$ is $\hat{f}$. By assumption we have

$$\begin{aligned}L_1^0(\bar{\mu})&=\max_\nu E(\bar{\mu},\nu)-\min_\mu\max_\nu E(\mu,\nu)\\&\ge\max_\nu E(\bar{\mu},\nu)-\max_\nu E(\hat{f},\nu)\end{aligned}$$

where we abuse the notation $E(\hat{f},\nu)$ to express the projection. Since $\hat{f}$ is feasible,

$$\max_\nu E(\hat{f},\nu)=\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}(X),Z)],$$

by strong duality, then we can obtain

$$\frac{\lambda}{2}\mathbb{E}_{\hat{\mathcal{D}}}\left[\|A\mathbb{E}_{\bar{\mu}}[\phi_1(X;\theta)]-b\|_2^2-\|A\hat{f}_1(X)-b\|_2^2\right]\le L_1^0(\bar{\mu}).$$

where $\hat{f}_1$ is the projection on hyperplane $Ax-b=0$, expand the error term we can obtain

$$\begin{aligned}&\frac{\lambda}{2}\mathbb{E}_{\hat{\mathcal{D}}}\left[\|A(\mathbb{E}_{\bar{\mu}}[\phi_1(X;\theta)]-\hat{f}_1(X))\|_2^2+2\langle A(\mathbb{E}_{\bar{\mu}}[\phi_1(X;\theta)]-\hat{f}_1(X)),b\rangle\right]\\=&\frac{\lambda}{2}\mathbb{E}_{\hat{\mathcal{D}}}\left[\|A(\mathbb{E}_{\bar{\mu}}[\phi_1(X;\theta)]-\hat{f}_1(X))\|_2^2\right]\le L_1^0(\bar{\mu}).\end{aligned}$$

Since $\mathbb{E}_{\bar{\mu}}[\phi_1(X;\theta)]-\hat{f}_1(X)$ lies in the row space of matrix $A$, hence by eigenvalue theory we can obtain

$$\|A(\mathbb{E}_{\bar{\mu}}[\phi_1(X;\theta)]-\hat{f}_1(X))\|\ge\underline{\sigma}\|\mathbb{E}_{\bar{\mu}}[\phi_1(X;\theta)]-\hat{f}_1(X)\|,$$

where $\underline{\sigma}$ denotes the smallest nonzero absolute value of the singular values of the constraint matrix $A$. Combining the above result, we have

$$\mathbb{E}_{\hat{\mathcal{D}}}\left[\left\|\mathbb{E}_{\bar{\mu}}[\phi_1(X;\theta)]-\hat{f}_1(X)\right\|_2\right]\le\sqrt{\frac{2L_1^0(\bar{\mu})}{\lambda\underline{\sigma}}}.$$

Similarly, we can project $\hat{f}_1$ on the region $\{g_i(x)\le 0\}$, which can be bounded by

$$\frac{\lambda}{2}\mathbb{E}_{\hat{\mathcal{D}}}\left[\sum_{i\in[d_\le]}(g_i(\hat{f}_1(X))^2\right]\le L_1^0(\bar{\mu}).$$

Hence by Assumption 2, we can iteratively make projection to satisfy the $i$-th constraint $\{g_i(x)\le 0\}$. For each constraint, since

$$\mathbb{E}_{\hat{\mathcal{D}}}\left[\left\|g(\hat{f}_1(X))-g(\hat{f}(X))\right\|_2\right]\le\sqrt{\frac{2L_1^0(\bar{\mu})}{\lambda}}.$$

Then since $g(\hat{f}(X)) = 0$ if it has been projected, then by Assumption 2, since the KKT condition is nondegenerate and $\|\nabla g_i(x)\|_2 \geq m$ in a sufficient large neighbor of zero, then

$$\mathbb{E}_{\hat{\mathcal{D}}}\left[\left\|\hat{f}_1(X) - \hat{f}(X)\right\|_2\right] \leq \frac{d_{\leq}}{m}\sqrt{\frac{2L_1^0(\bar{\mu})}{\lambda}}.$$

