A Dynamic Pricing Method in E-Commerce Based on PSO-trained Neural Network

Liang Peng and Haiyun Liu
School of economics, Huazhong University of Science and Technology,
Wuhan 430074

Abstract. Recently, dynamic pricing has been a common competitive maneuver in e-commerce. In many industries, firms adjust the product price dynamically by the current product inventory and the future demand distribution. In this paper, we used particle swarm optimization (PSO) algorithm to train neural networks, then introduced the PSO-trained neural network into e-commerce and presented a new dynamic pricing method based on PSO-trained neural networks. In the method, from production function principles we obtained the least variable cost, and by making the error of mean square between the actual outputs and expectation outputs minimal we got the optimal dynamic price of products. The PSO-trained neural network can simplify the rapid change of prices and can successfully set the optimal dynamic prices in e-commerce.

1 Introduction

The advancement of internet technology has enabled the establishment of electronic commerce which has made communication between buyers and sellers easy. This has resulted in the establishment of electronic price negotiation. Thus in recent years, the problem of dynamic pricing has drawn much attention. Dynamic pricing is a business strategy that adjusts the product price in a timely fashion in order to allocate the right service to the right customer at the right time. The rationale of dynamic pricing can be understood with many examples in e-commerce, such as airline tickets, new designer suits, and other marketing products associated with particular holidays etc. In all of these cases, the firm can improve its revenues by dynamically adjusting the price of the product rather than adopting a fixed price throughout the product's market life [1].

Dynamic pricing takes advantages to both buyers and sellers, some researchers have studied dynamic pricing methods [2-6]. Many ways to dynamic pricing are to change the price continuously over time by reacting to any shifts in demand

characteristics. However, such a practice is often either not feasible or too costly. This paper proposed a simple and feasible dynamic pricing method which can be used for on-line price setting. The method is based on the demand sensitive model where price changes depend on the quantity demanded and the variable cost changes with the quantity of product to be sold. In e-commerce, the information is incomplete. Considering neural networks have the ability to work with incomplete data and can be applied successfully in learning, predicting, and optimization functions [7-9], we introduced neural networks into dynamic pricing in e-commerce. Because the particle swarm optimization (PSO) has the ability of evolutionary learning [10], we used the PSO algorithm to train neural networks. In this method, the optimal dynamic price was set by the PSO-trained neural network. The process of dynamic pricing is the process that the PSO-trained neural network makes the cost function minimal.

The rest of this paper is organized as follows. Firstly we introduce the PSO algorithm and neural networks simply, and then build the PSO-trained neural network in section 2. In section 3, we presented the dynamic pricing method by using the PSO-trained neural network. The paper ends with conclusions in section 4.

2 The PSO-trained neural network

2.1 Preliminaries of neural networks

Generally a neural network often includes the following:

1. The number of layers in neural network.
2. The number of neural units in every layer.
3. The activation functions between layers.

By Kosmogorov Theory, under the condition of reasonable topology and right weights, a standard 3-layer neural network can approximate any continuous functions. Fig. 1 is the general topology of a neural network

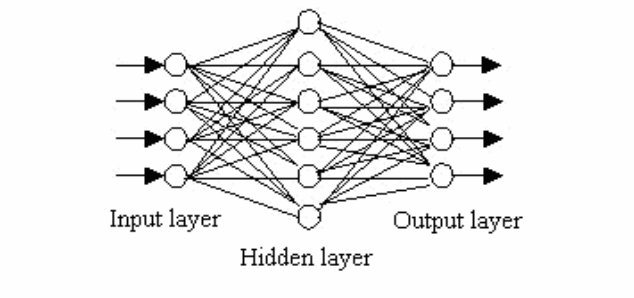

Figure. 1. The topology of a 3-layer neural network.

The normal hidden-units were logistic units with outputs in the range between 0 and 1. The input units and context units did not do any processing-they simply passed on their input. The output unit was a linear unit. Generally, sigmoid activation function was adopted in hidden layer. The sigmoid activation function is the following:

$$f(x) = \frac{1}{1+e^{-x}}$$

Generally, the network was trained using the error back-propagation algorithm with learning rate. The training process is to update the connection weighs and makes the network enhance learning. In this paper we used particle swarm optimization algorithm to train the neural network. After training has been completed, the neural network can record the corresponding output according to the inputs.

2.2 Particle swarm optimization (PSO)

The PSO technique was introduced by Kennedy and Eberhart. Inspired by the flocking behavior of birds, PSO has been applied successfully to function optimization, game learning, data clustering, and image analysis.

