<http://www.diva-portal.org>

Preprint

This is the submitted version of a paper presented at *1st Workshop on Interactive Data Mining, WIDM 2019, co-located with 12th ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, Australia; 15 February, 2019.*

Citation for the original published paper:

Calikus, E., Fan, Y., Nowaczyk, S., Pinheiro Sant'Anna, A. (2019)

Interactive-cosmo: Consensus self-organized models for fault detection with expert feedback

In: *Proceedings of the Workshop on Interactive Data Mining, WIDM 2019* (pp. 1-9).

New York: Association for Computing Machinery (ACM)

<https://doi.org/10.1145/3304079.3310289>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:hh:diva-41365>

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/334188153>

Interactive-COSMO: Consensus Self-Organized Models for Fault Detection with Expert Feedback

Conference Paper · February 2019

DOI: 10.1145/3304079.3310289

CITATIONS

0

READS

39

4 authors:

Ece Calikus

Halmstad University

8 PUBLICATIONS 2 CITATIONS

[SEE PROFILE](#)

Yuantao Fan

Halmstad University

15 PUBLICATIONS 22 CITATIONS

[SEE PROFILE](#)

Slawomir Nowaczyk

Halmstad University

58 PUBLICATIONS 211 CITATIONS

[SEE PROFILE](#)

Anita Sant'Anna

Halmstad University

38 PUBLICATIONS 154 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:

Self-Monitoring for Innovation (SEMI) [View project](#)

Pattern analysis and machine learning [View project](#)

Interactive-COSMO: Consensus Self-Organized Models for Fault Detection with Expert Feedback

Ece Calikus
Halmstad University
ece.calikus@hh.se

Sławomir Nowaczyk
Halmstad University
slawomir.nowaczyk@hh.se

Yuantao Fan
Halmstad University
yuantao.fan@hh.se

Anita Sant'Anna
Halmstad University
anita.santanna@hh.se

ABSTRACT

Diagnosing deviations and predicting faults is an important task, especially given recent advances related to Internet of Things. However, the majority of the efforts for diagnostics are still carried out by human experts in a time-consuming and expensive manner. One promising approach towards self-monitoring systems is based on the “wisdom of the crowd” idea, where malfunctioning equipments are detected by understanding the similarities and differences in the operation of several alike systems.

A fully autonomous fault detection, however, is not possible, since not all deviations or anomalies correspond to faulty behaviors; many can be explained by atypical usage or varying external conditions. In this work, we propose a method which gradually incorporates expert-provided feedback for more accurate self-monitoring. Our idea is to support model adaptation while allowing human feedback to persist over changes in data distribution, such as concept drift.

KEYWORDS

Anomaly Detection, Self-Monitoring, Active Learning, Human-in-the-loop

ACM Reference format:

Ece Calikus, Yuantao Fan, Sławomir Nowaczyk, and Anita Sant'Anna. 2019. Interactive-COSMO: Consensus Self-Organized Models for Fault Detection with Expert Feedback. In *Proceedings of Workshop on Interactive Data Mining, Melbourne, VIC, Australia, February 15, 2019 (WIDM'19)*, 9 pages. <https://doi.org/10.1145/3304079.3310289>

1 INTRODUCTION

The ability to diagnose deviations and predict faults effectively is an important task for various domains in the industry concerning cost reduction and sustainability. After advances in the Internet of Things (IoT), many modern industrial systems started to produce and preserve a large amount of data from their operations. However,

the majority of the efforts for diagnostics are still carried out by human experts in a time-consuming and expensive manner.

There have been many studies about monitoring and diagnostics for the industry, mostly related to anomaly detection[26]. One approach towards self-monitoring systems, able to automatically detect faults and deviations, is based on the “wisdom of the crowd” idea, where malfunctions are detected by understanding the similarities and differences in the group of systems that operates similarly. Many issues can be recognized by identifying groups of such peers and evaluating how well each of them conforms to its group.

It is, however, not realistic to imagine a fully autonomous fault detection system that could operate without interactions with humans. The most important reason is that not all deviations or anomalies correspond to faulty behaviors. Atypical usages or external conditions can explain many of them. Furthermore, most of the anomalous instances are not interesting, and domain expertise is required to separate the useful examples from the noise. It is crucial to develop interactive approaches that allow humans to be a part of the learning loop. People observing and interpreting the results of self-monitoring, as well as providing feedback, is necessary for building systems that can tackle many real-world problems[3].

A significant challenge in fault detection is adaptability. Monitored processes often change due to variations in external inputs, structural adaptations, maintenance, etc. Many existing interactive learning methods are not designed to cope with such changes without overloading the human.

