Infinity Learning: Learning Markov Chains from Aggregate Steady-State Observations

Jianfei Gao[1], Mohamed A. Zahran[2], Amit Sheoran[3], Sonia Fahmy[4], Bruno Ribeiro[5]
Department of Computer Science, Purdue University
305 N. University St, West Lafayette, IN 47907
{gao462[1], mzahran[2], asheoran[3]}@purdue.edu, {fahmy[4], ribeiro[5]}@cs.purdue.edu

Abstract

We consider the task of learning a parametric Continuous Time Markov Chain (CTMC) sequence model without examples of sequences, where the training data consists entirely of aggregate steady-state statistics. Making the problem harder, we assume that the states we wish to predict are unobserved in the training data. Specifically, given a parametric model over the transition rates of a CTMC and some known transition rates, we wish to extrapolate its steady state distribution to states that are unobserved. A technical roadblock to learn a CTMC from its steady state has been that the chain rule to compute gradients will not work over the arbitrarily long sequences necessary to reach steady state —from where the aggregate statistics are sampled. To overcome this optimization challenge, we propose ∞-SGD, a principled stochastic gradient descent method that uses randomly-stopped estimators to avoid infinite sums required by the steady state computation, while learning even when only a subset of the CTMC states can be observed. We apply ∞-SGD to a real-world testbed and synthetic experiments showcasing its accuracy, ability to extrapolate the steady state distribution to unobserved states under unobserved conditions (heavy loads, when training under light loads), and succeeding in difficult scenarios where even a tailor-made extension of existing methods fails.

Introduction

Can we learn a parametric sequence model given only aggregate statistics as training data? As machine learning expands into new applications, new learning paradigms emerge, such as learning a sequence model from a set of observations without any clear time order between them.

Traditional supervised and unsupervised learning methods are essentially tasked with problems that can be learned from examples (*interpolation*). In a host of key applications of parametric sequence models, we want to *extrapolate*, *i.e.*, take these aggregate observations and extrapolate them to a scenario *not observed* in the training data.

For instance, servers in the cloud collect system logs —aggregate statistics such as response-time distribution, queue length distribution— under light-load conditions. Under high-loads, however, these servers may disable statistics collection (logs) due to the potential performance penalty of logging (Newman 2017). Capacity planning requires knowing how the servers perform under medium to high load conditions, which requires extrapolated predictions of request loss probability and server response times from the collected light-load data.

Hence, in this work we consider the task of learning a parametric Continuous Time Markov Chain (CTMC) sequence model —with transition rate matrix $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})$ where $\boldsymbol{x}$ are known parameters but parameters $\boldsymbol{\theta}$ must be learned— without examples of sequences, where the training data consists entirely of aggregate steady-state statistics. Making the problem harder, we further assume that the states we wish to predict are unobserved in the training data. More specifically, given an inductive bias over the transition rates of a CTMC and some known transition rates, we wish to extrapolate its steady state distribution to states that are unobserved. We focus on the application of predicting failures in queuing systems under heavy loads —*e.g.*, predicting request loss rates in overloaded cloud services— with training data that contains only aggregate statistics of the system under light loads, and no observed losses. Traditionally, CTMCs are learned from observations of their transient (sequences given by transitions between states) not from observations of their steady state, even less so if only a subset of the state space is observable.

Remark 1. *Extrapolation v.s. generalization error: In our task we must make a distinction between* generalization error *—which is the error on unseen data that reduces with more training examples even without inductive biases— and* extrapolation error *(Marcus 1998) —which is a type of generalization error over unseen states and domains that does not reduce with more training data without the help of a modeling assumption. Our task is to learn a parametric model that is capable of extrapolation.*

Contributions. Our work introduces the general problem of learning a *parametric* CTMC from aggregate steady-state observations (frequencies) of part of the CTMC states, focusing on queueing systems as our application. We also introduce a novel method (∞-SGD) to learn parametric CTMCs from aggregate steady-state observations, which

work even if the observations are over a restricted set of states. Our approach, ∞-SGD, is a novel, theoretically principled, optimization approach that, among other things, uses randomly-stopped estimators (McLeish 2011). In our experiments ∞-SGD finds significantly better maximum likelihood estimates than the baselines in real testbed and synthetic scenarios, both for the training and test data. We also see that ∞-SGD can successfully extrapolate from training data under light queueing loads to predictions under heavy loads. We expect ∞-SGD to be a useful tool in applications that collect aggregate statistics but need to learn parametric CTMCs.

Preliminaries

Consider a stationary and ergodic Continuous-Time Markov Chain (CTMC) $\mathcal{Y} = (\mathbf{Y}_\tau)_{\tau \geq 0}$ over a finite state space $\mathbb{S}$, where $\mathbf{Y}_\tau$ is the state of the Markov chain at time τ. The CTMC is governed by Kolmogorov's Forward Equation

$$\frac{\partial}{\partial \tau} \mathbf{p}_{\boldsymbol{x},\boldsymbol{\theta}}(\tau)^\mathsf{T} = (\mathbf{p}_{\boldsymbol{x},\boldsymbol{\theta}}(\tau))^\mathsf{T} \boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{Q}(\boldsymbol{x};\boldsymbol{\theta})$ is a transition rate matrix parameterized by both $\boldsymbol{x}$ (a vector of observed parameters, *e.g.*, request rate) and $\boldsymbol{\theta}$ (a vector of hidden parameters), $\mathbf{p}_{\boldsymbol{x},\boldsymbol{\theta}}(\tau)$ is a column vector of dimension $|\mathbb{S}|$, with $\Pr[\mathbf{Y}_\tau = i] = \mathbf{p}_{\boldsymbol{x},\boldsymbol{\theta}}(\tau)_i$ as the probability of being at state $i \in \mathbb{S}$ at time $\tau \geq 0$, given that $\mathcal{Y}$ starts at state $j \in \mathbb{S}$ with probability $\Pr[\mathbf{Y}_0 = j] = \mathbf{p}(0)_j$.

The transition rate matrix $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})$ is such that for $i \neq j$, $(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))_{ij} \geq 0$ describes the rate of the process transitions from state i to state j. The diagonal $(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))_{ii}$ is such that each row of $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})$ sums to zero, irrespective of the values of $\boldsymbol{x}$ and $\boldsymbol{\theta}$. Because $\mathcal{Y}$ is stationary and ergodic, the solution to Equation (1) implies a unique steady state distribution $\boldsymbol{\pi}(\boldsymbol{x},\boldsymbol{\theta}) = \lim_{\tau\to\infty} \mathbf{p}_{\boldsymbol{x},\boldsymbol{\theta}}(\tau)$.

Parameterized transition rate matrix $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})$. We exemplify $\boldsymbol{Q}(\boldsymbol{x};\boldsymbol{\theta})$ with one of the simplest CTMCs: the birth-death process (BD). BD has two parameters: the request (birth) rate $\boldsymbol{x} = (\lambda)$ and the service (death) rate $\boldsymbol{\theta} = (\mu)$. The transition rate matrix is

$$\boldsymbol{Q}(\boldsymbol{x};\boldsymbol{\theta}) = \begin{bmatrix} -\lambda & \lambda & 0 & \cdots & 0 \\ \mu & -(\mu+\lambda) & \lambda & \cdots & 0 \\ 0 & \mu & -(\mu+\lambda) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & \cdots & -\mu \end{bmatrix},$$

where request rate λ is known but the service rate μ needs to be learned. In our work, $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})$ can be significantly more complex, as we only assume $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})$ is differentiable w.r.t. $\boldsymbol{\theta}$. More generally, we can have an $n \times n$ matrix

$$\boldsymbol{Q}(\boldsymbol{x};\boldsymbol{\theta}) = \begin{bmatrix} -\sum_{i\neq 1} f_{1,i}(\boldsymbol{x},\boldsymbol{\theta}) & f_{1,2}(\boldsymbol{x},\boldsymbol{\theta}) & \cdots & f_{1,n}(\boldsymbol{x},\boldsymbol{\theta}) \\ f_{2,1}(\boldsymbol{x},\boldsymbol{\theta}) & -\sum_{i\neq 2} f_{2,i}(\boldsymbol{x},\boldsymbol{\theta}) & \cdots & f_{2,n}(\boldsymbol{x},\boldsymbol{\theta}) \\ \vdots & \vdots & \ddots & \vdots \\ f_{n,1}(\boldsymbol{x},\boldsymbol{\theta}) & f_{n,2}(\boldsymbol{x},\boldsymbol{\theta}) & \cdots & -\sum_{i\neq n} f_{n,i}(\boldsymbol{x},\boldsymbol{\theta}) \end{bmatrix},$$

for some appropriate set of functions $\{f_{i,j}\}_{i,j}$ of $\boldsymbol{x}$ and $\boldsymbol{\theta}$ (whose image must be in $[0,\infty)$).

Learning task. Consider learning $\boldsymbol{\theta}$ from a set of steady-state observations from a subset $\mathbb{S}' \subseteq \mathbb{S}$ of the states of the CTMC $\mathcal{Y}$. That is, even though $\mathcal{Y}$ evolves over $\mathbb{S}$, the observations from states in $\overline{\mathbb{S}}' = \mathbb{S}\backslash\mathbb{S}'$ are unavailable to us —e.g., consider a system that disables statistics collection (logs) when it reaches a set of *system overload* states $\overline{\mathbb{S}}'$.

Training data: Our training data consists of M time windows from which we have observed aggregate steady state data: $\mathcal{D} = \{(\boldsymbol{x}_m, \boldsymbol{y}_m)\}_{m=1}^M$, where $y_{m,j} \equiv (\boldsymbol{y}_m)_j$ is the number of steady state observations of state $j \in \mathbb{S}'$ at the m-th time window.

Loss function: The minimum negative log-likelihood of the model must be conditioned on only observing states of $\mathbb{S}'$ in steady state (i.e., $\tau \to \infty$),

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \sum_{m=1}^{M} \mathcal{L}(\boldsymbol{y}_m, \lim_{\tau\to\infty} \mathbf{p}_{\boldsymbol{x},\boldsymbol{\theta}}(\tau)), \tag{2}$$

where

$$\mathcal{L}(\boldsymbol{y},\boldsymbol{\pi}) = -\sum_{j\in\mathbb{S}'} y_j \log\left(\frac{\boldsymbol{\pi}_j}{\sum_{j'\in\mathbb{S}} \boldsymbol{\pi}_{j'}}\right), \tag{3}$$

such that the denominator ensures the observations are conditioned on only observing states in $\mathbb{S}'$ —a detailed description of the math behind this conditional can be found in Meyer (1989).

In theory, we could optimize $\boldsymbol{\theta}$ in Equation (2) via gradient descent but the derivative of Equation (2) w.r.t. $\boldsymbol{\theta}$ requires computing the derivative of the steady state $\lim_{\tau\to\infty} \mathbf{p}_{\boldsymbol{x},\boldsymbol{\theta}}(\tau)$, which is challenging as our steady state distribution does not have a closed-form expression.

The identifyability of Q is irrelevant to our task: In our task, we wish to predict the steady-state distribution of unobserved states from samples from the steady state of observed states. Specifically, we wish to extrapolate those predictions such that we can predict these steady state distributions even when the observed parameters, $\boldsymbol{x}$ of $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})$ change. Because there are infinitely many $\boldsymbol{Q}$ that can give the same correct steady state distribution predictions (see Supplementary Material A1), it is irrelevant to us knowing whether we recovered the "true" $\boldsymbol{Q}$. In fact, in our formulation there is no notion that we can ever learn a "true" $\boldsymbol{Q}$. We only care if it gives the correct steady state distribution.

Next, we review the related work.

Related Work

Inverting an MC steady state. Bernstein and Sheldon (2016) is one of the most closely related works, showing an estimator for an existing optimization approach from econometrics, Conditional Least Squares (CLS) (Miller 1952; Van Der Plas and others 1983; Kalbfleisch and Lawless 1984), which can be used to learn a Markov chain from aggregate statistics. This approach, however, is not designed to learn a parametric model (our $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})$ needs derivatives w.r.t. $\boldsymbol{\theta}$) and thus, cannot extrapolate to unobserved states in the training data. Moreover, our Markov chain is not homogeneous across observation time windows, requiring $\boldsymbol{x}$ to also change, which conflicts with the assumptions in CLS.

Maystre and Grossglauser (2015) and Ragain and Ugander (2016) are the also closely related works, which learn the transition rates of a Plackett–Luce-type model CTMC from samples of its stationary distribution. In an earlier work, Kumar et al. (2015) learns a discrete-time Markov chain model similar to the Plackett–Luce's model in the context of Web navigation. These earlier works, however, make domain-specific assumptions on $\boldsymbol{Q}$ that make computing $\boldsymbol{\pi}$ from $\boldsymbol{Q}$ trivial. We consider a general parametric $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})$ that may have no trivial solution.

Hashimoto, Gifford, and Jaakkola (2016) uses far-apart observations of a MC to learn transition probabilities, and Pierson et al. (2018) uses cross-sectional data to learn a temporal model; these works are focused on specific diffusion processes. Our problem is also related to the more general problem of learning over distributions Szabó et al. (2016), which in our scenario requires a solution designed for the task.

Randomly stopped estimators have been used in unrelated machine learning tasks (Xu, Srivastava, and Sutton 2019) and (Filippone and Engler 2015), with significantly different tasks and estimators than ours. Applying randomly stopped estimators is mostly about proving that a specific estimator gives finite-variance estimates.

Queueing systems. Cloud computing has transformed IT operations and management by deploying services on commodity hardware in public or private data centers, saving millions of dollars in both capital and operational expenses (Armbrust et al. 2010; ETSI 2014). The savings in operational expenses can only be attained if the allocation of compute, memory, networking and storage resources scales based on the workload (Cao et al. 2018; Amazon 2019; Google Cloud 2019). A key problem in this elastic scaling is anticipating overload and failures in order to proactively allocate and initialize additional resources. This prediction needs to be done without sufficient data on overload and failures (Weiss and Hirsh 1998). Fortunately, several novel cloud computing services can be modeled by queueing systems (Yang et al. 2009; Khazaei, Misic, and Misic 2012). Existing approaches, however, require knowing the transition rate matrix rather than learning it from aggregate observations (as we do).

