Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

LEARNING THE TANDEM NETWORK LINDLEY RECURSION

Sergio Palomo

Systems Engineering
Cornell University
Ithaca, NY 14850, USA

Jamol Pender

Operations Research and Information Engineering
Cornell University
Ithaca, NY 14850, USA

ABSTRACT

In this paper, we attempt to learn the tandem version of Lindley's recursion directly from data. We combine stochastic simulation with current machine learning methods such as Gaussian Processes, K-Nearest Neighbors, Linear Regression, Deep Neural Networks, and Gradient Boosted Trees to learn the tandem network Lindley recursion. We uncover specific parameter regimes where learning the tandem network Lindley recursion is easy or hard.

1 INTRODUCTION

Despite the simplicity of the Lindley recursion and its tandem network extension, it has not been extensively studied in a data driven context. Even though the queueing community has a rich history of using data to inform stochastic models, there are very few papers that use data to learn the actual stochastic models that one is using. There are some notable exceptions such as Sutton and Jordan (2011), Bodík et al. (2009), Sutton and Jordan (2010), Wang et al. (2016), Armony et al. (2015) and Palomo and Pender (2020), Kyritsis and Deriaz (2019), Nii et al. (2020). To this end, our goal is to understand how well current machine learning methods can recover tandem network Lindley's recursion from waiting time data of the tandem version of the M/M/1 queue. To study this problem, we unite stochastic simulation of the tandem queue with well known machine learning methods for learning or predicting waiting times from previous data.

Fortunately, in recent years machine learning and more specifically deep learning has seen unprecedented success in a diverse number of applications. Thus, our idea is to use machine learning methods to learn the tandem network Lindley recursion. In this paper, we will restrict our study to the single server tandem network case to understand the simplest of network queues. Our tandem queueing model is relevant for a variety of applications such as sports, transportation, healthcare, cloud computing, and even telecommunications, see for example Fu and Whitt (2015), Whitt (2015), Armony et al. (2015), Shah et al. (2019), Pender and Phung-Duc (2016), Nirenberg et al. (2018), Brown et al. (2005), Koole and Mandelbaum (2002), Novitzky et al. (2019) and Dunlay Jr and Park (1978).

1.1 Contributions of Our Work

In this paper, we make the following contributions to the literature:

- We aim to understand how well we can learn the tandem network Lindley recursion for the M/M/1 queue.
- We aim to understand under what regimes is it easy or hard to learn the tandem network Lindley recursion. For example, do current machine learning methods work well when the queue is in light traffic or heavy traffic?

- Finally, we aim to understand which machine learning methods are the best at the learning the tandem network and to understand any insights we can observe from how it learns the tandem network waiting times.

1.2 Organization of the Paper

The remainder of the paper is organized as follows. Section 2 introduces the Lindley recursion and gives a brief history of the formula. In Section 3, we compute several conditional moments for the Lindley recursion and a two dimensional tandem network. We also explain how to construct a new recursion for the tandem network. In Section 4, we present the results of the machine learning methods we analyze. Finally, a conclusion and information about future directions is given in Section 5.

2 SINGLE STATION LINDLEY RECURSION

Consider the G/G/1 queueing system model where the customer inter-arrival distribution is general, the service distribution is general, and there is a single server. For the n^{th} customer to arrive to the queue, we define W_n to be the waiting time for the n^{th} customer. We also define A_n to be the inter-arrival time between the n^{th} and the $(n+1)^{th}$ customers and S_n to be the n^{th} customer's service time. For the case of an infinite buffer and a First Come First Served (FCFS) system, Lindley's recursion was given by (Lindley 1952) is:

$$W_{n+1} = \max\left(W_n + S_n - A_n, 0\right). \tag{1}$$

Previous analysis such as (Prabhu 1974) studies the steady state waiting time of the G/G/1 queue and shows that the steady state waiting time satisfies an integral equation known as Lindley's integral equation. However, these results are in steady state and are not transient in nature. Complementary to the steady state waiting time, one can also calculate the conditional waiting time of the $(n+1)^{th}$ customer given the waiting time of the n^{th} customer. This conditional moment is important from a data perspective as it allows us to predict the waiting time of the next customer using only the current waiting time information of the previous customer. Moreover, we know that the prediction is optimal in the mean square sense. Thus, in the M/M/1 setting, we have the following theorem for the conditional mean waiting time.

Theorem 1 For the M/M/1 queue where the arrival rate is equal to λ and the service rate is equal to μ, then the conditional m^{th} moment of the waiting time of customer $n+1$ is equal to

$$\mathbb{E}[W_{n+1}^m | W_n] = \frac{\lambda\mu}{\lambda+\mu}\left(\sum_{k=0}^{m}\binom{m}{k}W_n^k \cdot \frac{\Gamma(m-k+1)}{\mu^{m-k+1}} + \frac{(-1)^m e^{-\lambda W_n}\left(\Gamma(m+1,-\lambda W_n) - \Gamma(m+1)\right)}{\lambda^{m+1}}\right).$$

where

$$\Gamma(m,x) = \int_x^\infty y^{m-1}e^{-y}dy \quad \text{and} \quad \Gamma(m) = \int_0^\infty y^{m-1}e^{-y}dy$$

are defined as the incomplete gamma function and the gamma function respectively.

