Using Neural Networks to Guide Data-Driven Operational Decisions

Ningyuan Chen
Rotman School of Management, University of Toronto, Toronto, ON, Canada, ningyuan.chen@utoronto.ca
Saman Lagzi
UNC Kenan-Flagler Business School, Chapel Hill, NC, USA, saman_lagzi@kenan-flagler.unc.edu
Joseph Milner
Rotman School of Management, University of Toronto, Toronto, ON, Canada, Joe.milner@rotman.utoronto.ca

We propose to use Deep Neural Networks to solve data-driven stochastic optimization problems. Given the historical data of the observed covariate, taken decision, and the realized cost in past periods, we train a neural network to predict the objective value as a function of the decision and the covariate. Once trained, for a given covariate, we optimize the neural network over the decision variable using gradient-based methods with the analytically-computed gradient and Hessian matrix. We characterize the performance of our methodology based on the generalization bound of the neural network. We conduct comprehensive experiments on three signature problems in operations management: the newsvendor problem, the multi-product pricing problem, and the call center staffing problem. Comparing our framework to existing approaches including conditional stochastic optimization and analytical approximations, we demonstrate the strength of our method when the objective function is unknown, with moderate size data, and when the structure of the problem cannot be approximated by simple or parametric forms.

Key words: Deep Neural Networks; data-driven; newsvendor problem; product pricing; staffing

1. Introduction

We consider a decision maker, upon observing a covariate $\boldsymbol{X} = \boldsymbol{X}_0$, trying to solve

$$\min_{\boldsymbol{z}} \mathbb{E}[Y|\boldsymbol{X}_0, \boldsymbol{z}] \tag{1}$$

where $\boldsymbol{z}$ is the decision variable and Y is the random cost that is associated with the covariate $\boldsymbol{X}_0$ and decision $\boldsymbol{z}$. The objective function can be expressed as $f_*(\boldsymbol{X}, \boldsymbol{z}) \triangleq \mathbb{E}[Y|\boldsymbol{X}, \boldsymbol{z}]$. This framework encompasses a wide range of real-world applications.

Example 1 (Newsvendor). Consider a newsvendor problem where a firm needs to place an order for a product having observed some covariates in anticipation of future demand. The covariate $\boldsymbol{X}$ is the observable information that affects the demand such as weather and market conditions. The decision z is the order quantity for the product. Let D be the random demand whose distribution depends on $\boldsymbol{X}$ and $d \sim D|\boldsymbol{X}$ be the realized demand. Let b and h be, respectively, the unit back-ordering and holding costs. In this case, we have the realized cost $Y = b(d-z)^+ + h(z-d)^+$.

The objective function to minimize is $f_*(\boldsymbol{X}, z) = \mathbb{E}[b(D-z)^+ + h(z-D)^+ | \boldsymbol{X}]$ and the order quantity minimizing this objective function is

$$z^* := \arg\min \mathbb{E}\left[b(D-z)^+ + h(z-D)^+\right) | \boldsymbol{X}]. \tag{2}$$

If the cumulative distribution functions of the demand given $\boldsymbol{X}$ were known to be $F(\cdot)$, then the optimal decision is given by the well-known formula

$$z^* = \inf\left\{z : F(z) \geq \frac{b}{b+h}\right\}. \tag{3}$$

Example 2 (Personalized Pricing). Consider a pricing problem where a firm determines the prices charged for a set of goods for each customer. The covariate $\boldsymbol{X}$ represents the customer feature(s) such as income and age, potentially extracted from the account profile; the decision $\boldsymbol{Z}$ is the charged price for the products; Y is the realized revenue gained from the sold products. Let $d_j(\boldsymbol{z}|\boldsymbol{X})$ be the demand function (or purchasing probability) of a customer with feature $\boldsymbol{X}$ for product j, given the price vector $\boldsymbol{z}$. Then the objective function to minimize is

$$f_*(\boldsymbol{X}, \boldsymbol{z}) = -\sum_{j=1}^{|Z|} z_j d_j(\boldsymbol{z}|\boldsymbol{X})$$

Example 3 (Call Center Staffing). Consider a staffing problem where a firm needs to determine the staffing level in a call center on a given day. The covariate $\boldsymbol{X}$ represents the observable information that may affect that specific day's operations, such as seasonality and market conditions; the decision z is the staffing level; Y is the cost, including the staffing cost and implicit cost associated with the waiting and abandonment of customers over the day. Whether or not the arrival and service follow standard queueing assumptions, Y depends on z and $\boldsymbol{X}$ through a running stochastic system. The objective function to minimize is $f_*(\boldsymbol{X}, z) = \mathbb{E}[Y|\boldsymbol{X}, z]$, which, unlike the two previous examples, may not have an explicit expression. □

In these examples, the functional form of $f_*(\boldsymbol{X}, \boldsymbol{z})$ is typically unknown and the decision maker cannot simply optimize $f_*(\boldsymbol{X}_0, \boldsymbol{z})$ over $\boldsymbol{z}$. The recent literature focuses on a *data-driven* approach: the decision maker has access to the historical data $\{(\boldsymbol{X}^i, \boldsymbol{Z}^i, Y^i)\}_{i=1}^N$, which record the covariate, chosen decision, and realized cost of past decision epochs. Based on the historical data the decision-maker determines a good decision given the covariate $\boldsymbol{X}_0$.

We highlight three levels of uncertainty in the data-driven optimization represented by the three classic applications of Operations Management.

- In Example 1, the objective function depends on an intermediate variable D (the random demand). The covariate $\boldsymbol{x}$ affects the objective function $f_*(\boldsymbol{x}, \boldsymbol{z})$ through the unknown distribution $D|\boldsymbol{x}$, while the cost function given $\boldsymbol{z}$ and D is known. Therefore, the data informs the relationship $D|\boldsymbol{x}$. In the literature, this type of problem is referred to as "conditional stochastic optimization" (CSO). See Section 2 for a detailed discussion.

- In Example 2, the unknown quantity is the demand function $d_j(\boldsymbol{z}|\boldsymbol{x})$, which depends on the decision variable and the covariate. In similar problems, the unknown function $f_*(\boldsymbol{x}, \boldsymbol{z})$ may have limited known structure, such as the product form of price and demand in Example 2. This allows the use of particular models such as the Multinomial Logit choice model for the unknown component in the objective function.
- In Example 3, the unknown function $f_*(\boldsymbol{x}, \boldsymbol{z})$ is a black box. The data is the only source from which we can learn the function and the optimal solution. Additional assumptions are required to provide a parametric expression of the function or describe its structure. In many problems, a key contributor to the lack of structure is data aggregation. For example, because Y is only recorded daily, it is difficult to build and fit a high-resolution queueing model to map the system input to the cost.

In this study we propose a two-step procedure based on neural networks that can address all three levels of uncertainty. The steps in our framework, referred to as Data-driven Optimization with Neural Networks (DONN), can be informally described as follows:

1. Fit the historical data $\{(\boldsymbol{X}^i, \boldsymbol{Z}^i, Y^i)\}_{i=1}^N$ using neural networks. We treat it as a regression problem with Y being the output and $(\boldsymbol{X}, \boldsymbol{Z})$ being the input. Denote the fitted neural network as $\hat{f}(\boldsymbol{x}, \boldsymbol{z})$.
2. Optimize the fitted neural network $\hat{f}(\boldsymbol{X}_0, \boldsymbol{z})$ over $\boldsymbol{z}$, using the gradient and Hessian matrix of $\hat{f}$, whose closed-form expressions are provided in the paper.

We argue that there are two major benefits of this framework. First, neural networks, especially deep neural networks (DNNs), have seen tremendous success in the recent decade. Although the universal approximation theorem of neural networks has long been established (see Pinkus 1999 for a review of earlier papers), the superb empirical performance of DNNs in fitting any complex function well with sufficient data has recently been confirmed. The computational infrastructure needed to fit large and deep neural networks is widely accessible to scholars and industry practitioners. Our framework, especially Step 1, can benefit from the empirical performance of neural networks: if $\hat{f}$ is easy to fit and approximates f_* well, the resulting optimal decision is also expected to be close to the actual optimal solution. (We prove this rigorously in Section 4.)

Second, we show that the optimization of fitted neural networks is computationally efficient. In particular, we show that the gradient and Hessian matrix of $\hat{f}$ with respect to $\boldsymbol{z}$ can be derived in *closed form*. This allows off-the-shelf continuous optimization tools to be used directly. Although it has been shown that optimizing a ReLU DNN is in general NP-hard (Anderson et al. 2020), gradient-based algorithms allow for quick convergence to local minima, and if the objective function f_* and the fit $\hat{f}$ are convex, to the global minimum.

In studying the performance of our framework, we leverage the generalization bounds of trained DNNs (Ohn and Kim 2019, Sarraf 2020, Farrell et al. 2021) that have been established recently. Under some mild technical conditions, we provide a performance guarantee for the output decision of the framework. More precisely, we link the diminishing generalization error that captures how well the trained neural network fits the objective function as the sample size increases to the performance of the output decision from optimizing the fitted DNN. The translation from the generalization bound to the optimization performance bound is highly nontrivial, as the generalization bound is concerned with the fitting of the neural network in terms of the expectation of the Euclidean norm whereas the optimization bound requires uniform convergence, at least in the neighborhood of the optimum. We identify a set of conditions and prove the optimization bound under them.

We conduct extensive numerical studies to assess the effectiveness of our framework in three cornerstone problems in Operations Management: the data-driven newsvendor problem, the personalized assortment pricing problem, and the call center staffing problem. We compare our framework to existing approaches designed specifically for these problems. For the data-driven newsvendor problem, we use quantile regression and the kernel approach studied in Ban and Rudin (2019) and Bertsimas and Kallus (2020), end-to-end optimization with neural networks studied in Oroojlooyjadid et al. (2020), and the reproducing kernel Hilbert space approach studied in Bertsimas and Koduri (2022) as benchmarks. For the personalized assortment pricing problem, we compare our approach to a two-step procedure where first a demand model is estimated using conventional approaches and then prices are optimized. For the call center staffing problem, we compare against a two-step procedure where the objective function is approximated using Efficiency Driven and Quality and Efficiency Driven regimes (Zeltyn and Mandelbaum 2005, Mandelbaum and Zeltyn 2009), and then staffing levels are optimized. Details are given in Section 6.

We find that when the data size is small, existing approaches may outperform our framework. As the data size increases, however, the DNN can fit the data increasingly well, and as a result, our framework is on par with or outperforms the best of the existing approaches. Surprisingly, this performance is achieved even using neural networks with small sizes, such as 3 hidden layers and 7 nodes on each layer. numerical results suggest that with arguably small-sized DNNs the framework demonstrates competitive performance for complex tasks.

The contributions of this work are fourfold. To the best of our knowledge, we are the first to establish a framework to use DNNs to accurately learn objective functions in a broad array of problems, ranging from well structured to essentially black-box stochastic optimization problems. As such, our framework can be readily applied to a wide range of practical problems of interest in Operations Management. This is a particularly important contribution since a more unified

framework for utilizing DNNs in operations management could lead to a wider adaptation of these powerful tools. Due to its generality and computational efficiency, our approach can be used as a benchmark to assess the performance of other cutting-edge algorithms in a wide range of one-shot stochastic optimization problems.

Second, we characterize the gradient of the prediction value of the trained DNN with respect to the decision variables in closed form. Furthermore, we provide a recursive method for calculating the elements of the Hessian matrix of the trained DNN. This enables state-of-the-art non-linear optimization algorithms (such as quasi-Newton methods) to be readily applicable to the problem of optimizing the prediction value of a DNN. The theoretical properties of these algorithms such as the convergence to a local optimum have been studied extensively in the literature and can be applied to our procedure.

Third, we provide a performance guarantee for the decision made based on optimizing the prediction value of DNNs using the generalization bound of the fitted DNN. The generalization error of DNNs is an active research area. The result allows us to leverage the new advances in this literature and convert it to a performance bound of the output decision by our framework. This theoretical contribution helps explain why a neural network fitting the objective function well also leads to a well-behaved optimal solution, as we observe in the numerical experiments.

Fourth, we conduct a comprehensive numerical study and apply our framework to three classic applications in operations management, as summarized in Examples 1 to 3. We compare to a number of state-of-the-art approaches developed in the literature and demonstrate the generality of DNNs and its superb performance. Because the three applications represent various levels of structure and practical scenarios, this study provides strong empirical evidence for the use of DNNs in data-driven operational decisions.

2. Related Work

Our study is broadly related to four streams of literature. The first stream considers DNNs in general and their statistical learning capabilities specifically. The literature on DNNs is vast and we refer the readers to Fan et al. (2021) and Bartlett et al. (2021) for a comprehensive review. Two of the learning capabilities of DNNs related to our work are their universal approximation power and their generalization bounds. Cybenko (1989) show that neural networks with one hidden layer with the sigmoid activation function can approximate any continuous function arbitrarily well. Subsequently, Leshno et al. (1993) show that feedforward DNNs with a locally bounded piecewise continuous activation function can approximate any continuous function to any degree of accuracy if and only if the network's activation function is not a polynomial. While these universal approximation guarantees are very strong, they do not guarantee that a DNN can be trained. For that, a generalization bound is needed.

The *generalization bound* of a machine learning model is the expected loss of its predictions on the (unseen) test set (Valle-Pérez and Louis 2020, Jiang et al. 2019). Since one cannot measure the expected loss on unseen data, one would like to bound the potential difference between the empirical loss of the training set and the expected loss of a test set, referred to as the *generalization bound*. The establishment of generalization bounds to DNN's has been the topic of much recent literature. See Jiang et al. (2019), Valle-Pérez and Louis (2020), Bartlett et al. (2021). Of direct interest to our work, Hardt et al. (2016) show that for both convex and non-convex loss functions, provided the number of iterations is linear in the number of data points, the generalization bound of a DNN trained with the commonly used Stochastic Gradient Descent (SGD) method is bounded by a vanishing function of the sample size. More recently, for DNNs with ReLU activation functions Farrell et al. (2021) give tight generalization bounds that shrink in the size of the training data. For DNNs with activation functions that are twice differentiable, Sarraf (2020) provides similar tight generalization bounds. Our work complements this line of research by proving a performance guarantee for the decision that leads to the optimal prediction value of trained DNNs, based on their generalization bound.

In the context of optimizing the prediction value of DNNs, multiple recent works optimize trained feedforward ReLU DNNs over their input using mixed-integer programming (MIP) (Dutta et al. 2018, Botoeva et al. 2020, Grimstad and Andersson 2019, Wu et al. 2020, Tsay et al. 2021). Fischetti and Jo (2018) model trained DNNs with ReLU or max pooling activation functions as 0-1 mixed integer linear programs, and use a tightening technique to generate strong adversarial examples. Anderson et al. (2020), model similar trained DNNs as tight mixed integer linear programs that can be used for adversarial verification or solving decision problems. Strong et al. (2021) extend the existing DNN verifiers such as Reluplex algorithm (Katz et al. 2017) into optimizers that can be used to globally optimize trained ReLU DNNs. Perakis and Tsiourvas (2022) devise a scalable approach to globally optimize the trained ReLU DNNs. They utilize the piece-wise linear structure of ReLU to reduce the initial mixed-integer optimization problem into many easy-to-solve linear optimization problems through sampling and prove a sample complexity for their approach. Their framework can potentially be applied to other activation functions. Our paper compliments this body of work by providing guarantees for the performance of the decision obtained from optimizing the prediction value of a trained DNN.

Recent work has applied DNNs to problems of interest in operations management and marketing. Oroojlooyjadid et al. (2020) use the empirical cost of the newsvendor problem as the loss function to train ReLU DNNs and hence predict the optimal newsvendor order quantity. Neghab et al. (2022) apply DNNs to a newsvendor problem with partially unobservable features to predict optimal order quantities. In a study of the effects of various levels of data integration into inventory management,

Huber et al. (2019) use DNNs in a similar way and also as a method for estimating demand. Similarly, Seubert et al. (2020) develop a data-driven system of ordering for a bakery chain using DNNs to estimate demand and optimal order quantity. Babier et al. (2018) use a DNN-based approach to directly predict optimal solutions for constrained optimization problems with linear objective functions such as the portfolio optimization problem. Qi et al. (2020) apply DNNs to multi-period inventory management to predict the optimal replenishment quantity. Liu et al. (2021) study a similar problem, offering a performance guarantee for their framework. Other recent work (Oroojlooyjadid et al. 2022, Gijsbrechts et al. 2022) has shown strong performance in various hard multi-period inventory management problems by integrating DNNs within a reinforcement learning framework. Recently, Gabel and Timoshenko (2022), Cai et al. (2022), Aouad and Desir (2022) use DNNs to predict customer choice in the context of assortment planning and show strong performance in both simulated and real-world data.

There is a body of literature on using DNNs in Reinforcement Learning and contextual multi-armed bandit problems (Li 2017, Mousavi et al. 2018, Collier and Llorens 2018, Xu et al. 2020, Marković et al. 2021, Wang and Kadıoğlu 2021). Simchi-Levi and Xu (2022) consider the online contextual bandit problem with a discrete action space using an off-line regression oracle, possibly a DNN. While they investigate an online optimization problem in which the data arrive sequentially, we study the offline problem. Further, they study a discrete decision space with a specially designed exploration scheme by the decision maker, while we study the continuous decision space and assume sufficient exploration around the optimal solution and the domain. Our approach is better suited to the context of the problems we address. For applications such as assortment pricing, it is unclear if a discrete analysis is practical when the decision space is high-dimensional. Further, it is unreasonable to assume a specific algorithm or exploration scheme that governs the generation of the data when the offline data is potentially provided by another party. Finally, Simchi-Levi and Xu (2022) mainly focus on the theoretical performance of their algorithm, while we are also interested in the empirical performance of the proposed approach.

Our work is also related to the recent literature on data-driven prescriptive analytics, e.g., Ban and Rudin (2019), Oroojlooyjadid et al. (2020), Bertsimas and Kallus (2020), Elmachtoub and Grigas (2022), Bertsimas and Koduri (2022), Kallus and Mao (2022). In this literature, a variant of (1), referred to as conditional stochastic optimization, is considered:

$$\min_{\boldsymbol{z}} \mathbb{E}[c(\boldsymbol{z}, W)|\boldsymbol{X}_0], \tag{4}$$

where $\boldsymbol{z}$ is found to optimize known cost function $c(z, W)$ given covariate $\boldsymbol{X}_0$ and intermediate random variable W. In this framework, the conditional distribution $W|\boldsymbol{X}_0$ needs to be learned from

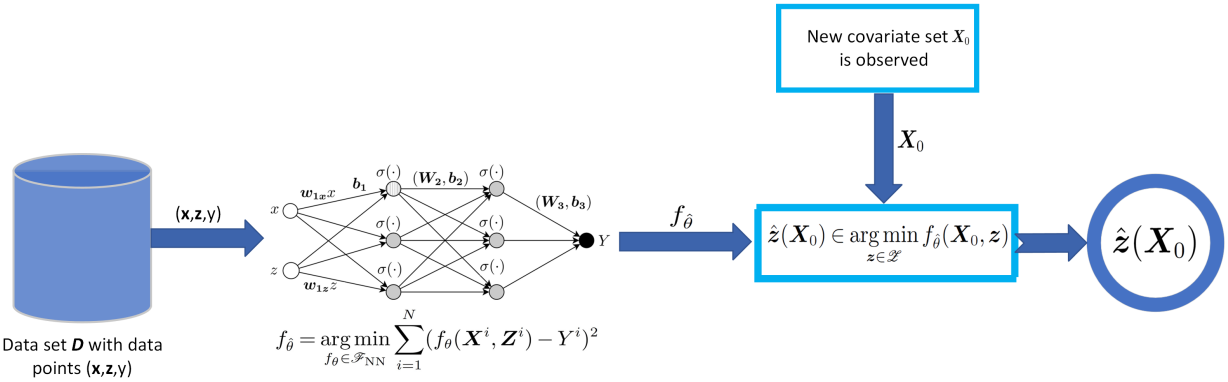


Figure 1 DONN Framework.

the historical data record $\{(\boldsymbol{X}^i, W^i)\}_{i=1}^N$. Our framework differs from this problem in two regards. First, we do not assume the knowledge of the cost function and the structure of an intermediate random variable. We can take (4) as a special case of ours by setting $Y = c(z, W)$. Second, this literature focuses on integrated approaches that do not first learn $W|\boldsymbol{X}$ and then plug in the estimation for downstream optimization. For example, Kallus and Mao (2022) use random forests but propose to split trees based on the cost $c(z, W)$ instead of the predictive error of W. Bertsimas and Koduri (2022) propose to use reproducing kernel Hilbert space to map the covariate directly to a decision based on the data. While Hu et al. (2022) posit that for contextual linear optimization problems, a predict then optimize approach may perform very well, the theoretical and empirical benefits of the integrated approach have been shown in different situations. In contrast, we use neural networks to first fit the objective $f_*(\boldsymbol{x}, \boldsymbol{z})$ and then optimize the fitted neural network over $\boldsymbol{z}$. The less integrated, two-step procedure is needed because of the unstructured problem we consider. We acknowledge that as problem (4) is a special case of (1), the approaches developed in the literature can be more effective than ours when the special structure holds.

3. The Data-driven Optimization with Neural Networks Framework

In this section, we briefly explain how our Data-driven Optimization with Neural Networks (DONN) approach works. We consider a decision maker who observes a covariate $\boldsymbol{X}_0$ and needs to make a decision $\boldsymbol{z}$. As introduced in (1), the problem is

$$\min_{\boldsymbol{z} \in \mathscr{Z}} f_*(\boldsymbol{X}_0, \boldsymbol{z}) = \min_{\boldsymbol{z} \in \mathscr{Z}} \mathbb{E}[Y | \boldsymbol{X}_0, \boldsymbol{z}], \tag{5}$$

where $\mathscr{Z}$ denotes some generic feasible set. In most practical situations, the function $f_*(\boldsymbol{X}_0, \boldsymbol{z})$ is *unknown*. Rather, the decision maker has data from previous decision epochs of the covariate, the decision that was made, and the resulting outcome at that time. Formally, suppose there are N previous decision epochs. For $i = 1 \ldots N$, let $\boldsymbol{X}^i \in \mathbb{R}^{d_X}$ be the observed covariate, $\boldsymbol{Z}^i \in \mathbb{R}^{d_Z}$ be the

decision vector and $Y^i \in \mathbb{R}$ be the observed realized cost in epoch i. The decision maker, using the historical samples of $\{(\boldsymbol{X}^i, \boldsymbol{Z}^i, Y^i)\}_{i=1}^N$, must determine $\boldsymbol{z}$ based on $\boldsymbol{X}_0$, to maximize the unknown function $f_*(\boldsymbol{X}_0, \boldsymbol{z})$. In the DONN approach we first fit the historical data $\{(\boldsymbol{X}^i, \boldsymbol{Z}^i, Y^i)\}_{i=1}^N$ using neural networks, and subsequently, given $\boldsymbol{X}_0$, we optimize the fitted neural network $f_{\hat{\theta}}(\boldsymbol{X}_0, \boldsymbol{z})$ over $\boldsymbol{z}$ using its gradient and Hessian matrix. The approach is depicted in Figure 1.