Our contribution allows incorporating expert knowledge into a self-monitoring method based on peer-group analysis, which allows human feedback to persist over changes such as concept drift. Consensus Self-Organising Models (COSMO) method[17] is an effective method for identifying deviations even in non-stationary environments. Its capability to detect changes in the characteristics of signals or data streams has been shown in [7]. This work proposes a method that incorporates human expertise into COSMO algorithm to adapt itself based on the user interactions for more accurate self-monitoring.

We employ exploration-exploitation trade-off in an interactive feedback process. In the beginning, the anomaly detection algorithm is applied to an unlabeled dataset in a purely unsupervised manner. As the interactive learner interacts with the experts and obtains feedback for selected instances in the form of ground-truth labels, the approach exploits those labels to update the model. Whenever the model is updated, new anomaly scores are computed, and instances are reclassified. Consequently, the method

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WIDM'19, February 15, 2019, Melbourne, VIC, Australia

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6296-2/19/02...\$15.00

<https://doi.org/10.1145/3304079.3310289>

Figure 1: Illustration of the changes in the model over several iterations. Labeled observations are represented with red and unlabeled observations with black. Circles are healthy samples while triangles are faulty.

gradually changes with expert feedback to separate more complex differences between normal and abnormal. The process continues until either all the data is exhausted, or the interactive learning runs out of the query budget.

We demonstrate our approach in two different settings. In the first setting, we measure how the accuracy of the model depends on the number of queries presented to the expert. This is one of the typical evaluation strategies in the active-learning community, where the goal is to obtain the highest classification accuracy with the least amount of human effort. The second setting, however, is more realistic for self-monitoring applications. In this case, the system is designed to perform continuous, online self-monitoring. Here, all instances in the training set need to be labeled by the system eventually. Over time, the correct classification of all of them also gets known, as the objects are analyzed either experienced failures or not.

2 RELATED WORK

Active learning has received significant attention in recent years. The learning algorithm can adaptively select training examples based on previous data with active learning and this adaptivity can lead to important improvements[2][24]. Iterative active learning approaches are often used to learn a classifier with minimal supervision[19]. Therefore, many of the works are concerned with the relative merits of different query criteria[12] such as uncertainty sampling[14][23], query-by-committee[21][5], expected error reduction[18], expected model change[20], querying points that are most certain[6].

Active learning has been applied to anomaly detection problem using different query strategies [1, 6, 9, 10, 12]. A nearest-neighbor method in active learning setting is proposed in [11]. Methods applying different query strategies such as query-by-committee and selective sampling in the context of outlier detection are discussed in [1, 12]. A method has also been proposed in [25] which incorporates analyst feedback for detecting malicious attacks on information systems. It combines ensemble of unsupervised outlier detection methods and a supervised learning algorithm.

Furthermore, outlier by example [28] method augments the user provided examples by comparing the deviations of the objects with user-provided examples. This method adds artificially generated samples to the training data, in order to increase the number of positive examples for the learning process.

[6] proposed an active learning approach using the unsupervised Loda anomaly detection algorithm [15]. They apply a semi-supervised extension of the accuracy-at-the-top [4] loss function to select the most anomalous unlabeled example for labeling and to update Loda's ensemble weights based on the user's feedback. None of the approaches above deal with non-stationary streaming environments.

3 METHODOLOGY

The majority of anomaly detection applications have to deal with non-stationary data. Therefore, the model of normality needs to be updated in the course of operation. Incorporation of new data points and the removal of irrelevant ones can be efficiently done by

using online algorithms. In this section, we describe our learning strategies for online fault detection.

First, we initialize our method on the unlabeled examples. After the first iteration, the model chooses a candidate x_i using active learning strategies and queries to the domain expert. The expert provides feedback, and the model is updated consequently. Labeling and re-training steps are repeated in two different settings.

In our interactive learning procedure, we request two types of labels from the expert. Given a dataset D with n instances, $D = x_1, \dots, x_n$, let $((x_1, y_1, z_1), \dots, (x_n, y_n, z_n))$ be the training set where x_i are the feature vectors representing the instances, $y_i \in \{\text{faulty}, \text{healthy}\}$ and $z_i \in \{\text{relative}, \text{absolute}\}$ which represents the position of the instance. We have discussed how we incorporate different types of feedback into the model further in this paper.

Figure 1 demonstrates a toy example how the model evolves after several iterations. Whenever true labels of data points are received, the model parameters are adapted to achieve better classification. On the right image, the model is updated to include labeled nominals inside the decision border while discriminating labeled anomalies.

