Machine Learning's Dropout Training is Distributionally Robust Optimal*

José Blanchet† Yang Kang‡ José Luis Montiel Olea§
Viet Anh Nguyen† Xuhui Zhang†

September 1st, 2020.

Abstract

Dropout training is an increasingly popular estimation method in machine learning that minimizes some given loss function (e.g., the negative expected log-likelihood), but averaged over nested submodels chosen at random.

This paper shows that dropout training in Generalized Linear Models is the minimax solution of a two-player, zero-sum game where an adversarial nature corrupts a statistician's covariates using a multiplicative nonparametric errors-in-variables model. In this game—known as a Distributionally Robust Optimization problem—nature's *least favorable distribution* is *dropout noise*, where nature independently deletes entries of the covariate vector with some fixed probability δ. Our decision-theoretic analysis shows that dropout training—the statistician's minimax strategy in the game—indeed provides out-of-sample expected loss guarantees for distributions that arise from multiplicative perturbations of in-sample data.

This paper also provides a novel, parallelizable, Unbiased Multi-Level Monte Carlo algorithm to speed-up the implementation of dropout training. Our algorithm has a much smaller computational cost compared to the naive implementation of dropout, provided the number of data points is much smaller than the dimension of the covariate vector.

Keywords: Generalized Linear Models, Distributionally Robust Optimization, Machine Learning, Minimax Theorem, Multi-Level Monte Carlo.

*We would like to thank Matias Cattaneo, Max Farrell, Michael Leung, Ulrich Müller, Hashem Pesaran, Ashesh Rambachan, Roger Moon, Stefan Wager, and seminar participants at the University of Southern California for helpful comments and suggestions. Jose Blanchet acknowledges support from NSF grants 1915967, 1820942, 1838576 and AFOSR MURI 19RT1056.

†Department of Management Science and Engineering, Stanford University

‡Department of Statistics, Columbia University

§Department of Economics, Columbia University

1 Introduction

Dropout training is an increasingly popular estimation method in machine learning.[1] The general idea consists in ignoring some dimensions of the covariate vector—chosen at random—while estimating the parameters of a statistical model. A common motivation for dropout training is that the random feature selection implicitly performs *model averaging*, potentially improving out-of-sample prediction error and thus mitigating overfitting; see Hinton, Srivastava, Krizhevsky, Sutskever, and Salakhutdinov (2012) for a discussion about this point in the context of neural networks.[2]

The main goal of this paper is to provide a novel decision-theoretic foundation for the use of dropout training in econometrics, statistics, and machine learning. Our work is part of a growing literature trying to provide theoretical results explaining the empirical success of dropout training in mitigating overfitting.[3] Our main result (Theorem 1) shows that dropping out input features when training Generalized Linear Models (McCullagh and Nelder, 1989) can be viewed as the minimax solution to an adversarial game known in the stochastic optimization literature (Shapiro, Dentcheva, and Ruszczyński (2014)) as a Distributionally Robust Optimization (DRO) problem.

Broadly speaking, a DRO problem is a two-player, zero-sum game between a decision maker (a statistician) and an adversary (nature).[4] The statistician wishes to choose an action to minimize a given expected loss (e.g. squared loss in a typical linear regression setting or, more generally, the negative of the log-likelihood function), while nature intends this loss to be maximal. The interest centers in understanding—theoretically and algorithmically—the minimax solution of the game.

To derive the main theoretical result, the paper suggests a framework in which nature is allowed to harm the statistician by corrupting the available data using a multiplicative nonparametric errors-in-variables model; as in the classical work of Hwang (1986). The statistician is aware of the data corruption and knows the distribution used by nature, but

[1] Section 7.12 of Goodfellow, Bengio, and Courville (2016) provides a textbook treatment on dropout training. Bishop (1995) and Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014) are seminal references on this topic.

[2] See also Draper (1994) and Raftery, Madigan, and Hoeting (1997) for classical results on the optimality of model averaging for prediction purposes.

[3] For example, Wager, Wang, and Liang (2013) show that dropout training can be interpreted as an L_2 regularization method in Generalized Linear Models up to a first-order approximation. Helmbold and Long (2015) show that, despite the first-order approximation, the regularization achieved by dropout training in deep neural networks can differ significantly from explicit L_2 regularizers. More recently, Wei, Kakade, and Ma (2020) have shown that in addition to the *explicit* regularization achieved in dropout training by changing the loss function, there is also *implicit* regularization when dropout is implemented via Stochastic Gradient Descent.

[4] This is an analogous formulation to the canonical statistical decision theory framework of Wald (1950)

does not have access to the realizations of the corruption noise. Under mild assumptions, nature's *least favorable distribution* in this game is shown to be *dropout noise*, where nature independently deletes entries of the covariate vector with some fixed probability δ. The Minimax Theorem (Morgenstern and von Neumann, 1953) is shown to also hold for this game: the minimax value coincides with the maximin solution, and these coincide with the payoffs in the game's Nash equilibrium. One direct consequence is that the statistician's selected procedure in the face of multiplicative nonparametric noise maintains optimal performance even if the adversary is allowed to corrupt after the statistician uses the training data.

Our main result (Theorem 1) shows that, by construction, dropout training indeed provides out-of-sample performance guarantees for distributions that arise from multiplicative perturbations of in-sample data. More precisely, given any fixed sample size, the *out-of-sample expected loss* is no larger than that obtained by dropout training *in-sample*, provided we consider out-of-sample distributions obtained as multiplicative perturbations of the in-sample distribution. Therefore, our result formally qualifies the ability of dropout training to enhance out-of-sample performance, which is one of the reasons often invoked to use the dropout method.

In addition to our theoretical result, this paper also suggests a new stochastic optimization implementation of dropout training. A well-known drawback of dropout is its computational cost. As we will explain, a d-dimensional covariate vector requires 2^d evaluations of the loss in order to integrate out the dropout noise for a particular data point. The computational cost is alleviated by implementing dropout training by using either Stochastic Gradient Descent (Robbins and Monro (1951)) or naive Monte-Carlo approximations to the expected loss, both of which require draws from the joint distribution of the data and dropout noise. Unfortunately, both of these approximations introduce bias to the solution of dropout training. Also, none of these procedures can exploit the increasing availability of parallel computing in order to alleviate their computational burden.

To address these issues, we borrow ideas from the Multi-level Monte Carlo literature—in particular from the work of Blanchet, Glynn, and Pei (2019a)—to suggest an unbiased (in a sense we will make precise) dropout training routine that is easily parallelizable and that has a much smaller computational cost compared to naive dropout training methods when the number of features is large (Theorem 2). Our algorithm thus complements the recent literature suggesting approaches to speed-up dropout training by either using a parallelized implementation of Stochastic Gradient Descent (Zinkevich, Weimer, Li, and Smola, 2010) or a fast dropout training based on Gaussian approximations (Wang and Manning, 2013).

The rest of the paper is organized as follows. Section 2 explains dropout training in the context of Generalized Linear Models. Section 3 presents a general description of the DRO

framework used in this paper. Section 4 specializes the DRO problem by using the negative log-likelihood of Generalized Linear Models to define a loss function for the statistician, and by allowing nature to harm the statistician via a multiplicative errors-in-variables model for the covariates. This section also presents our main theorem. Section 5 discusses different computational methods available for implementing dropout training (full integration, Stochastic Gradient Descent, Naive Monte Carlo integration) and presents our suggested *Unbiased Multi-level Monte Carlo* algorithm. Section 6 presents some simulations comparing our preferred implementation of dropout training to Stochastic Gradient Descent. Finally, Section 7 discusses extensions of our results to a particular class of feed-forward neural networks with a single hidden layer. Our calculations suggest that dropout training of the hidden units in the hidden layer is still minimax optimal, but dropping input features is only approximately optimal when the multiplicative noise is close to unity with high probability. All the proofs are collected in the Appendix.

2 Dropout Training in Generalized Linear Models

This section describes dropout training in the context of Generalized Linear Models.[5] As some other recent papers in the literature, we view Generalized Linear Models as a convenient, transparent, and relevant framework to better understand the theoretical and algorithmic properties of dropout training.[6]

2.1 Generalized Linear Models (GLMs)

A Generalized Linear Model—with parameters β and ϕ—is defined by a conditional density for the response variable $Y \in \mathcal{Y} \subseteq \mathbb{R}$ given $X = (X_1, \ldots, X_d)^\top \in \mathbb{R}^d$

$$f(Y|X, \beta, \phi) \equiv h(Y, \phi) \exp\left(\left(Y\beta^\top X - \Psi(\beta^\top X)\right)/a(\phi)\right), \tag{1}$$

see McCullagh and Nelder (1989, Equation 2.4). In our notation $h(\cdot, \phi)$ is a real-valued function parametrized by ϕ defined on the domain $\mathcal{Y}$, $a(\cdot)$ is a positive function of ϕ, and $\Psi(\cdot)$ is a convex, smooth function defined on all the real line. Normal, Logistic, and Poisson

[5] It is worth mentioning that the framework herein described does not encompass Neural Networks, where dropout training has been more widely adopted; see Hinton et al. (2012) and Bishop (1995). We discuss extensions of our results to a restrictive class of Neural Networks in Section 7.

[6] For example, Wager et al. (2013) showed that dropout training in Generalized Linear models is first-order equivalent to Maximum Likelihood Estimation with an L2 penalty. Wang and Manning (2013) used a logistic regression to illustrate how Gaussian approximations can be used to speed-up dropout training in neural networks.

Regression have conditional densities of the form (1).[7] For the sake of exposition, we provide details below for linear and logistic regression.

Example 1 (Linear regression with unknown variance)**.** *Consider the linear model* $Y = \beta^\top X + \varepsilon$*, in which* $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ *with unknown variance* $\sigma^2 \in \mathbb{R}_{++}$ *and* $\varepsilon \perp X$*. The conditional distribution of* Y *given* X *satisfies* (1) *with* $\phi = \sigma^2$*,* $a(\phi) = \phi$*,* $\Psi(\beta^\top X) = (\beta^\top X)^2/2$ *and* $h(Y, \phi) = (2\pi\phi)^{-\frac{1}{2}} \exp(-Y^2/(2\phi))$*.*

Example 2 (Logistic regression)**.** *Consider* $Y|X \sim$ *Bernoulli*$(1/(1+\exp(-\beta^\top X))$ *with* $\mathcal{Y} = \{0, 1\}$*. The conditional probability mass function of* Y *given* X *satisfies* (1) *with* $a(\phi) = 1$*,* $\Psi(\beta^\top X) = \log(1 + \exp(\beta^\top X))$ *and* $h(Y, \phi) = 1$*.*

Generalized Linear Models are typically estimated via Maximum Likelihood using (1). Given n i.i.d. data realizations or *training examples* (x_i, y_i), the Maximum Likelihood estimator $(\widehat{\beta}_{\text{ML}}, \widehat{\phi}_{\text{ML}})$ is defined as any solution of the problem

$$\min_{\beta,\phi} \sum_{i=1}^{n} - \ln f(y_i|x_i, \beta, \phi). \tag{2}$$

2.2 Dropout Training

An alternative to standard Maximum Likelihood estimation in GLMs is dropout training. The general idea consists in ignoring some randomly chosen dimensions of x_i while training a statistical model.

For a given covariate vector x_i—and an user-selected constant, $\delta \in (0, 1)$—define the d-dimensional random vector

$$\xi_i = (\xi_{i,1}, \ldots, \xi_{i,d}) \in \{0, 1/(1-\delta)\}^d,$$

where each of the d entries of ξ_i is an independent draw from a scaled Bernoulli distribution with parameter $1 - \delta$. This is, for $j = 1, \ldots, d$:

$$\xi_{i,j} = \begin{cases} 0 & \text{with probability } \delta, \\ (1-\delta)^{-1} & \text{with probability } (1-\delta). \end{cases} \tag{3}$$

Let $\odot$ denote the binary operator defining element-wise multiplication between two vectors of the same dimension. Consider the covariate vector

$$x_i \odot \xi_i \equiv (x_{i,1}\xi_{i,1}, \ldots, x_{i,d}\xi_{i,d}). \tag{4}$$

[7] See Table 2.1 p. 29 of McCullagh and Nelder (1989)

Some entries of the new covariate vector are 0 (those for which $\xi_{i,j} = 0$) and the rest are equal to $x_{i,j}/(1-\delta)$.

The estimators of (β, ϕ) obtained by *dropout training* correspond to any parameters $(\widehat{\beta}_{\text{DT}}, \widehat{\phi}_{\text{DT}})$ that solve the problem

$$\min_{\beta,\phi} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\xi_i} \left[-\ln f(y_i | x_i \odot \xi_i, \beta, \phi) \right]. \tag{5}$$

In a slight abuse of notation, $\mathbb{E}_{\xi_i}$ denotes the expectation with respect to the distribution of ξ_i.[8]

One possibility to solve the problem in (5) is to use Stochastic Gradient Descent (Robbins and Monro (1951)). This is tantamount to i) taking a draw of (x_i, y_i) according to its empirical distribution, ii) independently taking a draw of ξ_i using the distribution in (3) and iii) computing the stochastic gradient descent update using

$$\nabla \ln f(y_i | x_i \odot \xi_i, \beta, \phi). \tag{6}$$

We provide further details about the Stochastic Gradient Descent implementation of dropout training in Section 5.

2.3 Question of Interest

Adding noise to the Maximum Likelihood objective in (2) seems, at first glance, arbitrary. The dropout training estimators will generally not share the same probability limit as the Maximum Likelihood estimators whenever $\delta \neq 0$.[9]

Despite the lack of reasonable asymptotic properties, there is some literature that has provided empirical evidence that using intentionally corrupted features for training has the potential to improve the performance of machine learning algorithms; see Maaten, Chen, Tyree, and Weinberger (2013). Even if one is willing to accept that corrupting features is

[8] The parameter δ can be chosen by cross validation. A more detailed discussion about the selection of δ is given in the conclusion section.

