Knowledge-Based Systems 29 (2012) 44–53

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Neural network demand models and evolutionary optimisers for dynamic pricing

S. Shakya [a,*], M. Kern [a], G. Owusu [a], C.M. Chin [b]

[a] Business Modelling & Operational Transformation Practice, BT Innovate & Design, Ipswich IP5 3RE, UK
[b] Core Design Team, BT Innovate & Design, Ipswich IP5 3RE, UK

ARTICLE INFO

Article history:
Available online 18 July 2011

Keywords:
Dynamic pricing
Evolutionary computation
Neural networks
Price optimisation
Revenue management

ABSTRACT

Dynamic pricing is a pricing strategy where price for the product changes according to the expected demand for it. Some work on using neural network for dynamic pricing have been previously reported, such as for forecasting the demand and modelling consumer choices. However, little work has been done in using them for optimising pricing policies. In this paper, we describe how neural networks and evolutionary algorithms can be combined together to optimise pricing policies. Particularly, we build a neural network based demand model and use evolutionary algorithms to optimise policy over build model. There are two key benefits of this approach. Use of neural network makes it flexible enough to model a range of different demand scenarios occurring within different products and services, and the use of evolutionary algorithm makes it versatile enough to solve very complex models. We also evaluate the pricing policies found by neural network based model to that found by other widely used demand models. Our results show that proposed model is more consistent, adapts well in a range of different scenarios, and in general, finds more accurate pricing policy than other three compared models.

1. Introduction

Pricing is one of the key decisions that a firm needs to make in order to survive in a competitive marketplace. If done carefully, it can be a valuable tool to achieve number of different business goals, such as profit maximisation, demand management, value creation, etc. Dynamic pricing [28,15,5] is a pricing strategy where a firm adjust the price for their products and services as a function of its perceived demand at different times. In other words, dynamic pricing is to sell the product to the right customers, at the right time, with the right price, in order to maximise the profit. Traditionally, it has been applied in service industries, such as airlines, hotels and rentals [16]. For example, in airlines, the price for a seat changes according to the time remaining for the flight and according to the number of available seats. Recent developments in information technology and eCommerce have led the dynamic pricing to spread over wide range of other industries such as retails [13,7,2], wholesales [19] and auctions [22].

The key idea in dynamic pricing is to model the effect of interaction between number of different factors, such as price for the product in different times, have on demand for the product, and use that model (known as demand model [28] or price-response model [19]) to optimise the pricing policy. These demand models are mainly derived from the historical data about prices and demands for products in past. In practice, various assumptions are made on how these factors interact, resulting in different types of demand models. For example, assuming price and demand are linearly dependent to each other a linear demand model can be derived, similarly assuming a non-linear dependency a non-linear model can be derived.

In this paper, we purpose to use a neural network [30,1] based demand model for dynamic pricing. Neural networks are inspired by the way biological nervous systems, such as brains, process the information. Unlike traditional demand models, they do not make any advance assumptions about the relationship between different factors. Rather, they *learn* these relationships from the data itself. They are able to derive meaning from complicated or imprecise data and can be used to model the relationships that are too complex to be noticed by humans or computer techniques. This ability of neural networks makes them a good candidate for modelling demand in dynamic pricing.

Some work on using neural network for dynamic pricing have been previously reported, such as for forecasting the demand [21] and modelling consumer choices [31,9]. However, little work has been done in using them for optimising the pricing policies. Also, traditionally, numerical techniques, such as mathematical programming, have been used for optimising pricing policies. These techniques usually require the gradient information about the objective function. However, a neural network based objective function may not be well defined, and therefore the gradient information may not be expressed explicitly. For example in [29], Neural network is combined together with Dynamic programming [4]

* Corresponding author. Tel.: +44 01473605835.
E-mail addresses: sid.shakya@bt.com (S. Shakya), mathias.kern@bt.com (M. Kern), gilbert.owusu@bt.com (G. Owusu), eric.chin@bt.com (C.M. Chin).

0950-7051/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.06.023

to optimise the road tolls. The idea is to dynamically change the toll based on various different factors such as time of the day and the traffic volumes. Although, dynamic programming is a robust method, its computational complexity increases significantly as the parameter in the model increases. Also, it requires the pricing model to be well specified in order to calculate the gradients. This requirement may result in the use of simplified assumptions on the relationship between different factors, which may not capture the true interactions. In such scenarios, these traditional techniques may not give a good solution.

We propose to use evolutionary algorithms (EA) [8] as a technique for optimising dynamic pricing problem based on neural network demand model. EAs are a population based optimisation technique and are inspired by Darwin's theory of natural evolution. They use the concept of natural selection and random variation to *evolve* better solution to the problem. In contrast to gradient based optimisation techniques, EA does not depend on the function form of the model and can optimise over range of different complex models.

The technique to combining neural networks with other AI methods is not new and has been exploited for solving wide range of real world problems in various domains. For example, [12] combines neural networks, fuzzy systems and genetic algorithms together to build an online gaming improvement model that continuously modify the multiplayer online gaming environment. Similarly [6], use Neural networks to forecast the future sales of a printed circuit board factory. The idea is to combine neural network with a fuzzy clustering method known as *k*-means clustering. This method clusters the historical data into relevant clusters, which can then be used to effectively train the neural network. The network is then used for forecasting. We also take similar hybrid approach specifically tailored for dynamic pricing problem.

The rest of the paper is organised as follows. Section 2 presents a mathematical model of dynamic pricing and shows how it can be formulated as an optimisation problem. Section 3 describes some of the popular demand models used in practice, and shows how these models are fitted to the data to estimate their parameters. Section 4 describes in detail the proposed demand model based on neural networks. Section 5 describes how we use EA to solve the dynamic pricing problems. It also describes several EAs that have been tested for this problem. Section 6 presents the experimental results comparing the pricing policies found by the proposed model to that found by other popular models used in practice. It also presents the analysis of the results. Finally, Section 7 concludes the paper by summarising key findings and defining future works.

2. Dynamic pricing model

The dynamic pricing model presented in this section is adopted from [26]. Following are the notations used in this section.

N	number of periods in the planning horizon
t	any given period in the planning horizon
Q_t	number of production (sales) at period t
P_t	average price of a product at period t
C_t	cost of producing one extra product at period t
Π	Total profit during the entire planning horizon

The total profit, (Π), earned from a product during the planning horizon can be modelled as

$$\Pi = \sum_{t=1}^{N} (P_t Q_t - C_t Q_t), \tag{1}$$

where Q_t is the total sales (or the production) of the product (which is equal to, or less than, the demand for the product) in period t, $P_t Q_t$ is the total revenue at period t, and $C_t Q_t$ is the variable cost at t.

