Machine Learning Model Updates in Edge Computing: An Optimal Stopping Theory Approach

Ekaterina Aleksandrova, Christos Anagnostopoulos, Kostas Kolomvatsos
School of Computing Science, University of Glasgow, UK
2133352A@student.gla.ac.uk; {christos.anagnostopoulos; kostas.kolomvatsos}@glasgow.ac.uk

***Abstract*—This work studies a sequential decision making methodology of *when to update* machine learning models in Edge Computing environments given underlying changes in the contextual data distribution. The proposed model focuses on updates scheduling and takes into consideration the optimal decision time for minimizing the network overhead. At the same time it preserves the prediction accuracy of models based on the principles of the Optimal Stopping Theory (OST). The paper reports on a comparative analysis between the proposed approach and other policies proposed in the respective literature while providing an evaluation of the performances using linear and support vector regression models. Our evaluation process is realized over real contextual data streams to reveal the strengths and weaknesses of the proposed strategy.**

***Index Terms*—Edge computing, machine learning model updates, communication efficiency, optimal stopping theory.**

I. Introduction

The Internet of Thing (IoT) has been gaining popularity since the end of the 20th century. IoT has allowed the transformation of small scale computing environments into vast ecosystems, whose cores, i.e., cloud data centers, require a massive computational power to process all the received contextual data. In addition, there is the need for a substantial network overhead in order to receive all the raw data. However, as this has proven to be inefficient [20], a computing paradigm has been provided in the form of Edge Computing (EC), whose rationale is to *push most of the computations to the edge of the network*, e.g. sensors, mobile devices, etc. This allows to take advantage of the computational and sensing capabilities of modern devices and deliver the locally processed contextual data to the cloud in the form of partially or entirely extracted knowledge [3]. The edge-centric rationale contributes as a method for decreasing network traffic, as the delivered statistical learning model representations of the locally stored data have significantly smaller size compared to the raw contextual sensed data. The EC is expected to significantly reduce the required latency in the provision of knowledge while involving less computational power than cloud resources. However, if the acquired knowledge is still transferred to the cloud with a high frequency due to changes in the underlying data, then the considerable communication overhead remains along with other complications explained further ahead in this paper.

The main problem which this paper aims is estimating an *optimal waiting time criterion* for updating and delivering Machine Learning (ML) models from the edge network to the cloud based on sequential observations over multivariate contextual data. The proposed scheme should be able to decide when a ML model is to be sent from an edge node to an edge gateway to minimize the communication overhead and, *in parallel*, to preserve the prediction accuracy of the generated ML models. We express this as the research question: *How can the ML model delivery time and model accuracy at the network edge be maximized while minimizing the communication overhead and computational complexity at the edge node level?* We address this question adopting the principles of the Optimal Stopping Theory (OST) and propose a time-optimized stochastic model for ML model updates.

The paper is organized as follows: Section 2 presents related work, while Section 3 introduces the proposed methodology built on the OST. Section 4 reports on the performance and comparative assessment with other sequential decision making policies using real datasets and provides a summarized analysis, while Section 5 concludes the paper.

II. Related Work & Contribution

A. Knowledge Sharing in Edge Computing Environments

Given a sample path in a distributed environment consisting of multiple sensing and computing edge nodes, local edge gateways and a data center node as an endpoint, the multi-hop transmission increases energy consumption when the gateways are busy and the data have to be passed to the central node [14]. When data processing is not performed at the edge of the network and given that n d-dimensional data points $\{\mathbf{x}_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^d$ are sensed, collected and disseminated through the network, it results to at least n transmissions. On the other hand, if the data points $\{\mathbf{x}_i\}$ are locally processed at the edge nodes close to their source, the derived result corresponding to knowledge derivation can be expressed through a model representation of the data, where only the representative statistical parameters are disseminated through the network. This results in reduced energy consumption and expected communication overhead [17]. Moreover, wireless sensor networks in IoT environments used in remote locations, e.g., rain forest sensors [19], surface and mine monitoring [2], water pollution detectors [1], are usually being powered by a battery. Therefore, the energy consumption of the sensing devices should be minimized as much as possible to reduce the amount of human interaction required to replace the exhausted energy source and the corresponding costs. If the sensors are placed at locations with difficult access, that could also increase the financial cost or even cause danger to the person

responsible for the replacement. Moreover, sensor systems rely on renewable energy, most widely used is solar energy, which could be slower to harvest in environments such as rain forests [19], therefore, having a power efficient device is required in order to have a well-functioning system. Another concern with communicating processed data on every sensing & reporting period is that it is possible that the data contains bias, e.g., missing or corrupted data points [4]. When disseminating them to the gateway, it can potentially be used to make inaccurate predictions of the sensed/monitored environment. This can be avoided by *delaying* the reporting of data and the derived ML model update before making sure that the bias is not in fact a novelty in the data or due to data missingness.

A way of minimizing the energy cost in edge devices is by reducing the expected number of data transmissions [7]. This can be achieved by introducing a certain *delay in the delivery of the up-to-date data* [3], [4], [5], at the cost of allowing a reasonable reconstruction error in the data domain [8]. Adopting this delay-based rationale, our distributed approach departs from the data domain and heads to the derived ML models domain of the underlying data. Specifically, the concept is as follows: initially, a ML model, e.g., linear regression or support vector regression model, is generated at the edge nodes, which is sent to the edge gateway. In the case that the node does not detect a significant change between the initially modelled data via the derived ML model and the currently sensed data, no communication is made for update purposes. Thereofre, making the decision on *when* to send a new updated ML model to the edge gateway *before* the accuracy is expected to degrade given an application specific error tolerance threshold should be studied from an time-dependent stochastic optimization angle.

