Consistency of the Fittest: Towards Dynamic Staleness Control for Edge Data Analytics

Atakan Aral[0000−0002−2281−8183]✉ and Ivona Brandic[0000−0001−7424−0208]

Institute of Information Systems Engineering,
Vienna University of Technology, Vienna, Austria
{atakan.aral,ivona.brandic}@tuwien.ac.at
http://rucon.ec.tuwien.ac.at/

Abstract. A critical challenge for data stream processing at the edge of the network is the consistency of the machine learning models in distributed worker nodes. Especially in the case of non-stationary streams, which exhibit high degree of data set shift, mismanagement of models poses the risks of suboptimal accuracy due to staleness and ignored data. In this work, we analyze model consistency challenges of distributed online machine learning scenario and present preliminary solutions for synchronizing model updates. Additionally, we propose metrics for measuring the level and speed of data set shift.

Keywords: Edge Computing · Data Analytics · Consistency · Staleness.

1 Introduction

Traditional way of data production and consumption is being revolutionized by new generation Internet based services such as smart cities, buildings, grids, factories and many other applications of Internet of Things. In this ongoing paradigm shift, not only volume of data explodes, but also it is generated in distributed fashion and consumed in real-time. Accordingly, the way such data is processed and analyzed is also subject to change [17, 28]. Many applications require near real-time reaction based on streaming data. Such applications, including *intelligent traffic management*, *spam or fraud detection*, *transactive energy control* and *computational advertising*, require fast response to the events detected in distributed streams. Hence, traditional batch processing, where data is aggregated in a central processing facility (e.g. massive cloud data center), is no longer feasible for such applications due to the high cost of data transmission [26]. This cost includes both network delay and bandwidth usage.

Edge computing paradigm, which aims to bring processing power of cloud to the closer proximity to where data is being generated or used, intrinsically matches above-described requirements. One realization of this approach is so-called Cloudlets [27] that are located in business promises such as restaurants or public offices, much like wireless access points today, and serve as a local cloud to nearby clients. Another possibility is to utilize micro data centers that

contain multiple servers and provide computational capabilities at the edge of the network [10]. Regardless of the implementation choices, edge computing will benefit near real-time data stream processing (DSP) tasks in two distinct ways: (i) it replaces sensor-to-cloud round trips and consequent network latency; (ii) it saves significant bandwidth capacity by confining majority of data transmission to the local area network. Distributing DSP tasks that involve machine learning (ML) steps, however, is not straightforward. One particular issue is to maintain a consistent ML model that can be updated as data streams evolve. Our aim in this work is to better understand the model consistency challenges with distributed online machine learning (DOML) scenario and provide initial ideas for potential solutions. To that end, we provide background information on the DOML paradigm and non-stationarity along with a motivational scenario in Section 2. We introduce a novel inference accuracy optimizing mechanism for synchronizing model updates, which we call *consistency of the fittest*, in Section 3. Furthermore, in Section 4, we propose three metrics for measuring the level of non-stationarity and in the subsequent Section 5, present preliminary numerical results. Finally, we discuss related work in Section 6 and conclude the paper in Section 7. To the best of our knowledge, this study is the first to address ML model consistency challenges within edge computing context and also the first attempt to quantitatively measure the extent of non-stationarity in ML models.

2 Background

2.1 Distributed Online Machine Learning

When real time decision making and high velocity data streams are involved, DOML is a viable alternative to centralized data processing / ML techniques. In this scenario, data originating from geographically distributed sources (e.g. IoT sensors, client computers, streaming media publishers, etc.) are processed at a nearby edge computing node. Here, both inference and online training steps are executed in each node. The former is about deriving conclusions (e.g. classification, prediction) from a ML model, whereas the latter is the continuous process of improving and adapting that model. Fig. 1 demonstrates such distributed architecture where `DS` are data streams and `EN` are edge nodes. Based on input data (i) to the current ML model at each `EN`, actions (ii) are determined and sent back to the distributed actuators (e.g. traffic control signals, in-home smart devices). One issue in this scenario is that each `EN` has access to only a local fragment (`DS`) of all generated data, hence local model (iii) trained with that fragment is suboptimal. A centralized parameter server is typically implemented in order to easily combine and synchronize new information learned by distributed `EN`s as a global ML model [13]. This model can be hosted at a cloud data center (`DC`) and updated in iterative fashion based on contributions from `EN`s. Stale models at `EN`s should be replaced with the current global model (iv) so that they are more accurate in their inference and they build upon a global checkpoint avoiding multiple branches of models.