Learning Transition Rates from Aggregate Steady State Metrics

In this section, we will describe why a good parametric model of $\boldsymbol{Q}$ is key to learn a $\boldsymbol{\theta}^\star$ that can predict the steady state distribution of the states $\overline{\mathbb{S}}' = \mathbb{S} \backslash \mathbb{S}'$ that are not observed from the states that are observed $\mathbb{S}'$. We will then introduce a few naïve methods to learn $\boldsymbol{\theta}^\star$ from Equation (2) and show they are unsuitable for learning accurate CTMC transition rates, including an extension of BPTT. Finally, we will propose a novel approach to learn $\boldsymbol{\theta}^\star$ that is significantly more accurate and more computationally efficient than the naïve approaches.

For the ease of notation, sometimes we abbreviate transition rate matrix that $\boldsymbol{Q} \equiv \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})$ and we may denote $(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))_{ij}$ by q_{ij}.

The need for a good parametric model of $\boldsymbol{Q}$: Without tied parameters in $\boldsymbol{Q}$ through $\boldsymbol{\theta}$, the steady state distribution would be flexible enough to make $\boldsymbol{\pi}_i, \forall i \in \mathbb{S}'$, and $\boldsymbol{\pi}_j, \forall j \in \overline{\mathbb{S}}$, have arbitrarily different probabilities. This would make it impossible to correctly extrapolate the observed data and predict $\boldsymbol{\pi}_j$ for observations of states in $\overline{\mathbb{S}}'$. An example is provided in Supplementary Material A1.

Lagrangian multipliers (*e.g.*, Conditional Least Squares) are undesirable. To solve Equation (2), $t \gg 1$, we can add the condition $\boldsymbol{\pi}^T \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}) = 0$ as a Lagrangian multiplier as if $\boldsymbol{\pi}$ are extra learnable parameters. Then, the loss function is redefined as

$$-\sum_{j \in \mathbb{S}'} y_j \log \left(\lim_{t \to \infty} \frac{\boldsymbol{\pi}_j}{\sum_{j' \in \mathbb{S}'} \boldsymbol{\pi}_{j'}} \right) + \lambda \| \boldsymbol{\pi}^\mathsf{T} \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}) \|, \quad (4)$$

$\lambda > 0$, which is the Conditional Least Squares (Miller 1952) for a CTMC, a regularization applied the definition of a steady-state distribution. We found, however, that this approach is very challenging by design, since $\boldsymbol{\pi}$ is a function of $\boldsymbol{x}$ and $\boldsymbol{\theta}$, and $\boldsymbol{x}$ varies in the training data. Hence, the Lagrangian multiplier λ depends on the loss function (which is conditional) and on $\boldsymbol{x}$ and $\boldsymbol{\theta}$, a challenging task.

Moreover, if we assume a constant λ, the resulting approach needs to work as as a bi-level optimization procedure (Bhatnagar and Borkar 1998; Colson, Marcotte, and Savard 2007). In computing the derivatives of the loss w.r.t. $\boldsymbol{\theta}$, there is essentially no connection between the data ($\{y_j\}_{j \in \mathbb{S}'}$) and $\boldsymbol{\theta}$, which is the reason why the approach fails. Fixing these Lagrangian multiplier issues is future work.

Solution through Uniformization and Chain Rule

Our first step to a solution is to *uniformize* the Markov chain $\boldsymbol{Q}$ in order to transform the CTMC of Equation (1) into a discrete-time Markov chain (DTMC) with probability matrix $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))$ as described below (Jensen 1953). Hence, we will see that an approximation of the steady-state distribution can obtained by recursively applying $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))$, and the derivative of this recursive application of the transition probability can be obtained via chain rule.

Definition 1 (Uniformized Markov Chain)**.** *Let $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})$ be a stationary and ergodic CTMC. We define a set of Chapman-Kolmogorov equations representing the CTMC at the events (arrivals) of a Poisson process with rate $\gamma(\boldsymbol{x}, \boldsymbol{\theta}) > \max(-\mathit{diag}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})))$. The distribution after $t \geq 0$ of these events is*

$$\mathbf{p}^{(\mathit{events})}(t; \boldsymbol{x}, \boldsymbol{\theta})^\mathsf{T} = \mathbf{p}^{(\mathit{events})}(0)^\mathsf{T} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))^t, \quad (5)$$

where $\mathbf{p}^{(\mathit{events})}(0)$ is some initial distribution and $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})) = \boldsymbol{I} + \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})/\gamma(\boldsymbol{x}, \boldsymbol{\theta})$, where $\boldsymbol{I}$ is the identity matrix.

By construction, since $\mathcal{Y}$ is ergodic and has a steady state $\boldsymbol{\pi}$, the Markov chain described by $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))$ has the same steady state as the CTMC described by $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})$ (Jensen 1953), *i.e.*, for $\mathbf{p}_{\boldsymbol{x}, \boldsymbol{\theta}}(\tau)$ as described in Equation (1),

$$\begin{aligned} \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}) &= \lim_{\tau \to \infty} \mathbf{p}_{\boldsymbol{x}, \boldsymbol{\theta}}(\tau) = \lim_{t \to \infty} \mathbf{p}^{(\text{events})}(t; \boldsymbol{x}, \boldsymbol{\theta}) \\ &= \lim_{t \to \infty} (\mathbf{p}^{(\text{events})}(0)^\mathsf{T} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))^t)^\mathsf{T}. \end{aligned} \quad (6)$$

Note that $(\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^t)_{ij}$ is the probability that the CTMC starts at state i and reaches state j after $t \geq 0$ events of the Poisson process with rate $\gamma(\boldsymbol{x},\boldsymbol{\theta})$ given by Definition 1.

In what follows, we sometimes denote the probability matrix as $\boldsymbol{P} \equiv \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))$ and steady state distribution as $\boldsymbol{\pi} \equiv \boldsymbol{\pi}(\boldsymbol{x},\boldsymbol{\theta}) \equiv \boldsymbol{\pi}(\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})))$.

Chain rule to learn $\boldsymbol{\theta}$ from a steady-state approximation. Learning $\boldsymbol{\theta}^\star$ in Equation (2) through gradient descent can be approximated for a large enough value of $t^\star \gg 1$ through the chain rule (Werbos 1990). The derivative of the loss in Equation (2) is

$$\frac{\partial \mathcal{L}(\boldsymbol{y}, \mathbf{p}^{(\text{events})}(t^\star;\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k} = \sum_{\substack{i\neq j,\\ i,j\in\mathbb{S}}} \left(\frac{\partial \mathcal{L}(\boldsymbol{y}, \mathbf{p}^{(\text{events})}(0)^\mathsf{T} \boldsymbol{P}^{t^\star})}{\partial (\boldsymbol{P}^{t^\star})_{ij}} \frac{\partial (\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{t^\star})_{ij}}{\partial \boldsymbol{\theta}_k} \right), \quad (7)$$

In order to compute $\partial(\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})))^{t^\star}/\partial\boldsymbol{\theta}_k$, we have to recursively apply the chain rule, which leads to a backpropagation through time (BPTT)-style method.

BPTT challenges. Directly using BPTT, however, has both theoretical and practical barriers. The theoretical challenge is finding a large-enough value of $t^\star$ that allows $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{t^\star}$ to approximate the steady-state distribution $\boldsymbol{\pi}(\boldsymbol{x},\boldsymbol{\theta})$ for any assignment of $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})$ that our optimization might find. The computational challenge is both of computational resources and of numerical precision.

The following definition gives a divide-and-conquer aid to the computational challenge of calculating BPTT over $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{t^\star}$:

Definition 2 (Divide-and-Conquer BPTT (DC-BPTT))**.** *Assume $t^\star = 2^T$ for some $T > 1$. Rather than backpropagating over $t^\star$ time steps —which is difficult if $t^\star$ is large due to vanishing and exploding gradients—, we will use a divide-and-conquer approach to reduce the backpropagation steps to $\log_2 t^\star = T$, by noting that*

$$\boldsymbol{P}^{2^T} = ((\boldsymbol{P}^{2^{T-2}})^2)^2 \quad = (\cdots(\boldsymbol{P}^2)^2\cdots)^2. \quad (8)$$

That is, rather than multiplying an intermediate $\boldsymbol{P}^t$ by $\boldsymbol{P}$ to obtain $\boldsymbol{P}^{t+1}$, we multiply $\boldsymbol{P}^t$ by itself to obtain $\boldsymbol{P}^{2t}$.

Computing $\boldsymbol{P}^{t^\star}$, $t^\star = 2^T$ with $T > 1$, from Definition 2 is more computationally efficient than the naïve $t^\star$ multiplications $\boldsymbol{P}\cdots\boldsymbol{P}$ because the computation graph is a tree whose backpropagation paths from the root to the leaves give the same derivatives at the same tree height. Unfortunately, as we see in our experiments, DC-BPTT still fails in the most challenging tasks.

Solution via Infinity Learning

An alternative to BPTT is to dive deeper into the chain rule equations and look for mathematical equivalences. Rather than using BPTT to compute the gradient $\partial\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{t^\star}/\partial\boldsymbol{\theta}_k$, $k \in \mathbb{S}$, in Equation (7), we can make use of the following observation.

Lemma 1. *Let $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})$ be a K-state transition matrix and $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))$ be its uniformized Markov Chain. $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^t$ is the Markov chain after t steps where $t > 0$, then the gradients of $\boldsymbol{P}^t$ w.r.t. $\boldsymbol{\theta}_k$ is*

$$\nabla^{(t)}_{\boldsymbol{\theta}_k} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})) \equiv \frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^t}{\partial \boldsymbol{\theta}_k} = \sum_{l=1}^{t} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{t-l} \frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{l-1}, \quad (9)$$

where

$$\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k} = \sum_{ij} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} \frac{\partial q_{ij}(\boldsymbol{x},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_k}.$$

The proof is in the Supplementary Material B1. Because $\sum_{l=1}^{\infty} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{l-1}$ diverges for any valid $\boldsymbol{x}$ and $\boldsymbol{\theta}$, it is not obvious that Equation (9) converges to a unique fixed point for $t \to \infty$. In what follows we show that the gradient in Equation (9) exists and is unique as $t \to \infty$:

Proposition 1 (Infinite Gradient Series Simplification)**.** *Let $\boldsymbol{Q}$ be a K-state transition rate matrix of a stationary and ergodic MC. Equation* (9) *for $t \to \infty$, henceforth denoted $\nabla^{(\infty)}_{\boldsymbol{Q}} \boldsymbol{P}(\boldsymbol{Q}) \equiv \lim_{t\to\infty} \nabla^{(t)}_{\boldsymbol{Q}} \boldsymbol{P}(\boldsymbol{Q})$, **exists and is unique** and can be redefined as*

$$(\nabla^{(\infty)}_{\boldsymbol{Q}} \boldsymbol{P}(\boldsymbol{Q}))_{ij} \equiv \lim_{t\to\infty} \sum_{l=1}^{t} \left(\boldsymbol{P}^{t-l} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} \boldsymbol{P}^{l-1} \right) = \boldsymbol{\Pi} \sum_{l=0}^{\infty} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} \boldsymbol{P}^l, \quad (10)$$

where $\boldsymbol{\Pi}$ is a matrix whose rows are the steady state distribution $\boldsymbol{\pi}$. Note that the diagonal $i = j$ is trivial to compute but should be treated as a special case.

The proof in the Supplementary Material B2 shows that $\nabla^{(\infty)}_{\boldsymbol{Q}} \boldsymbol{P}(\boldsymbol{Q})$ converges because the term inside the sum converges to a matrix of zeros as $l \to \infty$. Using Proposition 1 it is easy to prove that Equation (9) converges as $t \to \infty$.

While Proposition 1 shows that $\nabla^{(\infty)}_{\boldsymbol{Q}} \boldsymbol{P}(\boldsymbol{Q})$ converges, evaluating the infinite sum in Equation (10) is challenging. Truncating the sum would make the gradient biased, deviating the fixed point solution of Equation (2). To circumvent the infinite sum in Equation (10), we propose ∞-SGD, a numerically stable stochastic gradient descent method that can optimize gradients with infinite sums —as long as the sum is a weakly convergent series. Our experiments show that ∞-SGD consistently outperforms BPTT in stability to hyperparameters in convergence rate, and in estimation accuracy.