B. Time-optimised Sequential Decision Making

The OST is epxected to solve the stochastic problem stated above, i.e., choosing a *best* time instance to take a given action. In our context, we need to decide when to stop observing changes on the derived ML model and send the *new* ML model to the edge gateway. The action is based on sequentially observed random variables in order to maximize expected reward trading off the communication overhead and expected accuracy. Based on this abstraction of the considered problem, several OST-based variations are adopted as a basis to abstract problems in computing science notably: the House-Selling (HS) problem and the Quickest Change Detection (QCD) problem [6]. The HS problem explores the attempt to stop and *sell* a house at the highest offer by taking into account a cost, such as living cost or real estate agency commission, for each rejected offer. This problem relates to the approach that waiting can be penalized, the way Tian et al. use it in their *cost-aware* update policy [18]. In our context, the decision on whether to update an outdated ML model with the current one is based on how much it was lost (how big the "regret" is) since the update with the better model was not performed in the past. Once a tolerance threshold is violated, an update is inevitable, which also prevents additional retraining computations on the edge node. The QCD problem deals with the exploration of the context distribution and stopping when a change is detected while penalizing false alarms [6]. Research has been focusing on two major approaches based on assumptions on the underlying data distribution. Shiryaev [15] provides an optimal solution to the problem using a Bayesian approach, while, on the other hand, a non-Bayesian approach is adopted using the well-known Cumulative Sum (CuSum) method [13]. The latter was proven to be optimal by Moustakides [12] based on Lorden's formulation of the problem [11]. Recently, Lau and Tay [16] introduce the "critical change" and "nuisance change" in a distribution, where the critical change is of highest importance and should be detected, as opposed to the nuisance change, which should be ignored. They propose two algorithms using Bayesian and non-Bayesian assumptions, compared to [13] and [15].

Contribution: Given these established methodologies for sequential decision making, we depart from the related work in the direction of detecting changes *not* in the underlying data *but* on the prediction capability of the derived ML models on edge nodes. Specifically, our contributions are:

1) An analytical time-dependent stochastic optimization model relying on the principles of OST that minimizes the communication overhead while preserving the prediction accuracy of the ML models in EC environments;
2) Comparative assessment of our model with established change detection rules and baseline solutions using real data-sets;
3) A statistical significance-based methodology for hyper-parameter optimization when applying the proposed optimal decision making policy.

III. Time-optimized Model Updates Methodology

We propose a *time-optimized ML model update postponing* policy in light of reducing the communication rate in the network edge but preserving the quality of prediction. We consider a distributed/EC environment with edge nodes capable of sensing and processing data and edge gateways. Initially, the edge node is responsible for gathering/sensing multivariate contextual data and generating a ML model $y = f(\mathbf{x})$ over the first W data observations $\{(\mathbf{x}_i, y_i)\}_{i=1}^{W}$, i.e., sliding window of length W. The ML model f is then temporarily stored on the edge node and is also communicated with the edge gateway. On each next observation round, the datum $(\mathbf{x}, y)$ is received and appended to the currently stored window W, while the oldest observation $(\mathbf{x}, y)$ is discarded. A new ML model $y = f'(\mathbf{x})$ can then be incrementally updated or generated representing the current updated dataset. By fitting the dataset on the old f and the new f' ML models, we obtain the two corresponding Mean Squared Errors (MSE) indicating the predictability capability of the models: (i) MSE $e = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2$ from the ML model f the edge gateway, which has been previously received from the edge node; (ii) MSE $e' = \frac{1}{n}\sum_{i=1}^{n}(y_i - f'(\mathbf{x}_i))^2$ from the ML model f' based on the most up-to-date sensed data at the edge node. Given that the predictability of the latest ML model f' of the

edge node satisfies certain criteria, the decision is made to communicate the f' to the edge gateway. This also requires that the temporarily stored outdated ML model f at the edge device is also updated to correctly represent the disseminated local knowledge in the EC environment.

A. Optimal Postponing Policy

The proposed Optimal Postponing (OP) policy for ML model updates is based on the *change in the error distribution* using the cumulative sum of the absolute error difference between the two MSEs e and e' at time instance of observation $t > 0$. Let us consider the definitions of the absolute error difference Z_t and cumulative error difference S_t:

$$Z_t = |e'_t - e_t| \quad (1)$$

$$S_t = \sum_{k=1}^{t-1} Z_k + Z_t = S_{t-1} + Z_t \quad (2)$$

Consider also the application-specific *error tolerance* threshold $\Theta > 0$ defined to determine the *acceptable* error sum. This will be used to determine if the edge node should proceed with a ML model update or not. Under this context, we define the random variable V_t as the reward of a ML model update decision reflecting the trade-off between communication efficiency and presentation of the model predictability w.r.t. tolerance Θ:

$$V_t = \begin{cases} t, & \text{if } S_t \leq \Theta, \\ -B, & \text{if } S_t > \Theta. \end{cases} \quad (3)$$