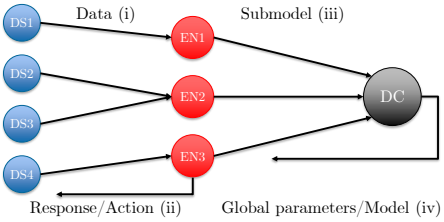

Fig. 1. DOML architecture.

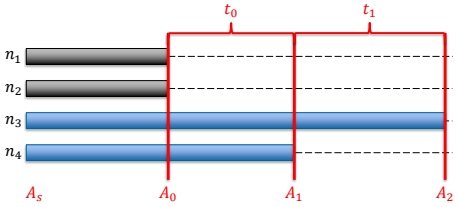

Fig. 2. Example scenario where two of the four edge nodes (in blue) are straggling.

2.2 Non-Stationarity

Data set shift (or concept drift) is defined as the discrepancy between training and test data of a ML model [25]. It can be observed in the joint distribution of inputs or outputs due to non-stationarity of the environment. In many real world applications, assumptions made in learning phase may become invalid over time. For instance, frauds or spammers may change their methods, which invalidates detection algorithms or in a financial forecasting model, external events such as mergers and acquisitions may change the learned dependencies between stock prices. However, each application has its own characteristics and rate of evolution. *Data horizon* is defined as the urgency of including new evidence and updating the model, whereas, *data obsolescence* is the velocity that old data becomes irrelevant to the model [23]. We address these concepts more formally in Section 4. Two well-known solutions for learning data coming from non-stationary processes are to periodically retrain the model and to incrementally update it. We consider the latter in DOML due to high computation overhead of the former. Use of a static model in an application area with short data horizon and rapid data obsolescence is impractical since inference accuracy will decrease over time. Moreover, in the case of DOML, it is also not efficient to update global model with each new data due to high communication cost. Hence, we consider a scenario similar to previous work [3, 17], where local models at each edge node are updated online, whereas, global model is periodically synchronized via a central node. We propose dynamic periodicity of ML model synchronization for DOML, where quorum size and staleness bound are controlled to maximize expected model accuracy. We also define metrics to measure data horizon and obsolescence to analyze in which application areas proposed technique is the most beneficial. Accuracy is used to represent ML model performance in this paper, however, any other metric such as precision, recall or f-measure is viable.

2.3 Motivational Scenario: Transactive Energy Control

Digital transformation of the electricity grid results in so-called smart grids [8, 9], which provide many capabilities such as distributed generation, pervasive control, load management, self-healing, emission control, etc. Also enabled by smart grids, transactive energy control is defined as *a system of economic and*

control mechanisms that allows the dynamic balance of supply and demand across the entire electrical infrastructure using value as a key operational parameter [20]. A NIST report [19] estimates that digitization and modernization of the power grid until 2030, will bring cost benefits that are the three to five times the required investment. In this endeavor, ML has many areas of application such as clustering users and producers into power profiles, detecting theft of electricity, understanding user behaviour, predicting supply and demand, etc. High volume and frequency of data generated by smart meters and sensors, their country-wide dispersion, and cruciality of fast decisions make smart grids challenging for traditional data processing, but an ideal application area for DOML. In conjunction with edge computing, DOML can address scalability issues by alleviating network load meanwhile reducing response time through data processing in high-bandwidth and low-latency proximity. In that sense, DOML is very promising for enabling massive scale smart grids.