[9] This can easily be verified in a linear regression model with known variance, which we discuss in Appendix A.2. In this model the Maximum Likelihood estimator of the slope parameters β is the Ordinary Least Squares estimator. We show that under minimal regularity conditions

$$\widehat{\beta}_{DT} \xrightarrow{p} (1-\delta)\beta_{\text{OLS}},$$

where β_{OLS} denotes the probability limit of the Ordinary Least Squares estimator of β. Relatedly, Farrell, Liang, and Misra (2020)—who study deep neural networks and their use in semiparametric inference—report that their numerical exploration of dropout increased bias and interval length compared to nonregularized models.

desirable for estimation, the choice of dropout noise in (3) remains quite arbitrary.

The main contribution of this paper is to provide a novel decision-theoretic foundation for the use of dropout training. We will argue there is a natural two-player, zero-sum game between a decision maker (statistician) and an adversary (nature) in which dropout training emerges naturally as a minimax solution. In this game, droput noise turns out to be nature's least favorable distribution, and dropout training becomes the statistician's optimal action. The framework we use is known in the stochastic optimization literature as Distributionally Robust Optimization and we describe it very generally in the next section.

3 Problem Setup

Consider a general problem where there is a multivariate predictor $X \in \mathbb{R}^d$ and a scalar outcome variable $Y \in \mathbb{R}$. A Distributionally Robust Optimization problem is a simultaneous two-player zero sum game between a decision maker (statistician) and an adversary (nature).[10] In this section we describe the action space for each player, their strategies, and the payoff function.

Actions and Payoff: The statistician's action space consists of vectors $\theta \in \Theta$. The ranking of the statistician's actions is contingent on the realization of (X, Y). This is captured by a real-valued loss function

$$\ell(X, Y, \theta).$$

We assume that the statistician is called to choose an action before observing the realization of (X, Y). If the statistician knew the distribution of (X, Y)—which we denote by $\mathbb{Q}$—the statistician's preferred choice of θ would be

$$\inf_{\theta \in \Theta} \mathbb{E}_{\mathbb{Q}} \left[\ell(X, Y, \theta) \right]. \tag{7}$$

Instead of assuming that the distribution $\mathbb{Q}$ is exogenously determined, we think of the distribution $\mathbb{Q}$ as being chosen by nature. Thus, nature's action space consists of a set of probability distributions denoted as $\mathcal{U}$. We refer to this set as the *distributionally uncertainty*

[10] Adversarial and/or distributionally robust formulations of decision problems are ubiquitous in economics. A seminal reference is the robust inventory control problem of Scarf (1958). More recently, Hansen and Sargent (2001) popularized the use of distributionally robust optimization problems in macroeconomics by establishing a connection between the maximin expected utility of Gilboa and Schmeidler (1989) and the robust-control theory of Dupuis, James, and Petersen (2000). In their set-up distributionally robust optimization arises naturally as a consequence of model uncertainty. Recent references describing the use of distributionally robust stochastic programs (as those considered in this paper) are Delage and Ye (2010) and Shapiro (2017). Christensen and Connault (2019) introduced distributionally robust optimization to econometrics in order to characterize the sensitivity of counterfactual analysis with respect to distributional assumptions in a class of structural models.

set. If nature knew the action selected by the statistician, nature's preferred action would be

$$\sup_{\mathbb{Q}\in\mathcal{U}} \mathbb{E}_{\mathbb{Q}}[\ell(X, Y, \theta)]. \tag{8}$$

Strategies and Solution: The choice of θ and $\mathbb{Q}$ are assumed to happen simultaneously. A statistician's strategy for this game consists of a choice of θ. Likewise, nature's strategy for this game consists of a choice of $\mathbb{Q}$.

A *Nash equilibrium* for this game is a pair $(\theta^\star, \mathbb{Q}^\star)$ such that: a) given $\mathbb{Q}^\star$, the parameter $\theta^\star$ solves (7) and b) given $\theta^\star$, the distribution $\mathbb{Q}^\star$ solves (8).

The *minimax solution* for this game is a pair $(\theta^*, \mathbb{Q}^*)$ that solves

$$\inf_{\theta\in\Theta} \sup_{\mathbb{Q}\in\mathcal{U}} \mathbb{E}_{\mathbb{Q}}[\ell(X, Y, \theta)], \tag{9}$$

while the *maximin* solution is based on the program

$$\sup_{\mathbb{Q}\in\mathcal{U}} \inf_{\theta\in\Theta} \mathbb{E}_{\mathbb{Q}}[\ell(X, Y, \theta)]. \tag{10}$$

If $\mathbb{Q}^*$ solves (10), we say that $\mathbb{Q}^*$ is *nature's least favorable distribution*. The mathematical program in (9) is typically referred to as a Distributionally Robust Optimization problem.

4 Dropout Training is Distributionally Robust Optimal

The previous section provided a general description of a Distributionally Robust Optimization problem. We specialize the general framework of Section 3 by imposing two restrictions. First, we use the negative log-likelihood of Generalized Linear Models (McCullagh and Nelder, 1989) as a loss function for the statistician. Second, we define nature's uncertainty set (i.e., the possible data distributions that nature can take) using the multiplicative errors-in-variables model of Hwang (1986).

4.1 Statistician's Payoff

We define the loss function for the statistician to be the negative of the logarithm of the likelihood in (1):

$$\ell(X, Y, \theta) = -\ln h(Y, \phi) + (\Psi(\beta^\top X) - Y(\beta^\top X))/a(\phi), \tag{11}$$

where $\theta \equiv (\beta^\top, \phi^\top)^\top \in \Theta$. Equation (11) defines the statistician's objective and its set of actions.

4.2 Nature's Distributionally Uncertainty Set

We now define the possible distributions that nature can choose.

We start out by letting $\mathbb{Q}_0$ denote some benchmark distribution over (X, Y). This distribution need not correspond to that induced by a Generalized Linear Model. In other words, our framework allows for the statistician's model to be misspecified.

Next, we define nature's action space by considering perturbations of $\mathbb{Q}_0$. Although there are different ways of doing this—for example, by using either f-divergences (such as the Kullback-Leibler) or the optimal transport distance to define a neighborhood around $\mathbb{Q}_0$—we herein use a nonparametric multiplicative errors-in-variables model as in Hwang (1986).

The idea is to allow nature to independently introduce measurement error to the covariates, using multiplicative noise. Let $\xi \equiv (\xi_1, \ldots, \xi_d)$ be defined as a d-dimensional vector of random variables that are independent of (X, Y). We perturb the distribution $\mathbb{Q}_0$ by considering the transformation

$$(X, Y) \mapsto (X_1\xi_1, \ldots, X_d\xi_d, Y).$$

As a result, each covariate X_j is distorted in a multiplicative fashion by ξ_j. We often abbreviate $(X_1\xi_1, \ldots, X_d\xi_d)$ by $X \odot \xi$, where $\odot$ is the element-wise multiplication.

We restrict the distribution of ξ in the following way. First, for a parameter $\delta_j \in (0, 1)$, we define $\mathcal{Q}_j(\delta_j)$ to be the set of distributions that are supported on the interval $[0, 1/(1-\delta_j)]$ and that have mean equal to 1. That is:

$$\mathcal{Q}_j(\delta_j) \equiv \left\{\mathbb{Q}_j : \mathbb{Q}_j \text{ is a probability distribution on } \mathbb{R},\ \mathbb{Q}_j([0, (1-\delta_j)^{-1}]) = 1,\ \mathbb{E}_{\mathbb{Q}_j}[\xi_j] = 1\right\}. \tag{12}$$

This set of distributions is popular in DRO problems due to its simplicity and tractability (Wiesemann, Kuhn, and Sim, 2014). From the perspective of an errors-in-variables model, these distributions are also attractive because they preserve the expected value of the covariates, assuming that X_j and ξ_j are drawn independently.

Consider now the joint random vector $(X, Y, \xi) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d$. For a vector of constants $\delta = (\delta_1, \ldots, \delta_d)$, $\delta_j \in (0, 1)$ consider the joint distributions over (X, Y, ξ) defined by

$$\mathcal{U}(\mathbb{Q}_0, \delta) = \left\{\mathbb{Q}_0 \otimes \mathbb{Q}_1 \otimes \ldots \otimes \mathbb{Q}_d : \mathbb{Q}_j \in \mathcal{Q}_j(\delta_j)\ \forall j = 1, \ldots, d\right\}, \tag{13}$$

where $\otimes$ is used to denote the product measure (meaning that the joint distribution is the product of the independent marginals $\mathbb{Q}_j$, $j = 0, \ldots, d$). Thus, in the game we consider $\mathcal{U}(\mathbb{Q}_0, \delta)$ is nature's action space or *nature's distributionally uncertainty set*. We will make only one assumption about the reference distribution $\mathbb{Q}_0$:

Assumption 1. $\mathbb{E}_{\mathbb{Q}}[\ell(X \odot \xi, Y, \theta)] < \infty$ *for any* $\mathbb{Q} \in \mathcal{U}(\mathbb{Q}_0, \delta)$*, any* $\theta \in \Theta$*, and any vector* $\delta \in (0, 1)^d$*.*

This assumption implies a minimal regularity condition to guarantee that the expected loss is well-defined for both the statistician and nature.

4.3 Dropout Training is DRO

We now present the main result of this section.

Theorem 1. *Consider the two-player zero sum game where the statistician has the loss function in* (11) *and nature has the action space in* (13) *for some reference distribution* $\mathbb{Q}_0$ *and a vector* $\delta \in (0, 1)^d$*. If Assumption 1 is satisfied, then the minimax solution of the two-player zero sum game defined by* (11)-(13)

$$\inf_{\theta \in \Theta} \sup_{\mathbb{Q} \in \mathcal{U}(\mathbb{Q}_0, \delta)} \mathbb{E}_{\mathbb{Q}} \left[\ell(X \odot \xi, Y, \theta) \right] \tag{14}$$

is equivalent to

$$\inf_{\theta \in \Theta} \mathbb{E}_{\mathbb{Q}^\star}[\ell(X \odot \xi, Y, \theta)], \tag{15}$$

where $\mathbb{Q}^\star = \mathbb{Q}_0 \otimes \mathbb{Q}_1^\star \otimes \ldots \otimes \mathbb{Q}_d^\star$*, and* $\mathbb{Q}_j^\star = (1 - \delta_j)^{-1} \times Bernoulli(1 - \delta_j)$ *is a scaled Bernoulli distribution for any* $j = 1, \ldots, d$*, i.e., under* $\mathbb{Q}_j^\star$

$$\xi_j = \begin{cases} 0 & \text{with probability } \delta_j, \\ (1 - \delta_j)^{-1} & \text{with probability } (1 - \delta_j). \end{cases} \tag{16}$$

In addition, let $\theta^\star \in \Theta$ *be a solution to* (15)*. Then* $(\theta^\star, \mathbb{Q}^\star)$ *constitutes a Nash equilibrium of the two-player zero sum game defined by* (11) *and* (13) *and* $\mathbb{Q}^\star$ *is nature's least favorable distribution.*

Proof. See Appendix A.1. □

The first part of theorem characterizes the statistician's best response to an adversarial nature that is allowed to corrupt the covariates using a multiplicative errors-in-variables model. From the statistician's perspective, nature's worst-case perturbation of $\mathbb{Q}_0$ is given by $\mathbb{Q}^\star$ in

(16), which is a slight generalization of the dropout noise defined in (3).[11] Under this *worst-case* distribution, nature independently corrupts each of the entries of $X = (X_1, \ldots, X_d)^\top$, by either dropping the j-th component (if $\xi_j = 0$) or replacing it by $X_j/(1-\delta_j)$. Dropout training—which here refers to estimating the parameter θ after adding dropout noise to X—thus becomes the statistician's preferred way of estimating the parameter θ when facing an adversarial nature. This gives a decision-theoretic foundation for the use of dropout training. Note that in order to recover the objective function introduced in (5) (the sample average of the contaminated log-likelihood) it suffices to set the reference measure—$\mathbb{Q}_0$—as the empirical distribution of $\{(x_i, y_i)\}_{i=1}^n$, which satisfies Assumption 1.

We provide now some intuition about how dropout noise becomes nature's worst-case distribution. Algebra shows that, in light of Assumption 1, the expected loss under an arbitrary distribution $\mathbb{Q}$ is finite and can be written as

$$\begin{aligned}\mathbb{E}_{\mathbb{Q}}[\ell(X \odot \xi, Y, \theta)] = &-\mathbb{E}_{\mathbb{Q}_0}\left[\ln h(Y, \phi)\right] \\ &+ \mathbb{E}_{\mathbb{Q}_0}\left[\mathbb{E}_{\mathbb{Q}_1 \otimes \ldots \otimes \mathbb{Q}_d}[(\Psi((\beta \odot X)^\top \xi) - Y((\beta \odot X)^\top \xi))/a(\phi)]\right].\end{aligned}$$

where the first expectation is taken with respect to the reference distribution, and the second one with respect to ξ. For fixed values of (X, Y, θ) we can define

$$A_{(X,Y,\theta)}((\beta \odot X)^\top \xi) \equiv (\Psi((\beta \odot X)^\top \xi) - Y((\beta \odot X)^\top \xi))/a(\phi).$$

Because $\Psi(\cdot)$ has been assumed to be a convex function defined on all of the real line, the function $A_{(X,Y,\theta)}(\cdot)$ inherits these properties. We show in the appendix that for these type of functions

$$\sup\left\{\mathbb{E}_{\mathbb{Q}_1 \otimes \ldots \otimes \mathbb{Q}_d}[A_{(X,Y,\theta)}((\beta \odot X)^\top \xi)] : \mathbb{Q}_j \in \mathcal{Q}_j(\delta_j)\right\} = \mathbb{E}_{\mathbb{Q}_1^\star \otimes \ldots \otimes \mathbb{Q}_d^\star}[A_{(X,Y,\theta)}((\beta \odot X)^\top \xi)], \quad (17)$$

for any θ, and this establishes the equivalence between (14) and (15). The proof of the equality above exploits convexity. In fact, to derive our result we first characterize the worst-case distribution for the expectation of a real-valued convex function (Lemma A.1) and then we generalize this result to functions that depend on ξ only through linear combinations, as $A_{(X,Y,\theta)}(\cdot)$ (Proposition A.2).

