Abstract

Successful formulation of queuing systems, where packets arrive, wait in various queues, receive service and exit after some time depends on the nature of the arrival rate, service time, number of servers, queue discipline, size of the population, number of entities in the system and delay. Such queuing processes are described by using the Kendall notation, which uses mnemonic characters that specify the queuing system A/B/C/D/E/F. Literature on queuing theory shows that, results based on theoretical formulations are well established for the queuing problems with Poisson arrivals and exponential service duration, and most of the queuing systems are modeled with combinations of single or multiple servers with finite or infinite capacity. However the presence of uncertainty in the distribution of arrival and service rates makes these systems challenging in respect of analysis and design. Queuing simulation as a method, is used to analysis how systems with limited resources distribute those resources to the elements waiting to be served. In this work, a single server infinite capacity queuing model simulated using ANN and the results are compared with those obtained from analytically.

Keywords: Queuing theory, ANN, simulation, single server, Infinite capacity.

1 Introduction

Queuing theory deals with problems which involve queuing or waiting. Queues are formed because, resources are limited. Any system in which arrivals place demand upon a finite capacity
resource may be termed as queuing system. Queuing systems aim to balance between service to customers and resources, thereby optimizes the cost of service. Queuing theory based analysis is regularly used in the field of telecommunications [7, 11, 51], predicting computer performance, computer networks [31], call centers [32], hospitals [33, 39], traffic [41, 60] etc. Queuing simulation is based on the idea of queuing theory and helps to analyze resource allocation and process duration, for a variety of different applications. Queuing simulation as a method is used to analyze, how systems with limited resources distribute those resources to elements waiting to be served. Queuing simulation is used to determine possible bottlenecks in resource allocation and this enables to analyze their consequence on waiting lines [2]. It thereby makes it possible to estimate waiting times or optimize server utilization. In this study, an Artificial Neural Network (ANN) approach has been applied to a single server infinite capacity queuing system and validated the results using classical mathematical analysis.

2 Queuing Models

Queuing theory is an analytical tool that has provided many insights to service providers, while designing new service systems and managing existing ones. This established theory helps us to quantify, the appropriate service capacity to meet demand, balancing system utilization and minimizing waiting time [39]. The theory of queues deals with the mathematical studies of random mass service phenomena. Such phenomena appear in physics, engineering, industry, transportation, commerce, business and several other fields. Queues and theory of queues were first analysed by Erlang in 1913 in the context of telephone facilities [11]. It is extensively practiced or utilised in industrial settings or retail sector operation management, and falls under the purview of decision sciences. Queuing theory application is an attempt to minimize the cost through
minimisation of inefficiencies and delays [33]. The most important findings that are connected with classical queuing theory involves investigation of the stochastic behaviour of the waiting time, the queue size, the busy periods, the idle periods and the departures [1, 2, 13, 16, 18]. It is of interest to find the distribution of the actual waiting time of the \( n^{th} \) customer, the virtual waiting time at time \( t \), the queue size immediately before the arrival of the \( n^{th} \) customer, the queue size at time \( t \), the queue size immediately after the \( n^{th} \) departure, and the length of the busy periods and the idle periods [3, 8, 26, 28, 42, 52, 53, 54, 58].

Classical results of queuing models have largely been extended in the past hundred years. Single server queuing models [9, 27], many server queuing models [25] with various service disciplines have been introduced like First In First Out (FIFO) [29], Last In First Out (LIFO) [35], random service [5, 14], priority service and service in batches [22]. Different arrival processes have been considered like, Poisson input [25], recurrent input [52], superposition of various input processes [38] and group arrivals [36]. Queuing networks have been investigated with queues in series [22], queues in parallel [37], queues with limited queue size [13] and limited waiting time. Attention also has been made on stationary (steady state) [18, 40] and time dependent behavior of queues [52]. Various generalizations for queuing model developed like, queues, in which customers are arriving in groups and are served in batches, priority queues and finite queues, in which the queue size or the waiting time is limited [13, 22, 57]. Mathematical methods have been developed for studying various queuing models and in a queuing system, minimizing the time that customers have to wait and maximizing the utilization of the servers or resources, are conflicting goals. It is seen from the review of the literature that, it is too hard to accurately determine, the performance of the queuing model using classical mathematical techniques [29, 30, 46, 56]. Simulation modeling has long been part of queue modeling and many of the research articles in the literature reported use of simulation models [47–49, 59–62]. This motivated the author to develop a queuing model using computer simulation.
3 Artificial Neural Networks (ANN)

