Doubly Stochastic Generative Arrivals Modeling

Yufeng Zheng
Zeyu Zheng
Department of Industrial Engineering and Operations Research, University of California, Berkeley
yufeng_zheng@berkeley.edu, zyzheng@berkeley.edu

This Version: December 30, 2020; Clickable link for most recent version

We propose a new framework named DS-WGAN that integrates the doubly stochastic (DS) structure and the Wasserstein generative adversarial networks (WGAN) to model, estimate, and simulate a wide class of arrival processes with general non-stationary and random arrival rates. Regarding statistical properties, we prove consistency and convergence rate for the estimator solved by the DS-WGAN framework under a non-parametric smoothness condition. Regarding computational efficiency and tractability, we address a challenge in gradient evaluation and model estimation, arised from the discontinuity in the simulator. We then show that the DS-WGAN framework can conveniently facilitate what-if simulation and predictive simulation for future scenarios that are different from the history. Numerical experiments with synthetic and real data sets are implemented to demonstrate the performance of DS-WGAN. The performance is measured from both a statistical perspective and an operational performance evaluation perspective. Numerical experiments suggest that, in terms of performance, the successful model estimation for DS-WGAN only requires a moderate size of representative data, which can be appealing in many contexts of operational management.

1. Introduction

In many areas in the domain of operations research and management science, such as service systems, logistics and supply chain systems, and financial systems, the randomness of arrivals is a primary source of uncertainty. The arrivals can be customer flows, demand orders, traffic, among others. To put the discussion in context, in this paper, we consider *demand arrivals* to a system that have cycles of operations. A cycle can be a day, a week, or a month. Demands arrive at these systems according to a continuous-time stochastic process whose arrival rate is typically non-stationary and likely also contains complicated dependence/correlation structure at different time scales. The uncertainties in these arrival processes cannot always be predicted. Appropriately modeling and statistically characterizing the arrival processes is critical for policy evaluation and performance evaluation in these systems. In addition, efficiently simulating new arrival data that contains certain statistical features is often needed in many decision-making tools and

future-planning tools. Based on these needs, our work centers around providing new and effective approaches for the modeling, estimation, and simulation of arrival processes that have potentially complicated non-stationary and stochastic structures.

A natural and powerful model for such arrival processes is non-stationary *doubly stochastic Poisson process* (DSPP). A non-stationary DSPP is a Poisson process with a random and time-varying rate (or, intensity) process. The Poisson structure is a natural assumption for arrival processes in a large number of applications, in particular when the arrivals come from people making their own decisions on whether and when to join the system. The Poisson superposition theorem is a mathematical support for the Poisson structure assumption. More theory and empirical justifications can be found in Whitt (2002), Brown et al. (2005), Koole (2013), Kim and Whitt (2014), Ibrahim et al. (2016), among others. In addition to the Poisson structure, empirical evidence suggests that the arrival rate process is not only time-varying but follows a stochastic process by itself. Standard non-homogeneous Poisson process (NHPP), on the other hand, does not have randomness in the arrival rate process. Empirical evidence has shown that NHPP is not a sufficient model in many cases as it fails to capture the complicated variance and correlation behaviors arised in real arrival data. See Avramidis et al. (2004), Brown et al. (2005), Steckley et al. (2005), Steckley et al. (2009), Channouf and L'Ecuyer (2012), Ibrahim et al. (2016) and Oreshkin et al. (2016) for theory and empirical support. In particular, in the seminal work by Oreshkin et al. (2016), the authors demonstrate strong empirical evidence for the DSPP model and propose a parametric DSPP model that uses Gamma-compounded Gaussian copula to model the random arrival rates. In our work, we make no parametric or structured distributional assumptions on the random arrival rate process. In this case, both the arrival process and the arrival rate process are infinite-dimensional random objects. In the data and in the general DSPP model, various statistical features including variance and correlation can arise with different magnitude at different time scales, such as 1-minute, 5-minute, half-hour, etc. It is therefore impossible for a fixed model to capture the statistical features at every time scale that can arise in the data. In this work, we build a neural-network based doubly stochastic model that is targeted to **match the statistical features presented in the given arrival data at time scales that are equal to or larger than a given resolution**. This resolution (e.g., five-minute, half-hour) can either be specified by users or be determined by the relevance to performance evaluation and decision making. For example, in high-volume service systems that are involved with queueing, the structure of an arrival process at small time scales (e.g., the time scale associated with inter-arrival time distribution) may not impact the queueing performance. This is supported by the theory given in Glynn (2014) and an empirical experiment given in Section 8.1 of this work.

In this paper, we presume that the system operates in cycles and that n independent copies of arrival process data are collected from n cycles of operations. Without loss of generality, we consider each cycle to be a day. The arrival process on each day is treated as a non-stationary DSPP with an arbitrary non-stationary random arrival rate process. One particular difficulty in estimation is that the random arrival rate process is not directly observed, and can only be inferred from the observed arrivals. We propose a framework called Doubly Stochastic Wasserstein Generative Adversarial Network (DS-WGAN) to learn from given data and simulate new non-stationary doubly stochastic Poisson processes that have the same statistics with the arrival data. Specifically, a day is partitioned into disjoint time intervals at a desired given resolution. The arrival count in each time interval therefore has a doubly stochastic Poisson distribution, and the random Poisson intensities in different time intervals are correlated. The DS-WGAN framework includes a *doubly stochastic generative neural network* that uses a generative neural network to simulate the jointly distributed random intensities, connected by a Poisson counts simulator to generate doubly stochastic Poisson counts. The doubly stochastic generative neural network is then adversarially trained such that the generated doubly stochastic Poisson counts match the statistics from real data. Once a vector of counts are simulated by the doubly stochastic generative neural network, we develop an *arrival epochs simulator* that can take a vector of arrival counts as inputs and generate sequential arrival epochs in continuous time.

The advantages of the DS-WGAN framework are three-fold. First, DS-WGAN exploits the doubly stochastic structure in arrival processes and assigns the generative neural network to only focus on capturing the statistical features in the unobserved random arrival rate process. No parametric or specific distribution assumption is added to the random arrival rate process. Second, because of the exploitation of the doubly stochastic structure, the successful model estimation of DS-WGAN only requires a moderate size of data. We show that a sample size of $n = 300$ days of data render robust model estimation of DS-WGAN using several experiments with synthetic and real data. We remark that this advantage can be particularly attractive in the contexts of operations research and management science, in which there are rarely massive size of representative data available (unlike the image area, as a comparison). Third, DS-WGAN enables what-if simulation analysis, which can support performance evaluation and decision making for systems that do not yet exist; see Nelson (2016) for a discussion on what-if simulation. Consider a what-if planning decision in the service system for the future three months. What is an appropriate workers scheduling design if the expected number of daily user arrivals in the future three months is 20% larger, compared to that in the historical data? To answer such what-if questions, there is a need to simulate new arrival processes that have 20% of increase in the daily level but reserve the time-of-day pattern and correlation structures within a day. The DS-WGAN is a natural fit for this task

while the plain WGAN framework cannot handle this task. We also discuss that the DS-WGAN framework can be integrated with other predictive models to facilitate predictive simulation.

The challenges brought by this new DS-WGAN framework are mainly from two aspects, the statistical aspect and the computational aspect. First, the statistical aspect involves two research questions. What types of doubly stochastic Poisson processes can be consistently learned by DS-WGAN? What is the statistical rate of convergence if the DS-WGAN is consistently learning the arrival process? In fact, even for the standard WGAN framework (Villani (2008) and Arjovsky et al. (2017)) without the doubly stochastic structure, limited theory studies exist on the statistical guarantees. Bai et al. (2018) and Chen et al. (2020) are the first to provide statistical theory for GAN and WGAN. In our work, we prove statistical guarantees for DS-WGAN. Specifically, we show that with appropriately chosen neural network structures, for an arbitrary doubly stochastic Poisson model, if the joint distribution of the underlying stochastic intensities has a smooth density, DS-WGAN can provide statistically consistent estimators and simulators. A guaranteed upper bound on the rate of convergence is also established as a theory support. The proofs build upon that of Bai et al. (2018) and Chen et al. (2020) while a key difficulty is in handling the doubly stochastic layer. We develop a coupling technique to circumvent this difficulty. Second, the computational aspect observes a challenge in the evaluation of stochastic gradient. Nominally, the model estimation (training) of WGAN-type framework is through stochastic gradient descent algorithms, and the computationally effective backpropogation routine is used to compute the gradients of the discriminator output with respect to the neural network parameters in the generator network. However, if the same routine were to be used for DS-WGAN, the backpropogation needs to go backward through the Poisson counts simulator. The Poisson counts simulation is essentially a stochastic function taking continuous inputs and generating discrete outputs. The sample-path gradient would vanish almost everywhere, creating gradients with value 0 using the backpropogation routine. We construct an approximation utilizing the properties of Poisson distribution that provides a non-vanishing and meaningful gradient. This approximated gradient is convenient to be integrated to packages such as PyTorch to implement the backpropogation routine. Numerical experiments suggest that this approximation supports stable and successful model estimation (training) of DS-WGAN.

The use of neural networks among other machine learning tools to model arrival processes is receiving rising interests. Du et al. (2016) and Mei and Eisner (2017) are two of the pioneering works in connecting Hawkes processes with recurrent neural networks (RNN). They model and learn the conditional intensity function and use that to generate the next inter-arrival time. Wang et al. (2020) consider a DSPP model in which the intensity process is driven by a general stochastic different equation (SDE). They discuss the use of neural networks to model the unknown drift

and diffusion functions of SDE, as well as the associated model training. Cen et al. (2020) use neural networks to model and predict the next inter-arrival time of a non-homogeneous Poisson process. Our work focuses on the doubly stochastic Poisson processes with general arbitrary random intensity processes and exploits the doubly stochastic structure to propose a new framework for modeling and simulation. Theory-wise, to the best of our knowledge, we are the first to provide statistical guarantees for neural-network based general doubly stochastic Poisson process models. From an operational point of view, Unlike the modeling and prediction of the next inter-arrival time (microscopic structure) in the aforementioned literature, our work focus on learning the dependence structures at a longer time scale that is relevant to service operations (mesoscopic structure). Our numerical experiments in Section 8.1 suggest that learning this mesoscopic structure is critical and sufficient to support service operations performance evaluation. However, only focusing on modeling the inter-arrival times may lose the dependence structure on longer time scales.

The rest of paper is organized as follows. Section 2 discusses the model setup. Section 3 discusses the DS-WGAN framework. Section 4 discusses the statistical theory for DS-WGAN. Section 5 discusses the computational aspect for the optimization problem associated with DS-WGAN. Section 6 introduces the arrival epochs simulator. Section 7 discusses the use of DS-WGAN to facilitate what-if simulation and predictive simulation. Section 8 provides numerical experiments.

2. Model Setup

We consider settings in which arrivals to a system are observed over n independent cycles of operations. The cycle length can be taken as a day, a week or a month, depending on the system. Without loss of generality, we assume that each cycle represents a day over an time interval $[0, T]$ in which $T > 0$. For example, if $T = 12$ (hours), the system is open for 12 hours during a day. The arrival rate within a day can change with time and can also be random itself. To capture these features, we consider a general class of arrival process models called *Doubly Stochastic Poisson Process* (DSPP). A DSPP is a point process $N = (N(t) : t \in [0, T])$ with a random intensity function $\lambda = (\lambda(t) : t \in [0, T])$. For a point process, with any given t, the random variable $N(t)$ denotes the number of arrivals in a day over the time interval $[0, t]$. The DSPP point process is *simple* in the sense that N increases exactly by one at each arrival epoch and no batch arrivals are possible. (The feature of batch arrivals can be separately modeled and then compounded with an underlying DSPP, but is not the focus of this work.) The random intensity function $\lambda = (\lambda(t) : t \in [0, T])$ is a random object defined on a functional space on $[0, T]$ with proper topology. For example, the function space can be $C[0, T]$ endowed with the Skorokhod J_1 topology, which includes all the continuous functions on $[0, T]$. Conditional on a realization of the random intensity function $\lambda =$

$(\lambda(t) : t \in [0,T])$, $N = (N(t) : t \in [0,T])$ is a nonhomogeneous Poisson process. That is, conditional on λ, for any set of disjoint time intervals $(t_1, t_1']$, $(t_2, t_2']$, ... , $(t_m, t_m']$ on $[0,T]$, the arrival counts on each time interval, $N(t_1') - N(t_1)$, $N(t_2') - N(t_2)$, ... , $N(t_m') - N(t_m)$ are independent Poisson random variables. Conditional on λ, the expectation of $N(t_k') - N(t_k)$ is given by

$$\int_{t_k}^{t_k'} \lambda(s)\, ds$$

for $k = 1, 2, \ldots, m$. Unconditionally, the marginal distributions of $N(t_1') - N(t_1)$, $N(t_2') - N(t_2)$, ... , $N(t_m') - N(t_m)$ are not necessarily Poisson nor independent, because of the stochastic structure of the underlying random intensity process. DSPP is also called *Cox Process* in the literature (Cox and Smith (1954)).