3.1 Interactive Self-Monitoring

We refer to a system that monitors its operation, learns typical behaviors and data characteristics over time, detects abnormalities and discovers faults as “self-monitoring”[17]. There are two important challenges to achieve self-monitoring systems in real-life. The first is “empowering” the human, i.e., allowing domain experts to be interactively involved during self-monitoring process. The second challenge is to maintain lifetime learning in dynamic environments. Algorithms that model the underlying data and processes must be able to cope with changes and adapt the decisions accordingly.

When designing interactive algorithms, it is crucial to understand the setting in which they will be used. Depending on the constraints, different approaches can be useful. In this work, we present two such modes of operation, which appear quite similar at first glance. They both involve anomaly detection with human feedback, however, differ in important aspects.

The first setting assumes that there is a cost associated with making a query to the expert. In a sense, the algorithm needs to decide whether it’s sufficiently confident in its knowledge to make a prediction, or whether it should instead ask the human for help. The aim is to show improvement in the performance based on the number of data instances that are queried. It has been applied in many works about active learning for evaluation. It also shows the trade-off between exploration and exploitation costs, since each query made is supposed to bring the highest possible accuracy increase. This first setting, which we refer to as “active learning” one, is formalized in Algorithm 1.

The second mode is more similar to the actual use of fault detection systems in real world. They need to exhibit incremental learning with continuous model adaptation based on a constantly arriving data stream. A self-monitoring system is presented with some number of data instances, corresponding to the machine that it supervises and needs to make predictions about all of them ultimately. However, over time, the real observations will either confirm or contradict those predictions. In this way, the actual feedback will be available for every decision the system makes – but

Algorithm 1: Active Learning Mode

Data: Unlabeled Data $\mathcal{U}$, query budget b

```

1  $\mathcal{L} = \emptyset$ 
2 while  $|\mathcal{L}| \leq b$  do
3   Select instance  $x^q$  from  $\mathcal{U}$ 
4   Query expert for the label on  $x^q$ 
5   Add  $x^q$  to  $\mathcal{L}$ 
6   Update model
7 end
8 Compute accuracy of the resulting model (using unseen data)
```

only after the fact. We model this setting as a continuous, online learning process which is carried out in a sequential manner until the entire dataset is exhausted. The model continuously learns with user labeled data and updates itself while making prediction for each selected instance. In reality, the feedback about a system is rarely available immediately, before the next decision needs to be made – but at this stage, we believe it to be a justifiable simplification. This second setting, which we refer to as “self-monitoring” one, is formalized in Algorithm 2.

Algorithm 2: Self-Monitoring Mode

Data: Unlabeled Data $\mathcal{U}$

```

1  $\mathcal{L} = \emptyset$ 
2 while  $|\mathcal{L}| \leq |\mathcal{U}|$  do
3   Select instance  $x^q$  from  $\mathcal{U}$ 
4   Make prediction about  $x^q$ 
5   Collect the label on  $x^q$  from the expert
6   Add  $x^q$  to  $\mathcal{L}$ 
7   Update model
8 end
9 Compute online accuracy of the process (over dataset  $\mathcal{U}$ )
```

It is important to note that self-monitoring is about discovering deviations or faults in units such as equipment, machines or other objects. In general, such objects generate multiple data samples per instance. Thus, in this work, we do not focus on marking individual observations as faulty or healthy. Those samples inherit the characteristics of individual objects, and it’s the objects that are tracked. These samples and observations can be collected from an individual at over time, as it performs different tasks under various conditions. Studying the variability of these samples and comparing them with individuals performing similar tasks under similar conditions are important and useful to capture anomalies.

3.2 COSMO Method

As the baseline deviation detection method, we use COSMO (Consensus Self-Organized Models) algorithm[17], an anomaly detection method based on the “wisdom of the crowd” approach. It subsequently compares each observation or model against others to find deviations from the consensus. In this way, the majority is used to provide a standard, or to describe normal behavior, together with its expected variability over time.

The method allows data to be represented using different models and then applies a consensus measure to estimate how different a model is from its peers. In here, we use "the distance to the most central object" as the consensus measure and Euclidean distance as the distance measure for simplicity, but other representations and metrics are compared in [7].

The input to the COSMO algorithm is a matrix of unlabeled data points $\mathcal{U}$ and the corresponding weights vector W . The first step is to compute the center of the (weighted) data distribution, i.e., the mean of all the observations when using Euclidean distance or peak of the elliptic Gaussian when using a Mahalanobis distance[16]. This center is selected as the *most central object* (denoted by c):

$$c = \frac{1}{N} \mathbf{1}^T \mathcal{U} W^T, \quad (1)$$

The COSMO algorithm then calculates the empirical distribution of distances from c to all the data samples. The *z-score* for any test sample i is the number of samples in the training set that lie further away from the most central object (c) than i :