Note that Y^i may be a noisy observation of the objective function. For example, it may be that $Y^i = f_*(\boldsymbol{X}^i, \boldsymbol{Z}^i) + \epsilon$ for some mean-zero noise ϵ. In any case, the historical samples are generated under the same functional relationship as (5), $\mathbb{E}[Y^i | \boldsymbol{X}^i, \boldsymbol{Z}^i] = f_*(\boldsymbol{X}^i, \boldsymbol{Z}^i)$. The data informs the decision maker of the objective function and can be used to make a decision in a new decision epoch when the covariate $\boldsymbol{X}_0$ is observed.

Recall Examples 1 to 3 introduced in Section 1. In the newsvendor problem, the objective function to minimize is $f_*(\boldsymbol{X}, z) = \mathbb{E}[b(D-z)^+ + h(z-D)^+ | \boldsymbol{X}]$ for covariate $\boldsymbol{X}$ encoding observable market conditions, order quantity z, random demand D whose distribution depends on $\boldsymbol{X}$, and the unit back-ordering and holding costs b and h. The historical samples $\{(\boldsymbol{X}^i, \boldsymbol{Z}^i, Y^i)\}_{i=1}^N$ are collected where Y^i is the realized newsvendor cost $Y^i = b(D^i - Z^i)^+ + h(Z^i - D^i)^+$ for realized demand $D^i \sim D|\boldsymbol{X}^i$. In the personalized pricing problem, to maximize the expected revenue $\sum_{j=1}^{|Z|} z_j d_j(\boldsymbol{z}|\boldsymbol{X})$, the decision maker relies on the data $\{(\boldsymbol{X}^i, \boldsymbol{Z}^i, Y^i)\}_{i=1}^N$, where for customer i, $\boldsymbol{X}^i$ represents the customer feature extracted from the account profile, $\boldsymbol{Z}^i$ is the charged prices for the products, and Y^i is the realized revenue gained from the sold products. The data helps the decision maker to learn the demand function $d_j(\boldsymbol{z}|\boldsymbol{X})$, a customer with feature $\boldsymbol{X}$ given the price vector $\boldsymbol{z}$ for product j. In the call center staffing problem, to minimize the expected cost $\mathbb{E}[Y|\boldsymbol{X}, z]$, the decision maker relies on the data $\{(\boldsymbol{X}^i, Z^i, Y^i)\}_{i=1}^N$, where for day i, $\boldsymbol{X}^i$ represents the observable features, such as customer arrival rate, Z^i is the day $i's$ staffing level, and Y^i is the realized cost at the end of the day.

Remark 1 (Connection to CSO). As stated in Section 2, Example 1 can be cast in the CSO framework. By letting $c(z, W)$ in (4) be the newsvendor cost with $W = D$, the unknown component in the objective function (the demand) is affected only by the market condition $\boldsymbol{X}$. The CSO framework requires data $\{(\boldsymbol{X}^i, D^i)\}_{i=1}^N$ to learn $D|\boldsymbol{X}$, instead of requiring $\{(\boldsymbol{X}^i, \boldsymbol{Z}^i, Y^i)\}_{i=1}^N$, the data structure in our framework. It does not allow for the decision history to affect the unknown values. As such it cannot be used to solve the personalized pricing problem in Example 2, because the pricing decision may affect the unknown demand function, or the staffing problem in Example 3, because the decisions may affect customer abandonment or probability of waiting in unknown ways. In other words, Examples 2 and 3 cannot be cast into (4). Nevertheless, they can be expressed as (5) and solved in the DONN framework. □

We propose to find the best fit objective function $\hat{f}(\boldsymbol{x},\boldsymbol{z})$ from the data $\{(\boldsymbol{X}^i,\boldsymbol{Z}^i,Y^i)\}_{i=1}^N$ within the class of DNN functions $\mathscr{F}_{\text{NN}}$. That is, assuming a quadratic loss function, and a clear specification of $\mathscr{F}_{\text{NN}}$, we seek:

$$\hat{f} = \underset{f\in\mathscr{F}_{\text{NN}}}{\arg\min} \sum_{i=1}^{N} (f(\boldsymbol{X}^i,\boldsymbol{Z}^i) - Y^i)^2. \tag{6}$$

The class $\mathscr{F}_{\text{NN}}$ is the set of functions that can be represented through a particular DNN. The function class $\mathscr{F}_{\text{NN}}$ is determined by the architecture of the network and the form of interactions between the nodes. We consider *feedforward* DNNs. Given a network architecture with L layers and H nodes in each layer, $\mathscr{F}_{\text{NN}}(L,H,\sigma)$, a function within the class is specified by parameters $\theta = (\boldsymbol{W}_{1x},\boldsymbol{W}_{1z},\boldsymbol{b}_1,\boldsymbol{W}_2,\boldsymbol{b}_2,\ldots,\boldsymbol{W}_L,\boldsymbol{b}_L,\boldsymbol{w}_{L+1},b_{L+1})$, where $\boldsymbol{W}_{1x}\in\mathbb{R}^{H\times d_X}$, $\boldsymbol{W}_{1z}\in\mathbb{R}^{H\times d_Z}$, $\boldsymbol{b}_k\in\mathbb{R}^H$ for $k=1,\ldots,L$, $\boldsymbol{W}_k\in\mathbb{R}^{H\times H}$ for $k=2,\ldots,L$, $\boldsymbol{w}_{L+1}\times\mathbb{R}^H$ and $b_{L+1}\in\mathbb{R}$. Given $(\boldsymbol{x},\boldsymbol{z})$ and a parameter θ,

$$f_\theta(\boldsymbol{x},\boldsymbol{z}) = \boldsymbol{w}_{\boldsymbol{L+1}}^\top \boldsymbol{\sigma}\Big(\cdots\boldsymbol{\sigma}\Big(\boldsymbol{W_3}\boldsymbol{\sigma}\big(\boldsymbol{W_2}\boldsymbol{\sigma}(\boldsymbol{W_{1x}}\boldsymbol{x}+\boldsymbol{W_{1z}}\boldsymbol{z}+\boldsymbol{b_1})+\boldsymbol{b_2}\big)+\boldsymbol{b_3}\Big)+\cdots\Big)+b_{L+1}. \tag{7}$$

Here, $\boldsymbol{\sigma}(\cdot):\mathbb{R}^H\to\mathbb{R}^H$ represents the vectorized activation function that applies $\sigma(\cdot):\mathbb{R}\to\mathbb{R}$ elementwise. We discuss the choice of the activation function in Section 5. The numerical results are given using Swish ($\sigma(x)=x/(1+e^{-x})$). For a more complete overview of DNNs used in this paper, we refer the reader to Section EC.1.

Let $\hat{\theta}$ denote the fitted DNN (the optimized weights $\boldsymbol{W}$ and constant terms $\boldsymbol{b}$), we propose to find:

$$\hat{\boldsymbol{z}}(\boldsymbol{X}_0) \in \underset{\boldsymbol{z}\in\mathscr{Z}}{\arg\min}\, f_{\hat{\theta}}(\boldsymbol{X}_0,\boldsymbol{z}). \tag{8}$$

In Section EC.2 we explain in detail how to do so and provide sufficient conditions for gradient-based algorithms, equipped with the exact gradient of $f_{\hat{\theta}}(\boldsymbol{X},\boldsymbol{z})$ with respect to $\boldsymbol{z}$, to converge to a stationary point of $f_{\hat{\theta}}(\boldsymbol{X},\boldsymbol{z})$. For brevity, here we only present our main results on characterizing the gradient and the Hessian matrix of $f_{\hat{\theta}}(\boldsymbol{X}_0,\boldsymbol{z})$ with respect to $\boldsymbol{z}$. To derive the expression for the gradient, we first introduce some additional notation. Let $\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})$ indicate the scalar output of the h^{th} node in the l^{th} layer of the fitted DNN:

$$\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z}) = \sigma\left(\boldsymbol{w}_{\boldsymbol{h,l}}^\top\boldsymbol{\sigma}\Big(\boldsymbol{W}_{l-1}\cdots\boldsymbol{\sigma}\big(\boldsymbol{W_2}\boldsymbol{\sigma}(\boldsymbol{W_{1x}}\boldsymbol{x}+\boldsymbol{W_{1z}}\boldsymbol{z}+\boldsymbol{b_1})+\boldsymbol{b_2}\big)+\cdots+\boldsymbol{b}_{l-1}\Big)+b_{h,l}\right),$$

where $\boldsymbol{w}_{\boldsymbol{h,l}}^\top$ and $b_{h,l}$ are the elements of $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ associated with the focal node. We write the derivative of $\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})$ with respect to $\boldsymbol{z}$ as $\frac{\partial\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial\boldsymbol{z}}$. Moreover, for $l\geq 2$, we write its derivative with respect to its scalar input value as $\sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})$ where:

$$\sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z}) = \frac{\partial\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial\big(\sum_{h'=1}^H w_{h,l}^{h'}\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z})+b_{h,l}\big)}$$

while if $l = 1$, we write

$$\sigma'_{h,1}(\boldsymbol{x}, \boldsymbol{z}) = \frac{\partial \sigma_{h,1}(\boldsymbol{x}, \boldsymbol{z})}{\partial\big(\sum_{j=1}^{d_X} w^j_{h,1\boldsymbol{x}} \boldsymbol{x_j} + \sum_{j=1}^{d_Z} w^j_{h,1\boldsymbol{z}} \boldsymbol{z_j} + b_{h,1}\big)},$$

where $w^{h'}_{h,l}$ are the elements of $\boldsymbol{w_{h,l}}$.

PROPOSITION 1 **(Gradient w.r.t. Inputs in Neural Networks)**. *Given a neural network in the form*

$$f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{w}^\top_{\boldsymbol{L+1}} \boldsymbol{\sigma}\Bigg(\cdots \boldsymbol{\sigma}\Big(\boldsymbol{W_3} \boldsymbol{\sigma}\big(\boldsymbol{W_2} \boldsymbol{\sigma}(\boldsymbol{W_{1x}} \boldsymbol{x} + \boldsymbol{W_{1z}} \boldsymbol{z} + \boldsymbol{b_1}) + \boldsymbol{b_2}\big) + \boldsymbol{b_3}\Big) + \cdots \Bigg) + b_{L+1},$$

we have

$$\frac{\partial f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{\partial \boldsymbol{z}} = \boldsymbol{w}^\top_{\boldsymbol{L+1}} \times \operatorname{diag}\big(\sigma'_{1,L}(\boldsymbol{x}, \boldsymbol{z}), \ldots, \sigma'_{H,L}(\boldsymbol{x}, \boldsymbol{z})\big) \times \cdots \times \boldsymbol{W_2} \times \operatorname{diag}\big(\sigma'_{1,1}(\boldsymbol{x}, \boldsymbol{z}), \ldots, \sigma'_{H,1}(\boldsymbol{x}, \boldsymbol{z})\big) \times \boldsymbol{W_{1z}}, \tag{9}$$

if σ is continuously differentiable function.

While Proposition 1 can help optimize the trained DNN by providing the gradient in closed form, it is also of independent interest as it provides managerial interpretability for the decisions made based on the DNN. Specifically, given observed covariates $\boldsymbol{x}$ and for action $\bar{\boldsymbol{z}}$, Proposition 1 provides the improvements in the prediction value of the objective function per unit change of each of the decision variables. For example, in the context of a personalized assortment pricing problem, for a vector of prices, Proposition 1 allows the decision maker to gauge the predicted change in revenue per unit change of price for each of the products in the assortment.

We note that state-of-the-art non-linear optimization algorithms can leverage the Hessian of the objective function to be readily applicable to Problem (8). We conclude this section by showing that the Hessian of the fitted DNN can be recursively calculated in closed-form, under the condition that $\sigma(\cdot)$ be twice continuously differentiable. To do so, we first need to introduce some additional notation. If $l \geq 2$, we write $\sigma''_{h,l}(\boldsymbol{x}, \boldsymbol{z})$ to denote the second derivative of $\sigma(\cdot)$ with respect to its scalar input value, where

$$\sigma''_{h,l}(\boldsymbol{x}, \boldsymbol{z}) = \frac{\partial \sigma'_{h,l}(\boldsymbol{x}, \boldsymbol{z})}{\partial\big(\sum_{h'=1}^{H} w^{h'}_{h,l} \sigma_{h',l-1}(\boldsymbol{x}, \boldsymbol{z}) + b_{h,l}\big)},$$

while if $l = 1$, we write

$$\sigma''_{h,1}(\boldsymbol{x}, \boldsymbol{z}) = \frac{\partial \sigma'_{h,1}(\boldsymbol{x}, \boldsymbol{z})}{\partial\big(\sum_{j=1}^{d_X} w^j_{h,1\boldsymbol{x}} \boldsymbol{x_j} + \sum_{j=1}^{d_Z} w^j_{h,1\boldsymbol{z}} \boldsymbol{z_j} + b_{h,1}\big)}.$$

PROPOSITION 2 **(Hessian w.r.t. Inputs in Neural Networks)**. *Given a neural network in the form*

$$f_{\hat{\theta}}(\boldsymbol{x},\boldsymbol{z}) = \boldsymbol{w}_{\boldsymbol{L+1}}^{\top}\boldsymbol{\sigma}\Big(\cdots\boldsymbol{\sigma}\Big(\boldsymbol{W_3}\boldsymbol{\sigma}\big(\boldsymbol{W_2}\boldsymbol{\sigma}(\boldsymbol{W_{1x}}\boldsymbol{x}+\boldsymbol{W_{1z}}\boldsymbol{z}+\boldsymbol{b_1})+\boldsymbol{b_2}\big)+\boldsymbol{b_3}\Big)+\cdots\Big)+b_{L+1},$$

we have the derivative of the h^{th} node of the first layer as

$$\frac{\partial\sigma_{h,1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j} = w_{h1}^{j}\sigma'_{h,1}(\boldsymbol{x},\boldsymbol{z}),$$

and we have the second derivative of the h^{th} node of the first layer as:

$$\frac{\partial^2\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k} = w_{h1}^{j}w_{h1}^{k}\sigma''_{h,l}(\boldsymbol{x},\boldsymbol{z}).$$

Moreover, we have the derivative of the h^{th} node of the l^{th} $(l\geq 2)$ layer as

$$\frac{\partial\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j} = \sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})\Big(\sum_{h'=1}^{H} w_{h,l}^{h'}\frac{\partial\sigma_{h,l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j}\Big),$$

and we have the second derivative of the h^{th} node of the l^{th} $(l\geq 2)$ layer as:

$$\frac{\partial^2\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k} = \sigma''_{h,l}(\boldsymbol{x},\boldsymbol{z})\Big(\sum_{h'=1}^{H} w_{h,l}^{h'}\frac{\partial\sigma_{h,l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_k}\Big)\Big(\sum_{h'=1}^{H} w_{h,l}^{h'}\frac{\partial\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j}\Big)+\sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})\Big(\sum_{h'=1}^{H} w_{h,l}^{h'}\frac{\partial^2\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k}\Big).$$

Finally, we have the second derivative of $f_{\hat{\theta}}$ as:

$$\frac{\partial^2 f_{\hat{\theta}}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k} = \sum_{h=1}^{H} w_{L+1}^{h}\frac{\partial^2\sigma_{h,L}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k},$$

if $\sigma(\cdot)$ is twice continuously differentiable and w_{L+1}^{h} are the elements of $\boldsymbol{w_{L+1}}$.

4. Generalization Bound to Optimization Performance

The investigation of the predictive power of (deep) neural networks has been an active research area. See Bartlett et al. (2021), Fan et al. (2021) for recent reviews of the advances in this area. One of the most widely used notion of the predictive power is the *generalization bound* of neural networks. In this section, we take a generalization bound of a neural network as given in the literature and convert it to the performance bound for the data-driven decision. That is, we attempt to answer the following question: If a neural network fits the objective function accurately with enough data, will the DONN framework output high-quality decisions?

We next introduce the technical conditions for the theoretical analysis. The first assumption is concerned with how the data is generated.

ASSUMPTION 1 **(i.i.d. samples)**. *The samples $\{(\boldsymbol{X}^i,Y^i,\boldsymbol{Z}^i)\}_{i=1}^{N}$ are i.i.d. copies of $(\boldsymbol{X},Y,\boldsymbol{Z})$, where $\|\boldsymbol{X}\|_\infty,\|\boldsymbol{Z}\|_\infty\in[-1,1]$, $Y\in[-M,M]$ for some $M>0$.*

The i.i.d. assumption is rather standard in statistical analysis. We assume all the data are bounded, presumably after normalization. Similarly, we restrict all decisions to $[-1, 1]$. By the definition of f_* in (5), Assumption 1 implies that $|f_*| \le M$.

In decision epoch i, the decision maker first observes the covariate $\boldsymbol{X}^i$, then makes a decision $\boldsymbol{Z}^i$, possibly depending on $\boldsymbol{X}^i$. Finally, the realized objective Y is observed. Therefore, one can think of the data generating process for the distribution of $(\boldsymbol{X}, Y, \boldsymbol{Z})$ in Assumption 1, as $\boldsymbol{X} \sim \mu_X$, $\boldsymbol{Z} \sim \mu_{Z|X}$, and $Y \sim \mu_{Y|X,Z}$. A typical form is $Y = f_*(\boldsymbol{X}, \boldsymbol{Z}) + \epsilon$ for some independent and mean-zero noise ϵ, although this is not assumed in this section. We use $\mu_{X,Y}$ to denote the marginal distribution of $(\boldsymbol{X}, \boldsymbol{Z})$.

The next assumption imposes a generalization bound of the neural network. In particular, after fitting an objective function with a neural network as explained in Section EC.1, we assume:

ASSUMPTION 2 **(Generalization bound)**. *The fitted neural network $f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ satisfies:*

$$\mathbb{E}_{\mu_{X,Z}}[(f_*(\boldsymbol{X}, \boldsymbol{Z}) - f_{\hat{\theta}}(\boldsymbol{X}, \boldsymbol{Z}))^2] \le \delta_N,$$

for some $\delta_N > 0$.

Note that in Assumption 2 we treat $f_{\hat{\theta}}$ as given and take the expectation over the random $(\boldsymbol{X}, \boldsymbol{Z})$. Also, $f_{\hat{\theta}}$ itself is dependant on the random sample $\{(\boldsymbol{X}^i, Y^i, \boldsymbol{Z}^i)\}_{i=1}^N$ and, in many studies, the generalization bound is derived as a high-probability event (see Fan et al. (2021) and Bartlett et al. (2021) for a review). We omit the dependence on the random sample in this paper. The generalization bound δ_N is typically a diminishing term in N. For a given data set and a trained DNN using this data, a bound δ_N can be calculated using bounds developed in many recent studies. It remains an active research area to investigate how the network architecture such as width, depth, and activation function affects this bound. For example, in Theorem 2 of Farrell et al. (2021), δ_N scales with $WL\log(W)\log(N)/N$, where L is the depth and W is the total number of parameters in the neural network with ReLU activation functions. Similarly, Sarraf (2020) provides generalization bounds that are architecture dependent for DNNs with two times continuously differentiable activation functions, such as sigmoid and Swish activation functions. In this study, we take δ_N as given and derive optimization performance based on the generalization bound.

The next assumption imposes the Lipschitz continuity of the objective function, which is standard in the literature.

ASSUMPTION 3 **(Lipschitz continuity)**. *$f_*(\boldsymbol{x}, \boldsymbol{z})$ is K_*-Lipschitz continuous with respect to the Euclidean norm.*

Similar to Assumption 3, we also require the Lipschitz continuity of the fitted neural network $f_{\hat{\theta}}$. If the activation function is Lipschitz continuous, which is true for most activation functions including ReLU and Swish, then $f_{\hat{\theta}}$ is Lipschitz continuous as it is a composition of activation functions and linear transformations. Moreover, an analytical upper bound on the Lipschitz constant of $f_{\hat{\theta}}$, $K_{\hat{\theta}}$ was established by Szegedy et al. (2013),

$$K_{\hat{\theta}} \leq (K_\sigma)^L \Big(\prod_{i=1}^{L} \|\boldsymbol{W_l}\|_2\Big) \|\boldsymbol{w}_{l+1}\|_2.$$

Here K_σ is the Lipschitz constant of the activation function used in $f_{\hat{\theta}}$ ($K_{\text{ReLU}} = 1$, $K_{\text{sigmoid}} = 1$, and $K_{\text{Swish}} = 1.1$), $\|\boldsymbol{W}_l\|_2$ indicates the spectral norm of the weight matrix of the l^{th} layer of the network and L is the number of hidden layers. Based on this result and Assumption 3, it follows that $|f_*(\boldsymbol{x}, \boldsymbol{z}) - f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})|$ will also be Lipschitz continuous, with a Lipschitz constant of

$$K = K_* + K_{\hat{\theta}}. \tag{10}$$

Our objective is to translate the performance of the fitted neural network (Assumption 2) to the performance of the prescribed decision for $\boldsymbol{X}_0$, that is, $\hat{\boldsymbol{z}}(\boldsymbol{X}_0)$ defined in (8). The key difference between the two measures is that the generalization bound is a *global* property concerned with the fit of the DNN in the $(\boldsymbol{x}, \boldsymbol{z})$ space according to the distribution $\mu_{X,Z}$, while the performance of $\hat{\boldsymbol{z}}(\boldsymbol{X}_0)$ primarily depends on the *local* fitting of the DNN near the optimal solution. Our goal is to show that the prescribed decision will be close to the actual optimal solution. To do so, we impose the following assumption.

