REAL-TIME ANOMALY DETECTION WITH SUPEREXPERTS

A PREPRINT

Raisa Dzhamtyrova\(^1\) and Carsten Maple\(^{1,2}\)

\(^1\)The Alan Turing Institute
\(^2\)WMG Cyber Security Centre, University of Warwick
rdzhamtyrova@turing.ac.uk cm@warwick.ac.uk

October 9, 2020

ABSTRACT

The increasing connectivity of data and cyber-physical systems has resulted in a growing number of cyber attacks. Real-time detection of such attacks, through identification of anomalous activity, is required so that mitigation and contingent actions can be effectively and rapidly deployed. We propose to apply the prediction with expert advice (PEA) framework to a real-time anomaly detection problem. We apply PEA on open-source real datasets and show that the combination of models, which we call experts, provides significantly better results than any single model. An important property of the proposed approaches is their theoretical guarantees that they perform close to the best expert or even the superexpert, which can switch between the best performing experts. In addition, the approaches are also straightforward to implement and require little memory to run on streaming data.

Keywords real-time anomaly detection · online learning · prediction with expert advice

1 Introduction

Anomaly detection is an important and well-studied problem which has applications in various domains. Anomalies are defined to be patterns in data which differ from the expected behaviour [1]. Early detection of data breaches, fraudulent activities, cyber attacks, and system failures can potentially prevent catastrophic events, and financial and reputational losses. The optimal choice of algorithm depends on the type of problem and the nature of the anomalies. Anomaly detection techniques can be divided into three groups: supervised, in which anomalies are labelled; semi-supervised, in which anomalies are partially labelled; and unsupervised, in which there are no labels available. The nature of anomalies can also be classified into three groups: point anomalies, where a single point can be anomalous; contextual anomalies, where a data instance is anomalous in a specific context; and collective anomalies, where a collection of related data instances is anomalous with regard to the entire dataset [1]. The paper [1] provides an extensive survey of anomaly detection techniques for intrusion detection, fraud detection, fault detection, and other domains. The survey of novelty detection methods that aim to identify test data, which is in some respect is different from the available training data can be found in [2]. The article [3] gives a comparative study and evaluation of a large number of anomaly detection techniques on artificial and publicly available datasets. Furthermore, recent papers [4, 5] provide surveys of deep learning methods and real-time big data processing techniques for anomaly detection.

In this paper we propose to apply the PEA framework to the problem of real-time anomaly detection. In PEA a learner competes with experts, which may be human experts, other predictive models or even classes of functions, such as linear [6], logistic [7,8], and quantile [9]. The application of PEA to solve real-world problems includes domains such as sport [10], renewable energy [9], and finance [11,12]. The theory goes back to [13,14]. The investment methods competitive with large parametric classes are introduced in the universal portfolios theory in [11], which compete against portfolio selection techniques. In the PEA framework the goal of the learner is to construct a prediction strategy which can predict almost as well as the best expert in terms of cumulative losses. This approach works in online mode, where one does not have a training dataset to find the best model in advance. It should be noted that the performance
of models changes with time. As a result, reliance on one model can be dangerous in practice since the best model can degrade with time and produce unreliable predictions. Instead of re-training a model periodically, we propose a different approach which mixes predictions of several models based upon their current performance.

The Aggregating Algorithm (AA) was introduced in [15] to solve the problem of competitive prediction. The method assigns initial weights to experts and at each step updates their weights based on experts’ current losses. The approach is a generalisation of the Bayesian mixture of probabilistic models, and coincides with the Bayesian mixture in the case of the logarithmic loss function [6]. However, the AA works with a wide range of functions [6], such as square-loss, which we consider in this paper alongside logarithmic loss. The AA provides a theoretical guarantee that in the case of a finite number of experts a learner who follows the strategy of the AA has the loss as small as the best expert’s loss plus a constant at any time step in the future.

In this paper, the term superexpert [16] will refer to a strategy which can ‘switch’ between experts. It is common that some models can provide good predictions in particular areas of their expertise. For example, in a time-series forecasting problem some models are good at detecting seasonality, and others at finding trends. In the case of anomaly detection, some models can be good at predicting system failures whereas others are good at forecasting security breaches. In practice, it is important to identify different types of anomalies. However, most models would perform good only at some particular tasks. Furthermore, some models provide better predictions at different time segments of data. We assume that the superexpert has the ability to predict as a particular expert at some time interval and then switch to another expert. Our goal is to find a strategy that can perform close to these superexperts. The AA performs as well as the best expert in terms of cumulative losses. The Fixed-share and Variable-share algorithms proposed in [17] are generalisations of the AA and can perform close to the best superexpert. In the case when the number of switches is $k$, the total number of steps is $T$, and the number of experts is $N$, the additional loss of the Fixed-share algorithm is bounded by $O(k \log N + k \log(T/k))$. When the loss per trial can be bounded, Variable-share produces an additional loss bounded by $O(k \log N + k \log(\hat{L}/k))$, where $\hat{L}$ is the loss of the best superexpert with $k$ switches and $N$ base experts. In comparison to Fixed-share, the bound of Variable-share does not contain the number of steps $T$. The bounds of these algorithms are worse than the AA, which is unsurprising since they compete with all superexperts, and the number of superexperts grows exponentially with the number of switches. In this paper we apply Fixed-share with logarithmic loss, and Variable-share with square loss. The algorithms are simple to implement and require only $O(1)$ time per weight in each trial.