Add these two terms and by Lipschitzness of $\Psi$, we can obtain

$$\begin{aligned}
&\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\bar{f}(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}(X), Z)] \leq \ell\mathbb{E}_{\hat{\mathcal{D}}}[\|\bar{f}(X) - \hat{f}(X)\|_2] \\
\leq &\ell\left(\sqrt{\frac{2L_1^0(\bar{\mu})}{\lambda\underline{\sigma}}} + \frac{d_{\leq}}{m}\sqrt{\frac{2L_1^0(\bar{\mu})}{\lambda}}\right) \\
\leq &\ell\left(\sqrt{\frac{2}{\lambda\underline{\sigma}}} + \frac{d_{\leq}}{m}\sqrt{\frac{2}{\lambda}}\right)\sqrt{\epsilon + 2d\tau\log(2\pi)}.
\end{aligned}$$

□

### E.5 Proofs for Section C.5

Add all the error term from Proposition 1, 2, 3, which is upper bounded by

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}}[\Psi(\hat{f}(X), Z)] - \mathbb{E}_{\mathcal{D}}[\Psi(f_\star(X), Z)] \\
=&\big(\mathbb{E}_{\mathcal{D}}[\Psi(\hat{f}(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}(X), Z)]\big) + \big(\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(f_K(X), Z)]\big) \\
+&\big(\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(f_K(X), Z)] - \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}_*(X), Z)]\big) + \big(\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}_*(X), Z)] - \mathbb{E}_{\mathcal{D}}[\Psi(f_\star(X), Z)]\big).
\end{aligned}$$

The first three term can be explicitly bounded by Proposition 1-3. For the last term, observe

$$\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(\hat{f}_*(X), Z)] - \mathbb{E}_{\mathcal{D}}[\Psi(f_\star(X), Z)] \leq \mathbb{E}_{\hat{\mathcal{D}}}[\Psi(f_\star(X), Z)] - \mathbb{E}_{\mathcal{D}}[\Psi(f_\star(X), Z)].$$

Since $f_\star$ is bounded, then by Hoeffding inequality, with probabilty $1 - \delta$ we can obtain

$$\mathbb{E}_{\hat{\mathcal{D}}}[\Psi(f_\star(X), Z)] - \mathbb{E}_{\mathcal{D}}[\Psi(f_\star(X), Z)] \leq \mathcal{O}\left(\sqrt{\frac{\log(2/\delta)}{M}}\right).$$

Hence adding all terms, we can obtain

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}}[\Psi(\hat{f}(X), Z)] - \mathbb{E}_{\mathcal{D}}[\Psi(f_\star(X), Z)] \leq \epsilon + 2\tau d\log(2\pi) \\
+&\ell\left(\sqrt{\frac{2}{\lambda\underline{\sigma}}} + \frac{d_{\leq}}{m}\sqrt{\frac{2}{\lambda}}\right)\sqrt{\epsilon + 2d\tau\log(2\pi)} + 2\sqrt{2}\ell\sqrt{\frac{2d_{\mathcal{Y}}C_0^2\epsilon}{M\tau}} + \mathcal{O}\left(\sqrt{\frac{\log(1/\delta)}{M}}\right).
\end{aligned}$$

# F Experiment Details

In this section, we present additional experimental results and provide a comprehensive description of the experimental setup and implementation details.

The implementation of experiments is based on PyTorch 2.4.1 and all the experiments were conducted on a Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz with Nvidia RTX3090 GPU. Gurobi 12.0.2 was used as the optimization solver for computing optimality gaps and OSQP was used for the projection step. The hyperparameters of our method and PDL were tuned using a grid search, if not otherwise specified.

## F.1 Experiment Setting: Multi-item Feature-based Newsvendor Model with Constraints (Linear Demand Model)

**Problem Definition.** The newsvendor problem is configured with the following parameters:

- Holding Costs ($h_{\text{cost}}$): $[5, 5, 5, 5, 5, 8, 8, 8, 8, 8]$
- Backorder Costs ($b_{\text{cost}}$): $[15, 15, 15, 15, 15, 10, 10, 10, 10, 10]$
- Resource Utility Matrix ($A$):

$$\begin{bmatrix} 1.0 & 1.5 & 0.5 & 2.0 & 1.2 & 0.8 & 1.7 & 1.1 & 0.9 & 1.3 \\ 0.8 & 1.2 & 1.0 & 1.5 & 0.6 & 1.8 & 1.4 & 0.7 & 2.0 & 1.0 \\ 1.2 & 0.7 & 1.8 & 0.9 & 1.5 & 1.1 & 0.5 & 2.0 & 1.3 & 0.4 \end{bmatrix}$$

