Les Cahiers du GERAD ISSN: 0711–2440

Forecasting Rail Transportation Demand Using Artificial Neural Network

S. Sharif Azadeh,
R. Labib, G. Savard

G–2009–71

November 2009

Forecasting Rail Transportation Demand Using Artificial Neural Network

Shadi Sharif Azadeh[*]
Richard Labib
Gilles Savard[*]

Department of Mathematics and Industrial Engineering
École Polytechnique de Montréal
Montréal (Québec) Canada, H3C 3A7

shadi.sharifazadeh@polymtl.ca
richard.labib@polymtl.ca
gilles.savard@polymtl.ca

[*] *and GERAD*

November 2009

Les Cahiers du GERAD

G–2009–71

Abstract

This study analyzes the use of neural network to produce accurate forecasts of total bookings and cancellations before departure, of a major rail operator. The model used is an improved Multi-Layer Perceptron (MLP) describing the relationship between number of passengers and factors affecting this quantity based on historical data. Relevant pre-processing approaches have been employed to make learning more efficient. The generalization of the network is tested to evaluate the accuracy prediction of the regression model for future trends of reservations and cancellations using actual railroad data. The result is an accurate forecast of the number of passengers with a prediction error around 8%.

Key Words: Demand forecasting, Pre-processing, Neural Network (NN), Multi-Layer Perceptrons (MLP), Transportation.

Résumé

Cette étude analyse le potentiel d'un réseau de neurones artificiel pour prédire les réservations et les annulations de billets, d'une classe de prix donnée, dans différentes gares données, lors d'un jour donné, pour une importante compagnie ferroviaire. Puisque la réservation est traitée plusieurs semaines avant le départ, il y a toujours une possibilité que la réservation soit annulée. Il est donc important de faire des prévisions fiables des réservations et des annulations avant le départ.

Le modèle utilisé est le Perceptron Multi-Couches (MLP). Ce réseau modélise la relation entre les variables dépendantes et indépendantes qui sont tirées des historiques des réservations et des annulations. Après, la capacité de généralisation du réseau de neurones est examinée. L'ensemble des données proviennent de la compagnie de transport européenne TGV pour l'itinéraire Paris-Bruxelles.

Un choix judicieux de prétraitement des données a été effectué pour accélérer l'apprentissage. Un réseau de neurones supervisé, avec deux couches cachées, chacune formée de cinq neurones, est entraîné en utilisant un algorithme d'apprentissage de rétro-propagation. Les résultats numériques sont concluants et démontrent une bonne généralisation du réseau de neurones, puisque l'erreur de prédiction est inférieure à 8%.

1 Introduction

Statistical forecasting techniques for applications such as pricing, inventory control, demand forecasting, and overbooking, are widely used in transportation industries (Sen (1985); Xiaolong (2007)). These methods examine historical data to verify the underlying process generating the variable under consideration (Devoto, Farci & Lilliu (2002); Chung & Lee (2002)). The selection of forecasting methods depends on several factors, such as the forecast format required, the availability of data, the desired accuracy and the ease of operation. Although these methods are vastly applied in demand forecasting, they have some drawbacks that motivate us to turn our attention to Artificial Neural Network (ANN). Classical forecasting techniques make use of, for instance, time series models, which are described as mathematical processes that can be extended into the future (Montgomery, Johnson & Gardiner (1990)). Despite the capabilities of the time series approach in transportation forecasting, these models cannot respond rapidly to sudden changes in bookings and cancellations (Sen (1985)). The use of regression analysis for a large dataset with numerous predictors and response variables can be complicated, and time consuming, by computing inverse matrices and normal equations' calculations (Wei & Hong (2004); Anderson, Sharfi & Gholston (2006); Varagouli, Simos & Xeidakis (2005); Rengaraju & Aracon (1992)). A major drawback of exponential smoothing, another classical method of forecasting, is that it is difficult to select an optimum value for the constant without making restrictive assumptions about the behavior of the phenomenon under consideration. This problem is compounded when the form of the underlying problem is changing through time (Montgomery, Johnson & Gardiner (1990); Godfrey & Powell (2000); Widiarta, Viswanathan & Piplani (2007); Snyder, Koehler & ord (2002)). Another approach applied to estimating parameters is the Bayesian method. Although this method works accurately to define the parameters of a regression model, there is still the open problem of determining the distribution of historical data (Miltenburg & Pong (2007); Popovic & Teodorovic (1997)).

To overcome some of the drawbacks of the mentioned conventional methods, we turn our attention to a method that works more precisely in nonlinear spaces: Artificial Neural Network (ANN). The output of traditional models is the linear sum of the weighted responses, whereas in neural network, multiple linear combinations are processed in parallel; that is, the activation in each neuron is a separate linear combination. The major advantage of the neural network approach is that it is flexible enough to model complex non-linear relationships in an automated fashion (Mozolin, Thill & Usery (2000)). Moreover, the most valuable property of multilayer feed-forward neural network is their ability to approximate as accurately as desired a function from training examples. In fact, a three-layer, fully connected feed-forward neural network with n input nodes, a sufficiently large number of hidden nodes and one output node, can be trained to approximate any $n \times 1$ mapping function (Mozolin, Thill & Usery (2000); Hashemi, Le Blanc, Rucks & Shearry (1995); Celikoglu & Cigizoglu (2007)). In this paper, we apply artificial neural network in demand forecasting for transportation and we improve the model by targeting pre-processing methods using exponential distribution in data normalization, as well as applying adaptive learning, momentum, and regularization.

