Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

LEARNING LINDLEY'S 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

Lindley's recursion is one of the most important formula's in queueing theory and applied probability. In this paper, we leverage stochastic simulation and current machine learning methods to learn the Lindley recursion directly from waiting time data of the G/G/1 queue. To this end, we use methods such as Gaussian Processes, k-Nearest Neighbors and Deep neural networks to learn the Lindley recursion. We also analyze specific parameter regimes for the M/M/1 to understand where learning the Lindley recursion may be easy or hard. Finally, we compare the machine learning methods to see how well we can predict the Lindley recursion multiple steps into the future with missing data.

1 INTRODUCTION

Lindley's recursion is one of the most important formula's in queueing theory and applied probability. Lindley's recursion is an explicit recursive equation that describes the recursive relationship between consecutive waiting times in a G/G/1 queue, which is a single server queue that can have general interarrival times and general service times. One should note that the G/G/1 does not require independence of the arrival or service processes and even in between arrival or service times. The Lindley recursion can also be modified to model the consecutive response times or departure times in a G/G/1 queue as well. The success of the Lindley recursion lies in its simplicity and it has been exploited to derive various properties of performance measures for the G/G/1 queue. One such example is the derivation of the G/G/1 waiting time distribution using the Lindley recursion Buzacott and Shanthikumar (1993). One can also prove that the waiting time is convex with respect to the arrival and service parameters, see for example Shaked and Shanthikumar (1988). Finally, the Lindley recursion has been exploited in optimization and performance evaluation contexts in Glasserman and Ho (1991), Fu and Hu (2012), Kin and Chan (2010), Baccelli and Brémaud (2013).

What makes the Lindley recursion attractive beyond its power to help prove structural insights about queues is that is only uses simple arithmetic operations to relate the successive waiting times. In fact, it only uses the addition and maximum operations, which provides much simplicity and scalability. Moreover, the recursion is also exact and gives the sample path waiting times of any customer. Despite the simplicity of the Lindley recursion, 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 construct actual stochastic models. There are some notable exceptions such as Sutton and Jordan (2011), Bodík et al. (2009), Sutton and Jordan (2010), Wang et al. (2016), Kim et al. (2015), Bartel et al. (2020), Armony et al. (2015). With this in mind, our goal is to understand the effectiveness of current machine learning methods to recover Lindley's recursion from waiting time data of the M/M/1 queue. To this end, we combine stochastic simulation with machine learning to learn the Lindley recursion function.

Despite, the simplicity of the Lindley recursion in the single server setting the extension to the multi-server setting is quite difficult and generally scales poorly with the number of servers. This scale issue

can be overcome if one can find a function that maps the previous waiting times to the next waiting time, see for example Kin and Chan (2010). Finding simple ways to compute these waiting times, has important implications for various applications such as call centers, cloud computing centers, healthcare networks, and even amusement parks, see for example Armony et al. (2015), Shah et al. (2019), Pender and Phung-Duc (2016), Delimitrou and Kozyrakis (2013), Nirenberg et al. (2018), Brown et al. (2005), Koole and Mandelbaum (2002). Having good solutions to this problem is of vital importance to the simulation community. Good solutions to this problem would dramatically reduce the computational time to compute waiting times for queueing systems with a large number of customers. 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 Lindley recursion. We will restrict our analysis to the single server case as it will serve as a base case for more complicated models. Moreover, if machine learning methods cannot perform well on this simple model, then it implies that the study of more complicated models is hopeless.

In this paper, our goal is to learn the Lindley recursion for the M/M/1 queue, which is a single server queue where the inter-arrival and service times follow independent exponential distributions with rates λ and μ respectively. Other than learning the Lindley recursion from simulated waiting time data, we also aim to understand under what regimes is it easy or hard to learn the Lindley recursion. For example, do current machine learning methods work well when the queue is in light traffic or heavy traffic? We also compare various machine learning methods for learning the Lindley recursion and understand their relative performance on in-sample and out of sample waiting time data. To this end, we list the contributions of our work below.

