Online Prediction via Continuous Artificial Prediction Markets

Fatemeh Jahedpari, *University of Bath*

Talal Rahwan, *Masdar Institute of Science and Technology*

Sattar Hashemi, *Shiraz University*

Tomasz P. Michalak, *Oxford University and Warsaw University*

Marina De Vos and Julian Padget, *University of Bath*

Wei Lee Woon, *Masdar Institute of Science and Technology*

In artificial prediction markets, algorithms trade contracts that reflect their levels of confidence. To date, they've been studied in the context of offline classification, but they can be extended to online regression.

Prediction markets are exchange-traded markets created to utilize the wisdom of the crowd, whereby constantly changing prices reflect what the crowd thinks about the probability of a certain future event. Examples include the market held on www.predictit.org to predict the winner of the 2016 US presidential election and the market held on www.predictwise.com to predict the winner of the 2016 Academy Award for Best Picture. More generally, these markets are increasingly being applied by governments and corporations as a means of collecting, summarizing, and aggregating dispersed private information.

Typically, a "market maker" runs a market in which participants buy and sell contracts, with the payoff depending on the outcome of the future event under consideration. This aggregation method has recently been introduced to the machine learning community in the form of an artificial prediction market (APM) that resembles real prediction markets, except for human traders are replaced with machine learning algorithms. To date, APMs have been studied mostly in the context of offline classifications, with quite promising results.[1,2]

Inspired by this success, we extend the APM framework to facilitate its use in online regression. For brevity, we henceforth refer to the proposed method as the continuous APM (c-APM), a multiagent system in which traders are modeled as intelligent agents. c-APM has the following advantages:

- Each trader, or "agent," has an adaptive trading strategy that uses reinforcement learning to dynamically identify the actions that maximize its own reward. This resembles what happens in real prediction markets, unlike the original APM framework,

1541-1672/17/$33.00 © 2017 IEEE
Published by the IEEE Computer Society

Related Work in Artificial Prediction Market and Prediction with Expert Advice

Adrian Barbu and Nathan Lay[1,2] proposed an artificial prediction market (APM) and compared it against several machine learning models. The traders therein are trained classifiers with constant betting functions, whereas the traders in our continuous APM (c-APM) use adaptive strategies.

Amos Storkey and colleagues[3] studied the theoretical underpinnings of APMs in which participants purchase securities for possible outcomes to maximize their utility. Their work focuses on classification, whereas ours focuses on regression.

Jinli Hu and Amos Storkey[4] extended the previous works by modeling agents using static risk measures. In each round of their market, only one agent trades with the market maker. This agent observes the prices at that round, which reflect the opinions of the agents who traded earlier. As such, only the last agent can infer the wisdom of the entire crowd. In contrast, the agents in our c-APM trade multiple times, and each agent observes the wisdom of the entire crowd. Another difference is that the agents in c-APM use adaptive trading strategies. Finally, our work focuses on regression, unlike Hu and Storkey,[4] who focus on classification. To date, the only existing APM suggested for regression was proposed by Lay and Barbu,[2] whereby the algorithms are assumed to report a conditional density on the possible responses of the target variable; we relax this assumption in our c-APM.

A well-established set of techniques that are directly comparable to our c-APM can be found in the literature of prediction with expert advice (PEA).[5] Here, a PEA forecaster combines the predictions of multiple experts, hoping to obtain an aggregate performance as close to the best individual expert as possible. Each expert is viewed as a black box, or an entity that's external to the PEA model. Conversely, in c-APM, each agent is given the ability to modify its performance over the course of the market and is offered a level of autonomy to choose its preferred trading strategy. Consequently, the agents choose how much to invest in each round and how to react to the predictions of others.

There are some similarities between c-APM and standard ensembles, but in the latter, each base model makes predictions but has no opportunity to revise it in response to other predictors, while in c-APM, each agent constantly observes the predictions of other agents and participates in the market according to its observations.