Theorem 1 (Infinity Stochastic Gradient Descent (∞-SGD))**.** *Let $\boldsymbol{Q}$ be the transition rate matrix of a stationary and ergodic CTMC. Assume strictly positive values for the learnable parameters $\boldsymbol{\theta}^{(h)}$ at the h-th step of the optimization. Let $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}^{(h)}))$ be its uniformized transition probability matrix per Definition 1. Let $\mathcal{L}(\boldsymbol{y},\boldsymbol{\pi})$ be as in Equation* (3)*. Reparameterize $\tilde{\mathcal{L}}(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta}^{(h)}) = \mathcal{L}(\boldsymbol{y},\boldsymbol{\pi}(\boldsymbol{x},\boldsymbol{\theta}^{(h)}))$ as the loss function with*

respect to $\boldsymbol{\theta}^{(h)}$. Let $X^{(h)} \sim$ Geometric$(p^{(h)})$, $X^{(h)} \in \mathbb{Z}^+$, be an independent sample of a Geometric distribution with $p^{(h)} < \delta^{(h)}$, where $\delta^{(h)}$ is the spectral gap of $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}^{(h)}))$. Then, for $0 < \epsilon \ll 1$ and for all learnable parameters $\boldsymbol{\theta}$,

$$\boldsymbol{\theta}_k^{(h+1)} = \max\left(\boldsymbol{\theta}_k^{(h)} - \eta^{(h)} \nabla_{\boldsymbol{\theta}_k} \tilde{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(h)}}, \epsilon\right),$$

where

$$\nabla_{\boldsymbol{\theta}_k} \tilde{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = \sum_{ij} \sum_{mn} (\mathbf{p}^{(\text{events})}(0))_m \left.\frac{\partial \mathcal{L}(\boldsymbol{y}, \boldsymbol{\pi})}{\partial \boldsymbol{\pi}_n}\right|_{\boldsymbol{\pi}=\boldsymbol{\pi}(\boldsymbol{x},\boldsymbol{\theta})} \times \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta})_n \Gamma_{ijmn}(\boldsymbol{x}, \boldsymbol{\theta}) \frac{\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})_{ij}}{\partial \boldsymbol{\theta}_k}$$

with $h = 0, 1, \ldots$, where $\boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta})$ is the steady state distribution defined in Equation (6)*, $\eta^{(h)}$ is the learning rate with $\sum_{h=0}^{\infty} \eta^{(h)} = \infty$, $\sum_{h=0}^{\infty} \left(\eta^{(h)}\right)^2 < \infty$, and*

$$\Gamma_{ijmn}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{t=0}^{X^{(h)}} \left[\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial q_{ij}} \frac{\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))^t}{\mathbb{P}[X^{(h)} > t]}\right]_{mn}, \quad (11)$$

is a stochastic gradient descent method that minimizes Equation (2)*.*

The proof of the theorem is in the Supplementary Material B3. The main insight is the use of Proposition 1 to produce a randomly-stopped unbiased estimator. The requirement in Theorem 1 that $p^{(h)} < \delta^{(h)}$ comes from a loose bound, *i.e.*, in practice $p^{(h)}$ can be relatively large (larger than the spectral gap) as our empirical results show —*e.g.*, all of our empirical results use the constant $p^{(h)} = 0.1, \forall h$. We have also tested some experiments with $p^{(h)} = 0.01$, which works as well as $p^{(h)} = 0.1$ (see Supplementary Material C4). As it is application-dependent, the value of $p^{(h)}$ should be seen as a hyperparameter. In what follows we introduce our empirical results.

Results

In this section, we contrast the accuracy and convergence of ∞-SGD (Theorem 1) against DC-BPTT (Definition 2) and find that ∞-SGD is more stable and consistently learns more accurate models. The primary application of our experiments is predicting *request loss rates in a queueing system* from data that has no observed losses, under the following conditions: (a) we learn $\boldsymbol{\theta}^\star$ of Equation (2) as a function of known request rate $\boldsymbol{x}^{\text{light}} \in \Lambda^{\text{light}}$ under light load (no losses) in the training data, and predict $\boldsymbol{\pi}^{\text{heavy}}$, the steady state request loss rates under heavy loads in the test data (out-of-sample extrapolation), where $\boldsymbol{\pi}^{\text{heavy}}$ is such that $\left(\boldsymbol{\pi}^{\text{heavy}}\right)^\mathsf{T} \boldsymbol{Q}(\boldsymbol{x}^{\text{heavy}}; \boldsymbol{\theta}^\star) = 0$ with $\boldsymbol{x}^{\text{heavy}} > \max(\Lambda^{\text{light}})$; moreover, (b) only part of the state space is observed in the training data, $\mathbb{S}' \subset \mathbb{S}$, and we wish to predict the steady state probability of the unobservable states $\overline{\mathbb{S}}' = \mathbb{S} \backslash \mathbb{S}'$.

Baseline method. Due to the absence of methods on parametric inference of CTMCs from steady-state observations, our main baseline is the DC-BPTT of Definition 2. In most of our simulations, we set $t^\star = 128 = 2^7$ and $\mathbf{p}^{(\text{events})}(0) = \mathbf{1}^\mathrm{T}/|\mathbb{S}|$ throughout all our experiments.We also tested BPTT without divide and conquer but find the optimization unstable due to the long backpropagation paths. We tested $t^\star \in \{16, 128\}$ and found that smaller values of $t^\star$ are easier to optimize but —as expected— generally produce worse approximations of the steady state for heavy loads.

Infinity learning. Our experiments also test our proposed approach, ∞-SGD, with $X^{(h)} \sim$ Geometric(p) of Theorem 1, where p is a constant success probability, *i.e.*, $\mathbb{E}[X] = 1/p$. In most of our experiments, $p = 0.1$, that is, on average we consider only the first ten terms in the sum of Equation (10). Contrast, ∞-SGD's 10 summation terms with matrix powers that need no chain rule, with the baseline DC-BPTT approach (Definition 2) where $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))^{128}$ needs to be computed together with a chain rule to compute gradients over the matrix multiplications. It is no surprise that ∞-SGD is a more stable optimization method (no vanishing or exploding gradients); interestingly, ∞-SGD also works well on the tested slow-mixing CTMCs, while baseline methods like DC-BPTT fail in these scenarios (see Supplementary Material C5).

Relaxing the parametric model. In some of our experiments, we will construct transition rate matrix $\boldsymbol{Q}'(\boldsymbol{x}, \boldsymbol{\theta}, \tilde{\boldsymbol{Q}}) = \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}) + \tilde{\boldsymbol{Q}}$ with an $\alpha \|\tilde{\boldsymbol{Q}}\|_2^2$ regularization penalty, $\alpha > 0$, where $\tilde{\boldsymbol{Q}}$ is an additional non-parametric learnable matrix s.t. $\tilde{\boldsymbol{Q}}_{ij} = 0$ whenever $(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))_{ij} \neq 0$, otherwise $\tilde{\boldsymbol{Q}}_{ij}$ is a learnable parameter of our model. This allows some uncertainty on the form of our parametric models. It also allows us to learn the parameters through an interpolation between parametric and non-parametric CTMC models.

The regularization term $\alpha \|\tilde{\boldsymbol{Q}}\|_2^2$ is added to the negative log-likelihood loss in Equation (2) to ensure that we can control how much flexibility we want. With small values of α, we are testing how overparameterization, *i.e.*, having too many extra parameters in $\boldsymbol{Q}'$, affects learning and generalization. Our experiments show that $\alpha \gg 1$ gives the best results, *i.e.*, the correct parametric model works best. We also see that $\alpha \approx 1$ still gives competitive results (refer to Supplementary Material C3), showing that some model flexibility is tolerable. In contrast, we see that $\alpha = 0.1$ tends to significantly hurt our ability to extrapolate queue losses in the test data.

Testbed Experiments

We now contrast DC-BPTT against ∞-SGD in a real-world testbed emulating a Voice-over-LTE (VoLTE) system in a wireless cellular network. The testbed is configured as a single server with a waiting queue of size $K = 20$. The training data (86 time windows) is generated under light loads (with mean 7.7 and median 3 call losses) and the test data (137 time windows) under heavy loads (with mean 135.2 and median 254 call losses). *Moreover, we also restrict the observations in the training data, $\mathbb{S}' \subset \mathbb{S}$, to queue sizes one and two, estimated from the request processing delays collected at the clients.* We define $\boldsymbol{Q}(\boldsymbol{x}; \boldsymbol{\theta})$ symbolically using

Table 1: [MAPE] Simulation results showing MAPE/100 ($\langle$Mean Absolute Error$\rangle$/$\langle$true value$\rangle$) errors between predicted steady state and ground-truth for failure states in test data (heavy load). Training data collected under light loads and restricted observed states (queues zero and one). Mixing rates are determined by the spectral gaps δ_n observed in training data over multiple time windows ($n = 1, \ldots, 50$). With 95% confidence intervals.

	δ_n (spectral gap)	DC-BPTT $t^\star = 16$	DC-BPTT $t^\star = 128$	∞-SGD ($p = 0.1$)
Testbed Emulation (Upper Trig.)	N/A	$1.43 \times 10^{1} \pm 0.00$	$1.88 \times 10^{1} \pm 0.00$	$\mathbf{9.33 \times 10^{-1} \pm 8.91 \times 10^{-2}}$
M/M/1/K (fast-mix)	[0.022, 0.043]	$2.04 \times 10^{-1} \pm 2.86 \times 10^{-4}$	$1.32 \pm 4.94 \times 10^{-3}$	$\mathbf{1.18 \times 10^{-2} \pm 8.01 \times 10^{-3}}$
M/M/1/K (slow-mix)	[0.005, 0.008]	$8.91 \times 10^{3} \pm 2.02 \times 10^{2}$	$6.68 \times 10^{1} \pm 7.61$	$\mathbf{8.88 \times 10^{-1} \pm 1.48}$
M/M/m/$m + r$	[0.013, 0.024]	$4.11 \times 10^{-1} \pm 4.47 \times 10^{-2}$	$4.01 \times 10^{-1} \pm 8.86 \times 10^{-2}$	$\mathbf{1.52 \times 10^{-1} \pm 8.50 \times 10^{-2}}$
M/M/Multiple/K	[0.068, 0.096]	$9.09 \times 10^{-1} \pm 1.20 \times 10^{-2}$	$4.03 \times 10^{1} \pm 6.74 \times 10^{-2}$	$\mathbf{2.27 \times 10^{-1} \pm 1.47 \times 10^{-2}}$

Table 2: [MSE] Simulation results showing MSE errors between predicted steady state and ground-truth for failure states in test data (heavy load). Training data collected under light loads and restricted observed states (queues zero and one). Mixing rates are determined by the spectral gaps δ_n observed in training data over multiple time windows ($n = 1, \ldots, 50$). With 95% confidence intervals.

	δ_n (spectral gap)	DC-BPTT $t^\star = 16$	DC-BPTT $t^\star = 128$	∞-SGD ($p = 0.1$)
Testbed Emulation (Upper Trig.)	N/A	$4.80 \times 10^{-1} \pm 0.00$	$8.45 \times 10^{-1} \pm 0.00$	$\mathbf{2.41 \times 10^{-3} \pm 5.20 \times 10^{-4}}$
M/M/1/K (fast-mix)	[0.022, 0.043]	$6.81 \times 10^{-3} \pm 1.99 \times 10^{-5}$	$2.45 \times 10^{-1} \pm 1.84 \times 10^{-3}$	$\mathbf{4.98 \times 10^{-5} \pm 5.37 \times 10^{-5}}$
M/M/1/K (slow-mix)	[0.005, 0.008]	$1.14 \times 10^{-2} \pm 1.54 \times 10^{-4}$	$\mathbf{4.84 \times 10^{-4} \pm 1.99 \times 10^{-4}}$	$\mathbf{9.36 \times 10^{-4} \pm 1.58 \times 10^{-3}}$
M/M/m/$m + r$	[0.013, 0.024]	$4.25 \times 10^{-2} \pm 7.25 \times 10^{-3}$	$2.87 \times 10^{-2} \pm 1.09 \times 10^{-2}$	$\mathbf{6.65 \times 10^{-3} \pm 5.61 \times 10^{-3}}$
M/M/Multiple/K	[0.068, 0.096]	$8.97 \times 10^{-3} \pm 1.24 \times 10^{-4}$	$6.84 \times 10^{-1} \pm 1.99 \times 10^{-3}$	$\mathbf{5.34 \times 10^{-4} \pm 8.23 \times 10^{-5}}$

Pytorch's autodiff function (Paszke et al. 2017). The Supplementary Material C1 contains the details of our experimental setup and methodology. We study two parametric $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})$ models:

(A) M/M/1/K model: We start with arguably the most fundamental CTMC parametric model of a queueing system, the M/M/1/K queue with a single server and a single queue of size $K = 21$, which can hold up to 20 requests waiting for service. The queue can be described by a CTMC $\boldsymbol{Q}(x_m, \theta^\star)$, where service requests arrive according to a Poisson process with request rate x_m assumed constant over time window m —the length of a time window is the minimum time resolution of our logs (one second). If a new request arrives when the queue is full, it is dropped by the server. Service times are exponentially distributed with rate $\theta^\star$, assumed constant —the service rate capacity of the system. We assume K is known. The request call rate x_m at time window n is known while θ is the only parameter that needs to be learned. The system is assumed in steady state even over short time windows.

(B) Upper Triangular model: Since it is difficult to fit a real system with an exact CTMC queueing model, we consider an embedded birth-death process $\boldsymbol{Q}(x_m; \boldsymbol{\theta}^\star)$, called *Upper Triangular model*, which sets the upper triangular portion of $\boldsymbol{Q}$ to x_m in the upper diagonal and zeros everywhere else. This model (setting the upper triangular to zeros) indicates that only a client request can increase the queue size. The lower triangular part of $\boldsymbol{Q}$ is populated with different parameters that we will learn, which implies learning $|\boldsymbol{\theta}^\star| = |\mathbb{S}|(|\mathbb{S}| - 1)/2$ parameters for a CTMC with $|\mathbb{S}|$ states. The learnable parameters $\boldsymbol{\theta}$ are initialized with zeros and learned with either DC-BPTT or ∞-SGD. We also know the maximum queue size $K = 21$ as in the M/M/1/K scenario. Since we want to predict overload, we add a new state to represent that case. Any arrival after reaching the last state, when the current queue size is K, will force the system to transit to this overload state. Thus, we have $|\mathbb{S}| = K + 2$.

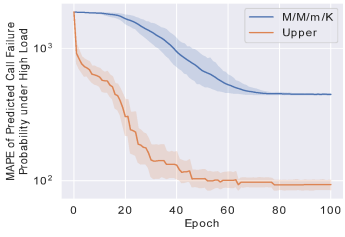
Results. Figure 1a shows the mean absolute percentage error (MAPE) between the predicted call drop probability (given the call request rate) and the true call drop probability in the test data under heavy loads, for the upper triangular and M/M/1/K parametric model learned with ∞-SGD. The *Upper Triangular* model achieves much lower test mean squared error (MSE) ($2.57 \times 10^{-3} \pm 5.75 \times 10^{-4}$) corresponding to MAPE of about 80% to 100% over test call drop probabilities in the range [0.0045, 0.1629], *i.e.*, it more accurately extrapolates the training data (light loads, just observing queues of size one and two) to the test (heavy loads, full queue). The M/M/1/K parametric model is too simple and performs poorly with test MAPE of 449% (test MSE is reasonable at 7.78×10^{-2}).