Suppose that n independent days of arrival data are observed. Let N_i denote the arrival process on the i-th day, for $i = 1, 2, \ldots, n$. We assume that the N_i's are n replications of a DSPP. In this work, we consider the general DSPP model to fit the arrival data, without imposing any distributional assumption on the random intensity function. In fact, because of the continuous-time nature, the random intensity function is an infinite-dimensional random object. Various statistical features such as variance and correlation can arise with different magnitude at different time scales, such as 1-minute, 5-minutes, half-hour, etc. It is therefore impossible for a fixed model to capture the statistical features at every time scale for a general DSPP, using only a finite size of data. Furthermore, a major difficulty in model estimation is that the random intensity function is not directly observed. Our first goal in this work is to overcome these difficulties, and use data to estimate a DSPP model and construct a simulator that can simulate new independent replications of the arrival processes. The simulated arrival processes are targeted to **match the statistical features of the given arrival data at all the time scales that are equal to or larger than a given resolution**. Specifically, for a given resolution, say 5 minutes, the simulated arrival processes from our proposed simulator are targeted to match the arrival data regarding the statistical features of the arrival counts at 5 minutes or longer. We next elaborate on this goal.

Consider a resolution $\Delta > 0$. Set $p = T/\Delta$ and we presume that p is an integer. The time horizon $[0,T]$ is partitioned into p time intervals of equal length. The equal length notion is for presentation simplicity. Our framework does not require the length of each time interval to be equal. Let $\mathbf{X} = (X_1, X_2, \ldots, X_p)$ be a random vector that records the interval counts in each time interval. Specifically, for $i = 1, 2, \ldots, p$,

$$X_i = N(i\Delta) - N((i-1)\Delta).$$

Because $N = (N(t) : t \in [0,T])$ is a doubly stochastic Poisson process, conditional on the random intensity function $\lambda = (\lambda(t) : 0 \le t \le T)$, the marginal distribution of X_i is Poisson with mean

$$\Lambda_i \triangleq \int_{(i-1)\Delta}^{i\Delta} \lambda(s)\, ds.$$

The unconditional joint distribution of $\mathbf{X}$ is therefore determined by the joint distribution of $\mathbf{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_p)$. Because the underlying random intensity function λ can have arbitrary stochastic structure, we do not impose any distributional assumption on $\mathbf{\Lambda}$ when building our model and simulator.

Denote the $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ as n independent and identically distributed copies (iid) of $\mathbf{X}$ observed in the n days of arrival data. We call the data set $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ as *interval count data*. Our simulator to built based on the data is composed of two parts. The first part is a doubly stochastic generative neural network that, once trained using the data, can generate independent copies of arrival counts vector that is targeted to have the same joint distribution as $\mathbf{X}$. The second part is a simulator, called *arrival epochs simulator*, that can take a vector of arrival counts as inputs and simulate arrival epochs in continuous time. We discuss the first part in Section 3.1 and the second part in Section 6.

3. Doubly Stochastic Wasserstein Generative Adversarial Network

In this section, we develop a new framework called *Doubly Stochastic Wasserstein Generative Adversarial Network* (DS-WGAN). In Section 3.1, we discuss the simulator part of the DS-WGAN framework. In Section 3.2, we discuss the discriminator part of the DS-WGAN framework. Given a set of interval count data as training data, DS-WGAN learns to simulate new interval count data with the same statistics as the training data. This new framework is developed upon but has distinct structure with the *Wasserstein Generative Adversarial Network* (WGAN). An important new modeling element is that DS-WGAN has an additional Poisson counts simulator that effectively incorporate the doubly stochastic structure of arrival data. This new modeling element brings challenges to both the analysis of the statistical properties of the estimated model and the computation involved with model estimation (training) procedure. We provide approaches to address these challenges in Section 4 and Section 5 respectively .

3.1. Modeling Framework of DS-WGAN: Simulator

Recall that $\mathbf{X} = (X_1, X_2, \ldots, X_p)$ represents the random vector in $\mathbb{Z}^p$ that records the arrival counts in each consecutive time interval within a day. Let μ denote the true joint probability distribution of $\mathbf{X}$. The training data is n iid copies of $\mathbf{X}$, given by $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$. Let $\hat{\mu}_n$ denote the empirical distribution of the data. Different from a classical routine of first estimating the probability distribution μ from data and then simulating new samples according to the estimated distribution, DS-WGAN contains a routine that implicitly learns the distribution from data and simultaneously obtains a simulator that can generate new samples from the desired distribution μ. The simulator aims at taking a vector of iid simple random variables (such as Uniform(0,1)) as inputs and mapping this vector into one that has the same distribution as μ. In the framework of DS-WGAN,

this simulator is composed of two parts, a neural-network based generator and a Poisson counts simulator. Figure 3 illustrates the simulator of DS-WGAN, with the details and notation to be specified right afterwards.

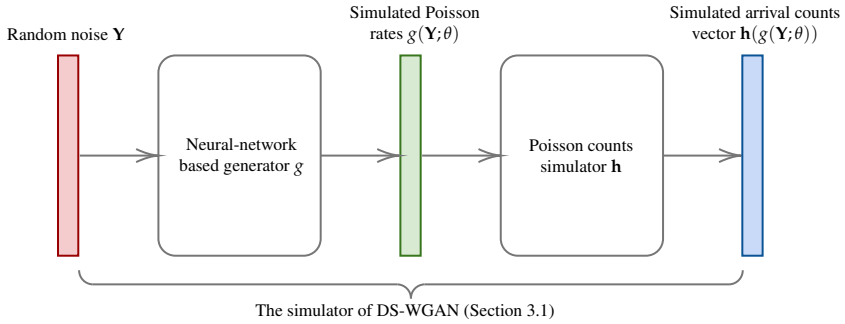


Figure 1 The simulator of DS-WGAN and the arrival epochs simulator.

The neural-network based generator is a mapping $g : \mathbb{R}^p \to \mathbb{R}^p$ that adopts a neural network (NN) architecture, parametrized by NN parameters. Specifically, given the number of layers $L \in \mathbb{Z}^+$ in the NN and $n_l \in \mathbb{Z}^+$ as the width of the l-th layer for $l = 1, 2, \cdots, L$, for an input variable $\mathbf{y} \in \mathbb{R}^p$, the functional form of the generator is given by

$$g(\mathbf{y}; \mathbf{W}, \mathbf{b}) = \mathbf{W}_L \cdot \sigma\left(\mathbf{W}_{L-1} \cdots \sigma\left(\mathbf{W}_1 \mathbf{y} + \mathbf{b}_1\right) \cdots + \mathbf{b}_{L-1}\right) + \mathbf{b}_L \tag{1}$$

in which $\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_L)$ and $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_L)$ represent all the parameters in the NN, with $\mathbf{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$, $\mathbf{b}_l \in \mathbb{R}^{n_l \times 1}$ for $l = 1, 2, \ldots, L$, where $n_0 = p$ is set as the dimension of the input variable. The operator $\sigma(\cdot)$ takes a vector of any dimension as input and is a component-wise operator. Specifically, for any $q \in \mathbb{Z}^+$ and any vector $\mathbf{x} = (x_1, x_2, \cdots, x_q)^\top \in \mathbb{R}^{q \times 1}$,

$$\sigma(\mathbf{x}) \triangleq (\max(x_1, 0), \max(x_2, 0), \cdots, \max(x_q, 0))^\top.$$

In the NN literature, the $\mathbf{W}_l$'s are often called weight matrices and the $\mathbf{b}_l$'s are called bias vectors. Since $g(\mathbf{y}; \mathbf{W}, \mathbf{b})$ has its function value taken in $\mathbb{R}^p$, we write

$$g(\mathbf{y}; \mathbf{W}, \mathbf{b}) = (g_1(\mathbf{y}; \mathbf{W}, \mathbf{b}), g_2(\mathbf{y}; \mathbf{W}, \mathbf{b}), \cdots, g_p(\mathbf{y}; \mathbf{W}, \mathbf{b}))^\top \tag{2}$$

in which $g_i(\mathbf{y}; \mathbf{W}, \mathbf{b}) : \mathbb{R}^p \to \mathbb{R}$ represents the i-th dimension of the function value of f, for $i = 1, 2, \cdots, p$. Denote $\tilde{n} = (n_1, n_2, \cdots, n_L)$. We define $\mathcal{G}_{\text{NN}}$ as a class of generators that satisfy some given regularity conditions on the NN parameters.

$$\mathcal{G}_{\text{NN}}(L, \tilde{n}, \kappa, K) = \Big\{ g : \mathbb{R}^p \to \mathbb{R}^p \mid g \text{ takes the form in (1) with } L \text{ layers, } n_l\text{'s as the width of each layer,}$$
$$\|\mathbf{W}_i\|_\infty \le \kappa, \|\mathbf{b}_i\|_\infty \le \kappa, \sum_{i=1}^{L} \|\mathbf{W}_i\|_0 + \|\mathbf{b}_i\|_0 \le K, \text{ for } i = 1, 2, \cdots, L \Big\}, \tag{3}$$

in which $\|\cdot\|_\infty$ denotes the max-norm that takes the max absolute value of all elements in the input matrix or vector, and $\|\cdot\|_0$ denotes the zero-norm that takes the number of all non-zero elements in the input matrix or vector. The parameters $L, \tilde{n}, \kappa, K$ regularizes the size and magnitude of elements of the generator neural network. In terms of notation, we may drop the dependence of $g(\cdot\,; \mathbf{W}, \mathbf{b})$ on the neural network parameters $\mathbf{W}, \mathbf{b}$ and conveniently write $g(\cdot)$. We also summarize the parameters $(\mathbf{W}, \mathbf{b})$ into θ for notation convenience.

Next we introduce the *Poisson counts simulator* that takes the output from the generator as input, and simulates a vector of Poisson counts as output. Denote $\mathbf{Y}$ as a random vector in which each dimension is an independent Uniform(0,1) random variable. Taking $\mathbf{Y}$ as inputs, the generator g outputs $g(\mathbf{Y}) = (g_1(\mathbf{Y}), g_2(\mathbf{Y}), \ldots, g_p(\mathbf{Y}))$ that is intended to be a realization of the vector of random Poisson rates for $\mathbf{X}$. The Poisson counts simulator, denoted as $\mathbf{h}$, is a random mapping from $\mathbb{R}_+^p$ to $\mathbb{Z}^p$. Mathematically, the Poisson counts simulator involves p mutually independent Poisson processes with constant rate one, denoted as $(M_1(t) : t \geq 0), (M_2(t) : t \geq 0), \ldots, (M_p(t) : t \geq 0)$, which are also independent of the randomness in $\mathbf{Y}$. The Poisson counts simulator $\mathbf{h}$ maps any realization of the random intensities $g(\mathbf{Y}) = (g_1(\mathbf{Y}), g_2(\mathbf{Y}), \ldots, g_p(\mathbf{Y}))$ into

$$\mathbf{h}(g(\mathbf{Y})) = (M_1(g_1(\mathbf{Y})), M_2(g_2(\mathbf{Y})), \ldots, M_p(g_p(\mathbf{Y})))^\top. \tag{4}$$

We use $h_i(g(Y))$ to represent the i-th dimension element of the output of $\mathbf{h}(g(\mathbf{Y}))$.

3.2. Modeling Framework of DS-WGAN: Wassertein Distance and Discriminator

In the previous section, the simulator generates $\mathbf{h}(g(\mathbf{Y}))$ that is intended to have the same distribution as true distribution μ. To evaluate how good a simulator is, we use the Wasserstein distance to quantify the distance between the distribution of $\mathbf{h}(g(\mathbf{Y}))$ and the empirical distribution $\hat{\mu}_n$ from data. The use of Wasserstein distance in quantifying the distance between high-dimensional random vectors has proven to be effective in the literature. For any given generator g, denote the distribution of $\mathbf{h}(g(\mathbf{Y}))$ as $\mu_{\mathbf{h}(g(\mathbf{Y}))}$. The Wasserstein distance is then given by

$$W\left(\hat{\mu}_n, \mu_{\mathbf{h}(g(\mathbf{Y}))}\right) = \inf_{\gamma \in \Pi\left(\hat{\mu}_n, \mu_{\mathbf{h}(g(\mathbf{Y}))}\right)} \mathbb{E}_{(\mathbf{X}_r, \mathbf{X}_g) \sim \gamma}[||\mathbf{X}_r - \mathbf{X}_g||_2], \tag{5}$$

where $\Pi\left(\hat{\mu}_n, \mu_{\mathbf{h}(g(\mathbf{Y}))}\right)$ denotes the set of all joint distributions whose marginals are respectively $\hat{\mu}_n$ and $\mu_{\mathbf{h}(g(\mathbf{Y}))}$, and $||\cdot||_2$ denotes the L_2 norm. The notion of $(\mathbf{X}_r, \mathbf{X}_g) \sim \gamma$ represents that $(\mathbf{X}_r, \mathbf{X}_g)$ is a random vector that follows the distribution γ. With the Wasserstein distance in hand, the best generator within a class $\mathcal{G}_{\text{NN}}$ is given by the following optimization problem

$$g^* \in \arg\min_{g \in \mathcal{G}_{\text{NN}}} W\left(\hat{\mu}_n, \mu_{\mathbf{h}(g(\mathbf{Y}))}\right). \tag{6}$$

However, the Wasserstein distance in the high dimension does not carry a closed-form computation. Alternatively, using the Kantorovich-Rubinstein duality, we have

$$\begin{aligned} W\left(\hat{\mu}_n, \mu_{\mathbf{h}(g(\mathbf{Y}))}\right) &= \sup_{\|f\|_L \le 1} \mathbb{E}_{\mathbf{X}_r \sim \hat{\mu}_n}[f(\mathbf{X})] - \mathbb{E}_{\mathbf{X}_g \sim \mu_{\mathbf{h}(g(\mathbf{Y}))}}[f(\mathbf{X}_g)] \\ &= \sup_{\|f\|_L \le 1} \mathbb{E}_{\mathbf{X}_r \sim \hat{\mu}_n}[f(\mathbf{X})] - \mathbb{E}_{\mathbf{Y} \sim \nu}[f(\mathbf{h}(g(\mathbf{Y})))], \end{aligned}$$

where the supremum is over all the 1-Lipschitz functions f (i.e., $|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \le ||\mathbf{x}_1 - \mathbf{x}_2||_2$ for any $\mathbf{x}_1$ and $\mathbf{x}_2$ in $\mathbb{R}^d$). The optimization problem in (6) is then equivalent to

$$g^* \in \arg\min_{g \in \mathcal{G}_{\text{NN}}} \sup_{\|f\|_L \le M} \mathbb{E}_{\mathbf{X}_r \sim \hat{\mu}_n}[f(\mathbf{X})] - \mathbb{E}_{\mathbf{Y} \sim \nu}[f(\mathbf{h}(g(\mathbf{Y})))] \tag{7}$$

for any given $M > 0$. Note that requiring 1-Lipschitz is equivalent to requiring M-Lipschitz in terms of the optimization problem, since the only difference is a scale.