Recently, ANN models have received increased attention due to their potential applications in several areas such as, mathematics [24], transportation problems [17, 21, 24, 55], engineering [4, 15, 20, 44], pattern recognition, signal processing, system identification and control, medicine, economics [6, 43], meteorology, psychology, neurology, weather and market trends forecasting [10], in the prediction of mineral exploration sites, in electrical and thermal load prediction [12, 23, 50, 63] in adaptive and robotic control and many others [4, 21]. Karim et al (2003) reported that, artificial neural networks have been used to solve complicated pattern recognition and estimation problems, not amenable to conventional mathematical modeling [24]. Zadeh et al (2005) reported that, neural networks are a class of nonlinear system capable of adaptively learning and performing tasks accomplished by other systems and used in the broad range of applications [61]. The adaptive learning power of neural networks, has also proven useful in various contexts of the literature [19]. The promise of neural network modeling approach is to replace the analytical difficulties encountered in the other modeling approaches, with a straight forward computational algorithm [60]. This new modeling technique is based on imitating the structure and mechanisms of the human brain, and being used in more and more applications where classical approaches fail or they are too complicated to be used. ANNs are applied to estimate desired output parameters when enough experimental data is provided. Therefore, ANNs allow the modelling of physical phenomena in complex system, without requiring explicit mathematical representations [12, 19].

For unknown nonlinear multi input and single output (multi output) function, artificial neural networks have been successfully proved as universal approximators because, they do not require prior knowledge of the input data distribution [63]. Although the concept of ANN analysis has been discovered nearly 50 years ago, it is only in the last two decades that application software has been developed to handle practical problems. Specifically, they are good for tasks involving incomplete data sets, fuzzy or incomplete information, and for highly complex and ill-defined problems, where humans usually decide on an intuitive basis [43].

The architecture of an ANN is usually divided into three parts, an input layer, a hidden layer(s) and output layer, as shown in Fig. 1. The information contained in the input layer is mapped to the output layer through the hidden layer(s). Each unit can send its output to the units only on the higher layer and receive its input from the lower layer. For a given modeling problem, the numbers of nodes in the input and output layers are determined from the physics of the problem, and equal to the numbers of input and output parameters, respectively [12]. Such a Multi Layer Feed forward (MLF) neural network, comprises layers of neurons interconnected by forward connections is shown in Fig. 2. The strength of these connections is characterized by their weights. A neuron in
MLF network, receives information from neurons in the preceding layer, processes the information using an internal model and passes its output to neurons in the succeeding layer. Although the structure of these neurons is quite simple, it consists of a summation operator, a local bias, and a nonlinear transfer function. The elaborate layered structure and massive connections of the neural network provide great interaction among these simple processing elements and enables them to collectively behave in a complex manner. ANN need to be trained by suitable algorithm using the input and output data available by classical methods or experiments. There are different learning algorithms that can be applied to train a neural network. The most popular of them is the back propagation algorithm. It is very difficult to know, which training algorithm will be the fastest for a given problem, and the best one is usually chosen by trial and error \[41\]. Sha et al. (2004) reported that, Back Propagation Neural Networks (BPN) are widely used and produce good results in prediction and pattern recognition \[49\]. Rumelhart et al. (1986) reported that, widely used procedure for training MLF networks is the back propagation algorithm \[45\]. This algorithm, which performs gradient descent in the error space, provides a recursive formula for updating the connection weights, starting from the output layer and working backwards toward the first (hidden) layer and developing adaptive nonlinear controllers for complex systems.

The performance of the ANN based prediction, is evaluated by a regression analysis between the network outputs i.e., predicted parameters, and the corresponding targets i.e., experimental (actual) values. The correlation coefficient, mean relative error and root mean square error are the three criteria that can be used in this evaluation. The correlation coefficient is a measure of how well the variation in the output is explained by the targets. This coefficient between the actual versus predicted output is defined as follows

\[
R(a, p) = \frac{\text{Cov}(a, p)}{\sqrt{\text{Cov}(a, a)\text{Cov}(p, p)}}
\]  

Where, Cov\((a, p)\) is covariance between \(a\) and \(p\) sets that, refer to the actual output and predicted output sets, respectively. Likewise, Cov\((a, a)\) and Cov\((p, p)\) are the auto covariance of \(a\) and \(p\) sets, correspondingly. The correlation coefficient ranges between -1 and +1. R values closer to +1 indicate a stronger positive linear relationship while, R values closer to -1 indicate a stronger negative relationship. The mean relative error, which shows the mean ratio between the errors and experimental values, is evaluated by

\[
\text{MRE} (%) = \frac{1}{N} \sum_{i=1}^{N} 100 \left( \frac{a_i - p_i}{a_i} \right).
\]  

