Adaptive Novelty Detection over Contextual Data Streams at the Edge using One-class Classification

Olga Jodelka[1], Christos Anagnostopoulos[1], Kostas Kolomvatsos[2]
[1]School of Computing Science, University of Glasgow, UK
[2]Department of Computer Science & Telecommunications, University of Thessaly, GR
2266755J@student.gla.ac.uk; christos.anagnostopoulos@glasgow.ac.uk; kostasks@uth.gr

***Abstract*—Online novelty detection is an emerging task in Edge Computing trying to identify novel concepts in contextual data streams which should be incorporated into models executed on edge nodes. We introduce an unsupervised adaptive mechanism for online novelty detection over data streams at the network edge based on the One-class Support Vector Machine; an instance of One-class Classification paradigm. Due to adjustable periodic model retraining, our mechanism timely recognises novelties and resource-efficiently adapts to data streams. Experimental evaluation and comparative assessment showcase the effectiveness and efficiency of our mechanism over real data-streams in identifying novelty conditioned on the necessary model retraining.**

***Index Terms*—Novelty detection; edge computing; contextual data streams; one-class classification; adaptation.**

I. Introduction

Novelty and anomaly detection have recently gained popularity in Edge Computing environments. Despite that, the terms *novelty* and *anomaly/outliers* are often confused or used interchangeably referred to identification of abnormal unexpected phenomena found in captured data [12]. Yet, there are many definitions of outliers and novelties in literature; ultimately, [9] states that outlier is 'an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism'. Novelties could then be initially defined as abnormal observations which with time form *patterns*, thus, no longer being abnormal. However, [7] states definitions of anomaly/outlier and novelty that distinguish between the terms: anomalies are described as *undesired* patterns, whereas novelties are emerging *new* concepts that should be incorporated to models. There are arguments claiming that novelty detection is more challenging since existing systems do not have enough knowledge about what contributes as a novelty [12], [19]. Novelty detection has various applications e.g., detecting broken environmental sensors in edge computing [7], [6] and intrusion detection [13]. In Internet of Things (IoT), increasing amounts of data streams gathered by edge nodes yielding issues of data quality [16], [15], [8]; at the edge, novelty detection mechanisms should be adopted to *expand* the knowledge on novel behaviours which mitigates data quality issues.

Novelty detection in stream processing splits into offline and online phase [7]. In offline phase, incremental statistical training of the model occurs locally on edge nodes, while real-time detection of new (previously unseen), unlabelled data occurs online. Offline phase depends on the availability of data labelling (supervised learning) or not (unsupervised learning) [1]. In edge computing, nodes should detect novelties using recent unlabelled data. We therefore focus on unsupervised learning coping with adaptation and model retraining.

A contextual data stream on an edge node is as a multivariate time-series (sequence) of observations of random variables captured sequentially in time [7]. In a node, data are not static, thus, a locally trained model over the initial data eventually becomes outdated with time. Newly incoming data could be classified as outliers when, in fact, they are novelties, i.e., data coming from different/unseen distributions than the original one(s) occurred during training *and* forming new valid patterns. This is also known as *concept drift* and *concept evolution*, which should expand/adapt to the current model. Evidently, to identify novelty on-line, it is deemed appropriate to determine periodic (not necessarily in regularly occurring intervals) retraining of the model over the recently captured data on edge nodes.

II. Related Work & Contribution

Thorough analysis of novelty detection is provided in [6] projected onto four perspectives: point difficulty, relative frequency, clusterdness and feature relevance. [6] claimed that model-based methods have marginal performance when dealing with dynamic data streams. Despite of this, supervised-based models, like Support Vector Machine (SVM), are widely used in detection tasks *given that* human experts have pre-labelled non-inlier data in training datasets, i.e., *before* training. Fundamentally, [17] introduced a paradigm that computes a decision function distinguishing regions in data space with different probability densities which can be the basis of *novelty detection under unsupervised learning*, i.e, no pre-labeling is required or is unavailable, which is not rare in edge computing environments. Modifications of the seminal SVM [5] rendered SVM-based models no longer to limit to linear data spaces, which has proved their capacity to cope with novelty identification *only* from knowledge extracted from the data [17], [8]. One-class SVM (OCSVM) [17] is a fundamental adaptation of SVM under the family of One-class Classification, which autonomously learns a decision boundary that achieves maximum separation between data points and origin under unsupervised learning. There are several OCSVM enhancements, e.g., [1] mitigating the impact of *known-in-advance* outliers on the decision boundary as opposed to inliers. Other OCSVM adaptations for novelty detection rely on Iterative Weighted Recursive Least Squares (IW-RLS) [3] and on deep belief network-based SVM [15]. The latter method is suitable for offline pre-labelling outlier detection over humongous data transferred from WSNs.