Let us consider energy supply and demand forecasting as an example task in this scenario. Accurate and real-time predictions are crucial for optimizing the generation and distribution of electricity, which is considered as a perishable good since it cannot be stored on a wide scale. Some example optimization scenarios include, coping with short-term spikes in demand, dynamically controlling voltage based on geographical demand to reduce losses, or increasing the utilization of generators. ML algorithms such as logistic regression, neural networks (e.g. LSTM), and support vector machines (SVM) can be adapted to forecast time-series data and all have successful applications in the literature [15, 30, 31]. Data originating from generators or consumers in close proximity can be collected in edge nodes as denoted with (i) in Fig. 1, and forecasts can be made locally to give quick responses (ii). Non-stationarity is a particular challenge in this area, which may stem from structural changes such as joining/leaving producers, forming/dissolving links in grid network, and technological advances or quantitative changes such as evolving consumption habits, unexpected events, and seasonality. Trained ML model that is used in forecasting local supply and demand at each edge node, has to be updated frequently with global knowledge from the centralized parameter server in order to avoid staleness and consequent accuracy drop. However, it is not trivial to collect all updates from edge nodes (iii) and decide when local forecast models have to be updated (iv). Both too late and too early updates may cause inaccurate prediction of supply or demand, and consequently inadequate or excessive electricity generation.

3 Consistency of the Fittest

3.1 Problem Definition

We focus on the decision problem of when to synchronize local models at edge nodes in DOML, so that they are informed about global knowledge which is learned collectively by others. Iterations at edge nodes may finish at different times due to heterogeneity of edge resources and volatility of streaming data, leaving us with decision *which (or how many) responses to wait for at each*

iteration, before updating global model and sending it to edge nodes. Too long synchronization period (e.g. waiting until all nodes respond) may cause suboptimal inference performance because edge nodes are obliged to the stale model for a longer time. Too short period, on the other hand, has the disadvantage of losing updates from straggler nodes as well as additional communication cost. Moreover, optimal period differs both over time and between applications due to changes in environment such as streaming rate, selection and capacity of edge nodes, unexpected events and failures, etc. Existing quorum- and bound-based approaches (described in Section 6) overlook such environmental dynamicity.

3.2 Dynamic Periodicity of Synchronization

The main idea behind the proposed technique is to push global model to the edge nodes when either all responses are received or waiting for the future responses is expected to result in lower average inference performance statistically based on previous outcomes. Consider a small-scale example with four nodes in Fig. 2. Here, at the beginning of the time period t_0, edge nodes n_1 and n_2 have already completed their iteration and sent their model updates to the parameter server, however, n_3 and n_4 are late. Stale model already distributed to nodes (M_S) have current accuracy A_s whereas incorporating currently received information results in a model (M_0) with accuracy A_0. First possibility is to push M_0 immediately to distributed nodes so that they will avoid staleness of M_S (assuming $A_s < A_0$). However, this would mean updates from remaining nodes will be ignored for this iteration and they will restart online training from M_0. The second option is to wait until n_4 responds (as it is predicted to be earlier than n_3), update the model with its contribution to M_1 with accuracy A_1, and then push that model. In that case, resulting model can be more accurate ($A_0 < A_1$), but M_S has to be tolerated until the end of time period t_0. Moreover, A_s and A_1 may also decrease over time due to staleness of models. Here, the decision should be made by considering expected magnitude of contribution by n_4 as well as expected length of t_0. A similar trade-off applies for n_3, as well. More formally, we are looking for the future response i among k stragglers such that average accuracy given in objective function in (1) is maximized. Here, τ_i is the waiting time for response i, and ϵ is the time period in consideration for accuracy (e.g. time until next iteration ends). The optimization problem, considering its small size in terms of parameters, can be efficiently solved via linear program solvers.

$$\begin{aligned} \underset{i}{\text{maximize}} \quad & \frac{A_s\tau_i + A_i\,(\epsilon - \tau_i)}{\epsilon} \\ \text{where} \quad & \tau_i = \sum_{j=0}^{i-1} t_j \\ \text{subject to} \quad & i \in \mathbb{Z},\ 0 \le i \le k. \end{aligned} \tag{1}$$

$$\begin{aligned} \underset{i}{\text{maximize}} \quad & \int_0^{\tau_i} A_s(x)\,dx + \int_{\tau_i}^{\epsilon} A_i(x)\,dx \\ \text{where} \quad & \tau_i = \sum_{j=0}^{i-1} t_j \\ \text{subject to} \quad & i \in \mathbb{Z},\ 0 \le i \le k. \end{aligned} \tag{2}$$

For simplicity, A_s and A_i are assumed to be constant in (1), however they are functions of time. A more realistic objective function with this consideration

is given in (2). Here, ϵ in denominator is also replaced since it does not affect the objective being a constant. Multiple accuracy functions and response times need to be calculated or predicted so that aforementioned objective function can be evaluated. We propose efficient mechanisms for these in the rest of this section.