Proof.

$$\begin{aligned}
\mathbb{E}[W_{n+1}^m|W_n] &= \mathbb{E}[(W_n - A_n + S_n)^{+^m}|W_n] \\
&= \int_{-\infty}^{\infty} (W_n+x)^{+^m} \frac{\lambda\mu}{\lambda+\mu} e^{-\mu max(x,0) - \lambda max(-x,0)} dx \\
&= \frac{\lambda\mu}{\lambda+\mu}\left(\int_0^{\infty} (W_n+x)^{+^m} e^{-\mu x} dx + \int_{-\infty}^{0} (W_n+x)^{+^m} e^{\lambda x} dx\right) \\
&= \frac{\lambda\mu}{\lambda+\mu}\left(\int_0^{\infty} (W_n+x)^{m} \cdot e^{-\mu x} dx + \int_{-W_n}^{0} (W_n+x)^{m} \cdot e^{\lambda x} dx\right) \\
&= \frac{\lambda\mu}{\lambda+\mu}\left(\int_0^{\infty} \left(\sum_{k=0}^{m} \binom{m}{k} W_n^k x^{m-k}\right) \cdot e^{-\mu x} dx + \int_{0}^{W_n} y^m \cdot e^{\lambda(y-W_n)} dy\right) \\
&= \frac{\lambda\mu}{\lambda+\mu}\left(\sum_{k=0}^{m} \binom{m}{k} W_n^k \cdot \frac{\Gamma(m-k+1)}{\mu^{m-k+1}} + \frac{(-1)^m e^{-\lambda W_n}\left(\Gamma(m+1,-\lambda W_n) - \Gamma(m+1)\right)}{\lambda^{m+1}}\right)
\end{aligned}$$

□

Corollary 2 For the M/M/1 queue where the arrival rate is equal to λ and the service rate is equal to μ, then the conditional mean and variance of the waiting time of customer $n+1$ is equal to

$$\begin{aligned}
\mathbb{E}[W_{n+1}|W_n] &= W_n + \frac{\lambda-\mu}{\lambda\mu} + \frac{\mu e^{-\lambda W_n}}{\lambda(\lambda+\mu)} \\
\text{Var}[W_{n+1}|W_n] &= W_n^2 \alpha_1(W_n)\cdot(1-\alpha_1(W_n)) + 2W_n\alpha_2(W_n)\cdot(1-\alpha_1(W_n)) - \alpha_2^2(W_n) + \alpha_3(W_n).
\end{aligned}$$

where

$$\begin{aligned}
\alpha_1(W_n) &= 1 - \frac{\mu}{\lambda+\mu} e^{-\lambda W_n} \\
\alpha_2(W_n) &= \frac{\mu}{\lambda(\lambda+\mu)} e^{-\lambda W_n}(1+\lambda W_n) - \frac{1}{\lambda} + \frac{1}{\mu} \\
\alpha_3(W_n) &= \frac{\mu}{\lambda^2(\lambda+\mu)} e^{-\lambda W_n}\left(2\lambda W_n - \lambda^2 W_n^2 - 2\right) - \frac{2\left(\frac{\lambda}{\mu}-1\right)}{\lambda^2} + \frac{2}{\mu^2}.
\end{aligned}$$

2.1 $H_\ell/H_n/1$ Queue

In the case where the arrival and service distributions are given by a hyper-exponential distribution, then we can even generalize the conditional waiting time formulas for the m^{th} moment. We prove this result in the following theorem.

Theorem 3 For the $H_\ell/H_n/1$ queue where the arrival rate vector and its associated probability vector is equal to $(\vec{\lambda},\vec{p_a})$ and the service rate vector and its associated probability vector is equal to $(\vec{\mu},\vec{p_s})$, then the conditional m^{th} moment of the waiting time of customer $n+1$ is equal to

$$\begin{aligned}
&\mathbb{E}[W_{n+1}^m|W_n] \\
&= \sum_{i=1}^{\ell}\sum_{j=1}^{n} p_{(a,i)}\cdot p_{(s,j)}\left[\frac{\lambda_i\mu_j}{\lambda_i+\mu_j}\left(\sum_{k=0}^{m}\binom{m}{k} W_n^k \cdot \frac{\Gamma(m-k+1)}{\mu_j^{m-k+1}} + \frac{(-1)^m e^{-\lambda_i W_n}\left(\Gamma(m+1,-\lambda_i W_n)-\Gamma(m+1)\right)}{\lambda_i^{m+1}}\right)\right].
\end{aligned}$$

Proof. The proof follows easily from conditioning on the state of the arrival rate and service rate. Then, we use the formula for the M/M/1 queue given in Theorem 1. □

3 TANDEM NETWORK LINDLEY RECURSION

Tandem queues occur when the departing units from one system become the new arrivals to a downstream system. To be more precise a tandem queue is a sequence of queues where the customers arrive to the first queue and once served in the first move on to the second and so forth until finishing service in every queue in the network. In a precise sense, the tandem network is the simplest queueing network one can have of a given size. An example of a tandem queue is given in Figure 1. In practice tandem queues can be observed in many applications. One of these applications is the health-care industry. In many health-care systems the service is set-up in sequence. In many of these systems before the primary health-care provider is seen it is necessary to first see the receptionist to fill out necessary paperwork, then it may be necessary to see the nurse to grab vital signs and information that the doctor may need. Another application of tandem network queues is transportation. In this application, the queues can be viewed as passenger stops, such as bus systems and train systems. In this case the entities we call 'customers' would represent a buses or trains. Each queue in the network represents a different stop, see Figure 2.

G/G/1
G/G/1
...
G/G/1
G/G/1

Figure 1: A tandem queue is a series of queues.

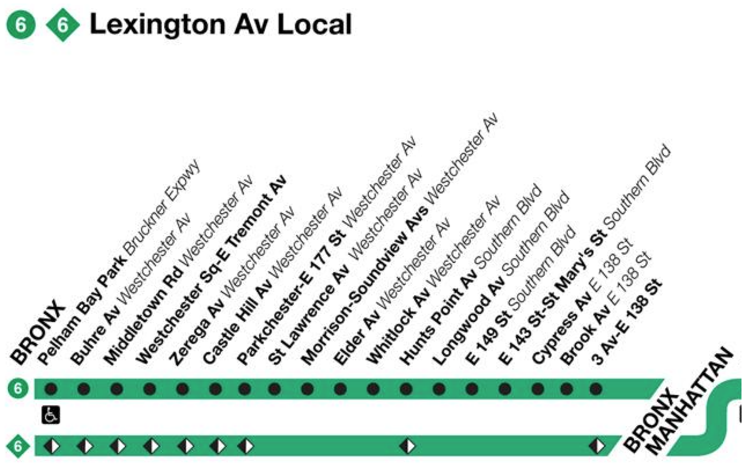
6
6
Lexington Av Local
BRONX
Pelham Bay Park Bruckner Expwy
Buhre Av Westchester Av
Middletown Rd Westchester Av
Westchester Sq-E Tremont Av
Zerega Av Westchester Av
Castle Hill Av Westchester Av
Parkchester-E 177 St Westchester Av
St Lawrence Av Westchester Av
Morrison-Soundview Avs Westchester Av
Elder Av Westchester Av
Whitlock Av Westchester Av
Hunts Point Av Southern Blvd
Longwood Av Southern Blvd
E 149 St Southern Blvd
E 143 St-St Mary's St Southern Blvd
Cypress Av E 138 St
Brook Av E 138 St
3 Av-E 138 St
BRONX
MANHATTAN

Figure 2: A train line is an example of a tandem queue, where the objects that are customers in the canonical model are the trains.