In the optimization problem (7), the computation to find the best f is not tractable. Instead of searching over the space of all M-Lipschitz functions, we use a neural network (NN) to approximate f and search over all M-Lipschitz functions parametrized by the NN architecture. Such a mapping f parametrized by NN is called *discriminator*. The architecture of a discriminator adopts the same form of (1), given by

$$f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \mathbf{W}_L \cdot \sigma\left(\mathbf{W}_{L-1} \cdots \sigma\left(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1\right) \cdots + \mathbf{b}_{L-1}\right) + \mathbf{b}_L, \tag{8}$$

with the exception that $n_L = 1$, so that the output of a discriminator is a scalar. Accordingly, we define $\mathcal{F}_{\text{NN}}$ as a class of discriminators.

$$\begin{aligned} \mathcal{F}_{\text{NN}}(L, \tilde{n}, \kappa, K) = \Big\{ & f : \mathbb{R}^p \to \mathbb{R} \mid f \text{ takes the form in (8) with } L \text{ layers, } n_l\text{'s as the width of each layer,} \\ & n_L = 1, \|\mathbf{W}_i\|_\infty \le \kappa, \|\mathbf{b}_i\|_\infty \le \kappa, \sum_{i=1}^{L} \|\mathbf{W}_i\|_0 + \|\mathbf{b}_i\|_0 \le K, \text{ for } i = 1, 2, \cdots, L \Big\}. \end{aligned} \tag{9}$$

To differentiate the parameters of the generator g and the discriminator f, we use θ to denote and summarize the parameters ($\mathbf{W}$ and $\mathbf{b}$) of the generator and use ω to denote and summarize the parameters of the discriminator. We sometimes use the abbreviated notation f_ω and g_θ, or $f(\cdot; \omega)$ and $g(\cdot; \theta)$ in the paper instead of $f(\cdot; \mathbf{W}, \mathbf{b})$ and $g(\cdot; \mathbf{W}, \mathbf{b})$, in order to distinguish the neural network parameters for the generator and for the discriminator. With both the generator and discriminator parametrized by NN, when n iid copies of data $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ are available, the optimization problem in (7) is approximated by

$$(g_n^*, f_n^*) \in \arg\min_{g_\theta \in \mathcal{G}_{\text{NN}}} \max_{f_\omega \in \mathcal{F}_{\text{NN}}} \mathbb{E}_{\mathbf{Y} \sim \nu}\left[f(\mathbf{h}(g(\mathbf{Y}; \theta); \omega))\right] - \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{X}_i; \omega). \tag{10}$$

where $\mathbf{Y} \sim \nu$ represents that $\mathbf{Y}$ is a p-dimensional random vector with each element as an independent Uniform(0,1) random variable.

4. Statistical Theory of DS-WGAN

We discuss the statistical aspect of DS-WGAN in this section and the computational aspect of DS-WGAN in the next Section. In this section, we prove that the DS-WGAN framework can effectively learn distributions of a wide class of doubly stochastic Poisson distributions with jointly distributed random intensities, if the neural network architectures are appropriately chosen. Recall that the true distribution μ to learn is the distribution of the vector of interval counts $\mathbf{X} = (X_1, X_2, \ldots, X_p)$, in which each X_i is a doubly stochastic Poisson distributed random variable with a random rate Λ_i. The dependence structure within $(X_1, X_2, \ldots, X_p)$ is determined by the dependence structure within the unobserved vector of random rates $\mathbf{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_p)$. We do not assume any parametric structure on the joint distribution of $\mathbf{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_p)$, but to support the theory analysis, we need some regularity conditions. One assumption is that $\mathbf{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_p)$ has a smooth joint density $f_{\mathbf{\Lambda}}$. We summarize the assumption on the joint density $f_{\mathbf{\Lambda}}$ as follows.

ASSUMPTION 1. *The vector of random intensities $\mathbf{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_p)$ has a joint probability density function $f_{\mathbf{\Lambda}}$. The joint density function $f_{\mathbf{\Lambda}}$ adopts the following regularity conditions.*

a.) The support of $f_{\mathbf{\Lambda}}$ is compact and convex.

b.) The joint density $f_{\mathbf{\Lambda}}$ is bounded away from 0 on its support. That is, the infimum of $f_{\mathbf{\Lambda}}$ over its support is positive.

c.) The joint density $f_{\mathbf{\Lambda}}$ is differentiable and has uniformly bounded first order partial derivatives across its support.

The convexity assumption of the support facilitates the successful finding of a transformation from the random seed to the desired distribution. The bounded-away-from-zero assumption is standard in the theory literature for Wasserstein distance and optimal transport. The smoothness of the density function facilitates the analysis of the convergence rate for the generator, which has been standard in the non-parametric statistics literature. We next present the main result.

THEOREM 1. *Suppose that n copies of iid data are available and that the underlying joint density function $f_{\mathbf{\Lambda}}$ satisfies Assumption 1. Under appropriate specifications of the neural network architecture, let g_n^* be the generator that solves the DS-WGAN optimization problem as in (10). We have*

$$\mathbb{E}\, W(\mathbf{h}(g_n^*(\mathbf{Y})), \mu) \leq C \cdot n^{-\frac{1}{2+p}} \log^2 n, \tag{11}$$

in which C is a constant that is independent of n and involves only polynomial terms of p.

Proof of Theorem 1. The proof of Theorem 1 is provided in EC.1. □

We next discuss the implications of Theorem 1. Theorem 1 shows that with appropriately chosen neural network structures, for an arbitrary doubly stochastic Poisson model, if the joint distribution of the underlying stochastic intensities has a smooth density, the DS-WGAN can provide statistically consistent estimated generators. The estimated generator, compounded by the Poisson counts simulator, can simulate new data that is consistent with the true distribution under the Wasserstein distance measurement. The result in Theorem 1 also establishes a guaranteed upper bound on the convergence rate and its dependence on the dimension p. This is the first statistical theory on using neural networks to estimate and simulate general doubly stochastic Poisson process models.

The proof for the statistical theory of DS-WGAN utilizes the function approximation theory to analyze the statistical properties and convergence rate for neural-network based statistical estimator. Bai et al. (2018) and Chen et al. (2020) are two of the representative works in this area, and the statistical theory proved in our work is inspired by them. In the DS-WGAN framework, the simulator is composed of a generative neural network and a Poisson counts simulator. This composition brings a key challenge in establishing the statistical theory, which we overcome by introducing a coupling technique. Besides the statistical aspect, a general challenge associated with neural network models arises in the computational aspect, which is how to avoid local minimum in solving the optimization problem. This challenge is essentially presented in a number of non-convex optimizations including neural network based models. The optimization of non-convex neural network models has no general guarantees and is itself an arising research area. See Sun et al. (2020) for a new understanding of the global landscape of GAN-based models and references within. In the next subsection, we discuss the challenges from the computational aspect of DS-WGAN and provide solutions.

5. Challenges and Solutions in Model Estimation of DS-WGAN

In this section, we provide an algorithm to solve the model estimation optimization problem of DS-WGAN. We discuss and address a unique challenge arised in the estimation of DS-WGAN, compared to the model estimation of the classical WGAN.

Given data $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$, the model estimation of DS-WGAN is to estimate the neural network parameters θ and ω for the generator $g(\cdot;\theta)$ and the discriminator $f(\cdot;\omega)$ in the optimization problem (10). The associated optimization problem is a non-convex two-player minimax game problem. How to effectively perform model estimation (or training) for such problems attracts keen attention in the machine learning and deep learning area. Specifically, for the model estimation of the classical WGAN without the doubly stochastic feature, variants of stochastic gradient descent (SGD) routines are popular and shown to produce stable performance. See Gulrajani et al. (2017) for example. The model estimation of DS-WGAN also adopts the SGD routine.

In order to develop SGD based optimization algorithms, gradients of the objective function with respect to model parameters in the neural networks need to be effectively evaluated. Before proceeding into the optimization algorithm and discussing the challenge, we first specify the notation that is needed to represent the gradients. The gradient of a critical component $f(\mathbf{h}(g(\mathbf{Y};\theta);\omega))$ in the objective function with respect to the discriminator model parameters ω is given by $\nabla f_\omega(\mathbf{x};\omega)|_{\mathbf{x}=\mathbf{h}(g(\mathbf{Y};\theta))}$. The gradient of the critical component $f(\mathbf{h}(g(\mathbf{Y};\theta);\omega))$ with respect to the generator model parameters θ is given by

$$\begin{aligned}\nabla_\theta f(\mathbf{h}(g(\mathbf{Y};\theta));\omega) =& \nabla_{\mathbf{x}} f(\mathbf{x};\omega)|_{\mathbf{x}=\mathbf{h}(g(\mathbf{Y};\theta))} \cdot \nabla_\theta \mathbf{h}(g(\mathbf{Y};\theta)) \\ =& \nabla_{\mathbf{x}} f(\mathbf{x};\omega)|_{\mathbf{x}=\mathbf{h}(g(\mathbf{Y};\theta))} \cdot \nabla_{\mathbf{\Lambda}} \mathbf{h}(\mathbf{\Lambda})|_{\mathbf{\Lambda}=g(\mathbf{Y};\theta)} \cdot \nabla_\theta g(\mathbf{Y};\theta). \end{aligned} \tag{12}$$

This gradient is evaluated using backpropogation through the chain rule, that goes backward from the discriminator neural network, through the Poisson counts simulator, and to the generator neural network. The red dashed line in Figure 2 illustrates this gradient propagation evaluation of 12.

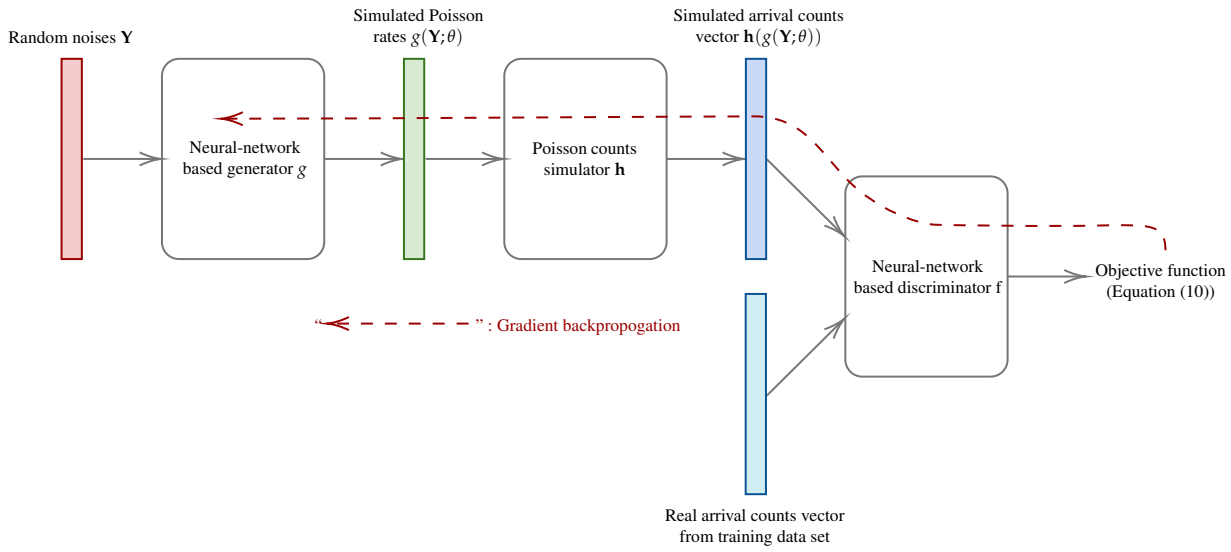


Figure 2 **Backpropogation diagram (red dashed line) on the gradient evaluation of $f(\mathbf{h}(g(\mathbf{Y};\theta);\omega))$ with respect to the parameters θ in the generator neural network g.**

For this gradient evaluation of $\nabla_\theta f(\mathbf{h}(g(\mathbf{Y};\theta));\omega)$ in equation (12), the product of three parts are involved. The first part $\nabla_{\mathbf{x}} f(\mathbf{x};\omega)|_{\mathbf{x}=\mathbf{h}(g(\mathbf{Y};\theta))}$ and the third part $\nabla_\theta g(\mathbf{Y};\theta)$ can be effectively computed by backpropagation (BP), which is standard in the literature and can be conveniently implemented in libraries such as PyTorch. The key challenge arises is the second part