To the best of our knowledge, there is no previous research which proposes an application of PEA to the anomaly detection problem. The authors in [18] propose the sequential anomaly detection method in the presence of noise and limited feedback, however it is based on online convex programming, which is different from our approach. We test our approach on the Numenta Anomaly Benchmark (NAB) which contains both artificial and real datasets with labelled anomalies. This is one of the first open-source environments for testing anomaly detection algorithms on streaming data. The data contains a mix of contextual and collective anomalies, clean and noisy data, artificial and real scenarios, and data streams where the statistics evolve over time. The NAB benchmark was proposed along with the Hierarchical Temporal Memory (HTM) algorithm [19], which is based on a theoretical framework for sequence learning in the cortex. The Numenta HTM algorithm continuously learns in unsupervised fashion and adapts to dynamic environments. The NAB framework actively encourages participation, and the comparison currently includes 15 different algorithms, such as EXPoSE [20], Etsy’s Skyline, Multinomial Relative Entropy [21], Twitter’s Anomaly Detection, Bayesian Online Changepoint detection [22]. The Numenta HTM algorithm currently produces the best results on the NAB framework in terms of NAB scoring, which is introduced as a formal set of rules for determining the performance of streaming anomaly detection algorithms. We show that our approach provides significantly better results on the NAB framework in comparison to other open-source models from the scoreboard. We also provide an open-source code of our implementations to enable reproduction of our results.

2 Framework

Suppose that we have access to the predictions of some pool of experts which predict outcomes $y_1, y_2, \ldots$, from some outcome space $\Omega$, which occur sequentially. The experts’ predictions come from some decision space $\Gamma$. In the PEA framework a learner’s goal is to construct a prediction strategy whose performance will be close to the performance of the best expert from the pool in terms of a specified loss function $\lambda: \Omega \times \Gamma \rightarrow \mathbb{R}$. A triple $\mathcal{G} = (\Omega, \Gamma, \lambda)$ is called a game. At each step $t$ expert $E_i$ outputs its prediction $\xi_t(i) \in \Gamma$, $i = 1, 2, \ldots, N$, where $N$ is the number of experts. After seeing all experts’ predictions, the learner outputs prediction $\gamma_t \in \Gamma$. After nature announces an outcome $y_t \in \Omega$, the experts and the learner suffer losses $\lambda(y_t, \xi_t(i))$ and $\lambda(y_t, \gamma_t)$ respectively. The aim of the learner is to keep its
total loss $L_t$ small compared to the total losses $L^*_i$ of all experts $E_i$. We define a regret or an additional loss to be the difference between the cumulative losses of the best expert and the learner $R_t = L_t - \min_{i=1,\ldots,N} L^*_i$.

In this paper we consider anomalies to be binary outcomes from the outcome space $\Omega = \{0, 1\}$, and predictions of experts and of the learner are anomaly probabilities from the prediction space $\Gamma = [0, 1]$. We consider two loss functions: logarithmic loss (log-loss):

$$\lambda(y, \gamma) = \begin{cases} -\ln \gamma & \text{if } y = 1, \\ -\ln(1 - \gamma) & \text{if } y = 0, \end{cases}$$

and square-loss:

$$\lambda(y, \gamma) = (y - \gamma)^2.$$ 

We refer to the games with the respective loss functions as the log-loss and square-loss games.

### 3 Aggregating Algorithm

To solve the problem of predicting as well as the best expert in the pool the Aggregating Algorithm (AA) was proposed by V. Vovk in [15]. The idea of the AA is to update experts’ weights at each step according to their current losses. At each step $t$ the AA updates the experts’ weights as follows:

$$w_t(i) = e^{-\eta \lambda(y_t, \xi_t(i))} w_{t-1}(i),$$

where $\eta$ is the learning rate and $w_0(i)$ is the experts’ prior distribution.