Assumption 4 **(Unique maximizer)**. *Given $\boldsymbol{x}_0$, $\boldsymbol{z} = \boldsymbol{z}^*(\boldsymbol{x}_0)$ is the unique minimizer of $f_*(\boldsymbol{x_0}, \boldsymbol{z})$.*

Based on the generalization bound and Assumption 4 we would like to control the error $\|\hat{\boldsymbol{z}}(\boldsymbol{x}_0) - \boldsymbol{z}^*(\boldsymbol{x}_0)\|$. (We use $\boldsymbol{x}_0$ instead to de-emphasize the random nature of $\boldsymbol{X}_0$ and focus on a generic covariate). Our first step is to narrow down the generalization bound in a neighborhood of $(\boldsymbol{x}_0, \boldsymbol{z}^*(\boldsymbol{x}_0))$. In particular, we show that $f_{\hat{\theta}}$ approximates f_* *uniformly* well in the neighborhood of $(\boldsymbol{x}_0, \boldsymbol{z}^*(\boldsymbol{x}_0))$. A uniform bound is needed because the generalization bound only bounds the expected error and does not rule out spurious *spikes* in the fitted DNN. These spikes do not violate the generalization bound but may significantly distort $\hat{\boldsymbol{z}}(\boldsymbol{x}_0)$. By establishing uniform convergence we can use the Lipschitz continuity of f_* and $f_{\hat{\theta}}$ to control the error $\|\hat{\boldsymbol{z}}(\boldsymbol{x}_0) - \boldsymbol{z}^*(\boldsymbol{x}_0)\|$.

The next lemma is key to establishing uniform convergence. We show that for a given covariate $\boldsymbol{x_0}$ and a corresponding decision $\boldsymbol{z}$, the output of the DNN can accurately predict $f_*(\boldsymbol{x}, \boldsymbol{z})$ within a ball B of radius b that contains $(\boldsymbol{x_0}, \boldsymbol{z})$ if the ball is sufficiently explored in the data, i.e., $\mu_{X,Z}(B)$ is bounded away from zero.

LEMMA 1. *Suppose Assumptions 1, 2 and 3 hold. Consider a ball* $B \subset \mathbb{R}^{d_X+d_Z}$ *of radius* b*. Then* $\max_{(\boldsymbol{x},\boldsymbol{z})\in B} |f_*(\boldsymbol{x},\boldsymbol{z}) - f_{\hat{\theta}}(\boldsymbol{x},\boldsymbol{z})| \le (\delta_N/S)^{1/2} + 2bK$*, where* $S = \int_{(\boldsymbol{x},\boldsymbol{z})\in B} d\mu_{X,Z}$ *and* K *is the Lipschitz constant defined in* (10)*.*

The bound exhibits two opposing effects of the size of the ball B, represented by b and S. The first term is decreasing while the second term is increasing in the size of B. Intuitively, on the one hand, a larger radius b would mean the DNN has seen more samples from the ball under consideration and hence better generalizes to samples from the ball. On the other hand, a large radius would allow for larger changes in the value of f_* across the ball, which is only limited by K_*, the Lipschitz constant of f_*, making prediction harder for the DNN. We will choose b carefully when using Lemma 1 in the proof of the main theorem.

To apply Lemma 1 to the neighborhood of $(\boldsymbol{x}_0, \boldsymbol{z}^*(\boldsymbol{x}_0))$, we require $\mu_{X,Z}$, to be bounded away from zero around $(\boldsymbol{x}_0, \boldsymbol{z}^*(\boldsymbol{x}_0))$. This would allow us to choose b freely as long as it is sufficiently small.

ASSUMPTION 5 **(Exploration around the optimal solution)**. *There exists* $r > 0$ *such that for* $(\boldsymbol{x},\boldsymbol{z}) \in \mathbb{R}^{d_X} \times \mathbb{R}^{d_Z}$ *and* $\|(\boldsymbol{x},\boldsymbol{z}) - (\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0}))\|_2 \le r$*, we have* $c_1 > 0$ *and the density* $d\mu_{X,Z} \ge c_1$*.*

In practice, Assumption 5 ensures that the covariates around $\boldsymbol{x}_0$ have positive probability to have appeared in the data, $z^*(\boldsymbol{x}_0)$ is strictly in the interior of $[-1,1]^{d_Z}$, and the decisions around it have been sampled sufficiently when the data is large. Combining Lemma 1 and Assumption 5, we can show uniform convergence within a vicinity of $(\boldsymbol{x}_0, \boldsymbol{z}^*(\boldsymbol{x}_0))$.

To control the error of $\|\hat{\boldsymbol{z}}(\boldsymbol{x}_0) - \boldsymbol{z}^*(\boldsymbol{x}_0)\|$, we need to rule out those candidates for $\hat{\boldsymbol{z}}(\boldsymbol{x}_0)$ that are far away from $\boldsymbol{z}^*(\boldsymbol{x}_0)$. In particular, for the optimal decision variable obtained from the DONN approach to be close to the true optimal decision, the value of f_* should increase enough when the decision variable is far away from the true optimal solution. Moreover, there needs to be enough exploration in the data regarding those far away regions so the DNN can generalize to them. Assumption 6 lays the ground work for this. From a practical point of view, Assumption 6 means that f_* admits a large enough increase when considering solutions far away from the optimal, and that the training data fed to the DNN includes sub-optimal decisions as well, allowing the DNN to generalize to those regions.

ASSUMPTION 6 **(Exploration of suboptimal regions)**. *For all* z *such that* $\|\boldsymbol{z} - \boldsymbol{z}^*(\boldsymbol{x_0})\| \ge r$*, there exists* $c_2 > 0$ *such that* $f_*(\boldsymbol{x_0},\boldsymbol{z}) - f_*(\boldsymbol{x_0}, \boldsymbol{z}^*(x_0)) \ge c_2$*. Moreover, for all* $(\boldsymbol{x_0},\boldsymbol{z}) \in [-1,1]^{d_X+d_Z}$ *such that* $\|(\boldsymbol{x_0},\boldsymbol{z}) - (\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0}))\| \ge r$*, there exists* $c_3 > 0$ *and* $c_4 > 0$ *such that* $\int_{(\boldsymbol{x},\boldsymbol{w}):\|(\boldsymbol{x},\boldsymbol{w})-(\boldsymbol{x_0},\boldsymbol{z})\|\le c_3} d\mu_{X,Z} \ge c_4$ *and we have* $c_2 > 2(c_3 + r)K$*.*

Our goal to control the error of $\|\hat{\boldsymbol{z}}(\boldsymbol{x}_0) - \boldsymbol{z}^*(\boldsymbol{x}_0)\|$ using δ_N can be stated as finding a smallest r_{δ_N} such that $\|\hat{\boldsymbol{z}}(\boldsymbol{x}_0) - \boldsymbol{z}^*(\boldsymbol{x}_0)\| \le r_{\delta_N}$. To do so, we need to ensure f_* admits some local curvature in close proximity of $(\boldsymbol{x}_0, \boldsymbol{z}^*(\boldsymbol{x}_0))$. The following assumption, which is relatively mild in only imposing a locally quadratic decay on f_*, ensures this.

ASSUMPTION 7 **(Local convexity)**. *Assume $\boldsymbol{x_0}$ is such that for all $\boldsymbol{z} \in \mathbb{R}^{d_Z}$ where $\|\boldsymbol{z} - \boldsymbol{z}^*(\boldsymbol{x_0})\| \le r$, there exists $c_5 > 0$ such that we have $f_*(\boldsymbol{x_0}, \boldsymbol{z}) \ge f_*(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0})) + c_5\|\boldsymbol{z} - \boldsymbol{z}^*(\boldsymbol{x_0})\|^2$.*

To ensure the result regarding the error of $\|\hat{\boldsymbol{z}}(\boldsymbol{x}_0) - \boldsymbol{z}^*(\boldsymbol{x}_0)\|$ and the performance guarantee of $\hat{\boldsymbol{z}}(\boldsymbol{x}_0)$ will be self-contained, we first define the volume of a generic $d_{\boldsymbol{X}} + d_{\boldsymbol{Z}}$ dimensional ball.

DEFINITION 1. $L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} := \pi^{(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})/2}/\Gamma\big((d_{\boldsymbol{X}} + d_{\boldsymbol{Z}})/2 + 1\big)$. Thus $L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} r^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}$ is the volume of a $d_{\boldsymbol{X}} + d_{\boldsymbol{Z}}$ dimensional ball of radius r (Here Γ is Euler's gamma function and π is the number pi.)

PROPOSITION 3. *Let Assumptions 1, 2, 3 and 4 hold. For some $\boldsymbol{x_0}$ that satisfies the conditions of Assumption 5, let $\hat{\boldsymbol{z}}(\boldsymbol{x_0}) \in \arg\min_{\boldsymbol{z}\in[-1,1]^{d_Z}} f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z})$. If Assumptions 6, and 7 hold, and δ_N is small enough, then we have*

$$\|\hat{\boldsymbol{z}}(\boldsymbol{x_0}) - \boldsymbol{z}^*(\boldsymbol{x_0})\| \le \left[\frac{2}{c_5}\left(\frac{\delta_N(d_{\boldsymbol{X}} + d_{\boldsymbol{Z}})}{4Kc_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}} + 2K\left(\frac{(d_{\boldsymbol{X}} + d_{\boldsymbol{Z}})^2\delta_N}{16K^2c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}\right)^{\frac{1}{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2}}\right)\right]^{\frac{1}{2}}.$$

Moreover, we have:

$$|f_*(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0})) - f_*(\boldsymbol{x_0}, \hat{\boldsymbol{z}}(\boldsymbol{x_0}))| \le K_*\left[\frac{2}{c_5}\left(\frac{\delta_N(d_{\boldsymbol{X}} + d_{\boldsymbol{Z}})}{4Kc_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}} + 2K\left(\frac{(d_{\boldsymbol{X}} + d_{\boldsymbol{Z}})^2\delta_N}{16K^2c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}\right)^{\frac{1}{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2}}\right)\right]^{\frac{1}{2}}.$$

The proof of Proposition 3 which appears in Section EC.3 in the E-companion, is rather involved. We provide some high level intuition here. We first develop an upper bound on $f_{\hat{\theta}}(\boldsymbol{x}_0, \hat{\boldsymbol{z}}(\boldsymbol{x}_0))$ and a lower bound on $f_{\hat{\theta}}(\boldsymbol{x}_0, \boldsymbol{z})$ for any $\boldsymbol{z}$ that has a Euclidean distance of more than r from $\boldsymbol{z}^*(\boldsymbol{x}_0)$. Then, by showing the lower bound is larger than the upper bound, we rule out faraway candidates for $\hat{\boldsymbol{z}}(\boldsymbol{x}_0)$. Furthermore, by optimizing the radius of the ball around $(\boldsymbol{x}_0, \boldsymbol{z}^*(\boldsymbol{x}_0))$, we find the tightest upper bound implied by Lemma 1 for $f_{\hat{\theta}}(\boldsymbol{x}_0, \boldsymbol{z}^*(\boldsymbol{x}_0))$. However, even with Assumption 7, the increase in the value f_* outside of this ball may not be large enough to rule out candidate points outside of this ball for $\hat{\boldsymbol{z}}(\boldsymbol{x}_0)$. Hence, we next find the smallest ball around $(\boldsymbol{x}_0, \boldsymbol{z}^*(\boldsymbol{x}_0))$ that the value of f_* increases enough outside of it to rule out candidate points. The radius of this latter ball serves as the bound for $\|\hat{\boldsymbol{z}}(\boldsymbol{x}_0) - \boldsymbol{z}^*(\boldsymbol{x}_0)\|$.

The bound in Proposition 3 is dependent on the dimensions of the covariates and decision variables, d_X and d_Z, respectively, in an intricate way. Both terms of the bound have a term $L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}$ appearing in the denominator, which shrinks as $d_X + d_Z$ grows very large. Intuitively, this would suggest a need for more data samples for the same performance guarantee in larger dimensions.

Proposition 3 constitutes our major theoretical result. It provides a performance guarantee for $\hat{\boldsymbol{z}}(\boldsymbol{x_0})$ that approaches 1 when δ_N approaches zero. Moreover, it controls the distance of $\hat{\boldsymbol{z}}(\boldsymbol{x_0})$ from the true optimal solution $\boldsymbol{z}^*(\boldsymbol{x_0})$, and this distance decreases in δ_N. This distance decreases in δ_N under four conditions. The first and second conditions are primarily concerned with ensuring the estimation quality of the DNN. They amount to assuming that the density of $\mu_{X,Z}$ around $\boldsymbol{z}^*(\boldsymbol{x_0})$ is bounded away from zero and that the decision maker occasionally has taken sub-optimal decisions. As discussed before, these assumptions are not particularly limiting assumptions. The third condition implies f_* has a minimum gradient at $\boldsymbol{x_0}$ for decisions that are far away (in the Euclidean sense) from $\boldsymbol{z}^*(\boldsymbol{x_0})$. This assumption is also not too limiting, since otherwise any learning approach may struggle to differentiate between the noise and the quality of the past decisions. Finally, the last conditions translates into f_* admitting a quadratic increase in proximity of $\boldsymbol{z}^*(\boldsymbol{x_0})$.

We point out that while Proposition 3 provides a performance guarantee for $\hat{\boldsymbol{z}}(\boldsymbol{x_0})$, it does not guarantee convergence to $\boldsymbol{z}^*(\boldsymbol{x_0})$ even though the numerical experiments in Section 6 suggest such a convergence. This is because, while the error of $\|\hat{\boldsymbol{z}}(\boldsymbol{x}_0) - \boldsymbol{z}^*(\boldsymbol{x}_0)\|$ shrinks with δ_N, it is possible that the Lipschitz constant of the DNN, $K_{\hat{\theta}}$, will increase as δ_N shrinks. Because using theoretical weight regularization in training DNNs is not necessarily well justified (Srivastava et al. 2014, Farrell et al. 2021), such an increase is possible. However, there is a growing body of literature, (Aziznejad et al. 2020, Gouk et al. 2021, Pauli et al. 2021) on regularizing the Lipschitz constant of DNNs while training, to ensure the trained DNNs will have a bounded Lipschitz constant. Moreover, Finlay et al. (2018) have recently shown that Lipschitz regularized DNNs generalize and converge. Therefore, in contexts where convergence to the true optimal decisions is necessary, our framework can be used by integrating trained DNNs that are Lipschitz regularized. Whether this leads to a stronger performance in real world problems is a question for future research.

Remark 2. We note that assuming covariate vector $\boldsymbol{X}$ is continuous is without loss of generality. In particular, if some or all of the covaiates are discrete, indicated by $\tilde{\boldsymbol{X}}$, one can scale δ_N inversely proportionate to the probability of having observed a specific value for the discrete covariates. For example, upon observing $\tilde{\boldsymbol{x}}_0$, if $\gamma(\tilde{\boldsymbol{x}}_0) > 0$ is a valid lower bound on the probability of observing $\tilde{\boldsymbol{x}}_0$, all of the results will hold, by rescaling δ_N to be $\frac{\delta_N}{\gamma(\tilde{\boldsymbol{x}}_0)}$. □

5. The Choice of Activation Functions

The choice of activation functions has a substantial impact on the performance of DNNs, as documented in a number of empirical studies (for example, see Ramachandran et al. 2017). Although the field of deep learning is evolving rapidly and new activation functions are constantly proposed, ReLU remains one of the most popular choices (Ramachandran et al. 2017, Agarap 2018, Nwankpa et al. 2018). In addition to the empirical success, DNNs with ReLU activation functions possess benign theoretical properties, achieving the minimax convergence rate (Schmidt-Hieber 2020).

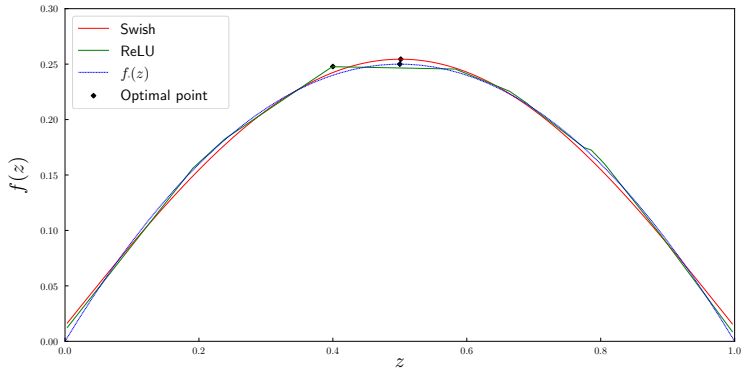


Figure 2 The illustration of the prediction value for DNNs using ReLU and Swish activation functions.

In our framework, since the quality of the data-driven decision is closely related to how well f_* can be learned from the data and ReLU has had tremendous empirical success in deep learning, it seems a natural choice to use ReLU. However, we point out a surprising observation that is specific to our framework using DNNs for data-driven decision-making: using ReLU may lead to *worse* data-driven decisions compared to smooth activation functions such as sigmoid and Swish. We illustrate this observation using the following toy example.

EXAMPLE 4 (USING RELU LEADS TO WORSE OUTCOMES). Consider a scenario without covariates and a one-dimensional objective function $f_*(z) = z - z^2,\ z \in [0, 1]$. The function is shown by the dashed lines in Figure 2. We compare the ability of two DNNs to fit this function. The DNNs have one hidden layer and 29 nodes. They differ only in their activation function, one with ReLU and one with Swish. We then sample a large number of observations of $f_*(z)$ and fit the DNNs. The ReLU network has a smaller out of sample prediction error (MSE 7.24×10^{-6}) compared to that of the Swish network (MSE 1.86×10^{-5}). After fitting the DNNs, we solve for the optimal $\hat{z}$ for the DNNs. In Figure 2 we observe the optimal $\hat{z}$ for the Swish network is much closer to z^* than that of the ReLU network.

6. Numerical Results

In this section, we numerically compare the performance of our methodology with other state-of-the-art approaches. The results demonstrate strong empirical performance of our framework, especially for large training datasets. Computationally, the required solution time for our approach increases more slowly in the data size compared with some of the benchmarks.

6.1. Performance on the Newsvendor Problem with Features

In this section, we investigate the performance of the DONN methodology on the newsvendor problem (Example 1). We create the synthetic dataset using the following setup. There are 10

products across 10 stores, and three observed covariates: the temperature, t, generated from a normal distribution with $t \sim N(20,4)$, the humidity, ψ, generated from a uniform distribution $\psi \sim U[0,10]$, and day of the day of the week, generated from a uniform distribution $d \sim U\{0,\cdots,6\}$. For given covariate $\boldsymbol{x} = (t,\psi,d)$, for product k at store j, the demand is distributed as

$$D|\boldsymbol{x} \sim N\big(100 + (t-20) + 20 \times (\psi-8)^+ + 5 \times \mathbb{I}(d = weekend) + \beta_k - \beta_j, 16\big) \tag{11}$$

For any k and j, β_k and β_j are randomly chosen from a normal distribution with $N(10,4)$. This is a similar setting to Bertsimas and Van Parys (2021) and Ban and Rudin (2019), and the extension to a multi-product, multi-store setting is inspired by Huber et al. (2019) and Seubert et al. (2020) which use data from a bakery chain with multiple bread products. We use a nonlinear trend $(\psi-8)^+$ to reflect the increased demand due to higher humidity and the $\mathbb{I}(d = weekend)$ for higher demand during weekends and demonstrate the potential pitfall for linear methods when the model is misspecified. As we have 10 products and 10 stores, on any given day 100 samples will be collected. We emphasize that in the synthetic data, on a given day products in the same store will experience the same temperature and humidity. We consider the cost function

$$\mathbb{E}[h(D-z)^+ + b(z-D)^+|\boldsymbol{X}], \tag{12}$$

where $b = 10$ and $h = 1$ are, respectively, the unit back-ordering and holding costs.

In each decision epoch in the sample, we record the covariate $\boldsymbol{X}_i = (t_i,\psi_i,d_i)$, the realized demand D_i generated according to (11), the ordered quantity that is uniformly distributed $Z_i \sim U([80,180])$ and the realized newsvendor cost $Y_i = h(D_i - z_i)^+ + b(z_i - D_i)^+$. In some of the benchmark methods described below, $(\boldsymbol{X}_i, D_i)$ along with b and h are required, instead of $(\boldsymbol{X}_i, Z_i, Y_i)$. However, because of the structure of the newsvendor problem, one can simulate $(\boldsymbol{X}_i, Z_i, Y_i)$ from $(\boldsymbol{X}_i, D_i)$ with a set of counterfactual and exploratory order quantities using the realized newsvendor cost Y_i. Therefore, the DONN framework has less stringent requirements for how the the data is generated.

We compare our framework to four benchmark policies that have been shown to perform well.

Quantile Regression (QR). The newsvendor problem is closely related to quantile regression because the optimal solution to (12) is the $b/(b+h)$-th quantile of the distribution $D|\boldsymbol{X}$. QR postulates a linear function of the covariate to predict the quantile. In particular, QR outputs a vector $\beta \in \mathbb{R}^{d_X}$ and recommends a decision $z_{QR}(\boldsymbol{X}_0) = \boldsymbol{X_0}^\top \beta$, where β minimizes

$$\beta = \arg\min \frac{1}{N} \sum_{i=1}^{N} [b(D_i - \boldsymbol{X}_i^\top \boldsymbol{\beta})^+ + h(\boldsymbol{X}_i^\top \boldsymbol{\beta} - D_i)^+]. \tag{13}$$

We use one-hot encoding for the day of the week in QR. Despite this, note that QR essentially misspecifies the model because the actual optimal solution depends on $(\psi-8)^+$ and cannot be

represented by a linear function. This is an intentional design: if the model is indeed linear, then QR tends to outperform most other methods due to the data efficiency of linear models. On the other hand, real-world applications rarely follow exact linear patterns. We consider arguably mild misspecification because $(\psi - 8)^+$ is piecewise linear with two segments. As we shall see, QR is significantly outperformed by other methods including DONN.