Now let us define some additional constraints a firm needs to impose when defining its policy for pricing a given product (service). Following are two of the most common constraints.

a. Capacity constraints – These are number of products that can be produced in a given period and regulates the resources such as number of workers or machines that should be used in a given period. Available capacity has the lower bound and upper bound. For all $t = 1,\ldots,N$

$$\begin{aligned} M_t &\leqslant Q_t - \text{Lower bound for the capacity constraint},\\ K_t &\geqslant Q_t - \text{Upper bound for the capacity constraint}. \end{aligned} \tag{2}$$

b. Price constraints – These are the prices for the product produced in a given period and regulate the value to the costumers. Low price may suggest low value so there should also be the thresholds (lower and upper bounds). For all $t = 1,\ldots,N$

$$\begin{aligned} \underline{P}_t &\leqslant P_t - \text{Lower bound for the price cap},\\ \overline{P}_t &\geqslant P_t - \text{Upper bound for the price cap}. \end{aligned} \tag{3}$$

As stated earlier, another important component of the dynamic pricing is the demand model, which models the effect that the interaction between different factors have on demand (or production/sales) for the product. The most important factors that influence demand are the prices for the product. More precisely, the demand for a product in a period depends on the price for that product in that period and also the prices for the product in other periods in the planning horizon. Therefore, we write demand in any period t as the function of prices in all the periods as

$$Q_t = \psi_t(P_1, P_2, \ldots, P_N), \tag{4}$$

where $\psi_t(\cdot)$ is the demand function for period t. Depending on different scenarios, $\psi_t(\cdot)$ can have different functional forms. We discuss this in detail in next section.

Substituting Q_t from (4) to (1), we get profit as the function of prices, P_t, (usually variable cost, C_t, is known in advance) which can be written as

$$\Pi = \sum_{t=1}^{N} [\psi_t(P_1, P_2, \ldots, P_N)(P_t - C_t)]. \tag{5}$$

Therefore, the general formulation for the dynamic pricing can be written in terms of optimisation problem as,

$$\begin{aligned} \max_{P_1,P_2,\ldots,P_N} \quad & \Pi = \sum_{t=1}^{N} [\psi_t(P_1, P_2, \ldots, P_N)(P_t - C_t)],\\ \text{subject to} \quad & M_t \leqslant \psi_t(P_1, P_2, \ldots, P_N) \leqslant K_t, \quad t = 1, \ldots, N,\\ \text{and} \quad & \underline{P}_t \leqslant P_t \leqslant \overline{P_t}, \quad t = 1, \ldots, N. \end{aligned} \tag{6}$$

Here, the goal is to find optimum pricing policy, (P_1,P_2,…,P_N), that maximise total profit Π, subject to constraints in (2) and (3).

3. Model of demand

Depending upon the assumptions made to the demand price relationship, $\psi_t(\cdot)$ can have number of different functional forms. Following are the three of the most widely used demand (price) models.

3.1. Linear model

Linear model is one of the most popular demand models in price optimisation. They assume that the price is linearly dependent to the production. This can be written, for all $t = 1,\ldots,N$ as

$$Q_t = \psi_t(P_1, P_2, \ldots, P_N) = a_t + \sum_{j=1}^{N} b_{jt} P_j, \tag{7}$$

where a_t is the intercept of the linear model representing the customer base (total customers willing to buy the product at period t), and b_{jt} are the parameters known as slopes which represent the impact of price at time j have on the demand at time t. Note that, in general, the parameter b_{tt} is negative, since higher price for the product in a period is likely to decrease the demand for that product in that period.

From (6) and (7), we get a formulation of dynamic pricing as an optimisation problem with linear demand model as.

$$\begin{aligned} &\max_{P_1,P_2,\ldots,P_N} \quad \Pi = \sum_{t=1}^{N}\left[\left(a_t + \sum_{j=1}^{N} b_{jt}P_j\right)(P_t - C_t)\right], \\ &\text{subject to} \qquad M_t \leqslant a_t + \sum_{j=1}^{N} b_{jt}P_j \leqslant K_t, \quad t = 1,\ldots,N, \\ &\text{and} \qquad \underline{P_t} \leqslant P_t \leqslant \overline{P_t}, \quad t = 1,\ldots,N. \end{aligned} \tag{8}$$

3.2. Exponential model

Exponential model (also know as log-liner model) assume that the relationship between price and demand is exponential. The model is very similar to the linear model which can be written for all $t = 1,\ldots,N$ as

$$Q_t = \psi_t(P_1, P_2, \ldots, P_N) = e^{a_t + \sum_{j=1}^{N} b_{jt}P_j}. \tag{9}$$

Here, a_t and b_{jt} are the parameters similar to the linear model representing the impact of the price on the production.

From (6) and (9), we get formulation of the dynamic pricing as an optimisation problem with exponential demand model as.

$$\begin{aligned} &\max_{P_1,P_2,\ldots,P_N} \quad \Pi = \sum_{t=1}^{N}\left[\left(e^{a_t + \sum_{j=1}^{N} b_{jt}P_j}\right)(P_t - C_t)\right], \\ &\text{subject to} \qquad M_t \leqslant e^{a_t + \sum_{j=1}^{N} b_{jt}P_j} \leqslant K_t, \quad t = 1,\ldots,N, \\ &\text{and} \qquad \underline{P_t} \leqslant P_t \leqslant \overline{P_t}, \quad t = 1,\ldots,N, \end{aligned} \tag{10}$$

3.3. Multinomial logit model

Multinomial logit is one of the most popular demand model used in practice. Its popularity lies in the fact that it can explicitly model the *consumer's choice*, i.e., explicitly estimate the probability of the consumers choosing to buy the product in a particular period given the prices for all the periods [28]. This provides some extra information for the practitioners. Multinomial-logit demand model can be written for all $t = 1,\ldots,N$ as

$$Q_t = \psi_t(P_1, P_2, \ldots, P_N) = B\frac{e^{-b_t P_t}}{1 + \sum_{j=1}^{N} e^{-b_j P_j}}. \tag{11}$$

Here, B is the customer base and b_j are the parameters of the model representing the impact of price at time j on demand. From (11), a multinomial probability can be separated as

$$p_t(P_1, P_2, \ldots, P_N) = \frac{e^{-b_t P_t}}{1 + \sum_{j=1}^{N} e^{-b_j P_j}}. \tag{12}$$

This is the probability that a customer choose to buy the product in period t, given the prices for the product in all the periods.

From (6) and (11), we get formulation of the dynamic pricing as an optimisation problem with multinomial logit demand model as.

$$\begin{aligned} &\max_{P_1,P_2,\ldots,P_N} \quad \Pi = \sum_{t=1}^{N}\left[\left(B\frac{e^{-b_t P_t}}{1 + \sum_{j=1}^{N} e^{-b_j P_j}}\right)(P_t - C_t)\right], \\ &\text{subject to} \qquad M_t \leqslant B\frac{e^{-b_t P_t}}{1 + \sum_{j=1}^{N} e^{-b_j P_j}} \leqslant K_t \quad t = 1,\ldots,N, \\ &\text{and} \qquad \underline{P_t} \leqslant P_t \leqslant \overline{P_t} \quad t = 1,\ldots,N. \end{aligned} \tag{13}$$

3.4. Estimating parameters of the demand model

In order to solve the dynamic pricing optimisation problems, the parameters of the demand model have to be given. Sometimes, they are given in advance by the market experts. However, most of the time, these parameters are not known in advance and have to be estimated. This estimation is mainly done by using the historical price-demand data for the product. In particular, given the data about past prices for the product and the corresponding sales, a demand model is fitted to the data to estimate the model parameters. The estimated model parameters are then used for optimising the pricing. For example, in the case of linear model (7) (or exponential model (9)), a linear regression can be done to fit the model to the historical price-demand data and model parameters, a_t and b_{jt}, are estimated. In case of multinomial-logit models, a nonlinear curve fitting algorithm or a maximum likelihood method could be used. The estimated model parameters are then passed as an input to the optimisation problem (8) and (10) or (13), which is then solved to get the profit maximising pricing policies.

