Predicting Commodity Prices Using Artificial Neural Networks

Andy Korth
University of Minnesota, Morris
600 East 4th St.
Morris, MN 56267
kort0061@umn.edu

ABSTRACT

Artificial neural networks have been used to predict the prices of various commodities in the virtual economy of World of Warcraft.

Keywords

Artificial Neural Networks, Economic Prediction

1. INTRODUCTION

World of Warcraft (hereafter WoW) is the world's largest roleplaying game, with over six million players worldwide investing 180 million hours every week online. Every one of these individuals is a player in an economy that centers around the game's auction house. Virtual economies are known to share many attributes with real economies of our world [1,2]. Thus, an understanding of this simplified economy could advance our knowledge of more complicated economic systems in real life. Needless to say, the analysis of the goods traded in the auction house could prove to be highly lucrative for anyone involved.

Neural networks have been demonstrated as a effective tool in various economic predictions [3]. Wei Cheng developed a neural network with 67% accuracy on purchase bids of 30-year US treasury bonds. Because of the simplified nature of this virtual economy and the availability of more perfect information, even higher accuracy could be expected.

Significant preprocessing of the gathered data was required to ensure the accuracy of this neural network. This included removal of outliers in the data and the hand selection of specific commodities to model. This preprocessing had a considerable effect on the accuracy of the model.

2. ECONOMIC SYSTEM OF WOW

The majority of significant transactions occur within the in-game auction house in World of Warcraft. Individuals may choose to place successively higher bids on auctions that last either 8, 12, or 24 hours. At the end of the prescribed time, the item goes to the highest bidder. Alternatively, the item may be purchased immediately if the bidder meets the seller's buyout price.

A wide variety of goods are available on the auction house, but for the purpose of this research, I focus only on heavily traded basic commodities; these are consistently available and thus data is more readily gathered for them. These primarily include the commodities that are inputs to consumable goods (non-durable). Therefore, there should be a constant demand for these products.

Care was taken to exclude goods where special circumstances have greatly modified their price. For example, the price of Major Mana Potions has increased five fold after a certain exploit was fixed. This exploit allowed certain individuals to rapidly acquire large quantities of this good with little effort. Because the neural network is not presented with this additional information, very little success would be expected in this case.

Among players, there is a widespread perception that weekends are the best times to sell goods, as the volume traded on those days are highest. In fact, other virtual economy researchers have observed that "Monday - Thursday is pretty poor for sales... Sunday night (server time) for US players is the best.". Trends such as these would be rapidly identified in a neural network.

3. TRAINING EXAMPLES

Although all of the auction information is digitally stored, it is not directly available to players. Thus, I have modified the open source auction scanning tool Auctioneer [4] to gather data suitable for this neural network. Once run, this software reads the bid price, buyout price, quantity, and more of every good listed

in the auction house. This data is then passed to another piece of software to for statistical analysis.

A total of 201,309 data points were gathered between February and April 2006. In addition to the prices recorded, the date and time on which the scan completed was also recorded.

4. CONSTRUCTING THE NEURAL NET

Initial Network Design

The initial neural network was designed to have four inputs nodes, five nodes in the hidden layer, and three distinct output nodes. The inputs were bid price, buyout price, number available for purchase, and the date on which this price was observed. The outputs were the expected buyout price change, the expected bid price change, and the expected new quantity.

Design Revision

After further consideration and investigation of other economic neural network prediction research [3,5], the model was simplified to have a single output node, indicating whether the commodity represented would be above or below it's average price.

Representing Linearly Separable Data

A single perceptron can only represent linearly separable data [6, pg 95]. For this reason, modeling the day of week on which a trade occurred is not trivial. For example, if there is a correlation between higher prices and the weekend, we might have positive training examples on days 0 and day 6, while having more negative examples in between. However, a single perceptron can not model this data.

The solution to this problem is to create 'boolean' data inputs. I have created a relevant inputs such as 'is this a weekend', 'is it evening', and 'is it afternoon'? These input nodes on the neural network can be either true or false, and can thus be used in determining weights without the problems seen with disjoint input from training examples.

Initial Squashing Functions

Although the input to a neural network may be in any range, they tend to function best with inputs [0, 1]. In addition, output of any perceptron is guaranteed to be [0,1] as well. The most basic and obvious method to squash a set of numbers to the set range is to divide each by by the maximum of the set.

Initial Results

The initial neural network simulation used the above basic squashing function for its four input nodes. There were five nodes in the hidden layer and one output node.

This network classified about 40 percent of the training examples incorrectly. These results were fairly positive, since the results were consistently better than random choice; thus, these predictions would be profitable in the long run.