That is, when the cumulative error difference is less than a predefined tolerance Θ, we desire to delay the ML model update, thus, we obtain a reward proportional to the waiting time up to t (given that each time instance t corresponds to a new data observation). On the other hand, should the accumulation of the error differences exceeds Θ, then we impose a penalty. The penalty is expressed via a penalty factor $B > 0$ denoting that the current cumulative S_t exceeds the tolerance Θ, thus, the edge node could have sent the updated ML model to the edge gateway. To be communication efficient, the edge node desires to *postpone* the ML model update, thus, increasing the value of V_t as much as possible. However, as long as the edge node delays the model update, then the cumulative error S_t approaches Θ, thus increasing the risk of exceeding it. The challenge is for the cumulative error difference S_t to reach *as close to the tolerance* Θ *as possible*, but without exceeding this. We formalize then our problem as follows:

Problem 1: *The edge node should find the best time instance t^* such that the expected reward for delaying a ML model update is maximized, i.e., the optimal stopping time t^* achieves the essential supremum: ess* $\sup_t \mathbb{E}[V_t]$.

We provide an estimate of the optimal stopping time in Problem 1 as stated in the Theorem 1.

Theorem 1: *The optimal stopping time t^* that maximizes the essential supremum ess* $\sup_t \mathbb{E}[V_t]$ *of the reward function in (3) in Problem 1 is the first time instance $t > 0$ such that:*

$$F_Z(\Theta - S_t) \leq \frac{t+B}{t+1+B},$$

where $F_Z(z)$ is the cumulative density function (CDF) of the absolute error difference $Z_t = |e_t - e'_t|$.

Proof of Theorem 1: From the reward in (3) we obtain the expected current reward $\mathbb{E}[V_t]$:

$$\begin{aligned} \mathbb{E}[V_t] &= t \cdot P(S_t \leq \Theta) - B \cdot (1 - P(S_t \leq \Theta)) \\ &= (t+B) \cdot P(S_t \leq \Theta) - B. \end{aligned}$$

Let now the filtration $\mathbb{F}_t = \{S_1, S_2, \ldots, S_t\} \cup \{Z_1, Z_2, \ldots, Z_t\}$ be the realization of all the random variables up to t, which is the whole information the edge node has accumulated up to t. The conditional expectation of the reward V_{t+1} given $\mathbb{F}_t$ is then:

$$\mathbb{E}[V_{t+1}|\mathbb{F}_t] = (t+1+B) \cdot P(S_{t+1} \leq \Theta|\mathbb{F}_t) - B.$$

Based on (2) and the fact that the probability of the sum S_{t+1} at $t+1$ being less then or equal to Θ given filtration $\mathbb{F}$ equals the CDF of Z at the value of $\Theta - S_t$, i.e., $P(S_{t+1} \leq \Theta|\mathbb{F}_t) = P(S_t + Z_{t+1} \leq \Theta|\mathbb{F}_t) = P(Z_{t+1} \leq \Theta - S_t|\mathbb{F}_t) = F_Z(\Theta - S_t)$, we obtain that:

$$\begin{aligned} \mathbb{E}[V_{t+1}|\mathbb{F}_t] &= (t+1+B) \cdot P(S_{t+1} \leq \Theta|\mathbb{F}_t) - B \\ &= (t+1+B) \cdot P(Z_{t+1} \leq \Theta - S_t|\mathbb{F}_t) - B \\ &= (t+1+B) \cdot F_Z(\Theta - S_t) - B. \end{aligned}$$

We postpone sending the ML model *if we expect* that the next iteration will increase the reward value, i.e., we stop at the first instance t where the current reward is higher than the expected future reward at $t+1$ or where $V_t = t \geq \mathbb{E}[V_{t+1}|\mathbb{F}_t]$:

$$\begin{aligned} t &\geq (t+1+B) \cdot F_Z(\Theta - S_t) - B \\ F_Z(\Theta - S_t) &\leq \frac{t+B}{t+1+B}, \end{aligned}$$

which completes the proof of Theorem 1. ■

Algorithm 1 illustrates the local process in the edge node of our optimal policy. **Note:** The CDF $F_Z(z)$ is approximated using training pairs of error differences $\{z_i\}$ before the process starts. This is achieved by the incremental CDF learning algorithm of Kernel Density Estimation (KDE) being space efficient and having constant time complexity [10], thus, not burdening the computations on the edge node.

B. Policies under Comparison

In order to assess the performance of our method, we implemented four other policies, which rely either on certain statistics of the data or on pure randomness, namely: the Median-based Policy, the Random Policy, the Accuracy Policy, and the optimal CuSum Policy [13].