4. Neural network demand models

All three demand models described in previous section requires some assumption on the functional form, such as linearity and non-linearity, of the data to be made in advance. However, in a real world scenario, such an assumption may not hold and the model may not correctly represent the true distribution of the data. Opti-

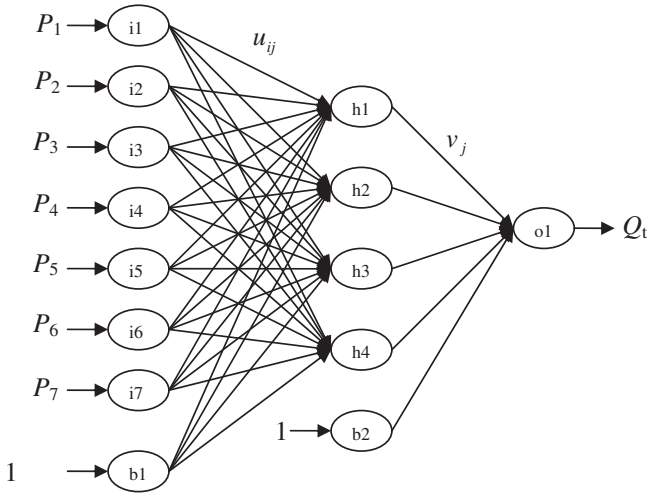

Fig. 1. The structure of a neural network demand model with seven inputs and a single output. This models the scenario when there are 7 periods in the planning horizon.

mizing pricing policy with such demand model could lead to a poor quality solution.

We purpose to use neural networks as the demand model for optimizing pricing policies. The key motivation for doing so is due to the fact that neural networks do not make any assumption about the data. Instead, they try to learn the functional form of the true model from the data itself. In this section we describe the way we use neural networks to model the price-demand relationship.

A neural network can be defined in terms of its topology and its weights. Network topology is the collection of nodes and the ways they are connected with each other. These nodes are grouped into several layers, where each layer consist of one or mode nodes. Normally, each node in a layer is connected to all the nodes in the subsequent layer. Network weights are the collection of values defining the strength of each of these connections.

Depending upon the number of layers and the ways they are connected to each other, wide range of different neural network topology can be defined. For the purpose of our work we use a fixed topology network with three layers, an input layer, a hidden layer, and an output layer. Nodes in the input layer define the inputs to the model; nodes in the output layer define the outputs of the model and the nodes in the hidden layer helps to correctly define the relationships between inputs and outputs. We build a set of N neural networks, each modelling the demand price relationship for a period.[1] In particular, each neural network represent the Eq. (4), which consist of N input nodes defining prices $(P_1, P_2, \ldots, P_N)$ and a single output node defining the production, Q_t. The number of hidden nodes, M, was defined as the half of sum of input and output nodes, $M = (N + 1)/2$. Fig. 1, shows the implemented structure of the neural network, assuming planning horizon with seven periods. Notice that the *bias nodes* (which have the fixed input value of 1, and help to fit the model to the data more accurately) are added to input layer and the hidden layer.

The parameters of the neural network are the weights associated with the connections. We use u_{ij} to represent the weight between input node i and hidden node j. Also, we use u_{bj} to represent the weights between the bias node in input layer and hidden node j. Similarly, we use v_j to represent the weight between hidden node j and the output node. Also, we use v_b to represent the weight between bias node in hidden layer and the output node. Given all the inputs and the weights, the output of the neural network can be calculated as

$$Q_t = \alpha\left(\sum_{j=1}^{M}\left(\alpha\left(\sum_{i=1}^{N} P_i u_{ij} + u_{bj}\right)v_j\right) + v_b\right). \tag{14}$$

Here, $\alpha(x)$ is known as the *activation function* of the neural network, which we choose to be of a *sigmoid* form given by

$$\alpha(x) = \frac{1}{1 + e^{-x}}. \tag{15}$$

Also, since we use sigmoid activation function, the value for the output of (14) will be between 0 and 1. Furthermore, it also requires the inputs P_i to be mapped to the 0 and 1. We therefore use P'_t and Q'_t to denote P_t and Q_t that are mapped to the values between 0 and 1. Here we do a linear mapping of P_t, which is given by

$$P'_t = mapped(P_t) = P_t^{\min} + \frac{P_t}{P_t^{\max} - P_t^{\min}}, \tag{16}$$

where $P_t^{\min}$ is the minimum value which we set to 0 and $P_t^{\max}$ is the maximum value which we set to $2\overline{P_t}$. From (14) and (15) demand function for a period t can be written in terms of a neural network with sigmoid activation function as

$$Q'_t = \frac{1}{1 + e^{-\left(\sum_{j=1}^{M}\left(\frac{1}{1+e^{-\left(\sum_{i=1}^{N} P'_i u_{ij} + u_{bj}\right)}}v_j\right) + v_b\right)}}. \tag{17}$$

Note that the output, Q'_t will be also between 0 and 1 and therefore has to be mapped back to actual Q_t which is obtained using following linear un-mapping procedure

$$\begin{aligned} Q_t &= \psi_t(P_1, P_2, \ldots, P_N) = unmapped(Q'_t) \\ &= Q_t^{\min} + Q'_t\left(Q_t^{\max} - Q_t^{\min}\right) \end{aligned} \tag{18}$$

where we $Q_t^{\min}$ is the minimum value which we set to 0 and $Q_t^{\max}$ is the maximum value which we set to $2K_t$. From (6) and (18), we get formulation of the dynamic pricing as an optimisation problem with neural network demand model as

$$\begin{aligned} \max_{P_1, P_2, \ldots, P_N} \quad & \Pi = \sum_{t=1}^{N} unmapped(Q'_t)(P_t - C_t), \\ \text{subject to} \quad & M_t \leqslant unmapped(Q'_t) \leqslant K_t \quad t = 1, \ldots, N, \\ \text{and} \quad & \underline{P_t} \leqslant P_t \leqslant \overline{P_t} \quad t = 1, \ldots, N. \end{aligned} \tag{19}$$

Estimating parameters of the neural networks

Now that we have built the neural network based demand model, the next step is to estimate the parameters of the model which could then be passed to Eq. (19) for optimization of the pricing policy. For the purpose of this paper, we use a well known neural network parameter estimation technique known as *back propagation*. It is a variant of *gradient decent* algorithm that iteratively improves the weights of the neural network to fits the data. The basic idea here is to minimize the squared error between the value predicted by the model and the actual value given by the data. It starts by randomly generating a set of values as weights for current model. By using prices from the data as the input to the current model, it then estimates the production as the output of the model. It then looks on the squared difference between the output and the actual production given by the data. If the difference is greater than the given threshold, or if the number of allowed iteration is not passed, it slightly perturbs the weight vector such that the difference is further minimized. This requires gradient information about the mean square function to be estimated. This process continues until any of the termination criteria is met. The final weight vector is given as the output of the algorithm. We do not go into further detail on the workflow of the back propagation algorithm. Interested readers are suggested to see [30,1].