How about the Nash Equilibrium of the two-player zero sum game defined by (11) and (13)? The equality in (17) clearly shows that $\mathbb{Q}^\star$ is nature's best response for any $\theta \in \Theta$. If there is a vector $\theta^\star$ that solves the dropout training problem in (15), then this vector is the statistician best's response to nature's choice of $\mathbb{Q}^\star$. Consequently, $(\theta^\star, \mathbb{Q}^\star)$ is a Nash

[11] The slight generalization allows each entry to have a different probability of being dropped out.

equilibrium.

Finally, we discuss the extent to which $\mathbb{Q}^\star$ can be referred to as nature's least favorable distribution, which has been defined as nature's solution to the maximin problem. It is well known that the maximin value of a game is always smaller than its minimax value:[12]

$$\sup_{\mathbb{Q}\in\mathcal{U}(\mathbb{Q}_0,\delta)} \inf_{\theta\in\Theta} \mathbb{E}_{\mathbb{Q}}[\ell(X\odot\xi, Y, \theta)] \leq \inf_{\theta\in\Theta} \sup_{\mathbb{Q}\in\mathcal{U}(\mathbb{Q}_0,\delta)} \mathbb{E}_{\mathbb{Q}}[\ell(X\odot\xi, Y, \theta)].$$

We have shown that the right-hand side of the display above equals (15). Therefore, if there is a $\theta^\star \in \Theta$ that solves such program, then $\mathbb{Q}^\star$ achieves the upper bound to the maximin value of the game. This makes dropout noise nature's least favorable distribution.

Now that we have established that dropout training gives the minimax solution of the DRO game, we discuss the implications of this result regarding the out-of-sample performance of dropout training. Suppose $\mathbb{Q}_0$ is the empirical measure $\widehat{\mathbb{P}}_n$ supported on n training samples $\{(x_i, y_i)\}_{i=1}^n$. The *in-sample* loss of dropout training is

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\mathbb{Q}^\star}[\ell(X\odot\xi, Y, \theta^\star)|X = x_i, Y = y_i]. \tag{18}$$

A typical concern with estimation procedures is whether their performance in a specific sample translates to good performance out of sample. In our context, the out-of-sample performance of dropout training can be thought of as the expected loss that would arise for some other data distribution $\tilde{\mathbb{Q}}_0$ over (X, Y) at the parameter estimated via dropout training:

$$\mathbb{E}_{\tilde{\mathbb{Q}}_0}[\ell(X, Y, \theta^\star)].$$

The minimaxity of dropout training shows that for any distribution $\tilde{\mathbb{Q}}_0$ over (X, Y) that can be obtained from $\widehat{\mathbb{P}}_n$ by perturbing covariates with mean-one independent multiplicative error $\xi_j \in [0, (1-\delta_j)^{-1}]$ we have

$$\mathbb{E}_{\tilde{\mathbb{Q}}_0}[\ell(X, Y, \theta^\star)] \leq \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\mathbb{Q}^\star}[\ell(X\odot\xi, Y, \theta^\star)|X = x_i, Y = y_i].$$

This means that the out-of-sample loss will be upper-bounded by the in-sample loss. Thus, our results give a concrete result about the class of distributions for which dropout training

[12] This follows from the fact that for any $\mathbb{Q} \in \mathcal{U}(\mathbb{Q}_0, \delta)$:

$$\inf_{\theta\in\Theta} \mathbb{E}_{\mathbb{Q}}[\ell(X\odot\xi, Y, \theta)] \leq \mathbb{E}_{\mathbb{Q}}[\ell(X\odot\xi, Y, \theta)] \leq \sup_{\mathbb{Q}\in\mathcal{U}(\mathbb{Q}_0,\delta)} \mathbb{E}_{\mathbb{Q}}[\ell(X\odot\xi, Y, \theta)].$$

See also the discussion of the minimax theorem in Ferguson (1967) p. 81.

estimation "generalizes" well.

5 An Algorithm for Dropout Training

Throughout this section, we consider the case in which $\mathbb{Q}_0$ is set to the empirical measure $\widehat{\mathbb{P}}_n$ supported on n training samples $\{(x_i, y_i)\}_{i=1}^n$. Our goal is to suggest an algorithm for solving the dropout training problem

$$\inf_{\theta \in \Theta} \mathbb{E}_{\mathbb{Q}^\star}[\ell(X \odot \xi, Y, \theta)],$$

where $\mathbb{Q}^\star$ is $\mathbb{Q}^\star = \widehat{\mathbb{P}}_n \otimes \mathbb{Q}_1^\star \otimes \ldots \otimes \mathbb{Q}_d^\star$ and $\mathbb{Q}_j^\star$, $j = 1, \ldots, d$ is the dropout noise distribution defined in (16). We will use $\theta_n^\star$ to denote the solution of the dropout training problem above. It will sometimes be convenient to write the dropout training problem as

$$\min_\theta \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{\mathbb{Q}^\star}[\ell(X \odot \xi, Y, \theta) \mid X = x_i, Y = y_i], \tag{19}$$

which coincides with expression (5). Conditioning on the values of (x_i, y_i) makes it clear that the expectation is computed over the d-dimensional vector ξ. We now briefly describe three common approaches to implement dropout training and we discuss some of its limitations.

5.1 Naive Dropout Training

Because $\mathbb{Q}_j^\star$ places mass on only two points, namely 0 and $(1 - \delta_j)^{-1}$, the support of the joint distribution $\mathbb{Q}_1^\star \otimes \ldots \otimes \mathbb{Q}_d^\star$ has cardinality 2^d. Thus, a naive approach to solve the dropout training problem in (19) is to expand the objective function as a sum with $n \cdot 2^d$ terms and apply a tailored gradient descent algorithm to the resulting optimization problem. Unfortunately, this approach is computationally demanding because the number of individual terms in the objective function grows exponentially with the dimension d of the features.

5.2 Dropout Training via Stochastic Gradient Descent

Another method to solve the dropout training problem in (15) is by stochastic gradient descent (henceforth, SGD). This gives us the commonly used dropout training algorithm. For the sake of comparison, we provide concrete details about this algorithm below.

Given a current estimate $\widehat{\theta}$, we compute an unbiased estimate of the gradient to the objective function of (15), and move in the direction of the negative gradient with a suitable

step size. Since $\mathbb{Q}^\star$ is discrete, the expectation under $\mathbb{Q}^\star$ can be written as a finite sum and by differentiating under the expectation, we have

$$\nabla_\theta \mathbb{E}_{\mathbb{Q}^\star}[\ell(X \odot \xi, Y, \widehat{\theta})] = \mathbb{E}_{\mathbb{Q}^\star}\left[\nabla_\theta \ell(X \odot \xi, Y, \widehat{\theta})\right]. \tag{20}$$

The standard SGD algorithm uses a naive Monte Carlo estimator as an estimate of the gradient (20), that is, at iterate $k \in \mathbb{N}$ with incumbent solution $\widehat{\theta}^k$,

$$\nabla_\theta \mathbb{E}_{\mathbb{Q}^\star}[\ell(X \odot \xi, Y, \widehat{\theta}^k)] \approx \nabla_\theta \ell(x_k \odot \xi_k, y_k, \widehat{\theta}^k),$$

where (x_k, y_k, ξ_k) is an independent draw from $\mathbb{Q}^\star$.

One drawback of using SGD to solve (15) is that it is not easily parallelizable, and thus its is implementation can be quite slow. Moreover, under strong convexity assumption of the loss function ℓ, SGD only exhibits linear convergence rate (Nemirovski, Juditsky, Lan, and Shapiro, 2009, Section 2.1). By contrast, the gradient descent (GD) enjoys exponential convergence rate (Boyd and Vandenberghe, 2004, Section 9.3.1).

5.3 Naive Monte Carlo approximation for Dropout Training

Consider solving the dropout training problem in (19) using a naive Monte Carlo approximation. Instead of using 2^d terms to compute

$$\mathbb{E}_{\mathbb{Q}^\star}[\ell(X \odot \xi, Y, \theta) \mid X = x_i, Y = y_i],$$

we approximate this expectation by taking a large number of K i.i.d. draws $\{\xi_i^k\}_{k=1}^K$, $\xi_i^k \in \mathbb{R}^d$, according to the distribution $\mathbb{Q}_1^* \otimes \ldots \otimes \mathbb{Q}_d^*$. When d is large this approximation is computationally cheaper than the naive dropout training procedure described above, provided that $K \ll 2^d$.

Thus, the naive Monte Carlo approximation of the dropout training problem is

$$\min_{\theta \in \Theta} \ \frac{1}{n} \sum_{i=1}^{n} \left[\frac{1}{K} \sum_{k=1}^{K} \ell(x_i \odot \xi_i^k, y_i, \theta) \right], \tag{21}$$

where the random vectors ξ_i^k are sampled independently—over both k and i—using the distribution $\mathbb{Q}_1^\star \otimes \ldots \otimes \mathbb{Q}_d^\star$.

Relative to the solution of the dropout training problem—which we denoted as $\theta_n^\star$—the minimizer of (21) is consistent and asymptotically normal as $K \to \infty$. This follows by standard arguments; for example, those in Shapiro et al. (2014, Section 5.1). There are,

however, two problems that arise when using (21) as a surrogate for the dropout training problem.

First, the solution to (21) is a biased estimator for $\theta_n^\star$. This means that if we average the solution of (21) over the $K \cdot n$ different values of ξ_i^k, the average solution need not equal $\theta_n^\star$. Second, implementing (21) requires a choice of K and, to the best of our knowledge, there is no off-the-shelf procedure for picking this number.

5.4 Unbiased Multi-level Monte Carlo Approximation for Dropout Training

To address these two issues, we apply some recent techniques suggested in Blanchet et al. (2019a) that we refer to as *Unbiased Multi-level Monte Carlo Approximations*.[13] Before providing a detailed presentation of the algorithm, we provide a heuristic description.

To this end, let $\widehat{\theta}_n^\star(K)$ denote the *level* K solution of the problem in (21); that is, the solution based on K draws. Define the random variable

$$\Delta_K \equiv \widehat{\theta}_n^\star(K) - \widehat{\theta}_n^\star(K-1).$$

and, for simplicity, assume $\widehat{\theta}_n^\star(0)$ is defined to equal a vector of zeros. Under regularity conditions

$$\sum_{K=1}^{\infty} \mathbb{E}[\Delta_K] = \lim_{K\to\infty} \mathbb{E}[\widehat{\theta}_n^\star(K)] = \theta_n^\star.$$

Consider now picking K^* at random from some discrete distribution supported on the natural numbers. Let $p(\cdot)$ denote the probability mass function of such distribution and consider a Monte Carlo approximation scheme in which—after drawing K^*—we sample $K^* \cdot n$ different random vectors ξ_i^k according to $\mathbb{Q}_1^\star \otimes \ldots \otimes \mathbb{Q}_d^\star$. The estimator

$$Z(K^*) \equiv \frac{\Delta_{K^*}}{p(K^*)}$$

has two sources of randomness. Firstly, the random choice of K^* and, secondly, the random draws ξ_i^k. Averaging over both yields

$$\mathbb{E}[Z(K^*)] = \sum_{K=1}^{\infty} \mathbb{E}[Z(K^*)|K^* = K] \cdot p(K) = \sum_{K=1}^{\infty} (\mathbb{E}[\Delta_K]/p(K)) \cdot p(K) = \theta_n^\star.$$

[13] Multi-level Monte Carlo methods (Giles, 2008, 2015) refer to a set of recent techniques for approximating the expectation of random variables. The adjective "multi-level" is used to emphasize the fact that random samples of different *levels* of accuracy are used in the approximation.

Thus, by taking into account the randomness in the selection of K, we have managed to provide a rule for deciding the number of draws (specifically, our recommendation is to pick K^* at random) and at the same time we have removed the bias of naive Monte Carlo approximations. Of course, formalizing these heuristic arguments requires an appropriate choice of $p(\cdot)$ and also of Δ_K.[14]

One concern with our suggested implementation is that the expected computational cost of $Z(K^*)$ could be infinitely large. Define the computational cost simply as the number of random draws that are required to obtain $Z(K^*)$. In the construction we have described above, we need $K^* \cdot n$ draws for the construction of the estimator. Thus, the average cost is

$$\mathbb{E}[K^* \cdot n] = n \sum_{K=1}^{\infty} K \cdot p(K)$$

which, under mild integrability conditions on $p(\cdot)$, will be finite.[15]

Algorithm for the Unbiased Multilevel Monte Carlo: We now present the algorithm that will be used to solve the dropout training problem. We present a parallelized version of it using L processors, but the suggested algorithm works even when $L = 1$. Parallel computing reduces the variance of the estimator, and our suggestion is to use as many processors as available in one run.

Fix an integer $m_0 \in \mathbb{N}$ such that $2^{m_0+1} \ll 2^d$. For each processor $l = 1, \ldots, L$ we consider the following steps.

i) Take a random (integer) draw, m_l^*, from a geometric distribution with parameter $r > 1/2$.

ii) Given m_l^*, take $2^{K_l^*+1}$ i.i.d. draws from the d-dimensional vector $\xi_i \sim \mathbb{Q}_1^\star \otimes \ldots \otimes \mathbb{Q}_d^\star$. where

$$K_l^* \equiv m_0 + m_l^*. \tag{22}$$

Repeat this step independently for each $i = 1, \ldots, n$.

[14] As we will discuss below, our choice of $p(\cdot)$ will be a geometric distribution, and we will define Δ based on the solution of the optimization problem for samples of size 2^{K+1} and 2^K.