Algorithm 1 Time-optimized Postponing Policy (OP)

Initial ML model f
$Z_0 = 0$; $S_0 = Z_0$; $t = 1$
while TRUE **do**
  sense $(\mathbf{x}_t, y_t)$
  update/derive new model f' w.r.t. $(\mathbf{x}_t, y_t)$
  calculate e_t and e'_t; $Z_t = |e_t - e'_t|$; $S_t = S_{t-1} + Z_t$
  if $F_Z(\Theta - S_t) \leq \frac{t+B}{t+1+B}$ **then**
    $f \leftarrow f'$ ▷ update & communicate model to gateway
    $Z_0 = 0$; $S_0 = Z_0$; $t = 1$
  else
    $t \leftarrow t + 1$
  end if
end while

1) Median-based Policy: This policy has an initially learned the median m_T from the previously seen absolute error difference values given that the initial ML model is never updated. The rationale behind this is to explore the worst case scenario of the error difference. This assumes that the closer the timestamps of the two models f and f' are, the lower the error difference between them is and on the contrary, the further away the timestamps of the two models are, the higher their error difference is. Using a fraction $\alpha \in (0, 1]$ of the median m_T, the policy determines whether the current absolute error difference is a tolerable amount or if it indicates that the newly sensed data is significantly outdated and the associated ML model needs to be communicated with the edge gateway. This policy is expected to reduce communication by being more tolerant towards initial small abrupt changes but detects continuously increasing values. The policy periodically updates the median value m_T, i.e., every T pre-defined observations. Algorithm 2 illustrates the local median-based process in the edge node.

Algorithm 2 Median-based Policy

Initial median m_T; initial ML model f; $t = 1$
while TRUE **do**
  sense $(\mathbf{x}_t, y_t)$
  update/derive new model f' w.r.t. $(\mathbf{x}_t, y_t)$
  calculate e_t and e'_t; $Z_t = |e_t - e'_t|$;
  if $Z_t > \alpha \cdot m_T$ **then**
    $f \leftarrow f'$ ▷ update & communicate model to gateway
  end if
  if $t \mod T = 0$ **then**
    update median m_T
  end if
  $t \leftarrow t + 1$
end while

2) Random-based Policy: This policy is intended to be sending updates of the latest ML model to the edge gateway with the *same* probability $p > 0$ as the OP policy. The aim is to show that *even if* the number of updates is approximated to the optimal value by OP, the accuracy of the models at the edge gateway would still suffer a decrease: that is because OP *knows when* to send an update to maximize (3). Algorithm 3 illustrates the local random-based process in the edge node.

Algorithm 3 Random-based Policy

Initial ML model f; $t = 1$
while TRUE **do**
  sense $(\mathbf{x}_t, y_t)$
  update/derive new model f' w.r.t. $(\mathbf{x}_t, y_t)$
  $k \leftarrow$ random(0,1)
  if $k \leq p$ **then**
    $f \leftarrow f'$ ▷ update & communicate model to gateway
  end if
  $t \leftarrow t + 1$
end while

3) Accuracy-based Policy: This policy compares the MSE e of the old model f with the MSE e' of the most up-to-date model f'. Once it detects a decrease in the accuracy, the latest model is sent to the edge gateway. The policy aims to present a baseline solution for reducing communication but preserving accuracy as high as possible shown in Algorithm 4.

Algorithm 4 Accuracy-based Policy

Initial ML model f; $t = 1$
while TRUE **do**
  sense $(\mathbf{x}_t, y_t)$
  update/derive new model f' w.r.t. $(\mathbf{x}_t, y_t)$
  calculate e_t and e'_t
  if $e_t > e'_t$ **then**
    $f \leftarrow f'$ ▷ update & communicate model to gateway
  end if
  $t \leftarrow t + 1$
end while

4) Cumulative Sum (CuSum) Policy: The Cusum algorithm implemented here relies on the recursive form of the CuSum [13]. CuSum is based on two assumptions for the distributions representing the two hypotheses, where we decide which hypothesis represents the current sample of data. Firstly, the no-change hypothesis $\mathcal{H}_0$ refers to the "good distribution", which in our context is assumed to be the distribution of the absolute error differences when the edge gateway model receives an up-to-date model on every observation on the edge node. On the contrary, the "bad distribution" represented by the changed distribution hypothesis $\mathcal{H}_1$ is assumed to be the distribution of the absolute error difference when the model is communicated only on the first iteration and then never again. The assumption is made that the absolute error differences $Z_0, Z_1, \ldots, Z_n$ are continuous independent random variables. In this case, a Gamma Γ distribution is a choice to represent the error difference values, as shown in Fig.1, relying on the two parameters: scale and shape of the Probability Density Function (PDF) $P_Z(z)$ of the Z distribution. When a new absolute error difference $z_t = |e_t - e'_t|$ is calculated, it is then fitted in the $P_Z(z)$ for each of the two distributions:

the *good* $P_0(z_t)$ and the the *bad* $P_1(z_t)$ corresponding to the $\mathcal{H}_0$ and $\mathcal{H}_1$ hypotheses, respectively. The logarithmic ratio $\log \frac{P_1(z_t)}{P_0(z_t)}$ belonging to either the good or the bad distribution determines whether $\mathcal{H}_0$ or $\mathcal{H}_1$ is rejected. Once the cumulative sum of the logarithmic ratios minus the minimum value of the current ratios exceeds an initially determined threshold Φ, as proposed in [13] for optimal concept drift detection, the up-to-date model is sent from the edge node to the edge gateway *indicating a significant change in the predictability behaviour of the ML model* shown in Algorithm 5.