We also investigate the transition rate matrix learned by upper triangular model in Figure 1c. We note that the learned queue is quite similar to an M/M/1/K, but the service rate is decreasing as the queue size is increasing. Surprisingly, this is a real phenomenon when real systems start to become overloaded (Jain and Ramakrishnan 1988). We also see some *reset* transitions, where the system goes from a full queue to a nearly empty queue. Finally, Figure 1b shows that only ∞-SGD can learn the $\boldsymbol{Q}$ of the upper triangular model —which has $|\mathbb{S}|(|\mathbb{S}| - 1)/2 = 210$ parameters—, while DC-BPTT has vanishing gradients for both $t^\star \in \{16, 128\}$ —we note that $t^\star = 16$ has a smaller loss than $t^\star = 128$. Finally, Tables 1 and 2 reaches the obvious conclusion that ∞-SGD learns significantly better models for extrapolation over the test data than DC-BPTT.

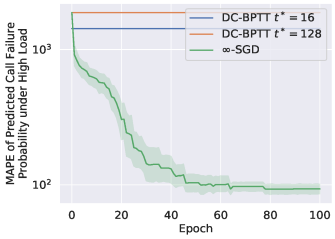
In what follows we explore the differences between DC-BPTT and ∞-SGD in synthetic experiments.

Synthetic Experiments

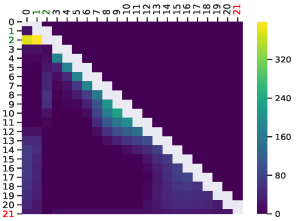
We now turn our attention to simulations. Due to space limitations, we give a succinct description of the experiments, relegating details and additional results to Supplementary Materials C2, C3, C4, and C5.



(a) (Training curves) Effect of parametric models on test MAPE (learned with ∞-SGD).



(b) (Training curves) Effect of learning methods on test MAPE of upper triangular model.



(c) $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}^\star)$ learned by *upper triangular* model with ∞-SGD. Upper triangular ($\boldsymbol{x}$) is removed or shows zeros, lower triangular shows $\boldsymbol{\theta}^\star$.

Figure 1: Real-world experiment results on VoLTE testbed. (a-b) Test error (MAPE) of unseeing failure state —here, *ground-truth* of dropped call probability— under heavy load (while training under light loads), as a function of training epochs. In these plots we verify the better generalization and stability of ∞-SGD. (a) Shows that more flexible *Upper Triangular* parametric model has much smaller (near-zero) test error than the more strict M/M/1/K parametric model. (b) Shows that ∞-SGD significantly outperforms DC-BPTT (which fails to learn). (c) Learned $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}^\star)$ by the upper triangular parametric model with ∞-SGD, showing an emergent block structure.

Birth-death queues: We start with arguably the most fundamental parametric CTMC queueing system, the diagonal structure of the *birth-death process*. The birth-death process approximates a number of queueing systems, such as the M/M/1/K queue with a single queue of size K and a single server, and the M/M/m/K queue with m servers that has been used to approximate cloud services (Yang et al. 2009; Khazaei, Misic, and Misic 2012). Since queue size K must be larger than m to support all servers, the M/M/m/K queue is typically denoted as M/M/m/$m + r$, where $r \geq 0$.

Training and testing data. Structures used to simulate data are provided in Supplementary Material A3. We predefine the service rates ($\theta^\star = 25$ for M/M/1/K (slow-mix and fast-mix), $\theta^\star = 5$ for M/M/5/5+r, $\boldsymbol{\theta}^\star = (15, 10, 5)$ for M/M/Multiple/K) and queue sizes $K = 20$ ($r = 15$), then at each time window in the training data we sample a request rate uniformly in the interval $\boldsymbol{x} \in [11, 15]$ (light load), except for M/M/1/K slow-mix $\boldsymbol{x} \in [21, 30]$ (to decrease the spectral gap). At test time, in the test data, we sample a request rate uniformly in the interval $\boldsymbol{x} \in [31, 60]$ (heavy load), except for M/M/1/K slow-mix $\boldsymbol{x} \in [11, 40]$ (to decrease the spectral gap). We assume we *only* observe the queue size if it is empty or it has exactly one request, *i.e.*, $\mathbb{S}' = \{0, 1\}$. This emulates a common trend in logging critical infrastructure systems, where logging stops as soon as the server load is non-trivial (Newman 2017). We also have extra results with different transition rates (in an easier task where ∞-SGD does even better) and more details on our training and test data generation in Supplementary Materials C2 and C5.

Results. Our goal is to predict the request loss probability against ground-truth under a range of both heavy and lighter loads, while training under a narrow range of light loads. In the M/M/1/K and M/M/m/m+r simulations, the training data consists of the aggregate frequencies observed for queue sizes zero and one during one second, along with the request rate. For M/M/Mutiple/K, aggregate frequencies for queue sizes zero to three are observed.

Tables 1 and 2 compare the extrapolation error of DC-BPTT and ∞-SGD in our synthetic experiment using MAPE and MSE errors, respectively. Our approach, ∞-SGD, is consistently better than DC-BPTT over all simulations and on both error metrics (MAPE and MSE). For a slow-mixing M/M/1/K, ∞-SGD extrapolation MAPE error is $1/100$-th of DC-BPTT MAPE error, considering the confidence interval. In some of the scenarios, DC-BPTT finds gradient vanishing problems (failing to learn) giving very large errors (see training curves in Supplementary Material C5), while ∞-SGD never fails to obtain gradients that can reduce the loss during the optimization.

MAPE result shows that $t^\star \leq 128$ is not enough to see the slow-mixing chain in steady state. Success in MSE for slow mixing while failing in MAPE shows that DC-BPTT has trouble learning parametric CTMCs well enough to predict out-of-sample (extrapolated) target states that have small probabilities. Moreover, DC-BPTT $t^\star = 16$ tends to achieve lower errors (both MAPE and MSE) than DC-BPTT $t^\star = 128$ in the M/M/1/K fast mixing scenarios.

We now look at ground-truth $\theta^\star$ parameters and their estimates $\hat{\theta}$ from ∞-SGD. We note that the estimates are very close to the true values. In the M/M/1/K model (true $\theta^\star = 25$), the (slow mixing) scenario gives $\hat{\theta} = 25.003$, and (fast mixing) gives $\hat{\theta} = 25.083$. For M/M/m/m+r (with true $\theta^\star = 5$) obtains $\hat{\theta} = 5.15$. For M/M/Multiple/K, ∞-SGD obtains $\hat{\boldsymbol{\theta}} = (13.5, 8.3, 5.4)$, close to the ground truth $\boldsymbol{\theta}^\star = (15, 10, 5)$. This conclusively shows ∞-SGD to be a reliable optimization method.

Conclusions

This work introduces ∞-SGD, the first theoretically principled optimization approach that can accurately learn general *parametric* Continuous Time Markov Chains (CTMCs) from aggregate steady-state observations. Our approach, ∞-SGD, works even when the observations are over a restricted set of states. We have shown that ∞-SGD finds significantly better maximum likelihood estimates than the baseline (DC-BPTT) in both a real testbed and synthetic scenarios. Moreover, in the context of queueing systems, ∞-SGD consistently better extrapolates from training data in light loads to heavy loads in test data. We expect ∞-SGD to be a useful tool in other tasks where parametric models are needed and sequence data is only available as aggregate frequencies.

Acknowledgement

This work has been sponsored in part by the ARO, under the U.S. Army Research Laboratory contract number W911NF-09-2-0053, the Purdue Integrative Data Science Initiative, and the National Science Foundation grants CNS-1717493, OAC-1738981, and CCF-1918483.

References

Amazon. 2019. Amazon EC2 Auto Scaling. https://aws.amazon.com/ec2/autoscaling/.

Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A. D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; and Zaharia, M. 2010. A view of cloud computing. *Communications of the ACM* 53(4):50–58.

Bernstein, G., and Sheldon, D. 2016. Consistently estimating Markov chains with noisy aggregate data. In *Artificial Intelligence and Statistics*, 1142–1150.

Bhatia, R. 1994. First and second order perturbation bounds for the operator absolute value. *Linear Algebra Appl.*

Bhatnagar, S., and Borkar, V. S. 1998. A two timescale stochastic approximation scheme for simulation-based parametric optimization. *Probability in the Engineering and Informational Sciences* 12(4):519–531.

Bottou, L. 1998. Online learning and stochastic approximations. *On-line learning in neural networks* 17(9):142.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer. 177–186.

Cao, L.; Fahmy, S.; Sharma, P.; and Zhe, S. 2018. Data-driven resource flexing for network functions virtualization. In *Proc. ANCS*.

Colson, B.; Marcotte, P.; and Savard, G. 2007. An overview of bilevel optimization. *Annals of operations research* 153(1):235–256.

d. Silva, D. V. C.; d. M. B. Domingues, G.; Velloso, P. B.; and d. A. Rocha, A. A. 2018. Analysis of mobile-live-users of a large cdn. In *IEEE ISCC*.

ETSI. 2014. ETSI Network Functions Virtualisation (NFV) Architectural Framework. http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf.

Filippone, M., and Engler, R. 2015. Enabling scalable stochastic gradient-based inference for gaussian processes by employing the unbiased linear system solver (ulisse). *arXiv preprint arXiv:1501.05427*.

Google Cloud. 2019. Autoscaling Groups of Instances. https://cloud.google.com/compute/docs/autoscaler/.

Hashimoto, T.; Gifford, D.; and Jaakkola, T. 2016. Learning population-level diffusions with generative rnns. In *International Conference on Machine Learning*, 2417–2426.

Jain, R., and Ramakrishnan, K. K. 1988. Congestion avoidance in computer networks with a connectionless network layer, part i: Concepts, goals and methodology. In *Proc. Computer Networking Symposium, Washington, D.C.*, 134–143.

Jensen, A. 1953. Markoff chains as an aid in the study of markoff processes. *Skand. Aktuarietidskrift* 36:87–91.

Kalbfleisch, J. D., and Lawless, J. F. 1984. Least-squares estimation of transition probabilities from aggregate data. *Canadian Journal of Statistics* 12(3):169–182.

Kamailio. 2019. *Kamailio SIP Server*. https://www.kamailio.org/w/.

Khazaei, H.; Misic, J.; and Misic, V. B. 2012. Performance analysis of cloud computing centers using m/g/m/m+r queuing systems. *IEEE Transactions on Parallel and Distributed Systems* 23(5).

Kumar, R.; Tomkins, A.; Vassilvitskii, S.; and Vee, E. 2015. Inverting a Steady-State. In *WSDM*.

Linux Manual Pages. 2019. *ping, ICMP ECHO REQUEST*. https://linux.die.net/man/8/ping.

Marcus, G. F. 1998. Rethinking eliminative connectionism. *Cognitive psychology* 37(3):243–282.

Maystre, L., and Grossglauser, M. 2015. Fast and Accurate Inference of Plackett–Luce Models. In *NIPS*.

McLeish, D. 2011. A general method for debiasing a monte carlo estimator. *Monte Carlo Methods Appl.*

Meyer, C. D. 1989. Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. *SIAM Rev.* 31(2):240–272.

Miller, G. A. 1952. Finite markov processes in psychology. *Psychometrika* 17(2):149–167.

Newman, A. 2017. *Benchmarking Java logging frameworks*. https://www.loggly.com/blog/benchmarking-java-logging-frameworks/.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. https://pytorch.org.

Pierson, E.; Koh, P. W.; Hashimoto, T.; Koller, D.; Leskovec, J.; Eriksson, N.; and Liang, P. 2018. Inferring multidimensional rates of aging from cross-sectional data. *arXiv preprint arXiv:1807.04709*.

Ragain, S., and Ugander, J. 2016. Pairwise Choice Markov Chains. In *NIPS*.

Rhee, C.-h., and Glynn, P. W. 2015. Unbiased estimation with square root convergence for sde models. *Operations Research* 63(5):1026–1043.

Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; Johnston, A.; Peterson, J.; Sparks, R.; Handley, M.; and Schooler, E. 2002. SIP: Session initiation protocol. RFC 3261, RFC Editor. http://www.rfc-editor.org/rfc/rfc3261.txt.

SIPp. 2014. *Welcome to SIPp*. http://sipp.sourceforge.net/.

Szabó, Z.; Sriperumbudur, B. K.; Póczos, B.; and Gretton, A. 2016. Learning theory for distribution regression. *The Journal of Machine Learning Research* 17(1):5272–5311.

Van Der Plas, A. P., et al. 1983. On the estimation of the parameters of markov probability models using macro data. *The Annals of Statistics* 11(1):78–85.

Weiss, G. M., and Hirsh, H. 1998. Learning to predict rare events in event sequences. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*.

Werbos, P. J. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78(10):1550–1560.

Wolff, R. W. 1982. Poisson arrivals see time averages. *Operations Research* 30(2):223–231.

Xu, K.; Srivastava, A.; and Sutton, C. 2019. Variational russian roulette for deep bayesian nonparametrics. In *International Conference on Machine Learning*, 6963–6972.

Yang, B.; Tan, F.; Dai, Y.; and Guo, S. 2009. Performance evaluation of cloud service considering fault recovery. In *Proc. of 1st International Conference on Cloud Computing*.