The work reported so far focuses on either unsupervised novelty detection on *static data* (i.e., no streaming data) or improvements based on previously known/labelled outliers typically operated over humongous static/at-rest data [13], [15], [8]. However, direct adoption of those methods to resource-constrained and dynamically data changing environments in edge nodes is not appropriate due to limited memory and computational capacity for inferring novelty. Moreover, supervised methods inherently require full availability of data streams' labelling which does not scale within edge computing environments. Our work focuses on adaptive memory-footprint controlled novelty identification over data streams in edge nodes. Yet, learning from data streams imposes challenges; the most common is concept drift. [7] formalises novelty detection in data streams introducing the concept drift and concept evolution challenges. Most decision-boundary algorithms become ineffective when confronted with such challenges, notably, concept drift itself makes it difficult to detect whether a new one occurs. Fundamental methods coping with concept drifts include ensemble of adaptive base-learners and windowing techniques dealing with limited memory [10], which we adopt in our paper. [7] and [2] proved that model *retraining* paradigm is the most appropriate way to cope with these issues concluding on certain limitations of current novelty detection approaches: (i) the assumption that *all* data points used to update a model are labelled. This is not applicable in cases where limited memory is available and/or where the most recent data are more important for model retraining than the old ones; (ii) the assumption that every change in data is considered as concept evolution. This fails to distinguish between known concepts and new concepts; (iii) the fact of ignoring recurrent concept drifts. This fails to recognize multiple concept drifts, which is not so rare in data streams. A sophisticated method for deciding on *when* to retrain an unsupervised novelty detection model taking into account resource constraints and inherent limitation (fundamentally memory, space/time complexity, and unavailability of data labelling) is required in edge computing.

Contribution: We rest on the One-class Classification (OCC) OCSVM paradigm and depart to adaptive novelty identification by investigating a resource-aware OCSVM-based mechanism. The fundamental differences lie on (i) adaptation to data stream trends, (ii) importance of the most recent points against oldest ones in a sliding-window framework, and (iii) adjustable technique for deciding when to retrain the model w.r.t. novelty detection rate and unnecessary model retraining. Adaptation is a key to our mechanism adjusting the model retraining frequency. Novelty identification is achieved by increasing our confidence that a non-inlier is a point which is completely new and can contribute to the capacity of the current model to expand and/or adjust its boundaries. The retraining decision is purely driven by the novelty rate and judgment on recent unnecessary model retrainings. This is evaluated against continuous retraining policies found in the literature [19], thus, leading us to introduce a *necessity rate* for model retaining showcased in evaluation Section IV-A over a real edge computing setting with computing & sensing nodes. Note: our mechanism is not tailored to specific OCSVM variant; in fact, any OCC can be adopted, which is also evidenced by adopting the variant in [19] for comparative assessment purposes.

III. Adaptive Novelty Detection at the Edge

A. OCSVM Preliminaries & Methodology Fundamentals

OCSVM [17] separates all N unlabeled data points from the origin (in feature space $\phi(\mathbf{x}) \in \mathcal{F}$) and maximizes the distance of this hyperplane to the origin. It learns a decision function $F(\mathbf{x}) \in \{1, -1\}$ which captures data regions with high probability density. $F(\mathbf{x}) = 1$ in a *small* region capturing inliers and $F(\mathbf{x}) = -1$ elsewhere with optimization objective:

$$\min_{\mathbf{w},\xi_i,\rho} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - \rho \tag{1}$$

$$\text{s.t. } \mathbf{w}^\top\phi(\mathbf{x}_i) \geq \rho - \xi_i, \xi_i \geq 0, \forall i = 1, \ldots, N, \tag{2}$$

where ν indicates upper bound on the fraction of non-inliers (points regarded as out-of-class) and lower bound on the number of points referred to as Support Vectors (SVs) with:

$$F(\mathbf{x}) = \sum_{i=1}^{m} a_i K(\mathbf{x}_i, \mathbf{x}) - \rho, \tag{3}$$

with $m < N$ SVs whose coefficients $a_i > 0$. Fig. 2 shows the decision boundary (learnt frontier) $F(\mathbf{x})$. In contextual data streams concepts do not stay constant. Since it is infeasible to store all data on the node due to limited resources, we adopt windowing techniques for storing the most recent data in a sliding window of size $W > 0$.