At each step $t$ the AA outputs the prediction $\gamma_t$, which satisfies the following inequality:

$$\forall y \in \Omega : \lambda(y, \gamma_t) \leq g_t(y),$$

where $g_t(y)$ is the generalised prediction at step $t$:

$$g_t(y) = \frac{1}{\eta} \ln\left(\sum_{i=1}^{N} e^{-\eta \lambda(y, \xi_t(i))} w^*_t(i)\right)$$

and $w^*_t(i)$ are normalized weights:

$$w^*_{t-1}(i) = \frac{w_{t-1}(i)}{\sum_{i=1}^{N} w_{t-1}(i)}.$$  

The game $G$ where predictions satisfy (2) is called mixable. Both the log-loss and the square-loss games are mixable. The AA’s predictions $\gamma_t$ for the log-loss game which satisfy (2) are as follows (Section 2.2 in [6]):

$$\gamma_t = \sum_{i=1}^{N} \xi_t(i) w^*_t(i),$$

and for the square-loss game (Section 2.4 in [6]):

$$\gamma_t = \frac{1}{2} - \frac{g_t(1) - g_t(0)}{2}.$$  

An important property of the AA is its theoretical guarantee. The following lemma provides the theoretical bound on the cumulative loss of the AA.

**Lemma 1 (Section 2.1 in [6])** For a mixable game $G$, a learning rate $\eta$, and the uniform initial distribution on experts $w_0(i) = 1/N$, the following upper bound on the cumulative loss of the Aggregating Algorithm holds for $T = 1, 2, \ldots$,

$$L_T \leq L_T(E) + \frac{1}{\eta} \ln N.$$  

The lemma shows that the regret of the AA is constant and does not depend on the number of steps, and the AA’s performance is almost the same as the best expert’s performance at any time step.

The optimal choice of $\eta$ depends on the type of the game and should minimise the regret from (7). For the log-loss game the optimal $\eta = 1$ (Section 2.2 in [6]), and for the square-loss game $\eta = 2$ (Lemma 2.5 from [23]).
4 Competing with superexperts

As was shown in the previous section, the AA can perform as well as the best expert. However, it is common that the nature of data changes with time, and as a consequence the best expert also changes. It is important that a prediction strategy adapts to a constantly changing environment. Assume that there are $k+1$ segments which have different best performing ‘base’ experts. The term superexpert $[16]$ is referred to a strategy, which can switch between ‘base’ experts. For example, on the first segment the superexpert will follow the predictions of expert $E_j$ and on the second segment it will switch to expert $E_j$. The goal of the learner is to compete with any such superexpert.

The Fixed-share and the Variable-share algorithms are adaptations of the AA, which allow a learner to compete with superexperts. It is also possible to maintain only the weights of the base experts $[4]$ instead of keeping track of an exponential number of the superexperts’ weights. The main difference of these algorithms and the AA is in the update of the experts’ weights. After the weights’ update $[1]$ Fixed-share calculates the share updates $\tilde{w}_t(i)$:

$$
\tilde{w}_t(i) = (1 - \alpha)w_t(i) + \sum_{j \neq i} \frac{\alpha}{N-1}w_t(j)
$$

and the Variable-share update is as follows:

$$
\tilde{w}_t(i) = (1 - \alpha)\lambda(y_t,E_t(i))w_t(i) + \sum_{j \neq i} \frac{(1 - (1 - \alpha)\lambda(y_t,E_t(j)))}{N-1}w_t(j),
$$

where $\alpha$ is the probability of switching between experts. The generalised prediction $[3]$ is then calculated with the normalised weights $\tilde{w}_t^\ast(i)$ instead of $\tilde{w}_t(i)$. Note that if $\alpha = 0$ the algorithms coincide with the AA. In Fixed-share each expert shares a fraction $\alpha$ of its weight with other experts. However, if one expert has the best performance for a long period of time, it is not optimal to keep sharing its weight with others. To overcome this problem, in Variable-share experts share their weights only if they have a large loss, but do not share if they perform well. However, if an expert starts to predict badly its weights will be shared with other experts, and the next best expert’s weights will recover quickly. Variable-share, however, works only for the games with bounded losses. For this reason we apply Variable-share for the square-loss game, and apply Fixed-share for the log-loss game.

The following is the pseudo-code of the algorithms (see optimal values of $\eta$ for the log-loss and the square-loss games at the end of Section $[3]$):

Protocol 1 (Fixed-share and Variable-share)

Input: learning rate $\eta$, switching rate $\alpha$

Initialise weights $w_0(i) = 1/N$, $i = 1, \ldots, N$

FOR $t = 1, 2, \ldots$

read experts’ predictions $\xi_t(i)$, $i = 1, \ldots, N$

normalise weights $\tilde{w}_t^\ast(i) = \tilde{w}_t(i)/\sum_{i=1}^{N} \tilde{w}_t(i)$

learner predicts $\gamma_t \in \Gamma$, such that

$\forall y \in \Omega : \lambda(y_t, \gamma_t) \leq -\frac{1}{\eta} \ln \sum_{i=1}^{N} e^{-\eta \lambda(y_t, \xi_t(i))} \tilde{w}_{t-1}^\ast(i)

observe the outcome $y_t \in \Omega$

update the experts’ weights $w_t(i) = \tilde{w}_{t-1}(i)e^{-\eta \lambda(y_t, \xi_t(i))}$

if Fixed-share:

pool = $\sum_{i=1}^{N} \alpha w_t(i)$

$\tilde{w}_t(i) = (1 - \alpha)w_t(i) + \frac{1}{N-1} (pool - \alpha w_t(i))$

if Variable-share:

pool = $\sum_{i=1}^{N} (1 - (1 - \alpha)\lambda(y_t, \xi_t(i))) w_t(i)$

$\tilde{w}_t(i) = (1 - \alpha)\lambda(y_t, \xi_t(i))w_t(i) + \frac{1}{N-1} (pool - (1 - (1 - \alpha)\lambda(y_t, \xi_t(i))) w_t(i))$