The PSO's operation is as follows. Each particle represents a possible solution to the optimization task. During each iteration each particle accelerates in the direction of its own personal best solution found so far, as well as in the direction of the global best position discovered so far by any of the particles in the swarm.

Let $\Pr esent$ denote the current position in the search space, V is the current velocity. During each iteration, each particle in the swarm is updated using (1) and (2).

$$\begin{aligned} V(t+1) = \omega * V(t) + c_1 * rand * (pBest(t) - \Pr esent(t)) \\ + c_2 * rand * (gBest(t) - \Pr esent(t)) \end{aligned} \quad (1)$$

$$\Pr esent(t+1) = \Pr esent(t) + V(t+1) \quad (2)$$

where, $rand \sim U(0,1)$ is the elements from uniform random sequence in the range (0,1), c_1 and c_2 is the acceleration coefficients, usually $c_1 = c_2 = 2$. ω is the weight coefficient, and $0.1 \le \omega \le 0.9$. $pBest$, the personal best position of each particle, is updated by

$$pBest(t+1) = \begin{cases} pBest(t), & if \ \ f(\Pr esent(t+1)) \le f(pBest(t)) \\ \Pr esent(t+1), & if \ \ f(\Pr esent(t+1)) > f(pBest(t)) \end{cases} \quad (3)$$

and $gBest$, the global best position, is the best position among all particles in the swarm during all previous steps. It means

$$gBest(t+1) = \arg\max_i f(pBest_i(t+1)), \quad for\ any\ particle\ i \quad (4)$$

The value of V can be clamped to the range $[-V_{\max}, V_{\max}]$ to ensure particles in the search space.

In this paper, the variable ω is the inertia weight, this means that the value of ω is typically setup to vary linearly from maximum to minimum during the course of iterations, and ω is formulated as follows.

$$\omega = \omega_{max} - iter \cdot \frac{\omega_{max} - \omega_{min}}{iter_{max}} \quad (5)$$

where, $iter_{max}$ is the times of maximum iteration, $iter$ is the times of current iteration.

2.3 The process of training

Using PSO algorithm to train neural networks, each particle corresponds to the complete augmented weight vector that connects the neural network's input-to-hidden and hidden-to-output layers. The error of mean square between the actual outputs and expectation outputs is looked as fitness function, using PSO algorithm to seek the optimal position (or the optimal weight value) and make the error of mean square minimal. The process of training is as follows:

- Step 1: Randomly choose a swarm of particles, the maximal iteration times and the minimal error of mean square, for each particle initialize the position and velocity.
- Step 2: Let $pBest$ be current best position, let also $gBest$ be current best position among all particles.
- Step 3: Judge whether the convergence condition is satisfied, if yes, go to step 6. Otherwise, go to step 4.
- Step 4: For any particles in the swarm:
  - i. Update position and velocity using (1) and (2).
  - ii. Update $pBest$ using (3).
  - iii. Update $gBest$ using (4).
- Step 5: If the convergence condition is satisfied, go to step 6. Otherwise, turn to step 4.
- Step 6: Output $gBest$.

3 Dynamic pricing by the PSO-trained neural network

Suppose a firm has to sell more than one product at a time. All products use the same factors of variable cost. The quantity to sell according to demand and pricing

decision is made simultaneously. A firm sells different levels of quantities by the demand. To facilitate maximizing profits the firm will try to use a combination of inputs that will minimize its total cost of a given level of quantity demanded. The firm is employing methods dynamically, adjusting price over time based on quantity demanded. By applying principles of production function, we can use the PSO-trained neural network to get a selling price at the least total cost. From production function principles:

$$Q = f(x) \tag{6}$$

where, Q is the vector of products $q_1, q_2, \cdots, q_n$, X is the vector of input variable factors $x_1, x_2, \cdots, x_m$.

All products use the same type of variable factors, and the total number of variable factors is the same. The relationship between the quantity demanded and the output quantity of the network is as follows:

$$q_n \geq q_n^t \tag{7}$$

where q_n^t is a network output quantity at time t, while q_n is a desired output quantity (quantity demanded). The variable cost function of output quantity q_n is:

$$c_v(q_n) = f(w_m, x_m) \tag{8}$$

where w_m is a constant input unit price of x_m and the total cost $C_T(q_n)$ is:

$$C_T(q_n) = c_v(q_n) + c_F \tag{9}$$

where c_F is a fixed cost.