2 Problem definition

In this research, we investigate demand forecasting of a major railroad. The aim is to predict the number of reservations (bookings) and cancellations and, consequently, the number of passengers (bookings minus cancellations) at the time of departure. For example, in Figure 1, the number of passengers is represented for different departure times and departure days in different classes (i.e. business, economy for different types of clients: junior, senior, or VIP, etc.,). Each case is indicated by an observation on the horizontal axis. The number of passengers for a sample consisting of 100 observations is illustrated in this figure.

The number of observations is equal to the number of passengers for a specific departure date and for a particular combination of class and product. Different classes are offered by the transportation company such as economy or business class, for which the prices and capacity differ. Moreover, the products are suggested by the company and are assigned to passengers, such as junior, senior, VIP and so on. We will expect our model to accurately estimate the number of passengers considered as response variables. The model proposed consists of two parts: a pre-processing phase and a regression phase. According to the

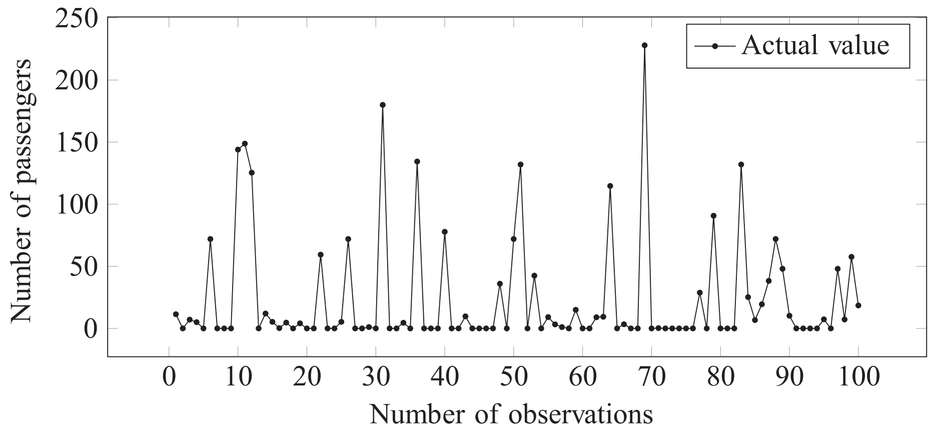


Figure 1: Sample pattern of the number of passengers at departure

transportation company's protocol, the reservation process starts 120 days before departure and there are 20 booking interval segments. During these 120 days, passengers register the information for their itinerary reservations. Some of the reservations may be cancelled throughout this period of time. There are several factors which affect the number of bookings and cancellations. We will investigate the impact of seven factors, which will serve as network inputs, on the number of passengers. This transport organization offers many departures every day at different hours. Depending on departure date and departure time the demand differs. The list of inputs and outputs is represented in Table 1.

Table 1: Table of inputs and outputs

Inputs		
Name	Data type	Inputs
Departure date	Code	$\{1,2\}$ 1: Weekday 2: Weekend
Departure time	Code	$\{1,2,3\}$ 1: [6:25 - 11:55] 2: [12:25-17:55] 3: [18:25-21:55]
Product	Code	19 types
Class	Code	14 types
Class average price	Real value	Max=117.5, Min=24.5
Itinerary average price	Real value	Max=117.5, Min=70.86
Day average price	Real value	Max=93.61, Min=78.49
Outputs		
Reservations	Positive real value	[0,340]
Cancellations	Positive real value	[0,340]

During each week there are a lot of business travelers; therefore, demand increases. During weekends there are more noticeable fluctuations in the quantity of passengers. Thus, we prefer to represent departure date in two codes: code 1 for weekdays and code 2 for weekends. Moreover, according to historical data, demand changes during the day. Therefore, the number of bookings and cancellations is affected by departure time. We express departure time using three codes. Code 1 represents the departures that are scheduled in the morning, code 2 for the afternoon and code 3 shows departures during the night. In this transportation context, tickets (i.e. products) are provided in 19 types that indicate the category of each passenger. Passengers can be students, employees, juniors, or seniors. Moreover, tickets are allocated in fourteen types of classes (i.e. business or economy). The output variables are reservations and cancellations. They vary between a minimum of zero passengers to a maximum of 330 passengers.

3 Pre-processing

The performance of the Multi-Layer Perceptrons (MLP) is directly influenced by the inputs that are fed to the network and the outputs which are used in the learning process. Therefore, an important part of the method is to deal with the data being used in the training procedure in which the parameters of the network are being fixed. Some data will have to be removed, some will be left unchanged and the rest will be transformed. This procedure is called pre-processing. As will be seen in the results, pre-processing contributes greatly in reducing the network generalization error. Inputs such as departure date, departure time, products and classes are determined by codes. That is, discrete numbers which are assigned to each category and do not follow a particular distribution. Thus, we may enter these data into the network without any transformations and they are left unchanged. Moreover, the daily prices for each class and itinerary differ. They are also used as inputs of the network and need to be pre-processed. But first, we have to detect the outliers of the outputs. Outliers impose a significant noise on the average and variance of the entire dataset. They can cause distortion in normalization and training. To discover them, we have divided the data into four distinct intervals. The reason is that the capacity of the train is limited; therefore, the number of bookings and cancellations in a single request for each itinerary is rarely more than 300 passengers. On the other hand, there are a lot of departures with fewer than 100 passengers for each available combination of class and product. Thus, for a specific combination of class, product, and price, the number of reservations and cancellations could be zero, less than 100, between 100 and 300, or more than 300. Table 2 represents the distribution of output data in these four intervals. The range shows the number of passengers in each departure.