[15] For example, if $p(\cdot)$ is selected as a geometric distribution with parameter r, the expected computational cost will be $n(1-r)/r$. Even if the computational cost of evaluating $Z(K^*)$ increases exponentially in K and takes the form $c \cdot 2^K$, the expected computational cost will be

$$\sum_{K=1}^{\infty} cr(2(1-r))^K = cr(1/2(1-r)),$$

provided $2(1-r) < 1$, or equivalently, $r > 1/2$. Constraining the variance requires then imposing $r < 3/4$. Ultimately, optimizing the product of computational cost and variance leads to the optimal selection $r = 1 - 2^{-3/2}$.

iii) Solve the problem in (21) using the first 2^{m_0} i.i.d. draws of ξ_i for each i. Let θ_{l,m_0} denote a minimizer.

iv) Denote by $\widehat{\theta}_n^\star(2^{K_l^*+1})$, $\widehat{\theta}_n^O(2^{K_l^*})$, and $\widehat{\theta}_n^E(2^{K_l^*})$ any solution to the following optimization problems (all of which are based on sample average approximations as (21)):

$$\widehat{\theta}_n^\star(2^{K_l^*+1}) \in \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^n \left(\frac{1}{2^{K_l^*+1}} \sum_{k=1}^{2^{K_l^*+1}} \ell(x_i \odot \xi_i^k, y_i, \theta) \right),$$

$$\widehat{\theta}_n^O(2^{K_l^*}) \in \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^n \left(\frac{1}{2^{K_l^*}} \sum_{k=1}^{2^{K_l^*}} \ell(x_i \odot \xi_i^{2k-1}, y_i, \theta) \right),$$

$$\widehat{\theta}_n^E(2^{K_l^*}) \in \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^n \left(\frac{1}{2^{K_l^*}} \sum_{k=1}^{2^{K_l^*}} \ell(x_i \odot \xi_i^{2k}, y_i, \theta) \right).$$

Intuitively, $\widehat{\theta}_n^O$ and $\widehat{\theta}_n^E$ denote the solutions to problem (21) but using a sample of size 2^{K_l} with only *odd* and *even* indices, respectively.

v) Define

$$\bar{\Delta}_{K_l^*} \equiv \widehat{\theta}_n^\star(2^{K_l^*+1}) - \frac{1}{2}(\widehat{\theta}_n^O(2^{K_l^*}) + \widehat{\theta}_n^E(2^{K_l^*}))$$

and let

$$Z(K_l^*) = \frac{\bar{\Delta}_{K_l^*}}{r(1-r)^{K_l^*-m_0}} + \theta_{l,m_0}.$$

Our recommended estimator is

$$\frac{1}{L}\sum_{l=1}^L Z(K_l^*). \tag{23}$$

We now show that the suggested algorithm gives an estimator with desirable properties. We do so under the following regularity assumptions.

Assumption 2. *Suppose that the parameter space Θ is compact. Suppose in addition that the optimal solution $\theta_n^\star$ to the dropout training problem in* (19) *is (globally) unique.*

Assumption 3. *Let $\widehat{\theta}_n^\star(K)$ denote the solution of the problem in* (21) *based on K draws. Suppose that as $K \to \infty$,*

$$\mathbb{E}[\|K^{\frac{1}{2}}(\widehat{\theta}_n^\star(K) - \theta_n^\star)\|_2^4) = O(1),$$

where the expectation is taken over the i.i.d dropout noise distribution used to generate ξ_i^k.

Assumption 4. *Assume that for each* (X, Y, ξ), $\ell(X \odot \xi, Y, \cdot)$ *is thrice continuously differentiable over* Θ *and that*

$$\nabla_{\theta\theta}\mathbb{E}_{\mathbb{Q}^\star}[\ell(X \odot \xi, Y, \theta_n^\star)]$$

is non-singular.

Theorem 2. *Under Assumption 2,* $\mathbb{E}[Z(K_l^*)] = \theta_n^\star$. *The number of random draws required to compute* $Z(K_l^*)$ *is* $n \cdot 2^{K_l^*+1}$ *and thus the expected computational complexity for producing* $Z(K_l^*)$ *equals*

$$\frac{n(2^{m_0+1})r}{2r-1} < n(2^{m_0+1}) \ll n2^d.$$

Suppose, in addition, that $\widehat{\theta}_n^\star(K)$ *is almost surely in the interior of* Θ *for* K *large enough. If Assumptions 3 and 4 hold, and* $r < 3/4$. *Then* $\mathrm{Var}(Z(K_l^*)) < \infty$

Proof. See Appendix A.3 □

Our suggested algorithm has finite expected computational complexity that does not grow exponentially with the dimension d, thus every time we need to obtain $\widehat{\theta}_n^\star(2^{K_l^\star+1})$ we can do so by applying gradient descent. Combined with parallelization, the Unbiased Multi-level Monte Carlo approach produces an unbiased estimator with a variance that can be made arbitrarily small if L is large enough, provided the regularity assumptions that give $\mathrm{Var}(Z(K_l^*)) < \infty$ are satisfied.[16]

6 Numerical Experiment

In this section we present a simple numerical experiment to illustrate the advantage of using the Unbiased Multi-level Monte-Carlo estimator suggested in Section 5.4. We consider the linear regression problem with known variance and we focus on solving the dropout training problem with $\delta_j = \delta = 0.5$ for all j.

Our simulation setting considers a high-dimensional linear regression model with covariate vector having dimension $d = 1000$ and sample size $n = 50$. We pick a known regression

[16]We decided not to analyze the additional regularization effects associated to the different algorithmic implementations of dropout training. This means that our suggested algorithm is recommended on the basis of its ability to implement dropout training without introducing additional bias and with an algorithm that can exploit parallelization. This is consistent with the objective of our paper, which is to provide a decision-theoretic foundation for the use of dropout training and explain why the idea of perturbing the original loss function mitigates overfitting. Wei et al. (2020) have shown that added noise in the stochastic gradient descent implementation of dropout can lead to additional benefits in terms of out-of-sample performance, due to *implicit* regularization imposed by the stochastic gradient descent routine.

coefficient $\beta_0 \in \mathbb{R}^d$ being a vector with all entries equal to 1. With fixed coefficients, we assume the covariate vector follows independent Gaussian, as well as for the regression noise. More specifically, we can get our $n = 50$ observations (x_i, y_i) via

- sampling $x_i \sim \mathcal{N}(0, I_d)$,$i = 1, \ldots, n$,
- sampling $y_i \in \mathbb{R}$ conditional on x_i, where y_i is given by the linear assumption and ε_i are i.i.d. random noise following $\mathcal{N}(0, 10^2)$, for $i = 1, \ldots, n$.

Our simulation setting consider a high-dimension setting (relative low ratio between sample size per dimension $n/d = 0.05$) with high noise to signal ratio (variability on residual noise is high compared to the variability on x_i).

If we set $\mathbb{Q}_0$ to be the empirical distribution of $\{(x_i, y_i)_{i=1}^n\}$, the dropout training problem in the linear regression model is

$$\min_{\beta \in \mathbb{R}^d} \ \mathbb{E}_{\mathbb{Q}^\star} \left[\left(\beta^\top (X \odot \xi) - Y \right)^2 \right].$$

Corollary 1, Appendix A.2 shows that in the linear regression model the dropout training problem can be written as:

$$\min_{\beta \in \mathbb{R}^d} \ \frac{1}{n} \left[(\mathbf{Y} - \mathbf{X}\beta)^\top (\mathbf{Y} - \mathbf{X}\beta) + \frac{\delta}{1-\delta} \beta^\top \mathbf{\Lambda} \beta \right]$$

where $\mathbf{Y} = [y_1, y_2, \ldots, y_n]^\top$, $\mathbf{X} = [x_1, x_2, \ldots, x_n]^\top$ and $\mathbf{\Lambda}$ is the diagonal matrix with its diagonal elements given by the diagonals of $\mathbf{X}^\top \mathbf{X}$. Moreover, there is a closed-form solution for the dropout training problem and it is given by the ridge regression formula:

$$\beta^\star = \left(\mathbf{X}^\top \mathbf{X} + \frac{\delta}{1-\delta} \mathbf{\Lambda} \right)^{-1} \mathbf{X}^\top \mathbf{Y}.$$

Since neither our suggested Multi-level Monte Carlo algorithm nor standard SGD (as defined in Section 5.2) uses closed-form formulae for their implementation, we analyze the extent to which these procedures can approximate the parameter β^*. We provide more details of the algorithms as follows. The two algorithms we compare are:

- Standard SGD algorithm with a learning rate 0.0001, and initialization at the origin. Note that however we take batched SGD instead of single-sample SGD introduced in Section 5.2.
- Multi-level Monte Carlo algorithm with the geometric rate $r = 0.6$ and the burn-in period $m_0 = 5$. Note that in each parallel running, we use gradient descent (GD) with

0.01 learning rate and initialization at origin for steps iii) and iv) in Section 5.4.

We run our simulation on a cluster with two Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz processors (with 10 cores each), and a total memory of 128 GB. We fix 60 seconds as a "wall-clock time", so that we terminate the two algorithms after 60 seconds.[17]

We run 1000 independent experiments. For each run, we calculate and report the average parameter estimation divergence to $\beta^\star$ and 1-standard deviation error bar for the divergence. We consider difference number of parallelizations (i.e., L in Section 5.4) from 1 to 2400. We cap the run at 2400 due to the saturation of divergence after $\sim$ 2000 parallelizations.

Figure 1 shows the l_2 divergence to the true $\beta^\star$ of the two algorithms, while Figure 2 and Figure 3 show l_∞ and l_1 divergence respectively. We observed that our unbiased estimator outperforms standard SGD algorithm once the number of parallel iterations reaches above some moderate threshold ($\sim$ 1000 here). We provide supporting evidence in Appendix A.4 to argue our choice of learning rate, initialization, and wall-clock time, where our proposed algorithm is robust to any reasonable choices.

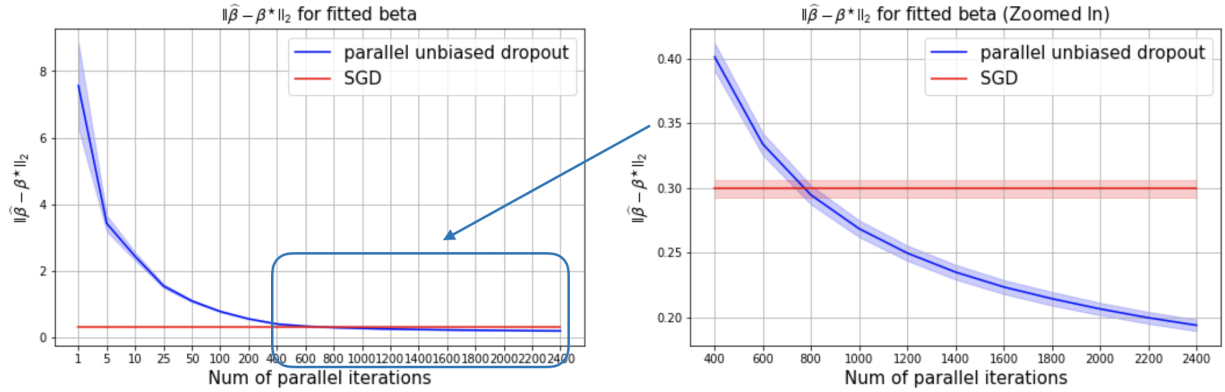

Figure 1: l_2 difference

7 Extensions

In this section we discuss the extent to which the decision-theoretic support for dropout training carries over to Neural Networks.

[17]The parameters for the SGD algorithm are appropriately tuned to achieve good convergence within 60s (see Appendix A.4 for the tuning procedure). However, we do not claim that this choice of parameters is optimal.

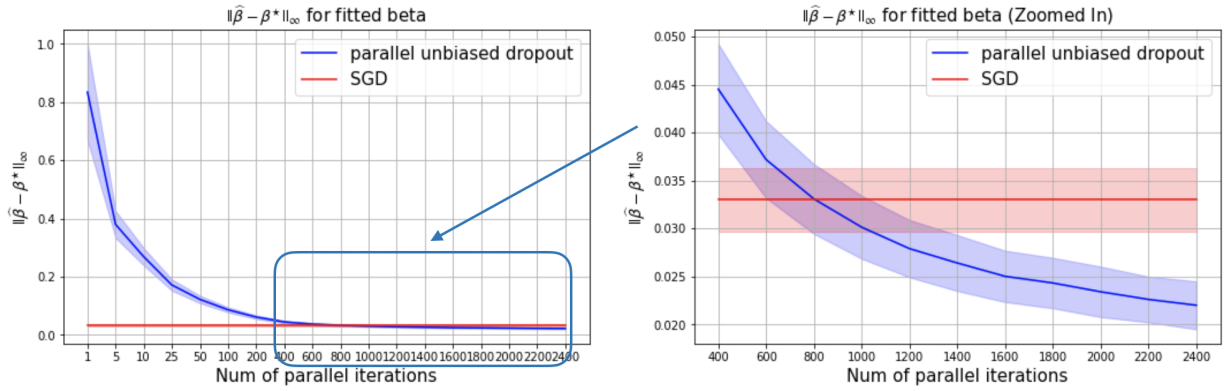

Figure 2: l_∞ difference

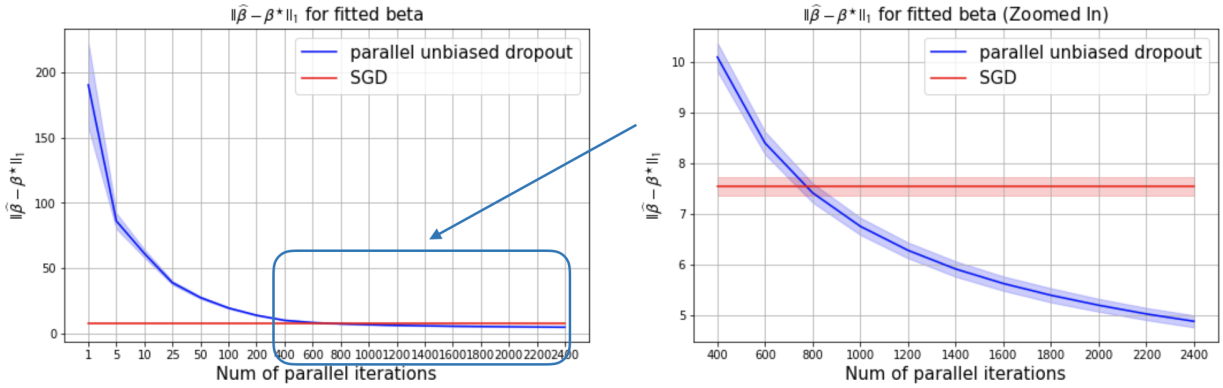

Figure 3: l_1 difference

7.1 Nonlinear Regression with One-hidden-layer Neural Networks

Suppose the scalar response variable Y is generated by the model

$$Y = \Omega_\theta(X) + \varepsilon, \quad \varepsilon | X \sim \mathcal{N}(0, \sigma^2), \tag{24}$$

where $\Omega_\theta(X)$ is a neural network with parameters θ and $X \in \mathbb{R}^d$. This is the nonparametric regression model that has been used recently to study deep neural networks; see Schmidt-Hieber (2017).

