Distributed and Centralized Options for Self-Learning

Fatemehsadat Tabatabaeimehr*, Marc Ruiz and Luis Velasco
Optical Communications Group (GCO), Universitat Politècnica de Catalunya, Barcelona, Spain
e-mail: ftabataba@ac.upc.edu

ABSTRACT
In general, the availability of enough real data from real fog computing scenarios to produce accurate Machine Learning (ML) models is rarely ensured since new equipment, techniques, etc., are continuously being deployed in the field. Although an option is to generate data from simulation and lab experiments, such data could not cover the whole features space, which would translate into ML models inaccuracies. In this paper, we propose a self-learning approach to facilitate ML deployment in real scenarios. A dataset for ML training can be initially populated based on the results from simulation and lab experiments and once ML models are generated, ML re-training can be performed after inaccuracies are detected to improve their precision. Illustrative numerical results show the benefits from the proposed self-learning approach for two general use cases of regression and classification.
Keywords: Distributed Computing, Fog computing, Self-learning

1. INTRODUCTION

The revolution brought by technologies like 5G and fog computing [1] requires profound changes, not only in the way applications are conceived but fundamentally, in the way they are managed. Assuming that applications are based on Machine Learning (ML) [2], one of the main problems that arise when it is applied to telecom scenarios is the lack of data required to train typical ML models, such as Artificial Neural Network (ANN) or Support Vector Machine (SVM). In fact, data availability is one of the main obstacles for the generalized deployment of ML-based algorithms. Ideally, one should start from a dataset with real data samples covering properly the considered features space. However, in many cases this is not possible, e.g., in the case of datasets belonging to specific pre-commercial technologies and to forecasted scenarios due to the vast combination of potential cases for ML models application. An alternative approach for ML training is to build a training dataset with data generated from simulation and/or lab experiments. This approach might work provided that one can reproduce in the lab a reasonably large number of patterns to generate the training dataset. Once trained, the ML models can be deployed in the field and used for prediction, classification, etc. However, it is not easy, and depending on the case virtually impossible, to reproduce the large number of patterns that can appear once the ML-based algorithms are deployed in the field; in consequence, the ML models will present inaccuracies, e.g., in the form of prediction or classification errors. Such inaccuracies can be reduced by re-training the ML models with augmented training datasets that include new samples in areas of the features space not yet covered. Hence, the accuracy of the ML models needs to be monitored in the field by comparing the results of applying the ML models against the real measurements from the network. Once patterns for which the ML models do not meet the accuracy requirements are detected, the training dataset can be augmented, and ML re-training can be triggered.

This paper faces the problem of how to deploy highly accurate ML algorithms reducing the need of generating complete training datasets; note that ML models can be re-trained to enhance its accuracy when model inaccuracies are detected thus, following a *self-learning* approach. A learning life-cycle specifically designed for the deployment of ML-based applications in fog computing scenarios is proposed. Starting from the motivation of ML re-training to improve the accuracy of ML models, the general options for ML training within such learning life-cycle are explored. *In-field re-training* procedures (self-learning) are explored afterwards, where starting from initially trained ML models, they are re-trained with augmented datasets that include not yet considered patterns, added as soon as they are detected. Strategies for its practical implementation considering individual learning, where each agent detects new patterns from their local sources and use such them for re-training are reviewed; as ML training is a hard task and requires large computation capabilities, analysis of distributed and centralized options reveals their pros and cons.

2. SELF-LEARNING

To illustrate self-learning, let us consider a use case for fog computing, where devices periodically generate monitoring data with measurements. Let us assume that applications running in fog nodes use the measured data from the devices and compare with the ML-based algorithms. In addition, we assume that a centralized application runs in the cloud for coordination purposes, among other tasks.

