Generative Differentiable Discrete Event Simulation (DES)

Empirical Evidence for Service Time Distribution Estimation

Yufeng, Mar 13 2025, ICS Conference

Given a M/M/1queue,
assume we can only observe the average waiting time,
how can we recover the service time distribution?

• Single server queue, FIFO • Average waiting time • Arrival process is Poisson process with known rate • Service time follows exponential distribution with unknown parameter	⇒	Service time distribution

M/M/1: Single server queue with Poisson arrival and exponential service time distribution

Given a M/M/1queue, assume we can only observe the average waiting time, how can we recover the service time distribution?

- Analytical solution
  - $$\text{Average waiting time} = \text{Average service time}\left(\frac{\text{Average service rate}}{1 - \text{Server utilization}}\right)$$
- This equation is derived from steady-state of a Continuous Time Markov Chain.

M/M/1: Single server queue with Poisson arrival and exponential service time distribution

Theorem 4. *Given a M/M/1 queuing system with arrival rate λ, average service time T, average service rate μ, and server utilization u, the average time an item spends in the queue W is given by the following.*

$$W = T\left(\frac{u}{1-u}\right) \tag{5}$$

Proof. Consider that the M/M/1 queuing system has a Poisson arrival process. For any time t and an interval Δt, the arrival rate λ admits the following probabilities.

$$\Pr\{\text{an item arrives during the interval } (t, t+\Delta t)\} = \lambda\Delta t + o(\Delta t)$$
$$\Pr\{\text{more than one arrival occurs during } (t, t+\Delta t)\} = o(\Delta t)$$

Similarly, as the servicing process is Poisson with rate μ, the probability that an item is serviced and more than one item are serviced are the following.

$$\Pr\{\text{an item is serviced during the interval } (t, t+\Delta t)\} = \mu\Delta t + o(\Delta t)$$
$$\Pr\{\text{more than one item is serviced during } (t, t+\Delta t)\} = o(\Delta t)$$

Because the number of arrivals and serviced items are both independent values within overlapping time intervals, the M/M/1 queuing system is a birth-death process where an arrival signifies a birth and an item being serviced signifies a death. The states $\mathcal{X}$ determine the number of items within the queuing system and the birth and death rates are $\lambda_i = \lambda$ and $\mu_i = \mu$ respectively for every state $i \in \mathcal{X}$.

We are interested in determining N the average number of items in the queuing systems, or in this case the average population size in the birth-death process. Consider the steady-state probability of being in state i. If $i = 0$, then via equation 4, this is the following.

$$p_0 = \left(1 + \sum_n^{\infty}\prod_{i=1}^{n}\frac{\lambda_{i-1}}{\mu_i}\right)^{-1} = \left(1 + \sum_n^{\infty}\prod_{i=1}^{n}\frac{\lambda}{\mu}\right)^{-1} = \frac{1}{1+\sum_{n=1}^{\infty}\left(\frac{\lambda}{\mu}\right)^n} = \frac{1}{\sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^n}$$

Notice that $u = \frac{\lambda}{\mu}$ and if $0 \le u < 1$, then the sum in the denominator converges to the following.

$$p_0 = \frac{1}{\sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^n} = 1 - \frac{\lambda}{\mu} = 1 - u$$

For any other $n > 0$, equation 3 dictates the steady-state probability of being in state n as given by the following.

$$p_n = p_0\prod_{i=1}^{n}\frac{\lambda_{i-1}}{\mu_i} = p_0\left(\frac{\lambda}{\mu}\right)^n = (1-u)u^n$$

Thus for any $i \in \mathcal{X}$ we have $p_i = (1-u)u^i$. Since we know the probability that there are i items inside the system for any i, we can calculate N the expected number of items inside the queue as follows.

$$\begin{aligned}
\mathbf{E}\{\#\text{ items inside queue}\} &= \sum_{i=0}^{\infty} i p_i = \sum_{i=0}^{\infty} i u^i (1-u) \\
&= u(1-u)\sum_{i=0}^{\infty} i u^{i-1} = u(1-u)\sum_{i=0}^{\infty}\frac{d}{du}(u^i) \\
&= u(1-u)\,\frac{d}{du}\left(\sum_{i=0}^{\infty} u^i\right) = u(1-u)\,\frac{d}{du}\left(\frac{1}{1-u}\right) \\
&= u(1-u)\,\frac{1}{(1-u)^2} = \frac{u}{1-u}
\end{aligned}$$

Now by Little's Law, the average time spent inside the queuing system is $\tau = \frac{N}{\lambda}$ or...

$$\tau = \frac{N}{\lambda} = \frac{u}{\lambda(1-u)} = \frac{1}{\lambda - \mu}$$

Finally, the time spent inside the system is simply the time spent waiting in the queue and the time spent being serviced. That is the following.

$$\tau = W + T \quad \Longleftrightarrow \quad W = \tau - T = \frac{1}{\lambda-\mu} - \frac{1}{\mu} = T\left(\frac{u}{1-u}\right)$$