Definition 1: Let a discrete time domain $t \in \mathbb{T} = \{1, 2, \ldots\}$. A sliding window of size $W > 0$ is an ordered set of d-dim. points ranked by their time indices $\{\mathbf{x}_{t-W+1}, \ldots, \mathbf{x}_t\}$. By sliding the window at $t+1$, we discard oldest point $\mathbf{x}_{t-W+1}$ and insert the next one from the stream $\mathbf{x}_{t+1}$.

As only a part of data is observable at a time, we need to retrain the novelty detection model every time horizon $H > 0$. This periodically updates the model reflecting the *concept drifts*. The ratio $\frac{H}{W}$ plays significant role on how the trained model includes some historical points (used in previous retraining phases) and some new ones in the current retraining phase. We introduce sliding window-based methods that allow for retraining the model and investigate appropriate adjustments of H. We locally train an OCSVM model at the node using only the data residing in window W allowing the model to draw a temporary decision boundary. Such boundary is a non-linear function over the *current* SVs $m_t < W$. The amount m_t of SVs affects the boundary shape and therefore model's detection capacity (theoretically, the expectation of $\mathbb{E}[m]$ is controlled by the parameter ν). The node monitors how m_t changes with time reflecting adaptation to data streams and compares with ν estimating storage complexity $O(m)$ and detection complexity $O(m)$. New data are added to the window whereas old data are removed reflecting potential changes in concepts as well as allowing for forgetting outdated concepts. In principle, our method decides *when* to retrain the model influencing complexity and detection efficiency.

B. Continuous Model Adaptation

In the *baseline* Continuous Model Adaptation mechanism (CMA) in Algorithm 1, we assume that *each* incoming point potentially has an impact on the underlying concepts. After training the OCSVM on the initial window W, we retrain it over every single point (setting horizon $H = 1$). Hence, every small change in the data is immediately identified and the model is adjusted accordingly. A point $\mathbf{x}_t$ is labelled as a novelty (-1) or as inlier (1) by the current model F, which has been trained at $t-1$. The consistent and continuous increase of identified novelty and simultaneous lack of change of the inlier detection rate, suggests that each incoming point is novel, i.e., lies outside the decision boundary learnt by the model. However, if the novelty counter c_t remains constant ($\Delta c_t = 0$) and the inlier detection rate increases, it suggests that there is no change in the underlying data. Regardless of the label of $\mathbf{x}_t$ given by F (trained at $t-1$), we inject $\mathbf{x}_t$ to the window W and discard the oldest one. Although the label does not affect whether a point is added to the window, it affects the reasons for which it is done. If $\mathbf{x}_t$ is an inlier, it should be added to the window W as being part of the known concept already represented by the current model. If $\mathbf{x}_t$ is a novelty, it implies that the underlying concepts are changing. Hence, the point should be injected to the window W so that the model reflects these changes and be able to adapt to them. The model is then retrained at t obtaining new model F' and replacing the previous one.

Given two versions of the trained model: F, trained at $t-1$ and F', trained at current time t, we compare the identification labels of $\mathbf{x}_t$ by model F and the retrained model F' to determine whether the retraining was *necessary*. If $\mathbf{x}_t$ is considered inlier by *both* models, it means that the underlying concepts have not changed (i.e., the decision boundary learnt by both models is the same), thus, the retraining was unnecessary. As the re-training horizon affects model efficiency and availability of computational resources, the node monitors the number of *unnecessary retrainings* n_t. However, it is unlikely that the underlying concepts will change with each single point, which suggests that the retraining frequency in CMA is expected to be high, which motivated us to introduce an adjustable mechanism.