The following is the process of dynamic pricing by PSO-trained neural network.

- Step 1: Input w_m, q_n as the units of input layer in the neural network.
- Step 2: According to $q_n^t = f(x_n)$, compute q_n^t by using the neural network.
- Step 3: If $q_n \geq q_n^t$, go to step 4, else turn to step 2.
- Step 4: Obtain the total cost $C_T(q_n)$ by equation $C_T(q_n) = c_v(q_n) + c_F$.
- Step 5: Output the selling price P from $P = C_T^{'}(q_n)$.

At the beginning of this method the difference between q_n^t and q_n is big and negative, it measures the error that the network makes. The error used to alter the input variable factors x_m in order that the network's response to the same input price w_m is better the next time around. At each alteration the difference will continue to decrease up to the point when it finds the least variable cost $c_v(q_n)$ for the quantity demanded q_q. The variable cost $c_v(q_n)$ is added to fixed cost c_F to give the total cost $C_T(q_n)$. The marginal cost $C_T^{'}(q_n)$ is the derivative of total production cost function with respect to the level of outputs. From Marginal Decision Rule the selling price is set at the point of production when marginal cost equals marginal revenue. At this point the production cost is the minimum. Setting selling price p equal to marginal cost $C_T(q_n)$ means that this is the minimum price the seller can sell the product, selling below this price leads to loss, and selling above this price maximizes profits. Profits are the difference between total revenues and total cost, where total revenues are the product of price and the quantity demanded.

The firm will maximize profits at the level of outputs where the slope of the profits curve is zero. Since profits are revenues less cost, the slope of the profits curve, marginal profits, is equal to marginal revenue less marginal cost. Therefore, when the firm maximizes profits where the slope of the profit curve is zero, marginal

revenue will be equal to marginal cost. The profits maximizing firm is selling product at its most efficient (lowest unit cost) of quantity demanded where marginal revenue equal to marginal cost. At that level the firm will sell the product at the price which is equal to the marginal cost, which will maximize the total revenues, in other words maximizing the total profits function. For any demanded quantity q_n, $c_v(q_n)$ must be the least variable cost function. The continuous changes in demand quantity permit the continuous input substitution within the input bundle. The changes in demand quantity q_n, change the input bundle and consequently change the least cost, and consequently change the selling price p.

4 Conclusion and future work

Dynamic pricing as a changing price in a marketplace is becoming characteristic of electronic commerce. This paper presented a novel dynamic pricing method based on PSO-trained neural network. According to production function principles, the PSO-trained neural network can simplify the calculations of the least variable cost function that is the main factor in dynamic pricing. The method can set the optimal dynamic price and can be used in e-commerce with different quantities and different levels of demand at one time. In the future work, we will give numerical examples and simulation experiments.

Acknowledgement

We would like to thank the Chinese National "985" Research Project ——"Science & Technology Development and Human Culture Sprit Innovation".

References

1. S. Jagannathan, J. Nayak, K. Almeroth et al, On Pricing Algorithms for Hatched Content Delivery Systems, *Electronic Commerce Research and Applications,* 1, 264-280 (2002).
2. R. Chatwin, Optimal Dynamic Pricing of Perishable Products with Stochastic Demand and a Finite Set of Prices, *European Journal of Operational Research,* 125,149-174 (2000).
3. J. Kephart and J. Hanson, A. Greenwald. Dynamic Pricing by Software Agents, *Computer Networks* ,32,731-752 (2000).
4. K. Lin, Dynamic Pricing with Real-time Demand Learning, *European Journal of Operational Research* ,174,522-538 (2006).
5. X. J. Wang, Y. L. Chen and W. Q. Zhu, The Pricing Strategies for Agents in Real E-Commerce, WINE 2005, LNCS 3828,887-894(2005).
6. S. Netessine, Dynamic Pricing of Inventory/Capacity with Infrequent Price Changes, *European Journal of Operational Research* ,174,553-580 (2006).

7. D. Arditi, F. E. Oksay and O. B. Tokdemir, Predicting the Outcome of Construction Litigation Using Neural Network, *Computer Aided Civil and Infrastructure Engineering* ,13,75-81 (1998).
8. L.C. Jiao, *Neural Network System Theory* ,Xidian University Press, Xi an(1996).
9. S.A. Cong, *Face the MATLAB toolbox nerve network theory and the application* , University of Science and Technology of China Press, Heifei(1998).
10. J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, *Proc IEEE international conference on Neural Networks* ,4,1942 -1948(1995).