In this paper, we concentrate on a tandem queue where each station has one server. In the case of a tandem queue with single server stations the waiting time of a customer to receive service from the next server is given by the Lindley recursion for tandem queues. It is similar to Equation 1 except the variables are updated to display which queue the customer is waiting at. So the waiting time of the n$^{\text{th}}$ customer at the k$^{\text{th}}$ queue is given by:

$$W_{n+1}^{(k)} = \max\left(W_n^{(k)} + S_n^{(k)} - A_n^{(k)}, 0\right). \tag{2}$$

where $A_n^{(k)}$ is the inter-arrival time between the nth and (n+1)st customer at the kth queue and $S_n^{(k)}$ is the service time of the nth customer at the kth queue. A natural question that arises when analyzing a tandem queue is whether there is a way to make predictions for the waiting times of customer in the kth queue by incorporating information from any of the first, second, ..., or (k-1)st queue. The answer to that question is that there is a way and it is given in Proposition 1. The equation of interest the waiting time of the n th customer in the kth queue written as function of the difference in departure times between the nth and n+1st customers

$$W_n^{(k)} = \max(0, S_{n-1}^{(k)} + W_{n-1}^{(k)} - (D_n^{(k-1)} - D_{n-1}^{(k-1)})) \tag{3}$$

where D is simply the service completion time. How we get Equation 3 is that the arrival time of the nth customer at the kth queue is equal to the departure time of the nth customer at the (k-1)st queue. The new formula can be thought of in terms of the original Lindley Recursion where instead of the arrival time we have the difference of service completion times between consecutive customers.

Proposition 1 The tandem network Lindley recursion is the solution to the following expression:

$$W_{n+1}^{(k)} = \left(W_n^{(k)} + S_n^{(k)} - \left(\left(W_n^{(k-1)} + S_n^{(k-1)} - A_n^{(k-1)} \right)^+ - W_n^{(k-1)} + A_n^{(k-1)} \right) \right)^+ .$$

Proof.

$$\begin{aligned}
W_{n+1}^{(k)} &= \left(W_n^{(k)} + S_n^{(k)} - A_n^{(k)} \right)^+ \\
&= \left(W_n^{(k)} + S_n^{(k)} - \left(D_{n+1}^{(k-1)} - D_n^{(k-1)} \right) \right)^+ \\
&= \left(W_n^{(k)} + S_n^{(k)} - \left(W_{n+1}^{(k-1)} + \sum_{i=1}^{n} A_i^{(k-1)} - W_n^{(k-1)} - \sum_{i=1}^{n-1} A_i^{(k-1)} \right) \right)^+ \\
&= \left(W_n^{(k)} + S_n^{(k)} - \left(W_{n+1}^{(k-1)} - W_n^{(k-1)} + A_n^{(k-1)} \right) \right)^+ \\
&= \left(W_n^{(k)} + S_n^{(k)} - \left(\left(W_n^{(k-1)} + S_n^{(k-1)} - A_n^{(k-1)} \right)^+ - W_n^{(k-1)} + A_n^{(k-1)} \right) \right)^+ .
\end{aligned}$$

□

Through subsequent use of the difference of service completion times we can further incorporate previous waiting time information into the current waiting time formula in the case of a tandem queue with more than 2 queues. This brings us to our first theorem concerning tandem queues. It gives the waiting time of the current customer in the (k+1)st queue as function of the waiting times of the previous customer in the kth and (k+1)st queue. The input of the every node in the system but the first is the output of the previous node. In order to find the arrival rate of one of these nodes it is necessary to find the departure rate from the previous node. In an $M/M/1/\infty$ queue the inter-departure time distribution is equal to the inter-arrival time distribution.This simplifies our analysis quite a bit and also has some interesting consequences.

Theorem 4 For the $M/M/1$ tandem queue where the arrival rate is equal to λ_1 and the service rate vector is equal to (μ_1, μ_2) , then the conditional expectation of the waiting time of customer $n+1$ given the waiting time information of customer n is equal to

$$\begin{aligned}
\mathbb{E}\left[W_{n+1}^{(k+1)}|W_n^{(k)},W_n^{(k+1)}\right] &= \frac{\mu_2 e^{-\lambda_1(W_n^{(k)}+W_n^{(k+1)})}}{\lambda_1(\mu_2+\lambda_1)} - \frac{e^{-\lambda_1 W_n^{(k)}}((1-W_n^{(k+1)}\lambda_1)\mu_2-\lambda_1)}{\lambda_1\mu_2} \\
&- \frac{\lambda_1\mu_2 e^{\mu_1 W_n^{(k)}-(\lambda_1+\mu_1)(W_n^{(k)}+W_n^{(k+1)})}}{(\lambda_1+\mu_1)^2(\lambda_1+\mu_1+\mu_2)} \\
&+ \frac{\lambda_1 e^{\mu_1 W_n^{(k)}-(\lambda_1+\mu_1)W_n^{(k)}}((1-W_n^{(k+1)}(\lambda_1+\mu_1))\mu_2-(\lambda_1+\mu_1))}{(\lambda_1+\mu_1)^2\mu_2} \\
&+ \frac{(1-e^{-\lambda_1 W_n^{(k)}})\mu_2 e^{-\mu_1 W_n^{(k+1)}}}{\mu_1(\mu_1+\mu_2)} \\
&- \frac{(1-e^{-\lambda_1 W_n^{(k)}})((1-W_n^{(k+1)}\mu_1)\mu_2-\mu_1)}{\mu_1\mu_2} - \frac{\mu_1\mu_2 e^{-\lambda_1 W_n^{(k)}-(\lambda_1+\mu_1)W_n^{(k+1)}}}{(\lambda_1+\mu_1)^2(\lambda_1+\mu_1+\mu_2)} \\
&+ \frac{e^{-\lambda_1 W_n^{(k)}}\mu_1((1-(\lambda_1+\mu_1)W_n^{(k+1)})\mu_2-(\lambda_1+\mu_1)}{(\lambda_1+\mu_1)^2\mu_2} + \frac{e^{-\lambda_1 W_n^{(k)}-\mu_1 W_n^{(k+1)}}\mu_2}{\mu_1(\mu_1+\mu_2)} \\
&- \frac{e^{-\lambda_1 W_n^{(k)}}((1-W_n^{(k+1)}\mu_1)\mu_2-\mu_1)}{\mu_1\mu_2}.
\end{aligned}$$

