In *AAMAS-07 Workshop on Agent Mediated Electronic Commerce* (AMEC 07),
Honolulu, Hawaii, May 2007.

# Adapting Price Predictions in TAC SCM

David Pardoe and Peter Stone

Department of Computer Sciences
The University of Texas at Austin, Austin TX 78712, USA
{dpardoe, pstone}@cs.utexas.edu

**Abstract.** In agent-based markets, adapting to the behavior of other agents is often necessary for success. When it is not possible to directly model individual competitors, an agent may instead model and adapt to the market conditions that result from competitor behavior. Such an agent could still benefit from reasoning about specific competitor strategies by considering how various combinations of these strategies would impact the conditions being modeled. We present an application of such an approach to a specific prediction problem faced by the agent TacTex-06 in the Trading Agent Competition's Supply Chain Management scenario (TAC SCM).

## 1 Introduction

In this paper, we present an adaptive approach used in the TAC SCM competition that is based on learning from simulations of various agent combinations. We describe a specific prediction problem faced by TacTex-06 (winner of the 2006 competition), present the learning approach taken, and evaluate the effectiveness of this approach through analysis of the competition results. We then explore methods of improving predictions through combining multiple sources of data reflecting various competitor behaviors. Although this paper only describes the application of these methods to the TAC SCM domain, the methods depend only on a need for some form of prediction and the ability to simulate a variety of potential opponent strategies, neither of which is uncommon in the real world. The work described here represents the main improvements over our 2005 agent, described fully in [6].

## 2 Learning and Adaptation in Agent-Based Markets

In competitive multiagent systems, the ability to adapt to the behavior of other agents can be the difference between success and failure. Often, this adaptation takes the form of opponent modeling [1] [2], in which a model is learned for each competing agent that can be used to predict the action the agent will take in any situation. In some systems, however, modeling agents directly may not be appropriate, or even possible. Market scenarios often fit this description for a number of reasons. For instance, an online seller might not interact with the same customer repeatedly, removing the incentive to model the individual customer's behavior. In large systems such as stock markets, the actions of a single agent may not be significant enough to have a noticeable effect on the system. Finally, in a market with limited transparency, such as one in which transactions are conducted through sealed-bid auctions, it may be impossible to directly observe the actions of other agents. In these situations, it may be necessary for an agent to observe and learn about the aggregate effect of all agents on the economy, rather than the behavior of specific agents. Learning is reduced to making predictions about properties of the economy, such as what a particular price will be. In effect, the competing agents become part of the agent's environment.

An agent using such an approach may still be able to benefit from reasoning about the types of behavior that might be exhibited by competing agents. In choosing an approach to adapting in the marketplace, an agent should take into consideration the range of strategies that other agents might use and how these strategies might affect the properties of interest. In general, an agent should consider the following questions:

- For which properties of the economy do predictions need to be made?
- Which of these properties are dependent on competitor strategies, and which tend to remain the same regardless of competitors?
- What predictive models should be used when starting out in a new market about which little information is available (i.e., what predictive models give the best expected performance across a variety of competitor behaviors)?
- As more information becomes available, what form of adaptation should be used to improve predictions?

One method of answering these questions, and the method that will be employed in this paper, is to implement a number of potential competitor strategies and run simulated markets using various combinations of these strategies. Using the results, it is possible to observe how market conditions vary based on the mix of competitors and to identify adaptive strategies that are effective across a range of possible scenarios. In the next two sections, we introduce the specific prediction problem to which we will apply this method.

## 3 The TAC Supply Chain Management Scenario

Supply chains have traditionally been created through the interactions of human representatives of the various companies involved. However, recent advances in autonomous agent technologies have sparked an interest in automating the process through the use of agents [3] [4]. The Trading Agent Competition Supply Chain Management (TAC SCM) scenario provides a unique testbed for studying and prototyping such agents. Though purely a simulated environment, TAC SCM is designed to capture a broad range of issues that come up in real-world supply chains, including limited supplies and manufacturing resources, competition for procurement leading to complicated price structures, competition for customer orders, storage costs, etc. A particularly appealing feature of TAC is that, unlike in real supply chains, strategies can be tested without risking large amounts of money, yet unlike in many simulation environments, the other bidders are real profit-maximizing agents with incentive to perform well, rather than strawman benchmarks.

In a TAC SCM game, six agents act as computer manufacturers in a simulated economy managed by a game server. The length of a game is 220 simulated days, with each day lasting 15 seconds of real time. The game can be divided into three parts: i) procuring components from suppliers, ii) selling computers to customers, and iii) production and delivery, as illustrated in Figure 1. We describe here only the sales task that is the focus of this paper, but full details are available in the official specification document [5].