Where \(N\) is the number of points in the data set \[20\].
Finally, the root mean square error is defined as

\[ RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (a_i - p_i)^2} \]  

(3.3)

From the literature review it is clear that, our search for a neural network approximation of queuing system is meaningful however, it does not tell us how to find such a neural network. The inputs and outputs of a neural network are usually determined by the application but, the number of hidden layers, the number of neurons in each hidden layer, the type of transfer functions, and the number of training iterations, all are subject to experimentation. Despite progresses made in understanding the basic behavior of systems controlled by neural networks [34], there are still no general rules regarding the selection of neural network topologies and parameters. Experimentation and exploitation of the special structure of a specific problem is often necessary, to achieve the desired performance. In this study, the values obtained from the formulas of queuing theory were used as a training set for ANN and is discussed in the next section.

4 M/M/1 queuing model Simulation with ANN

A queuing system is specified when we know (i) the input, (ii) the queue discipline and (iii) the service mechanism. Kendall [30] proposed queuing models using three factors written A/S/c in 1953, where A denotes arrival process, S service time distribution, c the number of servers at the node. It has since been extended to A/S/c/K/N/D, where K the number of places in the system, N the calling population and D the queue discipline. When the final three parameters are not specified (e.g. M/M/1 queue), it is assumed \( K = \infty \), \( N = \infty \) and \( D = FIFO \).

There are three type of inputs D, M and \( E_k \)

D(deterministic, or regular):

\[ A(u) = 0 (u < a) \]
\[ A(u) = 1 (u \geq a) \]  

(4.4)

M(random, or Poissonian):

\[ A(u) = 1 - e^{-u/a} \]  

(4.5)

With a D input the customers arrive at regular intervals of time, while with a M input the customers arrive at random and the third intermediate type of input is \( E_k \)

\[ E_k (\text{Erlangian}) : \quad dA(u) = \frac{(k/a)^k}{\Gamma(k)} e^{-ku/a} u^{k-i} du \]  

(4.6)
And if no special assumption is made the symbol G is used [30].

The distribution of arrival and service times are classified exactly the same way, with the aid of the symbols D, M, $E_k$ and G. With D distribution for the service time, each customer is served for exactly the same length of time. With M distribution for the service time, follows the negative exponential law. Once again the $E_k$ distribution, is of an intermediate form [30]. With these conventions a particular type of queuing system can be identified by giving it a label such as M/M/1(Poissonian arrivals; negative exponential service time; single server). Erlang (1908-29) reported M/M/S, M/D/S, and $M/E_k/1$ queuing models [8], Pollaczek (1930) reported $E_k/G/1$ queuing system, Khintchine (1932), Kendall (1951) reported M/G/1 queuing model. Numerous authors not mentioned over here investigated the various queuing models and presented heavy and difficult mathematical analysis to find solutions for the mathematical models developed. The equations available to find solutions are highly complex and arriving solution is time consuming. This work intends to simulate this complex queuing models using easy computer simulation. In this study, the Neural Network Toolbox in MATLAB is used to develop the ANN model for M/M/1 queuing system, since MATLAB is widely used [55].

Simulation is the process of conducting experiments, with a model of the system that is being studied or designed. It is a powerful technique for both analyzing and synthesizing engineering and other natural systems. The simulation procedure is basically an iterative procedure and may be described as an input-output study with feedback, provided to guide the changes in the input parameters. The inputs define the set of events and conditions, to which the system can be subjected in the real world, and the outputs predict the system response. For simulating the queuing model, Input and output values of M/M/1 queuing model, is analytically solved using Kandal and Little’s formulas and is used as input and output values to train ANN. Five data are used as input to train the ANN model they are, (i) Arrival rate $\lambda$, (ii) service rate $\mu$, (iii) No of customers in the system exceeds 'k', (iv) time 't' and (v) size of the population 'n', and fourteen properties of queuing model, calculated using (4.7) to (4.20) were targeted as output.