1.1 Contributions of Our Work

By using stochastic simulation for the M/M/1 queue and leveraging machine learning methods, we provide new insights to the following questions:

- What machine learning methods work well at learning the Lindley recursion for the G/G/1 queue?
- Is learning the Lindley recursion easier in the heavy traffic setting?
- How difficult is learning the Lindley recursion for increasing server utilization?
- How well can we predict Lindley's recursion k steps into the future?

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 present our simulation and machine learning results for various methods. We also explain how various parameters of the queueing model affect the performance of learning the Lindley recursion. Finally, a conclusion is given in Section 4.

2 LINDLEY'S RECURSION

Consider the M/M/1 queueing system model where the customer inter-arrival distribution is general, the service distribution is general, and there is a single server. For a complete description of this queueing system, see Shortle et al. (2018). For the n^{th} customer to arrive to the queue, we define W_n to be the waiting time for that 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 Kendall (1951), Lindley (1952) is:

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

Previous analysis such as Konheim (1975) and Prabhu (1974) studies the steady state waiting time of the G/G/1 queue and shows that the steady state waiting time satisfies the following integral equation, which is known as Lindley's integral equation

$$F(x) = \int_{0^-}^{\infty} K(x-y)F(\mathrm{dy}) \quad x \geq 0$$

where $K(x)$ is the distribution function of the random variable denoting the difference between the service time of the n^{th} customer and the inter-arrival time of the n^{th} and $n+1^{th}$ customer. In fact the Wiener–Hopf method can be used to solve this integral equation in closed form Prabhu (1974). In addition 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. In the Markovian single server queue 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 mean waiting time of customer $n+1$ is equal to

$$\mathbb{E}[W_{n+1}|W_n] \quad = \quad W_n + \frac{\lambda-\mu}{\lambda\mu} + \frac{\mu e^{-\lambda W_n}}{\lambda+\mu}.$$

Proof.

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

Table 1: Theoretical Expected Waiting Times.

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
$\mathbb{E}[W_{n+1}\|W_n]$	0.284	0.101	0.0376	0.02	0.299	0.105	0.0434	0.0161

The result in Theorem 1 provides the solution to the conditional expected waiting time of the $(n+1)^{th}$ customer given the current waiting time of the n^{th} customer. One can view this conditional expectation as the constant that minimizes the squared error when predicting the waiting time of the next customer with information of the current customer. In row 3 of Table 1, we compute the mean squared error of the conditional mean waiting time across 1000 customers and we report the average of ten samples. However, as seen in Table 1, the conditional mean of the waiting time is not really a good way to understand the value of the next customer's waiting time. Fortunately, we know the mapping to get the next customer's waiting time explicitly through Lindley's recursion. However, this made us ask the question if one can recover the Lindley equation directly from data without knowing the form of the max plus mapping. That

is, if I am given the previous waiting time W_n, the inter-arrival time A_{n+1} and the service time S_n, then I want to find a function $\hat{f}$ such that

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

This question is what we pursue in the remainder of the paper by leveraging simulation and machine learning together.

3 MACHINE LEARNING METHODS FOR LINDLEY'S RECURSION

In this section of the paper, we describe a number of machine learning techniques and show how to use them in approximating the Lindley recursion function.

3.1 How the Data is Collected and Analyzed

Before we describe the methods that we use to learn the Lindley's recursion, we will describe how we conduct our numerical experiments for the remainder of the paper. The experiments that follow were all trained on 1000 simulated waiting time observations and are implemented or tested on 10 independent sets for all methods and averaged. All numbers that are reported are based on the sample mean squared error for the 1000 samples i.e.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(W_i - \hat{W}_i\right)^2.$$

First, we initialized each sample path with $W_0 = 0$. Then in order to compute W_{i+1} from the sample W_i, we sample an exponential random variable with rate λ and an exponential random variable with rate μ. Then we applied the Lindley recursion mapping to those random variables in order to compute the waiting time of the subsequent customer.

3.2 Linear and Quadratic Regression

In this section, we aim to understand how linear and quadratic regression will perform in learning the Lindley recursion function. In the linear regression framework we need to find the coefficients $\beta_0, \beta_W, \beta_A, \beta_S$ in order to minimize the following loss function

$$\min_{\beta_0,\beta_W,\beta_A,\beta_S} \sum_{i=0}^{n-1}\left(W_{i+1} - \beta_0 - \beta_W W_i - \beta_A A_i - \beta_S S_i\right)^2.$$

In the quadratic setting, we one will minimize a more complicated function, that will take into account the squared terms and the cross terms of the independent variables (W_n, A_n, S_n).