5. Building EA based dynamic pricing solver

As we have mentioned earlier, the three demand models, presented in Section 3, are widely used by dynamic pricing community for optimising pricing policies, where the optimisation is usually done by means of numerical techniques, such as mathematical programming. For example, the optimisation problem with linear demand model defined by (8) (and with exponential model defined by (10)) can be seen as a constrained quadratic optimisation problem and can be solved by using a quadratic programming technique. Also, the optimisation problem with multinomial-logit demand model defined by (13) can be seen as nonlinear constrained optimisation problem and can be solved by using a nonlinear programming technique. However, the optimisation problem with neural network may not be well defined and therefore the mathematical programming may not be used. Let us describe the use of neural network for dynamic pricing in more detail.

[1] Although a single neural network with N output could also be defined instead of set of N single networks. However, for simplicity, and also since rest of the model used have N separate functions for each period, we choose to build N single output neural networks.

Some works have been reported on using neural network for forecasting the demand or modelling the consumer's choice. However, to the best of our knowledge, using them as the demand model for optimising pricing policy, as shown in Eq. (19), has not been reported. There could be several reasons for this. Firstly, neural networks are traditionally developed within the artificial intelligence community; whereas dynamic pricing is traditionally a discipline within operation research community. Secondly, the traditionally used mathematical programming techniques for optimising pricing policy may not give good results when applying on Eq. (19), mainly because of the complication involved in finding the gradient information. These make mathematical programming not a desirable option to solve the neural network based pricing optimisation problem. Finally, the result obtained with a mathematical programming is more likely to be a local optimal solution for the problem and may not be a global optimal solution.

In this paper, we propose to use evolutionary algorithms for solving neural network based dynamic pricing problem. Since evolutionary algorithms treat the objective function in the optimisation problem as a black box, they do not depend on the functional form of the problem and do not require the gradient information. Also, since they are population based optimisation techniques, they have the ability to simultaneously search over a large space of potential solutions, and are likely to give the solution closer to the global optimum. These properties of evolutionary algorithm make them a good candidate for solving the neural network based dynamic pricing problem. Moreover, since they treat the functional form of the problem as the black box, they can also be used to solve the dynamic pricing problem based on other traditional demand models, namely (8), (10), and (13). Next we describe the general workflow of an EA, and show how we use EA to optimise pricing policies with the proposed neural network demand model.

5.1. EA workflow

Typically, an EA starts by randomly generating a population of solutions. In our case the population is a set of pricing policies. Each policy is then evaluated by passing the prices in the policy to the neural network (17) to get the total profit. A subset of good policies from the population is then selected and is used to generate new population (known as child population). Different EAs, use different techniques to generate new policies from the selected set. For example, Genetic algorithms [8], a well know EA, use *crossover* and *mutation* approach to generating next population. A typical crossover operator randomly selects two policies from the subset of best policies, and exchanges some pricing between them. The aim is to recombine good solution to create even better solution. Similarly, mutation operator slightly modifies some part of the policies with a very small probability. The idea is to introduce a small variation in the child population in order to allow the search to progress towards unexplored part of the solution space. The created child population replaces old population and the next set of selection, crossover and mutation operators executes. This process continues until a termination criterion is met. A typical termination criterion is to terminate after a specified number of iteration is done, or to terminate after the population of policies converged to a specific region of the search space.

5.2. Constraint handling in EA

The neural network based dynamic pricing problem,[2] Eq. (19), can be seen as a *constraint nonlinear optimisation problem*. A general constrained optimization problem can be defined as $\max_x f(x)$, $x \in S \subset \Re^n$, subject to the linear or nonlinear constraints $g_i(x) \leqslant 0$, $i = 1,\ldots,m$. Here m is the total number of constraints. Since problem definition in EAs, generally, does not consider the constraint part, they can be thought of as unconstrained optimization algorithms [17]. In order to apply EA for the dynamic pricing model developed in previous section, we first need to find the way to handle constraints in EA.

One of the most popular ways to solving constrained optimization problems is by using a penalty function. The idea is to construct a penalty function that penalizes the original objective function for violating the constraints in the model. In order to avoid the penalty, the algorithm tries to focus its search to more of the feasible part of the search space. Let us describe one such technique adopted from [17] for our problem. We define the penalty function as (20).

$$F(x) = f(x) - h(k)H(x), \quad x \in S \subset \Re^n, \tag{20}$$

where $f(x)$ is the original objective function. In our case it is defined by profit, Π, in the general dynamic pricing problem formulations (6). $h(k)$ is a dynamically modified penalty value, where k is the algorithm's current iteration number; and $H(x)$ is a penalty factor, defined as (21).

$$H(x) = \sum_{i=1}^{m} \theta(q_i(x)) q_i(x)^{\gamma(q_i(x))}. \tag{21}$$

Here, function $q_i(x)$ is a relative violated function of the constraints defined as (22).

$$q_i(x) = \max\{0, g_i(x)\}, \quad i = 1,\ldots,m, \tag{22}$$

$\theta(q_i(x))$ is known as multi-stage assignment function; $\gamma(q_i(x))$ is the power of the penalty function and $g_i(x)$ are the constraints. In our case, $g_i(x) \leqslant 0$ are the constraints defined by Eqs. (2) and (3) and can be re-written as following four sets, for all $t = 1,\ldots,N$,

$$M_t - Q_t \leqslant 0, \quad Q_t - K_t \leqslant 0, \quad \underline{P_t} - P_t \leqslant 0, \quad P_t - \overline{P}_t \leqslant 0. \tag{23}$$

The functions $h(k)$, $\theta(q_i(x))$ and $\gamma(q_i(x))$ are problem dependent function. For the purpose of our work we set $h(k) = 2\sqrt{k}$; Also we set, $\theta(q_i(x)) = 10{,}000$ if $q_i(x) < 0.001$, else $\theta(q_i(x)) = 15{,}000$ if $q_i(x) < 0.1$, else $\theta(q_i(x)) = 20{,}000$ if $q_i(x) < 1$, else $\theta(q_i(x)) = 30000$. Furthermore, we set $\gamma(q_i(x)) = 1$ if $q_i(x) < 1$, otherwise we set $\gamma(q_i(x)) = 2$.

5.3. Solution representation in EA

A solution, x, is represented as a set$P = \{P_1, P_2, \ldots, P_N\}$, where each P_t is represented by a bit-string of length l. The total length of a bit-string solution, $x = \{x_1, x_2, \ldots, x_n\}$, where $x_i \in \{0,1\}$, is therefore, equal to $n = l \times N$. The goal of an algorithm is to maximize the penalty function defined in (20).

We can write the equation to decode the l bit to a P_t ranging between $\underline{P_t}$ to $\overline{P_t}$ as

$$P_t = P_t + \left[\frac{decoded\_l\_bit\_P_t}{2^l} \times (\overline{P_t} - P_t)\right], \tag{24}$$

where *decoded_l_bit_*P_t is the decimal decoded version of the l bit representing P_t.

5.4. Overview of the used EAs

As we have stated earlier, number of different EAs can be derived depending upon how they generate the child population of solution. While genetic algorithms use crossover and mutation for this purpose, estimation of distribution algorithms (EDAs) [11,14] use probabilistic approach. EDAs are a novel paradigm in EA. They build a probabilistic model of the solutions and sample

[2] The dynamic pricing problem based on rest of the model can also be seen as constraint non-linear optimisation problem and can be solved using our approach.

the model to generate the child population. In this paper we test two EDAs and a GA for solving the neural network based dynamic pricing problem. These algorithms have been previously tested for solving dynamic pricing problem based on linear model (8) [25]. These algorithms include Population Based Incremental Learning (PBIL) algorithm [3], Distribution Estimation using Markov Random Field with direct sampling (DEUMd) algorithm [23,24] and a GA [8]. We also use a non-population based algorithm known as Simulated Annealing (SA) [10] for this problem. For the fairness of the results, we use same algorithms to also solve the dynamic pricing problems based on other three demand models.