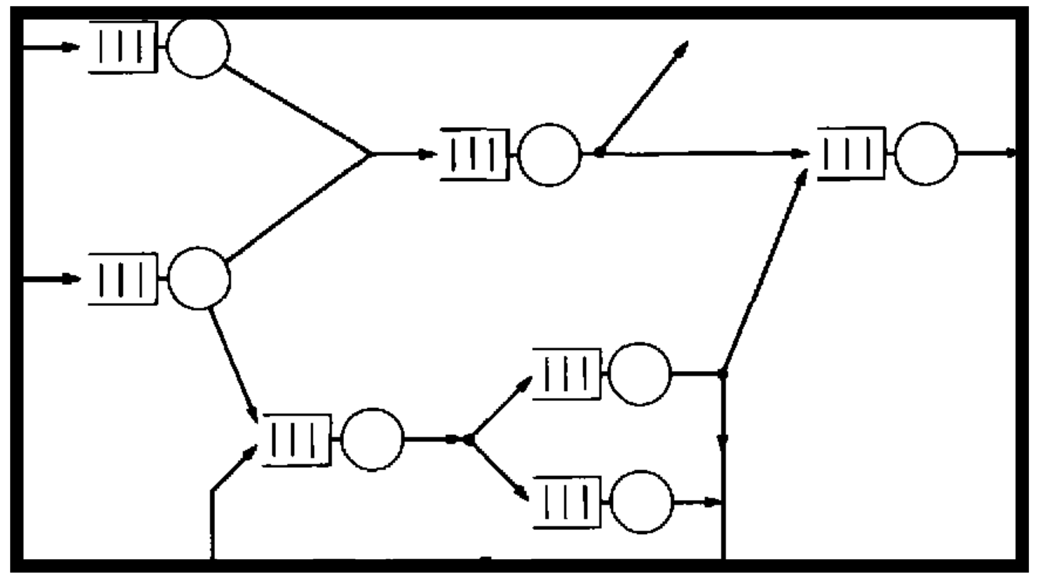
We thus have $W = T(\frac{u}{1-u})$ as required. □

https://antaresc.github.io/src/documents/classes/cs162-mm1-mg1.pdf

Given a M/G/1queue,
assume we can only observe the average waiting time,
how can we recover the service time distribution?

• Single server queue, FIFO • Average waiting time • Arrival process is Poisson process with known rate	$\Rightarrow$	Service time distribution

M/G/1: Single server queue with Poisson arrival and general service time distribution

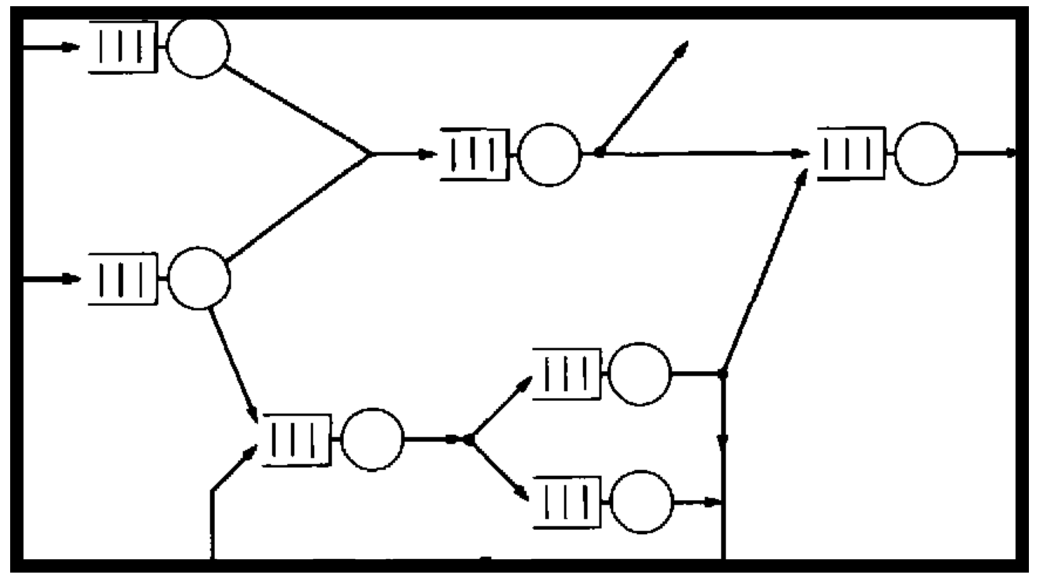


Given a queuing network with time-varying arrival rate and general service time distribution,
assume we can only observe the exit time of customers,
how can we find arrival process and service time distribution that could lead to the observed exit time?

- Routing Mechanisms, FIFO
- Exit time of customers

⇒

Service time distribution



Given a general queuing network and some flexible observation of the system,
how can we recover the service time distribution and arrival process?

- Routing Mechanisms, FIFO
- Exit time of customers

⇒

Service time distribution

Background

Discrete Event Simulation

- Discrete event simulation: simulates the operation of a system as a sequence of discrete events occurring at specific points in time.
  - The system state changes only at distinct times when events occur.
  - Events represent specific points of action (e.g., arrival, departure, failure).
- Examples:
  - Manufacturing systems (e.g., production lines).
  - Transportation (e.g., traffic flow, logistics).
  - Service systems (e.g., call centers, hospitals).

Background

Discrete Event Simulation

- Deterministic + Stochastic
- Deterministic Components:
  - **Service Discipline** Rules governing the order in which customers are served, such as First-In-First-Out (FIFO) or Last-In-First-Out (LIFO).
  - **Routing Mechanisms** Predetermined paths that customers follow through the system, including how they transition between servers.
  - **Number of Servers** The fixed count of service channels available in the system.

