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 Michalak, Oxford University and Warsaw University
Marina De Vos, University of Bath
Julian Padget, University of Bath
Wei Lee Woon, Masdar Institute of Science and Technology

***Abstract*—Prediction markets are well-established tools for aggregating information from diverse sources into accurate forecasts. Their success has been demonstrated in a wide range applications, including presidential campaigns, sporting events and economic outcomes. Recently, they have been introduced to the machine-learning community in the form of *Artificial Prediction Markets*, whereby algorithms trade contracts reflecting their levels of confidence. To date, those markets have mostly been studied in the context of offline classification, with quite promising results. We extend those markets to enable their use in online regression, and introduce: (i) adaptive trading strategies informed by individual trading history; and (ii) the ability of participants to revise their predictions by reflecting upon the wisdom of the crowd, which is manifested in the collective performance of the market. We empirically evaluate our model using multiple UCI data sets, and show that it outperforms several well-established techniques from the literature on online regression.**

***Index Terms*—Learning (artificial intelligence), Machine learning, Multi-agent systems, Prediction algorithms, Supervised learning.**

Introduction

Prediction markets are exchange-traded markets created to utilize the wisdom of the crowd, whereby the constantly-changing prices reflect what the crowd thinks about the probability of a certain future event. Examples include the market currently being held on `www.predictit.org` to predict the winner of the 2016 U.S. presidential election, and the market that was 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 the market, whereas the participants buy and sell contracts, the payoff of which depends on the outcome of the future event under consideration. This aggregation method has been recently introduced to the machine-learning community in the form of an *Artificial Prediction Market (APM)*, which resembles real prediction markets, except for replacing the human traders 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 their success, we extend the APM framework to facilitate their use in online regression. For brevity, we henceforth refer to the proposed method as the *continuous* Artificial Prediction Market (c-APM)—a multiagent system in which the traders are modeled as intelligent agents; it 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 in which participants use fixed strategies such as, e.g., 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 data sets from the widely-used UCI Machine Learning Repository; our experiments show that the c-APM outperforms several well-established alternatives from the literature on online regression.

Related Work

Artificial Prediction Markets: Barbu and Lay [1], [3] proposed an artificial prediction market, and compared it against several machine learning models. The traders therein are trained classifiers with constant betting functions, whereas the traders in our c-APM have adaptive strategies.

Storkey et al. [2] 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.

Hu and 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 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 their work which focuses on classification. To date, the only existing APM suggested for regression was proposed by Lay and Barbu [3], 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.

Prediction with Expert Advice: 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) [6]. 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. Here, each expert is viewed as a black box, or an entity that is 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.

Existing Ensembles: There are some similarities between c-APM and standard ensembles; however, in the latter each base model makes predictions but has no opportunity to revise their prediction 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.

THE C-APM MODEL

Inspired by APMs as well as real prediction markets, c-APMs 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: (i) a data source; (ii) a learning algorithm; (iii) a trading strategy; and (iv) a budget. Each agent receives data from its allocated source, then uses its learning algorithm to arrive at a prediction. Afterwards, based on its trading strategy, it places a bet on its predicted outcome; this bet reflects the agent's confidence in its prediction, and cannot exceed the agent's allocated budget. In so doing, the 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; however, we extend it for the prediction of continuous variables. Importantly, to allow the agents to update their predictions and investments using crowd-sourced 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 an **aggregation function** and computes the payouts using a **reward function** (see Algorithm 1). These functions will be described next.