Characteristic function We first learn a characteristic function, $C(x)$, for the accuracy drop of a static model over time. We then utilize the same function for all predicted accuracy values to emulate the impact of staleness on them. In order to obtain $C(x)$, we test the same model at different time steps, log average accuracy value of all edge nodes at each step, and fit a curve to these values. Based on our evaluation with multiple ML tasks on real world streaming data sets, we assume that $C(x)$ follows a sigmoidal function. However, any other curve can be fitted if it better describes data. We propose a four parameter logistic regression that is given by (3). Here, y is the performance value (e.g. accuracy, precision, recall, f-measure, percentage error etc.) of the ML model and x is the time of measurement. Four parameters, a, b, c, and d correspond to lower limit of y, upper limit of y, time of inflection, and the slope of the curve at time c, respectively. We normalize the range of the characteristic function to $[0, 1]$ so it can be used with different models by simply multiplying with initial accuracy as in (4). Characteristic function is specific to the application area as well as ML algorithm used, thus curve fitting should be repeated when one of these changes.

$$y = a + \frac{b-a}{1+\left(\frac{x}{c}\right)^d} \quad (3) \qquad A(x) = C(x)A \quad (4) \qquad R_i = \frac{A_i - A_{i-1}}{A_{i-1}} \quad (5)$$

Initial accuracy Second part of the problem is to predict initial accuracy of the model (A_i) that includes updates from prospective response i along with all preceding. A_i corresponds to the performance of M_i with test data that is collected immediately after its training data. Since M_s and M_0 are already available, their accuracy (A_s and A_0) can be directly computed with the test data collected from all edge nodes. For predicting $A_{i\geq 1}$, however, we resort to time series prediction. Time series data is collected at each response, i, by logging accuracy of current model, A_{i-1}; and accuracy of current model updated with the response, (A_i). We then calculate the magnitude of contribution as improvement rate, R_i as given in (5). Through historical trends of R for each node, time series forecasting algorithms such as autoregressive integrated moving average (ARIMA) [4] can estimate the next value, R_{i+1}. Given estimated R_{i+1} and A_i, it is possible to calculate prospective accuracy value A_{i+1} before response $i+1$ is received by solving (5) for A_i and replacing i with $i+1$.

Response time We model response characteristics of each edge node as a time series where observed response times are the data points. A time series forecasting algorithm can be used to estimate next response time from which elapsed time is subtracted to obtain time-to-response. We employ support vector machine regression algorithm, which demonstrates good accuracy in the similar task of forecasting time-to-failure of edge computing servers [1].

4 Metrics for Non-Stationarity

Consistency of the fittest technique is fairly generic with regard to applicable ML algorithms. It is compatible with any algorithm as long as (i) training is online; (ii) ML model is updatable with submodels; and (iii) its performance is measurable through accuracy or error rate. Majority of online regression, clustering, and classification algorithms meet these criteria and we employ some of the most prevalent ones such as SVM, Bayesian networks, and naive Bayes, in our evaluation (Section 5). However, impact of the proposed technique would be proportional to the non-stationarity of the data stream. Furthermore, edge computing brings new communication challenges to model consistency that greatly increases the significance of measuring the extent of non-stationarity, with respect to centralized or high-bandwidth environments. To the best of our knowledge, there is no other work as of today that handles data set shift in high granularity and proposes metrics for data horizon or obsolescence. Hence in this section, we introduce four metrics for that purpose.

Slope of the Characteristic Function (SCF) is the decrease rate of the ML model performance represented with the slope of the characteristic function at the point of inflection. This corresponds to the absolute value of d in (3) and can be used to evaluate both data horizon and obsolescence.