(i) Traffic intensity

$$\rho = \frac{\lambda}{\mu} \quad (4.7)$$

(ii) Probability of system being idle or free

$$P_0 = 1 - \frac{\lambda}{\mu} \quad (4.8)$$
(iii) Probability of \( n \) customers in the system

\[ P_n = \left( \frac{\lambda}{\mu} \right)^n \left( 1 - \frac{\lambda}{\mu} \right) \]  \hfill (4.9)

(iv) Probability that at-most one customer in the system

\[ P(n \leq 1) = P_0 + P_1 \]  \hfill (4.10)

(v) Average number of customers in the system

\[ L_s = \frac{\lambda}{\mu - \lambda} \]  \hfill (4.11)

(vi) Average number of customers in the queue

\[ L_q = \frac{\lambda^2}{\mu(\mu - \lambda)} \]  \hfill (4.12)

(vii) Average waiting time of a customer in the system

\[ W_s = \frac{1}{\mu - \lambda} \]  \hfill (4.13)

(viii) Average waiting time of a customer in the queue

\[ W_q = \frac{\lambda}{\mu(\mu - \lambda)} \]  \hfill (4.14)

(ix) Probability of queue length being greater than or equal to \( k \)

\[ P(N \geq k) = \rho^k \]  \hfill (4.15)

(x) Probability of queue length being greater than \( k \)

\[ P(N > k) = \rho^{k+1} \]  \hfill (4.16)

(xi) Probability that the waiting time of a customer in the system exceeds \( t \)

\[ P(W_s > t) = e^{-(\mu-\lambda)t} \]  \hfill (4.17)

(xii) Probability that the waiting time of a customer in the queue exceeds \( t \)

\[ P(W_q > t) = \frac{\lambda}{\mu} e^{-(\mu-\lambda)t} \]  \hfill (4.18)

(xiii) Average queue length

\[ L_w = \frac{\mu}{\mu - \lambda} \]  \hfill (4.19)

(xiv) Probability of ten customers in the system

\[ P_{10} = (1 - \rho)\rho^{10} \]  \hfill (4.20)

In order to develop the ANN for the queuing system, the available data set from the analytical work was divided into training, validation and test sets. The data sets consists of 160 input-output pairs. While 70% of the data set was randomly assigned as the training set, 15% was employed for validation and the remaining 15% for testing of the network. The architecture of the ANN for the queuing system with number of input and output parameters, is schematically illustrated in Fig.2.
5 Result and Discussions

As shown in the fig.2, an ANN model with single hidden layer is developed and trained in this work. The number of neurons in the input layer is equal to number of input parameters, in this case it is 5 neurons. The number of neurons in the output layer is equal to number of output parameters, in this case it is 14 neurons. It is by trial and error method that, the number of hidden layers, number of neurons in each hidden layer, transfer function between input layer to hidden layer and hidden layer to output layer for the network is fixed. For a particular problem performance of the developed ANN has to be analysed with regression analysis. Once the simulated output value is near by or equal to the target and within the predetermined error, the simulation process could be stopped.

Performance of ANN model developed to be tested by two parameters. one parameter is, the least number of epochs by which the desired error value is achieved by the ANN and another parameter is the capacity of the ANN to arrive least error value for a particular epoch. In this work, as shown in the figure 2, neural network with one hidden layer is trained with different number of neurons starting from 1 neuron to 28 neurons up to 1000 epochs. For each ANN model, the performance of the ANN is analyzed by comparing the error value arrived by developed models at 1000 epochs and also using regression analysis. Keeping the number of epochs fixed and comparing the value of minimum error which the ANN model could reach, is the best way to measure the performance of the ANN model along with regression analysis. Some of the training graphs is presented here for understanding the performance of the ANN model with 6, 9 and 28 neurons in the hidden layer. Up to 5 neurons in the hidden layer, no significant performance in terms of error value is achieved. Fig.3 shows the performance of 6 neurons and the error value achieved is
0.001 for 1000 epochs, but after 500 epochs no progress in error value is observed. It is observed that, the model could achieve the error value of 0.8 by 200 epochs and after no further progress is found. Fig.4 shows that, ANN model with 9 neurons, training stopped with highest error value of 78. It is observed from Fig.5 that, the ANN model with 28 neuron reached highest minimum error value at 1000 epochs and is converging further. ANN models with 29 and above neurons there is no significant improvement in the performance of the network found. Based on the performance analysis of various ANN model it is found ANN model with 28 neuron is resulting minimum error and hence ANN model with 28 neuron is selected for validation and tested with available data.

Fig.6 shows the regression analysis trend, with correlation coefficient (R) close to +1, for training, validation and test outputs with targets indicates stronger positive linear relationship with classical results obtained. The mean relative error and root mean square error is found within the acceptable limits. Fig. 6 shows that the training, validation and testing curves are converging and indicates that this neural network could be used to simulate M/M/1 queuing model problems.
6 Conclusion

An artificial neural network model capable of simulating M/M/1 queuing model is developed using feed forward back propagation algorithm with single hidden layer with 28 neurons. It is found that simulated values using ANN are matching with that of calculated values using classical mathematical approach. Regression analysis, mean relative error and RMSR values are within the acceptable limits.

References


Received: May 10, 2015; Accepted: June 23, 2015