Table 2: Table of frequencies of January 2005 for the number of bookings and cancellations

	Dataset			
	Training		Testing	
Range	Booking	Cancellation	Booking	Cancellation
$x = 0$	8907	10497	1906	2262
$0 < x < 100$	10502	9211	2248	1955
$100 < x < 300$	407	117	92	32
$x > 300$	10	1	3	0

The first column specifies the number of bookings and the second the number of cancellations. The first row shows that for a specific departure date and time, and a particular combination of class and product, in 8907 cases during January of 2005 we had zero reservations. For example, there were some cases in which there were no juniors in Business class for a specific departure date and time. According to the tables, we find that there is significant variance in both datasets, which causes distortion in the normalization and training process; it also creates noise in the performance of the network. To overcome this problem, we remove the outliers in an empirical way. As shown in the preceding table the majority of data is located in the first two intervals, motivating us to calculate the proportion of outliers for both bookings and cancellations. Table 3 represents the proportion of outliers for each set of data and for each output. These outliers are defined as the proportion of the departures with more than 100 passengers over the entire quantity of passengers.

Table 3: Proportion of outliers of output variables

	January 2005	
	(training, 80% of data)	(testing, 20% of data)
Bookings	2%	2%
Cancellations	0.50%	0.70%

The first result denotes the proportion of outliers for the number of bookings for the January 2005 training data. In this case, the ratio of outliers is 2%. The second result is the proportion of outliers for the number of cancellations, which is 0.5%. The numbers are considered negligible and the outliers can be removed.

Nonlinear behavior is brought into the neural network by activation functions. The most commonly used activation function in multilayer perceptrons is the sigmoid function. In this case, outputs are images of this function producing values between 0 and 1. Outputs greater than 1 need to be transformed in order to have a more accurate mapping; otherwise, the activation function will overweigh those features having larger values. It is preferable to fit a probability function that determines the characteristics of the data, which can also be reversed to calculate the error function. The Gaussian distribution is the most common one, used in normalization context but it does not have the property of being reversible because it is not bijective, that is, two different events may have the same probability of occurring. In our case, the output values are always positive, since the number of bookings minus cancellations is always positive. In addition, the shape of the data strictly decreases in terms of each interval; for example, there are many data equal to zero and as the sample approaches 100 (after eliminating the outliers), the number of bookings and cancellations decrease. This suggests selecting the exponential distribution to map the output values into the interval [0, 1], which is also bijective. The general formulation of the exponential distribution for variable x is:

$$f(x) = \frac{1}{\lambda} e^{-\frac{x}{\lambda}} \; x \geq 0 \tag{1}$$

The estimated value for the parameter λ of the exponential distribution is the average of the data. Hence, for each set of outputs (i.e., bookings and cancellations) we have different exponential distributions with different parameters. The exponentialized outputs will be obtained via the following equation:

$$f(y_{RealValue}) = \frac{1}{\lambda_i} e^{-\frac{y_{RealValue}}{\lambda_i}} = y_{Exponentialized} \tag{2}$$

where $y_{RealValue}$ is the actual value corresponding to the specific input and $y_{Exponentialized}$ is the output transformed by the exponential distribution that was fed into the network during the learning process; moreover, λ_i defines the parameter for each dataset. The weights are adjusted during the training process while the difference between the output of the network and the exponentialized real values are minimized. Because the learning process tries to approach the vector of the network outputs and the exponentialized real values as much as it can by using the least mean square algorithm, we suppose that the output of the network follows the same distribution as the real values. Once the predicted values have been generated, we reverse them in order to calculate the error. The absolute value of the transformed output and the corresponding error function are given by:

$$y_{Transformed} = \lambda_i \log(\lambda_i y_{NetworkOutput}) \tag{3}$$

$$Error = \vec{Y}_{RealValue} - \vec{Y}_{Transformed} \tag{4}$$

where $y_{NetworkOutput}$ is the prediction generated by the network and $y_{Transformed}$ represents the transformed predicted value. As before, λ_i is the parameter of the exponential distribution. The error function is interpreted as the difference between the vector of observed $y_{RealValue}$ and the vector of $y_{Transformed}$ produced by the network. The results will show that using an exponential distribution has a significant impact on improving the performance and generalization of the network.

4 Model

Throughout this section we consider the structure of the multilayer perceptron that will enable us to forecast the number of passengers. At first, we have to define the architecture of the network, and then choose an appropriate learning algorithm for training. Refining and fine-tuning the learning process will make it more efficient. After training, we will validate the accuracy of the network by the method of cross-validation. The results will be represented in the following section.

4.1 Architecture of the neural network

The typical network consists of an input layer, some hidden layers and an output layer. A neural network of minimum size is less likely to introduce noise into the training data and may result in better generalization. On the other hand, a large number of hidden neurons can mimic the phenomenon without understanding the underlying process. Therefore, finding a fairly convenient tradeoff between these two situations is critical. Choosing the number of hidden layers and hidden neurons is done empirically and there is no specific rule for it. A practical issue that arises in this context is that of minimizing the size of the network while maintaining good performance. In this study, we have chosen the network growing method, in which case we start with two neurons and then add progressively a new neuron or a new layer of hidden neurons. Preliminary results show that increasing the number of hidden neurons in the first layer does not reduce the error significantly. Thus, we add another hidden layer. At last, empirically we stop the growing network process at the point of two hidden layers each comprising five neurons. The final architecture of the network is illustrated in Figure 2.