$\nabla_{\mathbf{\Lambda}}\mathbf{h}(\mathbf{\Lambda})|_{\mathbf{\Lambda}=g(\mathbf{Y};\theta)}$. Specifically, for the Poisson counts simulator $\mathbf{h}$ as defined in (4), the input variable $\mathbf{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_p)$ and the output variable $\mathbf{h}(\mathbf{\Lambda}) = (M_1(\Lambda_1), M_2(\Lambda_2), \ldots, M_p(\Lambda_p))$ are both vectors of size $p \times 1$. The term $\nabla_{\mathbf{\Lambda}}\mathbf{h}(\mathbf{\Lambda})$ is given by a diagonal matrix of size $p \times p$:

$$\nabla_{\mathbf{\Lambda}} h(\Lambda) = \begin{bmatrix} \frac{\partial}{\partial \Lambda_1} M_1(\Lambda_1) & & \\ & \ddots & \\ & & \frac{\partial}{\partial \Lambda_p} M_p(\Lambda_p) \end{bmatrix}.$$

The challenge is in computing $\frac{\partial}{\partial \Lambda_1} M_1(\Lambda_1)$. Recall that $(M_1(t) : t \geq 0)$ is a unit-rate Poisson process, so $M_1(\Lambda_1)$ is a random function of a random variable Λ_1. The derivative $\frac{\partial}{\partial \Lambda_1} M_1(\Lambda_1)$ adopts a sample-path definition in the BP routine. In fact, because M_1 outputs discrete variables from $\{0, 1, 2, \cdots\}$ in each sample path, the derivative $\frac{\partial}{\partial \Lambda_1} M_1(\Lambda_1)$ is either zero or infinity.[1] Apparently using these sample path derivatives (zero or infinity) leads the gradient $\nabla_\theta f(\mathbf{h}(g(\mathbf{Y};\theta));\omega)$ to be meaningless, preventing the associated SGD algorithms to appropriately work. On the other hand, intuitively, when Λ_1 becomes larger, the output $M_1(\Lambda_1)$ on average becomes larger. To circumvent the challenge in taking derivative on a discrete function and to preserve the monotonicity, we provide an approximation to replace the derivative $\frac{\partial}{\partial \lambda} M_1(\lambda)$. That is,

$$\frac{\partial}{\partial \lambda} M_1(\lambda) \overset{\Delta}{\approx} 1 + \frac{M_1(\lambda) - \lambda}{2\lambda}.$$

First, the term on the right-hand-size does not vanish or explode. Second, this approximation can be supported in theory by a Gaussian approximation for Poisson distribution when the Poisson rate is not too small. Specifically, when λ is moderately large,

$$M_1(\lambda) = \lambda + \sqrt{\lambda}\, Z + O_p\Big(\frac{1}{\sqrt{\lambda}}\Big)$$

in which $Z \sim N(0,1)$ is a Gaussian random variable with mean zero and variance one. This approximation has been proved to be stable through the Berry-Esseen theorem and empirical results for Poisson distributions, when λ is moderately large ($\lambda > 20$). Using this approximation, the derivative can be defined using a sample path argument

$$\frac{\partial}{\partial \lambda} M_1(\lambda) \approx 1 + \frac{Z}{2\sqrt{\lambda}} \approx 1 + \frac{\sqrt{\lambda}\, Z + \lambda - \lambda}{2\lambda} = 1 + \frac{M_1(\lambda) - \lambda}{2\lambda}.$$

We note that this is not a formally rigorous justification, but instead, to replace the derivative with the provided approximation circumvent the challenge. Note that the derivative $\frac{\partial}{\partial \lambda} M_1(\lambda)|_{\lambda=\Lambda_1}$ is

[1] The need to take derivative of continuous functions with respect to discrete variables arise in the use of generative adversarial networks in text generation and variational autoencoder with discrete latent variable. A reparametrization trick using Gumbel-Softmax is discussed in Jang et al. (2016) and Maddison et al. (2016) to approximate the derivative of continuous functions with respect to discrete variables. The challenge in our case, differently, is to take derivative of discrete functions with respect to continuous variables.

used as an intermediate term in the chain rule computation of $\nabla_\theta f(\mathbf{h}(g(\mathbf{Y};\theta));\omega)$. This approximation is very convenient and simple to be integrated to the backpropagation framework built in prevalent packages such as PyTorch. The numerical experiments show that this approximation eventually leads to stable and successful model estimation.

Now that the challenge in computing the gradient with respect to the model parameters is addressed, we list the details an SGD algorithm for the model estimation as Algorithm 1 in EC.2. This algorithm integrates the use of gradient penalty to regularize the Lipschitz condition requirement on the discriminator. The algorithm iteratively updates the parameters of the generator and discriminator neural networks in an iterative manner.

6. Arrival Epochs Simulator

Given data $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$, denote $\hat{g}_n^*$ as the estimated generator returned by Algorithm 1. We can then generate iid copies of new data $\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2, \ldots$ by first generating random seed vectors $\mathbf{Y}_1, \mathbf{Y}_2, \ldots$, and then simply feeding them into the generator $\hat{g}_n^*$ and the Poisson counts simulator $\mathbf{h}$, obtaining

$$\tilde{\mathbf{X}}_1 = \mathbf{h}(\hat{g}_n^*(\mathbf{Y_1})), \tilde{\mathbf{X}}_2 = \mathbf{h}(\hat{g}_n^*(\mathbf{Y_2})), \ldots$$

Recall that each of $\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2, \ldots$ is intended to share the same joint distribution as $\mathbf{X}$, which records the arrival counts in the consecutive p time intervals in $[0, T]$. Some decision making problems not only require the simulation of arrival counts, but also exact arrival time epochs. To complete the DS-WGAN framework, we discuss two versions of *arrival epochs simulator* that take an interval count vector, say $\tilde{\mathbf{X}}_1 = (\tilde{x}_{11}, \tilde{x}_{12}, \ldots, \tilde{x}_{1p})$, as input and return a full sequence of arrival epochs in $[0, T]$. A basic requirement of the simulated full sequence of arrival epochs is that they preserve the interval counts information in each time interval. Figure 3 illustrates how the arrival epochs simulator is connected with the simulator of DS-WGAN.

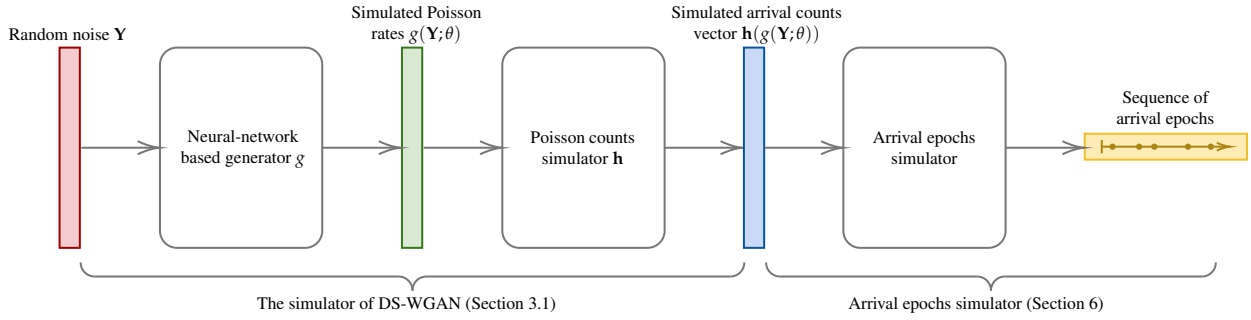


Figure 3 The simulator of DS-WGAN and the arrival epochs simulator.

For a given interval count vector, written as $(x_1, x_2, \ldots, x_p)$, the first arrival epochs simulator is set to generate and sort x_i copies of iid Uniform random variables supported on the interval $\left(\frac{(i-1)T}{p}, \frac{iT}{p}\right]$ for $i = 1, 2, \ldots, p$. This arrival epochs simulator is simple and easy to implement,

which corresponds to the piecewise constant (random) Poisson intensities discussed by Henderson (2003) and Oreshkin et al. (2016). However, if the underlying random intensity process lies in the continuous function space of $C[0,T]$, the aforementioned arrival epochs simulator may be unrealistic. In this case, even though it is impossible to recover the full continuous function of the realized intensity process by just observing the interval counts, we consider a *continuous piecewise linear arrival epochs simulator* to avoid the implausible discontinuities caused by the first arrival epochs simulator. Specifically, we fit a continuous piecewise linear intensity function given the interval count vector $(x_1, x_2, \ldots, x_p)$. The fitting procedure follows that provided in Zheng and Glynn (2017). Then, for a given time interval $\left(\frac{(i-1)T}{p}, \frac{iT}{p}\right]$, we randomly simulate and sort x_i copies of iid random variables according to a probability distribution supported on that time interval, whose density function is the normalized fitted intensity function on $\left(\frac{(i-1)T}{p}, \frac{iT}{p}\right]$. See EC.3 for the computational details of the arrival epochs simulators.

We illustrate the performance of DS-WGAN integrated by the piecewise linear arrival epochs simulator in the numerical experiment provided in Section 8.1.

7. What-if Simulation and Predictive Simulation using DS-WGAN

The modeling and simulation of arrival processes are key components for the performance evaluation and decision making in many stochastic systems. In this section, we first discuss that the DS-WGAN framework enables what-if simulation analysis. We explain how this what-if analysis supports performance evaluation and decision making for systems that do not yet exist. Second, we discuss the integration of DS-WGAN and other predictive models on the daily counts sequence, which is called *predictive simulation*.

To put the discussion into context, suppose that the iid data $\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_n$ were collected from exogenous user arrivals to a service system over n days of operations in the past. The users can be customers, patients, etc. Suppose that $p = 24$ and $\mathbf{X_i} = (X_{i1}, X_{i2}, \ldots, X_{i,p})$ represents the vector of hourly count on the i-th day in the data set. By feeding the data into DS-WGAN, a generator $\hat{g}_n$ can be estimated to learn the joint distribution of the underlying random intensities, including the variance and correlation structure. Consider a what-if planning decision in the service system for the future three months, with the objective to satisfy a certain level of user satisfaction, say, on average waiting time. For example, the question can be **what is an appropriate workers scheduling design if the expected number of daily user arrivals in the future three months is** 20% **larger, compared to that in the historical data?** To be more specific, this increase may be caused by the increased size of the underlying pool of potential users. As an example, for an app-based service at a growing stage, this increase indicates that the underlying pool of app users in the future three months has a larger size that is never seen before. Despite of the increased scale,

the time-of-day pattern and the correlation structure in the underlying random intensity process may preserve.[2]

To answer the what-if question and do the planning decision for future, there can be a need to simulate the vector of hourly counts (and then the arrival epochs) that reflect the 20% daily volume increase while realistically preserve the correlation structure of the underlying random intensity process. This need cannot be satisfied by re-sampling the arrivals from the historical data or by adding a multiplier once the hourly counts are re-sampled. The variance and correlation structure would not be appropriately preserved by re-sampling. In fact, the increased scale should be multiplied on the random intensity level, before the generation of Poisson counts. The DS-WGAN framework is a natural fit for this task. Specifically, to reflect the 20% scale increase, DS-WGAN can generate new data as

$$\tilde{\mathbf{X}}_1 = \mathbf{h}(1.2 \cdot \hat{g}_n(\mathbf{Y_1})), \tilde{\mathbf{X}}_2 = \mathbf{h}(1.2 \cdot \hat{g}_n(\mathbf{Y_2})), \dots \tag{13}$$

We further remark that compared to the standard WGAN framework, DS-WGAN not only exploits the doubly stochastic structure, but also conveniently facilitates what-if simulation analysis. The WGAN framework is not able to facilitate the aforementioned what-if simulation task.

We conclude this section by discussing the integration of DS-WGAN and other predictive models on the daily arrivals counts sequence, which is called *predictive simulation*. Many predictive models based on time series, machine learning, or deep learning target on the prediction of the future daily arrival counts. These models typically exploit statistical patterns arising from the daily counts sequence, such as seasonalities, long-term trend, etc. With a predicted future daily arrival count, there is a need to simulate the exact arrival epochs on that predicted day, which preserves the within day correlation structures and the time-of-day pattern. This naturally leads to the integration of the predictive model and the simulators in DS-WGAN.

8. Numerical Experiments

In this section, we test the performance of the DS-WGAN framework, using two sets of synthetic data (Section 8.1 and 8.2) and two sets of real data (Section 8.3 and Section 8.4). In Section 8.1, we also introduce run-through-queue experiments that demonstrate the performance of DS-WGAN from an operational performance point of view.

8.1. DSPP with Diffusion Intensity and Run-Through-Queue Experiments

In this subsection, we use a set of synthetic arrival data and run-through-queue experiments to test the performance of DS-WGAN.

[2] We note that the preserving of the correlation structure of the underlying random intensity process can also be interpreted as preserving the correlation structure of the *busyness factors*, a model that is introduced by Oreshkin et al. (2016).