END FOR

Note that for the log-loss and the square-loss games the learner outputs predictions according to $[5]$ and $[6]$ respectively with normalised weights $\tilde{w}_t^\ast(i)$.

We denote $\mathcal{S}_{T,N,k,e,t}$ to be the superexpert with $T$ number of steps, $N$ base experts and $k$ switches ($k < T$). The tuple $t$ divides the data into $k + 1$ segments $[t_0, t_1, [t_1, t_2), \ldots, [t_k, t_{k+1})$. The tuple $e$ has $k$ elements $(e_0, e_1, \ldots, e_k)$, such that $1 \leq e_j \leq N$ and $e_j \neq e_l$. The element $e_j$ denotes the expert $E_{e_j}$ associated with the $j$th segment $[t_j, t_{j+1})$.

The following lemma provides the upper bound on the loss of the Fixed-share algorithm.
Lemma 2 (Theorem 1 in [24]) For a mixable game $\mathcal{G}$, a learning rate $\eta$ and for any superexpert $S_{T,N,k,t,e}$ the total loss of the Fixed-share algorithm with parameter $\alpha$ satisfies

$$L_T \leq L_T(S_{T,N,k,t,e}) + \frac{1}{\eta} \left( \ln N + k \ln \frac{N-1}{\alpha} + (T - 1 - k) \ln \frac{1}{1 - \alpha} \right).$$

The minimum of the bound (10) is achieved by setting $\alpha = \frac{k}{T - 1}$. However, the number of steps $T$ and the number of switches $k$ is usually not known in advance. In practice it is often recommended to find the optimal $\alpha$ experimentally ([24]). In the case when the upper bound on the number of steps $\hat{T}$ and the lower bound on the number of switches $\hat{k}$ are known in advance the following corollary is realised.

Corollary 1 (Corollary 2 in [24]) Under the conditions of Lemma 2 and any positive reals $\hat{T}$ and $\hat{k}$, such that $\hat{k} < \hat{T} - 1$ by setting $\alpha = \frac{k}{T - 1}$, the loss of the Fixed-share algorithm can be bounded by

$$L_T \leq L_T(S_{T,N,k,t,e}) + \frac{1}{\eta} \left( \ln N + k \left( \ln \frac{\hat{T} - 1}{\hat{k}} + \ln(N - 1) + \hat{k} \right) \right),$$

where $S_{T,N,k,t,e}$ is any superexpert such that $T \leq \hat{T}$ and $k \geq \hat{k}$.

The following lemma provides the upper bound on the loss of the Variable-share algorithm.

Lemma 3 (Theorem 2 in [24]) Let $\mathcal{G}$ be a mixable game with a learning rate $\eta$, and a loss at each step bounded by an interval $[0, 1]$. Then for any superexpert $S_{T,N,k,t,e}$ the total loss of the Variable-share algorithm with parameter $\alpha$ satisfies

$$L_T \leq \left( 1 + \frac{1}{\eta} \ln \frac{1}{1 - \alpha} \right) L_T(S_{T,N,k,t,e}) + \frac{1}{\eta} \left( \ln N + k \left( \ln \frac{\hat{L}}{\hat{k}} + \ln(N - 1) + \ln \frac{9}{2} + \eta \right) \right).$$

The coefficient in front of the loss of the superexpert in (11) is greater than one, meaning that asymptotically the loss of Variable-share is worse than the loss of the superexpert $S_{T,N,k,t,e}$. The following corollary improves the asymptotic behaviour of the upper bound on Variable-share.

Corollary 2 (Corollary 3 in [24]) Under the conditions of Lemma 2 and any positive reals $\hat{L}$ and $\hat{k}$, by setting $\alpha = \frac{k}{2L + \hat{L}}$, the loss of the Variable-share algorithm can be bounded as

$$L_T \leq L_T(S_{T,N,k,t,e}) + \frac{1}{\eta} \left( \ln N + k \left( \ln \frac{\hat{L}}{\hat{k}} \right) + \ln(N - 1) + \ln \frac{9}{2} + \eta \right) + \hat{k}.$$
The NAB data corpus includes two artificial and five real data types each of which include several time-series. Examples of the real time-series include AWS server metrics; temperature sensor data of an industrial machine; a shut-down, a catastrophic failure; the key hold timings for several users of a computer; online advertisement clicking rates: cost-per-click and cost per thousand impressions. One of the artificial datasets does not include anomalies. The total number of time-series is 58. The data contains both contextual and collective anomalies, and the nature of data changes over time. The anomalies in these datasets do not come as single points, instead the datasets contain anomaly ‘windows’. It helps to detect anomalies in advance and take appropriate preventive measures.