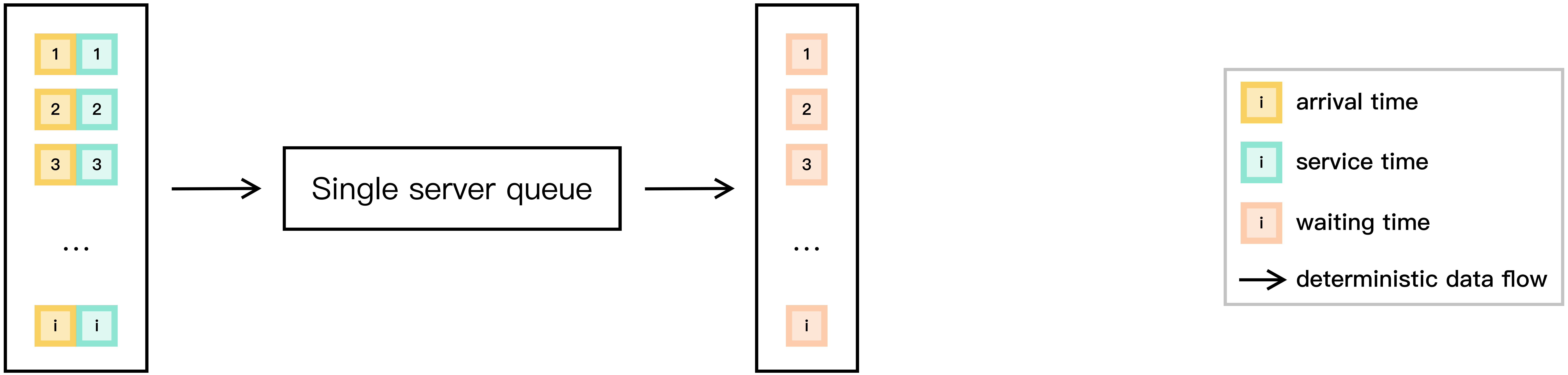
Background

Discrete Event Simulation

- Deterministic + Stochastic
- Stochastic Components:
    - **Arrival Processes**
    - **Service Times**
    - **Interarrival Times**

Background

Discrete Event Simulation - Single Server Queue



Background

Discrete Event Simulation - Single Server Queue

- Given arrival times and service times for each arrival, we can compute the waiting time and exit time recursively:
  - $W_{n+1} = \max\left(0, W_n + U_n\right)$
    - T_n is the time between the n-th and $(n+1)$-th arrivals.
    - S_n is the service time of the n-th customer, and
    - $U_n = S_n - T_n$
    - W_n is the waiting time of the n-th customer.
- We can implement this simulation algorithm in PyTorch!

Background

Discrete Event Simulation - Single Server Queue

```
arrival_times = PoissonArrival(1).sample_up_to(10)
service_times = ExponentialService(1).sample(10)

# Given the arrival_times and service_times, compute the wait_times and exit_times

n_arrival = len(arrival_times)
relu = torch.nn.ReLU()
wait_times = torch.zeros(n_arrival)
exit_times = torch.zeros(n_arrival)

idle_time = 0  # Assume idle_time starts fresh for each batch
for i in range(n_arrival):
    wait_times[i] = relu(idle_time - arrival_times[i])
    idle_time = arrival_times[i] + wait_times[i] + arrival_times[i]
    exit_times[i] = idle_time

# get gradient of wait_times[i] w.r.t. arrival_times[j]
wait_times[i].backward()
grad = arrival_times.grad[j]
print("grad:", grad)
```

- Can compute the gradient of any output (waiting time and exit time) w.r.t. any input (arrival time and service time)!
- Same for general queueing network.

Generative Discrete Event Simulation

1. Sample input to the queuing system I_θ. Input can be arrival times and service times.
   - Poisson arrival with parameter λ.
   - Exponential service time distribution with parameter μ.
   - Discrete distribution with point masses at x_i and corresponding probabilities p_i.
   - General continuous distribution from a Gaussian Mixture Model with components $(\pi_k, \mathcal{N}(\mu_k, \sigma_k^2))$.
   - Poisson arrival with time varying arrival rate $\lambda(t)$.
   - Doubly stochastic Poisson arrival given by a generative model.

Generative Discrete Event Simulation

1. Sample input to the queuing system I_θ. Input can be arrival times and service times.
2. Run simulation $O = q(I_\theta)$
   - q: simulation algorithm.
   - O: simulation output.

Generative Discrete Event Simulation

1. Sample input to the queuing system I_θ. Input can be arrival times and service times.
2. Run simulation $O = q(I_\theta)$
3. Compute loss using simulation output $M = l(O)$
   - l: some (differentiable) loss function.
     - For calibration task
       - (simulated average waiting time - target average waiting time)^2
       - (simulated average waiting time - target average waiting time)^2 $+w \cdot$ (simulated variance of waiting time - target variance of waiting time)^2
       - Wasserstein distance between simulated waiting time and real waiting time from data.
     - For optimization task
       - minimize average waiting time[1]

1. subject to some constraint, and solve via projected SGD, Lagrangian methods, etc,.

Differentiable Discrete Event Simulation

1. Sample input to the queuing system I_θ. Input can be arrival times and service times.
2. Run simulation $O = q(I_\theta)$
3. Compute loss using simulation output $M = l(O)$
4. Summarize
    - Want to solve $\min_\theta \mathbb{E}[l(q(I_\theta))]$, which requires computing gradients w.r.t. θ for optimization.
    - $\nabla_{I_\theta} q(I_\theta)$ can be computed easily.
    - Since the expectation involves a stochastic process, we use stochastic gradient estimation to approximate $\nabla_\theta \mathbb{E}[l(q(I_\theta))]$.