$$z(i) = \frac{|\{i = 1, \dots, N : d_{ic} > d_{mc}\}|}{N}, \quad (2)$$

The null hypothesis is that all observations are drawn from the same distribution, in which case z-scores should be uniformly distributed between zero and one [27]. This hypothesis is tested using a Z-test, comparing the arithmetic mean of z-scores over a certain number of observations n with the value expected from a uniform distribution. The negative logarithm of the one-sided *p-value* from this test is used as the deviation level for the instance m :

$$p(m) = -\log_{10} \left[\Phi \left(\frac{\bar{z} - 0.5}{\sigma_n} \right) \right], \quad (3)$$

where $\Phi(\cdot)$ is the normal cumulative distribution function, $\bar{z}$ is the mean of the z-scores, $\sigma_n = (12n)^{-1/2}$, and n is the number of observations in the window that were used when computing $\bar{z}$. The logarithm transform is applied mainly for making *p-values* more interpretable. The method is summarized as Algorithm 3.

Algorithm 3: COSMO

Data: Unlabeled data $\mathcal{U}$

Result: Deviation levels for all examples

```

1 Calculate the most central object  $c$ 
2 for each window  $w$  do
3   for every observations  $x_i \in w$  do
4     calculate  $z(i)$ 
5   end
6   calculate  $p(m)$ 
7 end
```

Detection is carried out by comparing the deviation level with threshold θ for instance x_m :

$$f(x_m) = \begin{cases} \text{faulty}, & \text{if } p(x_m) \geq \theta \\ \text{healthy}, & \text{otherwise} \end{cases} \quad (4)$$

This approach is similar to centroid-based anomaly detection methods[22], since the most central object can be treated as centroid and the threshold is the radius which corresponds to a spherical decision border. As an alternative to the idea presented here, expert input can also be incorporated in COSMO in the representation selection step[8].

3.3 Interactive Fault Discovery with COSMO

We assume that we do not have any labeled instances in the beginning, and thus our initial deviation detection is done in a purely unsupervised manner. However, over time, the system improves by identifying the most promising observations in the dataset to learn. The interactions between Active-COSMO and the expert can follow two different modes, as explained earlier, but the core algorithm is the same. In both cases, the algorithm exploits feedback in the form of ground-truth labels by adapting the model parameters.

The general method works as follows. COSMO, based on the observed data and, later on, labels received from the expert, assigns anomaly scores to each data instance. Those values are used to decide on the next instance to select, either just for a query (in the "active learning" setting) or to make a prediction (in the "self-monitoring" setting). After receiving the feedback, COSMO updates its model by updating two parameters: threshold θ and the most central object c .

Threshold parameter is a simple value which is independent of the data, so it can be memorized and updated to classify observations as faulty or healthy based on current deviation levels. The most central object, however, depends on both the feedback history and the new data – if the new data appears in a different region of the space than the previous one, this has to be taken into account. Therefore, c is continuously recalculated after every iteration and it affects the entire model where the z-scores of all instances will be changed as well.

The overall method is presented in Algorithm 4.

Algorithm 4: Active-COSMO

Data: Unlabeled data $\mathcal{U}$; query budget b

Result: Predicted labels all examples

```

1 Initialize threshold  $\theta$ 
2  $\mathcal{L} = \emptyset$ 
3 while  $|\mathcal{L}| \leq b$  do
4   Calculate weights vector  $W$  based on  $\mathcal{U}$  and  $\mathcal{L}$ 
5    $p(\mathcal{U}) \leftarrow \text{COSMO}(\mathcal{U}, W)$ 
6   select  $x^q$  according to query strategy
7   yield prediction about  $x^q$ 
8   get label  $\{\text{healthy}|\text{faulty}\}$  on  $x^q$ 
9   add  $x^q$  to  $\mathcal{L}$ 
10  update COSMO parameters
11 end
```

3.4 Query Strategies

In this paper we present two, diametrically opposite, query strategies. In Section 4, we show their respective benefits in the two settings, or modes, described above.

Figure 2: Illustration for the displacement of different type of instances when the most central object changes. Instances which have relative position are represented with red color while absolute positions are black. Green point shows the most central object.

One of the most common query strategy applied in active learning is to ask the user to label the instances which are least likely under the current model i.e., uncertainty sampling [14]. In our work, we measure the informativeness of an instance with its closeness to the threshold θ . Basically, the model achieves higher confidence on instances farther from the decision border about their true classes. Therefore, we employ query rule by sequentially choosing the instance that is closest to the threshold θ . For each object, let $p(x)$ be the deviation level computed by the current model. Then, the query instance x^q will be:

$$x^q = \arg \min_{x \in \{x_1, \dots, x_n\}} ||p(x_i) - \theta|| \quad (5)$$

The idea is that, when there is a limited budget for querying, each of them should be used to the highest potential. Information gain is higher when revealing instances in which the model is less confident. The query criteria based on uncertainty is, therefore, favorable when the labeling effort is heavily penalized.