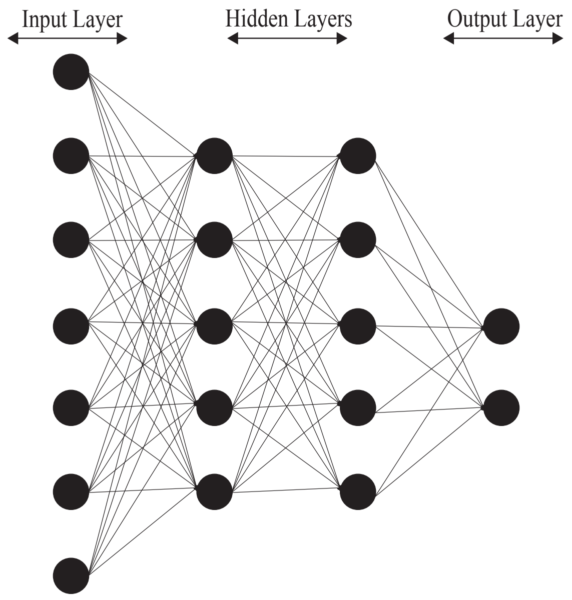


Figure 2: Final structure of the network having two hidden layers with 5 neurons each

4.2 Learning algorithm and parameter adjustments

In order to choose a learning algorithm for training and fixing the parameters we have to fully define the data set that will be fed to the network. In this study, we apply the data of January 2005 for training. To alleviate the overfitting, we use 80% of the data randomly to train the network and the remaining 20% to test the fixed parameters. This percentage is chosen empirically and it is the most common proportion for training and testing values. The neurons of each layer are connected via some coefficients, called weights, which have to be fixed during the training process. The most common learning algorithm for multilayer perceptrons, called back propagation, is applied in our case. This is an iterative process which will stop as soon as a local minimum is met with respect to a quadratic error function. After different trials we established 300 epochs to adjust the parameters during the network training. The reason is that after 300 iterations the performance of the network mostly remains constant and we cannot see any significant decrease in the error during the learning process. The learning process is maintained on an epoch-by-epoch basis until the synaptic weights of the network stabilize and the average squared error over the entire training set converges to a minimum target value. We have also chosen *batch-mode* learning, where weight updating is presented after entering all the training examples that constitute an epoch. The use of batch-mode training provides an accurate estimate of the gradient vector where convergence to a local minimum is thereby guaranteed under simple conditions (Haykin (1998)). The primary focus of regression methods is to smoothen the predicted output variable, and in neural network, this task is accomplished with the use of sigmoid functions. The sigmoid function, whose graph is S-shaped, is by far the most common form of activation function used in the construction of artificial neural network mainly because it is differentiable. It is defined as a strictly increasing function that exhibits a graceful balance between linear and nonlinear behavior. The general format of the sigmoid function is as follows,

$$\varphi(x) = \frac{1}{1+e^{-ax}} \tag{5}$$

where a is the slope parameter. When a is small, the network needs more data to be trained and when it is large, the generalization of the network is not good enough. In our study, after comparing the error of different trials we established a as being equal to 1.

4.3 Model improvements

Since back-propagation learning is basically a hill climbing technique, it runs the risk of being trapped in a local minimum where every small change in synaptic weights, w, increases the error function. The weight adjustments are done according to the following equation

$$w(n+1) = w(n) + \alpha[w(n-1)] + \eta\delta(n)y(n) \tag{6}$$

where δ represents the local gradients at each iteration n and y depicts the output of the corresponding neuron. η is the learning-rate parameter and α shows the *momentum* constant which increases the rate of learning yet avoids the danger of instability of training because the back-propagation algorithm provides an approximation to the trajectory in weight space computed by the method of steepest descent. Thus, the smaller we make the learning rate parameter η, the smaller the changes to the synaptic weights in the network will be from one iteration to the next and the smoother the trajectory will be in weight space. This improvement, however, is attained at the cost of a slower rate of learning. On the other hand, if we make η large in order to speed up the rate of learning, the resulting large changes in the synaptic weights assume such a form that the network may become unstable. Applying the momentum term helps us to avoid these problems. One technique that is often used to control the over-fitting phenomenon is that of regularization, which involves adding a penalty term to the error function in order to discourage the coefficients from reaching large values. The simplest such penalty term takes the form of a sum of squares of all of the coefficients, leading to a modified error function, E of the form

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}\|w\|^2 \tag{7}$$

where $\|w\|^2 \equiv w^T w = w_0^2 + w_1^2 + ... + w_M^2$, and t_i represents actual data. The coefficient λ governs the relative importance of the regularization term compared with the sum-of-squares error term. In order to determine the parameter of the learning ratio and modify the training process, we employ the *adaptive learning* method. The performance of the steepest descent algorithm can be improved if we allow the parameter to change during the training process. An adaptive learning rate will attempt to keep the step size as large as possible while keeping the training process stable. This parameter is made responsive to the complexity of the local error surface and it requires some changes in the training procedure. First, the initial network output and error are evaluated. At each epoch, new weights and biases are calculated using the current parameter. New outputs and errors are then established. As with momentum, if the new error exceeds the old error by more than a predefined ratio, the new weights and biases are discarded and the learning rate is decreased; otherwise, the new weights are kept. If the new error is less than the old error, then the parameter is increased. This procedure increases the learning rate, but only to the extent of learning without large error increments. Thus, a near optimal value is obtained for the local terrain (Haykin (1998)).