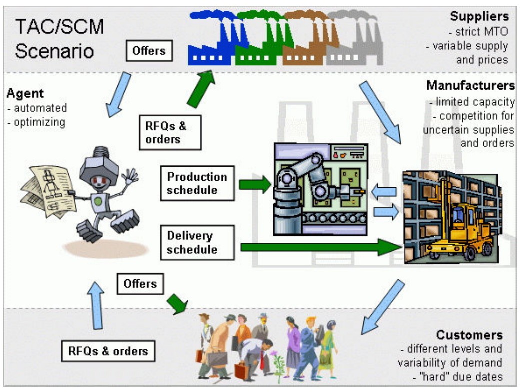

**Fig. 1.** The TAC SCM Scenario [2]

Customers wishing to buy computers send the agents requests for quotes (RFQs) consisting of the *type and quantity* of computer desired, the *due date*, a *reserve price* indicating the maximum amount the customer is willing to pay per computer, and a *penalty* that must be paid for each day the delivery is late. Agents respond to the RFQs by bidding in a first-price procurement auction: the agent offering the lowest price on each RFQ wins the order. Agents are unable to see the prices offered by other agents or even the winning prices, but they do receive a report each day indicating the highest and lowest price at which each type of computer sold on the previous day.

The number of RFQs sent by customers each day depends on the level of customer demand, which fluctuates throughout the game. Demand is broken into three segments, each containing about one third of the 16 computer types: high, mid, and low range. Each range has its own level of demand. The total number of RFQs per day ranges between roughly 80 and 320, all of which can be bid upon by all six agents. It is possible for demand levels to change rapidly, limiting the ability of agents to plan for the future with confidence.

## 4 TacTex-06 and the Computer Price Prediction Problem

We now give a brief overview of TacTex-06, and then introduce the problem addressed in this paper: predicting the price at which each type of computer will sell in the future. More information on the design of the agent is available in [6].

### 4.1 Agent Overview

In TacTex-06, tasks are divided between a *Supply Manager* module and a *Demand Manager* module. The Supply Manager handles all planning related to component inventories and purchases, and requires no information about computer production except for a projection of future component use, which is provided by the Demand Manager. The Demand Manager, in turn, handles all planning related to computer sales and production. The only information about components required by the Demand Manager is a projection of the current inventory and future component deliveries, along with an estimated replacement cost for each component used. This information is provided by the Supply Manager.

The goal of the Demand Manager is to maximize the profits from computer sales subject to the information provided by the Supply Manager. To accomplish this, the Demand Manager needs to be able to make predictions about the results of its actions and the future of the economy. Two predictive models are used to make these predictions: a *Demand Model* that predicts future customer demand levels, and an *Offer Acceptance Predictor* that predicts the probability of a particular offer winning an order from a customer, as described below.

### 4.2 Offer Acceptance Predictor

In order to bid on customer RFQs, the Demand Manager needs to be able to predict the orders that will result from the offers it makes. A simple method of prediction would be to estimate the winning price for each RFQ, and assume that any bid below this price would result in an order. Alternatively, for each RFQ the probability of winning the order could be estimated as a function of the current bid. This latter approach is the one implemented by the Offer Acceptance Predictor. For each customer RFQ received, the Offer Acceptance Predictor generates a function mapping the possible bid prices to the probability of acceptance. (The function can thus be viewed as a cumulative distribution function.) This approach involves two main components: a particle filter used to generate initial predictions, and a learned predictor that predicts how the prices of computers will change in the future.

A visual inspection of each day's winning prices for each type of computer in a typical completed game suggests that these prices tend to follow a normal distribution. To estimate these distributions during a game, the Offer Acceptance Predictor makes use of a separate particle filter for each computer type. Each of the 100 particles used per filter represents a normal distribution (indicating the probability that a given price will be the winning price on the computer) with a particular mean and variance. The distribution of winning prices predicted by the particle filter is simply the weighted sum of the individual particles' distributions, and from this distribution the function mapping each possible bid price to a probability of acceptance can be determined. Each filter is updated daily based on the information made available about computer prices: the high and low prices reported for the previous day and the offers received from customers.