Kernel Optimization (KO). KO is a non-parametric approach to solve (2). In this approach, the order quantity is estimated using a weighted sample average approximation from historical demand observations. The weights are determined based on the distance between the new covariate $\boldsymbol{X}_0$ and the historical covariates:

$$K_i = \frac{K_w(\boldsymbol{X}_0 - \boldsymbol{X}_i)}{\sum_{j=1}^N K_w(\boldsymbol{X}_0 - \boldsymbol{X}_j)}, \quad K_w(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\|u\|_2^2}{2w}\right), \tag{14}$$

where $K_w(\cdot)$ is the Gaussian kernel, and w is the kernel bandwidth that has to be tuned from the data. After calculating K_i for $i = 1, \ldots, N$, the order quantity for $\boldsymbol{X}_0$ is solved from $\inf\left\{z : \sum_{i=1}^N K_i \mathbb{I}(D_i \le z) \ge b/(b+h)\right\}$. We note that this method is also studied in Ban and Rudin (2019), Bertsimas and Kallus (2020) and Oroojlooyjadid et al. (2020).

End-To-End Optimization with Neural Networks (EENN). This approach is proposed in Oroojlooyjadid et al. (2020). Here the recommended decision is $z_{EENN}(\boldsymbol{X}_0) = \hat{f}(\boldsymbol{X_0})$, where $\hat{f}$ is a DNN that minimizes $\hat{f} = \arg\min_{f \in \mathscr{F}_{\text{NN}}} \frac{1}{N} \sum_{i=1}^N [b(D_i - \hat{f}(\boldsymbol{X_i}))^+ + h(\hat{f}(\boldsymbol{X_i}) - D_i)^+]$.

Reproducing Kernel Hilbert Space (RKHS). This approach is proposed in Bertsimas and Koduri (2022). It can be postulated that the order quantity for covariate $\boldsymbol{X}_0$ has the form $\sum_{i=1}^N K_w(\boldsymbol{X}_0 - \boldsymbol{X}_i) a_i$ for some coefficients $\{a_i\}_{i=1}^N$, where the Gaussian kernel is introduced in (14). The coefficients are chosen to minimize the empirical risk. While we omit the details, we note that this optimization problem can be formulated as quadratic programming (with regularization as stated in Bertsimas and Koduri 2022) with $3N$ variables and $4N$ linear constraints.

Next we explain the implementation of the four benchmarks along with DONN. The implementation of QR is relatively straightforward, as it does not have hyperparameters to be tuned. For KO, we select the best bandwidth w by randomly splitting the training data into 80% and 20%. The latter is used as the validation set to select w that has the best performance in it. We conduct a grid search for w from 0.1 to 2.1 with increments of 0.1 and record the average optimal cost output by KO relative to the true optimal cost in the validation set. We chose the optimal bandwidth and rerun KO in the whole training data. It is then examined in a separate test set. For EENN, we use the specialized loss function as explained above, and use a similar architecture as that of DONN, to isolate the specific effect of the choice of methodology. We use a feedforward fully connected network with $L = 3$ layers and $H = 7$ nodes in each layer. We use a Swish activation function.

As generally advised (Goodfellow et al. 2016, page 428, and in concordance with Oroojlooyjadid et al. (2020)) we consider learning rates from $\{10^{-3}, 10^{-4}, 10^{-5}\}$, and for each learning rate, we consider number of epochs starting from 0 to $4,000$, with a grid sizes of 100, and a batch size of 200. This specific batch size lead to best performance for EENN. For RKHS, similar to KO, we select the bandwidth w and the regularization hyperparameter using the validation-set approach. The bandwidth is searched on a grid from 0.1 to 20.1 with increments of 2 (on training data sets of size up to 7500 and fixed for subsequent data set sizes, as the optimal bandwidth stabilizes) and the regularization is selected from $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. To solve the optimization problem, we use Gurobi version 9.0.1. Note that RKHS is the most computationally intense among the approaches, and thus we are not able to test a finer grid for the hyperparameters.

For DONN, we use a feedforward, fully connected network with $L = 3$ layers and $H = 7$ nodes in each layer and a Swish activation function. To tune the hyperparameters, we use the validation-set approach to perform a grid search for the learning rate and number of epochs. We consider a learning rate of 10^{-7}, and the number of epochs starting from 0 to $1,000$, with a grid size of 100, and a batch size of 50. We use the Stochastic Gradient Descent algorithm with a Nesterov Momentum of 0.9 (see Sutskever et al. 2013 page 4 and Goodfellow et al. 2016 page 294) for training the DNN. The Stochastic Gradient Descent algorithm has been known to have superior generalization ability compared to other adaptive optimization methods (e.g., Adam, Adagrad or RMSprop) (Wilson et al. 2017, Keskar and Socher 2017). To calculate the optimal order quantity of the fitted DNN, we use the results in Propositions 1 and 2 and the scipy.optimize package in Python (it allows us to provide the closed-form gradient and Hessian matrix), with 100 equidistant starting points, and choose the one that results in the lowest predicted cost by the DNN after convergence.

To test the scaling of the approaches in the sample size, we consider $N \in \{2500, 10000, \ldots, 25000\}$, translating to 25 to 250 days of collected training data. For each N, we generate 25 pairs of train and test data sets where for each pair, the train set is of size N and the test set is of size 300. Figure 3a shows the average cost relative to the true optimal cost of the four approaches in the test sets. The x-axis indicates the number of days of collected training data. The y-axis indicates the average cost from each methodology, relative to the true optimal cost calculated from solving (3) on the test data set. We plot the error bar as the standard error for the 25 test sets.

We list a number of observations from Figure 3a. Due to the high-dimensional nature of the problem (we have 23 covariates as we one-hot encode store and product identifiers), KO suffers from the curse of dimensionality. We also note that both DNN-based approaches, EENN and DONN, are relatively data-hungry. It takes at least 50 days of collected training data for them to outperform other methodologies. Overall, DONN and EENN have relatively similar performances with DONN having a marginally better performance after 75 days of data has been collected,

(a) Performance of the tested methods. (b) Effect of humidity on predicted cost.

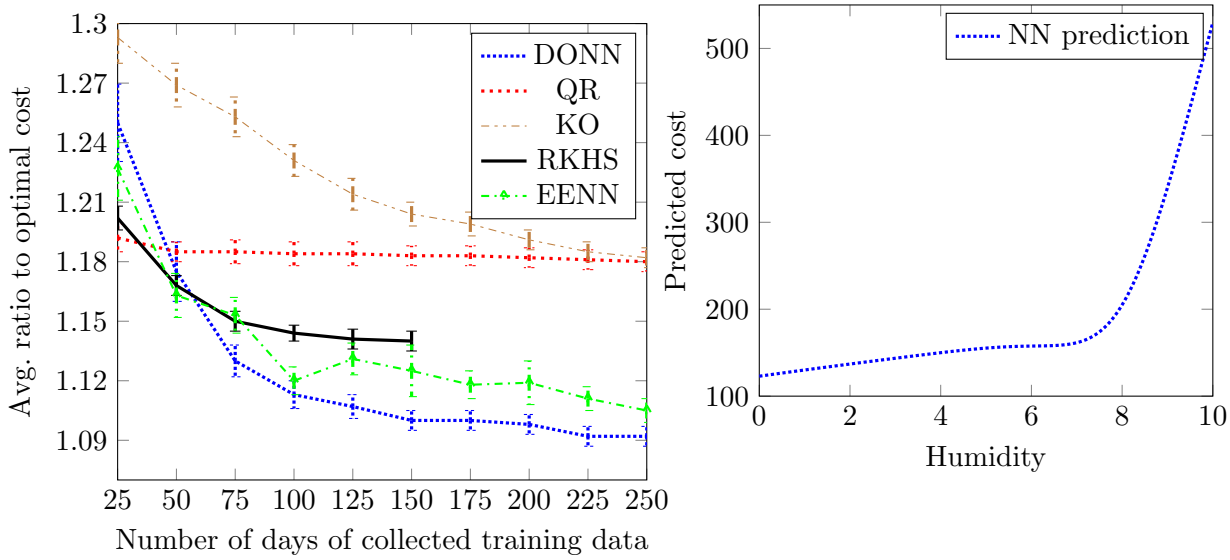


Figure 3 **Illustration of the performance of DONN in the newsvendor problem.**

eventually performing within 9% of the optimal cost. This is particularly interesting, as EENN is a highly specialized, integrated CSO approach that does not apply to Examples 2 and 3 where the decisions affect the uncertain parameters (see Remark 1 for further explanation). We suggest this result further accentuates the findings of Hu et al. (2022) on the excellent performance of two-stage predict-then-optimize approaches.

We note that a particularly interesting aspect of the DONN approach is how well the trained DNNs capture the nuances of the objective function. Figure 3b portrays the predicted effect of humidity on the newsvendor cost (after training with 250 days of data), when everything else is kept fixed. It accurately reflects the actual effect of humidity on demand, $20(h-8)^+$.

We remark on the computational time of the five approaches when $N = 15,000$. All of the algorithms were executed on a node in a large server equipped with 40 Intel "Skylake" cores at 2.4 GHz and 202 GB of RAM. The QR method takes less than 1 second to train, and outputs an order quantity for a test data point almost instantly (within less than 0.01 of a second). Similarly, for a given bandwidth, the KO methodology takes less than 3 seconds to output an order quantity for a test data point. The DONN approach, for a given learning rate and with a batch size of 50, takes less than 30 seconds to train for 100 epochs, and after training is completed, outputs an order quantity for a test data point within 5 seconds. Similarly, for a given learning rate and for a batch size of 200, EENN takes around 14 seconds to train for 100 epochs and instantly outputs an order quantity for a test data point. Meanwhile, for a given bandwidth and regularizing parameter, the RKHS methodology takes over 7 hours to train and outputs an order quantity for a test data point almost instantly. We note that even when the size of the training data grows to 25000, for a

given learning rate the DONN method takes less than 1 minutes to train for 100 epochs, and after training is completed, outputs an order quantity for a test data point within 5 seconds.

Finally, we remark on the satisfaction of the assumptions of Proposition 3 for the newsvendor problem. Assumptions 1 and 2 mainly pertain to how the data is generated and the quality of the fitted DNNs. In light of Remark 2, we can focus on the continuous covariates, and it is easy to check that (12) is Lipchitz continuous, as long as daily demand is finite, satisfying Assumption 3. Assumptions 4 and 7 are satisfied due to the structure of (12), as long as $h, b > 0$. Assumptions 5 and 6 are arguably the most restrictive ones. However, in the newsvendor problem, as long as the cost parameters are known, we can augment the data with any order quantity we like and observe its associated realized cost, making these assumptions naturally satisfied in this case.

6.2. Performance on the Personalized Assortment Pricing Problem with Observable Customer Features

Next we test the DONN methodology on Example 2, the personalized assortment pricing problem. For each customer, the firm observes a covariate vector $\boldsymbol{X}$ (related to the customer and the market conditions) and sets prices $\boldsymbol{z} = (z_1, \ldots, z_m)$ to maximize its conditional expected revenue from the customer who will purchase at most one of the m products. Such problems arise in many settings (Chen et al. 2022b). The conditional expected revenue under price vector $\boldsymbol{z}$ is:

$$\sum_{j=1}^{m} z_j d_j(\boldsymbol{z}|\boldsymbol{X}), \tag{15}$$

where $d_j(z|\boldsymbol{X})$ is the conditional probability of the customer purchasing product j after observing price vector $\boldsymbol{z}$ and $\sum_{j=1}^{m} d_j(\boldsymbol{z}|\boldsymbol{X}) \leq 1$. The optimization problem is:

$$\boldsymbol{z}^* = \arg\max_{\boldsymbol{z}\geq 0} \sum_{j=1}^{m} z_j d_j(\boldsymbol{z}|\boldsymbol{X}). \tag{16}$$

The difficulty in solving (16) is that the decision maker does not know the expression of $d_j(\boldsymbol{z}|\boldsymbol{X})$ in closed form and needs to learn it from past data. Traditionally, this is done by specifying a model for $d_j(\boldsymbol{z}|\boldsymbol{X})$ and estimating the parameters of this model from the data (Chen et al. 2022b,a).

We consider two experimental setups to create a synthetic test dataset. In both setups the covariate vector $\boldsymbol{X}$ represents the age of the selling season, *Age*, and the estimated income of the customer, *Income* (see, e.g., Shen et al. (2020)). In the first setup, the historical data is generated by a Multinomial Logit (MNL) choice model where the probability of customer i choosing product j from a set of m products priced at $(z_1, \ldots, z_m)$ is given by

$$\frac{\exp(\alpha_{ij}(\boldsymbol{X_i}) - \lambda z_j)}{1 + \sum_{k=1}^{m} \exp(\alpha_{ik}(\boldsymbol{X_i}) - \lambda z_k)}, \tag{17}$$

where $\alpha_{ij}(\boldsymbol{X_i}) = U_j + \boldsymbol{X_i'\beta}$ and $U_j \in \mathbb{R}$ and $\boldsymbol{\beta} \in \mathbb{R}^{dx}$. Parameters $\{\alpha_{ik}\}_{k=1}^m(\boldsymbol{X_i})$ and λ represent the average attractiveness of the products and the price sensitivity, respectively. We let:

$$\alpha_{ij}(\boldsymbol{X}_i) = U_j - w_{1j}Age_i + w_{2j}Income_i$$

where U_j is generated randomly from $N(2, 0.5)$, w_1 is generated uniformly from $[0, 1]$ and w_2 is generated uniformly from $[0, 1/70]$. *Age* is uniformly generated from $[0, 1]$ and *Income* is randomly distributed with $N(70, 10)$, in accordance with the reported median household income in the US (United States Census Bureau 2021). We let $\lambda = 0.5$. The historical price seen by customer i for each product j is generated randomly from $N(6, 2)$, truncated at 0. This distribution of historical prices reflects that decision makers may know the optimal price for the average customer while engaging in regular price markup and markdowns. For example, Amazon changes prices every 10 minutes for an average listed product (Mehta et al. 2018). After observing $\boldsymbol{z}_i$, each customer i decides on purchasing at most one of the products, based on probabilities calculated from (17).

In the second setup, historical data is generated based on a linear demand model. The choice probability of product j is assumed to be

$$\alpha_{ij}(\boldsymbol{X_i}) + \sum_{k=1}^{m} \lambda_{jk} z_{ik}, \tag{18}$$

where $\alpha_{ij}(\boldsymbol{X}_i) = U_j + \boldsymbol{X}_i'\boldsymbol{\beta}$ and $U_j \in \mathbb{R}$ and $\boldsymbol{\beta} \in \mathbb{R}^{dx}$. In the setup, $\alpha_{ij}(\boldsymbol{X_i}) = U_j - w_{1j}Age_i + w_{2j}Income_i$. The no-purchase probability is one minus the sum of the purchase probabilities of all the products. We let $\lambda_{jj} = -0.01$ for all j, while λ_{jk} is uniformly sampled from $[0.0002, 0.00024]$ for $j \neq k$. U_j is uniform generated from $[0.13, 0.15]$. w_{1j} are uniformly and independently generated from $[0, 0.05]$. w_{2j} are uniformly and independently generated from $[0, 0.05/70]$. As before, *Age* is uniformly distributed on $[0, 1]$ and *Income* is randomly distributed with $N(70, 10)$. The historical price seen by customer i for each product j is generated randomly from $N(8, 2.5)$, however, the price is truncated such that $d_{ij}(\boldsymbol{z}) \geq 0$ for all i and j, while $\sum_j d_{ij}(\boldsymbol{z}) \leq 1$.

In each dataset, for each customer i, the feature vector $\boldsymbol{X}_i$, the price vector $\boldsymbol{z}_i$, the choice of the customer and the revenue from the customer is recorded. For each experimental setup we simulate 25 independent training sets of size N and test sets of size 250.

We consider three different approaches to solving (16) and numerically test their performance on the synthetic dataset. First we assume $d_j(\boldsymbol{z}|\boldsymbol{x})$ is determined based on the MNL model specified in (17) and estimate the parameters of the model from the gathered data. We use the BIOGEME package developed by Bierlaire (2003) to estimate the parameters of the MNL model with the historical data, and then calculate the optimal price based on the fitted model.

Second, we assume $d_j(\boldsymbol{z}|\boldsymbol{x})$ is determined based on the linear demand model specified in (18) and estimate the parameters of the model from the gathered data, and then calculate the optimal price based on the fitted model.

Third, in the DONN approach, we estimate the conditional expected revenue using a DNN where the input for each historical customer i is $\boldsymbol{X}_i$ and $\boldsymbol{z}_i$ while the target variable is the revenue from customer i. The revenue from customer i is the price paid by the customer for her purchased product, or 0 if no purchase is made. For covariate vector $\boldsymbol{x}$ and price $\boldsymbol{z}$, the output of the DNN is $f_{\hat{\theta}}(\boldsymbol{X}, \boldsymbol{z})$. Then, for observed covariates $\boldsymbol{X}_0$ the optimal DONN price can be obtained by solving the following optimization problem:

$$z^*_{DONN} = \max_{\boldsymbol{z} \in [\underline{\boldsymbol{z}}, \bar{\boldsymbol{z}}]} \left\{ f_{\hat{\theta}}(\boldsymbol{X}_0, \boldsymbol{z}) \right\}, \tag{19}$$

where $\underline{\boldsymbol{z}}$ and $\bar{\boldsymbol{z}}$ are the highest and lowest prices recorded in the data.

The DNN architecture is the same as the one used in Section 6.1. To calculate the optimal prices of the fitted DNN, z^*_{DONN}, we use the scipy.optimize package from Python to solve for Problem (19), with 100 equidistant starting points, and choose the one that results in the highest predicted revenue by the DNN, after convergence.

Figure 4 shows the results where customer demand is given by the MNL choice model, where we fit the observed demand to the MNL model, the linear demand model, and use the DONN approach. We give the average ratio of the revenue generated to the optimal revenue for the MNL model when the true parameters are known. The size of the training transaction data, given on the x axis, grows on a log(10) scale. The MNL estimated revenue converges quickly to the true optimal revenue as is expected for the correctly specified model. We see that for a small dataset of 100 observations, DONN achieves 60% of the optimal revenue, but this increases as the size of the training data grows, approaching the optimal value. The revenue of the linear demand model does not converge to the true optimal revenue due to the underlying model misspecification.

Figure 5 displays the results for the second setup where customer demand is given by the linear demand model. As expected, the linear demand model is able to converge to the true optimal prices and it performs near-optimal with approximately 10000 training data points. The DONN methodology, even with as few as 100 training data points, is able to generate on average more than 50% of the optimal revenue, converging to the true optimal prices as the size of the training data grows. Interestingly, while the MNL estimation methodology is able to outperform the other two methodologies when the size of the training data is less than 1000 data points, due to its inherent misspecification, its performance stagnates and is unable to converge to the true optimal prices as the size of the training data increases.

Figure 4 The performance of DONN, MNL Estimation and Linear Demand Estimation, when MNL is correctly specified

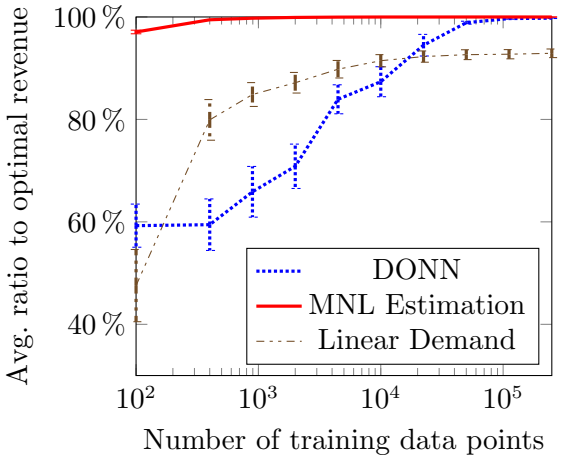
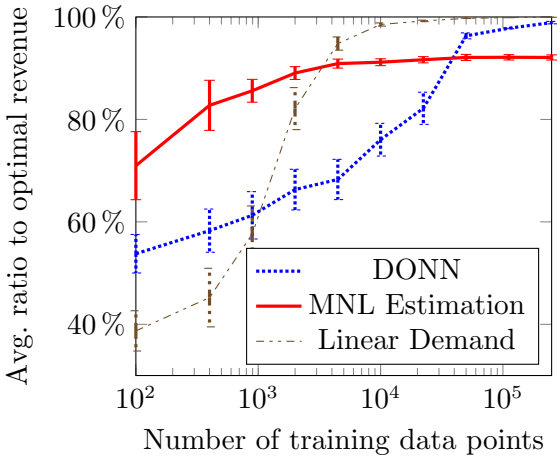


Figure 5 The performance of DONN, MNL Estimation and Linear Demand Estimation, when Linear Demand is correctly specified



In both setups, the DONN approach needs nearly 100,000 training data points to consistently perform near optimal. In the context of online sales, this translates to mere days of data collection. For example, on Prime Day 2022, Amazon recorded 100,000 purchases per minute (Amazon 2023).

We note other approaches may not be easily applicable to the assortment pricing problem we consider. Bertsimas and Kallus (2020) and Biggs et al. (2021) consider machine-learning-based pricing problems. However, both works primarily use discrete prices and only consider the problem of pricing a single product. It seems highly non-trivial to extend them to the setting with multiple substitutable products. Similarly, Simchi-Levi and Xu (2022) consider a discrete action space and extending their results to the continuous action space of assortment prices seems highly non-trivial. Further, their Algorithms 1 and 2 require an optimization problem in step 6 to be easy, which may no longer be true in the case of multiple products, even when considering discrete sets of prices.

Finally, we remark on how the assumptions of Proposition 3 are satisfied in this personalized assortment pricing problem. It is easy to check that (15) is Lipschitz continuous for both demand models (17) and (18), satisfying Assumption 3. Assumptions 4 and 7 are satisfied for both demand models. Assumptions 5 and 6 pertain to the historical price decisions of the firm and the assumed normal distribution of historical prices satisfies them as the size of the data grows. Moreover, these assumptions together are comparable to the positive propensity score assumptions used in causal inference for identifiability (Bertsimas and Kallus 2020, Biggs et al. 2021, Biggs 2022). For example, the overlap assumption of Biggs et al. (2021), Biggs (2022) would assume the probability density function of prices in the data is known and strictly positive over $\mathbb{R}^+$.