Looking at the training examples created by a Java program created for parsing the data from World of Warcraft revealed some interesting room for improvement. The vast majority of numbers were within 0.1 percent of a median value. This type of distribution makes it very difficult for the neural network to assign a significant and valid weight to that input node, because the values are close together. In addition, this input node would be extremely sensitive to high momentums or learning rates, since it would easily pass over the 'best' weight- where the most training examples were correctly classified. The reason these values were so close together was because of the effect of a very few outliers. In the instance tested, there were several thousand auctions of Runecloth scanned. One individual had posted an auction with a ridiculously high price, on the order of 750 times higher than the actual market value of the product. This commodity, posted at 1000 gold was the clear maximum chosen for the squashing function, despite the fact it was a clear outlier that should be removed.

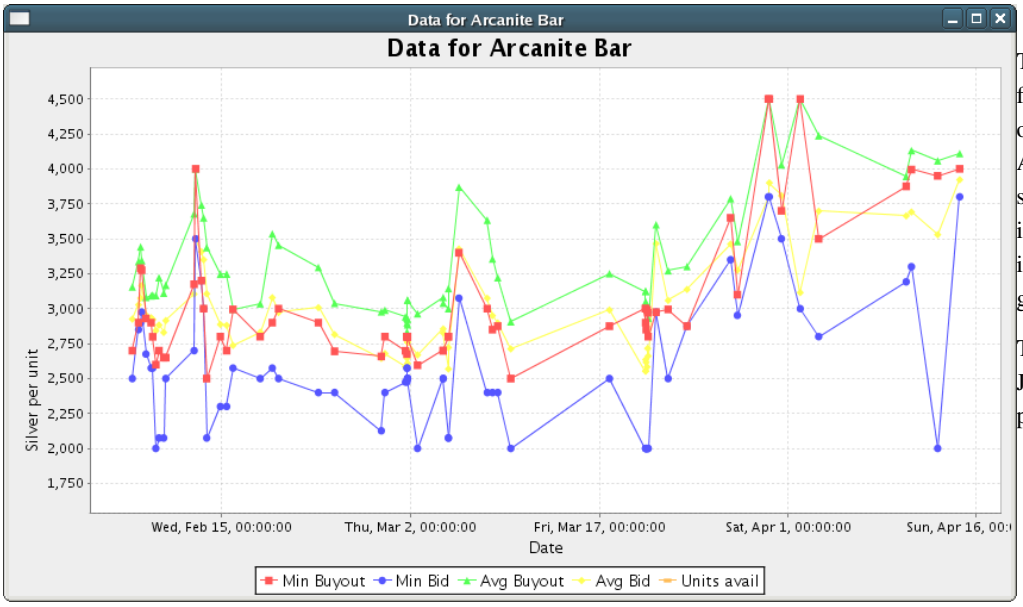
5. STATISTICAL ANALYSIS

Identification of Outliers

Removal of outliers and clearly incorrect data should significantly improve the accuracy of the neural network. Thus, the Java software that parses the raw data out of the saved files was changed to do some basic statistical analysis. Auction data tends to follow a binomial distribution, somewhat right-skewed (towards higher prices, since the low priced goods tend to be purchased immediately and removed from the scanned data set). For this reason the software was changed to calculate the standard deviation to identify outliers. First, the average for the commodity as a whole was calculated. Then, the standard deviation for any given scan was calculated, and individual auctions were removed if they were outside two standard deviations [6, pg 136].

Results

These changes can be fed into an improved version of the squashing function, where the data, with outliers removed, can be squashed between the minimum of the set and the maximum of the set. This again results in all of the data being between zero and one, but the information is much more distributed throughout that range.



This graph indicates the prices for arcanite bars once the outliers have been removed, Although the price is fairly stable, there are many variations in price that are not easily interpreted by the human eye given this incomplete data.

This data was graphed using Java software developed for this project.

6.FINAL DESIGN AND RESULTS

Using the improved squashing function and training examples with the outliers removed greatly improved the predictions. This final network consisted of ten input nodes, twelve hidden nodes, and a single output node. It was found that increasing the number of hidden nodes from seven to twelve improved performance by a fair bit, but increases beyond twelve were negligible.

The final inputs were:

- Mean bid price
- Mean buyout price
- Mean number available
- Minimum bid price
- Minimum buyout price
- Boolean: is it a weekend?
- Boolean: is tomorrow a weekend?
- Boolean: is it late evening?
- Boolean: is it the afternoon?

Again, the only output node was if the good would be above the average price next time or not.

Training examples used Doug's Momentum [7] set to 0.05 and a learning rate of 0.1. In short, Doug's Momentum is a modification of the standard momentum rule from Mitchell, except that there is an upper bound on the size of the update from the momentum. This allows the relatively high learning rate that was chosen.

In general, I found low error on many major commodities. These are relatively cheap goods, that are heavily traded- bought and sold by a large number of individuals. Economists might note that these goods fulfill the prerequisites of Perfect Competition [8]: A large number of buyers and sellers, a homogeneous (undifferentiated product), perfect information about the product, and negligible entry costs into the market (meaning no significant investment is required before one is able to gather these products).