8.1.1. Underlying Model for Synthetic Data: DSPP with Diffusion Intensity The underlying arrival process $N = (N(t) : t \in [0,T])$ is a doubly stochastic Poisson process (DSPP) whose intensity process $\lambda = (\lambda(t) : t \in [0,T])$ is given by a diffusion process called Cox-Ingersoll-Ross (CIR) process. Specifically, the intensity process is given by

$$\mathrm{d}\lambda(t) = \kappa(R(t) - \lambda(t))\mathrm{d}t + \sigma R(t)^{\alpha}\lambda(t)^{1/2}\mathrm{d}B(t), \tag{14}$$

where κ, σ, α are positive constants and $B(t)$ is a standard Brownian motion that is generated independent of other randomness in $(N(t) : t \in [0,T])$. The process $(R(t) : t \in [0,T])$ is a deterministic function of time that captures the time-of-day effect. To fully reflect the doubly stochastic nature, the initial value $\lambda(0)$ is set as $R(0) \cdot \mathrm{Gamma}(\beta,\beta)$, in which for any positive parameters a and b, $\mathrm{Gamma}(a,b)$ denotes a gamma-distributed random variable with mean a/b and variance a/b^2. This DSPP model with a non-stationary CIR process as the intensity process is an extension of the model provided in Section 3 of Zhang et al. (2014), in which they set $R(t)$ as a constant. As shown in Zhang et al. (2014), this DSPP model with CIR intensity process is aligned with a number of real data sets and well captures the time-varying covariance structure presented in the real data sets. We therefore generate synthetic data from this underlying model to test the performance of DS-WGAN. One major advantage of using synthetic data is that we have a known ground truth to compare with.

The model parameters are set as $\kappa = 0.2, \sigma = 0.4, \alpha = 0.3, \beta = 100$ and the time-of-day pattern function $(R(t) : t \in [0,T])$ is given in Figure 4, which roughly calibrates the time-of-day pattern in a call center data set used in Section 6.2 of L'Ecuyer et al. (2018). This center operates 11 hours a day so T is set as 11. For the data generation process, we simulate $n = 300$ replications of $N = (N(t) : t \in [0,T])$. In each replication, the underlying CIR process is simulated using the Euler–Maruyama discretization method (Kloeden and Platen (2013)) at a resolution of $\delta = 0.001$. The generation of Poisson process conditional on the simulated intensity process is through the thinning method (Lewis and Shedler (1979)). For the i-th replication of $N = (N(t) : t \in [0,T])$, where $i = 1,2,\ldots,n$, we collect the consecutive half-hour interval counts and record as $\mathbf{X}_i = (X_{i1}, X_{i2}, \ldots, X_{ip})$, in which $p = 22$.

The data set $\{\mathbf{X}_i\}_{i=1}^n$ is used as the input data for DS-WGAN. In the DS-WGAN framework, the parametrization of the generator neural network is given by $L = 4$, $\tilde{n} = (n_1, n_2, n_3, n_4) = (512, 512, 512, 22)$. The parametrization of the discriminator neural network is given by $L = 4$, and $\tilde{n} = (n_1, n_2, n_3, n_4) = (512, 512, 512, 1)$. The initialization of all the elements of the weight matrices $\mathbf{W}_l$'s of both the generator and discriminator networks are given by independent Gaussian random variables with mean 0 and variance 0.1. The vectors $\mathbf{b}_l$'s are initialized as constant 3. The

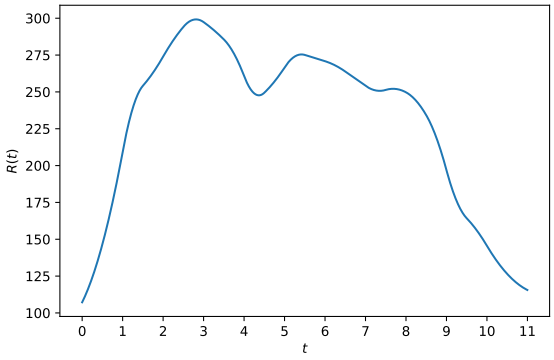


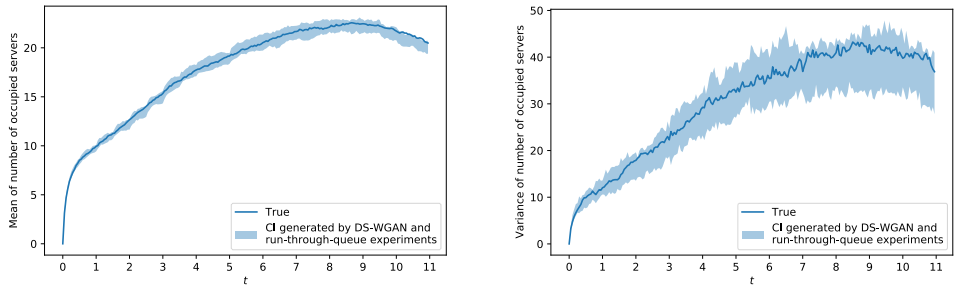
Figure 4 **The figure of $R(t)$ as a function of t**

neural network training is carried out with 50,000 iterations using the Adam optimizer (Kingma and Ba (2014)). The parameters for Adam optimizer are set as $\beta_1 = 0.5, \beta_2 = 0.9$ and the generator batch size is set as 256. The gradient penalty coefficient is set as 0.5. The discriminator is updated 10 times per generator iteration. An exponentially decaying learning rate is used for both the generator and discriminator, ranging from from 1e-4 to 1e-6. The training is implemented with PyTorch. The training takes about 50 minutes with one Nvidia K80 GPU.

With the trained DS-WGAN model, we can generate new iid copies of the vector of half-hour counts using the trained generator and the Poisson counts simulator. We represent a vector of half-hour counts generated by the trained DS-WGAN model as $\tilde{\mathbf{X}} = (\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_p)$ with $p = 22$. Given a copy of the generated vector $\tilde{\mathbf{X}}$, we can use the piecewise linear version of the arrival epochs simulator in Section 6 to generate a full sample path of arrival epochs on $[0, T]$. This full sample path generated by the DS-WGAN model is effectively a realization of a point process, which is denoted as $\tilde{N} = \{\tilde{N}(t) : t \in [0, T]\}$. Denote $\mathbf{X} = (X_1, X_2, \ldots, X_p)$ as the vector of half-hour counts that reflect the true underlying DSPP model. To examine whether the joint distribution of $\tilde{\mathbf{X}}$ matches that of $\mathbf{X}$, and further, whether the probability measure induced by $\tilde{N} = \{\tilde{N}(t) : t \in [0, T]\}$ matches that of the true underlying process $N = \{N(t) : t \in [0, T]\}$, we design the two run-through-queue experiments. Section 8.1.2 involves an infinite-server queue and performance measure as the distribution of number of people in system at different times. Section 8.1.3 involves a more realistic many-server queue and performance measure as the distribution of waiting time in system at different times. The general idea is that if the DS-WGAN generated arrival process $\tilde{N}$ and the true underlying process N have the same probability measure, the queueing performances for them when they are run through the same queueing system should resemble.

8.1.2. Run-through-queue Experiments: Infinite-server Queue Consider an infinite-server queue, in which the service time distribution is set as a log-normal distribution with mean 0.2 (in hours) and variance 0.1. We use the trained DS-WGAN model to generate $n = 300$ iid copies

of arrival process $\tilde{N} = \{\tilde{N}(t) : t \in [0, T]\}$. We run each of the generated arrival processes into the infinite-server queue, generating $n = 300$ independent realizations of the queueing operations on $[0, T]$. For results demonstration, we record the number of customers $V(t)$ in the system at time t, for which t ranges over the end of each minute. We compute the expectation and variance of $V(t)$ as a function of time using the $n = 300$ realizations of the queueing operations. We compute a 95% confidence interval (CI) for both the expectations and variances by repeating the experiments for 100 replications, and plot the confidence interval in light blue in Figure 5. We plot the true function value as blue solid polylines as a benchmark. The true function value is computed by independently simulating 3,000 replications of the queueing operations using the true underlying arrival process model, discretized at the resolution of $\delta = 0.001$.



(a) Mean of number of occupied servers (b) Variance of number of occupied servers.

Figure 5 Mean and variance of number of occupied servers at any time for the same Queue fed by the trained DS-WGAN model and the true underlying model. The blue solid polylines are the true values. The light blue areas are the confidence intervals given by DS-WGAN model and run-through-queue experiments.

Figure 5 shows that the queueing performance associated with the arrival process generated by the trained DS-WGAN model matches the true benchmark, suggesting the satisfying performance of DS-WGAN for this experiment. We remark that, the data set $\{\mathbf{X}_i\}_{i=1}^n$ provided to the DS-WGAN model only contains the half-hour interval counts data, without any information at higher resolution. Therefore DS-WGAN only captures the statistical information at the half-hour and higher resolution. On the contrary, the benchmark performance is computed using the true underlying continuous-time process. This experiment also supports the use of DS-WGAN to capture statistical features in the arrival processes that present at longer time scales (hour-hour) versus inter-arrival distribution time scale (0.3 minutes).

8.1.3. Run-through-queue Experiments: Many-server Queue Compared to an infinite-server queue, a many-server queue with finite number of servers is a more realistic setting for many applications, especially in the service systems. In those systems, because the service capacity is

not unlimited, customers usually have to wait. The probability distributions of customers waiting time are key performance measures, as they often represent customer satisfaction and largely impact customer retention. Driven by this need, we present a run-through-queue experiment using a many-server queue with two different staffing plans of servers. The purpose of this experiment is not to select an optimal staffing plan under some objective function, but is to show that for a queue with a given staffing plan, the customers waiting time performance when the DS-WGAN generated arrival process is fed into the queue is close to the true benchmark.

Consider a many-server queue, in which probability distribution of service time requirement for each customer is set as a log-normal distribution with mean 0.1 (in hours) and variance 0.1. The underlying customer arrival process, the synthetic data generation procedure, and the DS-WGAN model estimation procedure are the same as Section 8.1.1 and Section 8.1.2. The only difference is that the queueing staffing setting. Instead of infinite number of servers, we consider finite number of servers. We select two staffing plans for which the number of servers are changing with time. Let $s(t)$ denote the number of servers staffed in the queueing system as a function of time, with $t \in [0, 11]$. Consider staffing plans that change the number of servers staffed every hour hour. That is, $s(t)$ is a piecewise constant function, for which $s(t)$ is a constant, with the value denoted as $\{s_i\}$, on the time interval $[(i-1)*0.5, i*0.5)$, $i = 1, 2, \ldots, 22$. To set the specific value of s_i's as corresponding to the customer arrivals, on the i-th time interval $[(i-1)*0.5, i*0.5)$, $i = 1, 2, \ldots, 22$, denote $R_i = \int_{(i-1)*0.5}^{i*0.5} R(t)\, dt$. The first staffing plan in consideration is given by a square-root formula

$$s_i = R_i \mathbb{E}(S) + \beta \sqrt{R_i \mathbb{E}(S)}$$

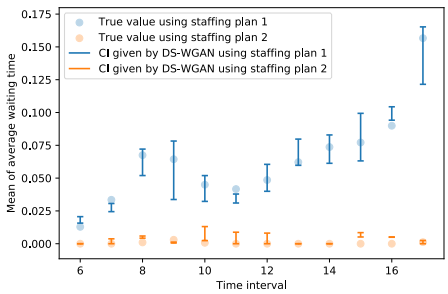
in which S in the random variable denoting the service time and $\beta = 1$. The second staffing plan is given by a formula considering the higher variability in arrivals for DSPP arrivals. This staffing plan is discussed in Zhang et al. (2014). Specifically, the second staffing plan is

$$s_i = R_i \mathbb{E}(S) + \beta \left(R_i \mathbb{E}(S)\right)^{\frac{1}{2}+\alpha}$$

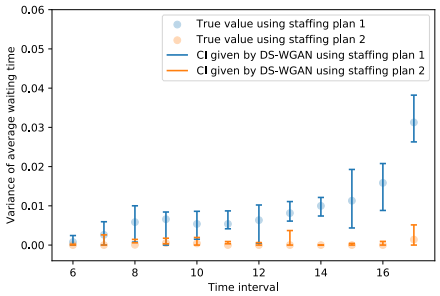
in which S in the random variable denoting the service time and $\beta = 1$. The parameter α is the variability parameter given in the random CIR arrival rate process, shown in equation (14).

We use the trained DS-WGAN model to generate $n = 300$ iid copies of arrival process $\tilde{N} = \{\tilde{N}(t) : t \in [0, T]\}$. We run each of the generated arrival processes into the many-server queue with the prescribed staffing plan, generating $n = 300$ independent realizations of the queueing operations on $[0, T]$. For results demonstration, for each queueing operation realization, we record the average waiting time $\bar{W}_i$ averaged over the waiting times of all customers that arrive at the system in that realization over the i-th time interval $[(i-1)*0.5, i*0.5)$ for a number of different

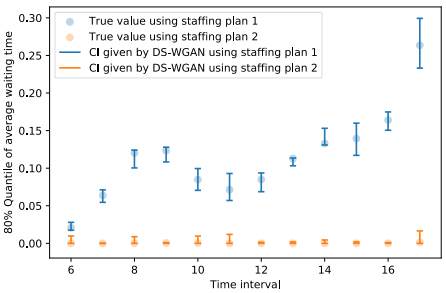
i's. Using the $n = 300$ independent realizations, we compute the expectation, variance, and 80% quantile of $\bar{W}_i$. We then repeat this procedure 100 times to construct 95% confidence interval (CI) for the expectations, variances, and quantiles, and plot the CI's in Figure 6. We also plot the true benchmark value of the expectation, variance and quantile for comparison. The true benchmark value is computed by independently simulating 3,000 replications of the queueing operations using the true underlying arrival process model, discretized at the resolution of $\delta = 0.001$.



(a) Mean of the average waiting time $\bar{W}_i$.



(b) Variance of the average waiting time $\bar{W}_i$.



(c) 80% quantile of the average waiting time $\bar{W}_i$.

Figure 6 Mean, variance and quantile of average waiting time for customers arrived in each time interval, fed by the trained DS-WGAN model and the true underlying model. The point scatters are the true value given by the underlying model. The vertical lines represents the CI's given by the DS-WGAN and run-through-queue experiments. The blue parts are the results using the first staffing plan, and the orange parts are the results when using the second staffing plan.