Proof.

$$\begin{aligned}
&\mathbb{E}\left[W_{n+1}^{(k+1)}|W_n^{(k)},W_n^{(k+1)}\right] \\
&= \mathbb{E}\left[\left(W_n^{(k+1)}+S_n^{(k+1)}-\left(\left(W_n^{(k)}+S_n^{(k)}-A_n^{(k)}\right)^+-W_n^{(k)}+A_n^{(k)}\right)\right)^+\middle|W_n^{(k)},W_n^{(k+1)}\right] \\
&= \int_0^\infty\int_0^\infty\int_0^\infty (W_n^{(k+1)}+x-((W_n^{(k)}+u-v)^+-W_n^{(k)}+v))^+ f_{S_n^{(k+1)}}(x) f_{S_n^{(k)}}(u) f_{A_n^{(k)}}(v)\,dx\,du\,dv \\
&= \int_0^\infty\int_0^\infty\int_0^{v-W_n^{(k)}} (W_n^{(k+1)}+x-(W_n^{(k)}+v))^+ f_{S_n^{(k+1)}}(x) f_{S_n^{(k)}}(u) f_{A_n^{(k)}}(v)\,du\,dv\,dx \\
&+ \int_0^\infty\int_0^\infty\int_{v-W_n^{(k)}}^\infty (W_n^{(k+1)}+x-u)^+ f_{S_n^{(k+1)}}(x) f_{S_n^{(k)}}(u) f_{A_n^{(k)}}(v)\,du\,dv\,dx.
\end{aligned}$$

The remaining integrals can be computed via integration by parts and is omitted due to length considerations. □

This result is useful for two reasons. First, it gives us the smallest mean squared error (MSE) among all predictors that incorporate solely the waiting time information. This can be seen with the bias-variance decomposition of the variance of the waiting times of queue $k+1$. Second, it incorporates previous waiting time information for the prediction of the waiting time of the current customer.

In what follows, we are interested in learning a function $f(\cdot)$ such that

$$\hat{W}_{n+1}^k = \hat{f}(W_n^k, A_{n+1}^k, S_n^k) \approx \max\left(W_n^k+S_n^k-A_{n+1}^k, 0\right)$$

for all of the queues in the tandem network. This question is what we pursue in the remainder of the paper by combining simulation and machine learning together.

4 NUMERICAL RESULTS

This section contains the numerical results for the different methods we used tot try to learn the Lindley recursion, We employed linear regression, Gaussian processes, k-nearest neighbors, deep neural networks, and boosted tree regression. We simulated a network of 10 $M/M/1$ queues, with arrival rate λ and service rate μ for each of the 10 servers. The experiment was run with $10,000$ customer waiting times for each queue in the network. The data was split into a training set and test set where the training set was three-quarters of the data set and the test set was the remaining quarter of the data set. We ran the experiment doing a 10 fold cross validation and calculated the average mean squared error (AMSE) for each fold and reported the average of the 10 AMSEs i.e.

$$AMSE = \frac{1}{10}\sum_{j=1}^{10}\left(\frac{1}{n}\sum_{i=1}^{n}\left(W_{ij}-\hat{W}_{ij}\right)^2\right).$$

The results are given in scientific notation because almost each AMSE is relatively close to 0 so what really matters is by how many orders of magnitude is AMSE close to 0.

4.1 Linear Regression

The first machine learning method we tried was linear regression. Linear regression is a natural choice for learning the the tandem Lindley recursion because it is function of a linear term. We observed that the error of the linear regression decreases as the server utilization increases. This occurs for the fact that as few customers have a 0 wait time the more points fall on the $f(x,y,z) = x+y-z$ hyper-surface. The hyper-surface $f(x,y,z)$ is a plane and linear regression can learn to predict the output for points on this surface exactly. Along with the error the intercept also increases. The intercept can be thought of as a correction factor. The more points that fall in the hyper-surface $f(x,y,z)$ the less there is a need to correct the sum $W_{n-1}+S_{n-1}-A_n$ for a random customer because most points are that sum exactly. While if fewer points are on the hyper-surface $f(x,y,z)$ the greater the need to add something extra to correct the fact that the waiting time is not exactly $W_{n-1}+S_{n-1}-A_n$. As the server utilization increases the coefficients move farther away from 1.