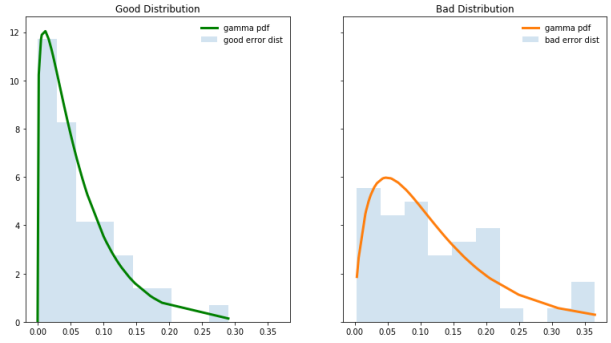

Fig. 1. Gamma distribution approximation of *good* and *bad* $P_Z(z)$ of the error difference Z over the GNFUV data using Linear Regression ML model.

Algorithm 5 *CuSum* Policy

Initial ML model f; $t = 1$
sense $(\mathbf{x}_t, y_t)$
update/derive new model f' w.r.t. $(\mathbf{x}_t, y_t)$
calculate e_t and e'_t; $Z_t = |e_t - e'_t|$;
log-ratio $r_t = \log \frac{P_1(Z_t)}{P_0(Z_t)}$; sum-ratio $u_t = r_t$
while TRUE **do**
  $t \leftarrow t + 1$
  sense $(\mathbf{x}_t, y_t)$
  update/derive new model f' w.r.t. $(\mathbf{x}_t, y_t)$
  calculate e_t and e'_t; $Z_t = |e_t - e'_t|$;
  calculate $r_t = \log \frac{P_1(Z_t)}{P_0(Z_t)}$; sum-ratio $u_t = u_{t-1} + r_t$
  if $u_t - \min_{k<t}\{u_k\} > \Phi$ **then**
    $f \leftarrow f'$ ▷ update & communicate model to gateway
    $r_t \leftarrow 0$; $u_t \leftarrow 0$
  end if
end while

IV. Performance Evaluation

A. Data Sets

Two real-date time-series datasets are used in order to apply the proposed policy OP and compare with the policies: media-based (M), accuracy-based (A), random (R), and CuSum (C).

GNFUV Unmanned Surface Vehicles Data Set [8]: this dataset is produced by four Unmanned Surface Vehicles (USV) sensing devices which observed and recorded the multivariate contextual environmental data using temperature and humidity sensors on the sea surface. The collected data will be associated with the Linear Regression ML model derived on each USV in the experiments for each policy. The generated linear regression models are aimed at predicting the environmental humidity based on the sensed temperature.

Gas sensors for Home Activity Monitoring Data Set [9]: this dataset contains readings from temperature, humidity and 8 metal-oxide (MOX) gas sensors of a contained environment whether or not a stimuli is presented. The collected data will be used in the Support Vector Regression (SVR) ML model with RBF kernel functions to assess all the policies. The generated SVR ML models predict the level of MOX gas based on the 2-dim vector of the detected humidity and temperature.

B. Experimentation with Linear Regression Models

The experiment using linear regression models was performed using the initial 100 data-points of the data-set for any pre-processing analysis depending on the targeted policy and the rest of the dataset was used to simulate an online approach using each of the discussed policies. Three of the policies require parameterization, which needs to be mentioned in advance along with the global parameter on the window size, which is set to be $W = 25$ for all experiments. The Median-based policy was performed using an α fraction of the median updated every $T = 100$ data observations; α was set to 0.5. The CuSum policy relies on the threshold $\Phi = 2$, up to which the cumulative sum of the logarithmic ratio is allowed to increase. The OP policy requires two parameter values, one for the error sum threshold Θ and one for the penalty value B. The box-plot analysis was performed by specifying a threshold Θ for the OP policy and running the simulation with penalty values ranging from $B = 2$ to $B = 15$. In Figure 2 for the USV PI3 (refer to [8]) it is shown that the highest mean on update delay is achieved using a threshold $\Theta = 3$ and a penalty value $B = 14$. These values are then used in the complete simulation of the policy comparison. The absolute error rate and the communication rate for each policy are depicted in Figure 3 and Figure 4 respectively.

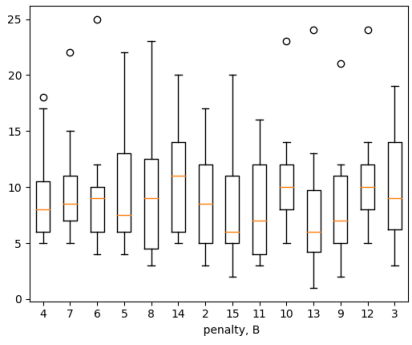

Fig. 2. Expected postponing time boxplot for $\Theta = 3$ and $B \in \{2, \ldots, 15\}$ for the OP policy.

As can be seen from Figure 3, the plot is very inconsistent due to the low accuracy of the linear regression models. The absolute error difference for the OP policy preserves a similar rate to policy A for six iterations, which is aimed at keeping the most accurate model. However, as seen in Figure 4, the OP

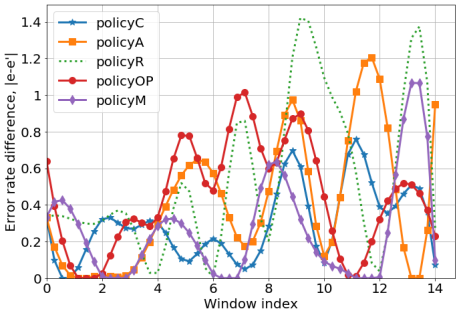

Fig. 3. Absolute error difference $Z = |e - e'|$ for USV PI3 [8] vs window size w using Linear Regression.