The NAB benchmark has a scoreboard, and anyone can test their algorithms and take part in the competition. The current number of models is 15, including the Numenta HTM algorithm. Numenta has two implementations of their algorithms which we refer later to as numentaTM and numenta. The Numenta HTM algorithm is currently leading the scoreboard with the highest NAB scores. NAB score is introduced by Numenta as a formal set of rules for determining the performance of streaming anomaly detection algorithms.

Let $A_{TP}, A_{FP}, A_{TN}, A_{FN}$ be the corresponding weights for true/false positives, true/false negatives. The weight of individual detections of the algorithm $A$ is defined by the sigmoid function:

$$\sigma^A(y) = (A_{TP} - A_{FP}) \left( \frac{1}{1 + e^{\sigma y}} \right),$$

where $y$ is the relative position of the detection within the anomaly window. The NAB score for a datafile is the sum of the scores from individual detections plus the impact of missing any windows:

$$S_d^A = \left( \sum_{y \in Y_d} \sigma^A(y) \right) + A_{FN} f_d,$$

where $Y_d$ is the set of data instances detected as anomalies for datafile $d$, and $f_d$ is the number of false negatives. For more details see [25] and the weights of true positives, false negatives, etc. can be found in Numenta white paper [26].

We run the algorithms with experts, which are the algorithms available on the NAB benchmark, for different parameters of $\alpha$ starting from 0 to 0.9. For log-loss we apply the Fixed-share and for square-loss we apply the Variable-share algorithms. Note, that for $\alpha = 0$ the predictions coincide with the AA’s predictions.

5.1 Analysis of losses and classification metrics

First, we compare algorithms in terms of losses and classification metrics. Table[1] shows logarithmic and square total losses of the algorithms on the real datasets. We indicate the lowest losses in each dataset in bold. We can see from the table that both Fixed-share and Variable-share have significantly lower losses compared to other algorithms. The lowest losses of Fixed-share achieved with $\alpha = 0.05$, whereas Variable-share has small losses for all algorithms with $\alpha > 0$. Surprisingly Variable-share achieves both the lowest total logarithmic and square losses, though it was applied only with square loss. Variable-share achieves the lowest log-loss equal to 10.2, whereas the closest expert is randomCutForest with log-losses equal to 98.3. As for the total square loss Variable-share achieves 2.4 and the closest competitor randomCutForest archives 27.1. Both Fixed-share and Variable-share have significantly lower losses than the competitors from the NAB benchmark.

Table[2] compares NAB standard score, Area under curve (AUC), F-score and classification accuracy of all the algorithms. The values in the table are sorted by NAB standard score. The highest NAB standard score is achieved by the Fixed-share algorithm with $\alpha = 0.1$. The highest values of each metric are indicated in bold. The F-scores in the table are the maximum F-scores for each algorithm, and classification accuracy is calculated for the threshold which maximizes F-score. Numenta has the highest NAB score out of all the expert models, but it is significantly lower compared to the most of the Fixed-Share and the Variable-share algorithms. We also calculated standard classification metrics, such as AUC, F-score and classification accuracy. We can see from the table that for most values of $\alpha$ both Fixed-share and Variable-share provide very good accuracy in terms of all classification metrics considered. The classification metrics in the table are calculated on the joint dataset which contains 58 time-series. Note, that the average number of anomalies on the total dataset is 9.2%. Therefore, any classification accuracy less than 90.8% is worse than the algorithm which always predicts ‘no anomalies’.

5.2 Visualisation of predictions

In this section we visualise the predictions of our proposed approaches for different $\alpha$ and compare them with numenta’s predictions. First we visualise Fixed-share and Variable-share with the highest NAB standard scores. Figure[1] shows
the predictions of Fixed-share with $\alpha = 0.1$ for three time-series from the real dataset with known anomaly causes. The blue lines correspond to the algorithm’s predictions, whereas the orange lines represent the anomaly windows. Figure 2 illustrates the predictions of Variable-share with $\alpha = 0.5$. The optimal thresholds for these algorithms that maximise the F-score are 0.41 and 0.37 for Fixed-share and Variable-share respectively. It is consistent with the figures, and it is clear that the algorithms ‘catch’ the shapes of the anomaly windows. For comparison Figure 3 illustrates the predictions of numenta. We can see that numenta’s predictions are more ‘spiky’. The optimal threshold of numenta on the whole dataset is around 0.03. It means that anything which is above that threshold should be classified as an anomaly. Figures 4 and 5 illustrate the predictions of the AA with square-loss and Fixed-share with $\alpha = 0.9$ respectively. The predictions for $\alpha = 0$ are very ‘spiky’ whereas for $\alpha = 0.9$ the predictions are very similar for points with and without anomalies. We will explain the behaviour of the algorithms for different values of $\alpha$ in the next section by analysing the experts’ weights.