Figure 6 suggests that the two different staffing plans result in totally different waiting time performances. For both staffing plans, the results show that the arrival processes generated by the trained DS-WGAN model using $n = 300$ samples render close waiting time performances as compared to the true underlying benchmark.

8.2. Performance Test on DS-WGAN using DSPP Model by Oreshkin et al. (2016)

In this subsection, we test the performance of DS-WGAN on a synthetic arrival data set. The underlying arrival process model is a general doubly stochastic Poisson process (DSPP) model introduced by Section 5 in the seminal paper Oreshkin et al. (2016). The random intensity model is given by a *PGnorta* model named by Oreshkin et al. (2016), which has been shown to have

strong flexibility to match the variance and covariance structure presented in several real arrival data sets. We next provide a brief summary of this model, but refer to Section 5 of Oreshkin et al. (2016) for a detailed description of the model.

8.2.1. A Brief Summary: PGnorta Model Notation-wise, recall that the Poisson rates vector is $\mathbf{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_p)$, and the arrival count vector is $\mathbf{X} = (X_1, X_2, \ldots, X_p)$, with $X_i \sim \text{Poisson}(\Lambda_i)$. In the doubly stochastic PGnorta model introduced by Oreshkin et al. (2016), Λ_j is given by $\lambda_j \cdot B_j$, where λ_j is a deterministic base rate and B_j is a random busyness factor. The vector $(B_1, B_2, \ldots, B_j)$ has a joint distribution, in which the marginal distribution of B_j follows $\text{Gamma}(\alpha_j, \alpha_j)$, where $\text{Gamma}(\alpha_j, \alpha_j)$ represents a gamma distribution with mean one and variance $1/\alpha_j$. The correlation structure of $(B_1, B_2, \ldots, B_p)$ in the doubly stochastic PGnorta model is given by a normal copula. Specially, there is an underlying joint multi-normal distributed random vector $\mathbf{Z} = (Z_1, \ldots, Z_p)$ with mean zero and covariance matrix $\mathbf{R}^Z$. For $j = 1, 2, \ldots, p$, $B_j = G_j^{-1}(\Phi(Z_j))$, where $G_j^{-1}(\cdot)$ is the inverse cumulative distribution function of $\text{Gamma}(\alpha_j, \alpha_j)$ and $\Phi(\cdot)$ is the standard normal cumulative density function.

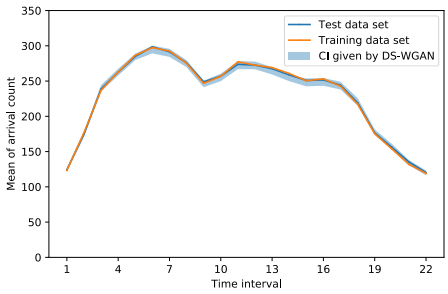
We summarize the parameters to be determined in this model as $\Theta = \left\{\mathbf{R}^Z, \{\alpha_j\}_{j=1}^p, \{\lambda_j\}_{j=1}^p\right\}$. To generate synthetic data using the doubly stochastic PGnorta Model, we set the model parameters to match a real data set in Section 6.2 of Oreshkin et al. (2016) with $p = 22$ where each time interval represents half hour.

8.2.2. Evaluation of DS-WGAN We simulate $n = 300$ iid copies of arrival count vectors using the calibrated doubly stochastic PGnorta model. These 300 iid copies of synthetic data are used as the training set. We describe and plot some summary statistics of this synthetic data set before discussing the training of DS-WGAN. We compute the marginal mean, marginal variance of arrival count for each time interval, as shown by the solid blue polylines in Figure 7(a) and 7(b). To illustrate the correlation structure of the arrival count vector, we adopt the concept of **correlation of past and future arrival count** used by Oreshkin et al. (2016). To prepare the notation for this concept, for an arrival count vector $\mathbf{X} = (X_1, X_2, \ldots, X_p)$, we define $Y_{j:j+d-1} = X_j + \cdots + X_{j+d-1}$ for $j \geqslant 1$ and $d \leqslant p - j + 1$. This is the total count of arrivals in d successive time intervals starting from j-th time interval. The correlation of past and future arrival count at the end of j-th time interval is defined as the correlation between the total count of arrivals in the first j periods ($Y_{1:j}$) and the total count of arrivals in the remaining $p - j$ periods ($Y_{j+1:p}$), and is denoted as $\text{Corr}(\mathbf{Y}_{1:j}, \mathbf{Y}_{j+1:p})$. We compute $\text{Corr}(\mathbf{Y}_{1:j}, \mathbf{Y}_{j+1:p})$ for the training data set as a function of j. The result is plotted by blue solid line in Figure 7(c).

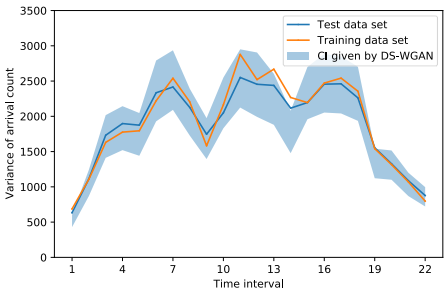
With the training set, we train the DS-WGAN model with the same neural network parametrization and training configurations as in Section 8.1. The trained generator of DS-WGAN

model is denoted as g_n^*. The arrival count vectors generated are thus $\mathbf{h}(g_n^*(\mathbf{Y}))$. Using the trained generator, we simulate 300 iid replications of $\mathbf{h}(g_n^*(\mathbf{Y}))$, and then compute the marginal mean, marginal variance of arrival count for each time interval, as well as the correlation of past and future arrival count. We repeat this generation process 100 times to compute a 95% confidence interval for each of the computed statistics. The confidence intervals are shown as light blue areas in Figure 7.

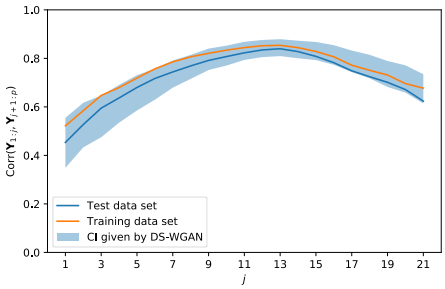
Independently from the training set, we generate 20,000 more iid copies of arrival count vectors from the true underlying doubly stochastic PGnorta model. We use them as the test data set. For the test data set, the marginal mean, marginal variance of arrival count for each time interval, as well as the correlation of past and future arrival count are computed and plotted as orange solid polylines in Figure 7.



(a) Mean of arrival count in each time interval.



(b) Variance of arrival count in each time interval.



(c) The correlation of past and future arrival counts.

Figure 7 Performance of DS-WGAN on synthetic data generated by Oreshkin et al. (2016). The solid blue polylines are the marginal mean, marginal variance and correlation of past and future arrival counts computed on test data set. The solid oranges polylines are for the training data set, and the light blue areas are the confidence intervals given by DS-WGAN model.

The DS-WGAN generated confidence intervals for the three sets of statistics (marginal mean, marginal variance, and correlation of past and future arrival count) cover that of the test and training data sets on almost every time interval. This experiment suggests that the DS-WGAN framework can perform well in terms of capturing these three sets of statistics, when the data

sample size is only $n = 300$. This data sample size requirement represents roughly a year length of data in many service systems operation, which is a moderate and reasonable requirement in the contexts of operations research and management science.

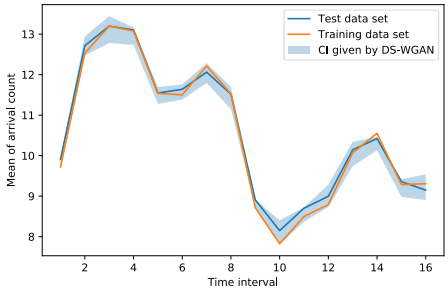
8.3. Test DS-WGAN on a Call Center Dataset

In this subsection, we use a real arrival data set from an Oakland call center to test the performance of DS-WGAN.

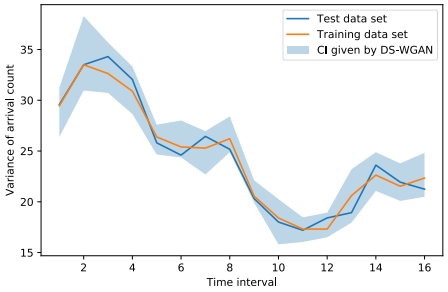
The data comes from an Oakland Public Works Call Center, in the city of Oakland in California, USA. This Oakland call Center opens 16 hours a day from 8 am to 8 pm, 7 days a week. We have arrival counts data over each hour ($p = 16$ time intervals per day) from January 1, 2013 to December 31, 2017, with holidays and weekends removed. Preliminary data analysis suggests that there is no significant day-of-week effect for weekdays, so we retain all the data from Monday to Friday. There's a total of 1,038 days retained.

Outliers are removed according to 2.5% and 97.5% percentiles for the data set. Specifically, the 2.5% and 97.5% percentiles for the arrival count in each time interval are computed. Any arrival count vector with at least one dimension smaller than the 2.5% percentile or larger than 97.5% percentile in that dimension is regarded as outlier. Based on this criterion, a total percentage of 12% of data are marked as outliers and removed. The cleaned data set is randomly split into a training data set and a test data set, with size ratio 2:1. (A total of 608 days in the training data set, and 304 days in the test data set.) The training data set is used as input data to the DS-WGAN framework. For the training of DS-WGAN, the neural network parametrization and training configurations are set the same as the setting in section 8.1, except for that the input data vector dimension p now is 16 and the gradient penalty coefficient is set to 5.

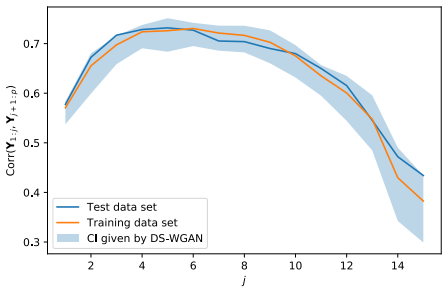
With the trained DS-WGAN, similar to Section 8.2, we compute the marginal mean, marginal variance of arrival count for each time interval, as well as the correlation of past and future arrival count. The computation is done for both the training data set and the test data set. The results are given in Figure 8. The orange solid lines are computed using the training data set, and the blue solid lines are for the test data set. With the trained DS-WGAN, we generate 304 iid copies of arrival count vector to compute the three sets of statistics (marginal mean, marginal variance, and correlation of past and future arrival count). We repeat the generation steps 100 times to construct a 95% confidence interval (CI). The CI results are given by the light blue areas in Figure 8. As the CI's on various statistics provided by the DS-WGAN generated arrival data cover the statistics computed from the test data, Figure 8 suggests a successful use of DS-WGAN on this call center data set.



(a) Mean of arrival count in each time interval.



(b) Variance of arrival count in each time interval.



(c) The correlation of past and future arrival counts.

Figure 8 Performance of DS-WGAN on a call center dataset.

8.4. Test DS-WGAN on a Bike Share Dataset

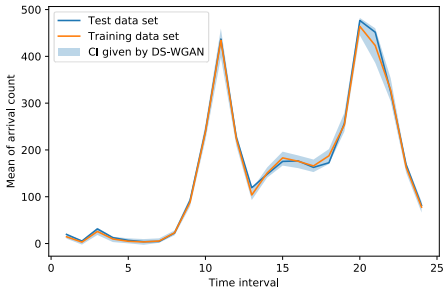
In this subsection, we use a real arrival data set from a bike-sharing system to illustrate the performance of DS-WGAN.

This arrival data set comes from a bike-sharing system named Capital Bikeshare located in Washington, D.C. The dataset is provided by Kaggle [https://www.kaggle.com/c/bike-sharing-demand/data] (Fanaee-T and Gama (2014)). This system operates 24 hours a day and customers can arrive to rent a bike at any time. The date range of the dataset is from January 20, 2011 to December 31, 2012. For the i-th day in this range, we have a 24-dimensional arrival count vector $\mathbf{x}^{(i)}$, whose j-th element represents the number of users who arrive and start to rent a bike in the j-th hour of this day. In this analysis, we focus on the hourly arrivals over the entire system and do not differentiate which specific location the customer arrivals at.

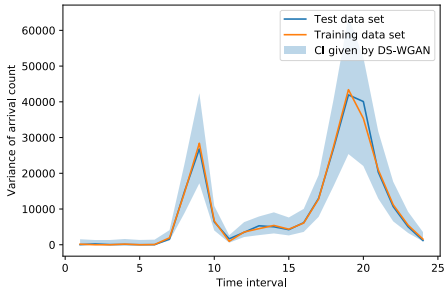
The data for weekends and holidays are removed. There is no significant day-of-week effect for weekdays suggested by preliminary data analysis, so all data from weekdays are retained. We note that out of all weekdays only about 65% of days are recorded and available in the original data set. A total of 311 days are retained. We randomly select 2/3 of these days (208 days) as the training data set, and the remaining 103 days as the test data set. The training data set is used as input data to the DS-WGAN framework. The gradient penalty coefficient is set to 10. The parametrization of the generator neural network is given by $L = 4$, $\tilde{n} = (n_1, n_2, n_3, n_4) = (512, 512, 512, 24)$. The parametrization of the discriminator neural network is given by $L = 4$, and

$\tilde{n} = (n_1, n_2, n_3, n_4) = (512, 512, 512, 24)$. The other training configurations are set the same as the setting in section 8.1.