4.4 Validation

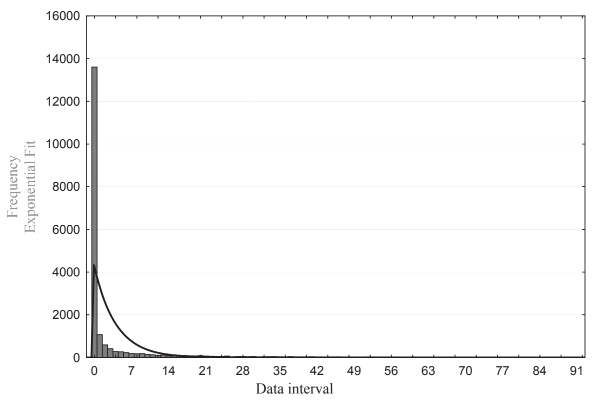
After training the network and fixing the parameters and also applying the improvement methods, we want to examine the generalization capability of the network. The motivation here is to validate the model on a different dataset than the one used for parameter estimation. Generalization is influenced by three factors: (1) the size of the training set and how representative it is of the environment of interest; (2) the architecture of the neural network; (3) the physical complexity of the problem at hand. To examine the network's generalizing ability we use cross-validation. Cross-validation, sometimes called rotation estimation, is the statistical practice of partitioning a sample of data into subsets such that the analysis is initially performed on a single subset, while the other subset(s) is (are) retained for subsequent use in confirming and validating the initial analysis. There is, however, the possibility that the model with the best-performing parameter values may end up overfitting the validation subset. In this study, we use multifold cross-validation by dividing the set into K subsets. The model is trained on all but one of the subsets and the validation error is measured by testing it on the remaining one. This procedure is repeated for a total of K trials, each time using a different subset for validation. The performance of the model is assessed by averaging the squared error under validation over all of the trials of the experiment. If K gets too small, the error estimate is pessimistically biased because of the difference in training-set size between the full-sample analysis and the cross-validation analysis. In contrast, if K is too large, it may require an excessive amount of computation since the model has to be trained K times with $1 \leq K \leq N$ where N is the number of examples. A value of 5 or 10 for K is popular for estimating the generalization error. The network is tested on an independent dataset that has not been used for training to give an unbiased estimate of the network performance. We trained the network on a randomly chosen subset of January 2005 for learning and validated the network with the data of March 2005.

5 Results

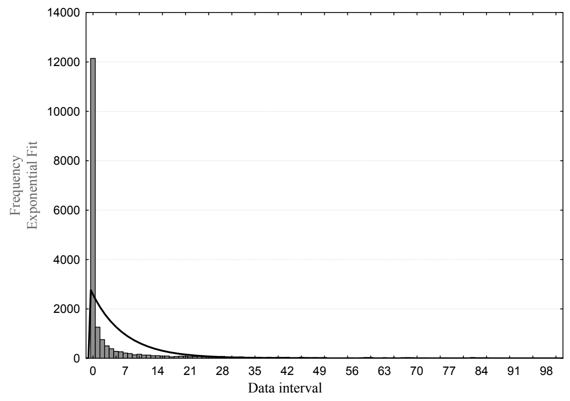
Outlier elimination was one of the improvement methods that we have applied in order to reduce the forecasting error. As mentioned before, the data should be normalized before entering the network. This process is done according to the data structure. The exponential distribution is chosen as an appropriate distribution in order to normalize the data before feeding it to the network. In Figure 3(a) and 3(b), the normalization process of the training set for both bookings and cancellations is presented. These figures show the exponential fit for the data that was used to train the network; we consider bookings and cancellations in two separate graphs. The same was done for the other two datasets.

As can be seen, the fitted curve does not cover the whole dataset. This is due to the outlier elimination procedure, which we have already implemented in the pre-processing step. Therefore, the fitted distribution does not take the outliers into account. In order to determine the architecture of the network, we start from a network with one hidden layer in which there are two hidden neurons; by increasing the number of hidden nodes, we consider the performance error of the network. As shown in Figures 4(a) and 4(b), the error is not reduced significantly when the quantity of neurons increases. The minimum error obtained by using just one hidden layer remains over 15%, which motivates us to add another hidden layer to see if we can decrease the error empirically. Preliminary results show that two hidden layers predict better than single hidden layer networks.