6.3. Performance on the Call Center Staffing Problem

Next we assess the performance of DONN on Example 3, the call center staffing problem. The firm observes covariates $\boldsymbol{X}$ (such as the weather conditions) that may help it forecast the arrival rate

of customers the next day. To simplify the comparison with the existing methods in the literature, we assume $\boldsymbol{X}$ only includes the forecast arrival rate. The firm then determines the staffing level z to minimize its conditional expected cost. We assume the cost rate function reflects the staffing costs, the probability of waiting, and the probability of abandonment (Zeltyn and Mandelbaum 2005, Mandelbaum and Zeltyn 2009):

$$C(z|\lambda,\mu) = C_s(z) + \frac{C_w \sum_{l=1}^{A} \mathbb{I}(W_l > 0)}{T} + \frac{C_{AB} \sum_{l=1}^{A} \mathbb{I}(AB_l = 1)}{T}, \tag{20}$$

where λ is the customer arrival rate, μ is the service rate, z is the number of agents working and T is the length of the day, e.g. 8 hours. A is the random variable indicating the number of customers arriving at the call center throughout $[0,T]$. W_l is the random variable indicating the amount of time customer l had to wait before receiving service. AB_l is the random binary variable indicating that customer l arrived at the call center but abandoned. $C_s(z)$ is the cost rate of staffing z agents, C_w is the cost per customer who waits, and C_{AB} is the cost per abandonment. The long-run expected value of C is:

$$\lim_{T\to\infty} \mathbb{E}[C(z|\lambda,\mu)] = C_s(z) + C_w \lambda \mathsf{P}(W > 0) + C_{AB} \lambda \mathsf{P}(AB = 1). \tag{21}$$

There are existing approaches to approximating (21). When the customer inter-arrival times and service times are independent and exponentially distributed and customer patience time is independently distributed, the call center is an $M/M/z+G$ queue, with G being the cumulative distribution of customer patience. Characterizing the performance metrics of such a queue analytically is very complex. However, assuming knowledge of G, under three different operational regimes, based on the offered load λ/μ, Zeltyn and Mandelbaum (2005), Mandelbaum and Zeltyn (2009) develop closed-form approximations for $\mathsf{P}(W > 0)$ and $\mathsf{P}(AB = 1)$. The three operational regimes are Quality Driven (QD), Efficiency Driven (ED), and Quality and Efficiency Driven (QED). For each regime, z is given by the offered load λ/μ and a tuning parameter:

$$\begin{cases} 1 - \frac{z_{QD}}{\lambda/\mu} = \gamma, & \gamma > 0, \\ \frac{z_{ED}}{\lambda/\mu} - 1 = \gamma, & \gamma > 0, \\ \frac{z_{QED} - \lambda/\mu}{\sqrt{\lambda/\mu}} = \beta, & \infty < \beta < \infty. \end{cases}$$

As suggested in Zeltyn and Mandelbaum (2005) "The QD approximations should be applied only for very high-performance systems. (For example, in emergency call centers.)". As such, we do not present the numerical results for this regime. Under the QED regime, based on the results from Zeltyn and Mandelbaum (2005), for a given β we have $\mathsf{P}(W > 0) = [1 + \sqrt{\frac{g_0}{\mu}} \cdot \frac{h(\beta)}{h(-\beta)}]^{-1}$, where $h(\cdot)$ is the hazard rate of the standard normal distribution, $g_0 = g(0)$, where $g(\cdot)$ is the

probability distribution function of customer patience, and $\hat{\beta} = \beta\sqrt{\frac{\mu}{g_0}}$. Further, we have $\mathsf{P}(AB = 1) = [1 + \sqrt{\frac{g_0}{\mu}} \cdot \frac{h(\hat{\beta})}{h(-\beta)}]^{-1} \cdot \frac{1}{\sqrt{z_{QED}}} \sqrt{\frac{g_0}{\mu}} \cdot [h(\hat{\beta}) - \hat{\beta}]$. Therefore,

$$z^*_{QED} = \underset{z_{QED}\geq 0, -\infty<\beta<\infty}{\arg\min} \left(C_s(z_{QED}) + C_W\lambda[1 + \sqrt{\frac{g_0}{\mu}} \cdot \frac{h(\hat{\beta})}{h(-\beta)}]^{-1} + C_{AB}\lambda[1 + \sqrt{\frac{g_0}{\mu}} \cdot \frac{h(\hat{\beta})}{h(-\beta)}]^{-1} \cdot \frac{1}{\sqrt{z_{QED}}} \sqrt{\frac{g_0}{\mu}} \cdot [h(\hat{\beta}) - \hat{\beta}] \right) \tag{22}$$

In the ED regime, for a given $\gamma > 0$, we have: $\mathsf{P}(W > 0) = 1 - [\frac{1}{\gamma} \cdot \sqrt{\frac{g(x^*)}{2\pi\lambda}} \cdot e^{-\lambda k(\gamma)}]$, where x^* is the unique solution to $G(x) = \gamma$ and $k(\gamma) = x^* \cdot (1 - \frac{z_{ED}(\gamma)\mu}{\lambda}) - \int_0^{x^*} G(u)du$. Moreover, $\mathsf{P}(AB = 1) = \gamma$. Therefore,

$$z^*_{ED} = \underset{z_{ED}\geq 0, \gamma>0}{\arg\min} \left(C_s(z_{ED}) + C_W\lambda\big(1 - [\frac{1}{\gamma} \cdot \sqrt{\frac{g(x^*)}{2\pi\lambda}} \cdot e^{-\lambda k(\gamma)}]\big) + C_{AB}\lambda\gamma \right) \tag{23}$$

As pointed out in Zeltyn and Mandelbaum (2005), when the utilization, $\lambda/(z\mu)$ is close to 1, the QED regime offers a near exact approximation, while for larger utilization, the ED regime may be more accurate. The difficulty for a manager would be to know ahead of time which of the two regimes will be more appropriate, depending on the expected offered load. Note that even when applied with the correct regime, both ED and QED are slightly mis-specified, as $\mathbb{E}[C(z|\lambda, \mu)]$ may differ from $\lim_{T\to\infty} \mathbb{E}[C(z|\lambda, \mu)]$ for finite T.

In contrast to ED and QED, the DONN approach does not rely on any specific assumptions regarding the utilization, the distribution of customer arrivals, service times, and customer patience distribution. In the DONN approach, we estimate $\mathbb{E}[C(z|\lambda, \mu)]$ using a DNN where the input for each historical day i is λ_i and $\boldsymbol{z}_i$ while the target variable is $C(z_i|\lambda_i, \mu_i)$. We use the same architecture and hyperparameter tuning as in Examples 1 and 2. For λ and staffing level z, the output of the DNN is $f_{\hat{\theta}}(\lambda, z)$. Then, for the new day's arrival rate λ_0 the optimal DONN staffing level can be obtained by solving the following optimization problem:

$$z^*_{DONN} = \underset{\boldsymbol{z}\in\{\underline{\boldsymbol{z}}, \bar{\boldsymbol{z}}\}}{\arg\min} f_{\hat{\theta}}(\lambda_0, \boldsymbol{z}), \tag{24}$$

where $\underline{\boldsymbol{z}}$ and $\bar{\boldsymbol{z}}$ are the highest and lowest staffing levels recorded in the data.

Given the low dimensional nature of the experiments in this chapter, one could posit that $\mathbb{E}[C(z|\lambda, \mu)]$ may be an easy function to estimate, especially since μ will be fixed in the experiments. To evaluate this, we consider an alternative methodology to DONN, where we fit a polynomial regression (PR) to the historically collected data to estimate $\mathbb{E}[C(z|\lambda, \mu)]$ as a function of λ and z. We fit polynomials of various degrees (1 to 6) using a quadratic loss function. Let f_{PR} be the best-performing polynomial regression and define

$$z^*_{PR} = \underset{z\geq 0}{\arg\min} f_{PR}(\lambda_0, z). \tag{25}$$

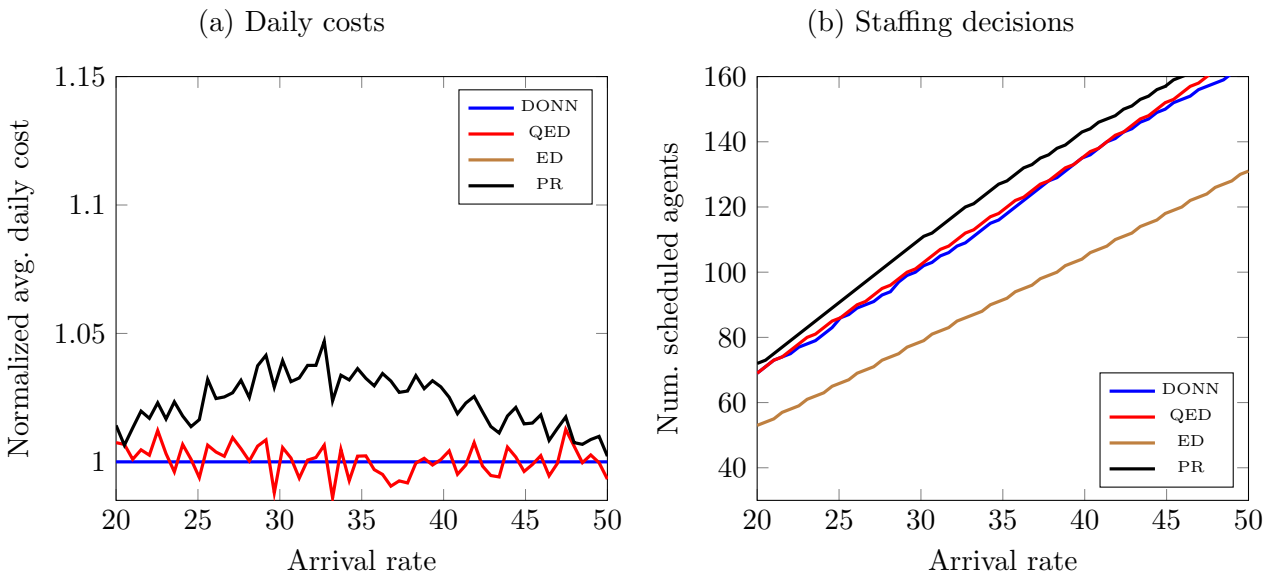


Figure 6 **M/M/z+G queue with linear staffing cost.**

We conduct three experiments. Throughout we assume the call center operates 8 hours a day, each customer experiencing positive wait time costs \$0.3 ($C_W = 0.3$) and each abandoning customer costs \$1 ($C_{AB} = 1$). We fix the mean service rate $\mu = 1/3$ per minute. As in Mandelbaum and Zeltyn (2009), we assume customers have exponentially distributed patience with either mean 1 or mean 5, each occurring with probability $p = 0.5$.

In the first experiment, as in Section 6.1. of Mandelbaum and Zeltyn (2009), we assume a linear staffing cost with each scheduled agent costs \$100 per day ($C_s(z) = (100/480)z$), Poisson arrivals with the mean rate each day between 20 and 50 arrivals per minute and exponentially distributed service times with mean 3 minutes. The synthetic historical data is generated as follows: we simulate 1000 days ($N = 1000$) based on a first-come/first-serve (FCFS) $M/M/z+G$ queue and for everyday i, the customer arrival rate per minute is chosen from a uniform distribution $\lambda_i \sim U[20, 50]$ and the staffing level is randomly distributed with $z \sim \max\{\lfloor N(\frac{\lambda}{\mu}, \frac{\lambda}{3\mu})\rfloor, 1\}$.

To assess the out-of-sample performance of the four methodologies we consider arrival rates from 20 to 50 in increments of 0.5 and simulate 10 days for each arrival rate and the prescribed staffing level of each of the methodologies. Figure 6a presents the normalized average daily cost for the M/M/z+G experiment with linear staffing cost where the result of each method is divided by DONN's average daily cost. Figure 6b presents the associated staffing for each of the solution approaches. We observe the cost of DONN is statistically the same as QED (mean difference 0.05% with standard error 0.05%), while PR is consistently higher (mean 2.3% higher with standard error 0.1%). The cost of ED exceeds 15% throughout and is not shown. For the range of arrival rates, $\lambda/(z^*_{QED}\mu)$ is near 1 and hence QED is the appropriate regime and z^*_{QED} must be nearly optimal

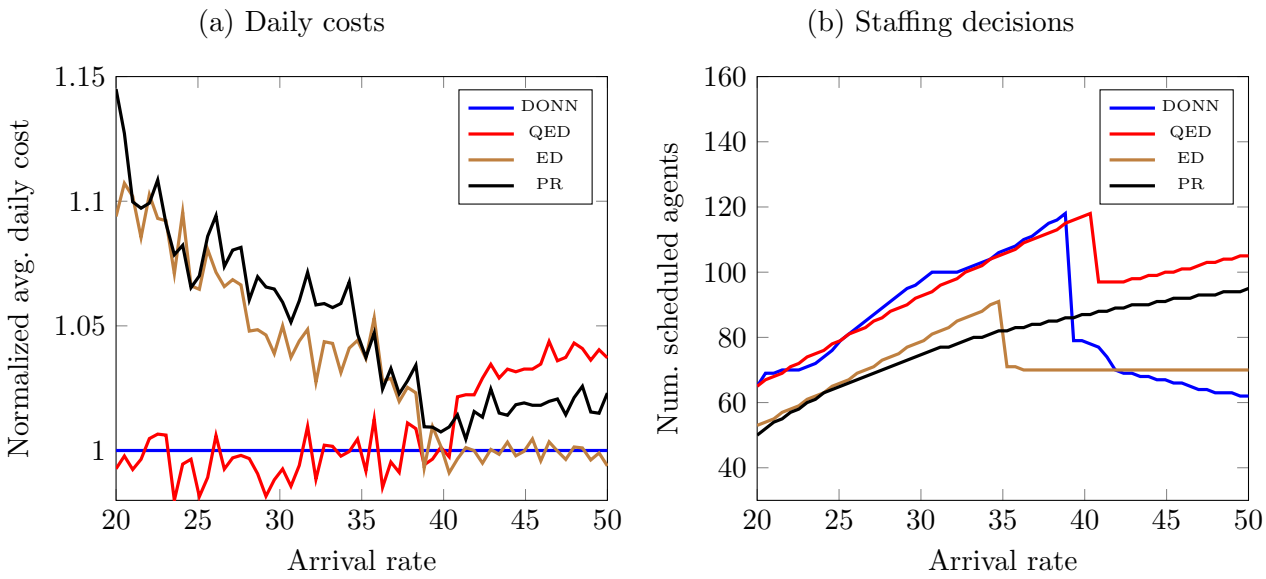


Figure 7 **M/M/z+G queue with convex staffing cost.**

(Zeltyn and Mandelbaum 2005). Figure 6b suggests that the staffing decisions made by DONN and QED are nearly identical while PR consistently over-staffs and ED greatly under-staffs.

Next we consider the case where the cost rate of staffing is a convex increasing function $C_s(z) = (100z + [(z-40)^+]^2)/480$. When the arrival rate is high, it would be more economical to under-staff and incur a high rate of customer waiting and abandoning. As the utilization increases, the ED staffing level may be more accurate than QED. Balancing between the staffing and customer waiting and abandonment costs may be difficult for managers. Figures 7a and Figure 7b presents the results in this case. We observe that for arrival rates less than 40, the DONN and QED solutions give the lowest cost, while for greater arrival rates, the DONN and ED provide the lowest dost. For lower arrival rates, a higher staffing level performs better and for higher arrival rates, a lower staffing level is better. The DONN approach switches between the two, performing within 0.3% of the cost of the best of ED and QED, with a standard error of less than 0.1%. In contrast, the polynomial regression approach consistently performs similarly to the worst of ED and QED suggesting the need for a nuanced data-driven approach to this complex staffing problem.

In the third experiment we consider non-Markovian arrival and service, i.e., a $G/GI/z+G$ queue with linear staffing costs. We assume customer interarrival times are uniformly distributed on $[1/\lambda - 0.01, 1/\lambda + 0.01]$ and the service times are uniformly distributed on $[1, 5]$. The customer patience distribution and the daily staffing cost are the same as before. Despite work in approximating asymptotic queue lengths for $G/GI/z+G$ (Whitt 2004, Mandelbaum and Momčilović 2012, Lu et al. 2016), closed-form approximations analogous to (22) and (23) are not available.

Figure 8a presents the results for this case. With 1000 training data points the DONN approach consistently outperforms both QED and ER by 2.5% (standard error 0.1%) and 8.2% (standard

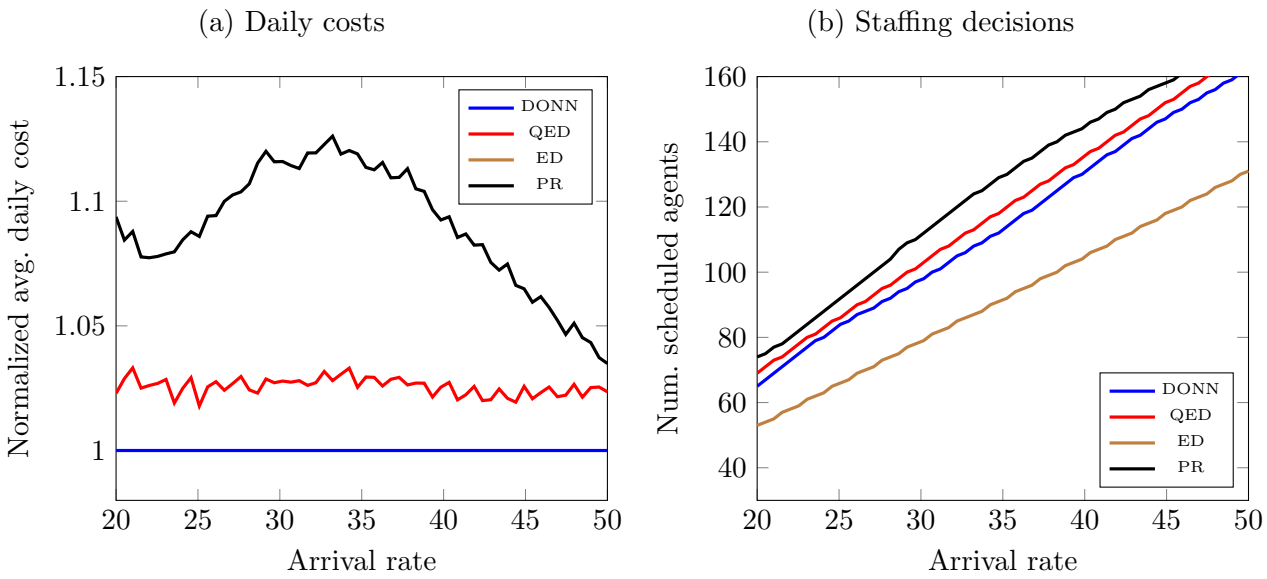


Figure 8 **G/GI/z+G queue with linear staffing cost.**

error 0.3%), respectively. The cost for ED exceeds 15%. Figure 8b suggests that QED and PR consistently over-staff while ED significantly under-staffs. In appendix EC.4 we show that PR cannot approximate well the non-convex cost structure while the relatively small neural network in the DONN offers a near-exact fit of the actual cost function.

Regarding the assumptions of Proposition 3, the assumed normal distribution of historical staffing levels around the offered load imply assumptions 5 and 6 are satisfied. Assumptions 3, 4 and 7 are not verifiable as (20) does not have a closed-form expression.

To summarize the takeaways from the numerical experiments we note:

- The DONN approach performs very well with small DNNs and moderate-size training data.
- The DONN approach computationally scales well with the size of the training data.
- These observations are robust across different problems and specifications.

7. Practical Considerations and Concluding Remarks

A few comments on applying our approach to real-world problems are due. Oroojlooyjadid et al. (2020) mentions when applying DNNs to large, real world problems, practitioners may need to carefully tune the hyper-parameters (such as the number of layers and nodes in each layer as well as the activation function) of the DNN to find the most suitable one for their application. However, our paper alleviates this concern to some extent by showing that even arguably toy sized DNNs trained with mostly open source and widely available packages (primarily PyTorch) can tackle very hard problems and lead to very strong performance in a variety of scenarios.

Another important criticism usually directed at using DNNs is their apparent lack of interpretability (Lipton 2018, Oroojlooyjadid et al. 2020). However, in this work we help ease managerial

concerns regarding using DNNs in two ways. First, by designing a framework that uses DNNs to learn the true objective function, and by providing performance guarantees for decisions in our framework, a decision maker may be able to trust the output, especially if the trained DNN shows strong generalization properties in the training data. Moreover, by providing the gradient of the trained DNN with respect to the decision variables in closed form, our framework demonstrates the direction of improvement predicted by the DNN for every reasonable decision value (e.g., increase or decrease in predicted revenue for a unit increase in the price of a product in the assortment) allowing for the decisions to be interpretable. This is especially relevant as DNNs are capable of approximating any arbitrary function and its derivative (Hornik et al. 1990, 1994).

We also note that the degree to which the solutions from our framework can be trusted depends on the Lipschitz constant of the trained DNN as well as its generalization bound. While there are several ways to gauge the generalization abilities of a DNN (including the performance of holdout data samples), assessing the Lipschitz constant exactly is NP-hard in general (Virmaux and Scaman 2018). However, there is a growing body of literature on estimating upper bounds on the Lipschitz constant of a trained DNN (Virmaux and Scaman 2018, Latorre et al. 2020, Herrera et al. 2020). Such work can be used in our framework to assess its performance guarantees.

References

Agarap AF (2018) Deep learning using rectified linear units (ReLU). *Working Paper* ArXiv preprint arXiv:1803.08375.

Ahn K, Zhang J, Sra S (2022) Understanding the unstable convergence of gradient descent. *International Conference on Machine Learning*, 247–257 (PMLR).

Amazon (2023) Amazon selling stats. URL `https://sell.amazon.com/blog/amazon-stats`, [Accessed October 16, 2023].

Anderson R, Huchette J, Ma W, Tjandraatmadja C, Vielma JP (2020) Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming* 183(1):3–39.

Aouad A, Desir A (2022) Representing random utility choice models with neural networks. *Working Paper* ArXiv preprint arXiv:2207.12877.

Aziznejad S, Gupta H, Campos J, Unser M (2020) Deep neural networks with trainable activations and controlled Lipschitz constant. *IEEE Transactions on Signal Processing* 68:4688–4699.

Babier A, Chan TC, Diamant A, Mahmood R (2018) Learning to optimize contextually constrained problems for real-time decision-generation. *Working Paper* ArXiv preprint arXiv:1805.09293.

Ban GY, Rudin C (2019) The big data newsvendor: Practical insights from machine learning. *Oper. Res.* 67(1):90–108.

Bartlett PL, Long PM, Lugosi G, Tsigler A (2020) Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences* 117(48):30063–30070.