Algorithm 1 c-APM

Input: b
Output: c-APM Prediction

```
Each agent receives an equal initial budget, b;
for every example, x, in the data set do
    The market maker instantiates a prediction market for x;
    Each agent observes (some) features of x;
    for each round do
        for each agent a_i do
            Compute and submit: ⟨prediction_i, bet_i⟩;
            budget_i ← budget_i − bet_i;
        end for
        Market maker aggregates the market prediction (using the
        aggregation function) and announces it;
    end for
    c-APM prediction ← market prediction of the final round;
    The market maker reveals the true outcome;
    for each agent a_i do
        Calculate reward_i (using the reward function);
        budget_i ← budget_i + reward_i;
        Update learning algorithm & trading strategy;
    end for
end for
```

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:

$$revenue_i = score_i \times bet_i, \tag{1}$$

where

$$score_i = \max\left\{\ln\left(accuracy_i\right), 0\right\}; \tag{2}$$

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

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 $(\varepsilon, 100]$, 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 (since ln(e) = 1), resulting in a positive reward, i.e., a revenue greater than the investment (see Equation 1) and vice versa. Note that the accumulated revenue of an agent is unbounded and that hence, 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, since 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:

$$\text{Prediction} = \frac{\sum_{i=1}^{n} prediction_i \times bet_i}{\sum_{i=1}^{n} bet_i} \tag{4}$$

Recall that Equation (1) allows the agents with accurate predictions to accrue more revenue over time. Now since 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. Next, we explain the rationale behind using those parameters.

First, without a *minimum* rate per transaction (MinRPT), the agents may find themselves in a situation where none of them has any incentive to invest, meaning that they each place a bet of 0. In such a case, c-APM would simply not return any outcome. Thus, by setting MinRPT to a (small) value greater than 0, we prevent such an undesirable outcome.

Second, without a *maximum* rate per transaction (MaxRPT), some agents may go bankrupt. This is undesirable as 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 agent uses those algorithms). Note that the bankruptcy of an agent, a_i, does not necessarily imply that a_i cannot bring any value to the collective performance, since a_i's prediction may improve in subsequent markets. By setting MaxRPT to a value smaller than 100%, 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 the agents' original prediction quality influences their budgets. More specifically, increasing MaxRPT leads to larger bets during the early rounds of the market, e.g., setting MaxRPT = 90% would encourage the agents to invest the majority of their original budgets during the first round, leaving only 10% for all subsequent rounds. This way, the quality of the agents' original predictions (i.e., 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 MaxRPT in subsequent rounds.

Agent Trading Strategy

In this section, we examine the following trading strategies:

1) **Constant Trading:** the agent invests a fixed percentage of its budget in each round, implying that the agent does not reflect upon the wisdom of the crowd;
2) **Q-learning:** 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.

Since the former strategy is rather clear, we only discuss the latter one (see Algorithm 2). This strategy checks whether it is either the first market or the first round (line 1). If the answer is "YES", then the agent cannot 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 cannot 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 the market prediction (see Equation 4). As can be seen, the error estimation is done by comparing one's prediction against the market's prediction rather than against the true outcome as this is not 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; these actions are: (i) *Preserve* the current prediction; and (ii) *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:

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

where

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

Essentially, the above 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 (since the true outcome is not yet known at this stage). Since the agent's estimated revenue equals $score'_i \times bet_i$, then if $score'_i < 1$, the agent would make a loss, and so 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" and "Large". Since the scale of errors may 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 [7].

Updating the Q-values: Once the true outcome of each market is known, the agents update their Q-values as follows.

Algorithm 2 Q-learning Trading Strategy of agent a_i

Input: *Prediction*, $prediction_i$, *round*
Output: $prediction_i$, bet_i.

```
if this is not the first market nor the first round then
    estError ← Prediction − prediction_i;
    state ← identifyState(estError, round);
    action ← argmax_{a∈{Change,Preserve}} Q(state, a);
    if action = Change then
        prediction_i ← prediction_i + (δ_round × estError);
    end if
    if score'_i < 1 then
        bet_i ← budget_i × MinRPT;
    else
        bet_i ← budget_i × MaxRPT;
    end if
else
    bet_i ← budget_i × MaxRPT;
end if
return ⟨prediction_i, bet_i⟩
```

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:

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

Since c-APM is proposed for supervised learning where the 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-values 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:

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

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 as for every real value $r \in R$ as follows:

$$truncate(r) = \begin{cases} 0, & \text{if } r < 0 \\ 1, & \text{if } r > 1 \\ r, & \text{otherwise} \end{cases}$$

Experiments

The following subsections describe the data and experimental procedures used to benchmark c-APM against various alternatives from the literature.