In contrast to the uncertainty approach, the second query criteria consist of picking the examples in which the model is the most confident. According to the assumption that the learner has higher confidence when the point is far from the decision border, the instances having the maximum gap between its score and threshold θ gets selected to be presented to the human.

$$x^q = \arg \max_{x \in \{x_1, \dots, x_n\}} ||p(x_i) - \theta|| \quad (6)$$

The motivation behind this strategy is to start learning with instances where the classifier less likely makes mistakes. Under the assumption that all the data instances will have to be classified eventually, this selection order leads to delaying as many mistakes as possible. This greedy strategy aims at learning as much as possible before approaching the most “challenging” objects, i.e., the ones that are close to the decision boundary.

3.5 Adaptive Threshold

Detecting faulty examples is a hard task even if the model is capable of identifying deviations successfully since not every deviation or outlier in the system corresponds to a fault. We evaluate different strategies to determine a proper threshold to discriminate faulty and healthy instances using active learning. In the first method,

the threshold $\hat{\theta}$ is obtained by computing the average of deviation levels of all labeled objects, after receiving labels in every iteration. This threshold adaptation has been employed using SVDD in [9]

$$\hat{\theta} = \begin{cases} \arg \max_{x \in \mathcal{L}} \{p(x)\}, & \text{if } n_h \geq 0 \wedge n_f = 0 \\ \arg \min_{x \in \mathcal{L}} \{p(x)\}, & \text{if } n_h = 0 \wedge n_f \geq 0 \\ \frac{\sum p(x)}{n_h + n_f}, & \text{if } n_h \geq 0 \wedge n_f \geq 0 \end{cases} \quad (7)$$

where n_h and n_f is the number of instances in $\mathcal{L}$ labeled as healthy and faulty, respectively.

In the second approach, we only consider instances where their expected labels and observed labels (given by the user) are different with respect to their recent deviation levels and the current threshold. Let m_h is the number of healthy observations misclassified and m_f is the number of faulty observations misclassified by the recent model. The new threshold function is defined as:

$$\hat{t}_h = \begin{cases} \theta - \delta, & \text{if } m_f > m_h \\ \theta + \delta, & \text{if } m_f < m_h \end{cases}, \quad (8)$$

$$\delta = \frac{1}{vn} \sum_{i=1}^n |p(x_i) - \theta| \quad (9)$$

where the difference between the deviation level of x_i and the threshold θ , shows how far away x_i lies on the other side of the threshold, n is the number of observations and v is the regularization parameter.

The third approach is similar to the adaptation proposed in [9]. However, we also concern the class imbalance problem in this threshold adaptation strategy. If the majority of the queried observations are healthy, the threshold is increased towards the average of the queried healthy observation’s deviation levels. If the majority of queried observations are faulty, the threshold is decreased towards the average of the queried healthy observation’s deviation levels. This approach is shown in the following equation:

$$\hat{\theta} = \begin{cases} \arg \max_{x \in \mathcal{L}} \{p(x)\}, & \text{if } n_h \geq 0 \wedge n_f = 0 \\ \arg \min_{x \in \mathcal{L}} \{p(x)\}, & \text{if } n_h = 0 \wedge n_f \geq 0 \\ \frac{1}{2} \left(\frac{\sum p(x)}{n_h + n_f} + \frac{\sum p_f(x)}{n_f} \right), & \text{if } n_h \geq n_f \\ \frac{1}{2} \left(\frac{\sum p(x)}{n_h + n_f} + \frac{\sum p_h(x)}{n_h} \right), & \text{if } n_h < n_f \end{cases} \quad (10)$$

Where $p_f(x)$ corresponds to the deviation levels of the queried faulty observations and $p_h(x)$ corresponds to the deviation levels of the queried healthy observations. n_f and n_h correspond to the number of faulty observations and the number of healthy observations, respectively.

3.6 Adapting the Most Central Object

To discriminate faulty examples efficiently, we aim to locate the most central object in the center of healthy observations. However, our first approximation – in the absence of any labeled data – places it by computing the mean of the entire dataset. This location, then, is contaminated by anomalies. Our goal is to progressively update this state until it is obtained from purely normal data.