Bartlett PL, Montanari A, Rakhlin A (2021) Deep learning: a statistical viewpoint. *Acta numerica* 30:87–201.

Bertsekas DP (1999) *Nonlinear programming* (Cambridge, MA, USA: MIT Press,).

Bertsimas D, Kallus N (2020) From predictive to prescriptive analytics. *Management Sci.* 66(3):1025–1044.

Bertsimas D, Koduri N (2022) Data-driven optimization: A reproducing kernel Hilbert space approach. *Oper. Res.* 70(1):454–471.

Bertsimas D, Van Parys B (2021) Bootstrap robust prescriptive analytics. *Mathematical Programming* 1–40.

Bianchi P, Hachem W, Schechtman S (2022) Convergence of constant step stochastic gradient descent for non-smooth non-convex functions. *Set-Valued and Variational Analysis* 1–31.

Bierlaire M (2003) Biogeme: A free package for the estimation of discrete choice models. *Proceedings of the 3rd Swiss Transportation Research Conference, Ascona, Switzerland.*, www.strc.ch.

Biggs M (2022) Convex surrogate loss functions for contextual pricing with transaction data. *arXiv preprint arXiv:2202.10944* .

Biggs M, Gao R, Sun W (2021) Loss functions for discrete contextual pricing with observational data. *arXiv preprint arXiv:2111.09933* .

Bolte J, Pauwels E (2021) Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Mathematical Programming* 188(1):19–51.

Botoeva E, Kouvaros P, Kronqvist J, Lomuscio A, Misener R (2020) Efficient verification of relu-based neural networks via dependency analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3291–3299.

Cai Z, Wang H, Talluri K, Li X (2022) Deep learning for choice modeling. *Working Paper* ArXiv preprint arXiv:2208.09325.

Chen N, Cire AA, Hu M, Lagzi S (2022a) Model-free assortment pricing with transaction data. *Management Sci.* Forthcoming.

Chen X, Owen Z, Pixton C, Simchi-Levi D (2022b) A statistical learning approach to personalization in revenue management. *Management Sci.* 68(3):1923–1937.

Collier M, Llorens HU (2018) Deep contextual multi-armed bandits. *arXiv preprint arXiv:1807.09809* .

Cybenko G (1989) Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* 2(4):303–314.

Dutta S, Jha S, Sankaranarayanan S, Tiwari A (2018) Output range analysis for deep feedforward neural networks. *NASA Formal Methods Symposium*, 121–138 (Springer).

Elmachtoub AN, Grigas P (2022) Smart "predict, then optimize". *Management Sci.* 68(1):9–26.

Fan J, Ma C, Zhong Y (2021) A selective overview of deep learning. *Statistical science* 36(2):264.

Farrell MH, Liang T, Misra S (2021) Deep neural networks for estimation and inference. *Econometrica* 89(1):181–213.

Finlay C, Calder J, Abbasi B, Oberman A (2018) Lipschitz regularized deep neural networks generalize and are adversarially robust. *arXiv preprint arXiv:1808.09540* .

Fischetti M, Jo J (2018) Deep neural networks and mixed integer linear optimization. *Constraints* 23(3):296–309.

Gabel S, Timoshenko A (2022) Product choice with large assortments: A scalable deep-learning model. *Management Sci.* 68(3):1808–1827.

Gijsbrechts J, Boute RN, Van Mieghem JA, Zhang DJ (2022) Can deep reinforcement learning improve inventory management? performance on lost sales, dual-sourcing, and multi-echelon problems. *Manufacturing Service Oper. Management* 24(3):1349—-1368.

Goodfellow I, Bengio Y, Courville A (2016) *Deep Learning* (MIT Press), http://www.deeplearningbook.org.

Gouk H, Frank E, Pfahringer B, Cree MJ (2021) Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning* 110(2):393–416.

Grimstad B, Andersson H (2019) ReLU networks as surrogate models in mixed-integer linear programs. *Computers & Chemical Engineering* 131:106580.

Hardt M, Recht B, Singer Y (2016) Train faster, generalize better: Stability of stochastic gradient descent. *International conference on machine learning*, 1225–1234 (PMLR).

Herrera C, Krach F, Teichmann J (2020) Estimating full Lipschitz constants of deep neural networks. *Working Paper* ArXiv preprint arXiv:2004.13135.

Hornik K, Stinchcombe M, White H (1990) Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks* 3(5):551–560.

Hornik K, Stinchcombe M, White H, Auer P (1994) Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives. *Neural computation* 6(6):1262–1275.

Hu Y, Kallus N, Mao X (2022) Fast rates for contextual linear optimization. *Management Science* 68(6):4236–4245.

Huber J, Müller S, Fleischmann M, Stuckenschmidt H (2019) A data-driven newsvendor problem: From data to decision. *European Journal of Operational Research* 278(3):904–915.

Jiang Y, Neyshabur B, Mobahi H, Krishnan D, Bengio S (2019) Fantastic generalization measures and where to find them. *Working Paper* ArXiv preprint arXiv:1912.02178.

Kallus N, Mao X (2022) Stochastic optimization forests. *Management Sci.* Forthcoming.

Katz G, Barrett C, Dill DL, Julian K, Kochenderfer MJ (2017) Reluplex: An efficient SMT solver for verifying deep neural networks. *International conference on computer aided verification*, 97–117 (Springer).

Keskar NS, Mudigere D, Nocedal J, Smelyanskiy M, Tang PTP (2016) On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836* .

Keskar NS, Socher R (2017) Improving generalization performance by switching from Adam to SGD. *Working Paper* ArXiv preprint arXiv:1712.07628.

Latorre F, Rolland P, Cevher V (2020) Lipschitz constant estimation of neural networks via sparse polynomial optimization. *Working Paper* ArXiv preprint arXiv:2004.08688.

Leshno M, Lin VY, Pinkus A, Schocken S (1993) Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks* 6(6):861–867.

Li H, Xu Z, Taylor G, Studer C, Goldstein T (2018) Visualizing the loss landscape of neural nets. *Advances in neural information processing systems* 31.

Li Y (2017) Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274* .

Lipton ZC (2018) The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue* 16(3):31–57.

Liu M, Qi M, Shen ZJM (2021) End-to-end deep learning for inventory management with fixed ordering cost and its theoretical analysis. *Available at SSRN 3888897* .

Lu H, Pang G, Zhou Y (2016) G/gi/n (+ gi) g/gi/n (+ gi) queues with service interruptions in the halfin–whitt regime. *Mathematical Methods of Operations Research* 83:127–160.

Mandelbaum A, Momčilović P (2012) Queues with many servers and impatient customers. *Mathematics of Operations Research* 37(1):41–65.

Mandelbaum A, Zeltyn S (2009) Staffing many-server queues with impatient customers: Constraint satisfaction in call centers. *Operations research* 57(5):1189–1205.

Marković D, Stojić H, Schwöbel S, Kiebel SJ (2021) An empirical evaluation of active inference in multi-armed bandits. *Neural Networks* 144:229–246.

Mehta N, Parth D, Agashe A (2018) Amazon changes prices on its products about every 10 minutes — here's how and why they do it. URL https://www.businessinsider.com/amazon-price-changes-2018-8, [Insider, Accessed August 1, 2022].

Mousavi SS, Schukat M, Howley E (2018) Deep reinforcement learning: an overview. *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 2*, 426–440 (Springer).

Neghab DP, Khayyati S, Karaesmen F (2022) An integrated data-driven method using deep learning for a newsvendor problem with unobservable features. *European Journal of Operational Research* 302(2):482–496.

Nwankpa C, Ijomah W, Gachagan A, Marshall S (2018) Activation functions: Comparison of trends in practice and research for deep learning. *Working Paper* ArXiv preprint arXiv:1811.03378.

Ohn I, Kim Y (2019) Smooth function approximation by deep neural networks with general activation functions. *Entropy* 21(7):627.

Oroojlooyjadid A, Nazari M, Snyder LV, Takáč M (2022) A deep q-network for the beer game: Deep reinforcement learning for inventory optimization. *Manufacturing Service Oper. Management* 24(1):285–304.

Oroojlooyjadid A, Snyder LV, Takáč M (2020) Applying deep learning to the newsvendor problem. *IISE Transactions* 52(4):444–463.

Pauli P, Koch A, Berberich J, Kohler P, Allgöwer F (2021) Training robust neural networks using Lipschitz bounds. *IEEE Control Systems Letters* 6:121–126.

Perakis G, Tsiourvas A (2022) Optimizing objective functions from ReLU neural networks in revenue management applications. *Working Paper* Presented at the RMP Conference 2022 (Virtual Format).

Pinkus A (1999) Approximation theory of the mlp model in neural networks. *Acta numerica* 8:143–195.

Qi M, Shi Y, Qi Y, Ma C, Yuan R, Wu D, Shen ZJM (2020) A practical end-to-end inventory management model with deep learning. *Available at SSRN 3737780* .

Ramachandran P, Zoph B, Le QV (2017) Searching for activation functions. *Working Paper* ArXiv preprint arXiv:1710.05941.

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *nature* 323(6088):533–536.

Sarraf A (2020) A tight upper bound on the generalization error of feedforward neural networks. *Neural Networks* 127:1–6.

Schmidt-Hieber J (2020) Nonparametric regression using deep neural networks with ReLU activation function. *The Annals of Statistics* 48(4):1875–1897.

Seubert F, Stein N, Taigel F, Winkelmann A (2020) Making the newsvendor smart–order quantity optimization with anns for a bakery chain AMCIS 2020 Proceedings.

Shen M, Tang CS, Wu D, Yuan R, Zhou W (2020) Jd. com: Transaction-level data for the 2020 msom data driven research challenge. *Manufacturing Service Oper. Management* Articles in Advance, pp. 1–9.

Simchi-Levi D, Xu Y (2022) Bypassing the monster: A faster and simpler optimal algorithm for contextual bandits under realizability. *Mathematics of Operations Research* 47(3):1904–1931.

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1):1929–1958.

Strong CA, Wu H, Zeljić A, Julian KD, Katz G, Barrett C, Kochenderfer MJ (2021) Global optimization of objective functions represented by ReLU networks. *Machine Learning* 1–28.

Sutskever I, Martens J, Dahl G, Hinton G (2013) On the importance of initialization and momentum in deep learning. *International conference on machine learning*, 1139–1147 (PMLR).

Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2013) Intriguing properties of neural networks. *Working Paper* ArXiv preprint arXiv:1312.6199.

Tsay C, Kronqvist J, Thebelt A, Misener R (2021) Partition-based formulations for mixed-integer optimization of trained relu neural networks. *Advances in Neural Information Processing Systems* 34:3068–3080.

United States Census Bureau (2021) Income and poverty in the united states: 2020. URL https://www.census.gov/library/publications/2021/demo/p60-273.html, [Accessed August 1, 2022].

Valle-Pérez G, Louis AA (2020) Generalization bounds for deep learning. *Working Paper* ArXiv preprint arXiv:2012.04115.

Virmaux A, Scaman K (2018) Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems* 31.

Wang X, Kadıoğlu S (2021) Modeling uncertainty to improve personalized recommendations via bayesian deep learning. *International Journal of Data Science and Analytics* 1–11.

Whitt W (2004) A diffusion approximation for the g/gi/n/m queue. *Operations Research* 52(6):922–941.

Wilson AC, Roelofs R, Stern M, Srebro N, Recht B (2017) The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems* 30.

Wu G, Say B, Sanner S (2020) Scalable planning with deep neural network learned transition models. *Journal of Artificial Intelligence Research* 68:571–606.

Xu P, Wen Z, Zhao H, Gu Q (2020) Neural contextual bandits with deep representation and shallow exploration. *arXiv preprint arXiv:2012.01780* .

Zeltyn S, Mandelbaum A (2005) Call centers with impatient customers: Many-server asymptotics of the m/m/n+ g queue. *Queueing Systems* 51:361–402.

Zhang C, Bengio S, Hardt M, Recht B, Vinyals O (2021) Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM* 64(3):107–115.

Table EC.1 Table of variables and parameters.

Acronym	Meaning
f_*	True unknown objective function
Y^i	A noisy observation of f_* in epoch i
$\boldsymbol{x}$	Observed covariates
$\boldsymbol{z}$	Decision taken
d_X	Dimension of the observed covariates
d_Z	Dimension of the decision taken
f_θ	A generic DNN function
$f_{\hat{\theta}}$	A realized fitted DNN function
$\sigma(\cdot)$	The activation function
$\sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})$	The point-wise first derivative of the activation function at the h-th node and l-th layer
$\sigma''_{h,l}(\boldsymbol{x},\boldsymbol{z})$	The point-wise second derivative of the activation function at the h-th node and l-th layer
$\boldsymbol{z}^*(\boldsymbol{x_0})$	The true optimal solution given $\boldsymbol{x_0}$
$\hat{\boldsymbol{z}}(\boldsymbol{x_0})$	The DNN based optimal solution given $\boldsymbol{x}_0$
∂	The partial derivative
K_*	The Lipschitz constant of f_*
$K_{\hat{\theta}}$	The Lipschitz constant of $f_{\hat{\theta}}$
K	The Lipschitz constant of $\|f_* - f_{\hat{\theta}}\|$
$\mu_{X,Z}$	The marginal distribution of $(\boldsymbol{X},\boldsymbol{Z})$
δ_N	The generalization bound with N data points.
c_1	The lower bound on the density function of $\mu_{X,Z}$
c_3	Min. radius of a ball away from the optimal solution to have probability measure c_4
c_4	Probability measure of a ball of radius c_3 away from the optimal solution.
c_2	Lower bound on the increase of f_* for solutions far away from optimal solution.
c_5	Quadratic increase rate of f_* for solutions near the optimal solution.
$L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}$	Constant for volume of a $d_X + d_Z$ dimensional ball.

E-Companion to *Using Neural Networks To Guide Data-driven Operational Decisions*

EC.1. Using Neural Networks to Fit the Objective Function

As mentioned in Section 3, we propose to find the best fit objective function $\hat{f}(\boldsymbol{x},\boldsymbol{z})$ from the data $\{(\boldsymbol{X}^i,\boldsymbol{Z}^i,Y^i)\}_{i=1}^N$ within the class of DNN functions $\mathscr{F}_{\text{NN}}$. That is, assuming a quadratic loss function, and a clear specification of $\mathscr{F}_{\text{NN}}$, we seek:

$$\hat{f} = \underset{f\in\mathscr{F}_{\text{NN}}}{\arg\min} \sum_{i=1}^{N} (f(\boldsymbol{X}^i,\boldsymbol{Z}^i) - Y^i)^2. \tag{EC.1}$$

The class $\mathscr{F}_{\text{NN}}$ is the set of functions that can be represented through a particular DNN. The function class $\mathscr{F}_{\text{NN}}$ is determined by the architecture of the network and the form of interactions between the nodes. We consider *feedforward* DNNs consisting of:

- L layers (the depth of the network),

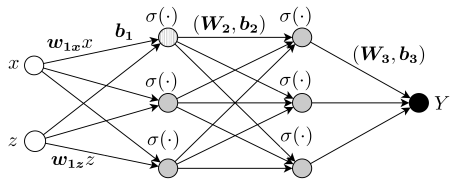


Figure EC.1 The illustration of a fully connected feedforward neural network with $L=2$ layers and $H=3$ nodes in each layer. The activation function is applied to the input at each of the nodes.

- H nodes per layer (the width)[1],
- fully connected layers,
- an activation function $\sigma(\cdot):\mathbb{R}\to\mathbb{R}$.

An example of a network architecture is illustrated in Figure EC.1. The neural network takes as input two scalars (x,z) and outputs a single scalar (y). It has $L=2$ layers and each layer has $H=3$ width (note that we do not count the first and the last layers).

The commonly use activation functions are:

1. **sigmoid**: $\sigma(x)=1/(1+e^{-x})$
2. **Swish**: $\sigma(x)=x/(1+e^{-x})$
3. **ReLU**: $\sigma(x)=\max\{0,x\}$.

Given a network architecture $\mathscr{F}_{\text{NN}}(L,H,\sigma)$, a function withing the class is specified by a parameterization $\theta=(\boldsymbol{W}_{1x},\boldsymbol{W}_{1z},\boldsymbol{b}_1,\boldsymbol{W}_2,\boldsymbol{b}_2,\ldots,\boldsymbol{W}_L,\boldsymbol{b}_L,\boldsymbol{w}_{L+1},b_{L+1})$, where $\boldsymbol{W}_{1x}\in\mathbb{R}^{H\times d_X}$, $\boldsymbol{W}_{1z}\in\mathbb{R}^{H\times d_Z}$, $\boldsymbol{b}_k\in\mathbb{R}^H$ for $k=1,\ldots,L$, $\boldsymbol{W}_k\in\mathbb{R}^{H\times H}$ for $k=2,\ldots,L$, $\boldsymbol{w}_{L+1}\times\mathbb{R}^H$ and $b_{L+1}\in\mathbb{R}$. Given $(\boldsymbol{x},\boldsymbol{z})$ and a parameter θ,

$$f_\theta(\boldsymbol{x},\boldsymbol{z})=\boldsymbol{w}_{\boldsymbol{L+1}}^\top\boldsymbol{\sigma}\Big(\cdots\boldsymbol{\sigma}\Big(\boldsymbol{W_3}\boldsymbol{\sigma}\big(\boldsymbol{W_2}\boldsymbol{\sigma}(\boldsymbol{W_{1x}}\boldsymbol{x}+\boldsymbol{W_{1z}}\boldsymbol{z}+\boldsymbol{b_1})+\boldsymbol{b_2}\big)+\boldsymbol{b_3}\Big)+\cdots\Big)+b_{L+1}.$$

Here, $\boldsymbol{\sigma}(\cdot):\mathbb{R}^H\to\mathbb{R}^H$ represents the vectorized activation function that applies $\sigma(\cdot):\mathbb{R}\to\mathbb{R}$ elementwise.

By this formulation, $\mathscr{F}_{\text{NN}}$ is the set of all functions $f_\theta(\boldsymbol{x},\boldsymbol{z})$. One may use Figure EC.1 to better understand $f_\theta(\boldsymbol{x},\boldsymbol{z})$. Consider the left-most layer (the input layer), the edges leading to the second layer (also referred to as the first hidden layer) and the term $\boldsymbol{w_{1x}}x+\boldsymbol{w_{1z}}z+\boldsymbol{b_1}$. The upper edges coming from node x represent $\boldsymbol{w_{1x}}x$ and the lower edges coming from node z represent $\boldsymbol{w_{1z}}z$. The vector $\boldsymbol{b_1}$ is referred to as the constant term (or bias). Observe that $\boldsymbol{w_{1x}}x:\mathbb{R}^{d_x}\to\mathbb{R}^H$. In this case the scalar x is transferred by the weights $\boldsymbol{w_{1x}}$ into H inputs to the second layer, and similarly for z. At each node in the second layer the function $\sigma(\cdot):\mathbb{R}^1\to\mathbb{R}^1$ is applied to the H-dimensional

[1] This is without loss of generality since H can be set equal to the number of nodes in the widest layer, and for other layers, one can consider dummy nodes with weight set to zero and the constant parameter set to be equal to the root point of the activation function (0 for ReLU and Swish and a large negative number such as -100 for sigmoid).

term $\boldsymbol{w_{1x}}x + \boldsymbol{w_{1z}}z + \boldsymbol{b_1}$ element wise to produce H outputs. Then, recursively, in a feedforward fashion, the results are fed to the edges between the second and third layer, represented by $\boldsymbol{W_2}$ and $\boldsymbol{b_2}$, and so on. In the output layer, an inner product using weight $\boldsymbol{w_{L+1}}$ is applied to the output of the penultimate layer (the last hidden layer). Adding the scalar constant term b_{L+1} gives the scalar output.

Given observations $\{(\boldsymbol{X}^i, \boldsymbol{Z}^i, Y^i)\}_{i=1}^N$, we seek f_θ that solves (EC.1). That is, we train the DNN by selecting $\boldsymbol{W}$'s and $\boldsymbol{b}$'s that minimize the quadratic loss. A common approach is to use Stochastic Gradient Descent (SGD) which relies on the observation that the gradient of f_θ with respect to θ can be given in closed form, when σ is differentiable (Rumelhart et al. 1986). SGD has shown good empirical performance (Zhang et al. 2021). See Goodfellow et al. (2016) for an overview of SGD applied to fitting DNNs.

EC.2. Optimize the Fitted Neural Network

In this section, we discuss how to choose $\boldsymbol{z}$ given a covariate $\boldsymbol{X}_0$ so that the output of the fitted DNN is optimized. Let $\hat{\theta}$ denote the fitted DNN (the optimized weights $\boldsymbol{W}$ and constant terms $\boldsymbol{b}$). Then, following (5), we would want to find

$$\hat{\boldsymbol{z}}(\boldsymbol{X}_0) \in \arg\min_{\boldsymbol{z}\in\mathscr{Z}} f_{\hat{\theta}}(\boldsymbol{X}_0, \boldsymbol{z}). \tag{EC.2}$$

We assume

$$\mathscr{Z} = \{\boldsymbol{z} \in R^{d_Z} | \underline{\boldsymbol{Z}}_i \le \boldsymbol{z}_i \le \bar{\boldsymbol{Z}}_i,\ \forall\ i \le d_Z\},$$

where $\underline{\boldsymbol{Z}}_i$ and $\bar{\boldsymbol{Z}}_i$ indicate the lowest and highest values of the i^{th} coordinate of $\boldsymbol{Z}$ in the data. Consider Figure EC.1. We are suggesting that we have $\boldsymbol{X}_0$ and $f_{\hat{\theta}}$ and want to choose $\boldsymbol{z}$ to minimize Y. Solving (EC.2) may be NP-hard, depending on the choice of the activation function (Katz et al. 2017, Anderson et al. 2020).

Define $\hat{\boldsymbol{z}}(\boldsymbol{x})$ to be a local minimum if there exists $\delta > 0$, such that

$$\forall \boldsymbol{z} \in \mathscr{Z} : \|\boldsymbol{z} - \hat{\boldsymbol{z}}(\boldsymbol{x})\|^2 \le \delta, f_{\hat{\theta}}(\boldsymbol{x}, \hat{\boldsymbol{z}}(\boldsymbol{x})) \le f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z}). \tag{EC.3}$$

We suggest using a gradient-based approach to finding a local minimum to (EC.2). We can use backpropagation to derive the gradient of $f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ with respect to $\boldsymbol{z}$ in closed form. Hence, we may hope to efficiently find all the local minima to (EC.2). Moreover, we can use the tools developed for SGD in the training of DNNs to find the $\hat{\boldsymbol{z}}(\boldsymbol{x})$ that minimizes $f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})$.