Table 2: Linear and Quadratic Regression One Step Prediction (Light Traffic) M/M/1 Queue. (In Sample)

μ	1	1	1	1	10	10	10	10
λ	0.5	0.75	0.9	.99	5	7.5	9	9.9
Linear	0.584	0.318	0.232	0.0367	0.00513	0.00365	0.00132	2.29e-4
Quadratic	0.134	0.118	0.128	0.0318	0.00123	0.00164	9.3e-4	2.01e-4

Table 3: Linear and Quadratic Regression One Step Prediction (Heavy Traffic) M/M/1 Queue. (In Sample)

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
Linear	4.86e-5	3.69e-5	1.78e-5	1.26e-5	4.9e-7	3.34e-7	1.8e-7	1.03e-7
Quadratic	1.02e-5	1.51e-5	8.41e-6	6.71e-6	1.12e-7	9.98e-8	9.66e-8	6.61e-8

Table 4: Linear and Quadratic Regression One Step Prediction (Light Traffic) M/M/1 Queue. (Out of Sample)

μ	1	1	1	1	10	10	10	10
λ	0.5	0.75	0.9	.99	5	7.5	9	9.9
Linear	0.482	0.301	0.234	0.0163	0.00715	0.00414	0.00161	2.12e-4
Quadratic	0.124	0.117	0.168	0.0135	0.0056	0.00174	0.00113	3.74e-4

Table 5: Linear and Quadratic Regression One Step Prediction (Heavy Traffic) M/M/1 Queue. (Out of Sample)

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
Linear	5.03e-5	3.34e-5	2.0e-5	1.35e-5	4.71e-7	3.62e-7	2.35e-7	4.1e-8
Quadratic	1.59e-5	1.35e-5	1.21e-5	1.26e-5	1.1e-7	1.41e-7	1.18e-7	5.84e-8

3.2.1 Summary of Tables

In Tables 2 - 5, we demonstrate the performance of linear and quadratic regression. We find that the quadratic regression performs better than the linear regression in the light traffic and heavy traffic settings. We also observe that the performance of both the linear and quadratic regressions improve as the queueing system moves into the heavy traffic regime. This is most likely because in the heavy traffic regime, the waiting time spends less time at the origin than in the light traffic setting. Intuitively since the Lindley Recursion is the max of two linear functions the learned linear and quadratic functions tend to approximate the Lindley function better when the system is in heavy traffic. Finally, we also observe that the in-sample performance is better, but not much better than the out of sample performance. This means that the functional approximation can extend well to out of sample waiting time data sets.

3.3 K-Nearest Neighbors

The k-nearest neighbors (KNN) algorithm is a simple and easy to implement supervised machine learning algorithm that can be used to solve both classification and regression problems. We will exploit KNN for regression in this paper. The algorithm can be summed up by the phrase "Birds of a feather flock together". This is because KNN exploits the idea of similarity or distance. In the context of the Lindley recursion, the KNN approach is to find similar data values such that the distance from (W_n, A_n, S_n) is minimized. Then, one can use these values as approximate values of our estimate of W_{n+1}. In some sense, KNN is finding the closest data points that are similar to the one we want to estimate and using those close data points to make that estimation. Below we use the k-nearest neighbors method for learning the Lindley recursion.

Table 6: K-Nearest Neighbor One Step Prediction (Light Traffic) M/M/1 Queue.

μ	1	1	1	1	10	10	10	10
λ	0.5	0.75	0.9	.99	5	7.5	9	9.9
k = 1	0.111	0.154	0.404	0.938	0.0125	0.00187	0.00336	0.27
k = 5	0.0805	0.109	0.233	0.485	8.95e-4	0.0014	0.00182	0.00192
k = 10	0.117	0.146	0.297	0.688	0.00119	0.00183	0.00252	0.00249
k = 25	0.226	0.242	0.545	1.17	0.00219	0.0039	0.00497	0.00494

Table 7: K-Nearest Neighbor One Step Prediction (Heavy Traffic) M/M/1 Queue.