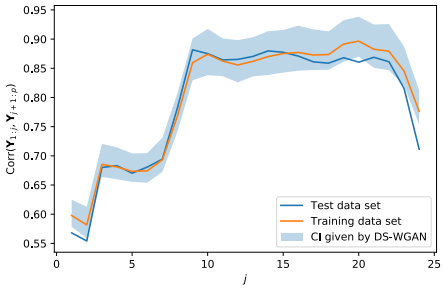
We compute the marginal mean, marginal variance of arrival count for each time interval, as well as the correlation of past and future arrival count. The computation is done for both the training data set and the test data set. The results are given in Figure 9. The orange solid lines are computed using the training data set, and the blue solid lines are for the test data set. With the trained DS-WGAN, we generate 103 iid copies of arrival count vector to compute the three sets of statistics (marginal mean, marginal variance, and correlation of past and future arrival count). We repeat the generation steps 100 times to construct a 95% CI. The CI results are given by the light blue areas in Figure 9. As the CI's on various statistics provided by the DS-WGAN generated arrival data cover the statistics computed from the test data, Figure 9 suggests a successful use of DS-WGAN on this bike share data set.



(a) Mean of arrival count in each time interval.



(b) Variance of arrival count in each time interval.



(c) The correlation of past and future arrival counts.

Figure 9 Performance of DS-WGAN on a bike share dataset.

9. Conclusion

We propose a framework named DS-WGAN that integrates the doubly stochastic (DS) structure and the Wasserstein generative adversarial networks (WGAN) to model, estimate, and simulate a wide class of arrival processes with non-stationary and stochastic arrival rates. We discuss and solve challenges in the statistical and computational aspects of the DS-WGAN framework. We

demonstrate through numerical experiments with synthetic and real data sets the performance of DS-WGAN, both from a statistical perspective and from an operational performance evaluation perspective. Numerical experiments suggest that the successful model estimation for DS-WGAN only requires a moderate size of data, which can be appealing in the contexts of operational management. Future work includes integrating DS-WGAN with decision-making optimization tools, including staffing and scheduling.

References

Arjovsky M, Chintala S, Bottou L (2017) Wasserstein gan. *arXiv preprint arXiv:1701.07875* .

Avramidis AN, Deslauriers A, L'Ecuyer P (2004) Modeling daily arrivals to a telephone call center. *Management Science* 50(7):896–908.

Bai Y, Ma T, Risteski A (2018) Approximability of discriminators implies diversity in gans. *International Conference on Learning Representations*.

Brown L, Gans N, Mandelbaum A, Sakov A, Shen H, Zeltyn S, Zhao L (2005) Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American statistical association* 100(469):36–50.

Cen W, Herbert EA, Haas PJ (2020) Nim: modeling and generation of simulation inputs via generative neural networks. *2020 Winter Simulation Conference (WSC)* (IEEE).

Channouf N, L'Ecuyer P (2012) A normal copula model for the arrival process in a call center. *International Transactions in Operational Research* 19(6):771–787.

Chen M, Liao W, Zha H, Zhao T (2020) Statistical guarantees of generative adversarial networks for distribution estimation. *arXiv preprint arXiv:2002.03938* .

Cox DR, Smith WL (1954) On the superposition of renewal processes. *Biometrika* 41(1-2):91–99.

Du N, Dai H, Trivedi R, Upadhyay U, Gomez-Rodriguez M, Song L (2016) Recurrent marked temporal point processes: Embedding event history to vector. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1555–1564.

Fanaee-T H, Gama J (2014) Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence* 2(2-3):113–127.

Glynn P (2014) Perspectives on traffic modeling. *Markov Lecture* .

Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC (2017) Improved training of wasserstein gans. *Advances in neural information processing systems*, 5767–5777.

Henderson SG (2003) Estimation for nonhomogeneous poisson processes from aggregated data. *Operations Research Letters* 31(5):375–382.

Ibrahim R, Ye H, L'Ecuyer P, Shen H (2016) Modeling and forecasting call center arrivals: A literature survey and a case study. *International Journal of Forecasting* 32(3):865–874.

Jang E, Gu S, Poole B (2016) Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* .

Kim SH, Whitt W (2014) Choosing arrival process models for service systems: Tests of a nonhomogeneous poisson process. *Naval Research Logistics (NRL)* 61(1):66–90.

Kingma D, Ba J (2014) Adam: A method for stochastic optimization. *International Conference on Learning Representations* .

Kloeden PE, Platen E (2013) *Numerical solution of stochastic differential equations*, volume 23 (Springer Science & Business Media).

Koole G (2013) Call center optimisation. *MG books*, Amsterdam.

Lewis PW, Shedler GS (1979) Simulation of nonhomogeneous poisson processes by thinning. *Naval research logistics quarterly* 26(3):403–413.

L'Ecuyer P, Gustavsson K, Olsson L (2018) Modeling bursts in the arrival process to an emergency call center. *2018 Winter Simulation Conference (WSC)*, 525–536 (IEEE).

Maddison CJ, Mnih A, Teh YW (2016) The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712* .

Mei H, Eisner JM (2017) The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in Neural Information Processing Systems*, 6754–6764.

Nelson BL (2016) Some tactical problems in digital simulation for the next 10 years. *Journal of Simulation* 10(1):2–11.

Oreshkin BN, Réegnard N, L'Ecuyer P (2016) Rate-based daily arrival process models with application to call centers. *Operations Research* 64(2):510–527.

Steckley SG, Henderson SG, Mehrotra V (2005) Performance measures for service systems with a random arrival rate. *Proceedings of the Winter Simulation Conference, 2005.*, 10–pp (IEEE).

Steckley SG, Henderson SG, Mehrotra V (2009) Forecast errors in service systems. *Probability in the Engineering and Informational Sciences* 23(2):305.

Sun R, Fang T, Schwing A (2020) Towards a better global loss landscape of gans. *Advances in Neural Information Processing Systems* 33.

Villani C (2008) *Optimal transport: old and new*, volume 338 (Springer Science & Business Media).

Wang R, Jaiwal P, Honnappa H (2020) Estimating stochastic poisson intensities using deep latent models. *arXiv preprint arXiv:2007.06037* .

Whitt W (2002) *Stochastic-process limits: an introduction to stochastic-process limits and their application to queues* (Springer Science & Business Media).

Zhang X, Hong LJ, Zhang J (2014) Scaling and modeling of call center arrivals. *Proceedings of the Winter Simulation Conference 2014*, 476–485 (IEEE).

Zheng Z, Glynn PW (2017) Fitting continuous piecewise linear poisson intensities via maximum likelihood and least squares. *2017 Winter Simulation Conference (WSC)*, 1740–1749, URL http://dx.doi.org/10.1109/WSC.2017.8247912.

Appendix

EC.1. Proof of Theorem 1

First note that g_n^* is the optimizer of the following optimization problem.

$$(g_n^*, f_n^*) \in \arg\min_{g_\theta \in \mathcal{G}_{\text{NN}}} \max_{f_\omega \in \mathcal{F}_{\text{NN}}} \mathbb{E}_{\mathbf{Y}\sim\nu}\left[f(\mathbf{h}(g(\mathbf{Y};\theta);\omega))\right] - \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_i;\omega). \tag{EC.1}$$

We need to provide an upper bound for $W(\mathbf{h}(g_n^*(\mathbf{Y})), \mu)$ in which for two arbitrary p-dimensional probability measures ν_1 and ν_2, the Wasserstein distance $W(\nu_1, \nu_2)$ can be written as

$$W(\nu_1, \nu_2) = \sup_{\|f\|_L \le 1} \mathbb{E}_{\mathbf{X}_1\sim\nu_1}[f(\mathbf{X}_1)] - \mathbb{E}_{\mathbf{X}_1\sim\nu_2}[f(\mathbf{X}_2)],$$

in which $\|f\|_L \le 1$ represents that f is a 1-Lipschitz function. Because the true distribution has a doubly stochastic structure, we have the following equal-in-distribution relationship

$$\mu \overset{\mathcal{D}}{=} \tilde{\mathbf{h}}(\tilde{\mathbf{\Lambda}}),$$

where $\tilde{\mathbf{h}}$ denotes a Poisson counts simulator as defined in (4) that is independent of $\mathbf{h}$ in terms of the source of randomness. The random vector $\tilde{\mathbf{\Lambda}}$ is defined to follow the true distribution of the underlying stochastic intensity vector, and the randomness in the random intensity $\tilde{\mathbf{\Lambda}}$ is independent of any other randomness. Meanwhile, in this proof, we set $\mathbf{\Lambda}$ as a random vector that is independent of $\mathbf{h}$ and is jointly distributed with $g_n^*(\mathbf{Y})$ such that the Wasserstein distance between $g_n^*(\mathbf{Y})$ and $\mathbf{\Lambda}$ is achieved by

$$W(g_n^*(\mathbf{Y}), \mathbf{\Lambda}) = \mathbb{E}[\|g_n^*(\mathbf{Y}) - \mathbf{\Lambda}\|_2]. \tag{EC.2}$$

We then have the following inequalities,

$$\begin{aligned}
W(\mathbf{h}(g_n^*(\mathbf{Y})), \mu) &= W(\mathbf{h}(g_n^*(\mathbf{Y})), \tilde{\mathbf{h}}(\tilde{\mathbf{\Lambda}})) \\
&= \sup_{\|f\|_L \le 1} \mathbb{E}[f(\mathbf{h}(g_n^*(\mathbf{Y})))] - \mathbb{E}[f(\tilde{\mathbf{h}}(\tilde{\mathbf{\Lambda}})] \\
&= \sup_{\|f\|_L \le 1} \mathbb{E}[f(\mathbf{h}(g_n^*(\mathbf{Y})))] - \mathbb{E}[f(\mathbf{h}(\mathbf{\Lambda}))] \\
&\le \sup_{\|f\|_L \le 1} \mathbb{E}[|f(\mathbf{h}(g_n^*(\mathbf{Y}))) - f(\mathbf{h}(\mathbf{\Lambda}))|] \\
&\le \mathbb{E}\|\mathbf{h}(g_n^*(\mathbf{Y})) - \mathbf{h}(\mathbf{\Lambda})\|_2 \\
&= \mathbb{E}\sqrt{\sum_{i=1}^{p}(\mathbf{h}_i(g_n^*(\mathbf{Y})) - \mathbf{h}_i(\mathbf{\Lambda}))^2} \\
&\le \sum_{i=1}^{p} \mathbb{E}\,|\mathbf{h}_i(g_n^*(\mathbf{Y})) - \mathbf{h}_i(\mathbf{\Lambda})|,
\end{aligned}$$

where the inequalities are from definition and standard Cauchy inequalities. A coupling approach is used in the third equation. Note that for any $i = 1, 2, \ldots, p$, the difference $\mathbf{h}_i(g_n^*(\mathbf{Y})) - \mathbf{h}_i(\mathbf{\Lambda})$ is given by

$$\mathbf{h}_i(g_n^*(\mathbf{Y})) - \mathbf{h}_i(\mathbf{\Lambda}) = M_i(g_{n,i}^*(\mathbf{Y})) - M_i(\mathbf{\Lambda}_i)$$

in which $(M_i(t) : t \geq 0)$ is a unit-rate Poisson process whose randomness is independent of any other source of randomness, $g_{n,i}^*$ represents the i-th element of the output of g_n^*, and $\mathbf{\Lambda}_i)$ is the i-th element of $\mathbf{\Lambda}$). Therefore, conditional on $g_n^*(\mathbf{Y}))$ and $\mathbf{\Lambda}$,

$$\mathbb{E}(|M_i(g_{n,i}^*(\mathbf{Y})) - M_i(\mathbf{\Lambda}_i)| \,|\, g_n^*(\mathbf{Y}), \mathbf{\Lambda}) = |g_{n,i}^*(\mathbf{Y}) - \mathbf{\Lambda}_1|$$

because of the independent and stationary increments property for unit-rate Poisson processes. As a result,

$$\begin{aligned} W(\mathbf{h}(g_n^*(\mathbf{Y})), \mu) &\leq \sum_{i=1}^{p} \mathbb{E}\,|\mathbf{h}_i(g_n^*(\mathbf{Y})) - \mathbf{h}_i(\mathbf{\Lambda})| \\ &\leq \sum_{i=1}^{p} \mathbb{E}\,|g_{n,i}^*(\mathbf{Y}) - \mathbf{\Lambda}_i| \\ &\leq \sqrt{p}\,\mathbb{E}\,\|g_n^*(\mathbf{Y}) - \mathbf{\Lambda}\|_2 \\ &= \sqrt{p}\,W(g_n^*(\mathbf{Y}), \mathbf{\Lambda}), \end{aligned} \tag{EC.3}$$

where the second last step follows the Chebyshev inequality and the last step follows (EC.2). Denote $\hat{\mu}_n$ as the empirical distribution of μ with n iid samples. Because of the doubly stochasticity, define $\hat{\mathbf{\Lambda}}_n$ as such that

$$\hat{\mu}_n \overset{\mathcal{D}}{=} \mathbf{h}(\hat{\mathbf{\Lambda}}_n).$$

By the same arguments for (EC.3), we have

$$W(\mathbf{h}(g_n^*(\mathbf{Y})), \hat{\mu}_n) \leq \sqrt{p} W(g_n^*(\mathbf{Y}), \hat{\mathbf{\Lambda}}_n) \tag{EC.4}$$

Then, define a generator $\tilde{g}_n^*$ as the optimizer of

$$(\tilde{g}_n^*, \tilde{f}_n^*) \in \operatorname*{arg\,min}_{g_\theta \in \mathcal{G}_{\mathrm{NN}}} \max_{f_\omega \in \mathcal{F}_{\mathrm{NN}}} \mathbb{E}_{\mathbf{Y}\sim\nu}\left[f(g(\mathbf{Y};\theta);\omega))\right] - \mathbb{E}_{\breve{}\sim\hat{\mathbf{\Lambda}}_n} f(\breve{};\omega). \tag{EC.5}$$

By applying Theorem 1 and Theorem 2 in Chen et al. (2020), we have the following lemma.