Similar to the the backpropagation algorithm (Rumelhart et al. 1986), we use the layer by layer structure of the DNN to calculate the derivative of $f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ with respect to $\boldsymbol{z}$.

We can find a local minimum of $f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ using a gradient descent algorithm. Let $\hat{g} = \frac{\partial f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{\partial \boldsymbol{z}}$ and let $\hat{K}$ be the Lipschitz constant of $\frac{\partial f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{\partial \boldsymbol{z}}$ and let and $[\boldsymbol{z}]^+$ be the projection operator onto the feasible set $\mathscr{Z}$, defined as

$$[\boldsymbol{z}]_i^+ = \begin{cases} \bar{\boldsymbol{Z}}_i & \text{if } \boldsymbol{z}_i \geq \bar{\boldsymbol{Z}}_i, \\ \underline{\boldsymbol{Z}}_i & \text{if } \boldsymbol{z}_i \leq \underline{\boldsymbol{Z}}_i, \\ \boldsymbol{z}_i & \text{otherwise.} \end{cases}$$

Algorithm 1 Projected gradient update

Require: $\hat{K}$, $0 < s < \frac{2}{\hat{K}}$

Require: Starting point decision value $\boldsymbol{z}^{(0)} \in \mathscr{Z}$

$k \leftarrow 1$

while Stopping criterion not met **do**

Compute gradient: $\hat{g} \leftarrow \frac{\partial \hat{f}_{\theta}(\boldsymbol{x}, \boldsymbol{z}^{(k)})}{\partial \boldsymbol{z}}$

Apply update: $\hat{\boldsymbol{z}}^{(k)} \leftarrow \boldsymbol{z}^{(k)} - s\hat{g}$

Project back to feasible set: $\boldsymbol{z}^{(k+1)} \leftarrow [\hat{\boldsymbol{z}}^{(k)}]^+$

$k \leftarrow k + 1$

end while

There is a large body of literature on how to choose the step size (s) in Algorithm 1 to ensure faster convergence (Bertsekas 1999). However, if $f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ is continuously differentiable, $\mathscr{Z}$ is a closed and nonempty convex set, and $\frac{\partial f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{\partial \boldsymbol{z}}$ is Lipschitz continuous with a known Lipschitz constant, Algorithm 1 as stated will converge in the limit to a stationary point of $f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ (Bertsekas (1999), Proposition 2.3.2). If $\frac{\partial f_{\hat{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{\partial \boldsymbol{z}}$ is only locally Lipschitz continuous or its Lipschitz constant is not known, one can alternatively use the celebrated Armijo rule to determine the step size schedule that will guarantee convergence in the limit (Bertsekas (1999), Proposition 2.3.3).

Remark EC.1. The differentiability of σ is imperative for Proposition 1. If σ is a non-differentiable function such as ReLU, as pointed out by several recent works such as Bolte and Pauwels (2021) and Bianchi et al. (2022), backpropagation for ReLU networks does not necessarily compute any kind of known derivative or even subdifferential of $f_{\hat{\theta}}$ and may not even return the correct derivative of $f_{\hat{\theta}}$ at points where it is differentiable. Moreover, we do not know of any research that shows in such a case, similar to Proposition 1, convergence to a stationary point of $f_{\hat{\theta}}(\boldsymbol{x}, \cdot)$ would result. In particular, the results of Bianchi et al. (2022) which are developed for SGD, can be extended to the deterministic case of Algorithm 1 for ReLU activated DNNs, with backpropagation used to calculate $\hat{g}$. However their results only guarantee convergence to the set of zeros of the conservative field of $f_{\hat{\theta}}(\boldsymbol{x}, \cdot)$ and this set is not necessarily identical to the set of

its stationary points. (See Bolte and Pauwels (2021) for a comprehensive review of conservative fields). □

EC.3. Proofs

Proof of Proposition 1 As σ is differentiable everywhere $f_{\hat{\theta}}(\boldsymbol{x},\boldsymbol{z})$ is differentiable. Thus, we can apply the chain rule layer by layer, inductively.

Then, we observe

$$\frac{\partial \sigma_{h,1}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}} = \sigma'(\boldsymbol{w}_{\boldsymbol{h,1x}}^{\top}\boldsymbol{x} + \boldsymbol{w}_{\boldsymbol{h,1z}}^{\top}\boldsymbol{z} + b_{h,1}).\boldsymbol{w}_{\boldsymbol{h,1z}}.$$

So

$$\frac{\partial \boldsymbol{\sigma_1}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}} = \operatorname{diag}\big(\sigma'_{1,1}(\boldsymbol{x},\boldsymbol{z}),\ldots,\sigma'_{H,1}(\boldsymbol{x},\boldsymbol{z})\big) \times \boldsymbol{W_{1z}}.$$

Furthermore,

$$\frac{\partial \boldsymbol{\sigma_2}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}} = \frac{\partial \boldsymbol{\sigma_2}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{\sigma_1}(\boldsymbol{x},\boldsymbol{z})} \times \frac{\partial \boldsymbol{\sigma_1}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}} = \operatorname{diag}\big(\sigma'_{1,2}(\boldsymbol{x},\boldsymbol{z}),\ldots,\sigma'_{H,2}(\boldsymbol{x},\boldsymbol{z})\big) \times \boldsymbol{W_2} \times \frac{\partial \boldsymbol{\sigma_1}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}}.$$

Assume for some $l \geq 2$, that

$$\frac{\partial \boldsymbol{\sigma_l}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}} = \operatorname{diag}\big(\sigma'_{1,l}(\boldsymbol{x},\boldsymbol{z}),\ldots,\sigma'_{H,l}(\boldsymbol{x},\boldsymbol{z})\big) \times \boldsymbol{W_l} \times \frac{\partial \boldsymbol{\sigma_{l-1}}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}}.$$

Then, we have

$$\frac{\partial \boldsymbol{\sigma_{l+1}}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}} = \frac{\partial \boldsymbol{\sigma_{l+1}}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{\sigma_l}(\boldsymbol{x},\boldsymbol{z})} \times \frac{\partial \boldsymbol{\sigma_l}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}} = \operatorname{diag}\big(\sigma'_{1,l+1}(\boldsymbol{x},\boldsymbol{z}),\ldots,\sigma'_{H,l+1}(\boldsymbol{x},\boldsymbol{z})\big) \times \boldsymbol{W_{l+1}} \times \frac{\partial \boldsymbol{\sigma_l}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}}.$$

Then, finally,

$$\frac{\partial f_{\hat{\theta}}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}} = \frac{\partial f_{\hat{\theta}}}{\partial \boldsymbol{\sigma_L}(\boldsymbol{x},\boldsymbol{z})} \times \frac{\partial \boldsymbol{\sigma_L}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}} = \boldsymbol{w}_{L+1}^{\top} \times \frac{\partial \boldsymbol{\sigma_L}(\boldsymbol{x},\boldsymbol{z})}{\partial \boldsymbol{z}},$$

which completes the proof. □

Proof of Proposition 2 We note that by assumption $\sigma(\cdot)$ is twice continuously differentiable and hence $f_{\hat{\theta}}(\boldsymbol{x},\boldsymbol{z})$ is twice continuously differentiable. Moreover, we can apply the chain rule to the nodes in each layer, inductively. Thus, the proof follows from applying the chain rule to $\sigma(\cdot)$ and its derivative. To elaborate, it is trivial to show that

$$\frac{\partial \sigma_{h,1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j} = w_{h1}^{j}\sigma'_{h,1}(\boldsymbol{x},\boldsymbol{z}),$$

$$\frac{\partial^2 \sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j \partial z_k} = w_{h1}^{j} w_{h1}^{k}\sigma''_{h,l}(\boldsymbol{x},\boldsymbol{z}).$$

Now, assume $l \geq 2$, from the chain rule we have

$$\frac{\partial \sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j} = \frac{\partial \sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial\big(\sum_{h'=1}^{H} w_{h,l}^{h'}\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z}) + b_{h,l}\big)} \frac{\partial\big(\sum_{h'=1}^{H} w_{h,l}^{h'}\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z}) + b_{h,l}\big)}{\partial z_j} = \sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})\Big(\sum_{h'=1}^{H} w_{h,l}^{h'} \frac{\partial \sigma_{h,l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j}\Big),$$

while

$$\frac{\partial^2\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k}=\frac{\partial\sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial z_k}\Big(\sum_{h'=1}^{H}w_{h,l}^{h'}\frac{\partial\sigma_{h,l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j}\Big)+\frac{\partial\big(\sum_{h'=1}^{H}w_{h,l}^{h'}\frac{\partial\sigma_{h,l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j}\big)}{\partial z_k}\sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})$$

by applying the chain rule to the term $\frac{\partial\sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial z_k}$, we get

$$\begin{aligned}\frac{\partial^2\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k}=&\frac{\partial\sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial\big(\sum_{h'=1}^{H}w_{h,l}^{h'}\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z})+b_{h,l}\big)}\frac{\partial\big(\sum_{h'=1}^{H}w_{h,l}^{h'}\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z})+b_{h,l}\big)}{\partial z_k}\Big(\sum_{h'=1}^{H}w_{h,l}^{h'}\frac{\partial\sigma_{h,l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j}\Big)\\&+\sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})\Big(\sum_{h'=1}^{H}w_{h,l}^{h'}\frac{\partial^2\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k}\Big),\end{aligned}$$

which leads to

$$\frac{\partial^2\sigma_{h,l}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k}=\sigma''_{h,l}(\boldsymbol{x},\boldsymbol{z})\Big(\sum_{h'=1}^{H}w_{h,l}^{h'}\frac{\partial\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_k}\Big)\Big(\sum_{h'=1}^{H}w_{h,l}^{h'}\frac{\partial\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j}\Big)+\sigma'_{h,l}(\boldsymbol{x},\boldsymbol{z})\Big(\sum_{h'=1}^{H}w_{h,l}^{h'}\frac{\partial^2\sigma_{h',l-1}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k}\Big).$$

The last part of the Proposition follows trivially from noticing that $f_{\hat\theta}(\boldsymbol{x},\boldsymbol{z})=\sum_{h'=1}^{H}w_{L+1}^{h'}\sigma_{h,L}(\boldsymbol{x},\boldsymbol{z})$, and hence:

$$\frac{\partial^2 f_{\hat\theta}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k}=\sum_{h=1}^{H}w_{L+1}^{h}\frac{\partial^2\sigma_{h',L}(\boldsymbol{x},\boldsymbol{z})}{\partial z_j\partial z_k}.$$

□

Proof of Lemma 1 By Assumption 2, we have that $\mathbb{E}\,\|f_*(\boldsymbol{X},\boldsymbol{Z})-f_{\hat\theta}(\boldsymbol{X},\boldsymbol{Z})\|^2\le\delta_N$. Now let $(\boldsymbol{x}^{min},\boldsymbol{z}^{min})\in\arg\min_{(\boldsymbol{x},\boldsymbol{z})\in B}\|f_*(\boldsymbol{x},\boldsymbol{z})-f_{\hat\theta}(\boldsymbol{x},\boldsymbol{z})\|$ and $(\boldsymbol{x}^{max},\boldsymbol{z}^{max})\in\arg\max_{(\boldsymbol{x},\boldsymbol{z})\in B}\|f_*(\boldsymbol{x},\boldsymbol{z})-f_{\hat\theta}(\boldsymbol{x},\boldsymbol{z})\|$.

We have:

$$\int_{(\boldsymbol{x},\boldsymbol{z})\in B}\|f_*(\boldsymbol{x}^{min},\boldsymbol{z}^{min})-f_{\hat\theta}(\boldsymbol{x}^{min},\boldsymbol{z}^{min})\|^2d\mu_{X,Z}\le\int_{(\boldsymbol{x},\boldsymbol{z})\in B}\|f_*(\boldsymbol{x},\boldsymbol{z})-f_{\hat\theta}(\boldsymbol{x},\boldsymbol{z})\|^2d\mu_{X,Z},$$

while we know :

$$\int_{(\boldsymbol{x},\boldsymbol{z})\in B}\|f_*(\boldsymbol{x},\boldsymbol{z})-f_{\hat\theta}(\boldsymbol{x},\boldsymbol{z})\|^2d\mu_{X,Z}\le\int_{(\boldsymbol{x},\boldsymbol{z})\in(\boldsymbol{X},\boldsymbol{Z})}\|f_*(\boldsymbol{x},\boldsymbol{z})-f_{\hat\theta}(\boldsymbol{x},\boldsymbol{z})\|^2d\mu_{X,Z}\le\delta_N.$$

$$\Longrightarrow S\|f_*(\boldsymbol{x}^{min},\boldsymbol{z}^{min})-f_{\hat\theta}(\boldsymbol{x}^{min},\boldsymbol{z}^{min})\|^2\le\delta_N$$

Thus we can see that:

$$|f_*(\boldsymbol{x}^{min},\boldsymbol{z}^{min})-f_{\hat\theta}(\boldsymbol{x}^{min},\boldsymbol{z}^{min})|\le\sqrt{\frac{\delta_N}{S}}.\tag{EC.4}$$

At this point, we need to relate $|f_*(\boldsymbol{x}^{max},\boldsymbol{z}^{max})-f_{\hat\theta}(\boldsymbol{x}^{max},\boldsymbol{z}^{max})|$ to $|f_*(\boldsymbol{x}^{min},\boldsymbol{z}^{min})-f_{\hat\theta}(\boldsymbol{x}^{min},\boldsymbol{z}^{min})|$. We have

$$\begin{aligned}&|f_*(\boldsymbol{x}^{max},\boldsymbol{z}^{max})-f_{\hat\theta}(\boldsymbol{x}^{max},\boldsymbol{z}^{max})|-|f_*(\boldsymbol{x}^{min},\boldsymbol{z}^{min})-f_{\hat\theta}(\boldsymbol{x}^{min},\boldsymbol{z}^{min})|\le\\&\Big|\,|f_*(\boldsymbol{x}^{max},\boldsymbol{z}^{max})-f_{\hat\theta}(\boldsymbol{x}^{max},\boldsymbol{z}^{max})|-|f_*(\boldsymbol{x}^{min},\boldsymbol{z}^{min})-f_{\hat\theta}(\boldsymbol{x}^{min},\boldsymbol{z}^{min})|\,\Big|.\end{aligned}$$

Moreover, since we know $|f_*(\boldsymbol{x},\boldsymbol{z}) - f_{\hat{\theta}}(\boldsymbol{x},\boldsymbol{z})|$ is K-Lipschitz continuous, and by definition $\|(\boldsymbol{x}^{max},\boldsymbol{z}^{max}) - (\boldsymbol{x}^{min},\boldsymbol{z}^{min})\| \leq 2b$, we have

$$\big|f_*(\boldsymbol{x}^{max},\boldsymbol{z}^{max}) - f_{\hat{\theta}}(\boldsymbol{x}^{max},\boldsymbol{z}^{max})| - |f_*(\boldsymbol{x}^{min},\boldsymbol{z}^{min}) - f_{\hat{\theta}}(\boldsymbol{x}^{min},\boldsymbol{z}^{min})\big| \leq 2bK.$$

Therefore, it follows that

$$|f_*(\boldsymbol{x}^{max},\boldsymbol{z}^{max}) - f_{\hat{\theta}}(\boldsymbol{x}^{max},\boldsymbol{z}^{max})| - |f_*(\boldsymbol{x}^{min},\boldsymbol{z}^{min}) - f_{\hat{\theta}}(\boldsymbol{x}^{min},\boldsymbol{z}^{min})| \leq 2bK.$$

Hence, we have

$$|f_*(\boldsymbol{x}^{max},\boldsymbol{z}^{max}) - f_{\hat{\theta}}(\boldsymbol{x}^{max},\boldsymbol{z}^{max})| \leq |f_*(\boldsymbol{x}^{min},\boldsymbol{z}^{min}) - f_{\hat{\theta}}(\boldsymbol{x}^{min},\boldsymbol{z}^{min})| + 2bK \leq \sqrt{\frac{\delta_N}{S}} + 2bK,$$

where the last inequality follows from (EC.4). □

Proof of Proposition 3. To ease with notation, for a given $(\boldsymbol{x_0},\boldsymbol{z}^*(\boldsymbol{x_0}))$, we define a $d_X + d_Z$ dimensional ball around it.

Definition EC.1. For some $\boldsymbol{x}_0$ that satisfies the conditions of Assumption 5, we define $B_r := \{(\boldsymbol{x},\boldsymbol{z}) \in \mathbb{R}^{d_X} \times \mathbb{R}^{d_Z} : \|(\boldsymbol{x},\boldsymbol{z}) - (\boldsymbol{x_0},\boldsymbol{z}^*(\boldsymbol{x_0}))\| \leq r\}$.

First, we will show that for any $\boldsymbol{z}$ such that $\|\boldsymbol{z} - \boldsymbol{z}^*(\boldsymbol{x_0})\| > r$, $\boldsymbol{z} \neq \hat{\boldsymbol{z}}(\boldsymbol{x_0})$. By Assumption 6 we have $c_2 - 2(c_3 + r)K > 0$.

Let δ_N is small enough such that $c_2 - 2(c_3 + r)K > (\sqrt{\frac{\delta_N}{c_4}} + \sqrt{\frac{\delta_N}{L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} c_1 r^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}})$. Define $S_r = \int_{(\boldsymbol{x},\boldsymbol{z})\in B_r} d\mu_{X,Z}$. Then we have

$$c_2 - 2(c_3 + r)K > (\sqrt{\frac{\delta_N}{c_4}} + \sqrt{\frac{\delta_N}{L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} c_1 r^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}}) \geq (\sqrt{\frac{\delta_N}{c_4}} + \sqrt{\frac{\delta_N}{S_r}}). \tag{EC.5}$$

The last inequality follows from Assumption 5 and the fact that

$$\int_{(\boldsymbol{x},\boldsymbol{z})\in B_r} d\mu_{X,Z} \geq c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} r^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}.$$

Now, consider some $\boldsymbol{z} \in [-1,1]^{d_Z}$ such that $\|\boldsymbol{z} - \boldsymbol{z}^*(\boldsymbol{x_0})\| > r$. Then $\|(\boldsymbol{x_0},\boldsymbol{z}) - (\boldsymbol{x_0},\boldsymbol{z}^*(\boldsymbol{x_0}))\| > r$. It directly follows from the definition of $\hat{\boldsymbol{z}}(\boldsymbol{x_0})$ and Lemma 1 that:

$$f_{\hat{\theta}}(\boldsymbol{x_0},\hat{\boldsymbol{z}}(\boldsymbol{x_0})) \leq f_{\hat{\theta}}(\boldsymbol{x_0},\boldsymbol{z}^*(\boldsymbol{x_0})) \leq f_*(\boldsymbol{x_0},\boldsymbol{z}^*(\boldsymbol{x_0})) + \sqrt{\frac{\delta_N}{S_r}} + 2rK. \tag{EC.6}$$

Moreover, from Assumption 6, we have that $f_*(\boldsymbol{x_0},\boldsymbol{z}) - f_*(\boldsymbol{x_0},\boldsymbol{z}^*(x_0)) \geq c_2$. Thus

$$f_*(\boldsymbol{x_0},\boldsymbol{z}^*(\boldsymbol{x_0})) + \sqrt{\frac{\delta_N}{S_r}} + 2rK \leq f_*(\boldsymbol{x_0},\boldsymbol{z}) - c_2 + \sqrt{\frac{\delta_N}{S_r}} + 2rK. \tag{EC.7}$$

Furthermore, given (EC.5), from Inequalities (EC.6) and (EC.7) we have:

$$f_{\hat{\theta}}(\boldsymbol{x_0},\hat{\boldsymbol{z}}(\boldsymbol{x_0})) \leq f_*(\boldsymbol{x_0},\boldsymbol{z}) - c_2 + \sqrt{\frac{\delta_N}{S_r}} + 2rK < f_*(\boldsymbol{x_0},\boldsymbol{z}) - \sqrt{\frac{\delta_N}{c_4}} - 2c_3K. \tag{EC.8}$$

Now, we need to bound the error of $f_{\hat{\theta}}$ at $(\boldsymbol{x_0}, \boldsymbol{z})$. By Assumption 6 we know that a $d_{\boldsymbol{X}} + d_{\boldsymbol{Z}}$ dimensional ball of radius c_3 around $(\boldsymbol{x_0}, \boldsymbol{z})$ has a probability measure of c_4. Hence we can use Lemma 1, suggesting:

$$|f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z}) - f_*(\boldsymbol{x_0}, \boldsymbol{z})| \leq \sqrt{\frac{\delta_N}{c_4}} + 2c_3K.$$

This would imply:

$$f_*(\boldsymbol{x_0}, \boldsymbol{z}) - \sqrt{\frac{\delta_N}{c_4}} - 2c_3K \leq f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z})$$

Along with Inequality (EC.8) we finally have:

$$f_{\hat{\theta}}(\boldsymbol{x_0}, \hat{\boldsymbol{z}}(\boldsymbol{x_0})) \leq f_*(\boldsymbol{x_0}, \boldsymbol{z}) - c_2 + \sqrt{\frac{\delta_N}{S_r}} + 2rK < f_*(\boldsymbol{x_0}, \boldsymbol{z}) - \sqrt{\frac{\delta_N}{c_4}} - 2c_3K \leq f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z}).$$

Thus, we have proven that for some $\boldsymbol{z}$ if $\|\boldsymbol{z} - \boldsymbol{z}^*(\boldsymbol{x_0})\| > r$, we have:

$$f_{\hat{\theta}}(\boldsymbol{x_0}, \hat{\boldsymbol{z}}(\boldsymbol{x_0})) < f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z}),$$

which proves we have:

$$\|\hat{\boldsymbol{z}}(\boldsymbol{x_0}) - \boldsymbol{z}^*(\boldsymbol{x_0})\| \leq r.$$

In other words, the optimal decision output by the fitted NN is within a neighborhood of the actual optimal decision. Next we find the smallest such neighborhood and show that they are getting closer as δ_N decreases.