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
k = 1	1.25e-5	1.83e-5	4.55e-5	2.93e-5	9.09e-8	1.75e-7	2.74e-7	5.91e-7
k = 5	7.07e-6	1.48e-5	1.53e-5	2.09e-5	5.82e-8	8.46e-8	1.28e-7	2.43e-7
k = 10	1.02e-5	2.47e-5	2.08e-5	2.57e-5	7.85e-8	1.05e-7	1.85e-7	3.3e-7
k = 25	1.84e-5	4.28e-5	3.84e-5	4.84e-5	1.57e-7	1.82e-7	3.62e-7	6.49e-7

3.3.1 Discussion of Results for (KNN)

In Tables 6 - 7, we demonstrate the results of using kNN on the Lindley recursion with $k = \{1,5,10,25\}$ nearest neighbors. All experiments performed in the tables were done out of sample since the in-sample results would result in a zero MSE in the case $k = 1$. We observe that using a high value of k does not improve the results. Thus, the results are good for lower values of k, especially the case where $k = 5$. Moreover, we observe that the k-nearest neighbor results work better in heavy traffic. One explanation for this is that in heavy traffic, the inter-arrival and service times are small, so one does not expect much change from a data point that is close to the one you want to predict.

3.4 Gaussian Processes

In this section, we apply Gaussian processes (GPs) to the Lindley recursion learning problem. Gaussian processes provide a function space view of modelling, whereby one places a prior distribution over functions, and reason about the properties of likely functions under this prior Williams and Rasmussen (1996). Given data, one infers a posterior distribution over functions to make predictions. A key ingredient to GPs is the kernel function, which measures similarity of new points with the training data. In general, we say that a stochastic process $f(x)$ is a Gaussian process (GP) if for any finite collection of inputs $X = x_1,\ldots,x_n \in R^d$,the vector of function values $[f(x_1),\ldots,f(x_n)]$ is is jointly Gaussian. In our context of Lindley's recursion, we use Gaussian processes to predict the one-step waiting times using the previous information.

Table 8: Gaussian One Step Prediction (Light Traffic) M/M/1 Queue. (In Sample)

μ	1	1	1	1	10	10	10	10
λ	0.5	0.75	0.9	.99	5	7.5	9	9.9
Exponential	1.77e-6	1.12e-5	1.39e-4	4.95e-4	1.52e-8	2.14e-7	6.47e-7	4.66e-7
Squared Exponential	3.39e-4	8.37e-4	0.00472	0.0134	5.61e-6	1.64e-5	2.84e-5	1.07e-5
Matern 52	1.34e-4	3.85e-4	0.00271	0.0122	1.22e-6	7.3e-6	1.34e-5	7.72e-6
Rational Quadratic	1.62e-4	4.95e-4	0.00345	0.0133	1.43e-6	9.25e-6	1.75e-5	9.4e-6

Table 9: Gaussian One Step Prediction (Heavy Traffic) M/M/1 Queue. (In Sample)

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
Exponential	1.84e-8	1.35e-8	1.24e-8	9.47e-9	1.6e-8	1.34e-8	1.08e-8	9.52e-9
Squared Exponential	2.47e-7	3.23e-7	1.97e-7	2.18e-7	2.86e-8	2.87e-8	3.09e-8	4.25e-8
Matern 52	1.33e-7	1.69e-7	1.27e-7	1.34e-7	1.99e-8	2.06e-8	2.51e-8	3.67e-8
Rational Quadratic	1.51e-7	2.07e-7	1.49e-7	1.64e-7	2.42e-8	2.87e-8	2.96e-8	4.2e-8

Table 10: Gaussian One Step Prediction (Light Traffic) M/M/1 Queue. (Out of Sample)