Table 1: Results for the Linear Regression

μ	1	1	1	1	10	10	10	10
λ	.5	.75	.9	.99	5	7.5	9	9.9
Q1 AMSE	4.69e-01	3.64e-01	2.10e-01	4.27e-02	5.24e-03	3.28e-03	1.41e-05	7.64e-05
Q2 AMSE	4.59e-01	3.66e-01	2.15e-01	5.21e-02	5.32e-03	3.46e-03	1.69e-04	9.35e-05
Q3 AMSE	4.26e-01	3.66e-01	2.01e-01	8.01e-02	5.05e-03	3.77e-03	4.01e-04	3.63e-04
Q4 AMSE	4.53e-01	3.52e-01	2.12e-01	9.24e-02	4.93e-03	3.58e-03	4.91e-04	3.10e-04
Q5 AMSE	4.62e-01	3.66e-01	2.39e-01	4.66e-02	5.19e-03	3.87e-03	1.05e-03	6.31e-04
Q6 AMSE	5.13e-01	3.82e-01	2.11e-01	9.69e-02	4.66e-03	3.97e-03	5.21e-04	7.52e-04
Q7 AMSE	4.72e-01	3.54e-01	2.03e-01	1.00e-01	4.84e-03	3.76e-03	1.22e-03	6.17e-04
Q8 AMSE	4.95e-01	3.65e-01	2.40e-01	1.08e-01	4.60e-03	3.90e-03	6.34e-04	1.29e-03
Q9 AMSE	4.70e-01	3.98e-01	2.07e-01	1.20e-01	4.66e-03	3.46e-03	1.00e-03	7.58e-04
Q10 AMSE	5.11e-01	3.96e-01	2.04e-01	1.71e-01	5.01e-03	3.52e-03	6.92e-04	1.29e-03

4.2 Gaussian Processes

We also used Gaussian Process regression with the squared exponential kernel to learn the Lindley Recursion. As observed from Tables 3 and 4 Gaussian processes performed the best across values of λ and μ. Unlike the other methods there is not a stark contrast in AMSE between the different arrival rates. This is because

Table 2: Results for Linear Regression 2

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
Q1 AMSE	4.79e-05	3.68e-05	1.76e-05	2.81e-06	4.88e-07	3.69e-07	1.72e-07	3.56e-08
Q2 AMSE	4.88e-05	3.67e-05	1.76e-05	4.40e-06	5.00e-07	3.65e-07	1.82e-07	5.15e-08
Q3 AMSE	4.70e-05	3.73e-05	1.62e-05	4.00e-06	4.75e-07	3.55e-07	1.77e-07	6.20e-08
Q4 AMSE	4.72e-05	3.60e-05	1.77e-05	5.59e-06	4.82e-07	3.77e-07	1.83e-07	7.05e-08
Q5 AMSE	4.81e-05	3.53e-05	1.97e-05	7.03e-06	4.87e-07	3.76e-07	1.83e-07	7.38e-08
Q6 AMSE	4.99e-05	3.65e-05	1.76e-05	6.05e-06	4.87e-07	3.64e-07	1.84e-07	9.21e-08
Q7 AMSE	4.86e-05	3.65e-05	1.83e-05	7.60e-06	5.00e-07	3.64e-07	1.85e-07	8.77e-08
Q8 AMSE	4.87e-05	3.78e-05	1.85e-05	8.65e-06	4.95e-07	3.66e-07	1.81e-07	9.10e-08
Q9 AMSE	4.91e-05	3.70e-05	1.99e-05	9.31e-06	4.91e-07	3.63e-07	1.92e-07	9.78e-08
Q10 AMSE	5.05e-05	3.68e-05	1.83e-05	8.56e-06	4.90e-07	3.78e-07	1.91e-07	1.08e-07

Gaussian processes do not depend on the shape of the test surface for prediction. Predictions are made with local information as weighted by the squared exponential kernel.

Table 3: Results for Gaussian Processes.

μ	1	1	1	1	10	10	10	10
λ	.5	.75	.9	.99	5	7.5	9	9.9
Q1 AMSE	1.81e-02	7.08e-03	6.78e-03	1.38e-03	4.62e-04	4.71e-05	1.41e-05	5.19e-05
Q2 AMSE	1.67e-02	2.46e-03	1.43e-03	1.76e-03	3.16e-04	4.40e-05	2.70e-05	9.35e-05
Q3 AMSE	6.66e-03	2.55e-03	9.34e-03	1.65e-03	2.28e-04	1.01e-05	7.98e-05	2.12e-05
Q4 AMSE	1.63e-02	7.91e-03	1.99e-03	2.78e-03	3.39e-04	5.40e-05	2.81e-05	2.34e-05
Q5 AMSE	1.17e-02	2.39e-03	6.49e-03	1.56e-03	3.50e-04	3.85e-05	1.30e-05	3.09e-05
Q6 AMSE	7.37e-03	4.82e-03	1.28e-03	1.70e-03	2.46e-04	4.71e-05	1.29e-05	6.47e-05
Q7 AMSE	2.02e-02	3.65e-03	1.53e-03	2.09e-03	2.57e-04	2.18e-04	2.43e-05	4.48e-05
Q8 AMSE	1.92e-02	7.16e-03	1.56e-03	1.65e-03	1.60e-04	1.10e-04	2.73e-05	1.99e-05
Q9 AMSE	1.77e-02	1.03e-03	1.46e-03	1.83e-03	1.64e-04	4.25e-04	4.54e-05	3.29e-05
Q10 AMSE	3.46e-02	5.91e-03	2.09e-03	1.03e-02	1.81e-04	1.12e-04	5.84e-05	5.41e-05

Table 4: Results for Gaussian Processes 2 .

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
Q1 AMSE	5.46e-07	3.82e-07	4.14e-07	4.03e-07	1.51e-08	1.47e-08	1.01e-08	2.97e-08
Q2 AMSE	5.70e-07	4.72e-07	2.38e-07	3.36e-07	1.46e-08	1.36e-08	1.07e-08	3.34e-08
Q3 AMSE	5.00e-07	6.11e-07	1.64e-07	2.53e-07	1.37e-08	1.42e-08	1.09e-08	4.00e-08
Q4 AMSE	6.79e-07	4.48e-07	2.41e-07	3.23e-07	1.50e-08	1.20e-08	1.06e-08	1.76e-08
Q5 AMSE	5.23e-07	3.98e-07	3.95e-07	3.68e-07	1.42e-08	1.30e-08	1.16e-08	1.74e-08
Q6 AMSE	7.30e-07	3.26e-07	1.90e-07	3.38e-07	1.51e-08	1.29e-08	1.04e-08	2.20e-08
Q7 AMSE	5.69e-07	3.58e-07	2.24e-07	4.63e-07	1.64e-08	1.15e-08	1.11e-08	1.92e-08
Q8 AMSE	5.32e-07	3.48e-07	3.26e-07	3.21e-07	1.45e-08	1.20e-08	1.09e-08	1.31e-08
Q9 AMSE	4.32e-07	5.71e-07	2.18e-07	2.81e-07	1.45e-08	1.22e-08	1.24e-08	1.35e-08
Q10 AMSE	9.16e-07	8.44e-07	3.89e-07	3.56e-07	1.42e-08	1.24e-08	1.12e-08	1.19e-08

4.3 K-Nearest Neighbors

We also implemented the K-nearest neighbors (K-NN) algorithm on the tandem network with $k = 5$. Here we see that unlike linear regression K-NN performs worst when arrival rate approaches the service rate, in other words as more waiting times are 0 the better K-NN performs. This is due to the fact that given a test point with a 0 label it is easier for K-NN to accurately predict that point the greater the proportion of 0 waiting times. As the arrival rate moves away from the service rate toward 0, the larger the proportion of customers with 0 waiting time.