Supplementary Material

Supplementary Material A1: Unidentifiability

Let us focus on the following two transition matrices of different Markov chains:

$$\boldsymbol{P} = \begin{bmatrix} 0.7 & 0.3 & 0 & 0 \\ 0.4 & 0.4 & 0.2 & 0 \\ 0 & 0.3 & 0.6 & 0.1 \\ 0 & 0 & 0.2 & 0.8 \end{bmatrix} \quad \boldsymbol{P}' = \begin{bmatrix} 0.7 & 0.3 & 0 & 0 \\ 0.4 & 0.5 & 0.1 & 0 \\ 0 & 0.3 & 0.5 & 0.2 \\ 0 & 0 & 0.1 & 0.9 \end{bmatrix}$$

Their steady state distribution can be achieved directly:

$$\boldsymbol{P}^{\infty} = [0.4 \quad 0.3 \quad 0.2 \quad 0.1]$$
$$\boldsymbol{P'}^{\infty} = [0.4 \quad 0.3 \quad 0.1 \quad 0.2]$$

Suppose in our learning task, a perfect aggregate frequencies collection is done for state 1 and state 2. In other words, we know the ground truth steady state distribution of state 1 and state 2, which are 0.4 and 0.3. If we regard state 4 as the failure state we want to extrapolate, we can easily notice that it is unidentifiable: A_1 and A_2 can both match the aggregate frequencies, while they have totally different distribution on state 4.

Supplementary Material A2: A Slow Mixing Event

We give an example on slow mixing M/M/1/K. Suppose we have following 20×20 matrix

$$\boldsymbol{Q} = \begin{bmatrix} -25 & 25 & & & & & \\ 24 & -49 & 25 & & & & \\ & 24 & -49 & 25 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 24 & -49 & 25 & \\ & & & & 24 & -24 \end{bmatrix}.$$

We can then get a stochastic matrix $\boldsymbol{P}$ by uniformizing $\boldsymbol{Q}$ as Definition 1. We say a matrix is fast on mixing if all rows of $\boldsymbol{P}^t$ are similar with small t, and vise versa.

We define

$$\text{mix}\left(\boldsymbol{P}, \epsilon\right) = \min \left\{ t \in \mathbb{N}+; \max_{i=1,\cdots,|\mathbb{S}|} \left(\sum_{j=1}^{|\mathbb{S}|} \left\| \boldsymbol{P}_{ij}^t - \frac{1}{|\mathbb{S}|} \sum_{k=1}^{|\mathbb{S}|} \boldsymbol{P}_{kj}^t \right\|_2^2 \right) \leq \epsilon \right\} \tag{12}$$

as the criterion for mixing speed, where $|\mathbb{S}|$ is the number of states in the CTMC. This represents the minimum exponent t for $\boldsymbol{P}^t$ to reaching a mixing status defined by ϵ. The larger value of t, the slower speed for $\boldsymbol{P}$ to mix. For instance, we will have mix $\left(\boldsymbol{P}, 10^{-5}\right) = 280$. This means that for the defined $\boldsymbol{P}$, we need $\boldsymbol{P}^{280}$ to reach mixing with our ϵ tolerance (which we found unyielding for the autodiff library of Pytorch (Paszke et al. 2017)). In contrast, there is no backpropagation in ∞-SGD, allowing it to easily deal with slow-mixing CTMCs.

If we replace in the lower diagonal line of $\boldsymbol{Q}$ the value 24 by 1, and fix related diagonal line from -49 to -26, we can get a new transition matrix $\boldsymbol{P}'$. Then, mix $\left(\boldsymbol{P}', 10^{-5}\right) = 27$ as a fast mixing example, and now backpropagation over P^{27} is supported by autodiff library.

Supplementary Material A3: Parametric Models

M/M/m/K Model. We illustrate M/M/m/K model (apply for M/M/1/K and M/M/m/m+r) in Figure 2 for better description of details. In Figure 2 green states are observed states $\mathbb{S}'$, and red state is the state we will test on.

In all simulation experiments, we set $K = 20$ for Figure 2. For M/M/1/K simulation, $m = 1$; and for M/M/m/m+r simulation, $m = 5$, $r = 15$ and $K = m + r = 20$. Thus, suppose we denote request rate as $\boldsymbol{x} = (x)$ and service rate as $\boldsymbol{\theta} = (\theta)$, we will have transition rate matrix

$$\boldsymbol{Q}(\boldsymbol{x}; \boldsymbol{\theta}) = \begin{bmatrix} -x & x & & & & & & & \\ \theta & -(\theta + x) & x & & & & & & \\ & 2\theta & -(2\theta + x) & x & & & & & \\ & & \ddots & \ddots & \ddots & & & & \\ & & & m\theta & -(m\theta + x) & x & & & \\ & & & & m\theta & -(m\theta + x) & x & & \\ & & & & & \ddots & \ddots & \ddots \end{bmatrix} \tag{13}$$

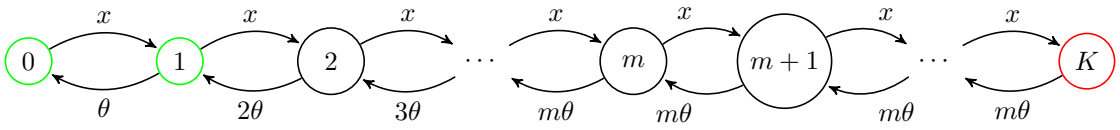


Figure 2: M/M/m/K system. Transition states are defined by i where i is the number of requests in the queue. States in green show observed states in the training data (0,1), and state in red shows state we need to extrapolate in the test data (K) [better visualized in color].

Upper Triangular Model. Upper triangular model is nearly the same as M/M/m/K, except that all lower triangular part are learnable parameters $\boldsymbol{\theta} = \{\theta_{ij}\}_{i,j\in\mathbb{S},i>j}$. We use matrix version $\boldsymbol{\theta}$ for the ease notation, and it is easy to flatten it back into vector version $\boldsymbol{\theta}$.

$$\boldsymbol{Q}(\boldsymbol{x};\boldsymbol{\theta}) = \begin{bmatrix} -x & x & & & & \\ \theta_{21} & -(\theta_{21}+x) & x & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \theta_{i1} & \cdots & \cdots & -(\sum_{j=1}^{i-1}\theta_{ij}+x) & x & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{14}$$

M/M/Multiple/K Model. M/M/Multiple/K model stands between M/M/m/K and upper triangular structures. It augments M/M/1/K structure by making lower 1 to d diagonal lines learanble where we pick $d = 3$ in our experiments. Thus, we will have $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$.

$$\boldsymbol{Q}(\boldsymbol{x};\boldsymbol{\theta}) = \begin{bmatrix} -x & x & & & & & \\ \theta_1 & -(\theta_1+x) & x & & & & \\ \theta_2 & \theta_1 & -(\sum_{j=1}^{2}\theta_i+x) & x & & & \\ \theta_3 & \theta_2 & \theta_1 & -(\sum_{j=1}^{3}\theta_i+x) & x & & \\ 0 & \theta_3 & \theta_2 & \theta_1 & -(\sum_{j=1}^{3}\theta_i+x) & x & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{15}$$

Supplementary Material B1: Proof of Lemma 1

We restate the lemma for completeness.

Lemma. *Let $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})$ be a K-state transition matrix and $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))$ be its uniformized Markov Chain. $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^t$ is the Markov chain after t steps where $t > 0$, then the gradients of $\boldsymbol{P}^t$ w.r.t. $\boldsymbol{\theta}_k$ is*

$$\begin{aligned} \nabla^{(t)}_{\boldsymbol{\theta}_k}\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta})) &\equiv \frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^t}{\partial \boldsymbol{\theta}_k} \\ &= \sum_{l=1}^{t} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{t-l} \frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{l-1}, \end{aligned}$$

where $\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k} = \sum_{ij} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} \frac{\partial q_{ij}(\boldsymbol{x},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_k}$.

Proof. In defining matrix derivatives, Example 2.1 in Bhatia (1994) uses the binomial expansion to define this derivative, which can be alternatively obtained by applying the chain rule recursively:

$$
\begin{aligned}
\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^t}{\partial \boldsymbol{\theta}_k} &= \frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{t-1}}{\partial \boldsymbol{\theta}_k}\boldsymbol{P} + \boldsymbol{P}^{t-1}\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k} \\
&= \left(\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{t-2}}{\partial \boldsymbol{\theta}_k}\boldsymbol{P} + \boldsymbol{P}^{t-2}\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k}\right)\boldsymbol{P} + \boldsymbol{P}^{t-1}\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k} \\
&= \left(\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))^{t-2}}{\partial \boldsymbol{\theta}_k}\right)\boldsymbol{P}^2 + \boldsymbol{P}^{t-2}\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k}\boldsymbol{P} + \boldsymbol{P}^{t-1}\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k} \\
&= \cdots \\
&= \sum_{l=1}^{t} \boldsymbol{P}^{t-l}\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_k}\boldsymbol{P}^{l-1}.
\end{aligned}
$$

□

Supplementary Material B2: Proof of Proposition 1

We restate the proposition for the sake of completeness.

Proposition. *Let $\boldsymbol{Q}$ be a K-state transition rate matrix of a stationary and ergodic MC. Equation* (9) *for $t \to \infty$, henceforth denoted $\nabla_{\boldsymbol{Q}}^{(\infty)}\boldsymbol{P}(\boldsymbol{Q}) \equiv \lim_{t\to\infty}\nabla_{\boldsymbol{Q}}^{(t)}\boldsymbol{P}(\boldsymbol{Q})$, **exists and is unique** and can be redefined as*

$$
\begin{aligned}
(\nabla_{\boldsymbol{Q}}^{(\infty)}\boldsymbol{P}(\boldsymbol{Q}))_{ij} &\equiv \lim_{t\to\infty}\sum_{l=1}^{t}\left(\boldsymbol{P}^{t-l}\frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}\boldsymbol{P}^{l-1}\right) \\
&= \boldsymbol{\Pi}\sum_{l=0}^{\infty}\frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}\boldsymbol{P}^{l},
\end{aligned}
$$

where $\boldsymbol{\Pi}$ is a matrix whose rows are the steady state distribution $\boldsymbol{\pi}$. Note that the diagonal $i = j$ is trivial to compute but should be treated as a special case.

Proof. Let

$$
\boldsymbol{Q} = \begin{bmatrix} -q_{11} & q_{12} & \cdots & q_{1|\mathbb{S}|} \\ \vdots & \vdots & \cdots & \vdots \\ q_{|\mathbb{S}|1} & q_{|\mathbb{S}|2} & \cdots & -q_{|\mathbb{S}||\mathbb{S}|} \end{bmatrix}
$$

be the rate transition matrix of the CTMC. Define $\boldsymbol{P}(\boldsymbol{Q})$ as in Definition 1. Then,

$$
\left(\frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}\right)_{kh} = \frac{1}{\gamma}\frac{\partial q_{kh}}{\partial q_{ij}} - \frac{1}{\gamma^2}q_{kh}\frac{\partial \gamma}{\partial q_{ij}}, \quad i,j,k,h \in \mathbb{S},
$$

where $\gamma = \max_{k\in\mathbb{S}}(q_{kk}) + \epsilon$, $\epsilon > 0$, and $q_{kk} = \sum_{i\in\mathbb{S}} q_{ki}$, $k \in \mathbb{S}$. Note that if q_{ij} is not one of the rates in γ, then

$$
\frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \cdots & -1/\gamma & \cdots & 1/\gamma & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \end{bmatrix},
$$

where $\frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}$ is a matrix of zeroes except at elements $\left(\frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}\right)_{ii} = -1/\gamma$ and $\left(\frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}\right)_{ij} = 1/\gamma$.

If q_{ij} is one of the rates in γ, then

$$
\frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} = \frac{1}{\gamma^2}\begin{bmatrix} q_{11} & -q_{12} & \cdots & -q_{1i} & \cdots & -q_{1j} & \cdots & -q_{1|\mathbb{S}|} \\ \vdots & \vdots & \cdots & \vdots & & \vdots & \cdots & \vdots \\ -q_{i1} & -q_{i2} & \cdots & \gamma + q_{ii} & \cdots & -\gamma - q_{ij} & \cdots & -q_{i|\mathbb{S}|} \\ \vdots & \vdots & \cdots & \vdots & & \vdots & \cdots & \vdots \\ -q_{|\mathbb{S}|1} & -q_{|\mathbb{S}|2} & \cdots & -q_{|\mathbb{S}|i} & \cdots & -q_{|\mathbb{S}|j} & \cdots & q_{|\mathbb{S}||\mathbb{S}|} \end{bmatrix},
$$

noting that $\frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}\mathbf{1} = \mathbf{0}$ regardless of whether q_{ij} is in γ or not.

To simplify the notation, denote $\boldsymbol{P} \equiv \boldsymbol{P}(\boldsymbol{Q})$. Now note that

$$\begin{aligned}
\lim_{t\to\infty}\sum_{l=1}^{t}\left(\boldsymbol{P}^{t-l}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l-1}\right) &= \lim_{t'\to\infty}\sum_{l=1}^{2t'}\left(\boldsymbol{P}^{2t'-l}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l-1}\right) \\
&= \lim_{t'\to\infty}\sum_{l=1}^{t'}\left(\boldsymbol{P}^{2t'-l}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l-1} + \boldsymbol{P}^{2t'-(2t'+1-l)}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{(2t'+1-l)-1}\right) \\
&= \lim_{t'\to\infty}\sum_{l=1}^{t'}\left(\boldsymbol{P}^{2t'-l}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l-1} + \boldsymbol{P}^{l-1}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{2t'-l}\right) \\
&= \lim_{t'\to\infty}\sum_{l_1=1}^{t'}\left(\boldsymbol{P}^{2t'-l_1}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l_1-1}\right) + \lim_{t'\to\infty}\sum_{l_2=1}^{t'}\left(\boldsymbol{P}^{l_2-1}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{2t'-l_2}\right) \\
&= \sum_{l_1=1}^{\infty}\left(\boldsymbol{\Pi}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l_1-1}\right) + \sum_{l_2=1}^{\infty}\left(\boldsymbol{P}^{l_2-1}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{\Pi}\right),
\end{aligned}$$

where $\boldsymbol{\Pi} \equiv \lim_{t'\to\infty}\boldsymbol{P}^{t'}$. The r.h.s. of Equation (10) is zero because $\frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}\mathbf{1} = \mathbf{0}$, where $\mathbf{1}$ is a column vector of ones, and

$$\boldsymbol{\Pi} \equiv \lim_{t'\to\infty}\boldsymbol{P}^{t'} = \begin{bmatrix} \boldsymbol{\pi}^{\mathsf{T}} \\ \vdots \\ \boldsymbol{\pi}^{\mathsf{T}} \end{bmatrix}. \tag{16}$$

Noting that

$$\lim_{l\to\infty}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l-1} = \frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{\Pi} = \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}\mathbf{1}\boldsymbol{\pi}^{\mathsf{T}} = 0. \tag{17}$$

Thus,

$$\begin{aligned}
\lim_{t\to\infty}\sum_{l=1}^{t}\left(\boldsymbol{P}^{t-l}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l-1}\right) &= \sum_{l_1=1}^{\infty}\left(\boldsymbol{\Pi}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l_1-1}\right) + \sum_{l_2=1}^{\infty}\left(\boldsymbol{P}^{l_2-1}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{\Pi}\right) \\
&= \sum_{l_1=1}^{\infty}\left(\boldsymbol{\Pi}\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l_1-1}\right) \\
&= \boldsymbol{\Pi}\sum_{l_1=1}^{\infty}\left(\frac{\partial \boldsymbol{P}}{\partial q_{ij}}\boldsymbol{P}^{l_1-1}\right)
\end{aligned}$$

will converge by two reasons: summation controlled by l_2 equals 0 according to Equation (17); and term inside summation controlled by l_1 converges to 0 as $l_1 \to 0$ according to Equation (17).