Background

Stochastic Gradient Estimation

- Given some utility function (or cost function) U, the expected utility is defined as

$$\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x};\boldsymbol{\theta})}[U(\boldsymbol{x})] = \int U(\boldsymbol{x})p(\boldsymbol{x};\boldsymbol{\theta})d\boldsymbol{x}$$

- Gradients w.r.t. distributional parameters θ of the expected utility:

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x};\boldsymbol{\theta})}[U(\boldsymbol{x})]$$

- Commonly seen in Variational inference, Reinforcement learning, etc,.
- Two common gradient estimators: Score-Function Gradient Estimators & Pathwise Gradient Estimators

https://mml-book.github.io/neurips2020/12-stochastic-gradient.pdf

Background

Stochastic Gradient Estimation - Score-Function Gradient Estimator

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\theta})}[U(\boldsymbol{x})] &= \nabla_{\boldsymbol{\theta}} \int U(\boldsymbol{x}) p(\boldsymbol{x};\boldsymbol{\theta}) d\boldsymbol{x} && \text{Expectation as integration} \\
&= \int U(\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} p(\boldsymbol{x};\boldsymbol{\theta}) d\boldsymbol{x} && \text{Move gradient inside} \\
&= \int U(\boldsymbol{x}) p(\boldsymbol{x};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x};\boldsymbol{\theta}) d\boldsymbol{x} && \text{Log-derivative trick} \\
&= \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\theta})}[U(\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x};\boldsymbol{\theta})] && \text{Integral as expectation}
\end{aligned}
$$

- Monte-Carlo estimator: $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\theta})}[U(\boldsymbol{x})] \approx \frac{1}{S} \sum_{s=1}^{S} U(\boldsymbol{x}^{(s)}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}^{(s)};\boldsymbol{\theta}), \quad \boldsymbol{x}^{(s)} \sim p(\boldsymbol{x};\boldsymbol{\theta})$

https://mml-book.github.io/neurips2020/12-stochastic-gradient.pdf

Background

Stochastic Gradient Estimation - Score-Function Gradient Estimator

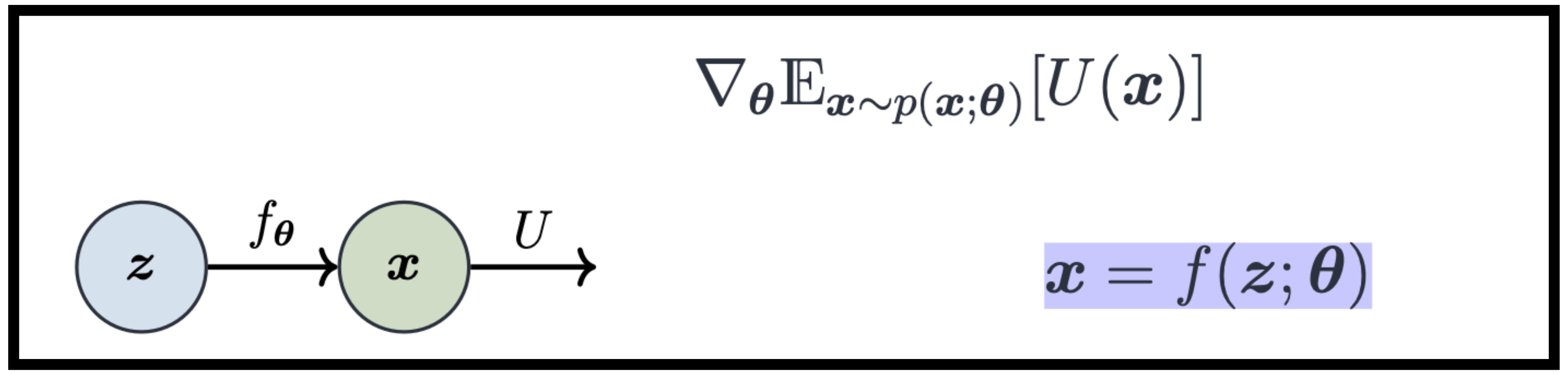
- Monte-Carlo estimator: $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\theta})}[U(\boldsymbol{x})] \approx \frac{1}{S} \sum_{s=1}^{S} U(\boldsymbol{x}^{(s)}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}^{(s)};\boldsymbol{\theta}), \quad \boldsymbol{x}^{(s)} \sim p(\boldsymbol{x};\boldsymbol{\theta})$
- p (pdf of x) must be differentiable w.r.t. θ
- Techniques to control the variance of the estimator (e.g., Greensmith et al., 2004; Titsias & Lázaro-Gredilla, 2015)

https://mml-book.github.io/neurips2020/12-stochastic-gradient.pdf

Background

Stochastic Gradient Estimation - Pathwise Gradient Estimators

- Assume data x can be obtained by a deterministic transformation f (path) of a latent variable $z \sim p(z)$, where $p(z)$ has no tunable parameters, e.g., $p(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$