Table 5: Results for the K-Nearest Neighbors Regression.

μ	1	1	1	1	10	10	10	10
λ	.5	.75	.9	.99	5	7.5	9	9.9
Q1 AMSE	2.10e-02	3.02e-02	4.98e-02	1.10e-01	2.86e-04	4.54e-04	9.50e-03	1.72e-03
Q2 AMSE	1.78e-02	2.77e-02	5.09e-02	1.15e-01	2.30e-04	4.05e-04	1.53e-03	2.35e-03
Q3 AMSE	1.30e-02	2.66e-02	6.50e-02	9.64e-02	1.95e-04	2.59e-04	2.24e-03	1.24e-03
Q4 AMSE	1.63e-02	3.44e-02	7.68e-02	1.48e-01	2.51e-04	5.22e-04	1.41e-03	1.40e-03
Q5 AMSE	2.13e-02	3.38e-02	5.98e-02	1.02e-01	4.64e-04	3.92e-04	7.03e-04	1.58e-03
Q6 AMSE	1.95e-02	3.83e-02	3.82e-02	9.61e-02	1.34e-04	3.53e-04	1.13e-03	2.00e-03
Q7 AMSE	2.64e-02	3.21e-02	7.65e-02	8.80e-02	2.32e-04	4.25e-04	1.27e-03	1.65e-03
Q8 AMSE	3.25e-02	3.42e-02	5.57e-02	9.06e-02	1.70e-04	3.85e-04	1.33e-03	9.20e-04
Q9 AMSE	2.42e-02	2.84e-02	5.92e-02	9.44e-02	2.28e-04	4.80e-04	1.47e-03	1.66e-03
Q10 AMSE	3.10e-02	3.57e-02	6.77e-02	1.13e-01	2.16e-04	2.97e-04	2.43e-03	9.55e-04

Table 6: Results for the K-Nearest Neighbors Regression 2 .

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
Q1 AMSE	2.87e-06	3.99e-06	7.05e-06	1.45e-05	2.25e-08	3.56e-08	6.90e-08	1.99e-07
Q2 AMSE	2.45e-06	3.72e-06	6.23e-06	1.32e-05	2.80e-08	3.81e-08	6.93e-08	1.42e-07
Q3 AMSE	2.17e-06	3.59e-06	6.05e-06	1.37e-05	2.46e-08	3.76e-08	8.55e-08	1.42e-07
Q4 AMSE	2.57e-06	3.92e-06	6.62e-06	1.32e-05	2.27e-08	3.91e-08	6.69e-08	1.16e-07
Q5 AMSE	2.37e-06	3.63e-06	6.62e-06	1.41e-05	2.34e-08	3.86e-08	7.30e-08	1.24e-07
Q6 AMSE	2.72e-06	3.77e-06	7.21e-06	1.39e-05	2.79e-08	3.52e-08	6.54e-08	1.13e-07
Q7 AMSE	2.63e-06	4.21e-06	7.05e-06	1.34e-05	2.76e-08	3.73e-08	5.76e-08	1.19e-07
Q8 AMSE	2.52e-06	3.83e-06	6.67e-06	1.11e-05	2.55e-08	4.18e-08	6.14e-08	1.06e-07
Q9 AMSE	2.13e-06	3.92e-06	6.74e-06	1.02e-05	2.49e-08	3.91e-08	6.53e-08	1.10e-07
Q10 AMSE	3.14e-06	3.35e-06	7.31e-06	1.28e-05	2.15e-08	3.78e-08	7.03e-08	1.03e-07

4.4 Deep Neural Networks

The neural network implementation we used was Google's Tensorflow. As for the model itself, it had two hidden layers each with 10 nodes and ReLU activation functions. The layers were densely connected. We used the RMSprop optimization function. We ran the fitting of the model for 20 epochs.

Like with k-nearest neighbors we see that deep neural networks do better when more points fall away from the $f(x,y,z) = x+y-z$ hyper-surface. This is because it is easier for a deep neural network with ReLU activation functions to learn a max-plus function than it is for it to learn a linear function. This follows from the nature of ReLU activation function, it is itself a max-plus function. With two dense layers there is more than enough capacity for deep neural network to learn the linear mapping $f(x,y,z) = x+y-z$, in fact it can be done with one hidden layer. As the arrival rate approaches the service rate more points fall on the $f(x,y,z) = x + y - z$ hyper-surface, making the test surface more linear.

Table 7: Results for the Deep Neural Networks Regression 1 .