We first present the workflow of these algorithms and then describe the key differences between them.

PBIL

1. Initialize a probability vector $p = \{p_1, p_2, \ldots, p_n\}$ with each $p_i = 0.5$. Here, p_i represents the probability of x_i taking value *1* in the solution.
2. Generate a population P consisting of M solutions by sampling probabilities in p
3. Select set D from P consisting of N promising solutions, where $N \leqslant M$
4. Estimate marginal probabilities of $x_i = 1$, for each x_i, as
$$p(x_i = 1) = \frac{\sum_{x \in D, x_i = 1} x_i}{N}.$$
5. Update each p_i in p using $p_i = p_i + \lambda(p(x_i = 1) - p_i)$
Here, $0 \leqslant \lambda \leqslant 1$ is a parameter of the algorithm known as the learning rate
6. Go to step 2 until termination criteria are meet

DEUMd

1. Generate a population P consisting of M solutions
2. Select set D from P consisting of N promising solutions, where $N \leqslant M$
3. For each solution, x, in D, build a linear equation of the form
$$\eta(F(x)) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n,$$
where $x_i = \{0, 1\}$ is replaced by $\{-1, 1\}$ for numerical purpose, function $\eta(F(x)) < 0$ is set to $-\ln(F(x))$, for which $F(x)$, the fitness of solution x, should be $\geqslant 1$; $\alpha = \{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n\}$ are the parameters of the equation;
4. Solve the build system of N equations to estimate $\alpha = \{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n\}$
5. Use α to estimate the distribution $p(x) = \prod_{i=1}^{n} p(x_i)$, where
$$p(x_i = 1) = \frac{1}{1 + e^{\beta \alpha_i}}, \quad p(x_i = -1) = \frac{1}{1 + e^{-\beta \alpha_i}}.$$
Here, β (inverse temperature coefficient) was set to $\beta = g \cdot \tau$; g is current iteration of the algorithm and τ is the parameter of the algorithm known as the cooling rate
6. Generate M new solution by sampling $p(x)$ and replace P
7. Go to step 2 until termination criteria are meet

GA

1. Generate a population P consisting of M solutions
2. Build a breeding pool by selecting promising solutions from P using a selection strategy
3. Perform crossover on the breeding pool to generate the population of new solutions
4. Perform mutation on new solutions
5. Replace P by new solutions and go to step 2 until termination criteria are meet.

SA

1. Randomly generate a solution x
2. For $i = 1$ to r do
2.1
Randomly mutate a variable in x to get x'
2.2
Set $\Delta F = F(x') - F(x)$
2.3
set $x = x'$ with probability
$$p(x') = \begin{cases} 1 & \text{if } \Delta F \leqslant 0, \\ e^{-\Delta F/T} & \text{otherwise,} \end{cases}$$
where temperature coefficient T was set to $T = 1/i \cdot \tau$; here, i is the current iteration and τ is the parameter of the algorithm called the cooling rate
3. Terminate with answer x

The two EDAs implemented here, PBIL and DEUMd, both fall in the category of univariate EDA. This means, and as can be seen from their workflow above, the model of distribution used by them assumes each variable, x_i, in the solution $x = \{x_1, x_2, \ldots, x_n\}$ to be independent. In other words, they do not take into account any possible interaction between variables in the solution. Other categories of EDA include, bivariate EDA, assuming at most pair-wise interaction between variables, and multivariate EDA, assuming interaction between multiple variables [11]. The main difference between these two algorithms, however, is in the way they estimate and sample the univariate marginal probabilities $p(x_i)$. As can be seen from their workflow above, PBIL counts the frequency of $x_i = 1$ in the selected set, D, for estimating $p(x_i = 1)$. By such, it allows each selected solution to have equal contribution in the estimation of the marginal probability. This marginal probability is then used to slightly update the vector of probability, p, which is then sampled to generate the new population. In contrast, DEUMd builds a model of fitness function and fits it to the selected set of solution to estimate the model parameters. These parameters are then used to estimate the univariate marginal probabilities $p(x_i)$, where their convergence to either -1 or 1 is regulated by using a temperature coefficient. These marginal probabilities are then directly sampled to generate the new population. It can be noticed that in contrast to PBIL, DEUMd uses fitness of a solution to regulate the contribution of that solution on estimation of marginal probabilities.

As with DEUMd, the simulated annealing algorithm also regulates the convergence by using a temperature coefficient. However, as can be seen from its workflow above, it does not work with a population of solution. Instead, it generates a single solution, and, at each iterations, slightly modifies the generated solution according to the estimated probability. This modification improves the value of the objective function for most of the iteration, however, can sometime worsen it depending upon the estimated probability. As such, simulated annealing can be seen as a sophisticated hill climbing algorithm with possible down-hill move [10]

6. Experiments and results

The aim of our experiment is to compare the pricing policy found by neural network based dynamic pricing to that found by dynamic pricing based on other three demand models in various different situations. For this purpose a large number of data sets were generated from a number of different source models. All four demand models were then fitted to the data to estimate the model parameters and these parameters were given as the input to the optimisation problem. The optimisation problem is then solved using EAs.

Table 1
RMSE for each demand model on the entire 45 instances.