$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\theta})}[U(\boldsymbol{x})] &= \nabla_{\boldsymbol{\theta}} \int U(\boldsymbol{x}) p(\boldsymbol{x};\boldsymbol{\theta}) d\boldsymbol{x} && \text{Expectation as integration} \\
&= \nabla_{\boldsymbol{\theta}} \int U((\boldsymbol{z};\boldsymbol{\theta})) p(\boldsymbol{z}) d\boldsymbol{z} && \text{Change of variables} \\
&= \int \nabla_{\boldsymbol{\theta}} U(f(\boldsymbol{z};\boldsymbol{\theta})) p(\boldsymbol{z}) d\boldsymbol{z} && \text{Move gradient inside} \\
&= \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\nabla_{\boldsymbol{\theta}} U(\underbrace{f(\boldsymbol{z};\boldsymbol{\theta})}_{=\boldsymbol{x}})] && \text{Integral as expectation}
\end{aligned}
$$

- Monte-Carlo estimator: $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\theta})}[U(\boldsymbol{x})] \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\theta}} U(f(\mathbf{z}^{(s)};\boldsymbol{\theta})), \quad \mathbf{z}^{(s)} \sim p(z)$

- with $\nabla_{\theta} U(f(\mathbf{z};\theta)) = \nabla_{\mathbf{x}} U(\mathbf{x}) \, \nabla_{\theta} f(\mathbf{z};\theta)$ (chain rule)

https://mml-book.github.io/neurips2020/12-stochastic-gradient.pdf

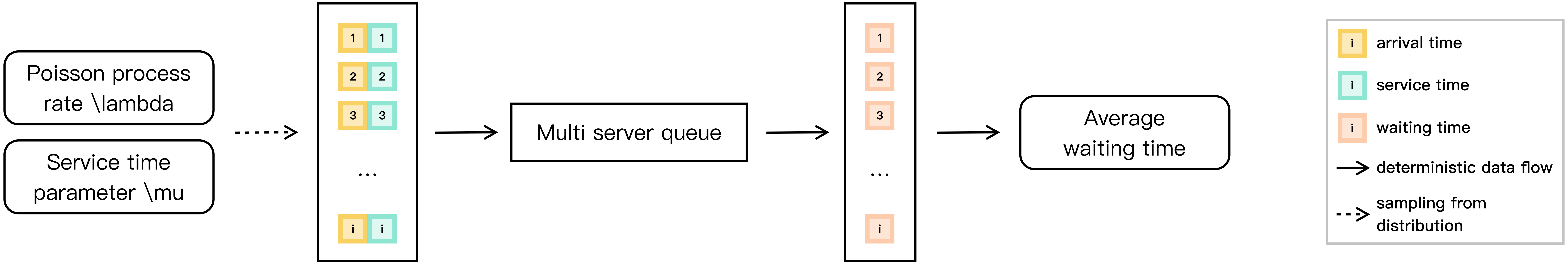
Background

Stochastic Gradient Estimation - Pathwise Gradient Estimators

- Monte-Carlo estimator: $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\theta})}[U(\boldsymbol{x})] \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\theta}} U(f(\boldsymbol{z}^{(s)};\boldsymbol{\theta})), \quad \boldsymbol{z}^{(s)} \sim p(\boldsymbol{z})$
- with $\nabla_\theta U(f(\mathbf{z};\theta)) = \nabla_{\mathbf{x}} U(\mathbf{x}) \nabla_\theta f(\mathbf{z};\theta)$ (chain rule)
- Utility U must be differentiable
- Path f must be differentiable

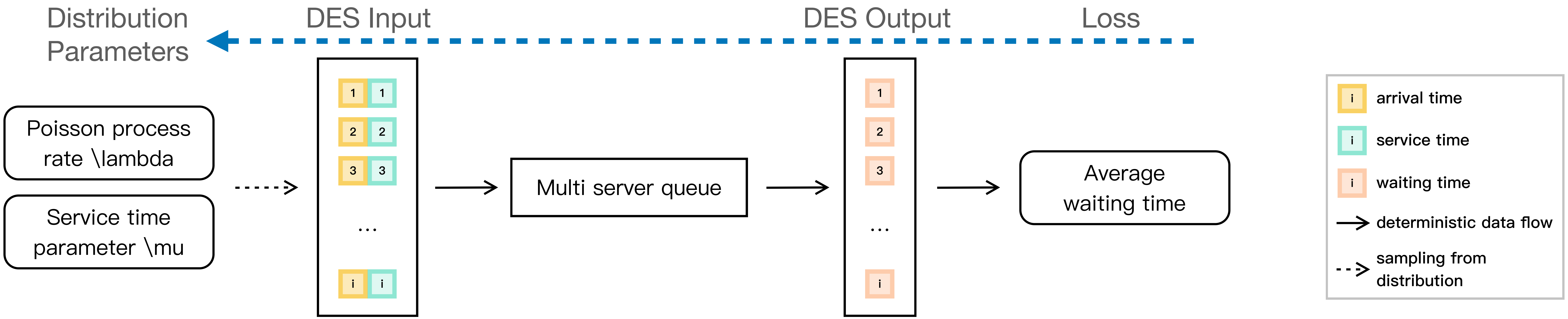
https://mml-book.github.io/neurips2020/12-stochastic-gradient.pdf

Generative Differentiable DES



- Generative View of DES:
  1. Sample arrivals and service times from given distributions (these might be simple distributions or even a generative model).
  2. Simulate in DES to generate outcomes.