References

1. A. Barbu and N. Lay, "An Introduction to Artificial Prediction Markets for Classification," *J. Machine Learning Research*, vol. 13, no. 1, 2012, pp. 2177–2204.
2. N. Lay and A. Barbu, "The Artificial Regression Market," 2012; http://arxiv.org/abs/1204.4154v1.
3. A.J. Storkey, J. Millin, and K. Geras, "Isoelastic Agents and Wealth Updates in Machine Learning Markets," *Proc. 29th Int'l Conf. Machine Learning*, 2012; http://icml.cc/2012/papers/886.pdf.
4. J. Hu and A.J. Storkey, "Multi-period Trading Prediction Markets with Connections to Machine Learning," *Proc. 31th Int'l Conf. Machine Learning*, 2014, pp. 1773–1781.
5. N. Cesa-Bianchi and G.Lugosi, *Prediction, Learning, and Games*, Cambridge Univ. Press, 2006.

in which participants use fixed strategies such as constant betting functions,[1,3] utility functions,[2] or static risk measures.[4]

- Each agent is armed with the ability to revise its prediction in response to those of other agents, thus incorporating the wisdom of the crowd. Arguably, the success of real prediction markets relies "critically on traders adjusting their beliefs in response to other traders' trades."[5]

We empirically evaluate our c-APM with multiple, diverse datasets from the widely used University of California at Irvine (UCI) Machine Learning Repository; our experiments show that c-APM outperforms several well-established alternatives from the literature on online regression (see the "Related Work in Artificial Prediction Market and Prediction with Expert Advice" sidebar).

The c-APM Model

Inspired by APMs as well as real prediction markets, c-APM includes a market maker who runs the market, processes the agents' transactions, and aggregates their predictions. Just like APMs, the market participants in c-APMs are software agents, each of which has a data source, a learning algorithm, a trading strategy, and a budget. Each agent receives data from its allocated source, then uses its learning algorithm to arrive at a prediction. Afterward, based on its trading strategy, it places a bet on its predicted outcome; this bet reflects the agent's confidence in its prediction and can't exceed the agent's allocated budget. In so doing, c-APM can aggregate multiple data sources by assigning them to different agents and can act as an ensemble by assigning a different learning algorithm to each agent.

The market itself is based on the pari-mutuel mechanism, which works as follows: first, participants place bets on a possible outcome. Then, the probability of each outcome is taken as the total bet on that outcome, divided by the total bet on all outcomes. Once the market is closed, the real outcome is revealed, and the "pot" is divided among the winners proportional to the amount they each wagered. This mechanism was originally used for predicting a discrete set of outcomes, but we extend it for the prediction of continuous variables. Importantly, to allow the agents to update their predictions and investments using crowdsourced information, we generalize the pari-mutuel mechanism from one to multiple rounds. More specifically, in each round, the agents send their predictions and bets to the market maker, who combines the predictions using

```
Input: b
Output: c-APM Prediction
1: Each agent receives an equal initial budget, b;
2: for every example, x, in the data set do
3:     The market maker instantiates a prediction market for x;
4:     Each agent observes (some) features of x;
5:     for each round do
6:        for each agent a_i do
7:           Compute and submit: (prediction_i, bet_i);
8:           budget_i ← budget_i - bet_i;
9:        end for
10:       Market maker aggregates the market prediction (using
            the aggregation function) and announces it;
11:    end for
12:    c-APM prediction ← market prediction of the final round;
13:    The market maker reveals the true outcome;
14:    for each agent a_i do
15:       Calculate reward_i (using the reward function);
16:       budget_i ← budget_i + reward_i;
17:       Update learning algorithm & trading strategy;
18:    end for
19:end for
```

Algorithm 1. c-APM.

an aggregation function and computes the payouts using a reward function (see Algorithm 1).

Reward Function

Once the true outcome is revealed, each agent receives a reward determined by the reward function, whereby for each bet of agent a_i, the revenue is computed as

$$revenue_i = score_i \times bet_i, \quad (1)$$

where

$$score_i = \max\{\ln(accuracy_i), 0\}; \quad (2)$$

$$accuracy_i = \max\left\{100\left(1 - \frac{|outcome - prediction_i|}{\text{outlier error thresohld}}\right), \varepsilon\right\}. \quad (3)$$

Before explaining the rationale behind Equation 2, we first explain how the accuracy of a_i's prediction is calculated. According to Equation 3, $accuracy_i$ is a real number in (ε, 100], which decreases linearly as the prediction error of a_i increases. In particular, whenever the error is equal to 0, we have $accuracy_i = 100$. In contrast, whenever the error approaches the outlier error threshold (which was calculated using the interquartile-range), we have $accuracy_i = \varepsilon$.