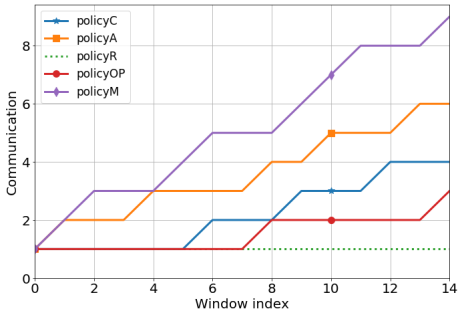

Fig. 4. Communication for USV PI3 [8] with window $w = 25$ using Linear Regression.

policy waits six iterations more compared to policy A before sending a ML model update. The number of iterations between two ML model updates are referred to as *waiting times* and they are further investigated by performing an ANOVA test and a Tukey's Honestly Significant Difference (HSD)[1] test for multiple mean differences in Table I. The results show p-values less than 5% which points to a *statistically significant* difference between the policy means for USV PI3 and USV PI4 on their waiting times and absolute error rates. We also perform a follow-up Tukey HSD test, which compares the means for each couple of policies and the results are plotted in Figure 5 and Figure 6.

TABLE I
ANOVA TEST RESULTS FOR LINEAR REGRESSION

p-value for ML model postponing time		
USV PI3	$1.24 \cdot 10^{-30}$	≤ 0.05
USV PI4	$7.89 \cdot 10^{-14}$	≤ 0.05
p-value for absolute error		
USV PI3	$1.24 \cdot 10^{-13}$	≤ 0.05
USV PI4	$2.77 \cdot 10^{-17}$	≤ 0.05

The plot in Figure 5 shows that the OP policy has a statistically significant higher waiting time (ML model postponing time) compared to policy M, policy C, and policy A and the same mean with the policy R, as intended. This shows the advantage in communication reduction introduced by the OP policy. Moreover, we perform the Tukey's HSD test on the absolute error difference of each policy along with policy E: the policy E communicates the up-to-date models on each iteration and is intended as a *lower bound*, which shows the absolute error rate of the models at its lowest. The results in Figure 6 show that the accuracy of the ML models does not suffer a significant deterioration compared to policy E and the other communication reduction policies but preserves the highest postponing time. Moreover, the plot shows that randomly updating the ML models leads to a *significant increase* in the absolute error, therefore, the OP policy does not just correctly guess the optimal number of updates but *also* the optimal time *between* the updates.

[1]The interest reader could refer to: J. Tukey, 'Comparing Individual Means in the Analysis of Variance', Biometrics, 5(2):99–114, 1949.

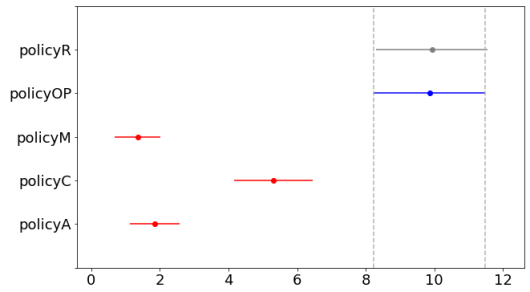

Fig. 5. Policy comparison in terms of statistical significance on the expected ML model postponing/waiting time in Linear Regression experiment.

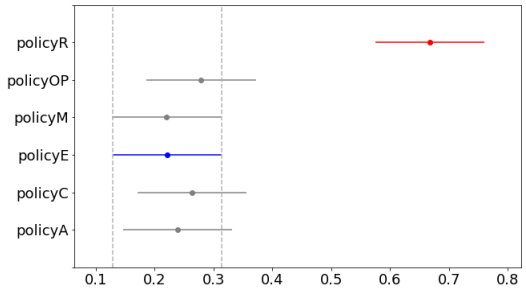

Fig. 6. Policy comparison in terms of statistical significance on the absolute error rate in Linear Regression experiment.

C. Experimentation with Support Vector Regression Models

After analyzing the second dataset, it was found that the initial 100 data-points of humidity and temperature contain a *concept drift* in the observed pattern. However, the rest of the data used in the implementation of the online algorithms have no noticeable changes, which results in no communication between the edge node and the edge gateway. This called for placing an artificial change (concept drift in the underlying distribution) in the data. One gradual long term change and one abrupt short term change affecting only the MOX sensor data and preserving the genuine data stream for the humidity and temperature. This is aimed to simulate a new pattern in the data which will cause a change in the communicated models. The gradual long term change occurs from the 50-th data point (observation) of the start of the experiment, where the MOX

sensor values are increased with 10% of the mean value in the next 12 observations and after that the change is kept persistent throughout the rest of the simulation. This drift in the data is intended to serve as a critical change [16] which should be detected by the algorithm and acted upon as soon as possible. The second abrupt short term change occurs in observation 108. The change in the MOX sensor values is increased with 2.5% of the mean value and then decreased to the genuine data-stream in the next 4 observations. This drift in the data distribution is intended to serve as a nuisance change [16], which should be of least importance to the algorithm.