Therefore, whenever the user provides a new observation x_i^q , at iteration $i \in N$, the learner's most central object c is updated respecting to the label of the new data point. When the system receives a "faulty" label from the expert, it calculates the new c by adding a negative weight to the x^q instance, in effect pushing the most central object away. If the label is healthy, x^q is weighted with a positive value. This, in turn, will move the most central object closer to the x^q object, making points in this area less likely to be marked as deviations in the future. Figure 1 illustrates the change of the most central object that can take place over several iterations. The following equation summarizes the change:

$$\hat{c} = \left(1 - \frac{1}{n+1}\right)c + \left(\frac{w}{n+1}\right)x^q \quad (11)$$

3.7 Absolute and Relative Positions

Many real-world applications applying fault detection and monitoring are deployed in non-stationary environments[13]. Especially, if data has been collected over a long period, it is likely that the distribution changes over time. By proposing different adaptation strategies, we aim for our model to remain robust over such changes. Most of the time, the relative characteristics of a group of machines remain same during lifetime, while their operation changes in the absolute terms. In such cases, the feedback provided by the user in the past should be updated according to the shift in the underlying distribution. However, there are certain types of behaviors that are always faulty, regardless of their relations to the rest of the group. Therefore, we propose that an interactive learning method should treat those cases separately.

The feedback provided by the expert includes not only the true label of an example, but also the information whether the true class of that example is based on the relative or the absolute position in the feature space. If the feedback of an instance is marked as

Figure 3: Accuracy of applying different threshold adaptation methods of mode 1. Uncertainty sampling strategy for querying observations and bias weighting on queried observations are applied. The x-axis corresponds to the number of unseen data sets.

“absolute position”, it never changes its location over time, even if the overall data distribution drifts. This is useful for capturing deviations which are independent of operational changes.

On the other hand, if the position feedback is marked as “relative”, it shows that the deviation level of that observation is also relative to its peer group. Therefore, when the most central object changes, instances labeled as relative by the expert should shift accordingly (Figure 2).

In the future, we will consider other methods to handle concept drift such as global and local replacements, adaptive sliding windows, forgetting mechanisms etc. by incorporating this type of feedback.

4 RESULTS

The experiments in this work are based on two types of synthetic data and are designed to demonstrate the performances between different query strategies and model adaptations for the most central object and the threshold. We introduce different distributions representing the two groups of instances/examples i.e., the healthy group and the faulty group.

The first type of synthetic data is stationary, i.e., streaming data points of the two classes are generated from fixed distributions, where the mean of the distribution does not change over time. Healthy examples are generated from two-dimensional Gaussian distribution where one of the Gaussian corresponds to the majority of the group, i.e. performing the regular operation, and the other corresponds to a small group of examples performing a special type of operation (still healthy but centered at a different location). Faulty examples are generated from one two-dimensional Gaussian distribution:

Figure 4: Accuracy of different querying strategies and bias weighting of mode 1. Threshold adaptation method based on equation 10 is applied.

Figure 6: Accuracy of different querying strategies and bias weighting of mode 2. Threshold adaptation method based on Equation 10 is applied here.

Figure 5: Accuracy of applying different threshold adaptation method of mode 2. Uncertainty sampling strategy for querying observations and bias weighting on queried observations are applied.

$$\mu_{Healthy, Majority} \sim N\left(\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\right) \quad (12)$$

$$\mu_{Healthy, Special} \sim N\left(\mu = \begin{bmatrix} 0 \\ 3 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}\right) \quad (13)$$

$$\mu_{faulty} \sim N\left(\mu = \begin{bmatrix} 0 \\ -3 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix}\right) \quad (14)$$

For each unit, 10 samples of that unit were generated from $N(\mu, 0.02)$, where μ is drawn from $\mu_{Majority}$, $\mu_{Special}$ or μ_{Faulty} .

These samples can be considered as set of observations of a unit performing operation at different time.

The second type of data synthetic data drifts overtime. Healthy and Faulty examples were generated as follows:

$$\mu_{Healthy, Majority} \sim N\left(\mu = \begin{bmatrix} 0 \\ 0 + 10 \cdot t \end{bmatrix}, \Sigma = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\right) \quad (15)$$

$$\mu_{Healthy, Special} \sim N\left(\mu = \begin{bmatrix} 0 \\ 3 + 10 \cdot t \end{bmatrix}, \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}\right) \quad (16)$$

$$\mu_{faulty} \sim N\left(\mu = \begin{bmatrix} 0 \\ -3 + 10 \cdot t \end{bmatrix}, \Sigma = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix}\right) \quad (17)$$

The drifting data set demonstrates the differences of acquiring two types of expert's feedback.

We evaluate the performance of our approach using two different scenarios, or modes, as explained in Section 3.1.

The first mode aims to measure how the accuracy changes based on the number of queries as it is explained in Algorithm 1. It shows the trade-off between the labeling effort and the desired performance. The model is trained with different query budgets and then evaluated on a separate, unseen dataset. It is important to employ this setting in order to capture how the algorithm identifies the best observations to label thereby producing the highest classification accuracy with the least amount of human effort.