In case of ML model inaccuracies (in this use case, an inaccuracy is defined as an incorrectly detected case) are locally detected by the application running in one fog node re-training can be immediately triggered. Because ML training is in general a computationally demanding task, the right place for its execution must be studied, as many

agents run in fog environments where computational resources are scarce. Assuming that the generated knowledge is used for training and updating just the ML model of the detecting device, two re-training alternatives can be implemented: *i*) *distributed training*, where training is executed locally in the fog node; and *ii*) *centralized training*, where training is implemented in the cloud. Fig. 1 illustrates the centralized and distributed learning approaches, where labels help to identify how data flow: the distributed (local) training does not require any data to be conveyed to the cloud at the expenses of requiring extra computational resources in the fog nodes for ML training, whereas in the centralized training data needs to be sent to the cloud where more computational resources are usually available.

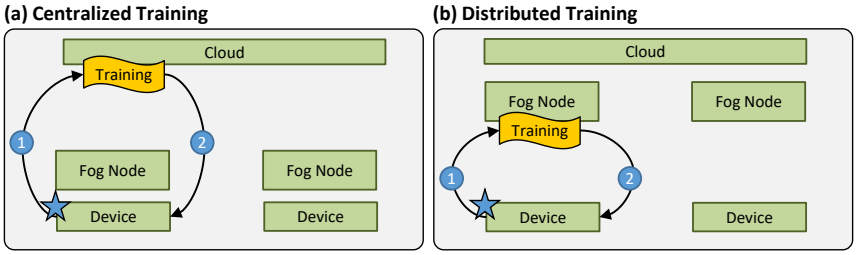

Fig. 1. Self-learning under centralized and distributed training.

For illustrative purposes, Fig. 2 shows how ML models are improved after a prediction or a classification error is detected, and re-training is carried out. Fig. 2a, represents a training dataset, where it can be observed how samples in the training dataset are not uniformly distributed and appear grouped in some areas (or subranges) of the features space, while few samples are available in other areas. This is a representation of a scenario where the training dataset comes from samples obtained by lab experiments and/or simulation, as well as for a scenario where the probability to observe certain patterns is low, whereas for other patterns is higher. Fig. 2a also shows the regression model obtained with the training dataset, as well as some confidence interval. Note that although a prediction can be done with the current regression model, there are some areas in the features space for which no training samples exist, so the prediction error around those areas could be potentially high. In fact, let us imagine that a prediction is needed for a value in one of such areas (represented by the blue point in Fig. 2a). Once the prediction error is detected after observing the real value measured in the network some time later, re-training can be carried out using the original training dataset augmented with the new pattern found. Fig. 2b represents the new regression model with improved accuracy.

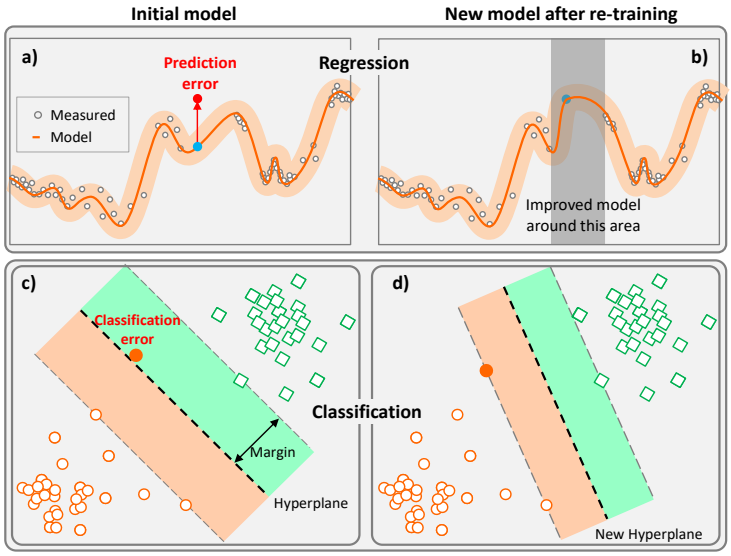

Fig. 2. Re-training regression (a-b) and classification (c-d) models.

Fig. 2c-d presents an example for binary classification using SVMs; for the sake of simplicity, we consider linearly separable classes in a two-dimensional features space. Fig. 2c illustrates that the current hyperplane perfectly separates the two classes of samples identified in the training dataset. After a classification error is identified by comparing the classification obtained by the current SVM classifier and the real data measured in the network, re-training is triggered and a more accurate SVM classifier is obtained (Fig. 2d).