Data	Model	I1	I2	I3	I4	I5	Avg
linear-1	Linear	0.7612337	1.2791611	1.7360962	1.4760198	2.8769078	1.6258837
	Exponential	80.492786	84.283994	66.780203	78.738281	84.496061	78.958265
	Multinomial logit	327.44054	329.66592	333.5761	332.29854	321.6613	328.92848
	Neural network	20.768401	18.537319	23.726112	15.815564	25.422217	20.853923
linear-2	Linear	1.4434159	1.8659656	1.974144	1.556086	1.0040161	1.5687255
	Exponential	222.04124	228.16515	223.7829	227.20262	227.20262	225.67891
	Multinomial logit	41.850623	42.690287	41.970023	54.888416	65.291271	49.338124
	Neural network	53.170748	38.984356	44.930152	35.931131	31.77334	40.957946
linear-3	Linear	0.9527093	1.3725661	2.0652103	1.9988703	1.0943243	1.496736
	Exponential	93.448863	95.131992	92.831744	92.831744	95.71908	93.992685
	Multinomial logit	47.691889	34.779342	64.004731	54.437112	87.855039	57.753622
	Neural network	39.997257	26.122539	32.516539	36.130061	31.343807	33.22204
exp-1	Linear	80.271788	80.33681	80.008664	80.271162	80.209132	80.219511
	Exponential	9.8965899	2.1762057	8.1612878	5.2555405	6.1677072	6.3314662
	Multinomial logit	189.35336	188.62464	202.70737	205.47824	192.59155	195.75103
	Neural network	56.508953	55.389404	58.302226	54.140675	54.057749	55.679801
exp-2	Linear	76.746176	76.925487	75.198672	74.392382	75.402508	75.733045
	Exponential	6.2099126	5.8167544	6.4029413	5.7280869	3.3821809	5.5079752
	Multinomial logit	356.55543	355.68068	356.67745	361.31298	359.58254	357.96182
	Neural network	50.742618	48.103444	51.674427	47.964919	49.13053	49.523188
exp-3	Linear	94.055195	93.881319	92.28717	93.619668	91.233442	93.015359
	Exponential	0.1777076	5.7127105	7.0175762	5.292536	8.4340588	5.3269178
	Multinomial logit	171.28977	165.25572	176.2023	161.76699	161.76699	167.25635
	Neural network	67.49488	63.612514	57.660203	65.26926	65.26926	63.861223
MNL-1	Linear	46.743109	47.842713	38.746164	38.569044	40.216592	42.423524
	Exponential	100.23751	84.650147	89.537608	99.736694	101.86418	95.205227
	Multinomial logit	35.943362	19.864375	23.966684	37.240888	32.418714	29.886804
	Neural network	44.031358	31.58043	31.988344	37.809671	45.002106	38.082382
MNL-2	Linear	67.560072	94.814345	68.288858	81.725959	63.484311	75.174709
	Exponential	169.79282	175.16026	171.80787	184.29299	185.86174	177.38314
	Multinomial logit	16.865255	5.1184162	10.226478	13.895572	13.62544	11.946232
	Neural network	58.889975	62.743353	67.153327	61.874807	65.162915	63.164875
MNL-3	Linear	71.83674	71.654559	72.078593	72.840033	73.677676	72.41752
	Exponential	53.049744	52.404925	52.832478	53.050885	53.346161	52.936838
	Multinomial logit	6.2708526	0.8838605	0.4302994	0.9095249	0.6054941	1.8200063
	Neural network	56.85719	52.150046	52.035179	43.186767	47.986221	50.443081

Table 2
Average RMSE for each models over each of the three data types.

Data	Model	Avg-1	Avg-2	Avg-3	Avg-all
Linear	Linear	1.6258837	1.5687255	1.496736	1.5637818
	Exponential	78.958265	225.67891	93.992685	132.87662
	Multinomial logit	328.92848	49.338124	57.753622	145.34008
	Neural network	20.853923	40.957946	33.22204	31.67797
Exp	Linear	80.219511	75.733045	93.015359	82.989305
	Exponential	6.3314662	5.5079752	5.3269178	5.7221197
	Multinomial logit	195.75103	357.96182	167.25635	240.32307
	Neural network	55.679801	49.523188	63.861223	56.354737
MNL	Linear	42.423524	75.174709	72.41752	63.338584
	Exponential	95.205227	177.38314	52.936838	108.5084
	Multinomial logit	29.886804	11.946232	1.8200063	14.551014
	Neural network	38.082382	63.164875	50.443081	50.563446

Similar to the observations made in [25], we found that, apart from SA, the three EAs tested in this paper, PBIL, DEUMd and GA, all performed well in this problem, giving very similar profit values. Since our aim is to compare the performance of the demand model, and not the performance of these algorithms, in this paper we only report the results obtained with PBIL. This is simply because the performance of PBIL was found to be more consistent in these problems [25] and is also supported by our experiments.[3]

Once the optimisation problem was solved, we record the pricing policy found by all four models and compare them to the pricing policy found by the original model from which the data was generated. The root mean square error (RMSE), given by (25), was used as the measure to compare the closeness of the pricing policy found by the fitted models to that found by the original model. The model that found the solution closest to that found by original model was then chosen as the best model. Let us describe the experimental setups in more detail.

$$RMSE = \sqrt{\frac{\sum_{i \in K} (expected_i - observed_i)^2}{K}}, \tag{25}$$

where K is the number of data samples.

[3] Although, we believe that both GA and DEUMd would have given us the similar conclusion with regard to the demand model performance.

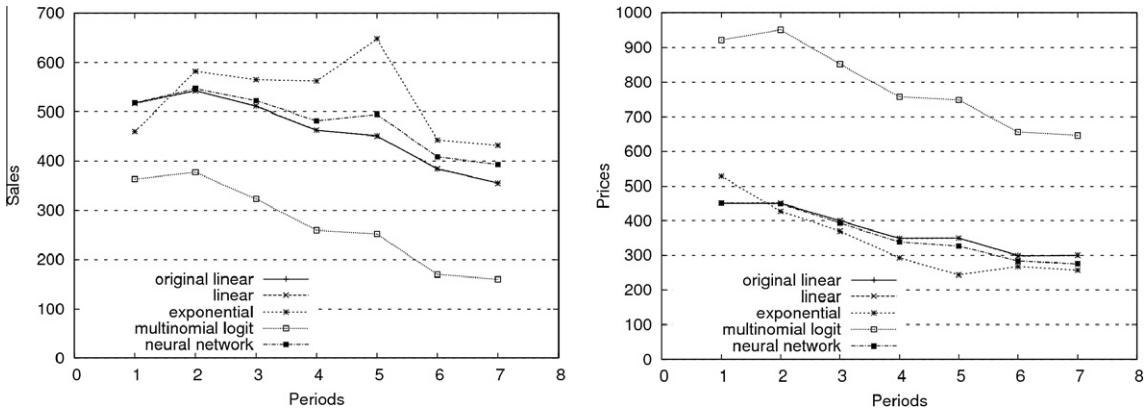

Fig. 2. A typical policy found by all four models on linear dataset.

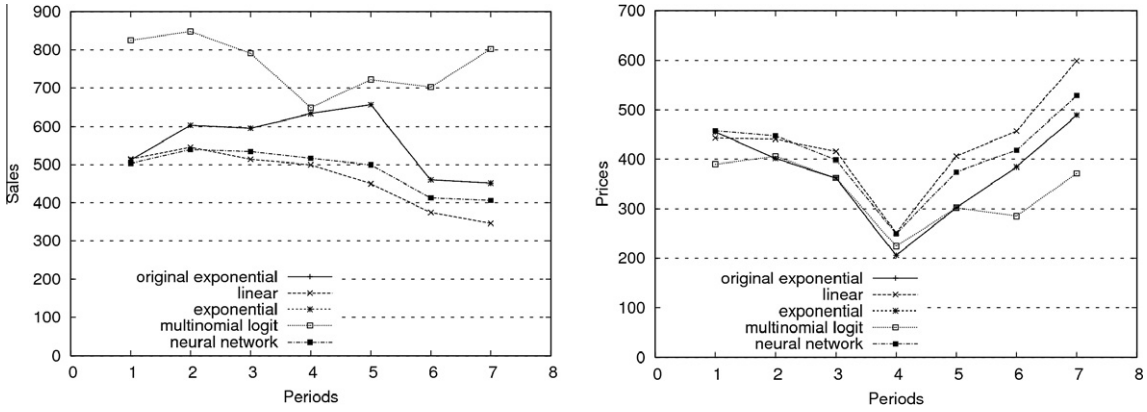

Fig. 3. A typical policy found by all four models on exponential dataset.