The second mode is designed to perform continuous, online self-monitoring. In this setting, all instances in the initial dataset need to be eventually classified (see Algorithm 2). After every decision that Active-COSMO makes, it checks whether this decision was correct according to the user feedback. The evaluation is the online performance that is achieved *while* learning (see Algorithm 2). This setting displays that it is possible to increase the online performance of the learner with an effective query strategy and user feedback. Although the model sequentially learns until the entire

Figure 7: Accuracy of using different expert feedback and querying method of mode 1. Biased weighting on queried observations and threshold adaptation method based on equation 10 is applied here.

Figure 8: Accuracy of using different expert feedback and querying method of mode 2. Biased weighting on queried observations and threshold adaptation method based on equation 10 is applied here.

dataset is exhausted, it shows that applying active-learning with different sampling strategies can be beneficial to maintain sustainable performance over time (Figure 6).

Experiments on Streaming Static Dataset

Figure 3 compares the accuracies of three different threshold adaptation methods based on the first type of evaluation. The result shows that adapting threshold according to equation 10, i.e., balancing the size of two classes, achieves the best performance.

Figure 4 compares the accuracies of two different query strategies, i.e., uncertainty sampling and most confident sampling, and two different weighting methods on queried samples, i.e., unbiased and with bias weights of 5, based on the first mode of evaluation. The result shows that the method with uncertainty sampling and biased weighting on queried samples achieves the best performance. Methods with uncertainty sampling perform better than the ones with the most confident sampling.

Figure 5 and 6 demonstrate the results from different methods on static dataset, evaluated with the second mode. 5 shows that, based on the second evaluation method, adapting the threshold based on balancing the queried examples outperforms the other two methods, which agrees with the result shown in Figure 3. Figure 6 shows that methods with biased weighting perform better than the ones without it. The method with uncertainty sampling and biased weighting performs better than other methods, which also agrees with the results from the first evaluation mode, shown in 4.

Experiments on Streaming Drifting Dataset

Figure 7 compares the performance of using different types of expert feedback and query strategy. As is shown in the figure, methods with relative position of queried observations as expert feedback performs better than the ones with absolute position as feedback type. However, figure 8 shows that all methods achieve similar performance according the second type of evaluation.

5 CONCLUSIONS AND FUTURE WORK

Combining self-monitoring methods based on peer-group analysis with interactive learning is an important and inspiring direction. Our early experiments have shown that the overall concept is sound and promising. This is only a single and quite simple experiment, of course, but we believe that the idea we present here, once fully developed, is going to prove very useful.

In this work, we propose several approaches combining fault detection with interactive learning strategies. The results of this work provide some indication that the overall concept towards interactive, life-long self-monitoring is promising. In particular,

Even though the evaluation shown here is quite brief and incomprehensive, we believe that once the idea we present is fully implemented, it will prove to be useful in practice.

In the future, we will extend our experiments with real data and improve query strategies. Currently, the active learning step is not followed by an optimization step. Although we update the model concerning recently labeled data, there is no measure of how well we are doing that. In the future, we will employ an optimization process as well.

One of the important features of COSMO algorithm is to look for the clues and interesting relationships among the signals and to build appropriate models to capture such relationships. In this paper, for simplicity of the presentation, we work directly in the data space. In the future, we would like to incorporate subjective interestingness, based on human feedback. It is especially important in changing environments to eliminate virtual drifts. Finally, at the moment all the feedback is stored verbatim, as a list of (example, label) pairs. This can grow quite fast, so we are investigating ways to aggregate this information, as well as to forget the irrelevant information over time.