The global parameters for this experiment are set the same way as in the experimentation using the linear regression models: window size $W = 25$ and initial pre-processing data containing 100 data points. The policy M uses $\alpha = 0.5$ and the cumulative sum threshold for the CuSum policy is set to $\Phi = 0.5$. For the OP policy, the cumulative sum threshold was set to $\Theta = 1$ as it allowed the needed sensitivity for the detection of the critical change. The box-plot in Figure 7 shows the ML model postponing times when performing for the MOX sensor R5 using $B \in \{2, \ldots, 15\}$. The highest mean waiting time is achieved using penalty $B = 3$. The results with this setting are shown in Figures 8 and 9.

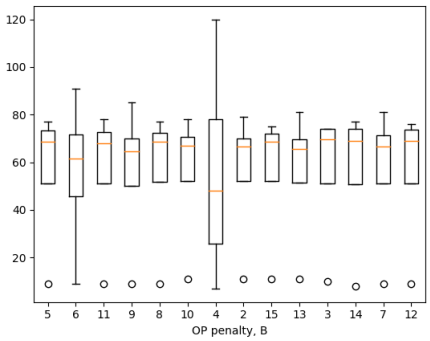

Fig. 7. Expected postponing time box-plot for $\Theta = 1$ and B values in the range 2 to 15 for the OP policy using SVR ML models.

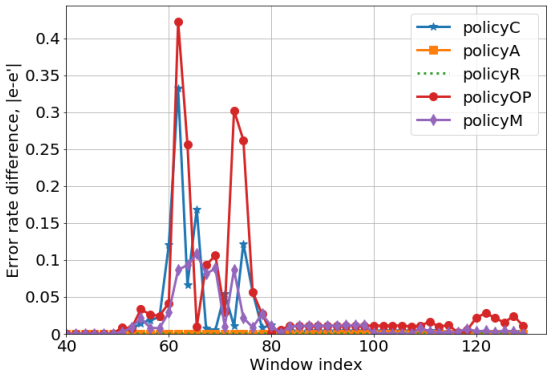

Fig. 8. Absolute error difference Z for MOX sensor R5 with window $w = 25$ using SVR ML models with RBF Kernel having a concept drift change at the 40-th observation.

In Figure 8, the absolute error line is distorted as expected after the critical change and less affected after the nuisance change. The error appears to increase the most for the CuSum policy and the proposed OP policy and less for the Median-based policy. An interesting observation is that the policy relying only on the accuracy (policy A) appears to be persistently close to 0, which can be explained by the good quality predictions that the SVR model delivers. However, when we focus our attention to the communication rate of the policies in Figure 4, despite the allowed error by the CuSum policy, the critical change appears to cause a communication overhead mid and post change, where even the Median-based policy manages to decrease the absolute error at the cost of less communication. On the contrary, the OP policy allows a high error rate but at the cost of just two sent messages as a result of the critical change, which happens shortly after the change and the nuisance change causes an updated model about 40 observations after the change. The waiting times for each policy is investigated further by performing ANOVA and Tukey's HSD tests for multiple mean differences in Table II .

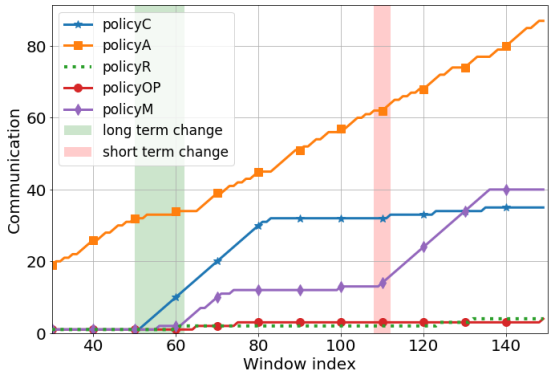

Fig. 9. Communication for MOX sensor R5 with window $w = 25$ using SVR ML models with RBF Kernel.

TABLE II
ANOVA TEST RESULTS FOR SVR

p-value for ML model postponing time		
MOX sensor R3	$9.19\ 10^{-18}$	≤ 0.05
MOX sensor R5	$1.47\ 10^{-32}$	≤ 0.05
p-value for absolute error		
MOX sensor R3	$6.00\ 10^{-23}$	≤ 0.05
MOX sensor R5	$1.06\ 10^{-11}$	≤ 0.05

The ANOVA test produces p-values less than 5%, which shows that there is a statistically significant difference between the means of the waiting time and the absolute error for sensors R3 and R5. Since the hypothesis that the policy means were the same was rejected, we performed the Tukey's HSD test with results plotted in Figure 10 and Figure 11. The plots show a significantly higher waiting time for the OP policy but also a very high error rate compared to the base accuracy of policy E in Figure 11. Analyzing the results it shows that the Median-based policy performs better than both the CuSum and the OP policy with regards to communication and error rate.

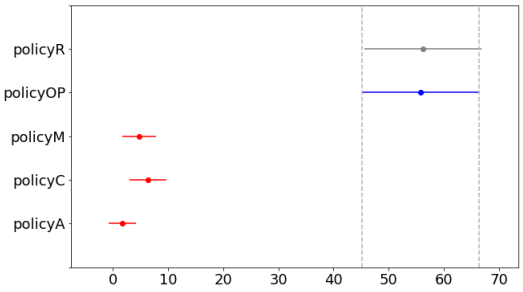

Fig. 10. Policy comparison in terms of statistical significance on the expected ML postponing/waiting time in SVR experiment.