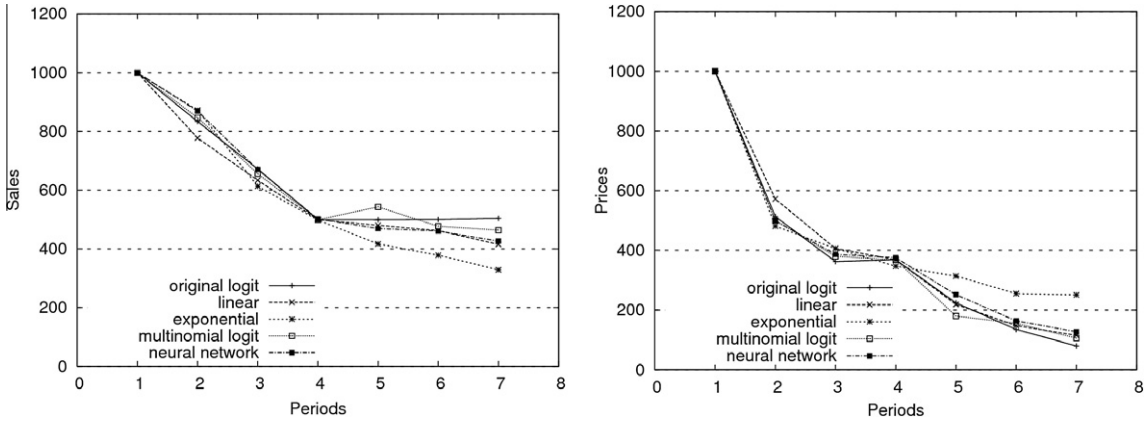

Fig. 4. A typical policy found by all four models on multinomial logit dataset.

6.1. Experimental setups

In this work, we look on weekly dynamic pricing problem, i.e. set the total period in planning horizon $N = 7$. In particular, we assume that the dynamic pricing problem is to correctly price the product for next week by looking on the historical demand price data for past 60 weeks.

Three different models, linear, exponential and multinomial-logit, were used to generate demand-price data. For each model, we choose three parameter sets, each modelling different scenarios. For each parameter set, we generate five different instances of demand price dataset. Each data set contained 60 records of weekly pricing policy and corresponding sales. The total dataset generated from a model was $5 \times 3 = 15$. Therefore, from three different models, total of $3 \times 15 = 45$ datasets were generated.

Following is the procedure for generating a dataset.

1. Randomly generate 60 pricing policies representing the policy for past 60 weeks
2. Pass each policy through the demand model and estimate the corresponding production
3. This gives a data set with prices and corresponding productions for past 60 weeks

Table 3
Grand average RMSE of all the models over all 45 dataset.

Data	Model	Avg-linear	Avg-exp	Avg-MNL	Grand-Avg
All	Linear	1.5637818	82.989305	63.338584	49.297224
	Exponential	132.87662	5.7221197	108.5084	82.369046
	Multinomial logit	145.34008	240.32307	14.551014	133.40472
	Neural network	31.67797	56.354737	50.563446	46.198718

In order to test the performance of the demand models following steps were performed.

First of all, the optimisation problem was solved with PBIL using the original model with the original parameter set. The found policy was taken as the best policy for that dataset. PBIL was tuned as follows: each P_t was represented using 12 bit vector, therefore the total length of the bit string solution was $12 \times 7 = 84$, population size was set to 400 and the selection size was set to 100, the learning rate was 0.1. The algorithm was allowed to do 100 iterations. These setups were determined empirically, i.e., they were the best setups out of number of tested setups.

Next, parameters for each of the four models on each of the 45 datasets were estimated by fitting them to each data set. We used linear regression for fitting linear demand model to the data. Singular value decomposition (SVD) [20] was used for solving the system of linear equations. For exponential demand model, the log was taken from the both sides to change it to the linear form and the linear regression, as with linear demand model, was used to estimate the parameters. With multinomial-logit demand model, a non-linear least square minimisation method was used to estimate the parameters. As for the neural network, as stated previously, a back propagation method was used to estimate the parameters.

Once the parameters were estimated, they were given as an input to the optimisation problem and PBIL was used to find the pricing policy. RMSE between these pricing policies and the optimal pricing policy was then estimated, which shows the competitive performance of all four models to find pricing policy closer to the optimal.

6.2. Results

Table 1 shows the RMSE for each demand model on the entire 45 datasets. Here, the first column defines the type of model with corresponding parameter set (represented by 1, 2 or 3) that was used to generate the data set; second column defines the type of the demand model that was fitted to the data. Next 5 columns define the RMSE for each model type in comparison to the original model for five different instances of the dataset. The final column, gives the average RMSE for each model type on these five instances.

We can notice that for the data set generated using a linear model (we call it linear data set), the linear demand model gives the best performance with very small RMSE value, i.e. with the policy closest to the optimal policy.[4] This is expected, since the linearity assumption made by the demand model exactly fits the data distribution. This result, therefore, is not very interesting. However, the interesting result here is that the neural network had the lowest RMSE amongst the remaining models, i.e. could give the closest-to-optimum results even without making any linearity assumption.

The case is similar with exponential and multinomial-logit data sets. As expected, the demand model that exactly matches the data set performs the best. The neural network is the best among remaining demand models. Table 2, shows this more clearly, where the overall average is presented for each data type. Also, in Fig. 2a and b, a typical policy found by each of these four models and how they compare to the policy found by original model on linear data set is shown, where Fig. 2a is for the production and Fig. 2b is for price. Similarly, Fig. 3a and b shows the same information for exponential data and Fig. 4a and b shows them for multinomial-logit dataset. Again, as expected, these figures show that the policy found by the model matching the dataset is closest to the optimal policy. The policy suggested by neural network is closest among the remaining models and has the similar curve as the optimal model. It is important to have the policy-curve similar to the optimal policy, since they verify that the model is correctly representing the scenario and by using such policy the profit will be closer to the optimal.

Finally, in Table 3, the overall average performance of all the models over all 45 dataset is presented. The key figure to note here is the overall average RMSE for neural network, which is the best among all four demand models. This result is particularly important since it shows that in a real world scenario, where the data distribution may not be available in advance, neural network gives the best estimate of demand price relationship. Subsequently, using such demand price relationship for optimising pricing policy, the policy found could be closer to the optimal policy. We note that the overall RMSE for rest of the models are somewhat biased, since they includes the very low RMSE that they obtained while optimising over the exact data. However, even when the comparison is made with such a biased results, the performance of neural network was better.

This confirms that neural network is the most consistent model for pricing, which can be fitted to the wide range of data. This is in contrast to the other models which do well when the data source matches the model but gives poor results when the dataset does not matches the model. This result also suggests that when the data source is not known, the neural network is safest model to use in order to get a reliable pricing policy.

7. Conclusions

Dynamic pricing can be a valuable tool for firm to maximise its revenue by varying prices according to the perceived demand. Moreover, it can also be used to control the flow of demand. By pricing higher during higher demand periods and lower during lower demand periods, the firm can shift the demand from higher demand periods to the lower demand periods.

In this paper we have shown how we can use neural network for optimising pricing policies in dynamic pricing. We build a neural network based demand model and showed how we can fitted it to the historical price-demand data to obtain it's parameters. The neural network demand model is then used to formulate the dynamic pricing problem. We also show how by using evolutionary algorithms we can successfully optimise pricing policy with neural network demand model.[5] Number of experiments has been performed comparing pricing policy obtained using neural network

[4] Although the model is the exact model of the data, the estimated parameters can have some estimation error. This error may depend on the level of noise in the data and the accuracy of the model fitting algorithm used. Consequently, the RMSE of the exact model is not equal to 0.