μ	1	1	1	1	10	10	10	10
λ	.5	.75	.9	.99	5	7.5	9	9.9
Q1 AMSE	3.47e+02	2.83e+01	1.04e+01	8.28e-02	2.37e-05	5.54e-05	9.39e-02	2.59e-03
Q2 AMSE	6.70e-01	6.17e-01	1.82e-01	7.81e-02	1.59e-05	8.10e-05	6.00e-03	9.11e+
Q3 AMSE	3.94e-01	1.25e-01	7.94e-02	4.49e-02	5.66e-05	4.32e-05	1.79e-04	1.20e-03
Q4 AMSE	3.59e-01	5.62e-02	5.78e-02	1.67e-02	2.75e-05	2.49e-05	7.51e-04	3.88e-04
Q5 AMSE	1.35e-01	1.58e-01	1.11e-02	7.84e-02	2.17e-05	3.87e-05	3.33e-04	5.85e-04
Q6 AMSE	2.80e-01	1.17e-01	2.73e-02	8.91e-02	1.76e-05	2.84e-05	7.24e-04	2.44e-04
Q7 AMSE	2.50e-01	5.21e-02	1.01e-01	6.62e-02	1.15e-05	5.04e-05	1.34e-03	7.05e-02
Q8 AMSE	5.69e-03	5.30e-02	5.12e-02	6.00e-02	2.24e-05	4.41e-05	3.75e-04	2.84e-04
Q9 AMSE	1.02e-01	1.05e-02	1.61e-02	9.16e-03	1.70e-05	2.72e-05	4.19e-04	2.69e-04
Q10 AMSE	1.71e+02	4.47e-02	1.39e-02	1.60e-02	3.94e-05	4.72e-05	8.75e-04	1.21e-04

Table 8: Results for the Deep Neural Networks Regression 2 .

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
Q1 AMSE	4.88e-06	8.69e-06	1.28e-05	2.95e-05	9.89e-07	5.02e-07	2.09e-06	6.30e-06
Q2 AMSE	4.29e-06	6.19e-06	1.46e-05	5.38e-05	1.62e-06	1.33e-06	2.10e-06	4.99e-06
Q3 AMSE	3.02e-06	8.69e-06	1.09e-05	2.58e-05	1.39e-06	7.73e-07	4.14e-06	6.05e-06
Q4 AMSE	2.61e-06	9.63e-06	1.54e-05	2.75e-05	1.30e-06	1.38e-06	2.34e-06	5.99e-06
Q5 AMSE	4.87e-06	1.02e-05	2.42e-05	1.87e-05	1.02e-06	1.06e-06	3.18e-06	3.63e-06
Q6 AMSE	3.69e-06	7.49e-06	1.54e-05	2.68e-05	1.06e-06	1.67e-06	2.30e-06	4.40e-06
Q7 AMSE	4.96e-06	7.81e-06	1.31e-05	1.20e-05	1.17e-06	2.00e-06	2.19e-06	4.15e-06
Q8 AMSE	5.87e-06	8.59e-06	1.26e-05	2.85e-05	1.10e-06	1.78e-06	2.59e-06	3.09e-06
Q9 AMSE	3.89e-06	9.92e-06	1.33e-05	2.82e-05	1.78e-06	8.24e-07	3.43e-06	4.03e-06
Q10 AMSE	3.45e-06	9.73e-06	1.68e-05	2.20e-05	1.39e-06	1.45e-06	1.66e-06	2.82e-06

4.5 Boosted Tree Regression

To implement Boosted Tree Regression, we used 500 weak learners with a maximum depth of 4 depth of levels. Boosted tree regression did fairly well and outperformed deep neural networks in some occasions. However, it was not as good as Gaussian Processes.

4.6 Summary of Numerical Results

In this section, we summarize our numerical results. We are able to learn the Lindley recursion quite well under every parameter regime using of Linear Regression, Gaussian Processes, K-Nearest Neighbors, Deep Neural Networks, and Boosted Tree Regression. As observed from the above tables, Gaussian processes performed the best as it achieved the lowest AMSE across all parameters regimes among all of the other machine learning models. Boosted tree regression and deep neural networks performed the worst. We also observed that these methods were worst is because of a small amount of data, which performed poorly. This poorly predicted data points tended to skew the performance metrics to seem very poor.

Table 9: Results for the Boosted Tree Regression 1.

μ	1	1	1	1	10	10	10	10
λ	.5	.75	.9	.99	5	7.5	9	9.9
Q1 AMSE	3.76e+02	4.42e+01	7.10e+	3.61e-01	4.55e-04	6.46e-04	7.24e-02	3.82e-03
Q2 AMSE	2.02e+	8.99e-01	5.98e-01	3.81e-01	4.13e-04	6.87e-04	6.86e-03	4.18e-03
Q3 AMSE	1.39e+	6.17e-01	4.03e-01	3.28e-01	4.20e-04	6.74e-04	4.52e-03	3.14e-03
Q4 AMSE	1.25e+	5.55e-01	3.08e-01	2.72e-01	4.11e-04	6.75e-04	3.64e-03	2.35e-03
Q5 AMSE	1.26e+	5.63e-01	3.36e-01	2.61e-01	4.43e-04	6.96e-04	2.88e-03	2.56e-03
Q6 AMSE	1.32e+	5.85e-01	3.49e-01	3.37e-01	4.52e-04	7.28e-04	3.22e-03	2.42e-03
Q7 AMSE	1.29e+	5.75e-01	3.53e-01	2.67e-01	4.05e-04	7.20e-04	3.33e-03	2.32e-03
Q8 AMSE	9.67e-01	4.29e-01	3.10e-01	2.04e-01	3.98e-04	6.66e-04	3.18e-03	2.50e-03
Q9 AMSE	9.01e-01	4.01e-01	3.34e-01	2.31e-01	4.37e-04	6.69e-04	2.93e-03	2.48e-03
Q10 AMSE	1.10e+	4.89e-01	2.59e-01	2.72e-01	4.83e-04	7.03e-04	3.05e-03	2.15e-03

Table 10: Results for the Boosted Tree Regression 2 .