REFERENCES

- [1] Naoki Abe, Bianca Zadrozny, and John Langford. 2006. Outlier detection by active learning. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06* (2006). <https://doi.org/10.1145/1150402.1150459>
- [2] Charu C Aggarwal. 2013. *Outlier Analysis* (1 ed.). Springer New York.
- [3] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35, 4 (2014), 105–120.
- [4] Stephen Boyd, Corinna Cortes, Mehryar Mohri, and Ana Radovanovic. 2012. Accuracy at the top. In *Advances in neural information processing systems*. 953–961.
- [5] Robert Burbidge, Jem J. Rowland, and Ross D. King. 2007. Active Learning for Regression Based on Query by Committee. *Intelligent Data Engineering and Automated Learning - IDEAL 2007* (2007), 209–218. https://doi.org/10.1007/978-3-540-77226-2_22
- [6] Shubhomoy Das, Weng-Keen Wong, Thomas Dietterich, Alan Fern, and Andrew Emmott. 2016. Incorporating Expert Feedback into Active Anomaly Discovery. *2016 IEEE 16th International Conference on Data Mining (ICDM)* (2016). <https://doi.org/10.1109/icdm.2016.0102>
- [7] Yuantao Fan, Slawomir Nowaczyk, and Thorsteinn Rögnvaldsson. 2015. Evaluation of Self-Organized Approach for Predicting Compressor Faults in a City Bus Fleet. *Procedia Computer Science* 53 (2015), 447–456. <https://doi.org/10.1016/j.procs.2015.07.322>
- [8] Yuantao Fan, Slawomir Nowaczyk, and Thorsteinn Rögnvaldsson. 2015. Incorporating Expert Knowledge into a Self-Organized Approach for Predicting Compressor Faults in a City Bus Fleet. *Frontiers in Artificial Intelligence and Applications* (2015). <https://doi.org/10.3233/978-1-61499-589-0-58>
- [9] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. 2013. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research* 46, 1 (2013), 235–262.
- [10] Tom S. F. Haines and Tao Xiang. 2013. Active Rare Class Discovery and Classification Using Dirichlet Processes. *International Journal of Computer Vision* 106, 3 (2013), 315–331. <https://doi.org/10.1007/s11263-013-0630-3>
- [11] Jingrui He and Jaime Carbonell. 2007. *Nearest-neighbor-based active learning for rare category detection*. 633–640.
- [12] T. M. Hospedales, Shaogang Gong, and Tao Xiang. 2013. Finding Rare Classes: Active Learning with Generative and Discriminative Models. *IEEE Transactions on Knowledge and Data Engineering* 25, 2 (2013), 374–386. <https://doi.org/10.1109/tkde.2011.231>
- [13] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2017. Detecting Change in Data Streams. *VLDB '04 Proceedings of the Thirtieth international conference on Very large data bases* 30 (2017), 180–191.
- [14] David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. *SIGIR '94* (1994), 3–12. https://doi.org/10.1007/978-1-4471-2099-5_1
- [15] Tomáš Pevný. 2016. Loda: Lightweight on-line detector of anomalies. *Machine Learning* 102, 2 (2016), 275–304.
- [16] Thorsteinn Rögnvaldsson, Henrik Norrman, Stefan Byttner, and Eric Jarpe. 2014. Estimating p-Values for Deviation Detection. *2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems* (2014). <https://doi.org/10.1109/saso.2014.22>
- [17] Thorsteinn Rögnvaldsson, Slawomir Nowaczyk, Stefan Byttner, Rune Prytz, and Magnus Svensson. 2018. Self-monitoring for maintenance of vehicle fleets. *Data mining and knowledge discovery* 32, 2 (2018), 344–384.
- [18] Nicholas Roy and Andrew McCallum. 2001. *Toward Optimal Active Learning through Sampling Estimation of Error Reduction*. 441–448.
- [19] Burr Settles. 2010. Active learning literature survey. *Computer Sciences Technical Report* 1648 (2010).
- [20] Burr Settles, Mark Craven, and Soumya Ray. 2017. Multiple-instance active learning. *Advances in Neural Information Processing Systems (NIPS)* 20 (2017), 1289? 1296.
- [21] H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92* (1992). <https://doi.org/10.1145/130385.130417>
- [22] John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.
- [23] Vikas Sindhwani, Prem Melville, and Richard D. Lawrence. 2009. Uncertainty sampling and transductive experimental design for active dual supervision. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09* (2009). <https://doi.org/10.1145/1553374.1553496>
- [24] Aarti Singh, Robert Nowak, and Parmesh Ramanathan. 2006. Active learning for adaptive mobile sensing networks. *Proceedings of the fifth international conference on Information processing in sensor networks - IPSN '06* (2006). <https://doi.org/10.1145/1127777.1127790>
- [25] Kalyan Veeramachaneni, Ignacio Arnaldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li. 2016. AI2: Training a Big Data Machine to Defend. *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)* (2016). <https://doi.org/10.1109/bigdatasecurity-hpsc-ids.2016.79>
- [26] Venkat Venkatasubramanian, Raghunathan Rengaswamy, Surya N. Kavuri, and Kewen Yin. 2003. A review of process fault detection and diagnosis. *Computers & Chemical Engineering* 27, 3 (2003), 327–346. [https://doi.org/10.1016/s0098-1354\(02\)00162-x](https://doi.org/10.1016/s0098-1354(02)00162-x)
- [27] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Conformal prediction*. Springer.
- [28] Cui Zhu, Hiroyuki Kitagawa, Spiros Papadimitriou, and Christos Faloutsos. 2004. OBE: Outlier by Example. *Advances in Knowledge Discovery and Data Mining* (2004), 222–234. https://doi.org/10.1007/978-3-540-24775-3_29