- Capacity ($C$): $[96.0, 96.0, 91.0]$

The holding costs and backorder costs are designed to test the heterogeneity of critical fractile, i.e. the $\frac{b}{b+h}$-quantile of the demand distribution. Each entry of the resource utility matrix represents the consumption of a certain resource by a certain item. The capacity is designed to be less than the uncapacitated quantity of resource consumption so that the constraints are binding.

**Dataset Configuration.** Under the linear demand setting, we assume that the feature vector $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$, and the conditional distribution of $Z$ for any feature $X$ follows

$$Z|X \sim \beta^{\mathsf{T}} X + \varepsilon,$$

where $\beta \in \mathbb{R}^{d_X \times J}$ is the weight parameter and $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is the noise term independent of the feature $X$. The weight matrix $\beta$ encodes the relationship between features and demand for $J$ products. The linear demand dataset generates the following synthetic demand data:

$$\beta = \begin{bmatrix} \beta_{0,1} & \beta_{0,2} & \cdots & \beta_{0,J} \\ \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,J} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{d_X-1,1} & \beta_{d_X-1,2} & \cdots & \beta_{d_X-1,J} \end{bmatrix},$$

where $\beta_{0,i}$ is the intercept term for item $i$. The size of the training dataset is 100. Further, we assume that for each item $i$, only $s$ features have non-zero weights $\beta$, while the rest are zero. The problem parameters adopted are listed in Table 3. In our experiment, we employ the parameter setting as shown in Table 4 for ALM-SGDA and the parameter setting as shown in Table 5 for PDL. Out-of-sample performance is evaluated against the ground truth distribution.

Table 3: Problem parameters of the multi-item feature-based newsvendor model in Section 4.1

| Category | Parameter | Value |
|---|---|---|
| Features | Dimension ($d_X$) | 100 |
| | Mean ($\mu_X$) | 0.0 |
| | Std. Dev. ($\sigma_X$) | 1.0 |
| Items | Quantity ($J$) | 10 |
| Weight Matrix $\beta$ | Intercept | 10.0 |
| | Sparsity per item ($s$) | 10 |
| | Active weight | 0.1 |
| | Noise std. dev. | 0.1 / 0.5 / 0.8 |

Table 4: Model configuration of experiments in Section 4.1 for ALM-SGDA

| Category | Parameter | Value |
|---|---|---|
| Network Architecture | Primal hidden dimension | 6000 |
| | Dual hidden dimension | 6000 |
| | Primal activation | Sigmoid |
| | Dual activation | ReLU |
| Optimization (Primal) | Learning rate | $1 \times 10^{-5}$ |
| | Weight decay factor | $1 \times 10^{-5}$ |
| | Noise std. dev. | $3 \times 10^{-5}$ |
| Optimization (Dual) | Learning rate | $5 \times 10^{-5}$ |
| | Weight decay factor | $1 \times 10^{-5}$ |
| | Noise std. dev. | $3 \times 10^{-5}$ |
| Training | Penalty | $\rho_{\text{init}} = 1000$ with decay factor of 0.99/epoch |
| | Epochs | 600 |
| | Batch size | 32 |

Table 5: Model configuration of experiments in Section 4.1 for PDL

| Category | Parameter | Value |
|---|---|---|
| Network Architecture | Primal hidden dim. | [120, 120] |
| | Dual hidden dim. | [120, 120] |
| | Primal activation | Sigmoid |
| | Dual activation | ReLU |
| Optimization (Primal/Dual) | Learning rate | $1 \times 10^{-4}$ |
| Training | Initial penalty ($\rho_0$) | 1.0 |
| | Penalty update multiplier | 10.0 |
| | Penalty upper safe guard | 1000.0 |
| | Penalty update tolerance | 0.5 |
| | Epochs | 300 |
| | Batch size | 32 |