In this section, we will assume that $\Omega_\theta(X)$ is a neural network with a *single hidden layer*, a *differentiable activation (squashing) function*, and *linear ouput function*.[18] Although these

[18]A function $h : \mathbb{R} \to [0, 1]$ is a squashing function if it is non-decreasing and if

$$\lim_{r \to \infty} h(r) = 1, \quad \lim_{r \to -\infty} h(r) = 0.$$

See Definition 2.3 in Hornik, Stinchcombe, and White (1989).

types of networks—which will be formally described below—are restrictive compared to the modern deep learning architectures, they can approximate any Borel measurable function from a finite-dimensional space to another, provided the hidden units in the hidden layer are large; see Hornik et al. (1989).

Consider a neural network with K units in the hidden layer, each using input weights $w_k \in \mathbb{R}^d$, $k = 1, \ldots, K$. Denote the activation function in the hidden layer as $h(\cdot)$. Assume the output function is linear with vector of weights $\beta \in \mathbb{R}^K$. Thus, the network under consideration is defined by the function:

$$\Omega_\theta(X) \equiv \beta_1 h(w_1^\top X) + \ldots + \beta_K h(w_K^\top X) = \beta^\top H(X),$$

where $H(X) = (h(w_1^\top X), \ldots, h(w_K^\top X))^\top$. The neural network is parameterized by $\theta \equiv (\beta^\top, w_1^\top, \ldots, w_k^\top)^\top$.

7.1.1 Statistician's Objective Function

We will assume, for simplicity, that σ^2 is known and that it is equal to 1. We will endow the statistician with a quadratic loss function $\ell(X, Y, \theta) = (Y - \Omega_\theta(X))^2$. This corresponds to the negative of the conditional log-likelihood for the model in (24).

7.1.2 Nature's Uncertainty Set

We allow nature to introduce additional noise to the statistician's model. We do this in two steps.

First, we allow nature to distort the distribution of X using multiplicative noise denoted as $\xi(1) \in \mathbb{R}^d$. This is exactly analogous to what we did in the GLM model, where nature was allowed to pick a distribution for the covariates of the form $(X \odot \xi(1))$. Using the jargon of neural networks, we allow nature to contaminate the *input layer* with independent and multiplicative noise.

Second, we also allow nature to contaminate *each of the hidden units* with multiplicative noise $\xi(2) \in \mathbb{R}^K$. That is, nature is also allowed to pick a vector $\xi(2) = (\xi(2)_1, \ldots, \xi(2)_K)^\top$, independently of $\xi(1) \in \mathbb{R}^d$, to distort the each of the K units in the hidden layer as

$$H(X) \odot \xi(2) \equiv (h(w_1^\top X)\xi(2)_1, \ldots, h(w_K^\top X)\xi(2)_K)^\top.$$

7.1.3 Minimax Solution

The minimax solution of the DRO game is given by

$$\inf_{\theta} \sup_{\mathbb{Q}} \mathbb{E}_{\mathbb{Q}} \left[\left(Y - \beta^\top [H(X \odot \xi(1)) \odot \xi(2)] \right)^2 \right], \tag{25}$$

where $\mathbb{Q}$ now refers to the joint distribution of $(X, Y, \xi(1), \xi(2))$. We continue working with the assumption that $\xi \equiv (\xi(1)^\top, \xi(2)^\top)^\top$ has independent marginals and that it is independent of (X, Y).

We would like to solve for the worst-case distributions of the random vectors $\xi(1)$ and $\xi(2)$, assuming that both of these satisfy restrictions analogous to (13). The solution for the distribution of $\xi(2)$ is straightforward, as it suffices to define

$$\tilde{X} \equiv H(X \odot \xi(1)),$$

and view (25) as a the DRO problem in a linear regression model where the data is $(\tilde{X}, Y)$ and $\xi(2) \in \mathbb{R}^K$ is simply the multiplicative noise that transforms the covariates into $(\tilde{X} \odot \xi(2))$.

The worst-case choice of the multiplicative error for the inputs is more difficult to characterize. Below, we provide an heuristic argument that suggests that dropout noise might approximate the worst-case choice under some conditions.

Let $\xi(1)_j$ denote the j-th coordinate of $\xi(1)_j$. Suppose that the distribution of this random variable places most of its mass on the interval $[1-\epsilon, 1+\epsilon]$.[19] This allows us to 'linearize' the output of each of the hidden units around the output corresponding to unperturbed inputs:

$$\begin{aligned} h(w_k^\top (X \odot \xi(1))) &= h(w_k^\top (X \odot (\xi(1) - \mathbf{1})) + w_k^\top X) \\ &\approx h(w_k^\top X) + \left(\dot{h}(w_k^\top X) \cdot (w_k \odot X)^\top (\xi(1) - \mathbf{1}) \right). \end{aligned}$$

In the notation above $\mathbf{1}$ denotes the d-dimensional vector of 1's. For the sake of exposition, ignore the approximation error in the linearization above. If we fix $(X, Y, \xi(2))$, then the worst-case choice for the distribution of $\xi(1)$, denoted $\mathbb{Q}(1)$ maximizes

$$\mathbb{E}_{\mathbb{Q}(1)} \left[\left(\sum_{k=1}^{K} \beta_k \cdot \xi(2)_k \cdot \left[h(w_k^\top X) + \left(\dot{h}(w_k^\top X) \cdot \sum_{j=1}^{d} w_{k,j} \cdot X_j \cdot (\xi(1)_j - 1) \right) \right] \right)^2 \right].$$

among all distributions with independent marginals for which $\mathbb{E}_{\mathbb{Q}(1)}[\xi(1)_j] = 1$ for all $j =$

[19] This is compatible with dropout noise for which δ is very close to zero.

$1, \ldots, d$. Algebra shows that such maximization problem is equivalent to maximizing

$$\mathbb{E}_{\mathbb{Q}(1)}\left[\left(\sum_{k=1}^{K}\beta_k\cdot\xi(2)_k\cdot\dot{h}(w_k^\top X)\cdot\left[\sum_{j=1}^{d}w_{k,j}\cdot X_j\cdot(\xi(1)_j-1)\right]\right)^2\right], \tag{26}$$

which in turn can be written as

$$\mathbb{E}_{\mathbb{Q}(1)}\left[\left(a^\top(\xi(1)-\mathbf{1})\right)^2\right],$$

for an appropriate choice of a vector $a\in\mathbb{R}^d$ that depends on $(\beta,\xi(2),h,\dot{h},w,X)$.[20] Proposition 1 in the Appendix shows that the solution to this problem is dropout noise.

8 Concluding Remarks

This paper aimed to contribute to the gainful connection between econometrics and machine learning. As pointed out recently by (Athey and Imbens, 2019), the literature on machine learning is growing rapidly, and its algorithms and methods have much to offer to the field of economics: "*being familiar with these [machine learning] methods will allow researchers to do more sophisticated empirical work, and to communicate more effectively with researchers in other fields*".[21]

In this paper we studied *dropout training*, an increasingly popular estimation method in machine learning. Dropout training is a fundamental part of the modern machine learning techniques for training very deep networks ((Goodfellow et al., 2016)).

Our main result (Theorem 1) established a novel decision-theoretic foundation for the use of dropout training. We showed that this method, when applied to Generalized Linear Models, can be viewed as the minimax solution to an adversarial two-player, zero-sum game between a statistician and nature. The framework used in this paper is known in the stochastic optimization literature ((Shapiro et al., 2014)) as a Distributionally Robust Optimization (DRO) problem.

[20] From (26) we can see that

$$a\equiv\sum_{k=1}^{K}\beta_k\cdot\xi(2)_k\cdot\dot{h}(w_k^\top X)\cdot(w_k\odot X)$$

[21] Other leading figures in econometrics have expressed similar, albeit more nuanced, opinions. See for example, the panel session organized by the Journal of Econometrics in the 2020 meetings of the American Economic Association "Econometrics in the 21st Century, Challenges and Opportunities" avaialable at https://sites.google.com/view/journalofeconometrics/home

Our minimaxity result showed, by construction, that dropout training indeed provides out-of-sample performance guarantees for distributions that arise from multiplicative perturbations of in-sample data. Our result thus qualified quite explicitly the ability of dropout training to enhance out-of-sample performance, which is one of the reasons often invoked to use the dropout method.

In addition to our theoretical result, we also suggested a new stochastic optimization implementation of dropout training. We borrowed ideas from the Multi-level Monte Carlo literature—in particular from the work of (Blanchet et al., 2019a)—to suggest an unbiased dropout training routine that is easily parallelizable and that has a much smaller computational cost compared to naive dropout training methods when the number of features is large (Theorem 2). Crucially, we showed that under some regularity conditions our estimator has finite variance (which means there are also theoretical, and not just practical, gains from parallelization).

We also discussed the extent to which our theoretical results extended to Neural Networks (in particular, to the universal approximators in (Hornik et al., 1989) consisting of a single-hidden layer and a squashing activation function). We hope that our results serve as a foundation to understand the benefits of dropout in Neural Networks with richer architectures.

An important issue that remains unsolved is the choice of δ, the (hyper)parameter that controls the dropout probability. The DRO interpretation of dropout training brings different views to this problem, all of which provide alternatives to cross-validation.

One approach consists in choosing δ so that true data generating process belongs to nature's choice set with some prespecified probability. This approach, which is often advocated in the literature in machine learning and robustness (Hansen and Sargent, 2008), often leads to a very pessimistic selection of δ, simply because the criterion is not informed at all by the loss function defining the decision problem.

Another approach involves using generalization bounds leading to finite sample guarantees; see, for instance a summary of this discussion in Section 6.2 of Rahimian and Mehrotra (2019). This method, while appealing, often requires either distributions with compact support or strong control on the tails of the underlying distributions. Also, often, the bounds depend on constants that may be too pessimistic or difficult to compute.

Finally, there is a recent method introduced in Blanchet, Kang, and Murthy (2019b) for the case in which nature's choice set is defined in terms of Wasserstein's distance around the data's empirical distribution. The idea therein is that—for a fixed δ—every distribution that belongs to nature's choice set corresponds to an optimal parameter choice for the statistician. Thus, one can collect each and every of the statistician's optimal choices associated to each

distribution in nature's uncertainty set, and treat the resulting region as a confidence set for the true parameter. This confidence set is increasing (in the sense of nested confidence regions) as δ increases. The goal is then to minimize δ subject to a desired level of coverage in the underlying parameter to estimate. This leads to a data-driven choice of δ that is explicitly linked to the statistician's decision problem.

References

Athey, S. and G. W. Imbens (2019): "Machine learning methods that economists should know about," *Annual Review of Economics*, 11, 685–725.

Bahr, B. V. (1965): "On the convergence of moments in the central limit theorem," *Annals of Mathematical Statistics*, 36, 808–818.

Bishop, C. M. (1995): "Training with noise is equivalent to Tikhonov regularization," *Neural Computation*, 7, 108–116.

Blanchet, J., P. Glynn, and Y. Pei (2019a): "Unbiased multilevel Monte Carlo: Stochastic optimization, steady-state simulation, quantiles, and other applications," *arXiv preprint arXiv:1904.09929*.

Blanchet, J., Y. Kang, and K. Murthy (2019b): "Robust Wasserstein Profile Inference and Applications to Machine Learning," *Journal of Applied Probability*, 56, 830–857.

Boyd, S. and L. Vandenberghe (2004): *Convex Optimization*, USA: Cambridge University Press.

Christensen, T. and B. Connault (2019): "Counterfactual sensitivity and robustness," *arXiv preprint arXiv:1904.00989*.

DasGupta, A. (2008): *Asymptotic Theory of Statistics and Probability*, Springer Verlag.

Delage, E. and Y. Ye (2010): "Distributionally Robust Optimization Under Moment Uncertainty with Application to Data-Driven Problems," *Operations Research*, 58, 595–612.

Draper, D. (1994): "Assessment and propagation of model uncertainty," *Journal of the Royal Statistical Society, Series B*, 56.

Dupuis, P., M. R. James, and I. Petersen (2000): "Robust properties of risk-sensitive control," *Mathematics of Control, Signals and Systems*, 13, 318–332.

Farrell, M. H., T. Liang, and S. Misra (2020): "Deep neural networks for estimation and inference," *Forthcoming at Econometrica*.

Ferguson, T. (1967): *Mathematical Statistics: A Decision Theoretic Approach*, vol. 7, Academic Press New York.

Gilboa, I. and D. Schmeidler (1989): "Maxmin expected utility with non-unique prior," *Journal of Mathematical Economics*, 18, 141–153.

Giles, M. B. (2008): "Multilevel Monte Carlo path simulation," *Operations Research*, 56, 607–617.

——— (2015): "Multilevel Monte Carlo methods," *Acta Numerica*, 24, 259.

Goodfellow, I., Y. Bengio, and A. Courville (2016): *Deep Learning*, MIT Press.

Hansen, L. and T. J. Sargent (2001): "Robust control and model uncertainty," *American Economic Review*, 91, 60–66.

Hansen, L. P. and T. J. Sargent (2008): *Robustness*, Princeton University Press.

Helmbold, D. P. and P. M. Long (2015): "On the inductive bias of dropout," *The Journal of Machine Learning Research*, 16, 3403–3454.

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov (2012): "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*.

Hornik, K., M. Stinchcombe, and H. White (1989): "Multilayer feedforward networks are universal approximators," *Neural Networks*, 2, 359–366.

Hwang, J. T. (1986): "Multiplicative errors-in-variables models with applications to recent data released by the US Department of Energy," *Journal of the American Statistical Association*, 81, 680–688.