Runecloth - 20,000 training epochs – 11% error
Dreamfoil - 25,000 training epochs – 8% error
Plaguebloom - 250,000 training epochs – 9% error

However, commodities which are less heavily traded, or are not modeled under perfect competition do not perform quite as well. For example, the arcane crystal market has a much lower volume and trades at a higher price. These prices are probably more closely watched, in general.

Arcane Crystal – 25,000 training epochs – 25% error

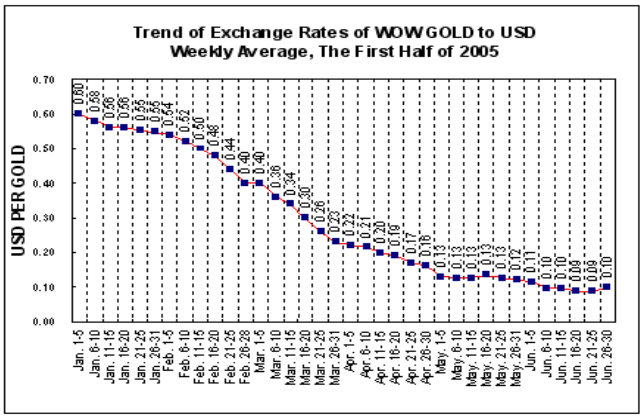
7. CONCLUSIONS

I have found artificial neural networks to be a highly effective means to model virtual economies. The simplified system of a virtual economy allows them to be modeled much more easily than real-life economies. This is largely a function of the purity of the market; the game code enforces a lot of variables. For example, all commodities are homogeneous, one person's runecloth is exactly the same as another's. In real life, differently branded, but otherwise identical products are valued very differently. The same perfect information about the product is provided in game, at the time of purchase to every player. Removal of these complexities greatly simplifies the market prediction. Much of the information that can be learned from virtual economies is useful not only to modelers of real

economies, but the information learned can be very valuable to economists as well.

In personal tests using the neural network and my associated software, I have increased my initial investment approximately eight fold in two weeks of commodity trading. Weekly fluctuations in consumable products or their raw materials tend to be the largest money makers and are most consistent.

With a total initial investment of 20 gold (about $1.40 USD at the current exchange rate), I have reached around 250 gold in on hand cash and assets (about $16).



"*Source:* WOW Gold Price Research [9]"

8. FUTURE RESEARCH

Overfitting of Data

The surprisingly low error in training examples, often around 10%, leads me to believe that overfitting may have occurred. Overfitting is defined as a hypothesis that has been found that has a smaller error over the training examples than the entire distribution of instances [6, pg 67].

There is some evidence of this occurring, although more research would be required to determine the extent of the overfitting. For example, a neural network trained on dreamfoil has a fairly high error when evaluating plaguebloom examples, although less than 40 percent. This is particularly interesting because these different commodities are extremely similar and are generally expected to follow the same trend.

Statistical Improvements

I believe there is still room for improvement in some of the statistical calculations done in my Java software. Recently, after my research was conducted, the developers of Auctioneer [4] moved to use a system more dependent on medians in place of the outlier sensitive 'mean'. In addition, they use interquartile ranges in a number of calculations as well.

Output Variables

Additional output variables, other than the price relative to the average price would increase the value of the network. I had chosen boolean output variables because of the difficulty of 'un-squashing' the result. However, others have had success in this area, so something as complicated as price prediction would be possible. An indicator of short range change would be very beneficial to possible users of this program; that is, will the price go up or down tomorrow?

9.ACKNOWLEDGMENTS

Thanks to Dian Lopez for her guidance on this project and for teaching the Machine Learning course at UMM. Thanks to the open source developers of Auctioneer, which was used, in part, to gather training examples. Finally, thanks to the House of the Amber Moon for their help in the game.

10.REFERENCES

[1] Bartle, Richard A. (2003). Designing Virtual Worlds. Indianapolis: New Riders.

[2] Castronova, Edward (2002). On Virtual Economies. CESifo Working Paper Series No. 752.

[3] Cheng, Wei, et all. (1996). Forecasting the 30-year U.S. Treasury Bond with a System of Neural Networks

[4] Auctioneer. http://sourceforge.net/projects/auctioneer-wow/

[5] Patterson, Joseph, et al. *Using Neural Networks to Predict Changes in Stock Prices.* University of Minnesota, Morris, 2004.

[6] Mitchell, Tom M. (1997). Machine Learning. New York: McGraw Hill.

[7] Lens Documentation. http://tedlab.mit.edu/~dr/Lens/thumb.html

[8] Microeconomics and Behavior, (2005), *Robert S. Frank*, 6th Edition. Irwin McGraw Hill Publishers, Inc.

[9] WOW Gold Price Research, http://www.gameusd.com/