Time of Inflection (TOI) is the time it takes to observe significant drop in the performance of a static model. More formally, it is the point in the characteristic function such that the second derivative is equal to zero, i.e. x such that $C''(x) = 0$. This corresponds to c in (3) and can also be used to evaluate both concepts.

Contribution of Updates (COU) is the magnitude of contribution observed in the performance when the model is updated with the most recent data. We measure it as the average percentage increase in accuracy (or decrease in error) of the ML model divided by elapsed time since previous update. This metric can be used to measure data horizon.

Depreciation by Stale Data (DSD) is the sensitivity of the ML model to the freshness of data. To measure DSD, we gradually add older data to the training set and observe its accuracy. It is calculated as the average rate of deterioration (or slope) per addition. This metric can be used to measure data obsolescence.

5 Numerical Results

In line with the motivational scenario described in Section 2.3, we train a SVM regression model for the demand forecasting in electricity market of New South Wales, Australia. We utilize `Elec2` data set described in [12]. Our training set consists of 100 half-hourly data points and we forecast five subsequent data points. After initial training, we run the model to predict demands that are increasingly toward the future and calculate accuracy at each step. Accuracy function used in this experiment is %100−Symmetric Mean Absolute Percentage Error (SMAPE), which is defined in (6). SMAPE is chosen for having an upper and lower bound on the values it can get, in contrast to other widely used error

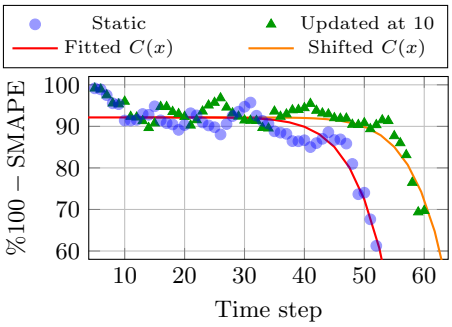

Fig. 3. Regression accuracy in `Elec2` data set as the SVM model stales.

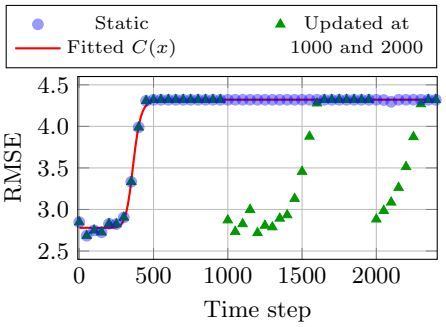

Fig. 4. Prediction error in `SETI@home` data set as the DBN model stales.

metrics such as mean absolute percentage error (MAPE) and mean squared error (MSE). Here, p_t and a_t are the predicted and actual values at time step t.

$$\text{Accuracy} = \%100 - \frac{\%100}{n}\sum_{t=1}^{n}\frac{|p_t - a_t|}{|p_t| + |a_t|} \tag{6}$$

Fig. 3 demonstrates that the accuracy of a static model drops following a sigmoidal characteristic function, $C(x)$, due to staleness. However, updating the model via retraining at time step 10, delays the accuracy drop. Hence, it is possible to maintain an accurate model through repeated updates. This experiment also shows that $C(x)$ is still valid after the model is updated. Note that, we present unnormalized $C(x)$ to facilitate comparison to accuracy values. Only for this experiment, we use model retraining as SVM is not an updatable model.

As an additional scenario, we consider availability prediction of massively distributed client computers for service reliability. We utilize failure traces [14] from the `SETI@home` volunteer computing project to train a Dynamic Bayesian Network (DBN) model for failure dependencies between nodes and predict availability rates through this model. To demonstrate the use of approach with different performance metrics, we use Root Mean Squared Error (RMSE) as in (7), which is one of the most widely used quality measures for estimators. As shown in Fig. 4, a sigmoidal characteristic function also fits to the increasing error rate. Figure demonstrates the impact of two model updates at time steps 1000 and 2000 against a static model. Initial model maintains the same performance for significantly longer time (around 250 time steps or two days) in comparison to the electricity price forecasting scenario (around 30 time steps or 15 minutes). This suggests less stationarity, more general ML model, or both.