Figure 5 depicts the training process of the network in an experiment with a specific predefined performance goal (i.e. the predefined error is 10^{-4}). The training process starts naturally with a large error

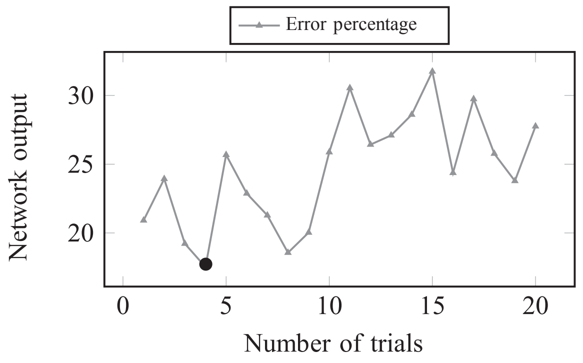


(a) Exponential distribution fit for cancellations in January 2005 (Train)

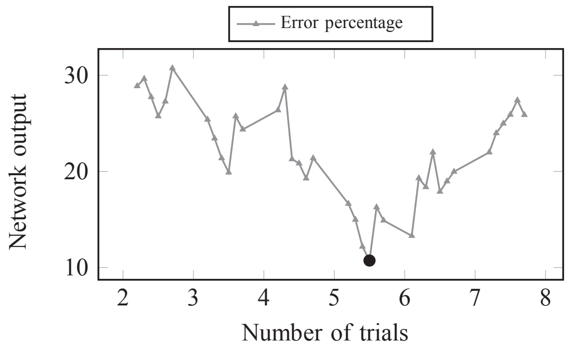


(b) Exponential distribution fit for bookings in January 2005 (Train)

Figure 3: Fitting exponential distributions to given dataset



(a) Number of hidden neurons determination for a one-hidden-layer network



(b) Number of hidden neurons determination in a two-hidden-layers network

Figure 4: Process of determining the number of hidden neurons according to the method of network growing

and, during the adjustment phase, the error decreases gradually. The training process does not reach the predefined performance error or get stuck in a local or global minimum after 300 iterations.

In this case, the predicted values are network outputs and the actual values are the numbers that were extracted from the transportation network. The preliminary results, without improvements, are shown in Figure 6. The results are clearly unsatisfactory because there are significant differences between the network outputs and the real values, making it necessary to employ some modification methods to improve the network's forecasting capability.

As we discussed earlier, some improvement methods were implemented to improve the results of the network. Before applying these methods, the error was in the 35%-45% interval, but after using exponential distribution, the results have improved dramatically giving an error rate lower than 25%. After applying the adaptive learning method to control the learning rate, the results are more stable and acceptable. However, our target error is about 8%-10%, and we are still far from this result. We used momentum and regularization to reduce the error and finally, reach our target by removing outliers. The summary of the error reduction process is represented in Table 4.

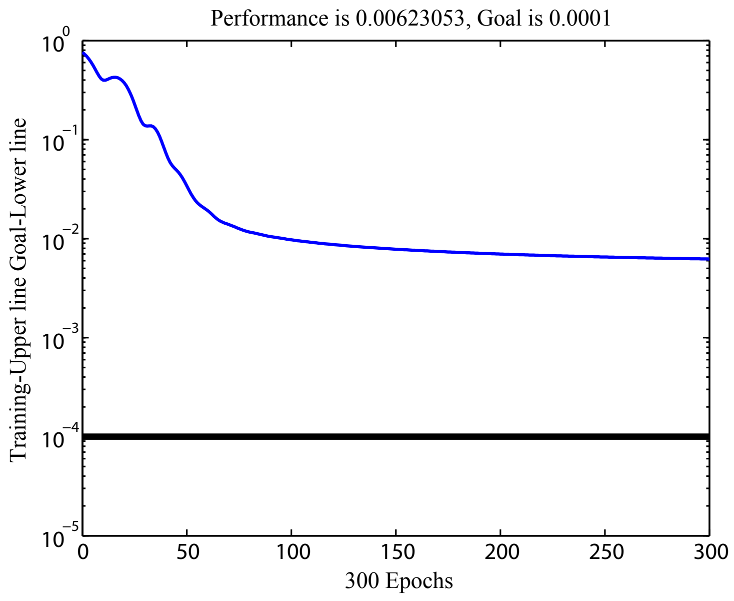


Figure 5: Network training through iterative process

Table 4: Error at each step by adding each method for improving the results

Method	Average error
Before improvement	40%
Exponential distribution	28%
adaptive learning	18%
momentum, regularization	15%
removing outliers	8%

Moreover, Figure 7 illustrates the improvements, in terms of errors, obtained by these techniques. As we can see, the residuals have been reduced significantly and the network is capable of developing almost the same format as the actual values.

Figure 8 illustrates the results obtained with the improvements. The figure shows that the network can reproduce, with accuracy, the actual data.

After developing the multi-layer perceptron and after applying the improvements, we expect that the network could generalize its ability of forecasting for unseen datasets as well. In order to validate the network, we performed a series of trial and error tests to determine how many folds give more appropriate results. To represent this analysis we have examined three different possibilities with K equal to 7, 3, and 5 folds, respectively.

As can be seen in Table 5, developing a 7-fold cross validation obtains an unrealistically low generalization error, which could cause unstable results when applied to large, new datasets. Here, we applied 86% of data to train the network and used the remaining 14% to test the generalization.

If we apply a completely new large dataset, the result will not remain the same, so we tried a 3-fold method, which is presented in Table 6.