In order to maximize revenue from the computers sold, the Demand Manager needs to consider not only the prices it will offer in response to the current day's RFQs, but also what computers it will wish to sell on future days. In fact, the Demand Manager plans ahead for several days and considers future RFQs (predicted by the Demand Model) as well as current RFQs when making offers. It is therefore important for the Offer Acceptance Predictor to be able to predict future changes in computer prices. To illustrate why this is important, Figure 2 shows the prices at which one type of computer sold during a single game of the 2006 competition. For each day, points representing one standard deviation above and below the average price are plotted. On most days, there is clearly little variance between the winning prices, but prices often change drastically over the course of a few days. This fact suggests that it may be even more valuable to be able to predict future changes in price than to predict the distribution of winning prices on a single day. By simply selling a computer a few days earlier or later, it might be possible for the Demand Manager to significantly increase the price it obtains.

In the remainder of this paper, we describe the use of machine learning methods to predict the amount by which the average sales price of each type of computer will change in ten days. Once the Offer Acceptance Predictor has learned to predict this quantity, it can predict the change in average price for any day between zero and ten days in the future through linear interpolation. No effort is made to predict changes in the shape of the distribution, i.e., the variance. Thus, to generate an offer acceptance function for a predicted future RFQ, the Offer Acceptance Predictor simply shifts the predicted distribution over winning prices up or down depending on the predicted change in average price, and bases the acceptance function on this modified distribution.

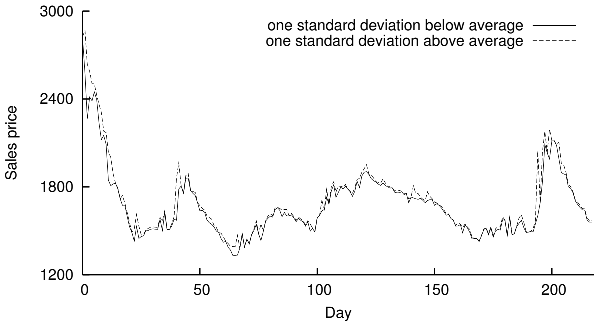

**Fig. 2.** Average prices at which one type of computer sold during one game of the 2006 finals. One standard deviation above and below the average are shown.

## 4.3 Learning Price Change Predictions

The problem explored in this paper is thus that of learning to predict changes in sales prices of computers. As discussed in Section 2, making an accurate prediction might depend on adapting to the behavior of the five competing agents. The structure of the TAC SCM competition encourages such adaptation: after a seeding round in which agents play games against random opponents, agents are divided into brackets of six and play a number of games against the same set of opponents, with the top three agents moving on to the next round. In addition, after each game a log is provided that details the complete events of the game, providing much information that was not available to the agent during the game. No human-made changes are allowed during a round, but agents are free to automatically adapt based on previous games during a round.

Although it is possible in principle to directly model the bidding behavior of specific opponents using data from games in the current round or previous rounds, we use the alternative approach mentioned in Section 2 of modeling the economy itself, treating opponents as part of the environment. We do so for two reasons. First, the information available during a game about opponents is extremely limited. An opponent's behavior is likely to be heavily dependent on information that cannot be observed, such as the opponent's inventory. Second, the behavior of an agent may be dependent on the mix of opponents in a game and the market conditions resulting from this mix. We were able to observe this fact clearly from the results of the 2005 competition. Table 1 shows the scores of the top 12 (out of 25) agents in the seeding round. Those agents in bold eventually advanced to the final round, the results of which are shown in Table 2. From these tables we can observe that scores decreased significantly from the seeding round to the final round as the competition increased, and in fact, some agents that were profitable in the seeding round lost money in the final round. Also, several of the top agents in the seeding round failed to advance to the final round. These observations confirm that, as is common in many market scenarios, TAC agents can behave and perform differently depending on market conditions, and that directly predicting an opponent's behavior may be difficult when the opponent is faced with unfamiliar market conditions. In fact, it might be better to base predictions on games with similar conditions but different agents than games with the same agents but different conditions.

| *Rank* | *Agent* | *Average Profit* |
|---|---|---|
| **1** | **TacTex-05** | $14.89M |
| 2 | GoBlueOval | $12.60M |
| 3 | FreeAgent | $12.06M |
| 4 | CMieux | $10.35M |
| **5** | **Deep Maize** | $10.23M |
| 6 | Botticelli | $10.11M |
| **7** | **SouthamptonSCM** | $10.05M |
| 8 | PhantAgent | $9.87M |
| **9** | **MinneTAC** | $9.86M |
| **10** | **Mertacor** | $9.30M |
| **11** | **Maxon** | $8.76M |
| 12 | CrocodileAgent | $8.48M |