μ	1	1	1	1	10	10	10	10
λ	0.5	0.75	0.9	.99	5	7.5	9	9.9
Exponential	0.00303	0.00287	0.00847	0.0173	0.00581	7.05e-5	1.58e-4	0.209
Squared Exponential	0.0342	0.0157	0.00803	0.00817	0.0468	1.86e-4	1.71e-4	0.375
Matern 52	0.00236	0.00152	0.00437	0.00728	0.02	2.35e-5	4.32e-5	0.049
Rational Quadratic	0.00336	0.0021	0.0052	0.00789	0.0136	4.28e-5	7.66e-5	0.14

Table 11: Gaussian One Step Prediction (Heavy Traffic) M/M/1 Queue. (Out of Sample)

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
Exponential	5.18e-7	1.42e-6	8.83e-7	1.4e-6	1.84e-8	4.64e-8	1.74e-8	2.88e-8
Squared Exponential	1.52e-6	2.02e-6	1.51e-6	7.75e-7	2.91e-8	3.78e-8	3.76e-8	3.07e-8
Matern 52	5.0e-7	8.44e-7	2.36e-7	2.58e-7	2.19e-8	2.74e-8	3.21e-8	2.47e-8
Rational Quadratic	5.46e-7	1.21e-6	3.69e-7	3.27e-7	2.55e-8	3.78e-8	3.64e-8	3.02e-8

3.4.1 Discussion of Results for (GPs)

In Tables 8 - 11, we demonstrate the results of using Gaussian processes on the Lindley recursion with four different kernels. We see that the GPs work quite well at approximating the Lindley recursion for all of the kernels. We see that the Matern 52 kernel performs the best out of all of the kerel used. We also observe that the out of sample performance is worst than the in-sample performance. However, this gap disappears when the queueing model has large rates. Unlike some of the other methods, the Gaussian processes do not work as well when the system is in heavy traffic. It is not a large difference in performance, but we do notice a small gap in performance in the heavy traffic regime.

3.5 Deep Neural Networks

In this section, we apply feed-forward deep neural networks to the Lindley recursion learning problem. Deep neural networks have found tremendous success in a variety of disciplines as they are quite flexible at representing any function, see for example Csáji (2001). Our goal is to understand how deep neural networks will help us approximate the Lindley recursion and also understand what activation functions work well in approximating the Lindley recursion.

Table 12: Deep Neural Network One Step Prediction (Light Traffic) M/M/1 Queue. (In Sample)

μ	1	1	1	1	10	10	10	10
λ	0.5	0.75	0.9	.99	5	7.5	9	9.9
ReLU	0.0603	11.3	67.9	0.919	0.00155	0.00352	0.0153	0.0287
Tanh	1.65	6.63	211.0	1620.0	0.00533	0.019	0.0251	0.0957
Sigmoid	3.18	13.4	266.0	1850.0	0.0309	0.231	0.706	0.653
Hard Sigmoid	3.69	14.3	266.0	1710.0	0.0297	0.247	0.669	0.631

Table 13: Deep Neural Network One Step Prediction (Heavy Traffic) M/M/1 Queue. (In Sample)

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
ReLU	1.3e-4	1.88e-4	2.56e-5	8.28e-5	3.28e-6	6.42e-6	1.99e-6	8.57e-6
Tanh	9.96e-5	1.14e-4	4.26e-5	3.33e-4	2.04e-6	5.31e-6	3.13e-6	1.49e-5
Sigmoid	2.61e-4	0.0019	0.0021	0.00412	8.3e-6	4.37e-5	4.85e-5	1.54e-4
Hard Sigmoid	3.36e-4	0.00214	0.00202	0.00508	2.53e-5	9.19e-6	9.77e-5	1.67e-4

Table 14: Deep Neural Network One Step Prediction (Light Traffic) M/M/1 Queue. (Out of Sample)

μ	1	1	1	1	10	10	10	10
λ	0.5	0.75	0.9	.99	5	7.5	9	9.9
ReLU	0.0522	5.05	17.0	0.865	0.00661	0.00312	0.0163	0.0312
Tanh	0.878	2.19	42.9	2560.0	0.0132	0.0145	0.014	0.912
Sigmoid	1.96	5.67	66.3	2910.0	0.098	0.0638	0.327	2.25
Hard Sigmoid	2.34	5.83	66.4	2700.0	0.0928	0.0862	0.343	2.17