We report calculated metrics for the first two scenarios in Table 2a. It also includes five variation of the first scenario with 10 to 50 forecast data points. Results from the second scenario at the bottom row are not directly comparable with others due to the use of a different ML model and they are intended for informative purposes only. COU and DSD are in percentage and TOI unit is

Table 1. Non-stationarity metric values (a) and their intercorrelation (b). TOI unit is number of time steps, whereas COU and DSD are in percentage. SCF is unitless.

DS	ML (#P)	SCF	TOI	COU	DSD
[12]	SVM (10)	6.702	91.682	8.244	22.49
[12]	SVM (15)	7.408	84.037	3.302	23.71
[12]	SVM (20)	8.264	78.034	10.32	24.81
[12]	SVM (40)	8.394	67.497	5.636	23.54
[12]	SVM (50)	10.97	37.941	3.738	25.38
[14]	DBN	14.27	363.83	32.00	2.430

(a)

	SCF	TOI	COU	DSD
TOI	**-0.9879**			
COU	-0.3849	0.4654		
DSD	**0.8571**	-0.7731	-0.1505	
#P	**0.8212**	**-0.9489**	-0.4506	0.6325

(b)

the number of time steps. SCF has no unit by definition of slope. As expected, extending the forecast horizon results in steeper (SCF) and earlier (TOI) accuracy drop. Moreover, models become slightly more sensitive to stale data (DSD), whereas no trend in COU is detected. In Table 2b, on the other hand, the correlation between the pairs of metrics are given. There exists strong (positive and negative) correlation between SCF, TOI, and DSD. SCF and TOI are also strongly correlated with the number of forecast data points (indicated by #P).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(p_t - a_t)^2} \quad (7) \qquad F_1 = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \quad (8)$$

Finally in Fig. 5 and 6, we present average results of 10 repetitions from our DOML simulation. To that end, we randomly split `Elec2` to five sets to represent distributed data streams. We train and update five naive Bayes classifiers, which instead represent ML models at edge nodes. The classification is for the prediction whether the electricity price will go up or down based on demand, supply, time of the day, etc. In Fig. 5, we report F_1 scores given by (8), in the case that there is no synchronization and each edge node maintains its own ML model. In Fig. 6, on the other hand, scores of global models are presented. It is clear that use of a parameter server and a global model not only increases classification accuracy (by 13% on average) but also smooths the fluctuations arising from local non-stationary. However, a static global model stales over time and loses its accuracy. We also provide results from two dynamic models that combine local models from three (random) and five (all) edge nodes, respectively. Updating the model significantly increases accuracy (by 3.3% with 5 nodes) even when some nodes are not considered (by 2.1% with 3 nodes).

6 Related Work

To cope with memory and bandwidth boundedness of traditional stream processing algorithms, several distributed stream processing engines including Apache Storm, Samza, Flink, and Spark Streaming, are proposed. They provide distributed, scalable, and fault-tolerant ways to handle streaming data flow. However, none of these explicitly deal with the problem of data set shift. When

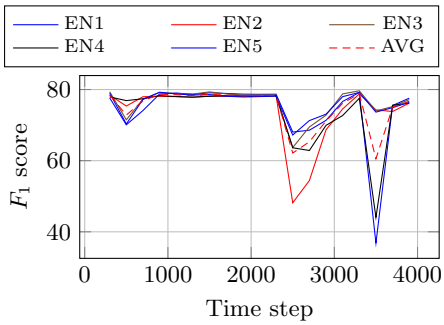

Fig. 5. Classification scores in the case that each edge node is trained separately.

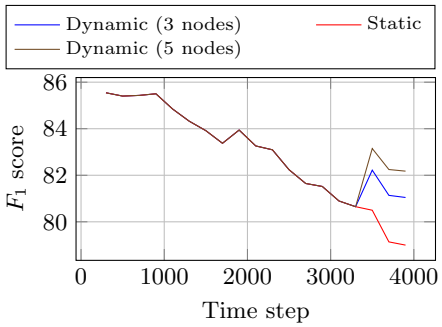

Fig. 6. Classification scores in the case that a global model is maintained.