### 5.3 Weights analysis

In this section we analyse the weight updates of the base experts for the AA, Fixed-share and Variable-share. Figure 6 illustrates the normalised weight updates of the algorithms for the real dataset with known causes of the machine system failure, which contains time-series of the machine’s temperatures. We can observe from the graphs that the AA with log-loss follows the predictions of twitterAdVec, and then switches to the predictions of randomCutForest. The AA with square-loss switches between more experts, however the best expert usually acquires the largest weight close to one, whereas other models almost do not participate in making predictions. Fixed-share observes less drastic weight changes compared to the AA. For $\alpha = 0.5$ we observe that two expert models acquire the largest weights: EXPoSE and windowedGaussian. However, other models have non-zero weights, and therefore participate in making predictions. The impact of $\alpha$ becomes more obvious when we increase it even more. For Fixed-share with a switching

| algorithm       | logarithmic | square |
|-----------------|-------------|--------|
| knncad          | 259.9       | 84.7   |
| numentaTM       | 151.4       | 29.6   |
| twitterADVec    | 500.7       | 31.1   |
| skyline         | 374.6       | 29.7   |
| earthgeckoSkyline| 503.8      | 31.3   |
| numenta         | 153.4       | 29.6   |
| bayesChangePt   | 499.0       | 37.2   |
| null            | 222.6       | 80.3   |
| expose          | 522.7       | 154.4  |
| relativeEntropy | 502.0       | 31.1   |
| htmjava         | 144.3       | 29.3   |
| randomCutForest | 98.3        | 31.1   |
| random          | 322.0       | 107.2  |
| contextOSE      | 250.3       | 35.1   |
| windowedGaussian| 433.1       | 143.4  |

|                   | logarithmic | square |
|-------------------|-------------|--------|
| AA, log           | 113.8       | 30.7   |
| Fixed-share, $\alpha = 0.05$ | 17.7       | 2.7    |
| Fixed-share, $\alpha = 0.1$  | 25.3       | 3.7    |
| Fixed-share, $\alpha = 0.2$  | 38.6       | 6.2    |
| Fixed-share, $\alpha = 0.3$  | 50.6       | 9.3    |
| Fixed-share, $\alpha = 0.5$  | 72.4       | 16.4   |
| Fixed-share, $\alpha = 0.7$  | 92.5       | 23.8   |
| Fixed-share, $\alpha = 0.9$  | 111.4      | 30.6   |

|                   | logarithmic | square |
|-------------------|-------------|--------|
| Variable-share, $\alpha = 0.05$ | 10.5       | 2.5    |
| Variable-share, $\alpha = 0.1$  | 10.3       | 2.4    |
| Variable-share, $\alpha = 0.2$  | 10.2       | 2.4    |
| Variable-share, $\alpha = 0.3$  | 10.2       | 2.5    |
| Variable-share, $\alpha = 0.5$  | 10.5       | 2.6    |
| Variable-share, $\alpha = 0.7$  | 11.1       | 2.9    |
| Variable-share, $\alpha = 0.9$  | 12.4       | 3.4    |