Table 15: Deep Neural Network One Step Prediction (Heavy Traffic) M/M/1 Queue. (Out of Sample)

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
ReLU	1.29e-4	1.62e-4	4.61e-5	6.52e-5	2.84e-6	7.36e-6	1.85e-6	7.43e-6
Tanh	1.04e-4	1.04e-4	4.33e-5	3.39e-4	1.95e-6	5.42e-6	2.88e-6	1.44e-5
Sigmoid	2.88e-4	0.00165	0.00334	0.00275	8.08e-6	4.45e-5	3.42e-5	1.33e-4
Hard Sigmoid	3.77e-4	0.00127	0.00333	0.0021	2.54e-5	1.04e-5	7.89e-5	1.7e-4

3.5.1 Discussion of Results for (DNNs)

In Tables 12 - 15, we find that the deep neural networks are able to recover the Lindley recursion quite well. We also observe that the function approximations improve as we scale the system larger. However, as similar to the Gaussian process setting, as we let $\lambda \to \mu$ we observe a slightly worse performance than the light traffic settings. We use four different activation functions and we observe that the best one is the ReLU activation function. We can explain this outstanding performance from the structure of the ReLU activation function itself. The ReLU activation function is a maximum function and the Lindley recursion naturally fits this activation function. Since we have only explored the one-step Lindley recursion in this section,

one can only wonder if the ReLU will also perform well for multiple steps since DNNs are convolutions of the activation functions and the multi-step Lindley recursion is also of this form.

3.6 Multiple Step Predictions

In this section, we are concerned with multiple step predictions of the wait time in the G/G/1 queue. We describe the multi-step problem as learning a max-plus convolution i.e.

$$\begin{aligned} W_{n+m} &= \max\left(W_{n+m-1} + S_{n+m-1} - A_{n+m-1}, 0\right) \\ &= \max\left(\max\left(W_{n+m-2} + S_{n+m-2} - A_{n+m-2}, 0\right) + S_{n+m-1} - A_{n+m-1}, 0\right) \\ &= f_{n+m-1} \circ f_{n+m-2}(X_{n+m-2}) \\ &= f_{n+m-1} \circ f_{n+m-2} \circ \cdots \circ f_n(W_n, A_n, S_n). \end{aligned}$$

We attempt to analyze the multi-step Lindley recursion only using the best functions from the one-step and present their results below.

Table 16: Deep Neural Network Multiple Step (m=2) Prediction (Heavy Traffic) M/M/1 Queue. (In Sample)

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
DNN (Relu)	1.74e-4	2.31e-4	7.78e-4	3.28e-4	5.1e-6	9.13e-6	5.6e-5	3.92e-6
DNN (Sigmoid)	4.38e-4	0.00112	0.00367	0.00814	2.42e-5	2.13e-5	8.95e-5	1.88e-4
GP (Squared Exp)	6.8e-5	1.27e-4	1.07e-4	1.33e-4	7.81e-7	1.08e-6	1.27e-6	1.12e-6
GP (Matern 52)	1.03e-4	1.73e-4	1.68e-4	1.94e-4	1.05e-6	1.47e-6	1.94e-6	1.85e-6
KNN (k = 5)	1.31e-4	2.39e-4	2.25e-4	2.51e-4	1.36e-6	1.89e-6	2.41e-6	2.4e-6
KNN (k = 10)	1.24e-4	2.22e-4	2.11e-4	2.39e-4	1.25e-6	1.73e-6	2.41e-6	2.37e-6

Table 17: Deep Neural Network Multiple Step (m=3) Prediction (Heavy Traffic) M/M/1 Queue. (In Sample)

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
DNN (Relu)	1.68e-4	2.53e-4	8.21e-4	3.9e-4	4.95e-6	4.35e-6	7.7e-6	1.32e-5
DNN (Sigmoid)	3.95e-4	5.63e-4	0.00371	0.00558	3.31e-5	1.3e-5	5.33e-5	4.05e-4
GP (Squared Exp)	1.37e-4	2.22e-4	3.47e-4	3.07e-4	1.58e-6	2.55e-6	4.07e-6	4.42e-6
GP (Matern 52)	1.36e-4	2.21e-4	3.42e-4	3.06e-4	1.57e-6	2.52e-6	4.07e-6	4.41e-6
KNN (k = 5)	1.74e-4	2.74e-4	4.2e-4	3.78e-4	1.96e-6	3.18e-6	5.23e-6	5.45e-6
KNN (k = 10)	1.59e-4	2.47e-4	3.97e-4	3.59e-4	1.85e-6	3.0e-6	4.82e-6	5.18e-6