processing is distributed and occurs at the source (i.e. horizontal parallelism), a centralized parameter server [13, 17] is typically implemented in order to easily combine and synchronize new information learned by distributed nodes as a global model, and to avoid multiple stale models. The unique issue in the geo-distributed case is that arrival times of updates from local nodes may exhibit high variation. State-of-the-Art staleness management techniques can be categorized as quorum and bound based ones. The works in the former category [11, 29], allow to continue synchronization as long as certain number of updates is reached, whereas the latter approaches [7, 13, 16, 28] allow asynchronous execution unless the level of staleness is over the predefined bound. Apache SAMOA (Scalable Advanced Massive Online Analysis) framework is proposed [21] to act as an abstraction for the aforementioned distributed stream processors and it provides rudimentary snapshot-based model consistency. However, none of these techniques are capable of providing the dynamicity in model update times and adaptability to data set shift that are necessitated by high volatility of DOML.

An overview of ML techniques and adaptability mechanisms under data set shift is studied in [32]. The focus is on traditional, centralized ML models, hence consistency issues stemming from distributed learning are not considered. Another work [22] investigates data set drift issues in classification algorithms. They propose the terminology, which we incorporate in this paper, and survey the types and common causes of data set shift as well as methods to detect its occurrence. In the context of edge computing, there exists architectures for DSP that support autonomous stateful migration [5, 6, 24]. However, management of model consistency across multiple nodes is not yet studied to the best of our knowledge. In [18], a data storage management mechanism to cope with limited capacity of edge nodes is proposed. It evaluates the sensitivity of time series forecasting algorithms (but not ML techniques as in this work) to the amount of input data and address the trade-off between storage space and forecast accuracy. Other works on DSP within the edge computing context can be found in a recent comprehensive survey [2].

7 Conclusion

We present a novel technique for efficiently scheduling machine learning model updates from a global parameter server to many distributed edge nodes. Proposed algorithms can be integrated into consistency management modules of DOML tools to outsource implementation challenges. This also applies to data collection from the pervasive edge nodes, which is required by the proposed non-stationarity metrics. Our preliminary evaluation results for both schedulers and metrics are highly promising. In the future, we plan to implement the algorithms as an extension to the prospective Apache SAMOA framework. Another side of the consistency problem left as future work, is how to distribute load to nearby edge nodes so that iterations complete in intended times without too much deviation between nodes. Factors to consider in this regard are resource capacity and transient unavailability of edge nodes as well as streaming rate of data. Depending on the environment, it may be necessary to redistribute load after each iteration based on previous outcomes.

Acknowledgements The work described in this paper has been funded through the Haley project (Holistic Energy Efficient Hybrid Clouds) as part of the TU Vienna Distinguished Young Scientist Award 2011 and Rucon project (Runtime Control in Multi Clouds), FWF Y 904 START-Programm 2015.

References

1. Aral, A., Brandic, I.: Dependency mining for service resilience at the edge. In: Proceedings of the Third ACM/IEEE Symposium on Edge Computing. ACM (Accepted, 2018)
2. de Assuncao, M.D., da Silva Veith, A., Buyya, R.: Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. Journal of Network and Computer Applications 103, 1–17 (2018)
3. Ben-Haim, Y., Tom-Tov, E.: A streaming parallel decision tree algorithm. Journal of Machine Learning Research 11(Feb), 849–872 (2010)
4. Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time series analysis: forecasting and control. John Wiley & Sons (2015)
5. Brogi, A., Mencagli, G., Neri, D., Soldani, J., Torquati, M.: Container-based Support for Autonomic DSP through the Fog. In: Auto-DaSP. pp. 17–28 (2017)
6. Cardellini, V., Presti, F.L., Nardelli, M., Russo, G.R.: Decentralized self-adaptation for elastic Data Stream Processing. Future Generation Computer Systems (2018)
7. Cipar, J., Ho, Q., Kim, J.K., Lee, S., Ganger, G.R., Gibson, G., et al.: Solving the straggler problem with bounded staleness. In: HotOS. vol. 13, pp. 22–22 (2013)
8. Erol-Kantarci, M., Mouftah, H.T.: Energy-efficient information and communication infrastructures in the smart grid: A survey on interactions and open issues. IEEE Communications Surveys & Tutorials 17(1), 179–197 (2015)
9. Farhangi, H.: The path of the smart grid. Power and Energy Magazine 8(1) (2010)
10. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The Cost of a Cloud: Research Problems in DC Networks. Computer Communication Review 39(1), 68–73 (2008)