### F.2 Battery Arbitrage Problem

Below is the mathematical formulation of the battery arbitrage problem in Section 4.2,

$$\begin{aligned} \underset{z_{\text{in}}, z_{\text{out}}, z_{\text{state}} \in \mathbb{R}^{24}}{\text{minimize}} \quad & \mathbb{E}_{y \sim p(y|x;\theta)} \left[ \sum_{i=1}^{24} y_i (z_{\text{in},i} - z_{\text{out},i}) + \lambda \left\| z_{\text{state}} - \frac{B}{2} \right\|^2 + \epsilon \|z_{\text{in}}\|^2 + \epsilon \|z_{\text{out}}\|^2 \right] \\ \text{subject to} \quad & z_{\text{state},i+1} = z_{\text{state},i} - z_{\text{out},i} + \gamma_{\text{eff}} z_{\text{in},i} \quad \forall i \\ & z_{\text{state},1} = B/2 \\ & 0 \le z_{\text{in}} \le c_{\text{in}}, 0 \le z_{\text{out}} \le c_{\text{out}}, 0 \le z_{\text{state}} \le B \end{aligned} \tag{25}$$

**Dataset Configuration.** The dataset, developed for energy price forecasting, comprises 2,183 feature-label pairs $(X, Y)$. The feature set integrates multimodal data, including historical prices, load forecasts, temperature readings, and temporal metadata. Features are standardized through z-score normalization ($\mu = 0$, $\sigma = 1$) following concatenation. The train-test split adheres to a 0.8:0.2 ratio, aligning with the partition methodology of Donti et al. [2017] to maintain temporal continuity in time-series modeling.

### F.3 Additional Experimental Results: Multi-item Feature-based Newsvendor Model with Constraints (Nonlinear Demand model)

**Problem Definition.** The parameter settings of resource coefficient $A$, holding cost $h$, backorder cost $b$ and capacity $C$ for the multi-item newsvendor model are the same as those in the linear demand case (Section F.1).

**Dataset Configuration.** A nonlinear demand model captures complex relationships between demand ($Z$) and features ($X$) using nonlinear functions. We consider the following nonlinear demand model with sinusoidal and exponential dependencies:

$$Z_i = c + \sin(2X_i^\top \beta_{0i}) + \sum_{j \neq i} \gamma_{ij} \cdot \exp(-16(X_j^\top \beta_{0j})^2) + \varepsilon,$$

where $\beta_{0i}$ represents item-specific coefficients, $\gamma_{ij}$ models cross-item interactions, $\varepsilon$ denotes noise with zero mean and $c$ is the constant intercept term. The size of the training data is 1000. The specific values of parameters are listed in Table 6.

Table 6: Problem parameters of the multi-item feature-based newsvendor model with nonlinear demand model

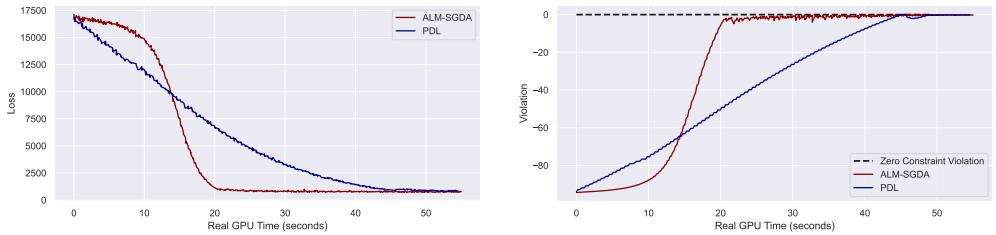
| **Category** | **Parameter** | **Value** |
|---|---|---|
| Features | Dimension ($d_X$) | 100 |
| | Mean ($\mu_X$) | 0.0 |
| | Std. Dev. ($\sigma_X$) | 1.0 |
| Items | Quantity ($J$) | 10 |
| Intercept Term | Constant ($c$) | 10.0 |
| Weight Matrix $\beta$ | Intercept $\beta_0$ | 5.0 |
| | Active weight | 1.0 |
| | Sparsity per item (s) | 20 |
| | Noise std. dev. | 0.1 / 0.5 / 0.8 |
| Interaction Term $\gamma_{ij}$ | Mean ($\mu_\gamma$) | 1.0 |
| | Std. Dev. ($\sigma_\gamma$) | 0.5 |
| | Sparsity ($s_\gamma$) | 0.3 |