**Table 1.** Top 12 agents in the 2005 seeding round. Agents in bold advanced to the final round.

| *Rank* | *Agent* | *Average Profit* |
|---|---|---|
| 1 | TacTex-05 | $4.71M |
| 2 | SouthamptonSCM | $1.60M |
| 3 | Mertacor | $0.55M |
| 4 | Deep Maize | *-$0.22M* |
| 5 | MinneTAC | *-$0.31M* |
| 6 | Maxon | *-$1.99M* |

**Table 2.** Results of the 2005 final round

The Offer Acceptance Predictor therefore attempts to predict changes in computer prices as a function of observable market conditions. As described in Section 4.2, the specific prediction made is the amount by which the average sales price of each type of computer will change in ten days. To make these predictions, the Offer Acceptance Predictor performs machine learning on data from past games. Each training instance consists of 31 features representing data available to the agent during the game, such as the date, estimated levels of customer demand, and current and recent prices of a given type of computer. The label for each instance is the amount by which the average price of that computer changes in ten days. The question addressed in the rest of the paper is how to best make use of all available data when generating predictors. In the next section, we explain how this question was answered for the 2006 competition.

## 5 The 2006 TAC SCM Competition

We now address how TacTex-06 performed prediction in the 2006 competition. First we describe the choice of opposing agents used in simulations and of a learning approach, and then we present the results of the final round of competition and additional experiments.

## 5.1 Agent Implementations

In order to develop a strategy for learning to make predictions, we ran a number of games using a variety of competing agents taken from the TAC Agent Repository,[1] a collection of agent binaries provided by the teams involved in the competition.[2] At the time we designed our agent, only agents from the 2005 competition were available; however, in the experiments of this section, we make use of additional agents that have become available since then, including some of the agents that participated in the 2006 competition, as this allows us to present experiments involving a wider variety of agents.

We chose four different agent groupings, and ran 50 games with each group. The groups are shown in Table 3. The first three groups contain TacTex-06 and fifteen additional agents. The fourth group includes what appear to be the strongest agents from the first three groups: TacTex-06, the 2005 version of TacTex, and the four other agents from the 2006 final round for which binaries are available.

We included TacTex-06 in each group because we are only interested in making predictions for games in which our agent plays, and we therefore would like to capture the effect of TacTex-06 on the economy in the predictive models learned. It is important to note that the choice of predictors can impact the behavior of TacTex-06 and thus the property of the economy (computer prices) we are trying to model. For the games played in this section, TacTex-06 used the same predictors that it used in the 2006 competition, so that the behavior of the agent is the same for all games (in or out of competition) discussed in this paper. We ultimately view consideration of this issue to be the responsibility of the agent, and not the learning process – an agent should be able to account for the fact that by behaving as its predictor suggests it should, it may be affecting the economy in a way that makes its predictions incorrect. As the focus of this paper is the learning process, we omit further discussion of this issue.

| *Group* | *Agents* |
|---|---|
| *1* | TacTex-06, GeminiJK-05, Mertacor-05, MinneTAC-06, PhantAgent-06, RationalAgent-05 |
| *2* | TacTex-06, TacTex-05, Botticelli-05, CrocodileAgent-05, DeepMaize-05, GoBlueOval-05 |
| *3* | TacTex-06, DeepMaize-06, Foreseer-05, Maxon-06, MinneTAC-05, PhantAgent-05, |
| *4* | TacTex-06, TacTex-05, DeepMaize-06, Maxon-06, MinneTAC-06, PhantAgent-06 |

**Table 3.** The agent groups used in the experiments

## 5.2 Learning Algorithms

When determining the learning approach to be used by TacTex-06, the first task was to identify a suitable machine learning algorithm. After limited experimentation (using default parameters and a limited amount of data) with the available regression algorithms from the WEKA machine learning package [7], we determined that the most promising candidates were M5 regression trees and additive regression with decision stumps (an iterative method in which a decision stump

[1] `http://www.sics.se/tac/showagents.php`

[2] The binaries of competing agents would admittedly not be available in a real scenario, but the approach described here could still be implemented by replacing these binaries with our own agents designed to exhibit a variety of behaviors.

is repeatedly fit to the residual from the previous step).[3] The results for Group 2 are shown in Figure 3, and are representative of the results for the other groups and in our experiments prior to the 2006 competition. For this and all other experiments in this paper except those involving data from the actual competition (for which a limited number of games are available), results are presented for four runs of five-fold cross validation (thus for each fold, 10 games are held out as the test set while a certain sized subset of the remainder is used for training). Root mean squared error is used as the measure of accuracy, and the values reported are fractions of the *base price* (a reference price based on maximum component costs) for each computer. For reference, we also determined the results of using a heuristic that performs linear regression on what TacTex-06 believes to be the average price of each computer over the past 10 days and predicts that the observed trend will continue: an average error of 0.1220 on Group 2, and similarly high error on other groups.