Data: We randomly chose ten UCI datasets from those identified as suitable for regression, namely: (i) Bike Sharing, (ii) Auto MPG, (iii) Yacht Hydro-dynamics, (iv) Istanbul Stock Exchange, (v) Servo, (vi) Forest Fires, (vii) Automobile, (viii) Housing, (ix) Airfoil Self Noise, and (x) Computer Hardware.

For each data set, 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 do not contribute to the prediction accuracy calculations.

Individual Predictors: We worked with a variety of models from R'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% and MaxRPT = 90% in the first round; MinRPT = 0.01% and MaxRPT = 1% in subsequent rounds. We experimented with two variations of c-APM: constant trading and Q-learning strategy based.

PEA Prediction: Four popular Prediction with Expert Advice (PEA) models were used, each based on weighted averages of the individual predictions as follows:

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

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:

i) *Exponentially Weighted Average Forecaster* (EWAF) [6]: Here, the weights are calculated recursively as follows:

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

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 towards the loss.

ii) *Tracking the Best Expert(TBE)* [8]: Here, 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 follows:

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

iii) *Following the Best Expert (FBE)* [9]: Here, the prediction is calculated as follows:

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

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

iv) *Exponentiated Gradient algorithm (EG)* [10]: Here, the weights are calculated using the *gradient* of the model' loss as follows:

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

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].

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

The performance of c-APM was evaluated and verified in terms of the *Prediction Accuracy* and *Comparison with Individual Predictors*. Each of these will now be presented in the following subsections.

Prediction Accuracy: The prediction accuracy was evaluated in terms of the Mean Squared Error (MSE). To ensure a fair comparison, a range of parameter values for both c-APM and the PEA predictors were tested. Specifically, for c-APM: number of rounds $\in \{1, 3, 10, 30\}$ and $\alpha \in \{0.1, 0.7, 1\}$; for EWAF: $\eta \in \{0, 50, 500, 5000\}$; for TBE: $\eta \in \{0, 50, 500, 5000\}$ and $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$.

For each meta-predictor, the number of times that each configuration (parameter combination) resulted in the lowest prediction error was counted. The configuration that had the highest count was then selected as the representative configuration for this predictor. The results are shown in Table I. The main observations are:

1) c-APM with Q-learning is the only one to be ranked top in 4 out of 10 data sets, and is also the only one to appear among the top two in 9 out of 10 data sets.
2) 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 data sets. The two exceptions were the *Istanbul Stock Exchange* and *Servo* data sets. Even in these two cases, we see that the Q-Learning based market is very close to the one with constant strategy.
3) Each meta-predictor produces the lowest error on at least one occasion. This suggests that the chosen set of component predictors and test data sets were sufficiently diverse, and that each meta-predictor had distinct advantages and disadvantages.

Comparison with Individual Predictors: Next, we evaluate the effectiveness of c-APM in aggregating predictions from different sources by comparing the prediction accuracy of c-APM against the individual predictors. The best PEA meta-predictors, namely EWAF and TBE, are also included as benchmarks. As the range of MSE differs significantly across various data sets, for better comparison, we present the performance of each model as a re-normalized form of the MSE, computed as follows:

$$Performance = \frac{\mathrm{MSE}_{\max} - \mathrm{MSE}}{\mathrm{MSE}_{\max} - \mathrm{MSE}_{\min}} \tag{14}$$