C. Adjusted-Frequency Model Adaptation

The adjusted-Frequency Model Adaptation (FMA) mechanism is provided in Algorithm 2. Since model retraining at every single point is computationally inefficient (due to a potentially relatively high number of model retrainings compared with the number of new concepts occurring in actual data streams), in FMA, we retrain the model with adjusted retraining horizon $H_t > 1$. It is expected to reduce the complexity in comparison to CMA. In FMA, the model is firslty trained on the initial window W. The label of $\mathbf{x}_t$ is determined by model F while $\mathbf{x}_t$ is injected to window W and the oldest one is removed. We do not retrain the model until horizon H_t is reached. We investigate two sub-cases: Case 1: $H_t < W$ and Case 2: $H_t \geq W$ to examine the impact of the *training memory* on the model. The adjusted H_t parameter controls the $\frac{H_t}{W}$ ratio. If $H_t < W$ (ratio less than 1) then during retraining we still retain some of previous knowledge in the new model, i.e., the created decision boundary is *partially* based on the SVs of the previous model. In this case, the knowledge of previous and new concepts could provide the model with better explanation of how the data change thus allowing it to refine the decision boundary. If $H_t \geq W$ then we completely forget the previous model (no previously SVs are transferred to the new retrained model). That is the decision boundary is created using *only* the SVs of the new model. As the model is retrained less frequently, it does not always reflect the underlying concepts. Since the knowledge of the previous model becomes irrelevant after some time, it is advantageous to forget the SVs of the older model to 'make space' in the memory for the SVs representing current concepts. Hence, in principle, the adjustment rules depend on novelty and unnecessary retraining trend: $\Delta H_t = \alpha H_{t-1}$ if $\Delta(\frac{n_t}{t}) \geq 0$ and $\Delta(\frac{c_t}{t}) \leq 0$, i.e., we prolong the next retraining phase by increasing horizon H_t by a factor $\alpha\%$ if the rate of unnecessary retrainings increases and the rate of detecting novelty decreases; otherwise, $\Delta H_t = -\alpha H_{t-1}$ (we set $\alpha = 0.1$). Note: H_t takes over once H_{t-1} has elapsed. Based on this transient adjustment, we expect to increase novelty rate before the retraining of the model and then a sudden drop right after. This is connected with the horizon length of retraining which affects the model's adaptability to changing concepts. It also results in smaller amount of unnecessary retrainings. A re-training is considered necessary

Algorithm 1 Continuous Model Adaptation (CMA).

```
Require: Sliding window $W$; parameter $\nu$
  Train OCSVM over window $W$ obtaining model $F$
  for $t = 1, \dots,$ do
    Observe data point $\mathbf{x}_t$
    if $F(\mathbf{x}_t) == -1$ then
      $c_t \leftarrow c_{t-1} + 1$; non-inlier/novelty
    end if
    Inject $\mathbf{x_t}$ to window $W$; remove oldest point
    Retrain OCSVM obtaining new model $F'$
    if $F(\mathbf{x}_t) == 1$ & $F'(\mathbf{x}_t) == 1$ then
      $n_t \leftarrow n_{t-1} + 1$; unnecessary retraining
    end if
    Replace model $F$ with $F'$
  end for
```

by comparing the labels of $\mathbf{x}_t$ assigned by F (trained $t-H_{t-1}$ instances ago) and current F' (trained at t). Node replaces then model F with model F'. FMA reduces the inherent computational complexity, while initial H_0 depends on the window size W.

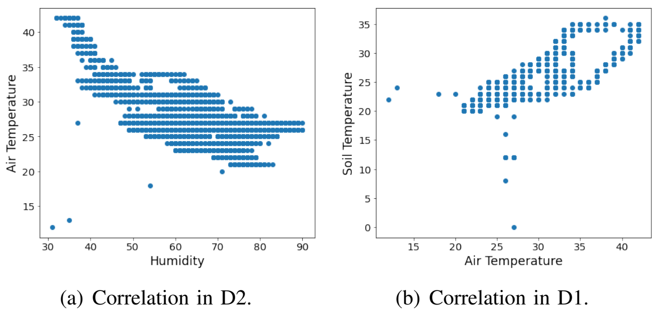
Algorithm 2 Adjusted-Frequency Model Adaptation (FMA).

```
Require: Sliding window $W$; parameter $\nu$
  Train OCSVM over window $W$ obtaining model $F$
  for $t = 1, \dots$ do
    Observe data point $\mathbf{x_t}$
    if $F(\mathbf{x_t}) == -1$ then
      $c_t \leftarrow c_{t-1} + 1$; non-inlier/novelty
    end if
    Inject $\mathbf{x_t}$ to window $W$; remove the oldest point
    if $(t\%H_t == 0)$ then
      Retrain OCSVM obtaining new model $F'$
      if $F(\mathbf{x_t}) == 1$ & $F'(\mathbf{x_t}) == 1$ then
        $n_t \leftarrow n_{t-1} + 1$; unnecessary retraining
      end if
    end if
    Replace model $F$ with $F'$ and adjust $H_t$
  end for
```