□

Supplementary Material B3: Proof of Theorem 1

We restate the theorem for the sake of completeness.

Theorem (Infinity Stochastic Gradient Descent (∞-SGD))**.** *Let $\boldsymbol{Q}$ be the transition rate matrix of a stationary and ergodic CTMC. Assume strictly positive values for the learnable parameters $\boldsymbol{\theta}^{(h)}$ at the h-th step of the optimization. Let $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}^{(h)}))$ be its uniformized transition probability matrix per Definition 1. Let $\mathcal{L}(\boldsymbol{y}, \boldsymbol{\pi})$ be as in Equation* (3)*. Reparameterize $\tilde{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}^{(h)}) = \mathcal{L}(\boldsymbol{y}, \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}^{(h)}))$ as the loss function with respect to $\boldsymbol{\theta}^{(h)}$. Let $X^{(h)} \sim$ Geometric$(p^{(h)})$, $X^{(h)} \in \mathbb{Z}^{+}$, be an independent sample of a Geometric distribution with $p^{(h)} < \delta^{(h)}$, where $\delta^{(h)}$ is the spectral gap of $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}^{(h)}))$. Then, for $0 < \epsilon \ll 1$ and for all learnable parameters $\boldsymbol{\theta}$,*

$$\boldsymbol{\theta}_k^{(h+1)} = \max\left(\boldsymbol{\theta}_k^{(h)} - \eta^{(h)}\nabla_{\boldsymbol{\theta}_k}\,\tilde{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(h)}}, \epsilon\right),$$

where

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_k}\tilde{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = \sum_{ij}\sum_{mn}(\mathbf{p}^{(events)}(0))_m\;&\frac{\partial \mathcal{L}(\boldsymbol{y}, \boldsymbol{\pi})}{\partial \boldsymbol{\pi}_n}\bigg|_{\boldsymbol{\pi}=\boldsymbol{\pi}(\boldsymbol{x},\boldsymbol{\theta})} \\
&\times \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta})_n\Gamma_{ijmn}(\boldsymbol{x}, \boldsymbol{\theta})\frac{\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})_{ij}}{\partial \boldsymbol{\theta}_k}
\end{aligned}$$

with $h = 0, 1, \ldots$, *where* $\boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta})$ *is the steady state distribution defined in Equation* (6)*,* $\eta^{(h)}$ *is the learning rate with* $\sum_{h=0}^{\infty} \eta^{(h)} = \infty$*,* $\sum_{h=0}^{\infty} \left(\eta^{(h)}\right)^2 < \infty$*, and*

$$\Gamma_{ijmn}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{t=0}^{X^{(h)}} \left[\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial q_{ij}} \frac{\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))^t}{\mathbb{P}[X^{(h)} > t]} \right]_{mn},$$

is a stochastic gradient descent method that minimizes Equation (2)*.*

Proof. We first derive the partial gradients $\nabla_{\boldsymbol{\theta}_k} \tilde{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta})$ from bottom. Similar to Equation (16), we define $\boldsymbol{\Pi} = \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))^{\infty}$ which is also a stack of steady state distribution $\pi(\boldsymbol{x}, \boldsymbol{\theta})^{\mathsf{T}}$.

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_k} \tilde{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}_k} \mathcal{L}(\boldsymbol{y}, \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}^{(h)})) \\
&= \nabla_{\boldsymbol{\theta}_k} \mathcal{L}(\boldsymbol{y}, \mathbf{p}^{(\text{events})}(0)^{\mathsf{T}} \boldsymbol{\Pi}(\boldsymbol{x}, \boldsymbol{\theta})) \\
&= \sum_{i,j \in \mathbb{S}} \frac{\partial \mathcal{L}(\boldsymbol{y}, \mathbf{p}^{(\text{events})}(0)^{\mathsf{T}} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))^{\infty})}{\partial q_{ij}} \frac{\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})_{ij}}{\partial \boldsymbol{\theta}_k} \\
&= \sum_{i,j \in \mathbb{S}} \left[\sum_{m,n \in \mathbb{S}} \frac{\partial \mathcal{L}(\boldsymbol{y}, \mathbf{p}^{(\text{events})}(0)^{\mathsf{T}} \boldsymbol{\Pi}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial \boldsymbol{\Pi}_{mn}} \frac{\partial \boldsymbol{\Pi}(\boldsymbol{x}, \boldsymbol{\theta})_{mn}}{\partial q_{ij}} \right] \frac{\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})_{ij}}{\partial \boldsymbol{\theta}_k} \\
&= \sum_{i,j \in \mathbb{S}} \sum_{m,n \in \mathbb{S}} (\mathbf{p}^{(\text{events})}(0)^{\mathsf{T}})_m \frac{\partial \mathcal{L}(\boldsymbol{y}, \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial \boldsymbol{\pi}_n} \frac{\partial \boldsymbol{\Pi}(\boldsymbol{x}, \boldsymbol{\theta})_{mn}}{\partial q_{ij}} \frac{\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})_{ij}}{\partial \boldsymbol{\theta}_k} \\
&= \sum_{i,j \in \mathbb{S}} \sum_{m,n \in \mathbb{S}} (\mathbf{p}^{(\text{events})}(0)^{\mathsf{T}})_m \frac{\partial \mathcal{L}(\boldsymbol{y}, \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial \boldsymbol{\pi}_n} \left[\frac{\partial \boldsymbol{\Pi}(\boldsymbol{x}, \boldsymbol{\theta})}{\partial q_{ij}} \right]_{mn} \frac{\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})_{ij}}{\partial \boldsymbol{\theta}_k} \\
&= \sum_{i,j \in \mathbb{S}} \sum_{m,n \in \mathbb{S}} (\mathbf{p}^{(\text{events})}(0)^{\mathsf{T}})_m \frac{\partial \mathcal{L}(\boldsymbol{y}, \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial \boldsymbol{\pi}_n} \left[\left(\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial \boldsymbol{Q}} \right)_{ij} \right]_{mn} \frac{\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})_{ij}}{\partial \boldsymbol{\theta}_k} \\
&= \sum_{i,j \in \mathbb{S}} \sum_{m,n \in \mathbb{S}} (\mathbf{p}^{(\text{events})}(0)^{\mathsf{T}})_m \frac{\partial \mathcal{L}(\boldsymbol{y}, \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial \boldsymbol{\pi}_n} \left[\boldsymbol{\Pi}(\boldsymbol{x}, \boldsymbol{\theta}) \sum_{l=0}^{\infty} \frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial q_{ij}} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))^l \right]_{mn} \frac{\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})_{ij}}{\partial \boldsymbol{\theta}_k}
\end{aligned}$$

In the proof we will use X rather than $X^{(h)}$, whose meaning will be clear from context. We then show that the above derivative proposed in the theorem is an unbiased estimation of above derivative.

First, assume we imposed the condition (we will later prove this to be true)

$$\sum_{x=1}^{\infty} \frac{\left\| \sum_{j=x+1}^{\infty} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} \boldsymbol{P}^j \right\|_2^2}{\mathbb{P}[X \geq x]} < \infty, \quad \forall i, j, \tag{18}$$

we imposed is equivalent to the condition

$$\sum_{x=1}^{\infty} \frac{\left\| \sum_{j=1}^{\infty} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} \boldsymbol{P}^j - \sum_{j=1}^{x} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} \boldsymbol{P}^j \right\|_2^2}{\mathbb{P}[X \geq x]} < \infty,$$

which we use to invoke Theorem 1 of Rhee and Glynn (2015), which shows that under these conditions the expectation exists, *i.e.*, $\boldsymbol{\Pi}\, \mathbb{E}_X[\Gamma_{ij}] = (\nabla_{\boldsymbol{Q}}^{(\infty)} \boldsymbol{P}(\boldsymbol{Q}))_{ij}$, with $\nabla_{\boldsymbol{Q}}^{(\infty)} \boldsymbol{P}(\boldsymbol{Q})$ as defined in Equation (10), and Γ_{ij} has also a finite second moment. Since the expectation exists, the second part of the proof starts with the expansion of the expectation

$$\mathbb{E}_X[\Gamma_{ij}] = \mathbb{E}_X \left[\sum_{l'=0}^{X} \frac{\frac{\partial \boldsymbol{P}}{\partial q_{ij}} \boldsymbol{P}^{l'}}{\mathbb{P}[X > l']} \right] = \sum_{x=1}^{\infty} \sum_{l=1}^{x} \frac{\frac{\partial \boldsymbol{P}}{\partial q_{ij}} \boldsymbol{P}^{l-1}}{\mathbb{P}[X \geq l]} \mathbb{P}[X = x].$$

By Fubini's theorem

$$\sum_{x=1}^{\infty} \sum_{l=1}^{x} \frac{\frac{\partial \boldsymbol{P}}{\partial q_{ij}} \boldsymbol{P}^{l-1}}{\mathbb{P}[X \geq l]} \mathbb{P}[X = x] = \sum_{l=1}^{\infty} \sum_{x=l}^{\infty} \frac{\frac{\partial \boldsymbol{P}}{\partial q_{ij}} \boldsymbol{P}^{l-1}}{\mathbb{P}[X \geq l]} \mathbb{P}[X = x],$$

while moving the sum that depends on x inside, noting that $\sum_{x=l}^{\infty} \mathbb{P}[X = x] = \mathbb{P}[X \geq x]$, and canceling the terms yields

$$\begin{aligned}
&\sum_{l=1}^{\infty} \left(\frac{\frac{\partial \boldsymbol{P}}{\partial q_{ij}} \boldsymbol{P}^{l-1}}{\mathbb{P}[X \geq l]} \sum_{x=l}^{\infty} \mathbb{P}[X = x] \right) \\
&= \sum_{l=1}^{\infty} \left(\frac{\frac{\partial \boldsymbol{P}}{\partial q_{ij}} \boldsymbol{P}^{l-1}}{\mathbb{P}[X \geq l]} \mathbb{P}[X \geq l] \right) \\
&= \sum_{l'=0}^{\infty} \frac{\partial \boldsymbol{P}}{\partial q_{ij}} \boldsymbol{P}^{l'}.
\end{aligned}$$

This part of the proof concludes by noting that

$$\boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta})_n \sum_{t=0}^{X^{(h)}} \left[\frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial q_{ij}} \frac{\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))^t}{\mathbb{P}[X^{(h)} > t]} \right]_{mn}$$

is an unbiased estimator of the gradient

$$\left[\boldsymbol{\Pi}(\boldsymbol{x}, \boldsymbol{\theta}) \sum_{l=0}^{\infty} \frac{\partial \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))}{\partial q_{ij}} \boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}))^l \right]_{mn}$$

with finite variance. As also the derivative of the log in $\partial \mathcal{L}(\boldsymbol{y}, \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}))/\partial \boldsymbol{\pi}_n$, and the derivative $\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta})_{ij}/\partial \boldsymbol{\theta}_k$ are infinitely differentiable. Thus, the gradient estimate $\nabla_{\boldsymbol{\theta}_k} \tilde{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta})$ can be used to find a fixed point in a Robbins-Monro stochastic optimization procedure (Bottou 2010; 1998) of Equation (2).