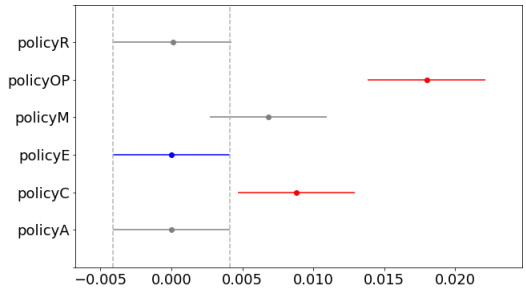

Fig. 11. Policy comparison in terms of statistical significance on the absolute error rate in SVR experiment.

D. Evaluation Summary

The results of the evaluation performance are concluded in the summarized Table III, which shows the trade-offs for each policy and the situations in which each one is recommended. For instance, when dealing with models which have high quality predictions over data of a small window size w, if we are willing to 'sacrifice' some of the accuracy then a basic policy such as the median-based policy can deliver the desired results. However, if the models are to be used occasionally without pressing short term deadlines then the OP policy eventually delivers the up-to-date accurate ML models and prevent a communication overhead as required in edge computing environments. On the contrary, if the generated ML models are expected to produce low quality predictions, then the CumSum policy and especially the OP policy are viable options. They expected to reduce the communication between the sensing/edge nodes and the edge gateways and preserve the initial quality of the contextual data and the corresponding derived machine learning models.

TABLE III
POLICY EVALUATION SUMMARY

Policy	High quality predictions	Low quality predictions
policy C	high error & high communication	✓
policy M	✓	high communication
policy OP	high error	✓
policy A	✓	high communication

V. CONCLUSIONS

This paper presents a time-optimized sequential decision making technique for optimal ML model update at the network edge based on the principles of the Optimal Stopping Theory. The proposed policy was assessed using real datasets in edge computing environments with multivariate time-series and compared with baseline solutions and the well-known optimal cumulative sum algorithm. The results showed that our policy preserves the accuracy of the generated ML models and significantly reduces the communication overhead, thus, being appropriate in efficient knowledge sharing in edge computing.

ACKNOWLEDGEMENT

This research is funded by EU-H2020 GNFUV (#Grant 645220) and EU-H2020 MSCA INNOVATE (#Grant 745829).

REFERENCES

[1] K. Adu-Manu et al. Water quality monitoring using wireless sensor networks: Current trends and future research directions. *ACM Trans. Sen. Netw.*, 13(1):4:1–4:41, Jan. 2017.
[2] M. A. Akkaş. Using wireless underground sensor networks for mine and miner safety. *Wireless Networks*, 24(1):17–26, Jan 2018.
[3] C. Anagnostopoulos. Time-optimized contextual information forwarding in mobile sensor networks. *J. Parallel Distrib. Comput.*, 74(5):2317–2332, May 2014.
[4] C. Anagnostopoulos. Quality-optimized predictive analytics. *Applied Intelligence*, 45:1034–1046, 06 2016.
[5] C. Anagnostopoulos and K. Kolomvatsos. A delay-resilient and quality-aware mechanism over incomplete contextual data streams. *Information Sciences*, 355–356:90–109, 2016.
[6] T. S. Ferguson. Optimal stopping and applications [http://www.math.ucla.edu/ tom/stopping].
[7] N. Harth and C. Anagnostopoulos. Quality-aware aggregation amp;amp; predictive analytics at the edge. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 17–26, Dec 2017.
[8] N. Harth and C. Anagnostopoulos. Edge-centric efficient regression analytics. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 93–100, July 2018.
[9] R. Huerta, T. Mosqueiro, J. Fonollosa, N. F. Rulkov, and I. Rodriguez-Lujan. Online decorrelation of humidity and temperature in chemical sensors for continuous monitoring. *Chemometrics and Intelligent Laboratory Systems*, 157:169–176, 2016.
[10] E. Hwang and D. W. Shin. Kernel estimators of mode under $$\psi$$-weak dependence. *Annals of the Institute of Statistical Mathematics*, 68(2):301–327, Apr 2016.
[11] G. Lorden. Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, 42:1897–1908, 12 1971.
[12] G. Moustakides. Optimal stopping times for detecting changes in distributions. *Annals of Statistics*, 14, 12 1986.
[13] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1–2):100–115, 1954.
[14] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3:1, 10 2016.
[15] A. Shiryaev. On optimum methods in quickest detection problems. *Theory of Probability & Its Applications*, 8(1):22–46, 1963.
[16] T. Siang Lau and W. P. Tay. Quickest change detection under a nuisance change. pages 6643–6647, 04 2018.
[17] Z. Tan, Y. Liu, and Z. Zhang. Performance requirements on energy efficiency in wsns. In *2011 3rd International Conference on Computer Research and Development*, volume 3, pages 159–162, March 2011.
[18] H. Tian, M. Yu, and W. Wang. Continuum: A platform for cost-aware, low-latency continual learning. *ACM Symposium on Cloud Computing (SoCC 18)*, 2018.
[19] T. Wark, W. Hu, P. Corke, J. Hodge, A. Keto, B. Mackey, G. Foley, P. Sikka, and M. Bruenig. Springbrook: Challenges in developing a long-term, rainforest wireless sensor network. pages 599–604, 01 2009.
[20] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang. A survey on the edge computing forthe internet of things. *IEEE Access*, 6:6900–6919, Jan 2018.