IV. Performance Evaluation

A. Experimental Setup & Performance Metrics

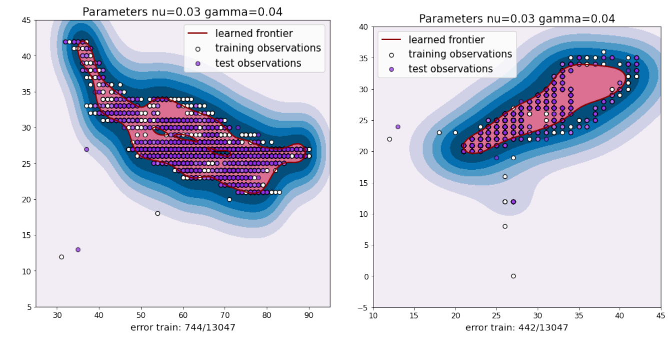
Datasets: We use the 3-dim. Greenhouse real datasets[1] ($d = 3$), where edge nodes (Coral DevBoard with DHT11 sensors in an open space GlassHouse) capture data streams of humidity (x_1), soil (x_2), and air (x_3) temperature every 3 minutes and generate two 2-dim. datasets: D1 with x_2 and x_3, and D2 with x_1 and x_3, which have the strongest correlation; see Fig. 1. Each dataset contains 16,039 2-dim. points.



(a) Correlation in D2. (b) Correlation in D1.

Fig. 1. Correlation of humidity and air-temperature (D2 dataset) (a); correlation of soil and air-temperature (D1 dataset) (b) of GlassHouse data.

OCSVM Parameters Tuning: We performed a series of experiments to choose the optimal set of parameters ν and γ for OCSVM to learn the optimal decision boundary; see Fig. 2 using $k = 10$ fold-cross validation; we implemented OCSVM using [14] and locally executed on the edge nodes. Due to our data non-linearity, we adopted the Radial Basis Function kernel widely used in SVM [18]: $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$, where $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$. The $\nu \in (0, 1)$ parameter indicates an upper bound on the fraction of margin errors and a lower bound of the fraction of SVs relative to the total number of points in window W [4]. By increasing ν, it increases the amount of SVs thus the decision boundary fits as many training points as possible (risk of model overfitting); while, by decreasing ν, it generalizes the model (the decision boundary is simplified). This happens because there are less SVs deliberately considered to be outliers, however, we risk model underfitting. Hence, ν it tuned to trade-off between overfitting and generalization. The γ represents kernel's width influencing the shape of the decision boundary and the size of the region covered [17]. By decreasing γ, it leads to simpler boundary shape, thus, decreasing the amount of SVs (model underfitting); conversely, increasing γ means stricter decision boundary (model overfitting). We experimented with $\nu \in \{.01, \dots, .5\}$ with step .05 and $\gamma \in \{.001, \dots, .2\}$ with step .01; our goal was to find the pairs of ν and γ such that OCSVM learns the most optimal decision boundary. By using trial-and-error and observing the error train changes (number of outliers in training set), we found the pair $(\nu, \gamma) = (.03, .04)$ that optimizes the model and the boundaries are shown on Fig. 2 per dataset. Both models obtain good results shown on Fig. 2(a) (D2) and 2(b) (D1).



(a) Decision boundaries for D2 (humidity & air temperature). (b) Decision boundaries for D1 (soil & air temperature).

Fig. 2. Decision boundaries on D1 (b) and D2 (a) of the GreenHouse dataset; $(\nu, \gamma) = (.03, .04)$.