Table 1: Total losses ($\times 10^3$) on the real datasets
| algorithm              | NAB standard | AUC       | F-score   | classification accuracy |
|------------------------|--------------|-----------|-----------|-------------------------|
| Perfect                | 100.0        | 1.000     | 1.000     | 1.000                   |
| Fixed-share, $\alpha = 0.1$ | 98.3         | 0.998     | 0.990     | 0.998                   |
| Fixed-share, $\alpha = 0.05$ | 98.2         | 0.998     | 0.990     | 0.998                   |
| Fixed-share, $\alpha = 0.2$ | 97.4         | 0.998     | 0.989     | 0.998                   |
| Variable-share, $\alpha = 0.5$ | 96.6         | 0.998     | 0.988     | 0.998                   |
| Variable-share, $\alpha = 0.7$ | 96.5         | 0.998     | 0.988     | 0.998                   |
| Variable-share, $\alpha = 0.1$ | 96.4         | 0.998     | 0.984     | 0.997                   |
| Variable-share, $\alpha = 0.3$ | 96.4         | 0.998     | 0.987     | 0.998                   |
| Variable-share, $\alpha = 0.2$ | 96.4         | 0.998     | 0.986     | 0.997                   |
| Variable-share, $\alpha = 0.3$ | 96.3         | 0.998     | 0.984     | 0.997                   |
| Variable-share, $\alpha = 0.05$ | 96.1         | 0.998     | 0.983     | 0.997                   |
| Variable-share, $\alpha = 0.9$ | 95.7         | 0.998     | 0.987     | 0.998                   |
| Fixed-share, $\alpha = 0.5$ | 91.5         | 0.996     | 0.950     | 0.991                   |
| Fixed-share, $\alpha = 0.7$ | 78.7         | 0.972     | 0.803     | 0.963                   |
| numenta                | 70.1         | 0.516     | 0.210     | 0.866                   |
| contextOSE             | 69.9         | 0.378     | 0.168     | 0.140                   |
| htmjava                | 66.1         | 0.518     | 0.218     | 0.843                   |
| Fixed-share, $\alpha = 0.9$ | 65.4         | 0.698     | 0.287     | 0.821                   |
| numentaTM              | 64.6         | 0.498     | 0.201     | 0.864                   |
| earthgeckoSkyline      | 58.2         | 0.501     | 0.168     | 0.092                   |
| knncad                 | 58.0         | 0.600     | 0.216     | 0.540                   |
| relativeEntropy        | 54.6         | 0.503     | 0.168     | 0.092                   |
| randomCutForest        | 51.7         | 0.616     | 0.233     | 0.836                   |
| twitterADVec           | 47.1         | 0.504     | 0.168     | 0.092                   |
| windowedGaussian       | 39.6         | 0.558     | 0.196     | 0.862                   |
| skyline                | 35.7         | 0.566     | 0.191     | 0.092                   |
| bayesChangePt          | 17.7         | 0.503     | 0.168     | 0.092                   |
| random                 | 17.7         | 0.502     | 0.168     | 0.097                   |
| expose                 | 16.4         | 0.557     | 0.181     | 0.557                   |
| AA, square             | 13.8         | 0.688     | 0.278     | 0.836                   |
| AA, log                | 13.0         | 0.730     | 0.275     | 0.811                   |
| null                   | 0.0          | 0.500     | 0.168     | 0.092                   |

Table 2: NAB score, AUC, F-score, classification accuracy

rate $\alpha = 0.9$ the base experts share the greatest amount of their weights, so the difference between experts’ weights becomes insignificant. Therefore the algorithm’s predictions become close to the experts’ average. This effect explains the behaviour of Fixed-share’s predictions at Figure 5. We observe a different behaviour in the case of Variable-share. If one expert performs well for a long period of time, it stops to share its weight with others. However, if an expert starts to suffer a large loss, its weights will be shared with other experts, which will help the recovery of the weights of the next best expert. We can see from the graphs that Variable-share allows rapid switching between base experts, reflecting their current performance. The difference between the behaviour of Variable-share with $\alpha = 0.5$ and $\alpha = 0.9$ does not seem to be significant.

5.4 Theoretical bounds

An important property of the considered algorithms is their theoretical bounds. At any point in the future we are sure that the performance of the algorithms will be close to the best expert’s performance in the case of the AA, and close to the superexpert’s performance in the case of Fixed-share and Variable-share. In the following experiments we visualise the theoretical bound of the AA. For this purpose we picked one time-series of the perfect square wave from the artificial dataset without anomalies. Figure 7a shows the difference between the total logarithmic losses of the expert models and the AA. Therefore, if the loss difference is below zero, the AA performs worse than the expert. For the illustration purpose we pick three models: numenta, earthgeckoSkyline and bayesChangePt. The last two models are taken because they have the lowest total losses on the artificial dataset. The figure also illustrates the theoretical bound of the AA. The meaning of this bound is that for any point in the future, even if the AA performs worse than the best expert, it will not
perform much worse. In the case of the AA, the theoretical bound is a constant and does not depend on time. From the figure we can see that the bound is ‘tight’. Figure 7b illustrates the same experiment for square loss.

6 Conclusions

In this work we have applied the PEA framework to the problem of real-time anomaly detection. The proposed approaches have theoretical guarantees on their performance. The Aggregating Algorithm provides a strategy that performs the same as the best expert up to an additive constant for any point in the future. Another two strategies, Fixed-share and Variable-share, provide performance close to the superexpert. Fixed-share can work with unbounded losses, however its additional loss depends on the number of steps. Variable-share works with bounded losses and has a more sophisticated weights’ update. The theoretical bound of Variable-share does not depend on the number of steps.

We have tested the proposed approaches on the NAB benchmark, which provides an open-source environment for testing anomaly detection algorithms on streaming datasets. Experimental results show that the proposed approaches, which combine models, provide significantly better results than any single model available at the NAB repository in terms of losses, the NAB score and several classification metrics. We explore the predictions of the proposed methods for different values of the switching rate $\alpha$. We explain the behaviour of the algorithms by analysing the weights’ updates for different $\alpha$. In practice it is often recommended to find the optimal $\alpha$ experimentally [24]. Our experiments show that Fixed-share performs better for small values of $\alpha$, whereas the choice of $\alpha$ does not have a big impact on the performance of Variable-share. However, it might be different on other datasets, and we also would recommend to run a few algorithms with different $\alpha$, and compare their performance. At last, we illustrate the ‘tightness’ of the Aggregating Algorithm’s theoretical bounds.