11. Hara, T., Madria, S.K.: Consistency management among replicas in peer-to-peer mobile ad hoc networks. In: 24th IEEE Symposium on Reliable Distributed Systems. pp. 3–12. IEEE (2005)
12. Harries, M.: SPLICE-2 Comparative Evaluation: Electricity Pricing. Tech. rep., The University of New South Wales, Sydney 2052, Australia (1999)
13. Ho, Q., Cipar, J., Cui, H., Lee, S., Kim, J.K., Gibbons, P.B., et al.: More effective distributed ML via a stale synchronous parallel parameter server. In: Advances in neural information processing systems. pp. 1223–1231 (2013)
14. Javadi, B., Kondo, D., Vincent, J., Anderson, D.: Mining for Statistical Availability Models in Large-Scale Distributed Systems: An Empirical Study of SETI@home. In: IEEE/ACM MASCOTS (2009)
15. Kim, K.j.: Financial time series forecasting using support vector machines. Neurocomputing 55(1-2), 307–319 (2003)
16. Lee, J.H., Sim, J., Kim, H.: BSSync: Processing near memory for machine learning workloads with bounded staleness consistency models. In: International Conference on Parallel Architecture and Compilation. pp. 241–252. IEEE (2015)
17. Li, M., Andersen, D.G., Park, J.W., Smola, A.J., Ahmed, A., Josifovski, V., et al.: Scaling distributed machine learning with the parameter server. In: USENIX Conference on Operating Systems Design and Implementation. pp. 583–598 (2014)
18. Lujic, I., De Maio, V., Brandic, I.: Efficient edge storage management based on near real-time forecasts. In: ICFEC. pp. 21–30. IEEE (2017)
19. McDonald, J., McGranaghan, M., Denton, D., Ellis, A., Imhoff, C., et al.: Strategic R&D opportunities for the smart grid. Tech. rep., NIST Steering Committee for Innovation in Smart Grid Measurement Science and Standards (2013)
20. Melton, R., Knight, M., et al.: GridWise Transactive Energy Framework (version 1). Tech. rep., The GridWise Architecture Council, WA, USA, PNNL-22946 (2015)
21. Morales, G.D.F., Bifet, A.: Samoa: scalable advanced massive online analysis. Journal of Machine Learning Research 16(1), 149–153 (2015)
22. Moreno-Torres, J.G., Raeder, T., Alaiz-Rodríguez, R., et al.: A unifying view on dataset shift in classification. Pattern Recognition 45(1), 521–530 (2012)
23. Parker, C.: Machine learning from streaming data: Two problems, two solutions, two concerns, and two lessons. https://blog.bigml.com/2013/03/12/ (2013)
24. Patel, P., Ali, M.I., Sheth, A.: On Using the Intelligent Edge for IoT Analytics. IEEE Intelligent Systems 32(5), 64–69 (2017)
25. Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N.D.: Dataset shift in machine learning. The MIT Press (2009)
26. Ranjan, R.: Streaming big data processing in datacenter clouds. IEEE Cloud Computing 1(1), 78–83 (2014)
27. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The Case for VM-based Cloudlets in Mobile Computing. IEEE pervasive Computing 8(4) (2009)
28. Xing, E.P., Ho, Q., Dai, W., et al.: Petuum: A new platform for distributed machine learning on big data. IEEE Transactions on Big Data 1(2), 49–67 (2015)
29. Yu, H., Vahdat, A.: Design and evaluation of a conit-based continuous consistency model for replicated services. ACM TOCS 20(3), 239–282 (2002)
30. Zeger, S.L., Qaqish, B.: Markov regression models for time series: a quasi-likelihood approach. Biometrics pp. 1019–1031 (1988)
31. Zhang, G.P.: Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing 50, 159–175 (2003)
32. Žliobaitė, I.: Learning under concept drift: an overview. Tech. rep., Vilnius University (2010), eprint arXiv:1010.4784