Importantly, by ensuring that the accuracy is always greater than 0, we can use it in the logarithmic function in Equation 2. With this equation, the accuracy is mapped to a score using the *logarithmic scoring rule*, an incentive-compatible scoring function used widely in the prediction market literature. This way, any accuracy greater than e receives a score greater than 1 (because $\ln(e) = 1$), resulting in a positive reward, that is, a revenue greater than the investment (see Equation 1) and vice versa. Note that the accumulated revenue of an agent is unbounded, so a perfect predictor would come to dominate the market. However, with a large rate per transaction, even a single poor prediction could cause such an agent to lose its dominance.

Aggregation Function

The market maker uses this function to aggregate n bets—one for each agent—resulting in the market prediction. This is done by assigning more weight to predictions that are backed by higher investments, because the level of investment reflects the level of confidence that the agent has in its own prediction. Formally, the market prediction, denoted by *prediction*, is defined as

$$prediction = \frac{\sum_{i=1}^{n} prediction_i \times bet_i}{\sum_{i=1}^{n} bet_i}. \quad (4)$$

Recall that Equation 1 allows the agents with accurate predictions to accrue more revenue over time. Because this revenue is added to the agent's budget (line 16 of Algorithm 1), agents with relatively high performance accumulate greater budgets, which in turn allows them to make greater investments, leading to even greater budgets, and so on. Based on this, as well as Equation 4, the participants with a history of accurate predictions can feed their proficiency into the market prediction.

Rate per Transaction Parameters

In c-APM, every bet is constrained by two parameters, namely, the *minimum rate per transaction* (MinRPT) and the *maximum rate per transaction* (MaxRPT), which specify the minimum and maximum percentage of the agent's budget that can be placed in a single bet, respectively.

Without a MinRPT, agents could find themselves in a situation in which none of them has any incentive to invest, meaning that they each place a bet of 0. In such a case, c-APM simply wouldn't return any outcome. Thus, by setting MinRPT to a (small) value greater than 0, we prevent such an undesirable outcome.

Without a MaxRPT, some agents could go bankrupt, which is undesirable because it could mean the loss of those agents' data sources (if no other agent has access to those sources) and the loss of any insight that those

agents' learning algorithms provide (if no other agents use those algorithms). Note that the bankruptcy of an agent, a_i, doesn't necessarily imply that a_i can't bring any value to the collective performance, because a_i's prediction could improve in subsequent markets. By setting MaxRPT to a value smaller than 100 percent, we ensure that a_i's budget never reaches 0, which leaves the door open for a_i to regain its competitiveness in the market whenever the opportunity arises. Another advantage of having MaxRPT is to control the extent to which agents' original prediction quality influences their budgets. More specifically, increasing MaxRPT leads to larger bets during the early rounds of the market, so, for example, setting MaxRPT to 90 percent would encourage the agents to invest the majority of their original budgets during the first round, leaving only 10 percent for all subsequent rounds. This way, the quality of the agents' original predictions (those made during the first round, before observing the crowd's aggregate prediction) would significantly influence their accumulated budgets. Based on this observation, we recommend setting MaxRPT to a large value in the first round and then decreasing it in subsequent rounds.

Agent Trading Strategy

In this section, we examine the following trading strategies:

- *constant trading*, in which the agent invests a fixed percentage of its budget in each round, implying that the agent doesn't reflect upon the wisdom of the crowd; and
- *Q-learning*, in which the invested amount is optimized using Q-learning, a standard reinforcement learning technique designed to find an action-selection policy that maximizes the agent's reward, given a Markov decision process.

```
Input: Prediction, prediction_i, round
Output: prediction_i, bet_i.
1: if this is not the first market nor the first round then
2:  estError ← Prediction - prediction_i;
3:  state ← identifyState(estError, round);
4:  action ← argmax_a∈{Change,Preserve} Q(state, a);
5:    if action = Change then
6:   prediction_i ← prediction_i + (δ_round × estError);
7:    end if
8:    if score'_i < 1 then
9:   bet_i ← budget_i × MinRPT;
10:    else
11:   bet_i ← budget_i × MaxRPT;
12:    end if
13: else
14:  bet_i ← budget_i × MaxRPT;
15: end if
16: return <prediction_i, bet_i>
```

Algorithm 2 . Q-learning trading strategy for agent a_i.