Generative Differentiable DES

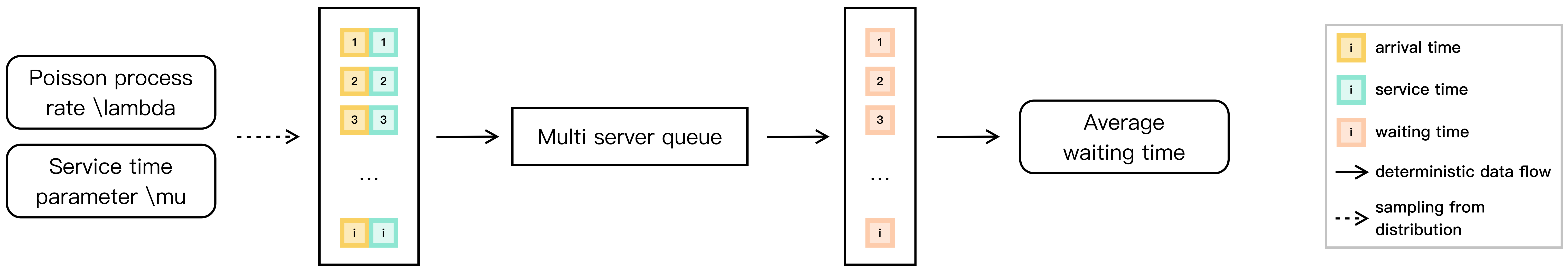


- Differentiable view of DES: Given a differentiable loss function,
  - We can compute gradients of performance metrics w.r.t. arrival and service time distribution parameters (e.g., arrival rate for Poisson process, parameters of a Generative model).

Generative Differentiable DES

Case Study

- M/M/1 queue.
- Arrival process: Poisson process with $\lambda = 0.95$
- Service time: Exponential with $\mu = 1$.
- $T = 500$.
- KNOWN: the mean and variance of waiting time, the arrival process. UNKNOWN: service time distribution.
- Given the observed mean and variance of the waiting time, we aim to recover the service time distribution.



Generative Differentiable DES

Case Study

- M/M/1 queue.
- Arrival process: Poisson process with $\lambda = 0.95$
- Service time: Exponential with $\mu = 1$.
- $T = 500$.
- KNOWN: the mean and variance of waiting time, the arrival process. UNKNOWN: service time distribution.
- Given the observed mean and variance of the waiting time, we aim to recover the service time distribution.
- Classical Queuing Methods
  - **Model Misspecification** The service time is assumed to follow a distribution from a restricted class, such as the exponential distribution, though this assumption may not always hold.
  - **Stationarity Requirement** Requires stationary performance; however, data from cold starts or short simulation runs (non-stationary cases) can still provide useful information.
  - **Complex Analysis** Derivations and computations can become cumbersome.