We now prove that Equation (18) is satisfied by our choice of $p^{(h)} < \delta^{(h)}$, where $\delta^{(h)}$ is the spectral gap of $\boldsymbol{P}$. In what follows we drop the dependence on (h) to simplify the notation. Note that we need to show

$$\sum_{x=1}^{\infty} \frac{\left\| \sum_{j=x+1}^{\infty} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} \boldsymbol{P}^j \right\|_2^2}{\mathbb{P}[X \geq x]} < \infty, \qquad \text{for all learnable } q_{ij}.$$

Without loss of generality, assume a fixed i and j. Because $\boldsymbol{P} \equiv \boldsymbol{P}(\boldsymbol{Q}_t)$ is stationary and ergodic for any positive values learnable parameters of $\boldsymbol{Q}_t$, the spectral decomposition of $\boldsymbol{P} = V \Lambda V^{-1}$ has eigenvalues $1 > \lambda_2 \geq \ldots \geq \lambda_{|\mathbb{S}|} > -1$. And let $\delta = 1 - \max(|\lambda_2|, \ldots, |\lambda_{|\mathbb{S}|}|)$ be the spectral gap of $\boldsymbol{P}$. We start the proof using the spectral decomposition in the series:

$$\begin{aligned}
\sum_{x=1}^{\infty} \frac{\left\| \sum_{j=x+1}^{\infty} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} \boldsymbol{P}^j \right\|_2^2}{\mathbb{P}[X \geq x]} &= \sum_{x=1}^{\infty} \frac{\left\| \sum_{j=x+1}^{\infty} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} V \Lambda^j V^{-1} \right\|_2^2}{\mathbb{P}[X \geq x]} \\
&= \sum_{x=1}^{\infty} \frac{\left\| \sum_{j=x+1}^{\infty} \lambda_i^j \langle \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}, V_{\cdot i} \rangle (V^{-1})_{\cdot i} \right\|_2^2}{\mathbb{P}[X \geq x]} \qquad (a),
\end{aligned}$$

because $V_{\cdot 1} = \boldsymbol{\pi}$ and $\langle \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}, \boldsymbol{\pi} \rangle = 0$ (see proof of Proposition 1), then $\langle \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}, V_{\cdot 1} \rangle = 0$, which yields

$$(a) = \sum_{x=1}^{\infty} \frac{\left\| \sum_{j=x+1}^{\infty} \sum_{i=2}^{|\mathbb{S}|} \lambda_i^j \langle \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}, V_{\cdot i} \rangle (V^{-1})_{\cdot i} \right\|_2^2}{\mathbb{P}[X \geq x]} \qquad (b)$$

Let $X \sim \text{Geometric}(p)$, $p \in (0, 1)$ with average $E[X] = 1/p$. Then, $\exists C(\boldsymbol{Q})$ s.t.

$$\begin{aligned}
(b) &= \sum_{x=1}^{\infty} \frac{\left\| \sum_{j=x+1}^{\infty} \sum_{i=2}^{|\mathbb{S}|} \lambda_i^j \langle \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}}, V_{\cdot i} \rangle (V^{-1})_{\cdot i} \right\|_2^2}{(1-p)^x} \\
&\leq \sum_{x=1}^{\infty} \frac{\left\| \sum_{j=x+1}^{\infty} \sum_{i=2}^{|\mathbb{S}|} |\lambda_i|^j C(\boldsymbol{Q}) \right\|_2^2}{(1-p)^x}
\end{aligned}$$

and because $(1-\delta) > |\lambda_i|$, for $i \geq 2$, we have

$$\sum_{x=1}^{\infty} \frac{\left\| \sum_{j=x+1}^{\infty} \sum_{i=2}^{|\mathbb{S}|} |\lambda_i|^j C(\boldsymbol{Q}) \right\|_2^2}{(1-p)^x}$$
$$\leq \sum_{x=1}^{\infty} \frac{\left\| \sum_{j'=1}^{\infty} (1-\delta)^{j'+x} \, |\mathbb{S}| \, C(\boldsymbol{Q}) \right\|_2^2}{(1-p)^x}$$
$$= \sum_{x=1}^{\infty} \left(\frac{(1-\delta)^2}{1-p} \right)^x \left\| \sum_{j'=1}^{\infty} (1-\delta)^{j'} \, |\mathbb{S}| \, C(\boldsymbol{Q}) \right\|_2^2$$

From the Cauchy–Schwarz inequality,

$$\sum_{x=1}^{\infty} \left(\frac{(1-\delta)^2}{1-p} \right)^x \left\| \sum_{j'=1}^{\infty} (1-\delta)^{j'} \, |\mathbb{S}| \, C(\boldsymbol{Q}) \right\|_2^2$$
$$\leq \sum_{x=1}^{\infty} \left(\frac{(1-\delta)^2}{1-p} \right)^x \sum_{j'=1}^{\infty} (1-\delta)^{2j'} \, |\mathbb{S}|^2 \, C(\boldsymbol{Q})^2,$$

and using the Cauchy product

$$\sum_{x=1}^{\infty} \left(\frac{(1-\delta)^2}{1-p} \right)^x \sum_{j'=1}^{\infty} (1-\delta)^{2j'} \, |\mathbb{S}|^2 \, C(\boldsymbol{Q})^2 = |\mathbb{S}|^2 \, C(\boldsymbol{Q})^2 \sum_{k=1}^{\infty} \beta_k$$

where

$$\beta_k = \sum_{l=0}^{k} \left(\frac{(1-\delta)^2}{1-p} \right)^l ((1-\delta)^2)^{k-l} = \sum_{l=0}^{k} \frac{(1-\delta)^{2k}}{(1-p)^l} = (1-\delta)^{2k} \frac{(1-p)^{-k} - (1-p)}{p}.$$

Finally, putting all the terms together

$$\sum_{x=1}^{\infty} \frac{\left\| \sum_{j=x+1}^{\infty} \frac{\partial \boldsymbol{P}(\boldsymbol{Q})}{\partial q_{ij}} \boldsymbol{P}^j \right\|_2^2}{\mathbb{P}[X \geq x]} \leq |\mathbb{S}|^2 \, C(\boldsymbol{Q})^2 \sum_{k=1}^{\infty} (1-\delta)^{2k} \frac{(1-p)^{-k} - (1-p)}{p},$$

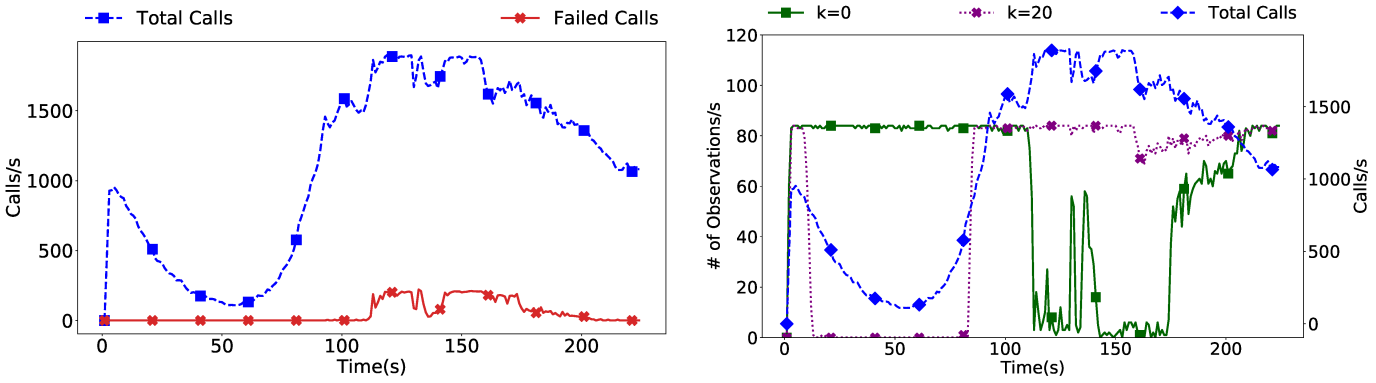
which is a convergent geometric series if $p < \delta$ as $0 \leq 1-\delta < 1$, concluding our proof. □

Supplementary Material C1: Experimental Testbed

The VoLTE testbed consists of (a) a Session Initiation Protocol (SIP) (Rosenberg et al. 2002) server implementing VoLTE functionality deployed using Kamailio (version version 5.0.4) (Kamailio 2019), and (b) a workload generator, SIPP (SIPp 2014), which generates a predefined number of SIP `REGISTER` messages per second. The VoLTE testbed is deployed on 2 HP ProLiant DL120G6 (Intel Xeon X3430 processor and 8 GB RAM) connected using one Gigabit Dell N2024 Switch. Both the SIP server and the workload generator run directly on a dedicated physical host.

System Architecture. The Kamailio server uses a pool of statically created threads to process requests from the client(s). Each thread reads data directly from the socket buffer, processes the SIP request and generates a response which is send to the client. Incoming request messages are stored in the Linux socket buffer before processing by the Kamailio application threads. Since the workload generator and SIP server communicate using User Datagram Protocol (UDP), any packets sent by the client after the socket buffer is full (Queue size $> K$) are discarded. The workload generator generates a predefined number of `REGISTER` messages each second (λ) and waits for a `200 OK` response from the SIP server. Any request that does not receive a successful `200 OK` response from the server within a predefined timeout (10 ms) is considered a failed call. In our experiments, the Kamailio server is configured to use a single application thread to process requests and the socket buffer is allocated to store up to 20 `REGISTER` requests (K=20). This setup therefore emulates an M/M/1/K queuing system with K=20.

Workload Characteristics The workload generator generates traffic according to the request-rate distribution of a video streaming server of one of the largest video streaming operators in Brazil (d. Silva et al. 2018). The request-rate distribution is shown in Figure 3a. We also measured the number of packets in the server queue during our experiments by instrumenting the workload generator code. An example of the number of packets enqueued at a server (k) is presented in Figure 3b. The figure presents the number of times an incoming request encountered a server queue size of 0 or 20 (maximum queue size in our testbed is 20). As evident from Figure 3b, at lower workloads, the number of packets waiting in the queue when a new request arrives are frequently 0 (0-100 second), but at higher workloads, the server queue is frequently full (100-175 second).



(a) Call distribution

(b) Number of times a queue size of K is encountered by an incoming packet

Figure 3: Details from VoLTE testbed experiments

Data Collection Methodology. In real production systems, it is not always possible to obtain the number of packets enqueued at the server. Applications/servers in a deployment environment typically do not generate statistics for system level socket buffers and internal queues. Therefore, for our experiments, we do not use the actual queue size generated by instrumenting the workload generator code. Instead, we estimate the number of packets in the server queue from the time taken by the server to respond to a request (server response time observed at the workload generator). Server response time is a combination of transmission delay, queueing delay, and processing time. That is, ResponseTime = PropagationTime + QueueingTime + ProcessingTime. We use the time taken by the server to respond to a ping packet (Linux Manual Pages 2019) as the PropagationTime, and the time taken by the server to process to the first request packet as the ProcessingTime. These quantities allow us to calculate the QueueingTime of a request from its ResponseTime. The server queue size observed by a request is then estimated from the QueueingTime as QueueSize = QueueingTime/ProcessingTime. Assuming Poisson arrivals, the observed queue sizes are true samples from the time average (via PASTA property (Wolff 1982)).

Supplementary Material C2: Synthetic Training and Test Data

Queue Simulation Configurations. At training time, we sample request rates of time window n from uniform intervals that keep the system load light, while at test time, we sample from request rates of heavy loads. For n-th training or test samples, we set $x_n^{\text{train}} \sim \text{Uniform}(\lambda_{\text{min}}^{\text{train}}, \lambda_{\text{max}}^{\text{train}})$, and test over $x_n^{\text{test}} \sim \text{Uniform}(\lambda_{\text{min}}^{\text{test}}, \lambda_{\text{max}}^{\text{test}})$.

In open queues (M/M/1/K and M/M/m/m+r), at each time window of length T, the request rate that arrive at the system when there are zero or one packets in the queue is given by $X_0 \sim \text{Poisson}(\lambda T \boldsymbol{\pi}_0)$ and $X_1 \sim \text{Poisson}(\lambda T \boldsymbol{\pi}_1)$, respectively, where λ is the request rate. The latter sampling is a good approximation of the aggregate observations since T is large enough for the system to reach steady state and, by the PASTA property (Wolff 1982), Poisson processes see time averages. In the training data, we observe as few as 10 samples for each $\boldsymbol{\pi}$ distribution.

Queue Simulation Details. All the queue simulations share the same data generation process given in Algorithm 1 where N is number of samples to generate, λ_{min} and λ_{max} are inclusive boundaries for x^{train} or x^{test}, and $\theta^\star$ is ground truth settings of all learnable parameters.

Supplementary Material C3: Effect of Parametric Strength α

In Figure 4, we can see that strong parametric model is a clear winner among all experiments. The failure of weak parametric models on extrapolation of all tasks may be caused by the failure on the convergence over training loss. To exclude that factor, we further investigate the training loss on those tasks in Table 3, and find that all parametric model strengths gives similar training loss.

Algorithm 1 Data Generation Process

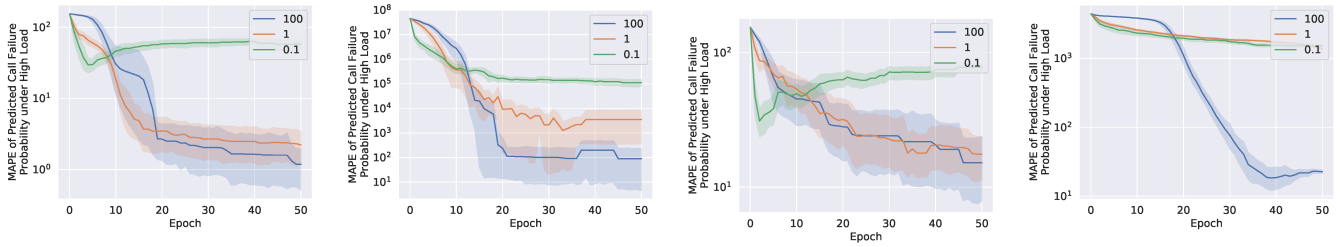
Require: $N, \lambda_{\min}, \lambda_{\max}, \boldsymbol{Q}(\cdot)$
  // $\boldsymbol{Q}(\cdot)$ is the queueing model that takes the request rate x as input.
  $T \leftarrow 1$
  $n \leftarrow 0$
  while $n < N$ **do**
    $x \sim \text{Uniform}(\lambda_{\min}, \lambda_{\max})$
    Get transition probability matrix $\boldsymbol{P}$ and logging rate γ of $\boldsymbol{Q}(x; \theta^\star)$ as Definition 1
    Get steady-state distribution $\boldsymbol{\pi}$ of $\boldsymbol{Q}(x; \theta^\star)$
    $t \leftarrow 0$
    $i \leftarrow 0$
    Sample initial state s_0 with respect to $\boldsymbol{\pi}$
    Sample interval between next event $d \sim \text{Exp}(\frac{1}{\gamma})$
    $t \leftarrow t + d$
    while $t < T$ **do**
      // Reach to next state.
      $i \leftarrow i + 1$
      Sample staying state of next event s_i with respect to $P_{s_i:}$
      Sample interval between next event $d \sim \text{Exp}(\frac{1}{\gamma})$
      $t \leftarrow t + d$
    end while
    Count the appearance of $\{s_0, \cdots, s_i\}$ and append count to data
    $n \leftarrow n + 1$
  end while

This conclusion verifies the need of parametric model: without parametric model, our models can easily overfit on partly observed states $\mathbb{S}'$, and lose the extrapolation ability on unseen states $\mathbb{S} \backslash \mathbb{S}'$.