Because the former strategy is rather clear, we only discuss the latter (see Algorithm 2), which checks whether it's either the first market or the first round (line 1). If the answer is yes, then the agent can't use Equation 6, as it requires either the outlier error threshold of the previous market or the market's prediction from the previous round. Consequently, the agent can't estimate its score, implying that its best option is to simply invest MaxRPT% of its budget (line 14). On the other hand, if the answer is no, the agent estimates its error as shown in line 2, where *prediction* denotes market prediction (see Equation 4). As can be seen, the error estimation is done by comparing a given prediction against the market's prediction rather than against the true outcome, as this isn't yet known. After that, the agent identifies its own state based on its estimated error and the current round number (line 3). The agent then considers two actions and chooses the one with the highest Q-value: preserve the current prediction or change the prediction to minimize the estimated error according to a parameter, δ, which reflects the agent's level of confidence in the wisdom of the crowd (lines 5 and 6). The agent then estimates its score as

$$score'_i = \max\left\{\ln\left(accuracy'_i\right), 0\right\}, \quad (5)$$

where

$$accuracy'_i = \max\left\{100\left(1 - \frac{\left|Prediction - prediction_i\right|}{\text{outlier error threshold}}\right), \varepsilon\right\}. \quad (6)$$

Essentially, these two equations are the same as Equations 2 and 3, except that the true outcome is replaced by the market prediction, and the *outlier error threshold* now refers to the previous market rather than the current one (because the true outcome isn't yet known at this stage). Since the agent's estimated revenue equals $score'_i$, then if $score'_i < 1$, the agent would have a loss, and thus must set its bet to be as small as possible (lines 8 and 9). On the other hand, if $score'_i \geq 1$, the agent sets its bet to be as large as possible (lines 10 and 11).

Identifying the State

Each state is a pair consisting of the estimated error and the number of rounds. As is commonly done for cases with low dimensional state spaces, we discretize and map every estimated error to one of these clusters: small, medium, or large. Because the scale of errors can change from

one market to another, each agent recomputes the decision boundaries of these clusters at the end of each market; this is done using the online version of k-means clustering.[6]

Updating the Q-Values

Once the true outcome of each market is known, the agents update their Q-values as follows. Each agent, a_i, computes the revenue that it could earn from each action in each state. Consequently, for each state, s, that the agent has visited and each action, a, that the agent could have made in that state, the Q-value is updated as follows, where α is the learning rate:

$$Q(s,a) \leftarrow (1-\alpha)\,Q(s,a) + \alpha \text{ revenue}. \quad (7)$$

Since c-APM is proposed for supervised learning where agents can access the correct answer to update their beliefs, agents can update not only the Q-value of the executed actions in each visited state but also the Q-value of other actions. As such, agents can follow the greedy policy, instead of trading off exploitation for exploration.

Updating the Confidence in the Crowd

At the end of each market, after the true outcome is revealed, each agent updates δ, the parameter reflecting its confidence in the wisdom of the crowd. This parameter, which is always between 0 and 1, is computed for every state that the agent has visited as follows:

$$\delta_s = (1-\alpha)\,\delta_s + \alpha \left[truncate\left(\frac{outcome - prediction_{i,s}}{prediction_s - prediction_{i,s}} \right) \right], \quad (8)$$

where α is the learning rate, $prediction_{i,s}$ denotes agent a_i's prediction at state s, $prediction_s$ denotes the market prediction at state s, and $truncate$: $\mathbb{R} \rightarrow [0,1]$ is defined for every real value $r \in R$ as follows:

$$truncate(r) = \left\{ \begin{array}{ll} 0, & \text{if } r < 0 \\ 1, & \text{if } r > 1 \\ r, & \text{otherwise} \end{array} \right\}.$$

Experiments

We randomly chose 10 UCI datasets from those identified as suitable for regression—namely, bike sharing, auto MPG, yacht hydrodynamics, Istanbul Stock Exchange, servo, forest fires, automobile, housing, airfoil self-noise, and computer hardware.