Generative Differentiable DES

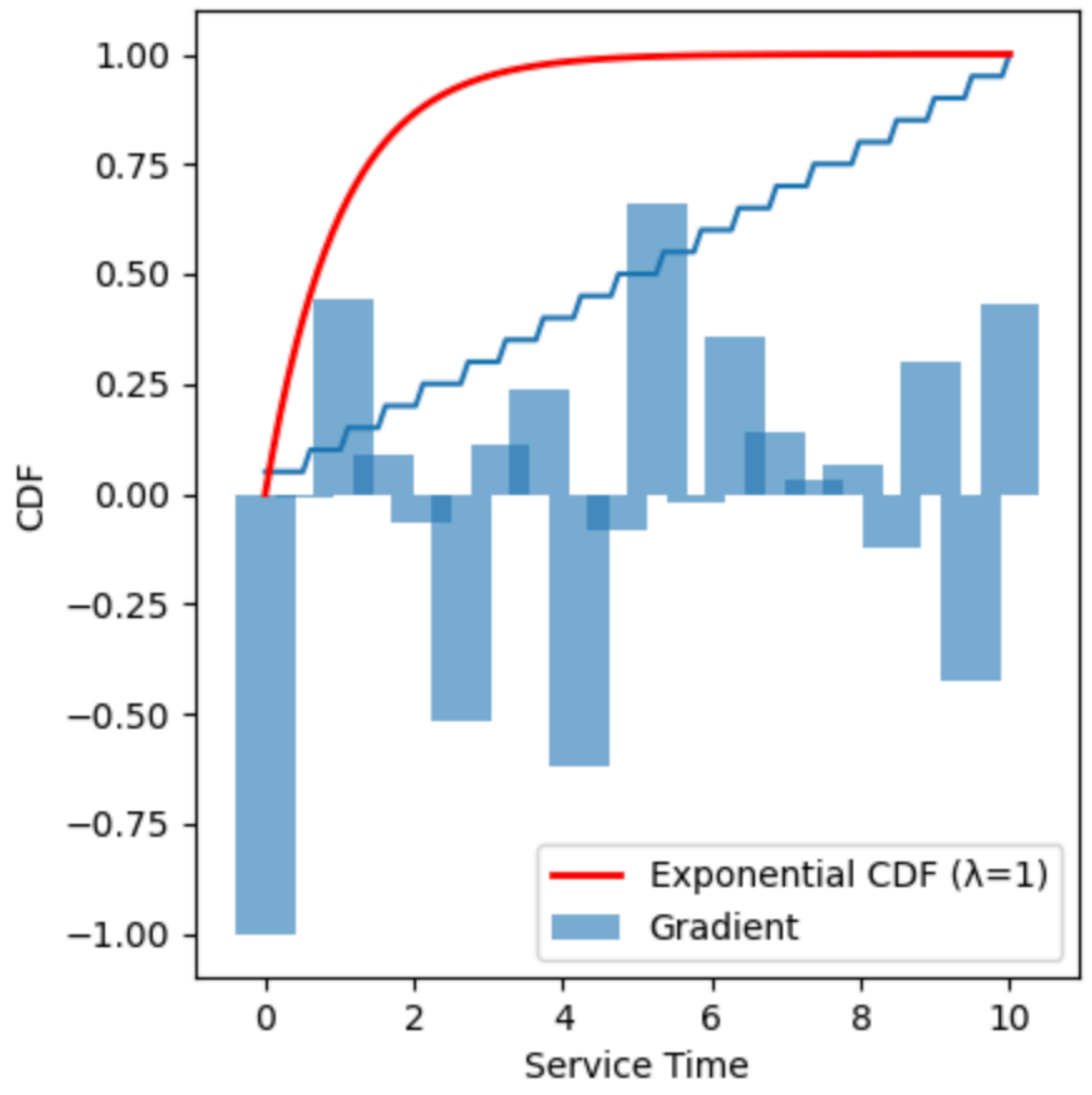
Case Study

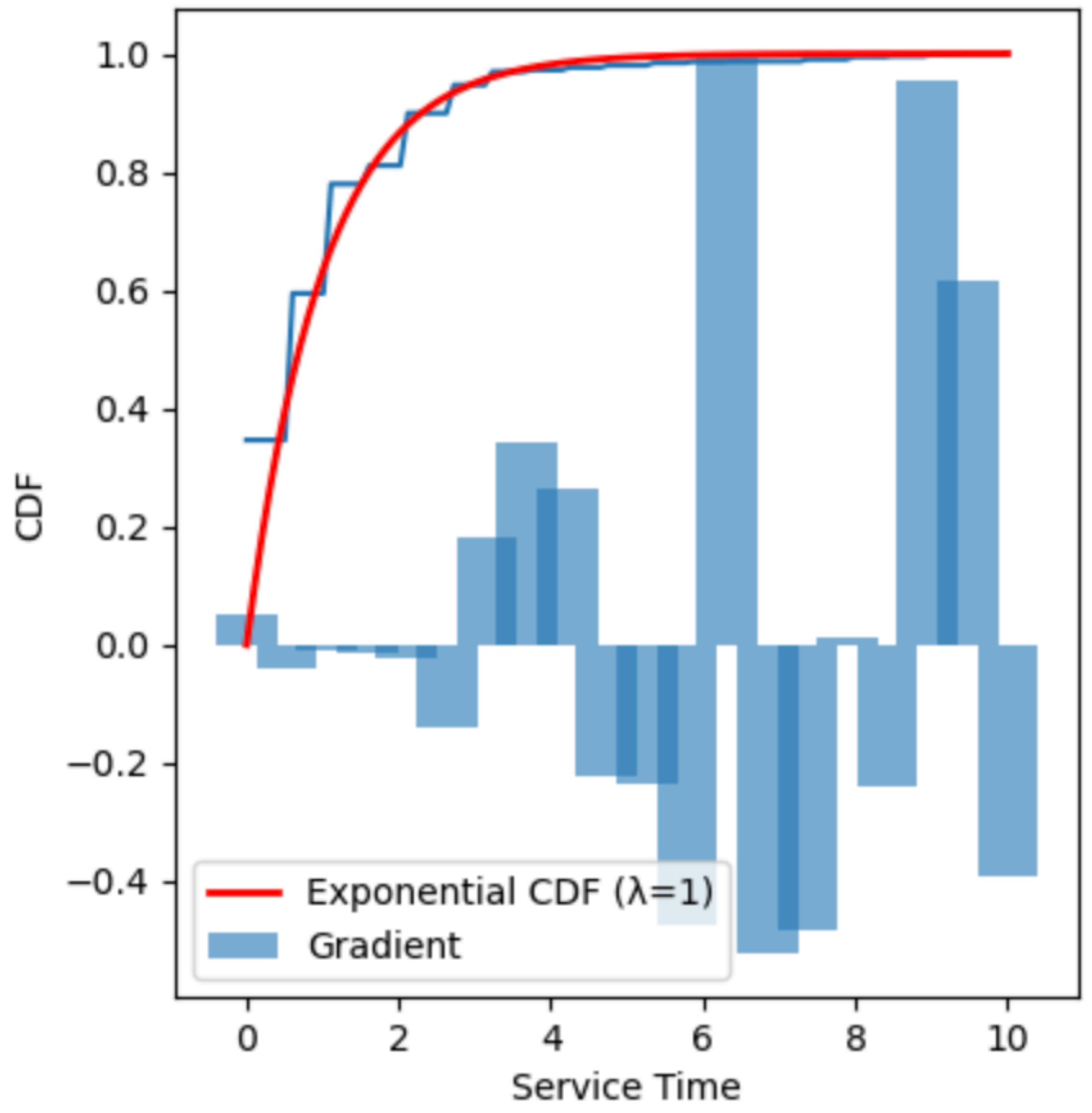
- Denote ground-truth mean and variance of waiting time as μ and σ^2.
- Parametrize service time distribution as Categorical distribution with weights of each mass point $\boldsymbol{\theta}$.
- Loss function: $l(q(\boldsymbol{\theta})) = (\mu_{\text{cat}}(\boldsymbol{\theta}) - \mu)^2 + \lambda \cdot (\sigma^2_{\text{cat}}(\boldsymbol{\theta}) - \sigma^2)^2$, λ is the weight parameter for balancing the terms.
- Algorithm
  1. Randomly initialize the Categorical distribution weights $\boldsymbol{\theta}^{(0)}$.
  2. Run simulation use $\boldsymbol{\theta}^{(i)}$, get one sample of $l(q(\boldsymbol{\theta}^{(i)}))$ and $\nabla_{\boldsymbol{\theta}^{(i)}} l(q(\boldsymbol{\theta}^{(i)}))$.
  3. Parameter update: $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)} + \eta \nabla_{\boldsymbol{\theta}^{(i)}} l(q(\boldsymbol{\theta}^{(i)}))$
  4. Go back to Step 2.