Comparison Models & Performance Metrics: The window size $W = 1000$ points in D1 and D2 reflecting memory capacity of edge nodes, while horizon H varies from the adjustment rules (Cases 1 and 2). We evaluate and compare CMA and FMA. CMA adopts the continuous model retraining paradigm in [19] using quarter-sphere OCSVM [11], which is replaced here with the ν-OCSVM [17] for fair comparison. We record the amount of unnecessary retraining n, total retrainings, amount of identified novelty c, which allows us to observe how novelty develops over time, per model per dataset. After each retraining, we calculate the number of SVs m per each model, whose expected value should converge to ν, and investigate how m changes over time. Note: metric m reflects the memory requirement for building decision

[1]http://iprism.eu/index.php/datasets-ppts

TABLE I
EXPERIMENTAL RESULTS

Metric	CMA	FMA
Unnecessary retrainings n	D1:12341; D2:11772	D1:18; D2:7
Total retrainings	D1:15309; D2:14698	D1:30; D2:15
Novelty counter c	D1:2605; D2:3537	D1.Case-1:5542 D1.Case-2:6013 D2.Case-1:6084 D2.Case-2:6480
Average #SVs $\mathbb{E}[m] \approx \nu$	D1:36; D2:32	D1.Case-1:36 D1.Case-2:37 D2.Case-1:33 D2.Case-2: 33

boundaries and detection complexity which is $O(m)$; see (3).

B. Performance & Comparative Assessment

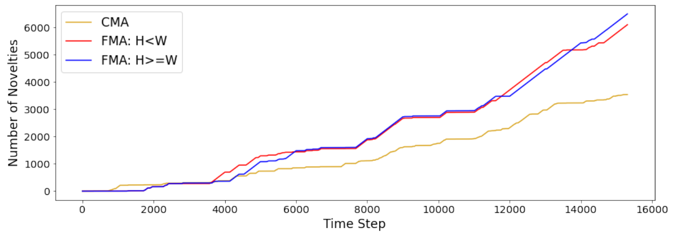
Experimental results are shown in Table I and in Figures 3 and 4. We obtain similar model behaviours indicating that our methods are not dataset dependent. In CMA and FMA, the amount of unnecessary retrainings n is greater than 50% of the total retrainings. It suggests that concepts in data do not change as frequently, and therefore, model re-training at every point (CMA policy) is inefficient. FMA significantly reduces the rate of total retrainings (3-4 orders of magnitude in D1 and D2) indicating efficiency of the horizon adjustment rules.

The evolution of the amount of identified novelties in CMA and FMA is shown in Figures 3(a) and 4(a). This illustrates the trade-off between the ability of the model to detect novel concepts and incorporate them into the normal pattern, and computational efficiency. In D1 and D2, FMA detected 2 to 3 times more novelties than CMA due to its adjustable horizon. There was a marginal difference in number of novel points detected by the two FMA Cases, with Case 2 ($\frac{\mathbb{E}[H]}{W} \geq 1$) detecting 10% more novel points than Case 1. This indicates a memory-less behaviour, where, in Case 2, the model forgets entirely its previously trained boundaries, thus, between two consecutive retrainings, the model is 'surprised' with completely new unseen points. The sudden increase in FMA's identified novelties as observed in Figures 3(a) and 4(a) is correlated with the model retraining time indicating the rate at which the model turns obsolete. This is not observed in CMA's behaviour as frequent retraining adapts it gradually to any change in the data. Every change is instantaneously reflected by CMA and incorporated as normal pattern, however, at the expense of decreasing computational efficiency. On the contrary, FMA balances by prolonging retraining, thus, avoiding unnecessarily wasting resources while identifying novelty, deemed appropriate in our resource-constrained context.

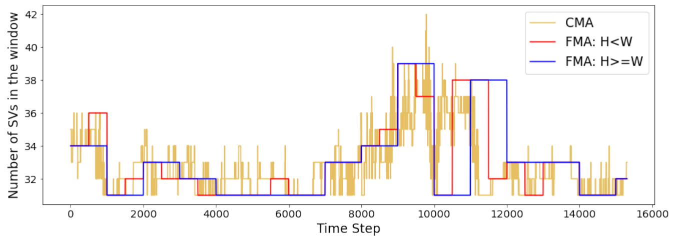
The evolution of the amount of SVs m_t is shown in the Figures 3(b) and 4(b). Frequent retraining in CMA results in large fluctuations of m_t with relatively high variance. Conversely, m_t in FMA remains constant for longer time periods due to adjustable retraining horizon. On average, m_t in FMA comes with less variance, thus, the space (memory) and time complexity is controlled in FMA increasing our certainty on the expected required resources. The ν parameter of OCSVM controls the model detection time and space complexity and the number of SVs. This results in less complex model and easier decision boundaries should the amount of SVs be low. Both CMA and FMA lead to expected m close to ν as shown in Table I; the average m is similar for both methods in both datasets and does not exceed 3.45%, i.e. it does not deviate much from the theoretical $\nu = 3.0\%$. However, CMA results in higher variance of SVs related to continuous retrainings. As a result we are confident that the memory footprint remains small and constant for the duration of the novelty detection task. Fluctuations in m occur also due to dynamic nature of data. Therefore, we do not deal with unpredictable memory requirements, which is undesirable in edge computing environments.