From these results, we can see that both learning algorithms greatly outperformed the heuristic, illustrating the difficulty of the prediction task. Additive regression outperformed M5 trees when sufficiently many games were available (and this result was statistically significant with at least 95% confidence according to paired t-tests when eight or more games were used). When only one or two games were available, M5 trees produced lower errors, but this result was not statistically significant, suggesting that the optimal choice of learning algorithm is unclear in this case and that further exploration of the issue may be needed. Nevertheless, additive regression was the only machine learning algorithm used by TacTex-06 in the 2006 competition, and it is the algorithm that will be used for the remainder of the paper.

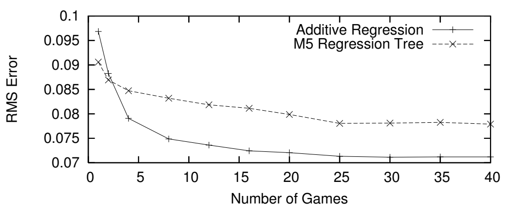

**Fig. 3.** Results of the two learning algorithms using games from Group 2

## 5.3 Comparing Results for Different Groups of Agents

From Figure 3, it appears that about 30 games are needed for training before prediction error reaches its minimum level, and about eight games before the error comes somewhat close to this level. Since a typical round of the TAC SCM competition involves 16 games, these results are somewhat concerning, as it might not be possible to learn sufficiently accurate predictors in time for them to be useful if only data from the current round is used.

We now consider the possibility of training predictors on games involving a different group (or groups) of agents. For each of the four groups of agents, we generated predictors by training on 40 games from that group and using four runs

[3] For the parameters of these two algorithms, we determined a minimum leaf size of 10 and the choice of a regression tree (not model tree) to be best for M5 trees, and a shrinkage rate of 0.7 and 200 iterations to best for additive regression.

of five-fold cross-validation as before, but each predictor generated was also evaluated on one fold of each other group, allowing the results to be directly compared for each fold as part of a paired t-test. In addition, for each group a predictor was trained on all games from the other three groups combined and evaluated for each fold of that group. Figure 4 shows the average results of evaluating each model on each group.

The most important observation from these results is that while the predictive models that give the best results for each group are those trained on that group (and this is statistically significant in each case with 99% confidence according to paired t-tests), the difference is fairly small. It appears that the differences between the agents in each group do not have a large impact on the nature of computer price trajectories. While prediction appears to be more difficult for Group 2, this difficulty seems to affect all models to a similar degree. Also, generalization from other groups to Group 4 does not appear to suffer from the fact that this group represents the most competitive economy. Finally, for each group the predictor trained on all games from the other three groups does about as well as the best of the three predictors trained on only one of these groups, if not better, suggesting that training a predictor on games from all available groups is an effective strategy when it is not known which group will give the best results. In fact, after making this observation during our experimentation prior to the competition, we chose to use this strategy to learn the predictor that TacTex-06 used throughout the competition. Because there appeared to be little variation between the results for different agents, we learned a single predictor before the start of the competition and did not adapt this predictor during the competition. The predictor was trained on all games that we ran between different groups of agent binaries available at the start of the 2006 competition.

| | *Test Data* | | | |
|---|---|---|---|---|
| *Model* | *1* | *2* | *3* | *4* |
| *heuristic* | 0.1173 | 0.1220 | 0.1074 | 0.1107 |
| *1* | *0.0606* | 0.0740 | 0.0657 | 0.0647 |
| *2* | 0.0636 | *0.0711* | 0.0676 | 0.0656 |
| *3* | 0.0641 | 0.0763 | *0.0615* | 0.0634 |
| *4* | 0.0640 | 0.0766 | 0.0637 | *0.0597* |
| *other 3* | 0.0620 | 0.0743 | 0.0641 | 0.0632 |

**Table 4.** RMS error when predictive models are learned using games from one group and tested on games from another group

## 5.4 Results of the 2006 Final Round

The results of the 2006 final round (consisting of 16 games) are shown in Table 5. Although it is difficult to assign credit for an agent's performance to particular components, an analysis of the game logs shows that TacTex-06 generally sold computers at higher prices than other agents, which would suggest that the attempt to predict changes in computer prices paid off. In fact, during the first third of each game, TacTex-06 had a higher average sales price than any opponent for every type of computer.