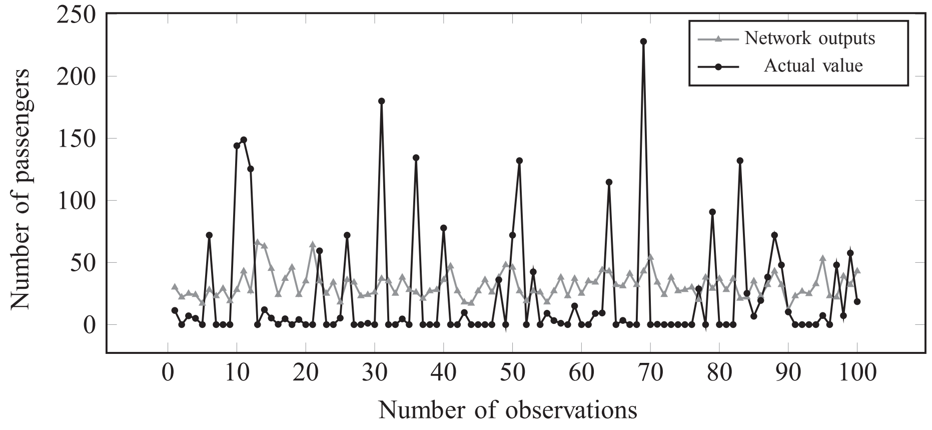


Figure 6: Prediction accuracy before imposing improvement methods

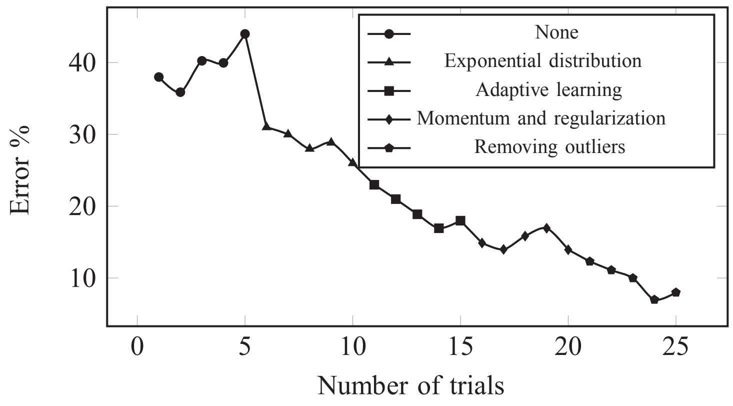


Figure 7: Improvements impact error reduction

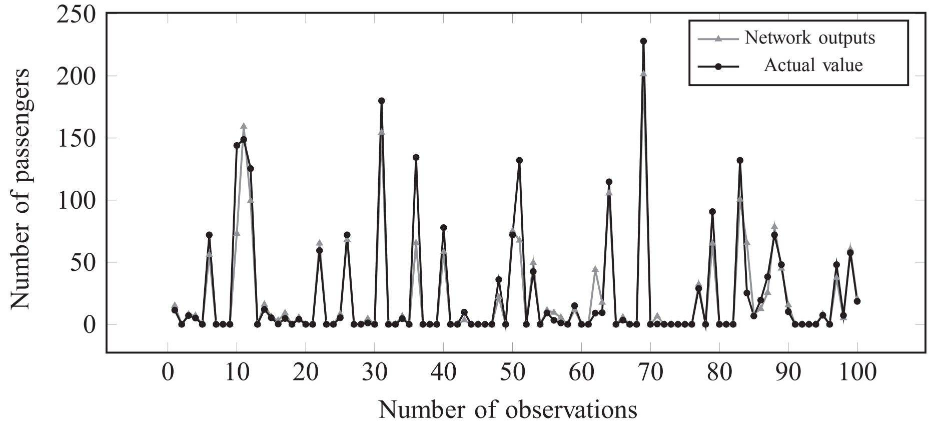


Figure 8: Prediction accuracy after imposing improvement methods

In 3-fold cross validation, the network could suffer from overfitting (i.e. while training the network, the error rate is low whereas the generalization error is high) because we use only 66% of the data to train the network. Also, the error is high because of the lack of training. Finally, we decided to choose a 5-fold method,

Table 5: 7-Fold cross validation

No. of folds	No. of incorrent estimations		Prediction error	
	Booking	Cancellation	Booking	Cancellation
1	120	143	3.55	4.24
2	132	123	3.89	3.63
3	151	134	4.46	3.96
4	92	170	2.72	5.02
5	180	127	5.32	3.75
6	141	172	4.17	5.08
7	201	172	5.94	5.08
Average			4.29	4.53

Table 6: 3-Fold cross validation

No. of folds	No. of incorrent estimations		Prediction error	
	Booking	Cancellation	Booking	Cancellation
1	1027	982	13.02	12.45
2	628	826	7.96	10.47
3	923	889	11.7	11.27
Average			10.89	11.39

in which we extract 80% of data randomly to train the network and use the remaining 20% to test it. We repeated this method five times and then we calculated the average value of the runs. The results are a good representation of the generalization error of the network. Table 7 shows the results of this experiment.

Table 7: 5-Fold cross validation

No. of folds	No. of incorrent estimations		Prediction error	
	Booking	Cancellation	Booking	Cancellation
1	364	314	8.7626	7.559
2	364	273	8.7626	6.572
3	439	695	10.5681	16.7309
4	367	283	8.8349	6.8127
5	376	272	9.0515	6.5479
Average			9.19594	8.8445

The generalization error for booking is 9.19% and for cancellation is 8.84%. As the second method of evaluating the generalization of the network, we used the dataset of March 2005 that had not been used in the training process. As illustrated in Figure 9, for 20 repetitions, the error is always steady in the 6%-12% interval. The fluctuations in the graph are due to the different subsets of the whole dataset that we have applied randomly. As shown in the graph, the average generalization error is around 8%.