Maaten, L., M. Chen, S. Tyree, and K. Weinberger (2013): "Learning with marginalized corrupted features," in *International Conference on Machine Learning*, 410–418.

McCullagh, P. and J. Nelder (1989): *Generalized Linear Models*, Chapman & Hall.

Morgenstern, O. and J. von Neumann (1953): *Theory of Games and Economic Behavior*, Princeton University Press.

Nemirovski, A., A. Juditsky, G. Lan, and A. Shapiro (2009): "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on Optimization*, 19, 1574–1609.

Raftery, A. E., D. Madigan, and J. A. Hoeting (1997): "Bayesian model averaging for linear regression models," *Journal of the American Statistical Association*, 92, 179–191.

Rahimian, H. and S. Mehrotra (2019): "Distributionally robust optimization: A review," *arXiv preprint arXiv:1908.05659*.

Robbins, H. and S. Monro (1951): "A stochastic approximation method," *The Annals of Mathematical Statistics*, 400–407.

Scarf, H. (1958): "A min-max solution of an inventory problem," *Studies in the mathematical theory of inventory and production*, 10, 201–209.

Schmidt-Hieber, J. (2017): "Nonparametric regression using deep neural networks with ReLU activation function," *arXiv preprint arXiv:1708.06633*.

Shapiro, A. (2017): "Distributionally robust stochastic programming," *SIAM Journal on Optimization*, 27, 2258–2275.

Shapiro, A., D. Dentcheva, and A. Ruszczyński (2014): *Lectures on Stochastic Programming: Modeling and Theory*, SIAM.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014): "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, 15, 1929–1958.

Wager, S., S. Wang, and P. S. Liang (2013): "Dropout training as adaptive regularization," in *Advances in Neural Information Processing Systems 26*, 351–359.

Wald, A. (1950): *Statistical Decision Functions*, Oxford, England: Wiley.

Wang, S. and C. Manning (2013): "Fast dropout training," in *Proceedings of the 30th International Conference on Machine Learning*, 118–126.

Wei, C., S. Kakade, and T. Ma (2020): "The Implicit and Explicit Regularization Effects of Dropout," *Pre-proceedings of the International Conference of Machine Learning*.

Wiesemann, W., D. Kuhn, and M. Sim (2014): "Distributionally robust convex optimization," *Operations Research*, 62, 1358–1376.

Zinkevich, M., M. Weimer, L. Li, and A. J. Smola (2010): "Parallelized stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2595–2603.

A Appendix

A.1 Proof of Theorem 1

The proof of Theorem 1 relies on the following two preparatory results.

Lemma 1 (Extremal expectation of a univariate convex function)**.** *For any $-\infty < a < b < +\infty$, let ζ be a random variable in $[a,b]$ with mean $\mu \in [a,b]$. For any function $f : [a,b] \to \mathbb{R}$ convex and continuous, the distribution of ζ that maximizes $\mathbb{E}[f(\zeta)]$ among all distributions over $[a,b]$ with a given mean $\mu \in [a,b]$ is a scaled and shifted Bernoulli distribution, i.e.,*

$$\zeta = \begin{cases} a & \text{with probability } (b-\mu)/(b-a), \\ b & \text{with probability } (\mu-a)/(b-a). \end{cases} \tag{27}$$

Proof of Lemma 1. Let Q^* denote the probability measure induced by the random variable in (27). By definition

$$\mathbb{E}_{Q^*}[f(\zeta)] = \frac{b-\mu}{b-a} f(a) + \frac{\mu-a}{b-a} f(b).$$

Suppose first that $\mu = a$. In this case, Jensen's inequality implies that for any other probability measure Q over $[a,b]$ with mean $\mu = a$,

$$\mathbb{E}_Q[f(\zeta)] \leq f(\mathbb{E}_Q[\zeta]) = f(a) = \mathbb{E}_{Q^*}[f(\zeta)].$$

An analogous result holds if $\mu = b$.

Consider then the case in which $\mu \in (a,b)$. For an arbitrary probability measure Q over $[a,b]$ with mean $\mu \in (a,b)$, we have

$$\int_{[a,b]} f(\zeta) \mathrm{d}Q = \int_{[a,b]} f\big(a\frac{b-\zeta}{b-a} + b\frac{\zeta-a}{b-a}\big) \mathrm{d}Q \leq \int_{[a,b]} \big(\frac{b-\zeta}{b-a} f(a) + \frac{\zeta-a}{b-a} f(b)\big) \mathrm{d}Q,$$

where the inequality follows from the convexity of f. By the linearity of the integral operator and the fact that $\int_{[a,b]} \zeta \mathrm{d}Q = \mu$, we find

$$\int_{[a,b]} f(\zeta) \mathrm{d}Q \leq \frac{b-\mu}{b-a} f(a) + \frac{\mu-a}{b-a} f(b).$$

Because the probability measure Q was chosen arbitrarily, this implies that the distribution of ζ in (27) maximizes the expectation of $f(\zeta)$. □

Proposition 1. *Fix a vector of tuning parameters $\delta \in (0,1)^d$. Let $\mathcal{Q}_j(\delta_j)$ be defined as in*

(12). *Suppose that A is a convex and continuous function on $\mathbb{R}$. For any $\theta \in \mathbb{R}^d$, we have*

$$\sup\left\{\mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_d}[A(\theta^\top\xi)] : \mathbb{Q}_j \in \mathcal{Q}_j(\delta_j)\right\} = \mathbb{E}_{\mathbb{Q}_1^\star\otimes\ldots\otimes\mathbb{Q}_d^\star}[A(\theta^\top\xi)], \tag{28}$$

where $\mathbb{Q}_j^\star$ is a scaled Bernoulli distribution of the form $\mathbb{Q}_j^\star = (1-\delta_j)^{-1} \times Bernoulli((1-\delta_j))$ for each $j = 1, \ldots, d$.

Proof of Proposition 1. First note that $\mathbb{Q}_j^\star \in \mathcal{Q}_j(\delta_j)$ for each j, and thus $\mathbb{Q}_1^\star \otimes \ldots \otimes \mathbb{Q}_d^\star$ is a feasible solution to the maximization problem. It suffices to show that for any set of feasible measures $\mathbb{Q}_j \in \mathcal{Q}_j(\delta_j), j = 1, \ldots, d$, we have

$$\mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_d}[A(\theta^\top\xi)] \leq \mathbb{E}_{\mathbb{Q}_1^\star\otimes\ldots\otimes\mathbb{Q}_d^\star}[A(\theta^\top\xi)].$$

Towards this end, pick any $k \in \{1, \ldots, d\}$. By Fubini's theorem, we can write

$$\mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_d}[A(\theta^\top\xi)] = \mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_{k-1}\otimes\mathbb{Q}_{k+1}\otimes\ldots\otimes\mathbb{Q}_d}\mathbb{E}_{\mathbb{Q}_k}[A(\theta^\top\xi)].$$

For any fixed value $(\xi_1, \ldots, \xi_{k-1}, \xi_{k+1}, \ldots, \xi_d)$ the function $\xi_k \mapsto A(\sum_{j\neq k}\theta_j\xi_j + \theta_k\xi_k)$ is convex in the variable ξ_k over the interval $[0, (1-\delta_k)^{-1}]$. Thus by Lemma 1,

$$\mathbb{E}_{\mathbb{Q}_k}[A(\sum_{j\neq k}\theta_j\xi_j + \theta_k\xi_k)] \leq \mathbb{E}_{\mathbb{Q}_k^\star}[A(\sum_{j\neq k}\theta_j\xi_j + \theta_k\xi_k)] \quad \text{for any fixed } (\xi_1, \ldots, \xi_{k-1}, \xi_{k+1}, \ldots, \xi_d).$$

Thus by the monotonicity of the expectation operator,

$$\mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_d}[A(\theta^\top\xi)] \leq \mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_{k-1}\otimes\mathbb{Q}_{k+1}\otimes\ldots\otimes\mathbb{Q}_d}\mathbb{E}_{\mathbb{Q}_k^\star}[A(\theta^\top\xi)] = \mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_{k-1}\otimes\mathbb{Q}_k^\star\otimes\mathbb{Q}_{k+1}\otimes\ldots\otimes\mathbb{Q}_d}[A(\theta^\top\xi)].$$

By cycling through all possible values of $k \in \{1, \ldots, d\}$ we conclude that

$$\mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_d}[A(\theta^\top\xi)] \leq \mathbb{E}_{\mathbb{Q}_1^\star\otimes\ldots\otimes\mathbb{Q}_d^\star}[A(\theta^\top\xi)].$$

Therefore, equation (28) holds. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. Note that for $\mathbb{Q} \in \mathcal{U}(\mathbb{Q}_0, \delta)$, Assumption 1 implies $\mathbb{E}_{\mathbb{Q}}[\ell(X \odot \xi, Y, \theta)]$ is finite for any $\theta \in \Theta$ and any vector $\delta \in (0,1)^d$. Therefore, from Fubini's theorem and the

definition of loss function:

$$\begin{aligned}\mathbb{E}_{\mathbb{Q}}[\ell(X\odot\xi,Y,\theta)] &= \mathbb{E}_{\mathbb{Q}_0}\left[\mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_d}[\ell(X\odot\xi,Y,\theta)]\right]\\ &= \mathbb{E}_{\mathbb{Q}_0}\left[\mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_d}[-\ln h(Y,\phi)+(\Psi(\beta^\top(X\odot\xi))-Y(\beta^\top(X\odot\xi)))/a(\phi)]\right]\\ &= -\mathbb{E}_{\mathbb{Q}_0}\left[\ln h(Y,\phi)\right]\\ &\quad+\mathbb{E}_{\mathbb{Q}_0}\left[\mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_d}[(\Psi(\beta^\top(X\odot\xi))-Y(\beta^\top(X\odot\xi)))/a(\phi)]\right].\end{aligned}$$

Algebra shows that for any β, X and ξ:

$$\beta^\top(X\odot\xi)=(\beta\odot X)^\top\xi.$$

Thus, we can fix the values of (X,Y,θ) and define the function

$$A_{(X,Y,\theta)}((\beta\odot X)^\top\xi)\equiv(\Psi(\beta^\top(X\odot\xi))-Y\beta^\top(X\odot\xi))/a(\phi). \tag{29}$$

Note that $A_{(X,Y,\theta)}$ satisfies the condition of Proposition 1. Therefore

$$\sup\left\{\mathbb{E}_{\mathbb{Q}_1\otimes\ldots\otimes\mathbb{Q}_d}[A_{(X,Y,\theta)}((\beta\odot X)^\top\xi)]:\mathbb{Q}_j\in\mathcal{Q}_j(\delta_j)\right\}=\mathbb{E}_{\mathbb{Q}_1^\star\otimes\ldots\otimes\mathbb{Q}_d^\star}[A_{(X,Y,\theta)}((\beta\odot X)^\top\xi)],$$

for any (X,Y,θ), which completes the proof. □

A.2 Dropout Training in Linear Regression

Corollary 1 (Linear regression with $\phi=1$)**.** *For linear regression with $\ell(x,y,\beta)=(\beta^\top x-y)^2$, we have*

$$\min_{\beta\in\mathbb{R}^d}\max_{\mathbb{Q}\in\mathcal{U}(\widehat{\mathbb{P}}_n,\delta)}\mathbb{E}_{\mathbb{Q}}\left[\left(\beta^\top(X\odot\xi)-Y\right)^2\right]=\min_{\beta\in\mathbb{R}^d}\mathbb{E}_{\mathbb{Q}^\star}\left[\left(\beta^\top(X\odot\xi)-Y\right)^2\right],$$

where $\mathbb{Q}^\star=\widehat{\mathbb{P}}_n\otimes\mathbb{Q}_1^\star\otimes\ldots\otimes\mathbb{Q}_d^\star$ and $\mathbb{Q}_j^\star=(1-\delta)^{-1}\times Bernoulli(1-\delta)$ for each $j=1,\ldots,d$. Moreover,

$$\min_{\beta\in\mathbb{R}^d}\mathbb{E}_{\mathbb{Q}^\star}\left[\left(\beta^\top(X\odot\xi)-Y\right)^2\right]=\min_{\beta\in\mathbb{R}^d}\ \frac{1}{n}\left[(\mathbf{Y}-\mathbf{X}\beta)^\top(\mathbf{Y}-\mathbf{X}\beta)+\frac{\delta}{1-\delta}\beta^\top\mathbf{\Lambda}\beta\right], \tag{30}$$

which implies that the dropout training estimator equals

$$\widehat{\beta}^\star=\left(\mathbf{X}^\top\mathbf{X}+\frac{\delta}{1-\delta}diag(\mathbf{X}^\top\mathbf{X})\right)^{-1}\mathbf{X}^\top\mathbf{Y}.$$

Finally, if

$$\widehat{\beta}_{OLS} \equiv \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y} \xrightarrow{p} \beta_{OLS}$$

and

$$\mathbf{X}^\top \mathbf{X}/n \xrightarrow{p} \Lambda,$$

where Λ *is a diagonal matrix with strictly positive entries then*

$$\widehat{\beta}^\star \xrightarrow{p} (1-\delta)\beta_{OLS}.$$

Proof. The first part of the corollary follows directly from (14) and (15) in our main theorem. For the second part of the corollary, algebra shows that

$$\begin{aligned}
&\min_{\beta \in \mathbb{R}^d} \ \mathbb{E}_{\mathbb{Q}^\star}\left[\left(\beta^\top (X \odot \xi) - Y\right)^2\right] \\
&= \min_{\beta \in \mathbb{R}^d} \ \mathbb{E}_{\widehat{\mathbb{P}}_n}\left[\left(\sum_{j=1}^d \beta_j X_j - Y_j\right)^2\right] + \sum_{j=1}^d \mathbb{E}_{\mathbb{Q}_j^\star}[(\xi_j - 1)^2]\mathbb{E}_{\widehat{\mathbb{P}}_n}[X_j^2]\beta_j^2 \\
&= \min_{\beta \in \mathbb{R}^d} \ \mathbb{E}_{\widehat{\mathbb{P}}_n}\left[\left(\sum_{j=1}^d \beta_j X_j - Y_j\right)^2\right] + \sum_{j=1}^d \frac{\delta}{1-\delta}\mathbb{E}_{\widehat{\mathbb{P}}_n}[X_j^2]\beta_j^2 \\
&= \min_{\beta \in \mathbb{R}^d} \ \frac{1}{n}\left[(\mathbf{Y} - \mathbf{X}\beta)^\top (\mathbf{Y} - \mathbf{X}\beta) + \frac{\delta}{1-\delta}\beta^\top \mathrm{diag}(\mathbf{X}^\top \mathbf{X})\beta\right].
\end{aligned}$$