Table 18: Deep Neural Network Multiple Step (m=2) Prediction (Heavy Traffic) M/M/1 Queue. (Out of Sample)

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
DNN (Relu)	1.42e-4	1.97e-4	0.00648	3.27e-4	4.84e-6	1.05e-5	8.63e-5	1.47e-5
DNN (Sigmoid)	3.42e-4	9.02e-4	0.0213	0.00563	2.37e-5	2.38e-5	7.97e-5	6.65e-4
GP (Squared Exp)	9.85e-5	1.61e-4	0.0012	1.89e-4	1.03e-6	1.69e-6	1.44e-6	1.31e-5
GP (Matern 52)	9.34e-5	1.55e-4	0.00125	1.87e-4	9.88e-7	1.63e-6	1.39e-6	1.91e-6
KNN (k = 5)	1.09e-4	1.87e-4	0.00135	2.38e-4	1.19e-6	2.08e-6	1.77e-6	1.45e-5
KNN (k = 10)	1.05e-4	1.75e-4	0.00144	2.28e-4	1.11e-6	1.94e-6	1.57e-6	1.55e-5

Table 19: Deep Neural Network Multiple Step (m=3) Prediction (Heavy Traffic) M/M/1 Queue. (Out of Sample)

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
DNN (Relu)	1.47e-4	4.11e-4	0.00117	5.09e-4	4.67e-6	5.44e-6	7.67e-6	1.73e-5
DNN (Sigmoid)	2.96e-4	0.00106	0.00554	0.00507	3.1e-5	1.62e-5	3.33e-5	3.16e-4
GP (Squared Exponential)	1.37e-4	3.1e-4	4.1e-4	5.93e-4	1.59e-6	2.92e-6	3.4e-6	3.29e-6
GP (Matern 52)	1.37e-4	2.98e-4	4.02e-4	4.59e-4	1.59e-6	2.91e-6	3.43e-6	3.29e-6
KNN (k = 5)	1.6e-4	3.94e-4	5.61e-4	5.83e-4	1.86e-6	3.55e-6	4.25e-6	4.6e-6
KNN (k = 10)	1.46e-4	3.97e-4	5.19e-4	5.57e-4	1.71e-6	3.27e-6	3.93e-6	4.6e-6

3.6.1 Discussion of Results for Multi-step Predictions

In Tables 16 - 19, we analyze the multi-step problem of trying to recover the Lindley recursion. We apply different machine learning methods to uncover the best method. We find that all of the method do quite well, however, we observe that the best method uses Gaussian processes, especially with the Matern 52 kernel. We also observe that the function approximations improve as we scale the system larger and as we let $\lambda \to \mu$ we observe a slightly worse performance than the light traffic settings.

3.7 Comparison of Machine Learning Methods

Overall, we observe that by using current machine learning methods we can successfully replicate the Lindley recursion in in-sample and out of sample experiments. We find that all of the machine learning methods do quite well, however, Gaussian processes seems to outperform the other methods. This is significant in the sense that as the number of training samples scales to be large, GPs become much more intractable since they involve matrix inversions. However, in our setting where the number of samples is 1000, we are not spending much time computationally.

4 CONCLUSION

In this paper, we show the power of current machine learning methods to approximate the Lindley recursion. We show in this work that the best method for approximating the Lindley recursion is Gaussian processes with the Matern 52 kernel. 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. However, there remain many interesting questions that need to be explored in this domain.

First, although the Lindley recursion works for general service and general inter-arrival times, we were unable to explore the impact of general distributions due to space constraints. This is still an open problem to understand how the general distributions will affect the learning problem for the general G/G/1 queue.