**Experiment Results.** We compare the empirical performance of ALM-SGDA against PDL in terms of convergence speed and solution quality. Figure 2 provides a side-by-side comparison of ALM-SGDA (red) and PDL (blue) across two critical metrics: training loss (Figure 2(a)) and mean constraint violation (Figure 1(b)), both plotted against real GPU time (in seconds). In Figure 2(a), PDL exhibits a steeper initial decline in training loss during early phase (for $t < 10$ sec) while ALM-SGDA overcomes its burn-in period quickly and achieves faster convergence afterward. Figure 2(b) evaluates constraint satisfaction by plotting mean constraint violation relative to a zero threshold (black dashed line), demonstrating how each method maintains feasibility. Both subplots illustrate ALM-SGDA's improved computational efficiency, particularly in achieving optimality and moving towards feasibility.

Table 7 reports the average out-of-sample relative optimality gap, relative to the groudtruth, of the solutions from ALM-SGDA and PDL onto the feasible region, where the noise scale is varied in $\{0.1, 0.5, 0.8\}$. Since we do not report constraint violation as a separate metric, for PDL, we perform the same projection as in Step 8 of Algorithm 1 to enforce feasibility as well. We observe similar trends to the linear demand case—ALM-SGDA consistently outperforms PDL across varying noise scale.

### F.4 Additional Experimental Results: Comparison with Differentiable Convex Optimization Layers

In this section, we provide results of numerical experiments that illustrate the computational running time of ALM-SGDA, compared with the following differentiable convex optimization layer approaches: **qpth/OptNet**, **cvxpylayers**, **Alt-Diff** and **BPQP**.




(a) Training Loss vs Real GPU Time (b) Mean constraint violation vs Real GPU Time

Figure 2: The training loss $\Psi(f)$ and mean constraint violation under nonlinear demand model

Table 7: Optimality gaps of ALM-SGDA (Ours) and PDL for nonlinear demand model

| Noise scale | Method | Opt. Gap (%) |
|---|---|---|
| $\sigma = 0.1$ | ALM-SGDA | **12.06** (0.001) |
| | PDL | 20.26 (0.001) |
| $\sigma = 0.5$ | ALM-SGDA | **17.41** (0.000) |
| | PDL | 26.03 (0.002) |
| $\sigma = 0.8$ | ALM-SGDA | **25.07** (0.000) |
| | PDL | 33.87(0.001) |

**Experiment Setup.** We evaluate these approaches based on the widely adopted baseline neural network: multilayer perceptron (MLP). All baseline methods use a fully-connected MLP architecture with three hidden layers, each followed by ReLU activations with batch normalization and dropout ($p = 0.2$). The MLP output is then projected onto the feasible region of the problem by the embedded differentiable optimization layer. The dimension of each hidden layer is set as 200. The baseline models are trained with Adam optimizer and the learning rate is set as $5e-3$. The input dimension of the optimization layers, or the number of variables passed into the solver, is varied as $d_Y \in \{10, 100, 500, 1000\}$. A training dataset of 1000 samples is generated for each running instance with a training batch size of 128. For all models, training is stopped if the validation loss does not improve for consecutive 10 epochs. The maximum training time is set as 120 minutes. The sample size in the test dataset is fixed to be a large number, 10000, to mimic the groundtruth, and we report the resulting optimality gap and inference time per sample.

Since Alt-diff benefits from truncation of tolerance level $\epsilon$ for computational speed Sun et al. [2023b], we tested Alt-diff with two different tolerance levels $\epsilon = 10^{-1}$ and $\epsilon = 10^{-3}$. For BPQP, the OSQP solver is adopted as the solver for both forward pass and backward pass to stay consistent with Pan et al. [2024].

**Results.** Table 8 reports the training time of different methods across various problem sizes until convergence. We fix the feature dimension as $d_X = 100$ and the noise scale as $\sigma_\varepsilon$=0.1. These numerical results demonstrate that ALM-SGDA achieves the shortest durations across all $d_Y$ settings and consistently outperforms other baseline methods in training efficiency. It should be pointed out that Optnet utilizes qpth, a GPU-based differentiable optimizer to gain acceleration while other baseline methods involve an optimization solver that are only run in a CPU environment.