Figure 4 shows a comparison between the results of using a fixed predictive model (here we used the model from Section 5.3 that was trained on all games from Groups 1, 2, and 3, as Group 4 is very similar to the actual agents competing

| *Rank* | *Agent* | *Average Profit* |
|---|---|---|
| 1 | TacTex-06 | $5.85M |
| 2 | PhantAgent | $4.15M |
| 3 | Deep Maize | $3.58M |
| 4 | Maxon | $1.75M |
| 5 | Botticelli | $0.48M |
| 6 | MinneTAC | *-$2.70M* |

**Table 5.** Results of the 2006 final round

in the finals) and the results that would have been obtained by learning only from completed games. To determine the latter for $N$ completed games, we averaged the results of 20 runs in which we randomly chose $N$ games for training and used the remaining $16 - N$ games as the test set, except in the cases of $N = 1$ and $N = 15$, for which we performed 15 runs by using each game once as the training ($N = 1$) or testing ($N = 15$) set. Although we could have simply trained on the first $N$ games to give the actual results that would have been obtained during the competition, we felt that this would give results that were too noisy. Generating the results as we did also requires the assumption that game order is insignificant (i.e., no trend of changes as agents adapt over time), which appeared to be the case. The results show that the fixed predictor performed as well as or better than the alternative for at least the first 8 games, and somewhat worse afterwards.

### 5.5 Additional Experiments

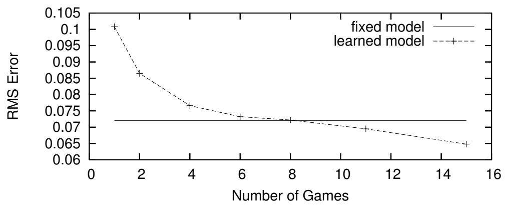

**Fig. 4.** Comparison between the fixed predictor and learning from games

In order to better measure the effect of learning to predict changes in computer prices on the performance of TacTex-06, we performed two additional experiments using variations of TacTex-06 in which this ability was weakened or removed. In each experiment, 30 games were run using the agents of Group 4 (as this group contains the four opponents from the 2006 finals for which binaries are available), except that TacTex-05 was replaced with an altered version of TacTex-06. In Experiment 1, the altered version predicted no changes in computer prices, and in Experiment 2, the altered version used the heuristic from Section 5.2 in place of the learned predictor. Table 6 shows the differences between the scores and revenues of the normal and altered versions. Differences are statistically significant with 99% confidence according to paired t-tests. The difference between scores in each case is larger than TacTex-06's margin of victory, and the difference is largely accounted for by the loss in revenue. From these results we conclude that learning to predict the changes in computer prices had a significant impact on the performance of TacTex-06 in the 2006 competition.

## 6 Additional Learning Approaches

In the previous section, we chose between using a fixed predictor trained on a variety of games from our own simulations and the alternative of learning a predictor using only the games from the current round of competition. In this section, we explore the use of more sophisticated learning approaches that make use of both sources of data.

| *Exp. #* | *Description* | *Score* | *Revenue* |
|---|---|---|---|
| 1 | no price change prediction | -4.27M | -3.05M |
| 2 | heuristic price change prediction | -1.79M | -1.21M |

**Table 6.** Experiments comparing the performance of one altered version of TacTex-06 and one unaltered version. Numbers represent the difference between the two.

One way to make use of all available game data is to train on some combination of data from the current round (which we will call "new data") and other sources (which we will call "old data" and could include games from past rounds or the simulated competition of the previous section). This type of approach has previously been applied to the TAC Travel scenario (a separate competition) [8]. The primary difficulty with this approach is deciding what the ratio of new data to old data should be. When only a few games have been played, it may be better to place more weight on old data, but as more games are played, it likely makes sense to decrease the weight of the old data until at some point only new data is used. This hypothesis is supported by Figure 4.

We address this issue by using leave-one-out cross validation to choose the fraction of old data to be added to the complete set of new data. To test a particular choice of fraction when $N$ games are available from the current round, we use each game once as the testing set while training a predictor on the combination of that fraction of old data and the remaining $N-1$ games. The fraction that produces the highest average accuracy over all $N$ trials is then chosen, and the predictor to be used is trained on all $N$ games plus that fraction of the old data. When only one game is available, we simply set the fraction to 1 and use all available old data. It is important to note that when taking a fraction of the old data, we are taking that fraction from all games, and not all data from that fraction of the games. We note that this approach may cause a larger fraction of old data to be used than is optimal because evaluations are made using predictors trained on $N-1$ games instead of the full $N$ games.