The results are presented in Figure 1. As can be seen, the figure consists of 10 subplots, one for each data set. The performance (as defined above) of each individual predictor is depicted using a bar chart, which is sorted in decreasing order. Although there are 13 predictors in total, predictors outside of the top 8 were performing very poorly and were excluded from the figure (however, during the experiment, the meta-predictors were presented with all 13 predictions in all cases). For c-APM, EWAF and TBE, the accuracy is depicted as a horizontal line within each subplot. Our observations are as follows:

1) 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, though in both these cases c-APM still matches the second best individual predictor, a good result given that it has 13 predictors to pick from.
2) 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 does not 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 weigh the predictions from the 13 different predictors.
3) Certainly, it is 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 is often not possible to choose the best predictor in advance.

Finally, experiments were conducted to determine the convergence characteristics of the agents; these showed that as the number of markets exceed 10, all agents converge. Unfortunately, the corresponding plots were omitted due to space constraints.

CONCLUSION

We proposed the Continuous Artificial Prediction Market (c-APM) as a new online machine-learning technique, which aggregate the predictions of an arbitrary set of learning algorithms to provide a collective outcome. c-APM (with Q-learning) was the only meta-predictor to be ranked top in 4 out of 10 data sets, and in the top two in 9 out of 10 data sets. When compared against the individual predictors, c-APM matched or outperformed the best predictor in 8 out of 10 data sets. This shows that c-APM is an effective aggregation strategy that can often extract the best outcome from a group of diverse predictors where no one single predictor dominates.

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

Acknowledgements:
Tomasz Michalak was supported by the European Research Council under Advanced Grant 291528 (RACE).

An extended abstract of this work appeared in the *IJCAI Doctoral Consortium 2015*; it did not discuss the model in detail, nor did it provide any experimental results. Another preliminary version appeared in 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).

Aggregator	Bike Sharing	Auto MPG	Yacht HD	Istanbul SE	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

TABLE I: MSE for c-APM and PEA predictors on UCI data sets. For each meta-predictor, only the results for the overall best set of parameters are presented. These are as follows: c-APM (Q-Learning): 30 Rounds, $\alpha = 1$; EWAF: $\eta = 500$; TBE: $\eta = 500$,$\alpha = 0$; EG: default settings as provided in [10]. For each column, the MSE for the best model is highlighted in dark (red) color, whereas for the second best it is highlighted in light (orange) color. Here, the lower the value, the better.