Table 9 reports the average optimality gap (%) and average inference time (ms) per sample evaluated across different levels of noise scale $\sigma_\varepsilon \in \{0.1, 0.5, 0.8\}$. ALM-SGDA maintains the lowest optimality gap and inference time over all noise scales. These results underscore ALM-SGDA's efficacy in balancing computational speed and accuracy, particularly in high-dimensional and noisy scenarios.

Figure 3 shows the optimality gaps and training time of different methods for the multi-item newsvendor problem under linear demand model with $d_X = 100$, $J = 10$ and $\sigma_\varepsilon = 0.1$. This further highlights ALM-SGDA's dominance, illustrating that it attains the smallest optimality gap ($\sim 0.6\%$) with minimal training time ($< 30$ seconds) in this case.

Table 8: Comparison of running time (s) of training. Std. dev. in parenthesis is evaluated across 5 independent runs. "-" indicates the method did not converge within the maximum training time.

| Method | $d_Y = 10$ | $d_Y = 100$ | $d_Y = 500$ | $d_Y = 1000$ |
|---|---|---|---|---|
| ALM-SGDA (ours) | **27.464**(0.254) | **32.952**(2.162) | **41.774**(5.766) | **56.104**(2.056) |
| OptNet | 36.540(6.210) | 49.926(5.062) | 76.880(20.293) | 105.438(24.245) |
| CvxpyLayer | 208.544(73.89) | 233.352(66.76) | 388.080(49.63) | 671.720(129.41) |
| Alt-Diff ($\epsilon = 10^{-1}$) | 87.530(16.440) | 93.698(9.512) | 218.39(60.304) | 478.968(232.205) |
| Alt-Diff ($\epsilon = 10^{-3}$) | 470.920(43.826) | - | - | - |
| BPQP | 119.46(46.36) | 169.604(44.26) | 681.5(173.531) | - |

Table 9: Performance results for ALM-SGDA and baseline methods

| Method | $\sigma_\varepsilon = 0.1$ | | $\sigma_\varepsilon = 0.5$ | | $\sigma_\varepsilon = 0.8$ | |
|---|---|---|---|---|---|---|
| | Opt. Gap (%) | Inference Time (ms/sample) | Opt. Gap (%) | Inference Time (ms/sample) | Opt. Gap (%) | Inference Time (ms/sample) |
| ALM-SGDA(ours) | **0.604** | **1.996** | **6.96** | **2.130** | **16.66** | **1.099** |
| OptNet | 1.260 | 2.485 | 7.48 | 2.560 | 18.09 | 1.874 |
| CvxpyLayer | 3.890 | 4.919 | 9.33 | 4.009 | 18.07 | 2.567 |
| Alt-Diff ($\epsilon = 10^{-1}$) | 3.140 | 5.158 | 10.13 | 5.192 | 20.45 | 3.271 |
| Alt-Diff ($\epsilon = 10^{-3}$) | 0.700 | 8.570 | 8.61 | 6.255 | 18.03 | 4.719 |

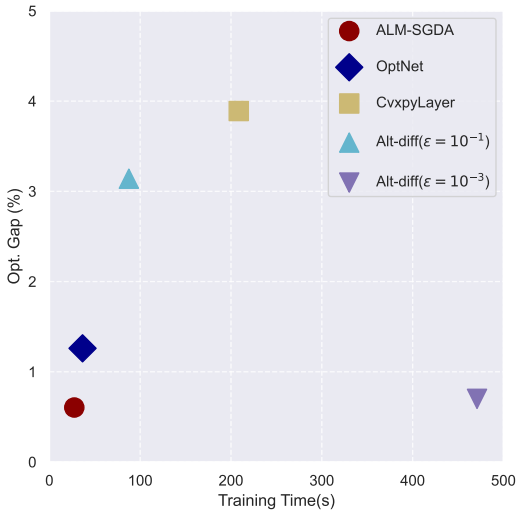



Figure 3: The optimality gap vs. training time