In the experiments of this section, we apply this approach of mixing data to the 2006 final round using all games from Groups 1, 2, and 3 of the previous section as the old data. To choose the fraction of old data to use at each step, we test each of 0, 1, 2, 3, 4, and 5 percent as described and choose the best. Fractions over five percent do not appear to be needed. As the old data consists of 150 games, each percent is 1.5 games worth of data. (The use of a more advanced approach to searching for the best fraction might improve accuracy somewhat at the cost of more time spent training predictors.) Results are shown in Figure 5. The fraction of old data determined to be best decreased from 5% when two games were available to 1% when 15 games were available.

Instead of combining the old and new data, another possible approach is to combine the predictors themselves into an ensemble. We present here a method that is somewhat analogous to the data combination approach – instead of finding weights for the old and new data, we find weights to be used in combining an "old predictor" and a "new predictor" through weighted averaging of their predictions. Given two predictors and a set of training data, we determine the weights of each predictor by evaluating both predictors on each training instance and performing linear regression to find the weights that best combine these outputs to match the correct labels. It is interesting to note that the weights may not sum to 1 – a sum below 1 might indicate that the changes in computer prices for a particular group of agents are less pronounced than for the groups on which the predictors were trained. Negative weights are also possible.

As with the experiments on combining training data, we apply this approach to the 2006 final round using predictors trained on games from Groups 1, 2, and 3 of the previous section as the old predictors. To determine the correct weights, we again use a form of leave-one-out cross validation. As described above, we perform linear regression on the outputs of both the old and new predictors on data from all available games; however, to determine the outputs of the new predictor for a specific game, we use a predictor trained on all games but that one. This use of cross-validation is needed to prevent overfitting: if the weight of the new predictor is determined by performing the regression step on the full new predictor itself, the new predictor will likely receive nearly all of the weight because it was trained specifically on the same data being used to learn the weights. Once the weights are determined, the full new predictor is trained on all available games and used along with the old predictor in the ensemble. When only one game is available, the old predictor is used by itself.

We are now left with the question of which predictor to use as the old predictor. Rather than using a single predictor, we will in fact use all of them: the predictors trained on each of the three groups alone along with the predictor trained on all three. The regression step described above can be performed using any number of predictors, and so we choose to perform linear regression on five variables: a weight for each of the four old predictors and a weight for the new predictor. For comparison, we also present the results of performing regression using only the four old predictors without learning a new predictor. The results of both approaches are shown in Figure 5.

We can see from the results that none of the approaches described in this section significantly outperform the fixed model for the first four games, but that both the method of combining data and the method of combining new and old predictors outperform the fixed and learned predictors when six or more games are available for training. The method of combining new and old predictors results in the lowest error, and this result is statistically significant with at least 95% confidence after at least six games have been played.

It should be noted that in the actual TAC SCM competition, the long training times of the learning approaches described in this section would be an issue, as there is only limited time between games in which to perform learning. Still, the results of this section suggest that significant improvement over the methods of the previous section should be possible.

## 7 Related Work

A number of agent descriptions for TAC SCM have been published presenting a wide variety of approaches to the tasks faced by an agent.[4] For instance, agents

[4] See `http://tac.eecs.umich.edu/researchreport.html` for a complete collection of papers on TAC agents.

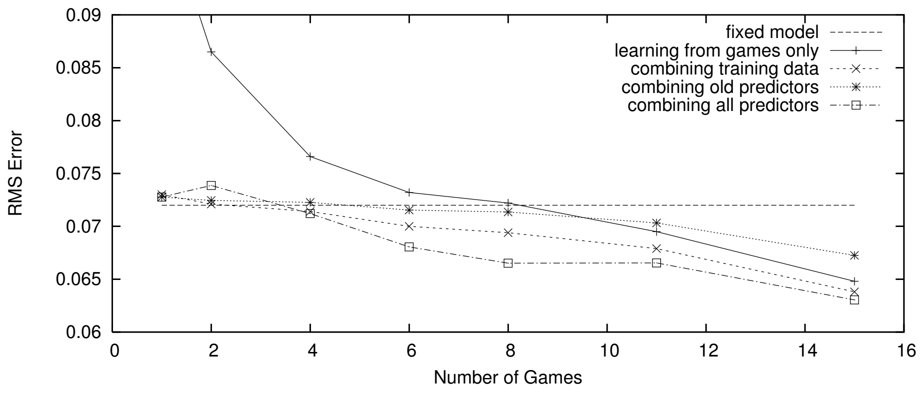

**Fig. 5.** Predictor accuracy

have addressed the problem of bidding on customer RFQs that is described in this paper by using solutions ranging from game-theoretic analysis of the economy [9] to fuzzy reasoning [10].