The number of *marginal non-inliers* ℓ detected in the window depends on ν; ℓ is upper bound of points per window annotated by model F as non-inliers to establish the decision boundary. Figures 3(c) and 4(c) show the evolution of ℓ_t with time indicating the correlation with the model retraining. The ℓ metric represents the rate in which the model turns obsolete. The fluctuations in ℓ observed in CMA comes from highly frequent model retrainings and its adaptation to concept drifts. In FMA, the ℓ metric increases until the model is retrained after which a sudden drop occurs. This reflects FMA's ability to adapt to new concepts. The spikes suggest that FMA, especially in Case 2, turns obsolete thus not capturing effectively all the underlying concepts. This is the feedback to the Hadjustment rules, which trigger model retraining allowing for incorporating novelties to normal pattern and drastically decreasing ℓ. In Case 2, FMA becomes memory-less thus, the rate of increase of ℓ triggers model retraining. In Case 1, however, model retrainings are more frequent, thus ℓ's spikes are significantly lower w.r.t. Case 2 improving the adaptability of the model. FMA (Case 1) exhibits again an effective trade-off by reducing unnecessary retraining (being eminently more resource efficient than CMA) and being more adaptive to identified novelty (than FMA (Case 2)).

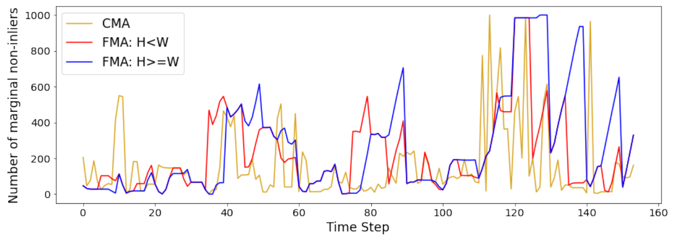
CMA and FMA (Cases 1 and 2) use only recent edge data to identify novelty *explicitly* in an unsupervised manner; the ground truth is purely data-driven determined by the decision boundaries (no human-centric labelling) while the windowing framework allows for transferring knowledge between model retrainings. Memory requirement is more predictable in FMA rather than in CMA governed by ν resulting in non-complex models determined by a small amount of SVs. CMA adapts to every change in the data and easily incorporates novelty patterns as evidenced by steadily increasing c. However, most of these frequent retrainings are unnecessary yielding CMA inefficient in terms of computational resources. FMA's adjustment rules render it computationally efficient. Notably, FMA Case 1 allows partial knowledge transfer between retraining thus incrementally adjusts (expands/shrinks) the boundary following streaming trend and novelty occurrence. FMA's flexibility to control resources consumption and knowledge adaptation to data streams makes it appropriate for adoption to edge computing environments.



(a) Number of novelties (c_t) in D1 (soil & air temperature).

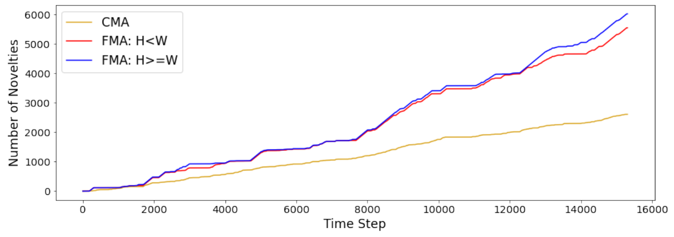


(b) Number of SVs (m_t) in D1 (soil & air temperature).

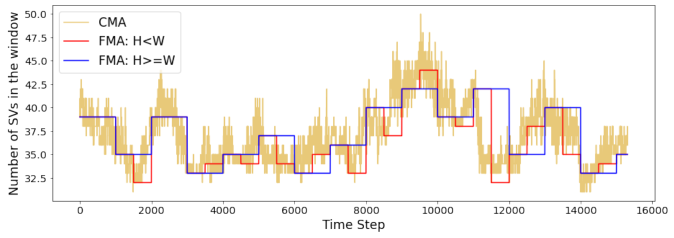


(c) Number of marginal non-inliers (ℓ_t) in D1 (soil & air temperature).