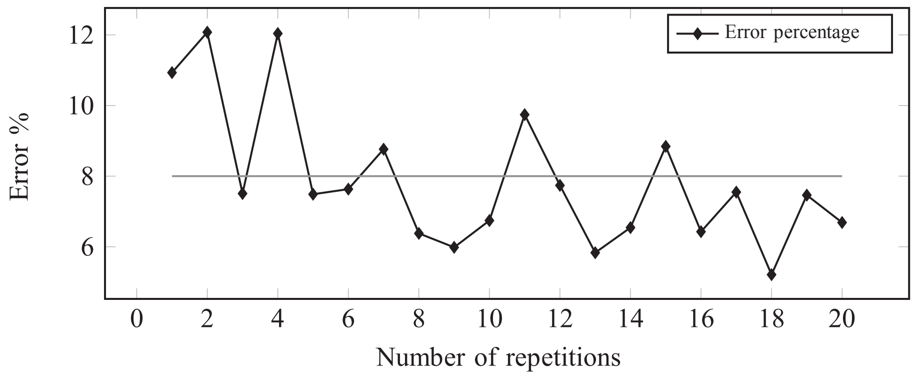


Figure 9: Cross validation with March 2005 data

6 Conclusion

Reliable passenger forecasting models play a crucial role in the transportation industry. For example, they help the transport organizations to determine seat availabilities, verify the quantity of crew members at each itinerary, and plan price settings. In this study, we have proposed a neural network to be used in transportation demand forecasting. Classical methods of statistics, such as regression or time series, struggle to cope with high dimensional data sets and sometimes refuse to respond accurately to sudden changes. In our proposed model we have chosen a Multi-Layer Perceptron (MLP) to circumvent the drawbacks of classical models. A neural network is more flexible when dealing with sudden changes in the format of data, missing information, and high dimensional data sets. We have opted to improve a typical MLP by using our knowledge of the transportation problem. This knowledge has helped us to accurately eliminate the outliers without losing too much information. Moreover, our understanding of the transportation problem has motivated us to apply exponential distribution in the process of data preparation, which reduced the forecasting error significantly. In addition, we have applied more technical approaches to improve the network. The efficiency of our model has been validated throughout this study. The results have shown a forecast error of around 8%, which is considered quite acceptable.

References

Anderson, M. D., Sharfi, K., & Gholston, S. E. (2006). Direct demand forecasting model for small urban communities using multiple linear regression. *Transportation Research Record*, (1981), 114–117.

Celikoglu, H. B. & Cigizoglu, H. K. (2007). Public transportation trip flow modeling with generalized regression neural networks. *Advances in Engineering Software*, *38*(2), 71–9.

Chung, J.-H. & Lee, D. (2002). Structural model of automobile demand in korea. *Transportation Research Record*, (1807), 87–91.

Devoto, R., Farci, C., & Lilliu, F. (2002). Analysis and forecast of air transport demand in sardinia's airports as a function of tourism variables. Advances in Transport, pp. 699–709, Seville, Spain: WITPress.

Godfrey, G. A. & Powell, W. B. (2000). Adaptive estimation of daily demands with complex calendar effects for freight transportation. *Transportation Research, Part B (Methodological)*, *34B*(6), 451–69.

Hashemi, R. R., Le Blanc, L. A., Rucks, C. T., & Shearry, A. (1995). A neural network for transportation safety modeling. *Expert Systems with Applications*, *9*(3), 247–56.

Haykin, S. (1998). *Neural networks: a comprehensive foundation.* Prentice Hall.

Miltenburg, J. & Pong, H. C. (2007). Order quantities for style goods with two order opportunities and bayesian updating of demand. part i: No capacity constraints. *International Journal of Production Research*, *45*(7), 1643–1663.

Montgomery, D. C., Johnson, L. A., & Gardiner, J. S. (1990). Forecasting and time series analysis.

Mozolin, M., Thill, J. C., & Usery, E. L. (2000). Trip distribution forecasting with multilayer perceptron neural networks: a critical evaluation. *Transportation Research, Part B (Methodological)*, *34B*(1), 53–73.

Popovic, J. & Teodorovic, D. (1997). An adaptive method for generating demand inputs to airline seat inventory control models. *Transportation Research, Part B (Methodological)*, *31B*(2), 159–75.

Rengaraju, V. R. & Aracon, V. T. (1992). Modeling for air travel demand. *Journal of Transportation Engineering*, *118*(3), 371–380.

Sen, A. (1985). Examining air travel demand using time series data. *Journal of Transportation Engineering*, *111*(2), 155–161.

Snyder, R. D., Koehler, A. B., & Ord, J. K. (2002). Forecasting for inventory control with exponential smoothing. *International Journal of Forecasting*, *18*(1), 5–18.

Varagouli, E. G., Simos, T. E., & Xeidakis, G. S. (2005). Fitting a multiple regression line to travel demand forecasting: The case of the prefecture of xanthi, northern greece. *Mathematical and Computer Modelling*, *42*(7-8), 817–36.

Wei, F. & Hong, C. (2004). Cluster model for flight demand forecasting. vol. Vol.4 of *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788)*, pp. 3170–3, Piscataway, NJ, USA: IEEE.

Widiarta, H., Viswanathan, S., & Piplani, R. (2007). On the effectiveness of top-down strategy for forecasting autoregressive demands. *Naval Research Logistics*, *54*(2), 176–88.

Xiaolong, Z. (2007). Inventory control under temporal demand heteroscedasticity. *European Journal of Operational Research*, *182*(1), 127–44.