LEMMA EC.1. *Suppose that Assumption 1 holds. Set $\epsilon = n^{-\frac{1}{2+p}}$. There exists a generator network class $\mathcal{G}_{\mathrm{NN}}$ with layers $L = O(\log(1/\epsilon))$, max width $\tilde{n} = O(p\epsilon^{-p})$, a constant weight upper bound, and overall weights $K = O(p\epsilon^{-p}\log(1/\epsilon))$, and a discriminator network class $\mathcal{F}_{\mathrm{NN}}$ with layers $L = O(\log(n)/(2+p))$, max width $\tilde{n} = O(n^{p/(2+p)})$, a constant weight upper bound, and overall weights $K = O(n^{p/(p+2)}(\log n)/(2+p))$, such that*

$$\mathbb{E}\,W(\tilde{g}_n^*(\mathbf{Y}), \mathbf{\Lambda}) = O(n^{-\frac{1}{2+p}}\log^2 n) \tag{EC.6}$$

in which O hides polynomial terms in p.

The second last step in the proof of Lemma EC.1 is done by establishing

$$\mathbb{E}\, W(\tilde{g}_n^*(\mathbf{Y}), \mathbf{\Lambda}) \le \mathbb{E}\, W(\tilde{g}_n^*(\mathbf{Y}), \hat{\mathbf{\Lambda}}_n) + \mathbb{E}\, W(\hat{\mathbf{\Lambda}}_n, \mathbf{\Lambda}) = O(n^{-\frac{1}{2+p}} \log^2 n)$$

in which the approximation error and the statistical error are separately handled. Combining Lemma EC.1, (EC.3), (EC.4) and (EC.5), we have

$$\begin{aligned}
\mathbb{E}\, W(\mathbf{h}(g_n^*(\mathbf{Y})), \mu) &\le \mathbb{E}\, W(\mathbf{h}(g_n^*(\mathbf{Y})), \hat{\mu}_n) + \mathbb{E}\, W(\hat{\mu}_n, \mu) \\
&\le \mathbb{E}\, W(\mathbf{h}(\tilde{g}_n^*(\mathbf{Y})), \hat{\mu}_n) + \mathbb{E}\, W(\hat{\mu}_n, \mu) \\
&= \mathbb{E}\, W(\mathbf{h}(\tilde{g}_n^*(\mathbf{Y})), \mathbf{h}(\hat{\Lambda}_n)) + \mathbb{E}\, W(\hat{\mu}_n, \mu) \\
&\le \sqrt{p}\, \mathbb{E}\, W(\tilde{g}_n^*(\mathbf{Y}), \hat{\mathbf{\Lambda}}_n) + \sqrt{p}\, \mathbb{E}\, W(\hat{\mathbf{\Lambda}}_n, \mathbf{\Lambda}) \\
&= O(n^{-\frac{1}{2+p}} \log^2 n).
\end{aligned}$$

This concludes the proof.

EC.2. The model estimation algorithm of DS-WGAN

In this section we present the model estimation algorithm of DS-WGAN in Algorithm 1.

EC.3. Specifications of Two Versions of the Arrival Epochs Simulator

In this part, we give the specific formulation and simulation algorithm of the two versions of the arrival epochs simulator. Specifically the *piecewise constant arrival epochs simulator* and the *continuous piecewise linear arrival epochs simulator*. Both these two versions of arrival epochs simulator takes an arrival count vector $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ as input, which records the arrival counts in the consecutive p time intervals in $[0, T]$. The total number of arrivals within $[0, T]$ is $\sum_{i=1}^{p} x_i$. The arrival epochs simulator assigns an arrival epoch for each of these $\sum_{i=1}^{p} x_i$ arrivals.

Given the arrival count vector $\mathbf{x}$, the realization of the underlying random intensity process can be inferred. The main difference between the two versions of arrival epochs simulator is that the piecewise constant arrival epochs simulator approximates the realized intensity process as a piecewise constant intensity function, while the piecewise linear arrival epochs simulator approximates the realized intensity process as a piecewise linear intensity function. Once the intensity process is inferred, within each time interval, the exact arrival epochs are assigned according to the intensity process over that time interval. We next discuss the details of the two arrival epochs simulator, respectively in EC.3.1 and EC.3.2.

Algorithm 1 Model estimation of DS-WGAN

Require: The gradient penalty coefficient ζ, the batch size m, the number of iterations of the critic per generator iteration n_{critic}, Adam hyper-parameters α, β_1 and β_2.

Require: w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, \ldots, n_{\text{critic}}$ **do**
    Sample a batch of real arrival count vectors from the training data set, $\{\mathbf{x}^{(i)}\}_{i=1}^{m} \sim \hat{\mu}_n$.
    Sample a batch of random noises independently, $\mathbf{y}^{(i)} \sim \nu$, for $i = 1, 2, \ldots, m$.
    $\tilde{\mathbf{x}}^{(i)} \leftarrow \mathbf{h}(g(\mathbf{y}^{(i)}; \theta))$, for $i = 1, 2, \ldots, m$.
    Sample random number $\epsilon^{(i)} \sim U[0,1]$, for $i = 1, 2, \ldots, m$.
    $\hat{\mathbf{x}}^{(i)} \leftarrow \epsilon^{(i)} \mathbf{x}^{(i)} + (1 - \epsilon^{(i)}) \tilde{\mathbf{x}}^{(i)}$, for $i = 1, 2, \ldots, m$.
    Gradient penalty $\leftarrow \zeta \cdot \left(\left\| \nabla_{\hat{\mathbf{x}}^{(i)}} f\left(\hat{\mathbf{x}}^{(i)}; \omega\right) \right\|_2 - 1 \right)^2$
    $L_\omega \leftarrow \sum_{i=1}^{m} f\left(\tilde{\mathbf{x}}^{(i)}; \omega\right) - \sum_{i=1}^{m} f\left(\mathbf{x}^{(i)}; \omega\right) +$ Gradient penalty
    $\omega \leftarrow \omega +$ Adam $\left(L_\omega, \omega, \alpha, \beta_1, \beta_2\right)$
  end for
  Sample a batch of random noises independently, $\mathbf{y}^{(i)} \sim \nu$, for $i = 1, 2, \ldots, m$.
  $\tilde{\mathbf{x}}^{(i)} \leftarrow \mathbf{h}(g(\mathbf{y}^{(i)}; \theta))$, for $i = 1, 2, \ldots, m$.
  $L_\theta \leftarrow \sum_{i=1}^{m} f\left(\mathbf{h}\left(g\left(\mathbf{y}^{(i)}; \theta\right)\right); \omega\right)$
  $\theta \leftarrow \theta -$ Adam $\left(\nabla_\theta L_\theta, \theta, \alpha, \beta_1, \beta_2\right)$
end while

EC.3.1. Piecewise Constant Arrival Epochs Simulator

The piecewise constant arrival epochs simulator approximates the underlying rate function $\lambda(t), t \in [0, T]$ as a piecewise constant function, with the specific form

$$\lambda(t) = \lambda_j, \text{ for } \frac{(j-1) \cdot T}{p} < t \leq \frac{jT}{p}, \text{ for } j = 1, 2, \ldots, p,$$

where λ_j is a constant for j-th period $\left(\frac{(j-1)T}{p}, \frac{jT}{p}\right]$. Under this approximation, given the arrival count x_j in the j-th time interval, the arrival epochs are order statistics of x_j iid uniform random variables supported on the j-th time interval. Therefore, the arrival epochs simulator simulates x_j copies of iid uniform random variables supported on the j-th time interval, orders them, and returns them as the arrival epochs for the j-th time interval. Each time interval is handled separately and independently in the same pattern. We summarize the procedure for the piecewise constant arrival epochs simulator in Algorithm 2.

Algorithm 2 Piecewise constant arrival epochs simulator.

Require: The arrival count vector $\mathbf{x} = (x_1, x_2, \ldots, x_p)$, the length of the time range T.

Output: A list E that contains the sequential arrival epochs for all the arrivals in time range $[0, T]$.

1: $E \leftarrow \Phi$.
2: **for** $j = 1, 2, \ldots, p$ **do**
3: Simulate x_j iid copies of Uniform $\left(\frac{(j-1)T}{p}, \frac{jT}{p}\right)$, denoted as $\{s_{ji}\}_{i=1}^{x_j}$.
4: $E \leftarrow E \cup \{s_{ji}\}_{i=1}^{x_j}$.
5: **end for**
6: Sort E.
7: **return** E.

EC.3.2. Piecewise Linear Arrival Epochs Simulator

For the piecewise linear arrival epochs simulator, the underlying intensity function is approximated by a piecewise linear function, where $\{\lambda(t) : 0 \le t \le T\}$ is specified such that $\lambda(t)$ is linear over each of the time intervals $\left(0, \frac{T}{p}\right], \left(\frac{T}{p}, \frac{2T}{p}\right], \ldots, \left(\frac{(p-1)T}{p}, T\right]$. In particular, we use λ_i to denote the intensity at the boundaries of all the time intervals. That is, $\lambda_j = \lambda(\frac{jT}{p})$ for $j = 0, 1, \ldots, p$. The full piecewise linear function can be determined by the intensity values on the boundaries of all the time intervals. Specifically,

$$\lambda(t) = \lambda_{j-1} + \left(\frac{\lambda_j - \lambda_{j-1}}{T/p}\right)\left(t - \frac{(j-1)T}{p}\right), \text{ for } t_{j-1} \le t \le t_j, 1 \le j \le p.$$

Given the arrival count vector $\mathbf{x}$, the piecewise linear intensity function can be inferred using a maximum likelihood estimation approach introduced by Zheng and Glynn (2017). Specifically, this approach takes the arrival count vector as inputs and simultaneously returns the estimations for λ_j's for $j = 0, 1, \ldots, p$. The estimations are denoted as $\hat{\lambda}_j$'s for $j = 0, 1, \ldots, p$. Then, the full intensity function can be estimated through interpolation of the estimated intensity values $\hat{\lambda}_j$'s. Specifically,

$$\begin{aligned}\hat{\lambda}(t) &= \hat{\lambda}_{j-1} + \left(\frac{\hat{\lambda}_j - \hat{\lambda}_{j-1}}{T/p}\right)\left(t - \frac{(j-1)T}{p}\right) \\ &= \left(\frac{\hat{\lambda}_j - \hat{\lambda}_{j-1}}{T/p}\right) t + \hat{\lambda}_{j-1} j - \hat{\lambda}_j (j-1), \text{ for } t \in \left[(j-1)\frac{T}{p}, j\frac{T}{p}\right].\end{aligned}$$

With the estimated intensity function in hand, for each time interval, the exact arrival epochs can be simulated using the inversion method on each time interval. The detailed procedure of the piecewise linear arrival epochs simulator is given in Algorithm 3.

Algorithm 3 Piecewise linear arrival epochs simulator.

Require: The arrival count vector $\mathbf{x} = (x_1, x_2, \ldots, x_p)$, the length of the time interval T.

Output: A list E that contains the arrival epochs for all the arrivals in time interval $[0, T]$.

1: $E \leftarrow \Phi$

2: Compute the estimators $\hat{\lambda}_j$ for $j = 0, 1, \ldots, p$ using the MLE approach.

3: **for** $j = 1, 2, \ldots, p$ **do**

4: $m \leftarrow \frac{\hat{\lambda}_j - \hat{\lambda}_{j-1}}{T/p}$

5: $b \leftarrow \hat{\lambda}_{j-1} j - \hat{\lambda}_j (j-1)$

6: $t_0 \leftarrow (j-1)\frac{T}{p}$

7: $t_1 \leftarrow j\frac{T}{p}$

8: The estimated intensity function $\hat{\lambda}(t)$ over j-th interval is given by

$$\hat{\lambda}(t) = mt + b, \text{for } t \in [t_0, t_1].$$

9: Define the intergral of $\hat{\lambda}(t)$ over the time interval $[t_0, t]$ with t taking value within $[t_0, t_1]$ as

$$\begin{aligned} \Lambda(t) &\triangleq \int_{t_0}^{t} \hat{\lambda}(t)\, dt, \\ &= \frac{mt^2}{2} + bt - (\frac{mt_0^2}{2} + bt_0), \text{ for } t \in [t_0, t_1]. \end{aligned}$$

10: Compute $\Lambda(t_0) = 0$ and $\Lambda(t_1) = \frac{mt_1^2}{2} + bt_1 - (\frac{mt_0^2}{2} + bt_0)$.

11: Simulate x_j iid copies of Uniform $(\Lambda(t_0), \Lambda(t_1))$, denoted as $\{u_{ji}\}_{i=1}^{x_j}$.

12: $s_{ji} \leftarrow \Lambda^{-1}(u_{ji}) = \frac{1}{m}\left(-b + \sqrt{b^2 + 2m\left(\frac{mt_0^2}{2} + bt_0 + u_{ji}\right)}\right)$, for $i = 1, 2, \ldots, x_j$.

13: $E \leftarrow E \cup \{s_{ji}\}_{i=1}^{x_j}$.

14: **end for**

15: Sort E.

16: **return** E.