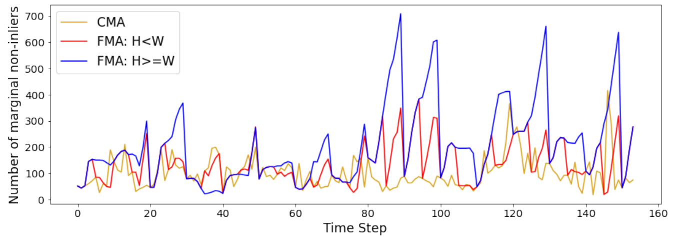
Fig. 3. Comparison of CMA and FMA (Cases 1 and 2) over D1 (soil & air temperature data): number of novelties in (a); number of SVs in (b); number of marginal non-inliers (shown every 100 points for readability) in (c).



(a) Number of novelties (c_t) in D2 (humidity & air temperature).



(b) Number of SVs (m_t) in D2 (humidity & air temperature).



(c) Number of marginal non-inliers (ℓ_t) in D2 (humidity & air temperature).

Fig. 4. Comparison of CMA and FMA (Cases 1 and 2) over D2 (humidity & air temperature): number of novelties in (a); number of SVs in (b); number of marginal non-inliers (shown every 100 points for readability) in (c).

V. Conclusions

We investigate online light-weight novelty detection mechanism over data streams in edge computing focusing on One-Class SVM. Our mechanism adjusts periodic model retraining to follow data streams trends and identifies novelties being resource-efficient adapting to concepts. Our evaluation indicates the effectiveness of our mechanism taking into account the novelty rate and unnecessary model retraining to adjust future model retraining. Future work includes data-streams aware adjustability and confidence-driven novelty detection.

References

[1] M. Amer, M. Goldstein, and S. Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *ACM SIGKDD Outlier Detection and Description*, pages 8–15, USA, 2013.

[2] C. Anagnostopoulos. Edge-centric inferential modeling analytics. *Journal of Network and Computer Applications*, 164:102696, 2020.

[3] J. Arenas-García, V. Gómez-Verdejo, and Ángel Navia-Vázquez. RLS adaptation of One-Class SVM for time series novelty detection. In *Learning 04 International Conference*, Elche, Spain, 2004.

[4] C.-C. Chang and C. J. Lin. Training v-support vector classifiers: Theory and algorithms. *Neural Computation*, 13:2119–2147, 2001.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[6] A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. A meta-analysis of the anomaly detection problem. *Computer Science*, 2015.

[7] E. R. F. et al. Novelty detection in data streams. *Artificial Intelligence Review*, 45:235–269, 2016.

[8] V. Gomez-Verdejo, J. Arenas-Garcia, M. Lazaro-Gredilla, and Navia-Vazquez. Adaptive one-class support vector machine. *IEEE Trans. Signal Processing*, 59(6):2975–2981, 2011.

[9] D. M. Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.

[10] T. R. Hoens, R. Polikar, and N. V. Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1:89–101, 2012.

[11] P. Laskov, C. Schäfer, I. Kotenko, and K.-R. Müller. Intrusion detection in unlabeled data with quarter-sphere support vector machines. 27(4):228–236, 2004.

[12] J. Ma and S. Perkins. Time-series novelty detection using one-class support vector machines. In *Intl. Conf. Neural Networks*, volume 3, pages 1741–1745, Portland, OR, USA, 2003. IEEE.

[13] Q. T. Nguyen, K. Phuc Tran, P. Castagliola, T. Thu Huong, M. K. Nguyen, and S. Lardjane. Nested one-class support vector machines for network intrusion detection. In *7th IEEE ICCE*, pages 7–12, 2018.

[14] F. e. a. Pedregosa. Scikit-learn: Machine learning in Python.

[15] Y. Qiao, X. Cui, P. Jin, and W. Zhang. Fast outlier detection for high-dimensional data of wireless sensor networks. *International Journal of Distributed Sensor Networks*, 16, 2020.

[16] D. Rao, V. N. Gudivada, and R. V. Vijay. Data quality issues in big data. *IEEE Big Data*, pages 2654–2660, 2015.

[17] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylort, and J. Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12:582–588, 1999.

[18] J.-P. Vert, K. Tsuda, and B. Schölkopf. *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA, USA, 2004.

[19] Y. Zhang, N. Meratnia, and P. Havinga. Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks. In *AINA*, pages 990–995, 2009.