For each dataset, records were presented one at a time, and the models were re-trained every time. Records with missing values were discarded. The first five records were used to initialize the online predictors, and hence don't contribute to the prediction accuracy calculations.

We worked with a variety of models from the R language's widely used caret package (version 6.0-37); these are listed in Figure 1. Any parameters of those models were kept at their default values. We constructed a c-APM in which every agent has a unique learning algorithm corresponding to one of the above. As discussed earlier, the parameters of the c-APM were set as follows: MinRPT = 0.01 percent and MaxRPT = 90 percent in the first round; MinRPT = 0.01 percent and MaxRPT = 1 percent in subsequent rounds. We experimented with two variations of c-APM: constant trading and Q-learning strategy based.

We used four popular prediction with expert advice (PEA) models, each based on weighted averages of the individual predictions as follows:

$$p_t = \frac{\sum_{i=1}^{n} w_{i,t-1} f_{i,t}}{\sum_{j=1}^{n} w_{j,t-1}}, \quad (9)$$

where p_t is the model's prediction at time t; n is the number of experts used; $w_{i,t}$ is the weight for expert i at time t; and $f_{i,t}$ is the prediction of expert i at time t. The different models vary in the way they calculate the weights $w_{i,t}$ as specified below.

For the exponentially weighted average forecaster (EWAF),[7] the weights are calculated recursively as

$$w_{i,t} = w_{i,t-1} exp\left(-\eta L(f_{i,t}, y_t)\right), \quad (10)$$

where $L(f_{i,t}, y_t)$ is the loss for prediction $f_{i,t}$ given the true output y_t (in our experiments, this was taken as the squared error), whereas η is a scaling parameter that determines the sensitivity of the weights toward the loss.

For tracking the best expert (TBE),[8] the weights are first computed as in Equation 10. We subsequently used the "fixed-share" variant of this method, whereby the weights are updated as

$$w_{i,t} = (1-\alpha) w_{i,t} + \sum_{j \neq i} \frac{\alpha}{n-1} w_{j,t}. \quad (11)$$

For following the best expert (FBE),[9] the prediction is calculated as

$$p_t = f_{E,t},$$
$$\text{where } E = \arg\min_{i \in \mathcal{E}} \sum_{t=1}^{T-1} L(f_{i,t}, y_t). \quad (12)$$

That is, at any point in time, FBE simply uses the expert with the lowest total loss thus far.

Finally, for the exponentiated gradient (EG) algorithm,[10] the weights are calculated using the gradient of the model's loss as

$$w_{i,t} = w_{i,t-1} exp\left(-\eta L'(p_t, y_t) f_{i,t}\right). \quad (13)$$

In this model, a typical learning rate could be $\eta = 2/(3R^2)$, where R is an upper bound on the maximum difference between the predictions of different experts at time t.[10]