For some $r_1 < r$, We want to show that for any $\boldsymbol{z}$ such that $\|\boldsymbol{z} - \boldsymbol{z}^*(\boldsymbol{x_0})\| > r_1$, we have

$$f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0})) < f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z}).$$

Now consider $\boldsymbol{z'}$ such that $r_1 < \|\boldsymbol{z'} - \boldsymbol{z}^*(\boldsymbol{x_0})\| \leq r$. We know from Assumption 7 that:

$$f_*(\boldsymbol{x_0}, \boldsymbol{z'}) > f_*(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0})) + c_5 r_1^2.$$

At this point, we would like to lower bound $f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z'})$ using $f_*(\boldsymbol{x_0}, \boldsymbol{z'})$. To that avail, consider a $(d_X + d_Z)$ dimensional ball B_{r_2} that contains $(\boldsymbol{x_0}, \boldsymbol{z'})$ with radius r_2 that is entirely within the Euclidean distance r from $(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0}))$. As long as $r_2 < r$, we can always achieve this, since this ball does not need to be centered around $(\boldsymbol{x_0}, \boldsymbol{z'})$. From Lemma 1 we have:

$$f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z'}) \geq f_*(\boldsymbol{x_0}, \boldsymbol{z'}) - \sqrt{\frac{\delta_N}{S}} - 2r_2K \geq f_*(\boldsymbol{x_0}, \boldsymbol{z'}) - \sqrt{\frac{\delta_N}{c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} r_2^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}} - 2r_2K, \tag{EC.9}$$

where $S = \int_{(\boldsymbol{x},\boldsymbol{z}) \in B_{r_2}} d\mu_{X,Z}$. The last inequality follows from Assumption 5 and the fact that

$$\int_{(\boldsymbol{x},\boldsymbol{z}) \in B_{r_2}} d\mu_{X,Z} \geq c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} r_2^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}.$$

Next, we would like to also develop an upper bound on $f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0}))$. Thus, let us consider another Euclidean ball around $(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0}))$ with radius r_2. Then, by Lemma 1 we know that

$$f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0})) \leq f_*(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0})) + \sqrt{\frac{\delta_N}{c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} r_2^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}} + 2r_2K. \tag{EC.10}$$

Then following Inequality (EC.10) and remembering that by construction we had $\|\boldsymbol{z'} - \boldsymbol{z}^*(\boldsymbol{x_0})\| > r_1$, form Assumption 7 it follows that:

$$f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0})) \leq f_*(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0})) + \sqrt{\frac{\delta_N}{c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} r_2^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}} + 2r_2K < f_*(\boldsymbol{x_0}, \boldsymbol{z'}) - c_5 r_1^2 + \sqrt{\frac{\delta_N}{c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} r_2^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}} + 2r_2K. \tag{EC.11}$$

Therefore, if the following inequality holds

$$\sqrt{\frac{\delta_N}{L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} c_1 r_2^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}} + 2r_2K \leq \frac{c_5 r_1^2}{2}, \tag{EC.12}$$

for the right hand side of Inequality EC.10 and from Inequality (EC.9) we will have:

$$f_*(\boldsymbol{x_0}, \boldsymbol{z'}) - c_5 r_1^2 + \sqrt{\frac{\delta_N}{c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} r_2^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}} + 2r_2K \leq f_*(\boldsymbol{x_0}, \boldsymbol{z'}) - \sqrt{\frac{\delta_N}{c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} r_2^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}} - 2r_2K \leq f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z'}).$$

This, along with Inequality (EC.11), proves that

$$f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0})) < f_{\hat{\theta}}(\boldsymbol{x_0}, \boldsymbol{z'}).$$

To find the smallest r_1 for which Inequality (EC.12) holds, we note that r_1 will achieve its lowest value (that satisfies Inequality (EC.12)) when r_2 is set to be the minimizer of the left hand side of Inequality (EC.12), which happens to be convex with respect to r_2. To prove the convexity, we notice that:

$$\left(\sqrt{\frac{\delta_N}{L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} c_1 r_2^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}} + 2r_2K\right)' = -\frac{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}{2}\sqrt{\frac{\delta_N}{L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} c_1}} r_2^{-\frac{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2}{2}} + 2K,$$

while:

$$\left(-\frac{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}{2}\sqrt{\frac{\delta_N}{L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} c_1}} r_2^{-\frac{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2}{2}} + 2K\right)' = \frac{(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2)}{4}\sqrt{\frac{\delta_N}{c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}} r_2^{-\frac{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+4}{2}} \geq 0.$$

Hence, the best r_2 would satisfy the first order condition of the left hand side of Inequality (EC.12) with respect to r_2. Therefore, we have:

$$r_2 = \left(\frac{(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})^2 \delta_N}{16K^2 c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}\right)^{\frac{1}{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2}}.$$

Then by plugging r_2 into the left hand side of Inequality (EC.12), we will have:

$$\sqrt{\frac{\delta_N}{c_1 L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}} r_2^{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}} + 2r_2K = \frac{\delta_N(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})}{4Kc_1L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}} + 2K\left(\frac{(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})^2\delta_N}{16K^2c_1L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}\right)^{\frac{1}{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2}}.$$

Therefore, setting

$$r_1 = \sqrt{2\frac{\frac{\delta_N(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})}{4Kc_1L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}} + 2K\left(\frac{(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})^2\delta_N}{16K^2c_1L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}\right)^{\frac{1}{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2}}}{c_5}},$$

completes the proof. We note that as long as δ_N is small enough such that:

$$\left(\frac{(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})^2\delta_N}{16K^2c_1L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}\right)^{\frac{1}{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2}} \leq \sqrt{2\frac{\frac{\delta_N(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})}{4Kc_1L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}} + 2K\left(\frac{(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})^2\delta_N}{16K^2c_1L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}\right)^{\frac{1}{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2}}}{c_5}} \leq r,$$

we have also ensured the ball B_{r_2} containing $(\boldsymbol{x}, \boldsymbol{z'})$ was entirely within radius r of $(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0}))$.

To finalize the proof, we note that since $\|\hat{\boldsymbol{z}}(\boldsymbol{x_0}) - \boldsymbol{z}^*(\boldsymbol{x_0})\| \leq r_1$, it follows directly from Assumption 3 that:

$$|f_*(\boldsymbol{x_0}, \boldsymbol{z}^*(\boldsymbol{x_0})) - f_*(\boldsymbol{x_0}, \hat{\boldsymbol{z}}(\boldsymbol{x_0}))| \leq K_* \sqrt{2\frac{\frac{\delta_N(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})}{4Kc_1L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}} + 2K\left(\frac{(d_{\boldsymbol{X}}+d_{\boldsymbol{Z}})^2\delta_N}{16K^2c_1L_{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}}}\right)^{\frac{1}{d_{\boldsymbol{X}}+d_{\boldsymbol{Z}}+2}}}{c_5}}.$$

□

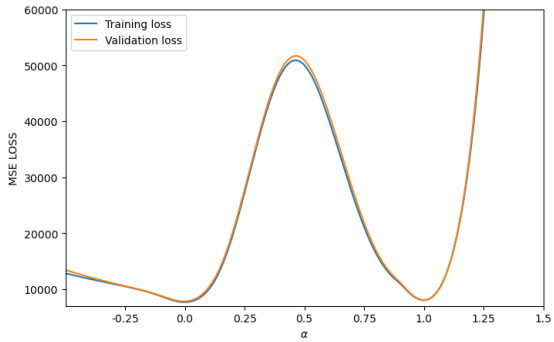


Figure EC.2 MSE loss of the linear interpolation of small and large batch DNNs.

EC.4. Additional Investigations and Figures

In some applications of neural networks such as image classification, researchers have pointed out that neural networks trained with SGD with large batch sizes may not generalize well (Keskar et al. 2016, Li et al. 2018). The existing research suggests this could be because large batch methods tend to converge to sharp minimizers of the training and testing loss functions (Keskar et al. 2016). Given the importance of generalization bounds in our paper, we look into such potential sharpness in Equation (6). Similar to Li et al. (2018), to visualize such sharpness, we consider two batch sizes of 50 (small batch) and 1000 (large batch) in the newsvendor experiment. On a data set comprised of $20,000$ training data points and $5,000$ test data points, we look at the converged DNNs of the small batch SGD and that of the large batch one. Let $\hat{\theta}^S$ denote the parameters of the fitted DNN with small batch, and $\hat{\theta}^L$ that of the large batch. Following Li et al. (2018), let $f_{\hat{\theta}^\alpha}$ be the neural network with parameters $\alpha \times \hat{\theta}^L + (1-\alpha) \times \hat{\theta}^S$. Figure EC.2 portrays the training and validation MSE loss of $f_{\hat{\theta}^\alpha}$ as a function of α varying for $-.5$ to 1.5. The result suggests that in the application, such sharpness of loss function for neural networks trained with large batch size SGD may not be present.

Figure EC.3 depicts the training and validation error as a function of training epochs for one of the DNNs in Example 1 when training with $20,000$ data points and validating with $5,000$. It portrays a robust effect across the experiments. The training error does not shrink to zero as observed in other applications of DNNs (most probably because we use relatively small DNNs), and further, the validation error tends to be slightly larger than the training error. In other words, we do not observe benign overfitting documented in, e.g., Bartlett et al. (2020). This is because we focus on the classic statistical learning regime in which the data size is not huge and the number of parameters is significantly less than the number of data points. Moreover, the figure provides evidence that in the newsvendor experiment, the SGD method used to train the DNNs does not suffer from unstable convergence, as reported elsewhere in the literature (Ahn et al. 2022).

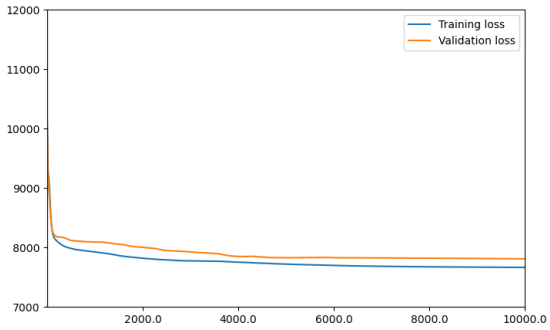


Figure EC.3 Training versus Validation Loss.

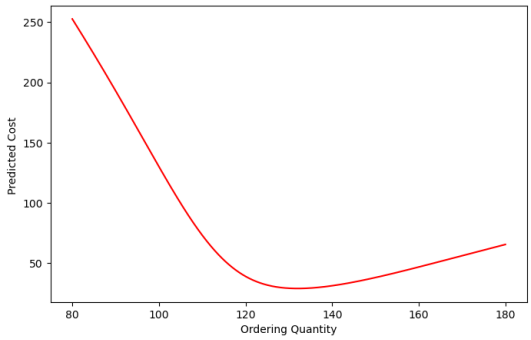


Figure EC.4 The illustration of the effect of order quantity on predicted cost.

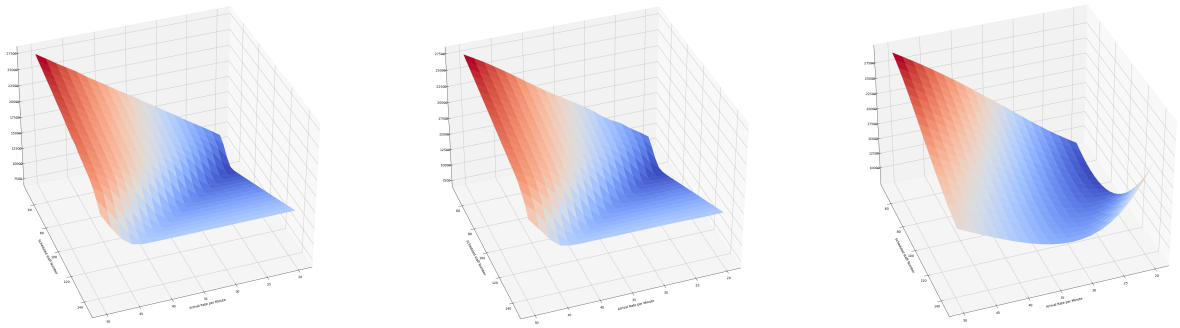


Figure EC.5 $G/GI/z+G$ actual and predicted costs. Left is the actual cost, middle is the DONN prediction and right is the PR prediction.

Figure EC.4 depicts the predicted newsvendor cost as a function of order quantity when all other covariates are fixed (after training with 250 days of data for one of the DNNs). The figure suggests a smooth and well-behaved objective function that strongly resembles the actual newsvendor objective function in Equation 12. Hence, if the true objective function does not suffer from local optima, we expect the fitted DNNs prediction value to exhibit a similar trend. In Example 1 we optimize the DNNs from multiple starting points and all of the results are nearly identical suggesting a lack of local optima issues.

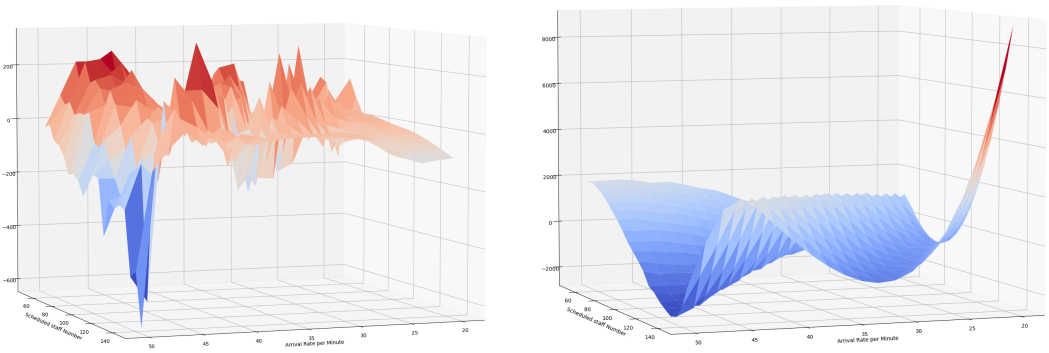


Figure EC.6 Prediction error of DNN and Polynomial Regression in the $G/GI/z+G$ case.

Figure EC.5 presents the actual cost (left figure) and predicted cost (DONN center figure, PR right figure) of the $G/GI/z+G$ queue for various arrival rates and staffing levels. Figure EC.6 compliments Figure EC.5. The left-hand side figure suggests that when predicting the daily cost of the call center staffing problem (Example 3) for the $G/GI/z+G$ case, the error by the DNN is quite small (generally less than 1%) and centered around zero throughout the tested ranges of staffing levels and arrival rates. The error is never larger than 5% of the actual objective function (portrayed in the left figure in Figure EC.5). The right figure in Figure EC.6 suggests the same cannot be said about Polynomial Regression method that we tested and it may suffer from very large errors in certain regions.

Figures EC.7a, EC.7b and EC.7c pertain to the predicted optimal cost using each methodology, in the linear staffing case, increasing convex staffing and $G/GI/z+G$ case, respectively. We emphasize that these results are not simulated and are the values obtained by plugging each method's optimal staffing level into its approximation of Equation (21).

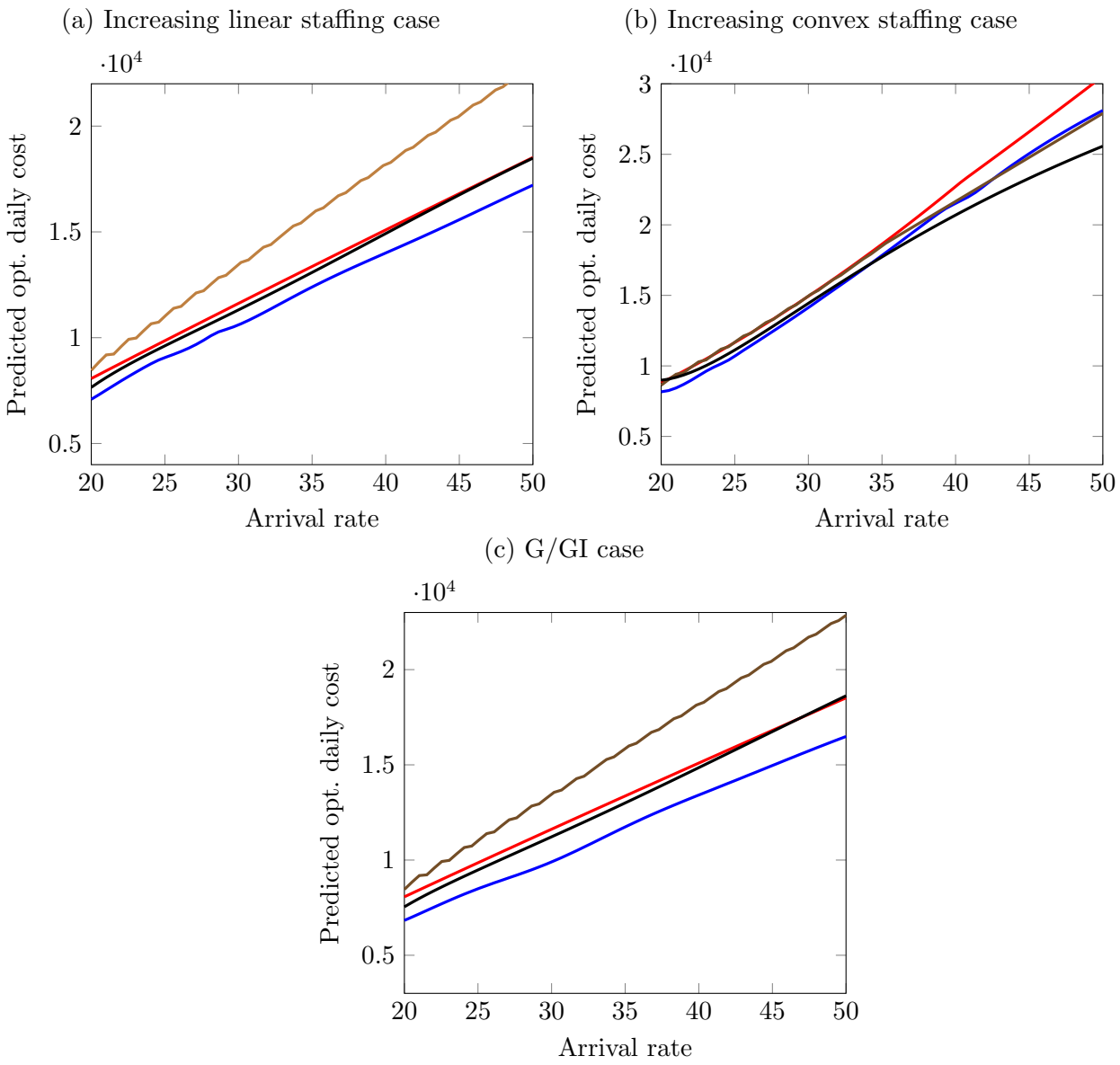


Figure EC.7 The illustration of the predicted costs of each methodology in the queuing setting.

EC.5. Example

In this section, we look at an example on how to apply our framework in details. [2]. Consider an example with one scalar covariate x and one scalar decision variable z where $f_*(x,z) = (x-z)^2$. We simulate a large data set with $50,000$ data points where x and z are uniformly and independently drawn from $[-10,10]$. We train a DNN with $L=3$, $H=3$ and Swish activation function. The fitted DNN has the following weight matrices and constant terms.

[2] See https://github.com/saman-lagzi/Data-driven-Optimization-with-Neural-Networks for a detailed Python implementation of Example EC.5 along with the generated dataset.

$$\boldsymbol{W_1}=\begin{bmatrix} -1.7124 & 1.7100 \\ 0.0081 & -0.0080 \\ 1.7235 & -1.7209 \end{bmatrix}, \boldsymbol{W}_{1z}=\begin{bmatrix} 1.7100 \\ -0.0080 \\ -1.7209 \end{bmatrix}, \boldsymbol{b}_1=\begin{bmatrix} -1.6824 \\ 1.3022 \\ -1.7184 \end{bmatrix},$$

$$\boldsymbol{W_2}=\begin{bmatrix} 4.6518 & 1.9831 & 4.5889 \\ 0.2762 & -1.0031 & -0.2200 \\ -1.4228 & 3.8292 & -1.4879 \end{bmatrix}, \boldsymbol{b}_2=\begin{bmatrix} 2.3721 \\ -2.2213 \\ 3.6026 \end{bmatrix},$$

$$\boldsymbol{W_3}=\begin{bmatrix} 1.4862 & -1.1508 & -5.0347 \\ -0.1134 & -0.1684 & -0.1325 \\ 4.0344 & -0.6634 & -1.0397 \end{bmatrix}, \boldsymbol{b}_3=\begin{bmatrix} 2.4532 \\ 0.1484 \\ 2.0276 \end{bmatrix},$$

$$\boldsymbol{W_4}=\begin{bmatrix} 2.1921 \\ 0.8658 \\ 3.2429 \end{bmatrix}, b_4 = 0.9145.$$

First, we note that the derivative of the fitted DNN with respect to the decision variable z at any given point, calculated through Proposition 1 is identical to the value calculated using torch.autograd function in PyTorch. For example, at input point $[1,1]$ both methods return a gradient of $\frac{\partial f_{\hat{\theta}}}{\partial z}(1,1) = 0.183$. Moreover, the point $x = 0$ translates into 0.0014 when standardized for input to the fitted DNN, and using both Proposition 1 and the torch.autograd in PyTorch we have $\hat{z}(0.0014) = -0.0055$ where $\frac{\partial f_{\hat{\theta}}}{\partial z}(0.0014, -0.0055) = 0$.

To find $\hat{z}(0.0014) = \arg\min_z f_{\hat{\theta}}(0.0014, z)$, we use the scipy.optimize package from Python with a starting point of $z_0 = -1.96$. We provide the gradient $\frac{\partial f_{\hat{\theta}}(0.0014,z)}{\partial z}$, using both our approach highlighted in Proposition 1 and the torch.autograd function in PyTorch. While both methods lead to $\hat{z}(0.0014) = -0.0055$, the execution time is 0.10 second for the first method and 0.12 second for the latter. The faster execution time based on Proposition 1 is well within expectation as it provides the closed form formula for $\frac{\partial f_{\hat{\theta}}(0.0014,z)}{\partial z}$ based on the weight matrices and constant terms of $f_{\hat{\theta}}$, however, torch.autograd is a generalized method and needs to apply the chain rule, layer by layer to $f_{\hat{\theta}}$ to find $\frac{\partial f_{\hat{\theta}}(0.0014,z)}{\partial z}$.