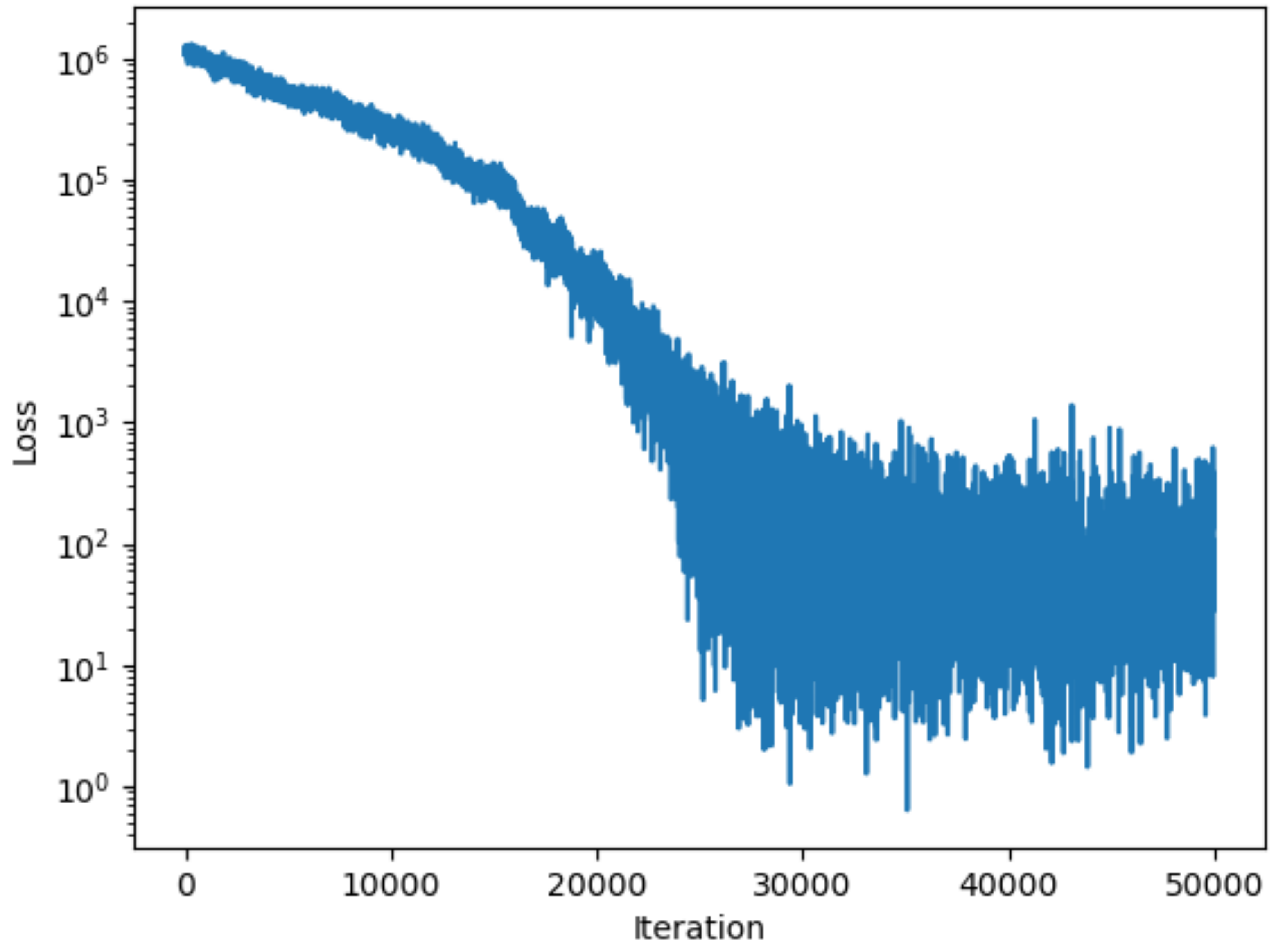
- M/M/1 queue.
- Arrival process: Poisson process with $\lambda = 0.95$
- Service time: Exponential with $\mu = 1$.
- $T = 500$.
- KNOWN: the mean and variance of waiting time, the arrival process.
- UNKNOWN: service time distribution.

Generative Differentiable DES

Case Study - Numerical Result







Generative Differentiable DES

Issues and Future Work

- **Identification Issue** For parameter estimation, the service time distribution that leads to the target average waiting time and variance might not be unique
- **Non-Convex Loss Landscape**
- **Variance reduction** Gradient estimator that has lower variance
- **Convergence guarantee**
- **Parallelization to speed up simulation**

Generative Differentiable DES

Issues and Future Work

- Other applications
  - Admission control
  - Routing mechanisms optimization
  - Minimizing waiting time
  - Maximizing throughput
  - Fairness
  - Online Resource Allocation
  - Experiment design
- As long as we can define a differentiable loss function.

Open to collaboration!

Github

Email