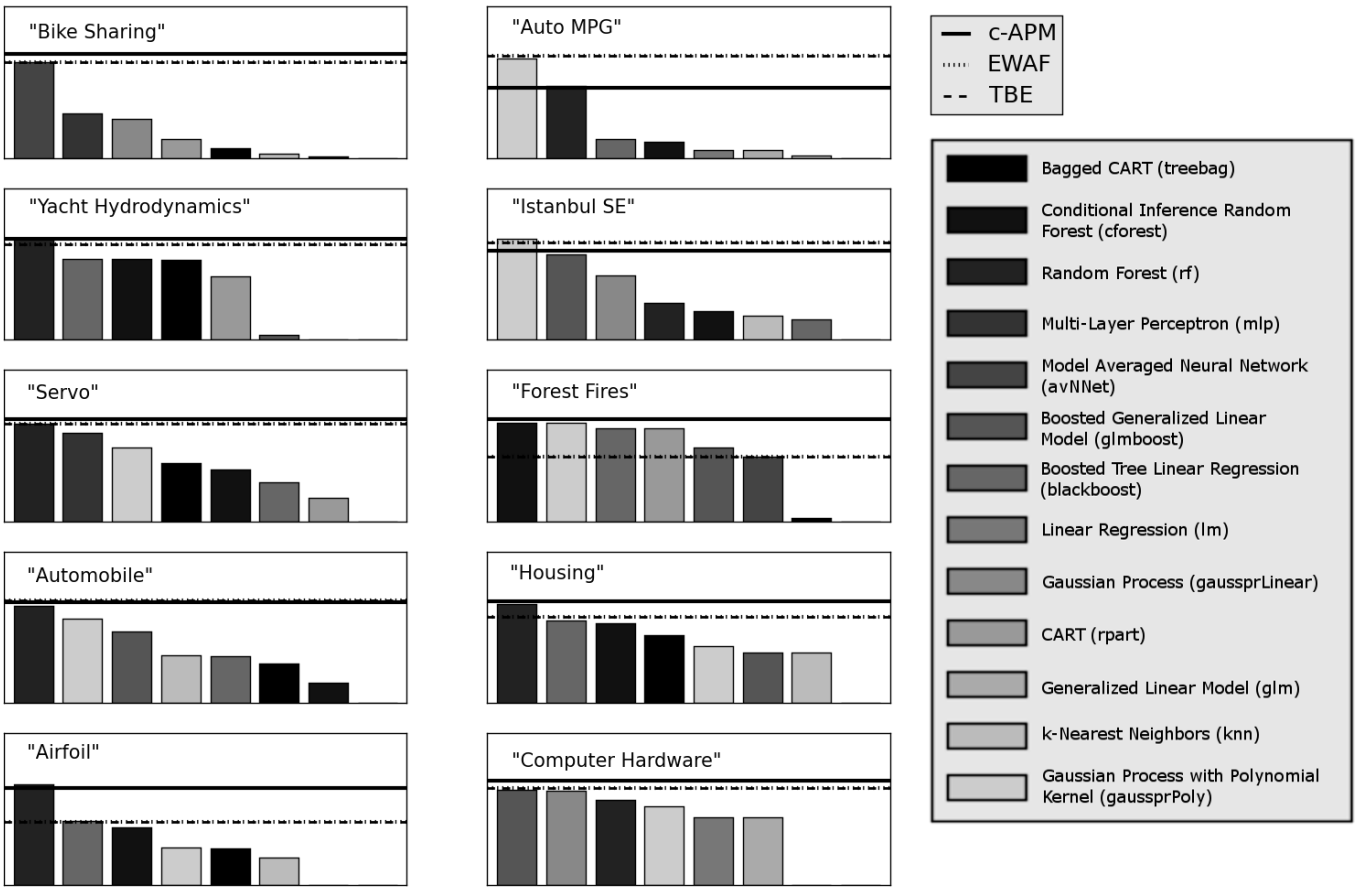

Fig. 1: Performances of c-APM, EWAF and TBE with respect to the top 8 individual component predictors (relative to each data set). We benchmark against EWAF and TBE as these were the two top performing PEA predictors. Here, the higher the value, the better.

REFERENCES

[1] A. Barbu and N. Lay, "An introduction to artificial prediction markets for classification," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2177–2204, 2012.

[2] A. J. Storkey, J. Millin, and K. Geras, "Isoelastic agents and wealth updates in machine learning markets," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[3] N. Lay and A. Barbu, "The artificial regression market," *arXiv preprint arXiv:1204.4154*, 2012, available to download from the followign link: http://arxiv.org/abs/1204.4154v1 (Retrieved Feb 1, 2016).

[4] J. Hu and A. J. Storkey, "Multi-period trading prediction markets with connections to machine learning," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, ser. JMLR Proceedings, vol. 32. JMLR.org, 2014, pp. 1773–1781.

[5] S. Dimitrov and R. Sami, "Non-myopic strategies in prediction markets," in *Proceedings of the 9th ACM conference on Electronic commerce*. ACM, 2008, pp. 200–209.

[6] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*.

Cambridge University Press, 2006.

[7] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1. Oakland, CA, USA., 1967, pp. 281–297.

[8] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Machine Learning*, vol. 32, no. 2, pp. 151–178, 1998.

[9] H. Robbins, "Asymptotically subminimax solutions of compound statistical decision problems," in *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, pp. 1–63, 1997.