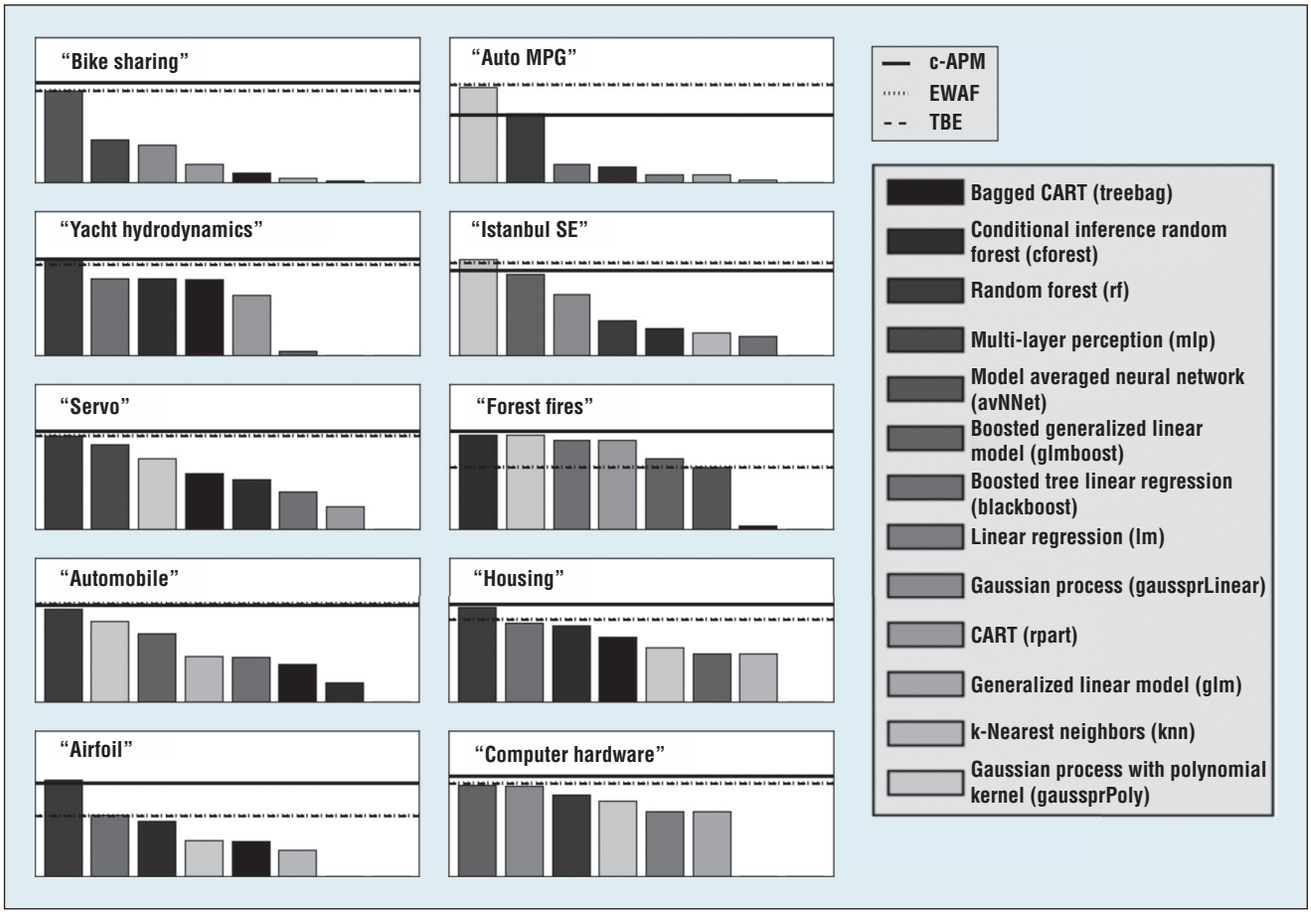

Figure 1. Performances of c-APM, Exponentially Weighted Average Forecaster (EWAF), and Tracking the Best Expert (TBE) with respect to the top eight individual component predictors (relative to each dataset). We benchmark against EWAF and TBE, as these were the two top performing prediction with expert advice (PEA) predictors. Here, the higher the value, the better.

Table 1. Mean square error for c-APM and PEA predictors on UCI datadets.*

Aggregator	Bike sharing	Auto MPG	Yacht hydrodynmaics	Istanbul Stock Exchange	Servo	Forest fires	Automobile	Housing	Airfoil	Computer hardware
c-APM (constant)	1.77e+06	9.80e+00	1.15e+01	1.47e-04	6.17e-01	4.14e+03	1.24e+07	1.59e+01	9.74e+00	5.87e+03
c-APM (Q-learning)	1.47e+06	9.69e+00	6.96e+00	1.49e-04	6.21e-01	4.10e+03	8.79e+06	1.52e+01	7.56e+00	4.36e+03
EWAF	2.38e+06	8.52e+00	1.14e+01	1.47e-04	6.53e-01	4.22e+03	8.43e+06	1.89e+01	1.32e+01	5.02e+03
FBE	2.38e+06	8.75e+00	7.43e+00	1.50e-04	7.42e-01	4.19e+03	1.12e+07	2.07e+01	7.00e+00	6.33e+03
TBE	2.38e+06	8.52e+00	1.14e+01	1.47e-04	6.53e-01	4.22e+03	8.43e+06	1.89e+01	1.32e+01	5.02e+03
EG	1.11e+05	9.77e+00	1.48e+01	1.49e-04	6.43e-01	4.35e+03	5.52e+07	1.78e+01	1.45e+01	6.74e+03

*For each column, the MSE for the best model is highlighted in dark red, whereas the second best is highlighted in orange. Here, the lower the value, the better.