The necessary first-order conditions of this optimization problem are

$$-\mathbf{X}^\top (\mathbf{Y} - \mathbf{X}\beta) + \frac{\delta}{1-\delta}\mathrm{diag}(\mathbf{X}^\top \mathbf{X})\beta = \mathbf{0}. \tag{31}$$

Solving this linear system of equations yields:

$$\widehat{\beta}^\star = \left(\mathbf{X}^\top \mathbf{X} + \frac{\delta}{1-\delta}\mathrm{diag}(\mathbf{X}^\top \mathbf{X})\right)^{-1} \mathbf{X}^\top \mathbf{Y}.$$

For large enough n

$$\widehat{\beta}^\star = \left(\mathbf{X}^\top \mathbf{X} + \frac{\delta}{1-\delta}\mathrm{diag}(\mathbf{X}^\top \mathbf{X})\right)^{-1} \left(\mathbf{X}^\top \mathbf{X}\right) \widehat{\beta}_{\mathrm{OLS}}.$$

Therefore, under the assumptions of the corollary

$$\widehat{\beta}^\star \xrightarrow{p} (1-\delta)\beta_{\mathrm{OLS}}.$$

This completes the proof. □

A.3 Proof of Theorem 2

Proof of Theorem 2. By definition

$$Z(K_l^*) = \frac{\bar{\Delta}_{K_l^*}}{r(1-r)^{m_l^*}} + \theta_{l,m_0},$$

where K_l^* is a discrete random variable with probability mass function:

$$p(K_l^*) = r(1-r)^{K_l^* - m_0},$$

and supported on the integers larger than m_0.

We first show that the estimator $Z(K_l^*)$ is unbiased (as we average over both K_l^* and ξ_i^k). Algebra shows that

$$\begin{aligned}
\mathbb{E}[Z(K_l^*)] &= \sum_{K=m_0}^{\infty} \mathbb{E}[Z(K_l^*)|K_l^* = K]p(K) \\
&= \sum_{K=m_0}^{\infty} \mathbb{E}\left[\frac{\bar{\Delta}_{K_l^*}}{p(K_l^*)} + \theta_{l,m_0}\middle| K_l^* = K\right] p(K) \\
&= \sum_{K=m_0}^{\infty} \mathbb{E}\left[\frac{\bar{\Delta}_{K}}{p(K)} + \theta_{l,m_0}\middle| K_l^* = K\right] p(K) \\
&= \left(\sum_{K=m_0}^{\infty} \mathbb{E}\left[\widehat{\theta}_n^{\star}(2^{K+1}) - \frac{1}{2}(\widehat{\theta}_n^{O}(2^K) + \widehat{\theta}_n^{E}(2^K))\right]\right) + \mathbb{E}[\theta_{l,m_0}] \\
&= -\frac{1}{2}\left(\mathbb{E}[\widehat{\theta}_n^{O}(2^{m_0})] + \mathbb{E}[\widehat{\theta}_n^{E}(2^{m_0})]\right) + \mathbb{E}[\theta_{l,m_0}] + \lim_{K\to\infty} \mathbb{E}[\widehat{\theta}_n^{\star}(2^{K+1})].
\end{aligned}$$

The expectations in the last line are all finite because Θ is compact. In addition, since the draws are i.i.d. and θ_{l,m_0} is the solution to the problem (21) when 2^{m_0} draws are used we have

$$-\frac{1}{2}\left(\mathbb{E}[\widehat{\theta}_n^{O}(2_0^m)] + \mathbb{E}[\widehat{\theta}_n^{E}(2_0^m)]\right) + \mathbb{E}[\theta_{l,m_0}] = 0.$$

Moreover, the sequence of random variables

$$\{\widehat{\theta}_n^{\star}(2^{K+1})\}$$

is uniformly integrable, because Θ is a compact subset of a finite-dimensional Euclidean

space. Finally, we know that

$$\widehat{\theta}_n^\star(2^{K+1}) \xrightarrow{p} \theta_n^\star,$$

as $K \to \infty$. The uniform integrability of the sequence of estimators then implies

$$\lim_{K\to\infty} \mathbb{E}[\widehat{\theta}_n^\star(2^{K+1})] = \mathbb{E}\left[\lim_{K\to\infty} \widehat{\theta}_n^\star(2^{K+1})\right] = \theta_n^\star,$$

see Theorem 6.2 in DasGupta (2008). We conclude that

$$\mathbb{E}[Z(K_l^*)] = \lim_{K\to\infty} \mathbb{E}[\widehat{\theta}_n^\star(2^{K+1})] = \theta_n^\star.$$

Now we show that the expected computational cost of $Z(K_l^*)$ is finite. In order to compute $Z(K)$ for a given K we need $n \cdot 2^{K+1}$ random draws. Thus, the expected computational cost of $Z(K_l^*)$ is

$$\begin{aligned}\sum_{K=m_0}^{\infty} n2^{K+1} r(1-r)^{K-m_0} &= n \cdot (2^{m_0+1}) \cdot r \sum_{K=m_0}^{\infty} 2^{K-m_0}(1-r)^{K-m_0} \\ &= n \cdot (2^{m_0+1}) \cdot r \sum_{K=m_0}^{\infty} (2(1-r))^{K-m_0}.\end{aligned}$$

The term above converges to

$$\frac{n \cdot (2^{m_0+1}) \cdot r}{1 - 2(1-r)} = \frac{n \cdot (2^{m_0+1}) \cdot r}{2r-1}$$

provided $2(1-r) < 1$, which holds because we have chosen $r > 1/2$.

For the proof on finite variance, we intend to show that

$$\mathbb{E}\left[\bar{\Delta}_K^\top \bar{\Delta}_K\right] = O(2^{-2K}) \tag{32}$$

as $K \to \infty$. Equation (32) guarantees that every processor generates an estimator $Z(K_l^*)$ with finite variance. Since K_l^* is a discrete random variable with probability mass function

$$p(K_l^*) = r(1-r)^{K^*-m_0},$$

$$
\begin{aligned}
\mathbb{E}[Z(K_l^*)^\top Z(K_l^*)] &= \sum_{K=m_0}^{\infty} \mathbb{E}\left[Z(K_l^*)^\top Z(K_l^*)|K_l^*=K\right] p(K) \\
&= \sum_{K=m_0}^{\infty} \mathbb{E}\left[\left(\frac{\bar{\Delta}_K}{p(K)}+\theta_{l,m_0}\right)^\top \left(\frac{\bar{\Delta}_K}{p(K)}+\theta_{l,m_0}\right)\right] p(K) \\
&\leq 2\left(\sum_{K=m_0}^{\infty} \mathbb{E}\left[\frac{\bar{\Delta}_K^\top \bar{\Delta}_K}{p(K)^2}\right] p(K) + \sum_{K=m_0}^{\infty} \mathbb{E}\left[\theta_{l,m_0}^\top \theta_{l,m_0}\right] p(K)\right) \\
&\leq C\left(\sum_{K=m_0}^{\infty} \frac{2^{-2K}}{p(K)} + \sup_{\theta\in\Theta} \|\theta\|_2^2 p(K)\right) \\
&\leq C\left(\sum_{K=m_0}^{\infty} \frac{1}{2^{2m_0}2^{2(K-m_0)}p(K)} + \sup_{\theta\in\Theta} \|\theta\|_2^2 p(K)\right) \\
&\leq C_1\left(\sum_{K=m_0}^{\infty} \frac{1}{r4^{m_0}} \frac{1}{(4(1-r))^{K-m_0}}\right) + C_2
\end{aligned}
$$

the geometric sum in the last expression is finite because we have assumed that $r < \frac{3}{4}$.

To show (32), we do a Taylor expansion of the first-order conditions of the problem (21) around the $\theta_n^\star$. The Karush-Kuhn-Tucker optimality condition for the level 2^K solution $\widehat{\theta}_n^\star(2^K)$ of the problem in (21) implies

$$
0 = \sum_{i=1}^{n}\left[\frac{1}{2^K}\sum_{k=1}^{2^K} \nabla_\theta \ell(x_i \odot \xi_i^k, y_i, \widehat{\theta}_n^\star(2^K))\right]
$$

It follows by the Taylor expansion and Assumption 4 that

$$
\begin{aligned}
0 &= \sum_{i=1}^{n}\left[\frac{1}{2^K}\sum_{k=1}^{2^K} \nabla_\theta \ell(x_i \odot \xi_i^k, y_i, \theta_n^\star)\right] + \sum_{i=1}^{n}\left[\frac{1}{2^K}\sum_{k=1}^{2^K} \nabla_{\theta\theta} \ell(x_i \odot \xi_i^k, y_i, \theta_n^\star)\right]\left(\widehat{\theta}_n^\star(2^K) - \theta_n^\star\right) \\
&\quad + R_{K,\theta} \\
&= \sum_{i=1}^{n}\left[\frac{1}{2^K}\sum_{k=1}^{2^K} \nabla_\theta \ell(x_i \odot \xi_i^k, y_i, \theta_n^\star)\right] + \sum_{i=1}^{n} \nabla_{\theta\theta} \mathbb{E}_{\mathbb{Q}^\star}\left[\ell(X \odot \xi, Y, \theta_n^\star)|X=x_i, Y=y_i\right]\left(\widehat{\theta}_n^\star(2^K) - \theta_n^\star\right) \\
&\quad + R_K + R_{K,\theta}, \qquad (33)
\end{aligned}
$$

where

$$
R_K \equiv \left(\sum_{i=1}^{n}\left(\frac{1}{2^K}\sum_{k=1}^{2^K} \nabla_{\theta\theta} \ell(x_i \odot \xi_i^k, y_i, \theta_n^\star) - \nabla_{\theta\theta} \mathbb{E}_{\mathbb{Q}^\star}\left[\ell(X \odot \xi, Y, \theta_n^\star)|X=x_i, Y=y_i\right]\right)\right)\left(\widehat{\theta}_n^\star(2^K) - \theta_n^\star\right)
$$

and

$$\|R_{K,\theta}\|_2 \leq \sum_{i=1}^{n} \sup_{\theta\in\Theta,\xi} \|\nabla_{\theta\theta\theta}\ell(x_i \odot \xi, y_i, \theta)\|_2 \left\|\widehat{\theta}_n^\star(2^K) - \theta_n^\star\right\|_2^2 \leq C_3 \left\|\widehat{\theta}_n^\star(2^K) - \theta_n^\star\right\|_2^2$$

by Assumption 4. Thus by Assumption 3, we have

$$\mathbb{E}[R_{K,\theta}^\top R_{K,\theta}] = O(2^{-2K})$$

as $K \to \infty$. Moreover, by the multivariate version of Theorem 2 in Bahr (1965) which follows from the Cramér-Wold theorem, we have that

$$\mathbb{E}\left[\left\|\sum_{i=1}^{n}\left(\frac{1}{2^K}\sum_{k=1}^{2^K}\nabla_{\theta\theta}\ell(x_i \odot \xi_i^k, y_i, \theta_n^\star) - \nabla_{\theta\theta}\mathbb{E}_{\mathbb{Q}^\star}\left[\ell(X \odot \xi, Y, \theta_n^\star)|X = x_i, Y = y_i\right]\right)\right\|_2^4\right]$$

is $O(2^{-2K})$.

We can express $R_K^\top R_K$ as $\|R_K\|^2$. The Cauchy-Schwarz inequality implies

$$\begin{aligned}
&\mathbb{E}[R_K^\top R_K] \\
\leq&\mathbb{E}\left[\left\|\sum_{i=1}^{n}\left(\frac{1}{2^K}\sum_{k=1}^{2^K}\nabla_{\theta\theta}\ell(x_i \odot \xi_i^k, y_i, \theta_n^\star) - \nabla_{\theta\theta}\mathbb{E}_{\mathbb{Q}^\star}\left[\ell(X \odot \xi, Y, \theta_n^\star)|X = x_i, Y = y_i\right]\right)\right\|_2^2 \cdot \right. \\
&\left.\left\|\widehat{\theta}_n^\star(2^K) - \theta_n^\star\right\|_2^2\right].
\end{aligned}$$

By Hölder's inequality we have

$$\begin{aligned}
&\mathbb{E}[R_K^\top R_K] \\
\leq&\mathbb{E}\left[\left\|\sum_{i=1}^{n}\left(\frac{1}{2^K}\sum_{k=1}^{2^K}\nabla_{\theta\theta}\ell(x_i \odot \xi_i^k, y_i, \theta_n^\star) - \nabla_{\theta\theta}\mathbb{E}_{\mathbb{Q}^\star}\left[\ell(X \odot \xi, Y, \theta_n^\star)|X = x_i, Y = y_i\right]\right)\right\|_2^4\right]^{\frac{1}{2}} \times \\
&\qquad\qquad \mathbb{E}\left[\left\|\widehat{\theta}_n^\star(2^K) - \theta_n^\star\right\|_2^4\right]^{\frac{1}{2}} \\
\leq&O(2^{-2K}).
\end{aligned}$$

Finally, consider the solutions $\widehat{\theta}_n^\star(2^{K_l^\star+1}), \widehat{\theta}_n^O(2^{K_l^\star}), \widehat{\theta}_n^E(2^{K_l^\star})$ conditional on $K_l^\star = K$. Denote the remainder terms in (33) corresponding to the level 2^{K+1} solution $\widehat{\theta}_n^\star(2^{K+1})$ as