References

[1] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: a survey. ACM Computing Surveys, 41:1–72, 2009. https://doi.org/10.1145/1541880.1541882
Figure 3: Predictions of numenta

Figure 4: Predictions of the Aggregating Algorithm with square loss

[2] V. Chandola, V. Mithal, and V. Kumar. Comparative evaluation of anomaly detection techniques for sequence data. In *Eighth IEEE International Conference on Data Mining*, pages 743–748, 2008. [https://doi.org/10.1109/ICDM.2008.151](https://doi.org/10.1109/ICDM.2008.151).

[3] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Process*, 99:215–249, 2014. [https://doi.org/10.1016/j.sigpro.2013.12.026](https://doi.org/10.1016/j.sigpro.2013.12.026).

[4] R. Chalapathy and S. Chawla. Deep learning for anomaly detection: A survey. *A Preprint*, 2019. [https://arxiv.org/pdf/1901.03407.pdf](https://arxiv.org/pdf/1901.03407.pdf).

[5] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. Abaker, I. A. T. Hashem, E. Ahmed, and M. Imran. Real-time big data processing for anomaly detection: A survey. *International Journal of Information Management*, 45:289–307, 2019. [https://doi.org/10.1016/j.ijinfomgt.2018.08.006](https://doi.org/10.1016/j.ijinfomgt.2018.08.006).

[6] V. Vovk. Competitive on-line statistics. *International Statistical Review*, 69(2):213–248, 2001.

[7] F. Zhdanov and Y. Kalnishkan. Universal algorithms for probability forecasting. *International Journal on Artificial Intelligence Tools*, 21(04), 2012.

[8] R. Dzhamtyrova and Y. Kalnishkan. Universal algorithms for multinomial logistic regression under kullback-leibler game. *Neurocomputing*, 2019.

[9] R. Dzhamtyrova and Y. Kalnishkan. Competitive online quantile regression. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 499–512, 2020.

[10] V. Vovk and F. Zhdanov. Prediction with expert advice for the Brier game. *Journal of Machine Learning Research*, 10:2445–2471, 2009.

[11] T. M. Cover and E. Ordentlich. Universal portfolios with side information. *IEEE Transactions on Information Theory*, 42(2), 1996.

[12] R. Dzhamtyrova and Y. Kalnishkan. Prediction with expert advice for Value at Risk. In *International Joint Conference on Neural Networks (IJCNN)*, 2020.
Figure 5: Predictions of Fixed-share with $\alpha = 0.9$.

Figure 6: Weights update of the Aggregating Algorithm, Fixed-share and Variable-share for different $\alpha$.

[13] N. Littlestone and M. K. Warmuth. The Weighted Majority Algorithm. *Information and Computation*, 108:212–261, 1994.

[14] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[15] V. Vovk. Aggregating strategies. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pages 371–383, San Mateo, CA, 1990. Morgan Kaufmann.

[16] Y. Kalnishkan. Lectures in online machine learning. *Royal Holloway, University of London*, 2018.

[17] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 1998.

[18] M. Raginsky, R. Willett, C. Horn, J. Silva, and R. Marcia. Sequential anomaly detection in the presence of noise and limited feedback. *IEEE Trans. Info. Theory*, 58, 2012.

[19] S.. Ahmad, A. Lavin, S. Purdy, and Z. Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.
Figure 7: Loss difference between algorithms and Aggregating Algorithm

[20] M. Schneider, W. Ertel, and F. Ramos. Expected similarity estimation for large-scale batch and streaming anomaly detection. *Machine Learning*, 105:305–333, 2016. [https://doi.org/10.1007/s10994-016-5567-7](https://doi.org/10.1007/s10994-016-5567-7)

[21] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan. Statistical techniques for online anomaly detection in data centers. In *the 12th IFIP/IEEE International Symposium on Integrated Network Management*, pages 385–392, 2011. [https://doi.org/10.1109/INM.2011.5990537](https://doi.org/10.1109/INM.2011.5990537)

[22] R. P. Adams and D. J. C. Mackay. Bayesian online changepoint detection. [https://arxiv.org/abs/0710.3742](https://arxiv.org/abs/0710.3742), 2007.

[23] F. Zhdanov. Theory and applications of competitive prediction. *PhD thesis*, 2011.

[24] M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[25] A. Lavin and S. Ahmad. Evaluating real-time anomaly detection algorithms – the numenta anomaly benchmark. In *Proceedings of the 14th International Conference on Machine Learning Application*, 2015. [https://doi.org/10.1109/ICMLA.2015.141](https://doi.org/10.1109/ICMLA.2015.141)

[26] Numenta. The numenta anomaly benchmark: White paper. [https://github.com/numenta/NAB/wiki#nab-whitepaper](https://github.com/numenta/NAB/wiki#nab-whitepaper), 2015.

[27] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.