In our experiments, as is the case with c-APM, every PEA forecaster was given a single expert for each of the aforementioned learning algorithms.

Results

We evaluated and verified c-APM's performance in terms of the prediction accuracy and comparison with individual predictors.

Specifically, prediction accuracy was evaluated in terms of the mean squared error (MSE). To ensure a fair comparison, we tested a range of parameter values for both c-APM and the

THE AUTHORS

Fatemeh Jahedpari is a lecturer in University of Wollongong in Dubai. She received a PhD in computer science from the University of Bath. Her research interests include machine learning and multiagent systems. Contact her at jahedpari@acm.org.

Talal Rahwan is an assistant professor at the Masdar Institute of Science and Technology. Rahwan received the British Computer Society's Distinguished Dissertation award and was selected by the IEEE Computer Society as one of the top 10 young artificial intelligence researchers in the world. Contact him at trahwan@masdar.ac.ae.

Sattar Hashemi is an associate professor in the Electrical and Computer Engineering School at Shiraz University. His research interests include machine learning, social networks, data stream mining, game theory, and adversarial learning. Hashemi received a PhD in computer science from the Iran University of Science and Technology in conjunction with Monash University, Australia. Contact him at s_hashemi@shirazu.ac.ir.

Tomasz P. Michalak is a part-time postdoctoral researcher in the Computer Science Department, University of Oxford, and lecturer at the Faculty of Mathematics, Informatics, and Mechanics, University of Warsaw. His research falls at the intersection of artificial intelligence and economics. Michalak received a PhD in economics from the University of Antwerp. Contact him at tomasz.michalak@cs.ox.ac.uk.

Marina De Vos is a senior lecturer in the Department of Computer Science at the University of Bath. Her research areas are knowledge representation and reasoning and multiagent systems. De Vos received a PhD in computer science from the Vrije Universiteit Brussel, Belgium. Contact her at m.d.vos@bath.ac.uk.

Julian Padget is a senior lecturer at the University of Bath. His research interests include intelligent agents, with a particular interest in norm representation and reasoning. Padget received a PhD in computer science from the University of Bath. Contact him at j.a.padget@bath.ac.uk.

Wei Lee Woon is an associate professor in the Electrical Engineering and Computer Science Department at the Masdar Institute of Science and Technology. His research interests include the analysis of large, complex datasets and include time series prediction, text mining, and technology forecasting. Contact him at wwoon@masdar.ac.ae.

PEA predictors. For c-APM, the number of rounds $\in \{1, 3, 10, 30\}$ and $\alpha \in \{0.1, 0.7, 1\}$; for EWAF, $\eta \in \{0, 50, 500, 5000\}$; and for TBE, $\eta \in \{0, 50, 500, 5000\}$ and $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$.

For each metapredictor, we counted the number of times that each configuration (parameter combination) resulted in the lowest prediction error. The configuration that had the highest count was then selected as the representative configuration for this predictor. Table 1 shows the results:

- c-APM with Q-learning is the only one to be ranked at number one in 4 out of 10 datasets, and it's also the only one to appear among the top two in 9 out of 10 datasets.
- When comparing the two c-APM trading strategies (constant and Q-learning based), we find that Q-learning outperforms the constant strategy in 8 out of 10 datasets. The two exceptions were the Istanbul Stock Exchange and servo datasets. Even in these two cases, we see that the Q-learning-based market is very close to the one with the constant strategy.
- Each metapredictor produces the lowest error on at least one occasion. This suggests that the chosen set of component predictors and test datasets were sufficiently diverse, and that each metapredictor had distinct advantages and disadvantages.