$R^\star_{K+1}, R^\star_{K+1,\theta}$. Similarly, denote the remainder terms in (33) corresponding to the level 2^K solution $\widehat{\theta}^O_n(2^K)$ (and, respectively, $\widehat{\theta}^E_n(2^K)$) as $R^O_K, R^O_{K,\theta}$ ($R^E_K, R^E_{K,\theta}$). By the construction of $\widehat{\theta}^O_n(2^K), \widehat{\theta}^E_n(2^K)$ using odd and even indices, we have, from (33)

$$\begin{aligned}
&-\sum_{i=1}^n \nabla_{\theta\theta}\mathbb{E}_{\mathbb{Q}^\star}[\ell(X\odot\xi, Y, \theta^\star_n)|X=x_i, Y=y_i]\left(\widehat{\theta}^\star_n(2^{K+1}) - \frac{1}{2}(\widehat{\theta}^O_n(2^K)+\widehat{\theta}^E_n(2^K))\right)\\
=&R^\star_{K+1} - \frac{1}{2}(R^O_K + R^E_K) + R^\star_{K+1,\theta} - \frac{1}{2}(R^O_{K,\theta}+R^E_{K,\theta}).
\end{aligned}$$

By Assumption 4,

$$\sum_{i=1}^n \nabla_{\theta\theta}\mathbb{E}_{\mathbb{Q}^\star}[\ell(X\odot\xi, Y, \theta^\star_n)|X=x_i, Y=y_i] = n\cdot\nabla_{\theta\theta}\mathbb{E}_{\mathbb{Q}^\star}[\ell(X\odot\xi, Y, \theta^\star_n)]$$

is invertible. Thus, we have shown that

$$\begin{aligned}
\bar{\Delta}_K &\equiv \widehat{\theta}^\star_n(2^{K+1}) - \frac{1}{2}(\widehat{\theta}^O_n(2^K)+\widehat{\theta}^E_n(2^K))\\
&= (n\cdot\nabla_{\theta\theta}\mathbb{E}_{\mathbb{Q}^\star}[\ell(X\odot\xi, Y, \theta^\star_n)])^{-1}\left(R^\star_{K+1} - \frac{1}{2}(R^O_K+R^E_K) + R^\star_{K+1,\theta} - \frac{1}{2}(R^O_{K,\theta}+R^E_{K,\theta})\right).
\end{aligned}$$

Since each of the terms on the right-hand side have been shown to be $O(2^{-2K})$, we conclude that $\mathbb{E}[\bar{\Delta}_K^\top\bar{\Delta}_K] = O(2^{-2K})$. □

A.4 Additional Numerical Results

Here we try to provide some justifications for our choice of parameter.

Learning Rate: We first fix an all zeros initialization scheme, and vary the learning rate. We summarize the average parameter divergence and 1-standard deviation error for 20 repetitions of the SGD algorithm in Table 1. We can observe the learning rate 0.0001 shows a clear advantage.

Initialization: Next we fix the learning rate to be 0.0001, and consider different initialization schemes. We note that the mean value (resp, absolute value) of elements in $\beta^\star$ is 0.0434 (resp, 0.1710). Table 2 shows the average parameter divergence and 1-standard deviation from 20 repetitions of the SGD algorithm. We see marginal difference for difference choices of initialization, and the initialization at origin is a fair choice. The robustness to initialization is as expected, since our linear regression has a convex objective function, and fixed learning rate guarantees the coefficient is able to get out of the neighbourhood of initialization.

Wall-Clock Time: We then document the numerical results for 30s/120s wall-clock

Learning rate	0.001	**0.0001**	0.00001
$\|\widehat{\beta}_{SGD} - \beta^\star\|_\infty$	0.1034 ± 0.0090	$\mathbf{0.0332 \pm 0.0036}$	0.0783 ± 0.0007

Table 1: Comparison for different learning rates, with fixed zero initializations.

Initializations	**all zeros**	all 0.2's	all 1's
$\|\widehat{\beta}_{SGD} - \beta^\star\|_\infty$	$\mathbf{0.0331 \pm 0.0036}$	0.0332 ± 0.0021	0.0328 ± 0.0022
Initializations	i.i.d $\mathcal{N}(0,1)$	i.i.d $\mathcal{N}(0,10)$	i.i.d $\mathcal{N}(0,10^2)$
$\|\widehat{\beta}_{SGD} - \beta^\star\|_\infty$	0.0317 ± 0.0022	0.0316 ± 0.0020	0.0319 ± 0.0022

Table 2: Comparison for different initialization schemes with fixed learning rate 0.0001.

time, see Figures 4 - 6 for the case of 30s, and Figures 7 - 9 for the case of 120s. Except for the l_∞ parameter divergence with a 30s wall-clock time (Figure 5), we see that the proposed unbiased approach outperforms the standard SGD when the number of parallel iterations reaches above some threshold.

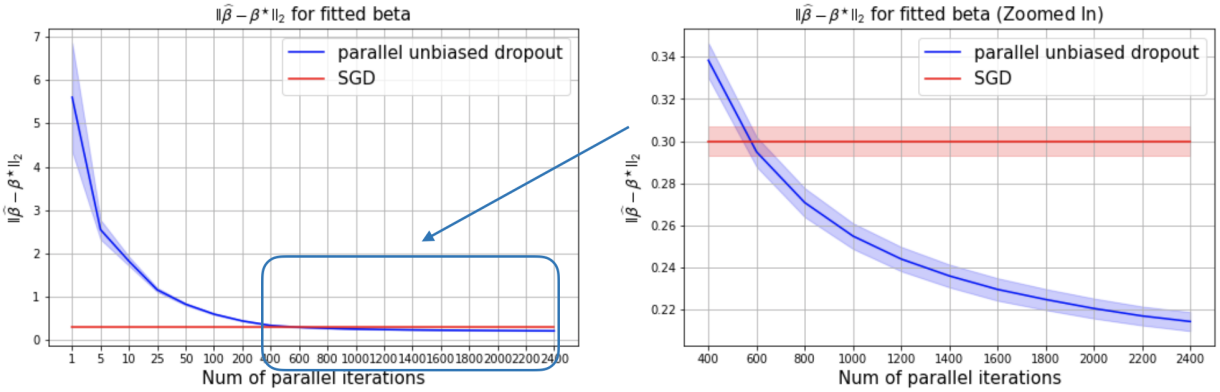

Figure 4: l_2 difference for 30s wall-clock time

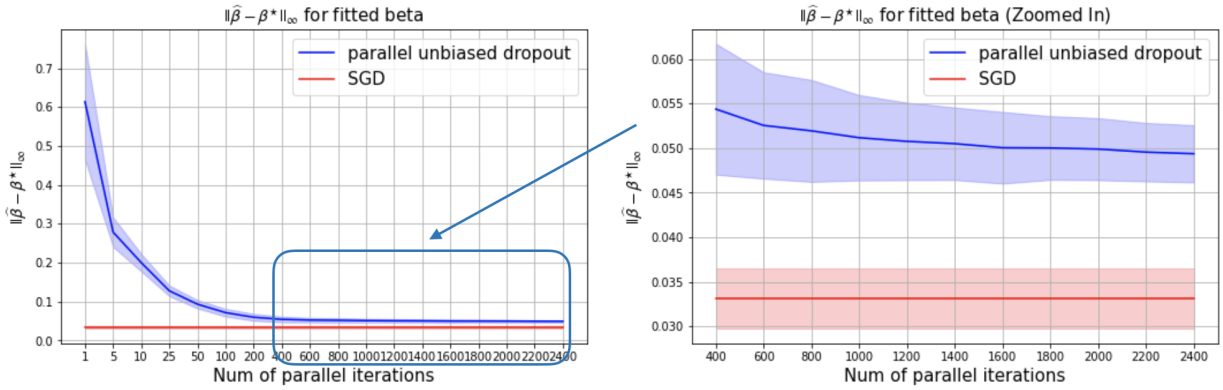

Figure 5: l_∞ difference for 30s wall-clock time

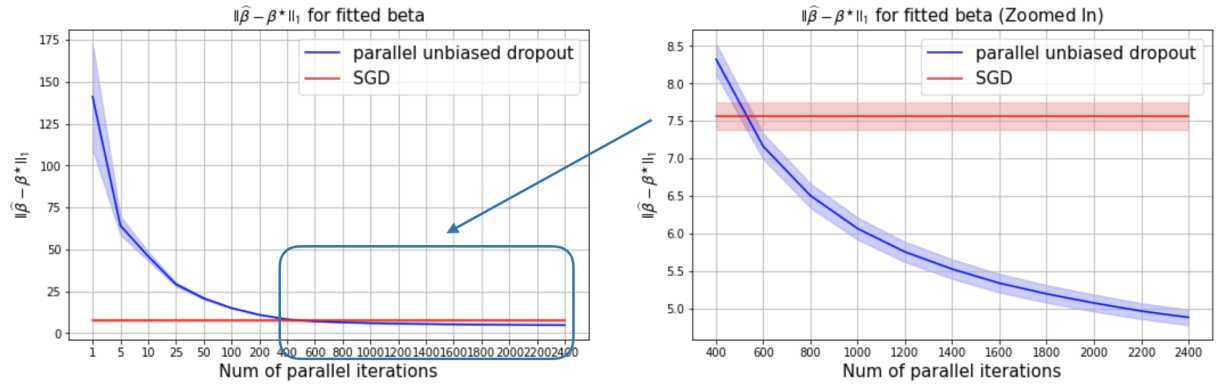

Figure 6: l_1 difference for 30s wall-clock time

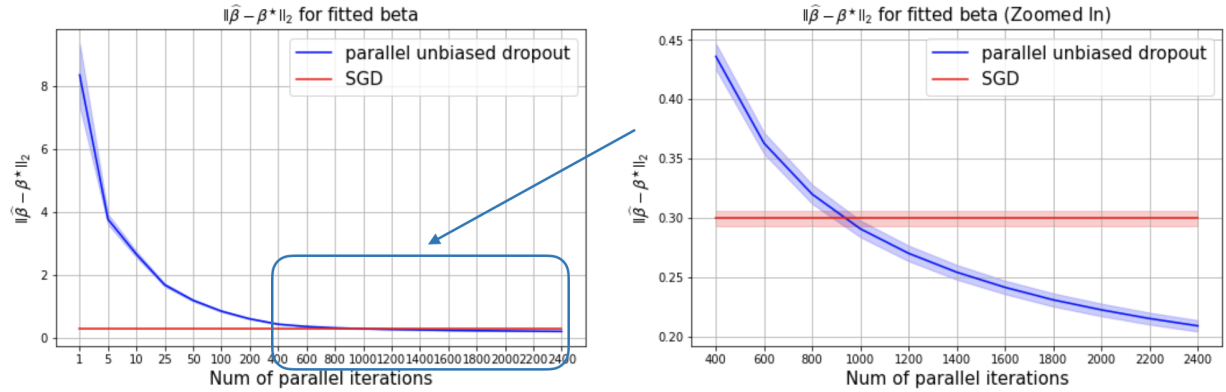

Figure 7: l_2 difference for 120s wall-clock time

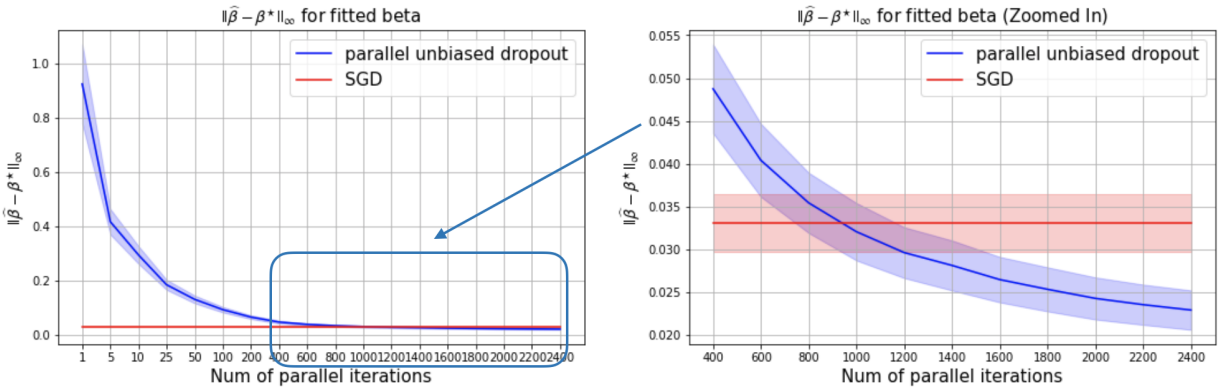

Figure 8: l_∞ difference for 120s wall-clock time

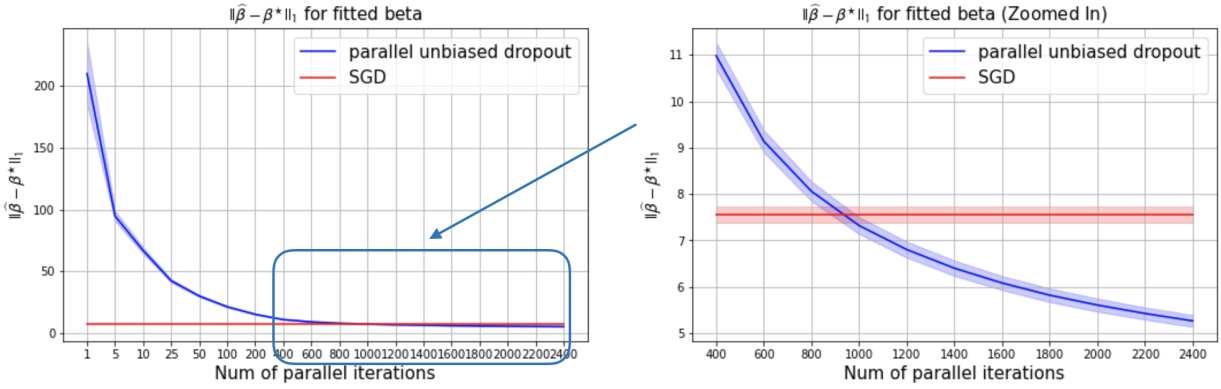

Figure 9: l_1 difference for 120s wall-clock time