The learning approach in which we combine previously trained predictors is an example of an online learning method designed to make use of a number of experts, a class of methods that has received much attention and includes the weighted majority algorithm for binary classification problems [11]. Rettinger et al. [12] take a somewhat similar approach to modeling opponents in a robotic soccer task. Given a number of existing opponent models, they quickly learn a model for a new opponent by using an extension of AdaBoost in which the existing models are included among the weak learners used in the boosting process. In general, the methods described in Section 6 can be considered instances of inductive transfer or transfer learning, in which experience with one task or set of tasks is used to improve learning on another task [13].

## 8 Conclusions and Future Work

In this paper we described a number of approaches to learning to predict computer sales prices in the TAC SCM domain. The use of this prediction was shown to be an important part of the winning performance of TacTex-06 in the 2006 competition. One reason this prediction problem is difficult is that while trends in computer prices depend on opponent behavior, this behavior is difficult to model directly because little information is provided about the actions of opponents. We presented methods that addressed this difficulty by modeling the economy itself and by making use of game simulations involving a variety of opponent strategies to determine how patterns in computer prices vary for different groups of agents.

There are many ways in which this work could be extended. The effects of a wider variety of opponent behavior could be explored by designing our own agents to behave in particular ways. Many ensemble methods other than weighted averaging of predictors could be tried. It is not clear how adaptation would be affected if other agents are themselves adapting in ways that impact the economic properties being modeled.

## Acknowledgments

We would like to thank the SICS team for developing the TAC SCM game server and all teams that have contributed to the agent repository. This research was supported in part by NSF CAREER award IIS-0237699 and NSF grant EIA-0303609.

## References

1. Carmel, D., Markovitch, S.: Opponent modeling in multi–agent systems. In Weiß, G., Sen, S., eds.: Adaptation and Learning in Multi–Agent Systems. Springer-Verlag: Heidelberg, Germany (1996) 40–52
2. Hu, J., Wellman, M.P.: Online learning about other agents in a dynamic multiagent system. In Sycara, K.P., Wooldridge, M., eds.: Proceedings of the 2nd International Conference on Autonomous Agents, New York, ACM Press (1998) 239–246
3. Sadeh, N., Hildum, D., Kjenstad, D., Tseng, A.: Mascot: an agent-based architecture for coordinated mixed-initiative supply chain planning and scheduling. In Workshop on Agent-Based Decision Support in Managing the Internet-Enabled Supply-Chain, at Agents '99 (1999)
4. Chen, Y., Peng, Y., Finin, T., Labrou, Y., Cost, S.: A negotiation-based multi-agent system for supply chain management. In Workshop on Agent-Based Decision Support in Managing the Internet-Enabled Supply-Chain, at Agents '99 (1999)
5. Collins, J., Arunachalam, R., Sadeh, N., Eriksson, J., Finne, N., Janson, S.: The supply chain management game for the 2006 trading agent competition. Technical report (2005) Available from http://www.sics.se/tac/tac06scmspec_v16.pdf.
6. Pardoe, D., Stone, P.: TacTex-2005: A champion supply chain management agent. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence. (2006) 1489–94
7. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)
8. Stone, P., Schapire, R.E., Littman, M.L., Csirik, J.A., McAllester, D.: Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. Journal of Artificial Intelligence Research **19** (2003) 209–242
9. Kiekintveld, C., Wellman, M., Singh, S., Estelle, J., Vorobeychik, Y., Soni, V., Rudary, M.: Distributed feedback control for decision making on supply chains. In: Fourteenth International Conference on Automated Planning and Scheduling. (2004)
10. He, M., Rogers, A., Luo, X., Jennings, N.R.: Designing a successful trading agent for supply chain management. In: AAMAS '06: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, New York, NY, USA, ACM Press (2006) 1159–1166
11. Littlestone, N., Warmuth, M.: The weighted majority algorithm. Information and Computation **108** (1994) 212–261
12. Rettinger, A., Zinkevich, M., Bowling, M.: Boosting expert ensembles for rapid concept recall. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence. (2006)
13. Thrun, S., Pratt, L., eds.: Learning To Learn. Kluwer Academic Publishers (1997)