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
Q1 AMSE	4.08e-06	6.75e-06	1.29e-05	3.45e-05	3.82e-08	7.10e-08	1.23e-07	4.76e-07
Q2 AMSE	4.40e-06	7.00e-06	1.16e-05	3.01e-05	4.48e-08	6.71e-08	1.31e-07	3.56e-07
Q3 AMSE	4.33e-06	7.42e-06	1.14e-05	2.73e-05	4.31e-08	6.90e-08	1.26e-07	2.83e-07
Q4 AMSE	4.10e-06	6.59e-06	1.22e-05	2.69e-05	4.04e-08	7.07e-08	1.21e-07	2.20e-07
Q5 AMSE	4.44e-06	7.02e-06	1.32e-05	2.86e-05	4.46e-08	7.22e-08	1.28e-07	2.42e-07
Q6 AMSE	4.09e-06	6.26e-06	1.40e-05	3.04e-05	4.55e-08	6.93e-08	1.12e-07	2.46e-07
Q7 AMSE	4.30e-06	6.48e-06	1.15e-05	2.96e-05	3.86e-08	7.24e-08	1.11e-07	2.51e-07
Q8 AMSE	5.17e-06	7.13e-06	1.23e-05	2.37e-05	4.29e-08	6.97e-08	1.11e-07	2.20e-07
Q9 AMSE	3.87e-06	7.46e-06	1.19e-05	2.11e-05	4.38e-08	7.34e-08	1.28e-07	2.37e-07
Q10 AMSE	4.44e-06	6.97e-06	1.26e-05	2.63e-05	4.11e-08	7.43e-08	1.11e-07	2.03e-07

In addition to which methods performed the best, we also observed some important trends in the numerical experiments. First, we observed that in of the all machine learning methods we implemented, the AMSE decreases by a few orders of magnitude as the we increase the order of μ and λ i.e. we scale up the parameters. Second, we observe that as λ approaches μ (heavy traffic) there is either an increase or decrease in AMSE, depending on how the machine learning model is trained and makes predictions. So not all machine learning methods are better in heavy traffic. Linear regression and Gaussian processes do particularly better in the heavy traffic regime, while K-NN, deep neural networks, and boosted tree regression do the worst.

5 CONCLUSION

In this paper, we show the power of current machine learning methods to approximate the Lindley recursion for tandem queues. We show in this work that the best method for approximating the Lindley recursion is Gaussian processes. This might be because deep neural networks converge to a Gaussian process as one lets the number of layers tend to infinity. We show that deep neural network methods work the second best, especially with the ReLU activation function. The ReLU activation function is especially relevant as it is a max-plus function just like the Lindley recursion, thus is is not a surprise that it works well in approximating the Lindley recursion. Despite our analysis in showing that the machine learning methods work extremely well at approximating the Lindley recursion, there remain many interesting questions that need to be explored in this domain.

REFERENCES

Armony, M., S. Israelit, A. Mandelbaum, Y. N. Marmor, Y. Tseytlin, and G. B. Yom-Tov. 2015. “On Patient Flow in Hospitals: A Data-Based Queueing-Science Perspective”. *Stochastic Systems* 5(1):146–194.

Bodík, P., R. Griffith, C. A. Sutton, A. Fox, M. I. Jordan, and D. A. Patterson. 2009. “Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters.”. *HotCloud* 9:12–12.

Brown, L., N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, and L. Zhao. 2005. “Statistical Analysis of a Telephone Call Center: A Queueing-Science Perspective”. *Journal of the American Statistical Association* 100(469):36–50.

Dunlay Jr, W. J., and C.-H. Park. 1978. “Tandem Queue Algorithm for Airport User Flows”. *Transportation Engineering Journal of ASCE* 104(2):131–149.

Fu, Q., and W. Whitt. 2015. “Analyzing the pace of play in golf”. *Journal of Sports Analytics* 1(1):43–64.

Koole, G., and A. Mandelbaum. 2002. “Queueing Models of Call Centers: An Introduction”. *Annals of Operations Research* 113(1-4):41–59.

Kyritsis, A. I., and M. Deriaz. 2019. “A Machine Learning Approach to Waiting Time Prediction in Queueing Scenarios”. In *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*, 17–21. IEEE.

Lindley, D. V. 1952. “The Theory of Queues with a Single Server”. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Volume 48, 277–289. Cambridge University Press.

Nii, S., T. Okudal, and T. Wakita. 2020. “A Performance Evaluation of Queueing Systems by Machine Learning”. In *2020 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan)*, 1–2. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Nirenberg, S., A. Daw, and J. Pender. 2018. “The Impact of Queue Length Rounding and Delayed App Information on Disney World Queues”. In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 3849–3860. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Novitzky, S., J. Pender, R. H. Rand, and E. Wesson. 2019. “Nonlinear Dynamics in Queueing Theory: Determining the Size of Oscillations in Queues with Delay”. *SIAM Journal on Applied Dynamical Systems* 18(1):279–311.

Palomo, S., and J. Pender. 2020. “Learning Lindley’s Recursion”. In *Proceedings of the 2020 Winter Simulation Conference*, edited by K. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, 644–655. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Pender, J., and T. Phung-Duc. 2016. “A Law of Large Numbers for M/M/c/Delayoff-Setup Queues with Nonstationary Arrivals”. In *International conference on analytical and stochastic modeling techniques and applications*, 253–268. Springer.

Prabhu, N. 1974. “Wiener-Hopf Techniques in Queueing Theory”. In *Mathematical methods in queueing theory*, 81–90. Springer.

Shah, A., A. Wikum, and J. Pender. 2019. “Using Simulation to Study the Last to Enter Service Delay Announcement in Multiserver Queues with Abandonment”. In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 2595–2605. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sutton, C., and M. I. Jordan. 2010. “Inference and Learning in Networks of Queues”. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 796–803.

Sutton, C., and M. I. Jordan. 2011. “Bayesian Inference for Queueing Networks and Modeling of Internet Services”. *The Annals of Applied Statistics*:254–282.

Wang, W., G. Casale, and C. Sutton. 2016. “A Bayesian Approach to Parameter Inference in Queueing Networks”. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 27(1):1–26.

Whitt, W. 2015. “The maximum throughput on a golf course”. *Production and Operations Management* 24(5):685–703.

AUTHOR BIOGRAPHIES

SERGIO PALOMO is a PhD student in Systems Engineering at Cornell University. He holds a Masters in Mathematics from City College of New York and a Bachelors in Applied Mathematics from Stony Brook University. His research interests are in queueing theory, machine learning and stochastic simulation. His e-mail address is sdp85@cornell.edu.

JAMOL PENDER is an associate professor in ORIE at Cornell University. He earned his PhD in ORFE at Princeton University. His research interests include queueing theory, stochastic simulation, dynamical systems and applied probability. His e-mail address is jjp274@cornell.edu. His website is https://blogs.cornell.edu/jamolpender/.