Supplementary Material C4: Effect of Geometric Parameter $p^{(h)}$

Theorem 1 gives a bound on $p^{(h)} < \delta^{(h)}$, where $\delta^{(h)}$ is the spectral gap of $\boldsymbol{P}(\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\theta}^{(h)}))$ on the training data, to guarantee that ∞-SGD converges. This bound is loose. In our experiments, $\delta^{(h)}$ can always reach magnitude of 0.01, and even reach magnitude of 0.001 for slow mixing M/M/1/K. This would require $p^{(h)} \leq 0.01$ to guarantee convergence, and for slow mixing M/M/1/K, it would be $p^{(h)} \leq 0.001$. In the following experiments we see that such small values of $p^{(h)}$ are not required in practice.

Note that the smaller $p^{(h)}$ is, the more likely we will sample a large $X^{(h)}$, which will greatly increases the cost of computing $\Gamma^{(h)}_{ijmn}$ of Equation (11). We then test ∞-SGD with $p^{(h)} = 0.1$ and $p^{(h)} = 0.01$. For slow mixing M/M/1/K, $p^{(h)} = 0.001$ is also included. The MAPE error results are provided in Figure 5. We can see that $p^{(h)} = 0.01$ similar performance as $p^{(h)} = 0.1$



(a) Effect of parametric model on M/M/1/K (fast mixing) ($\alpha \in \{0.1, 1, 100\}$).

(b) Effect of parametric model on M/M/1/K (slow mixing) ($\alpha \in \{0.1, 1, 100\}$).

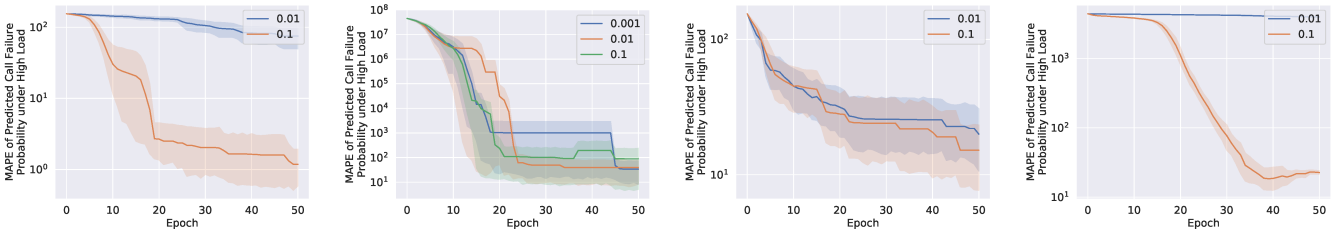
(c) Effect of parametric model on M/M/m/m+r ($\alpha \in \{0.1, 1, 100\}$).

(d) Effect of parametric model on M/M/Multiple/K ($\alpha \in \{0.1, 1, 100\}$).

Figure 4: Effect of parametric strength on different queues on test error (all optimized with ∞-SGD). (a) and (c) shows that strong strength ($\alpha = 100$) sometimes may be same as weaker strength ($\alpha = 1$). (b) and (d) shows that strong prior ($\alpha = 100$) beats small prior ($\alpha = 1$). (a-d) shows that too small strength ($\alpha = 0.1$) will be harmful. (a-d) together shows that using strong prior ($\alpha = 100$) does not harm, and may even achieve great improvement.

Table 3: Simulation results showing training loss (negative log-likelihood). We can see that all training losses have similar mean and variance regardless of parametric strength α.

	δ_n (spectral gap)	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 100$
M/M/1/K (fast-mix)	[0.022, 0.043]	$3.59 \times 10^1 \pm 7.33 \times 10^{-1}$	$3.59 \times 10^1 \pm 8.02 \times 10^{-1}$	$3.60 \times 10^1 \pm 7.45 \times 10^{-1}$
M/M/1/K (slow-mix)	[0.005, 0.008]	$8.11 \pm 3.42 \times 10^{-1}$	$8.12 \pm 4.47 \times 10^{-1}$	$8.22 \pm 4.71 \times 10^{-1}$
M/M/m/$m + r$	[0.013, 0.024]	$2.22 \times 10^1 \pm 1.64$	$2.23 \times 10^1 \pm 1.60$	$2.19 \times 10^1 \pm 1.39$
M/M/Multiple/K	[0.068, 0.096]	$9.30 \times 10^1 \pm 1.46$	$9.51 \times 10^1 \pm 1.48$	$9.33 \times 10^1 \pm 1.60$



(a) Effect of geometric sampling on M/M/1/K (fast-mix) ($p \in \{0.01, 0.1\}$).

(b) Effect of geometric sampling on M/M/1/K (slow-mix) ($p \in \{0.001, 0.01, 0.1\}$).

(c) Effect of geometric sampling on M/M/m/m+r ($p \in \{0.01, 0.1\}$).

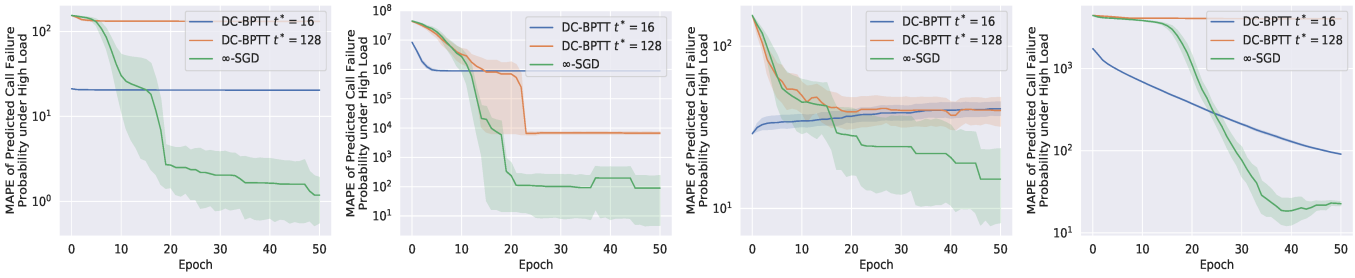
(d) Effect of geometric sampling on M/M/Mutiple/K ($p \in \{0.01, 0.1\}$).

Figure 5: Effect of geometric sampling on different queues on test error (all optimized with ∞-SGD). (a-d) together show that taking geometric sampling with $p = 0.1$ gives nearly the same performance as $p = 0.01$.

on slow mixing M/M/1/K and M/M/m/m+r, and fails to converge in 50 epochs on fast mixing M/M/1/K and M/M/Multiple/K. In addition, for M/M/1/K, $p^{(h)} = 0.001$ also gives similar result as $p^{(h)} = 0.1$. For the tasks where small $p^{(h)}$ converges, the time cost for each each epoch of $p^{(h)} = 0.1$ is cut to half of $p^{(h)} = 0.01$, and is cut to one fourth of $p^{(h)} = 0.001$. For those where small $p^{(h)}$ does not converge, we may need even more epochs for $p^{(h)} = 0.01$. We note however, that the gains in training are not reflected in the generalization performance, as shown in the training losses for $p^{(h)} = 0.1$ and $p^{(h)} = 0.01$ in Table 4. Thus, we use $p^{(h)} = 0.1$ in most of our experiments.

Supplementary Material C5: Other Synthetic Result Details

Training Curves. Looking at the training curves of synthetic results in Figure 6 and Figure 7, we can clearly see that DC-BPTT has a gradient vanishing problem, while ∞-SGD always computes useful gradients.



(a) M/M/1/K (fast-mix)

(b) M/M/1/K (slow-mix)

(c) M/M/m/$m + r$

(d) M/M/Multiple/K

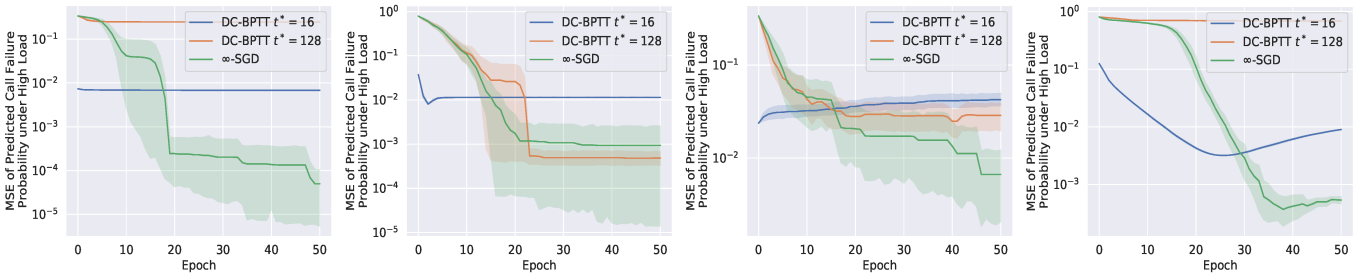
Figure 6: Training curves of MAPE on synthetic experiments.

Table 4: Simulation results showing training loss (negative log-likelihood) and variance on failure states are of the same magnitude between difference p settings.

	δ_n (spectral gap)	$p = 0.1$	$p = 0.01$
M/M/1/K (fast-mix)	[0.022, 0.043]	$3.87 \times 10^1 \pm 1.96$	$3.59 \times 10^1 \pm 7.33 \times 10^{-1}$
M/M/1/K (slow-mix)	[0.005, 0.008]	$8.25 \pm 3.75 \times 10^{-1}$	$8.11 \pm 3.42 \times 10^{-1}$
M/M/m/$m + r$	[0.013, 0.024]	$2.21 \times 10^1 \pm 1.68$	$2.22 \times 10^1 \pm 1.64$
M/M/Multiple/K	[0.068, 0.096]	$1.12 \times 10^2 \pm 8.88 \times 10^{-1}$	$9.30 \times 10^1 \pm 1.46$

Table 5: Parameter configurations for synthetic training and test data whose spectral gap is controlled.

		Training		Test	
	μ	$\lambda_{\text{min}}^{\text{train}}$	$\lambda_{\text{max}}^{\text{train}}$	$\lambda_{\text{min}}^{\text{test}}$	$\lambda_{\text{max}}^{\text{test}}$
M/M/1/K (Gap Controlled)	25	11	40	11	40
M/M/m/m+r (Gap Controlled)	25	21	30	11	40
M/M/Multiple/K (Gap Controlled)	[15, 10, 5]	16	45	16	45



(a) M/M/1/K (b) M/M/1/K (slow-mix) (c) M/M/m/$m + r$ (d) M/M/Multiple/K

Figure 7: Training curves of MSE on synthetic experiments.

Other Synthetic Configurations. In our experiments, we also tried some variations of the simulated transition rate parameters. Here, we control the spectral gap on training and test data to have a better understanding of our methods on slow mixing events. These transition rates are listed in Table 5.

Final results on those configurations are provided in Table 6 (MAPE) and Table 7 (MSE). We can see that ∞-SGD still outperforms DC-BPTT on MAPE metric, while is only comparable to DC-BPTT $t^\star = 16$ on MSE metric in the M/M/m/m+r case. Thus, ∞-SGD shows to be a much more stable optimization method.

Table 6: [MAPE] Simulation results showing MAPE/100 errors between predicted steady state and ground-truth for failure states in data whose spectral gap is controlled.

	δ_n (spectral gap)	DC-BPTT $t^\star = 16$	DC-BPTT $t^\star = 128$	∞-SGD ($p = 0.1$)
M/M/1/K (gap controlled)	$[0.005, 0.032]$	$2.55 \times 10^4 \pm 3.37 \times 10^3$	$9.06 \times 10^4 \pm 5.03 \times 10^4$	$\mathbf{6.06 \times 10^{-1} \pm 5.91 \times 10^{-1}}$
M/M/m/$m + r$ (gap controlled)	$[0.010, 0.027]$	$1.32 \times 10^3 \pm 3.01 \times 10^1$	$1.31 \times 10^2 \pm 4.27 \times 10^1$	$\mathbf{5.14 \times 10^1 \pm 3.29 \times 10^1}$
M/M/Multiple/K (gap controlled)	$[0.012, 0.057]$	$1.19 \times 10^3 \pm 1.07 \times 10^2$	$2.88 \times 10^5 \pm 9.11 \times 10^3$	$\mathbf{6.69 \times 10^{-1} \pm 7.00 \times 10^{-1}}$

Table 7: [MSE] Simulation results showing MSE errors between predicted steady state and ground-truth for failure states in data whose spectral gap is controlled.

	δ_n (spectral gap)	DC-BPTT $t^\star = 16$	DC-BPTT $t^\star = 128$	∞-SGD ($p = 0.1$)
M/M/1/K (gap controlled)	$[0.005, 0.032]$	$8.56 \times 10^{-3} \pm 9.92 \times 10^{-5}$	$2.13 \times 10^{-1} \pm 1.15 \times 10^{-1}$	$\mathbf{5.00 \times 10^{-4} \pm 6.28 \times 10^{-4}}$
M/M/m/$m + r$ (gap controlled)	$[0.010, 0.027]$	$\mathbf{2.27 \times 10^{-2} \pm 9.17 \times 10^{-4}}$	$\mathbf{4.57 \times 10^{-2} \pm 1.37 \times 10^{-2}}$	$\mathbf{3.97 \times 10^{-2} \pm 1.61 \times 10^{-2}}$
M/M/Multiple/K (gap controlled)	$[0.012, 0.057]$	$4.34 \times 10^{-5} \pm 2.89 \times 10^{-6}$	$5.61 \times 10^{-1} \pm 1.24 \times 10^{-2}$	$\mathbf{4.00 \times 10^{-7} \pm 5.15 \times 10^{-7}}$