[5] It should be noted that we do not use EAs to evolve the weights of the neural network; instead we use EAs to solve the dynamic pricing problem based on neural network demand model.

with that obtained by other well established demand models. And the results show that other models perform well when the data source matches the model and perform poorly when the data source does not matches the results. In contrast neural networks are the most consistent models over wide range of data generated from different sources, and gives result closer to optimal in range of different scenarios. This is an important result which encourages dynamic pricing community to implement neural networks as an alternative model for optimising pricing policy. This is particularly important since in real life scenarios the data source is most likely to be not known.

Although, it is well known that for linear data, a simpler neural network with only the input layer and output layer (without any hidden layer) can accurately model the data, for the fair comparison, and also since we assume that the data distribution is not known, we use the same network for all three data sets.

It has been shown that a correctly build neural network (by giving enough hidden unit) can approximate any nonlinear functions, i.e. correctly choosing the network topology can give a more accurate model. We believe that by extending this approach to implement the topology learning process, the policy found by neural network could be significantly better than the one reported in this work and can be even more closer to the one found by the exact model. Also, a simple back propagation has been used to fit the neural network to the data, due to their popularity, and due to the acceptable quality of results produced by them. There are however range of other techniques, including the use of evolutionary algorithm, for training neural networks. Using more advanced training methods are likely to give better fit model, again resulting in more accurate pricing policy. It would also be interesting to compare the performance of EA optimised pricing policy with that optimised using other traditional methods, such as mathematical programming. All of these remain the part of the future works.

This work is a continuation to the work presented in [27], where a generic pricing system was described as a component of the Field Optimisation Toolkit [18]. The addition of proposed neural network based demand modelling makes the system more versatile, as it enables automatic learning of the model from the historical data, rather than the manual model building process currently required.

References

[1] M.A. Arbib, The Handbook of Brain Theory and Neural Networks, MIT Press, 1995.
[2] W. Baker, M.V. Marn, C. Zawada, Price smarter on the net, Harvard Business Review 79 (2) (2001).
[3] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, Technical Report, CMU-CS-94-163, Pittsburgh, PA, 1994.
[4] R. Bellman, Dynamic Programming, Princeton University Press, Princeton, New Jersey, 1957.
[5] M. Bichler, J. Kalagnanam, K. Katircioglu, A.J. King, R.D. Lawrence, H.S. Lee, G.Y. Lin, Y. Lu, Applications of flexible pricing in business-to-business electronic commerce, IBM Systems Journal 41 (2) (2002) 287–302.
[6] P.-C. Chang, C.-H. Liu, C.-Y. Fan, Data clustering and fuzzy neural network for sales forecasting: a case study in printed circuit board industry, Knowledge-Based Systems 22 (5) (2009) 344–355.
[7] K. Ferdows, M.A. Lewis, J.A.D.M. Machura, Rapid-fire fulfilment, Harvard Business Review 82 (11) (2004) 104–110.
[8] D. Goldberg, Genetic Algorithms in Search, optimization, and Machine Learning, Addison-Wesley, 1989.
[9] H. Hruschka, W. Fettes, M. Probst, An empirical comparison of the validity of a neural net based multinomial logit choice model to alternative model specifications, European Journal of Operations Research 159 (2004) 166–180.
[10] S. Kirkpatrick, C.D. Jr. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[11] P. Larrañaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, 2001.
[12] S.G. Li, L. Shi, L. Wang, The agile improvement of MMORPGs based on the enhanced chaotic neural network, Knowledge-Based Systems 24 (5) (2011) 642–651.
[13] G. McWilliams, Lean machine: how dell fine-tunes its pc pricing to gain edge in slow market, Wall Street Journal (2001).
[14] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions: I. binary parameters, in: H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature– PPSN IV, Springer, Berlin, 1996, pp. 178–187.
[15] Y. Narahari, C.V. Raju, K. Ravikumar, S. Shah, Dynamic pricing models for electronic business, Sadhana 30 (2005) 231–256. Part 2 and 3.
[16] S. Netessine, R. Shumsky, Introduction to the theory and practice of yield management, INFORMS Transactions on Education 3 (1) (2002). <http://ite.informs.org/Vol3No1/NetessineShumsky>.
[17] K.E. Parsopoulos, M.N. Vrahatis, Particle swarm optimization method for constrained optimization problems, in: P. Sincak, J. Vascak, V. Kvasnicka, J. Pospichal (Eds.), Intelligent Technologies – Theory and Application: New Trends in Intelligent Technologies, Frontiers in Artificial Intelligence and Applications, vol. 76, IOS Press, 2002. pp. 214–220.
[18] G. Owusu, C. Voudouris, M. Kern, A. Garyfalos, G. Anim-Ansah, B. Virginas, On optimising resource planning in BT with FOS, in: Proceedings International Conference on Service Systems and Service Management, 2006, pp. 541–546.
[19] R.L. Phillips, Pricing and revenue optimization, Stanford University Press, 2005.
[20] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, second ed., Cambridge University Press, Cambridge, UK, 1993.
[21] M. Qi, S. Yang, Forecasting consumer credit card adoption: what can we learn about the utility function?, International Journal of Forecasting 19 (2003) 71–85.
[22] A. Sahay, How to reap higher profits with dynamic pricing, MIT Sloan Management Review, 1532-9194 48 (4) (2007) 53–60.
[23] S. Shakya, J. McCall, D. Brown, Using a Markov network model in a univariate EDA: an emperical cost-benefit analysis, in: Proceedings of Genetic and Evolutionary Computation COnference (GECCO, 2005), ACM, Washington, DC, USA, 2005.
[24] S. Shakya, J. McCall, Optimization by estimation of distribution with DEUM framework based on Markov random fields, International Journal of Automation and Computing 4 (3) (2007) 262–272. Science Press, co-published with Springer-Verlag GmbH, ISSN 1476-8186 (Print) 1751–8520 (Online).
[25] S. Shakya, F. Oliveira, G. Owusu, An application of GA and EDA to dynamic pricing, in: Proceedings of Genetic and Evolutionary Computation COnference (GECCO, 2007), ACM, London, UK, 2007, ISBN 978-1-59593-697-4, pp. 585–592.
[26] S. Shakya, F. Oliveira, G. Owusu, Analysing the effect of demand uncertainty in dynamic pricing with EAs, in: M. Bramer, F. Coenen, M. Petridis (Eds.), Proceedings of AI-2008, Research and Development in Intelligent Systems XXV, Springer-Verlag, London, Cambridge, UK, 2008.
[27] S. Shakya, C.M. Chin, G. Owusu, An AI-based system for pricing diverse, products, services, Knowledge Based Systems, 0950-7051 23 (4) (2010) 357–362.
[28] K.T. Talluri, G.J. van Ryzin, The Theory and Practice of Revenue Management, Springer, Berlin, Heidelberg, New York, 2004.
[29] D. Teodorovic, P. Edara, Real-time road pricing system: the case of a two-link parallel, network, Computers and Operations Research 34 (1) (2007) 2–27.
[30] P.D. Wasserman, Neural computing theory and practice, Van Nostrand Reinhold, 1989.
[31] P.M. West, P.L. Brockett, L.L. Golden, A comparative analysis of neural networks and statistical methods for predicting consumer choice, Marketing Science 16 (4) (1997) 370–391.