To evaluate the effectiveness of c-APM in aggregating predictions from different sources, we compared its prediction accuracy against individual predictors. The best PEA metapredictors, namely, EWAF and TBE, are also included as benchmarks. As the range of MSE differs significantly across various datasets, for better comparison, we present the performance of each model as a renormalized form of the MSE, computed as follows:

$$performance = \frac{MSE_{max} - MSE}{MSE_{max} - MSE_{min}}. \quad (14)$$

Figure 1 shows the results in 10 subplots, one for each dataset. Performance of each individual predictor is depicted using a bar chart, with items sorted in decreasing order. Although there are 13 predictors in total, predictors outside of the top 8 performed very poorly and were excluded from the figure (however, during the experiment, the metapredictors were presented with all 13 predictions in all cases). For c-APM, EWAF, and TBE, accuracy is depicted as a horizontal line within each subplot:

- c-APM appears to be extremely effective at combining the outputs of multiple machine learning predictors. In 8 out of 10 cases, it closely matches or visibly exceeds the best individual predictor. The two exceptions are the auto MPG and Istanbul Stock Exchange datasets, although in both cases, c-APM still matches the second best individual predictor—a good result given that it has 13 predictors to pick from.
- The ordering of the individual predictors varies significantly over the 10 datasets. The random forest algorithm is the strongest, with 5 out of 10 "wins," yet it doesn't dominate (for example, the Gaussian process predictor is a strong second, with 3 out of 10 wins). This shows how difficult it is to correctly weight the predictions from the 13 different predictors.
- Certainly, it's clear that the overall performance of c-APM with

Q-learning is better than that of any individual predictor. This is particularly valuable in the context of online learning, where it often isn't possible to choose the best predictor in advance.

Finally, we conducted experiments to determine the agents' convergence characteristics. Our results showed that as the number of markets exceeds 10, all agents' predictions converge to a single value. Unfortunately, the corresponding plots were omitted due to space constraints.

For future work, we intend to study different trading strategies, different attitudes toward risk, and different market mechanisms, such as auctions. ■

Acknowledgments

Tomasz Michalak was supported by the European Research Council under Advanced Grant 291528 (RACE). An extended abstract of this work appeared at the IJCAI Doctoral Consortium, 2015; it didn't discuss the model in detail, nor did it provide any experimental results. Another preliminary version appeared at the CEUR Workshop, 2014. In that version, the model focused on classification (rather than regression), the agents used zero intelligence plus (rather than Q-learning), and the experiments were restricted to a single application domain, namely, syndromic surveillance (instead of covering datasets from different domains).

References

1. A. Barbu and N. Lay, "An Introduction to Artificial Prediction Markets for Classification," *J. Machine Learning Research*, vol. 13, no. 1, 2012, pp. 2177–2204.
2. A.J. Storkey, J. Millin, and K. Geras, "Isoelastic Agents and Wealth Updates in Machine Learning Markets," *Proc. 29th Int'l Conf. Machine Learning*, 2012; http://icml.cc/2012/papers/886.pdf.
3. N. Lay and A. Barbu, "The Artificial Regression Market," 2012; http://arxiv.org/abs/1204.4154v1.
4. J. Hu and A.J. Storkey, "Multi-period Trading Prediction Markets with Connections to Machine Learning," *Proc. 31th Int'l Conf. Machine Learning*, 2014, pp. 1773–1781.
5. S. Dimitrov and R. Sami, "Non-myopic Strategies in Prediction Markets," *Proc. 9th ACM Conf. Electronic Commerce*, 2008, pp. 200–209.
6. J. MacQueen, "Some Methods for Classification and Analysis of Multi Variate Observations," *Proc. 5th Berkeley Symp. Mathematical Statistics and Probability*, vol. 1, 1967, pp. 281–297.
7. N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*, Cambridge Univ. Press, 2006.
8. M. Herbster and M.K. Warmuth, "Tracking the Best Expert," *Machine Learning*, vol. 32, no. 2, 1998, pp. 151–178.
9. H. Robbins, "Asymptotically Subminimax Solutions of Compound Statistical Decision Problems," *Herbert Robbins Selected Papers*, Springer, 1985, pp. 7–24.
10. J. Kivinen and M.K. Warmuth, "Exponentiated Gradient versus Gradient Descent for Linear Predictors," *Information and Computation*, vol. 132, no. 1, 1997, pp. 1–63.