Second, although the Lindley equation does not directly apply in the multi-server setting, it still is an interesting question to apply these techniques to multi-server systems and systems where there is blocking such as Erlang-loss queues. As we have noticed that system performance is improved when the queues are heavily loaded, it might be the case that the machine learning methods might not work well in queues with blocking as blocking reduces the queue length.

Third, it would be interesting to see how well machine learning methods approximate the waiting time in networks of queues. First, one would need to construct a Lindley recursion for a network of queues.

Lastly, there are many types of recursions beyond the Lindley recursion. For example, there is the random sign Lindley recursion Vlasiou and Palmowski (2010) and the non-increasing Lindley equation Vlasiou (2007) that could be studied from a simulation and machine learning perspective. We plan to work on these generalizations in future work.

ACKNOWLEDGEMENTS

We would like to thank the Cornell University and the Sloan Foundation for providing Sergio Palomo with a Sloan Fellowship for his research. Sergio Palomo would like to also thank his mother Bertila Sanchez for supporting him throughout his graduate student career. Jamol Pender would like to acknowledge Misha Padidar and Matthew Davidow for their help in learning Gaussian processes.

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.

Baccelli, F., and P. Brémaud. 2013. *Elements of queueing theory: Palm Martingale calculus and stochastic recurrences*, Volume 26. New York: Springer Science & Business Media.

Bartel, A. P., C. W. Chan, and S.-H. Kim. 2020. "Should hospitals keep their patients longer? the role of inpatient care in reducing postdischarge mortality". *Management Science* 66(6):2326–2346.

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.

Buzacott, J. A., and J. G. Shanthikumar. 1993. *Stochastic models of manufacturing systems*, Volume 4. Prentice Hall Englewood Cliffs, NJ.

Csáji, B. C. 2001. "Approximation with artificial neural networks". *Faculty of Sciences, Etvs Lornd University, Hungary* 24(48):7.

Delimitrou, C., and C. Kozyrakis. 2013. "QoS-aware scheduling in heterogeneous datacenters with paragon". *ACM Transactions on Computer Systems (TOCS)* 31(4):1–34.

Fu, M. C., and J.-Q. Hu. 2012. *Conditional Monte Carlo: Gradient estimation and optimization applications*, Volume 392. New York: Springer Science & Business Media.

Glasserman, P., and Y.-C. Ho. 1991. *Gradient estimation via perturbation analysis*, Volume 116. New York: Springer Science & Business Media.

Kendall, D. G. 1951. "Some problems in the theory of queues". *Journal of the Royal Statistical Society: Series B (Methodological)* 13(2):151–173.

Kim, S.-H., C. W. Chan, M. Olivares, and G. Escobar. 2015. "ICU admission control: An empirical study of capacity allocation and its implication for patient outcomes". *Management Science* 61(1):19–38.

Kin, W., and V. Chan. 2010. "Generalized Lindley-type recursive representations for multiserver tandem queues with blocking". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 20(4):1–19.

Konheim, A. G. 1975. "An elementary solution of the queuing system G/G/1". *SIAM Journal on Computing* 4(4):540–545.

Koole, G., and A. Mandelbaum. 2002. "Queueing models of call centers: An introduction". *Annals of Operations Research* 113(1-4):41–59.

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.

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.

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.

Shaked, M., and J. G. Shanthikumar. 1988. "Stochastic convexity and its applications". *Advances in Applied Probability* 20(2):427–446.

Shortle, J. F., J. M. Thompson, D. Gross, and C. M. Harris. 2018. *Fundamentals of queueing theory*, Volume 399. John Wiley & Sons.

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.

Vlasiou, M. 2007. "A non-increasing Lindley-type equation". *Queueing Systems* 56(1):41–52.

Vlasiou, M., and Z. Palmowski. 2010. "Tail asymptotics for a random sign Lindley recursion". *Journal of applied probability* 47(1):72–83.

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.

Williams, C. K., and C. E. Rasmussen. 1996. "Gaussian processes for regression". In *Advances in Neural Information Processing Systems*, 514–520.

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 assistant professor in operations research and information engineering and systems engineering at Cornell University. He earned his PhD in the Department of Operations Research and Financial Engineering (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/.