3. PRELIMINARY RESULTS

Let us start analyzing the performance of re-training for a regression use case. For simplicity, let us consider that a response variable $y \in \mathbb{R}$ is to be predicted as a function of a single feature $x \in \mathbb{R}$. A model $f(x)$ is defined in the whole x range so that $|f(x)-y| \leq \varepsilon$; however, no simple correlation (e.g., linear) between x and y can be assumed. In addition, let us assume that during the training phase only some subranges within the whole x range could be observed. Thus, upon the detection of an inaccuracy for a sample x' (i.e., $|f(x')-y| > \varepsilon$), re-training needs to be triggered to improve the model.

For numerical evaluation, a large dataset of pairs $<x, y>$ was synthetically generated so that x and y are highly correlated within given subranges and randomly correlated in other subranges. Next, an initial model obtained with a fraction of subranges was trained (Fig. 3a); for the sake of simplicity, we assume a piece-wise linear model connecting averaged values [4]. One year in-field operation is emulated by randomly selecting samples from the whole data set. Upon the detection of inaccuracies, the model is re-trained with the new data, so that the model is continuously improved until reaching a steady model (Fig. 3b) for which no significant further improvement is achieved.

For the sake of a complete study, different scenarios have been considered, according to the characteristics of the data used for training and the complexity of the relationship between x and y. To this end, let density δ be the proportion of x subranges contained in the initial training dataset, and ρ the measure of correlation defined as the cubic correlation between x and y [3] that can be used as estimator of the relationship complexity between both variables. Fig. 3c illustrates the accumulated number of inaccuracies detected along the operation time for three different scenarios, assuming that new samples arrive every minute. Scenario $<\delta$=90%, ρ=40%> mimics a situation where a realistic behavior can be likely reproduced during training and consequently, a few number of inaccuracies are detected in-operation. In contrast, scenario $<\delta$=10%, ρ=15%> reproduces a more challenging situation, where most of the behavior is learned during operation. Nevertheless, as shown in Fig. 3c, accuracy improves fast in all the cases and the number of inaccuracies drops until reaching the steady model (<< 0.1% of model inaccuracies). In view of the results, we can conclude that re-training cycles allow obtaining accurate models, speeding up ML-based algorithm deployment, whatever the characteristics of the scenario.

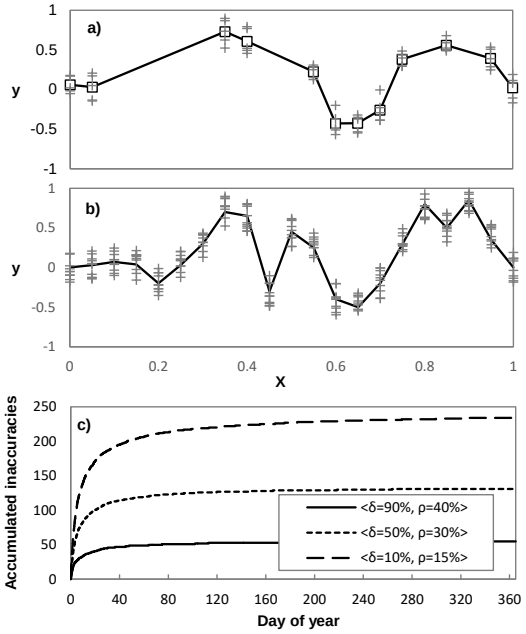

Fig. 3. Initial (a) and steady (b) regression models. Accuracy evolution (c).

Let us now focus on classification using SVMs; in this case, let us consider an example where the categorical variable y with classes c_0 and c_1 is classified as a function of two features $x_1, x_2 \in \mathbb{R}$. Similarly to the regression use case, we generated synthetic data according to two different scenarios: *i*) the *balanced* scenario assumed that the probability of generating both classes is similar, i.e., $P(c_0) \approx P(c_0)$, and that features x_1 and x_2 can be synthetically reproduced with high likelihood; *ii*) the *unbalanced* scenario, where $P(c_0) >> P(c_1)$ and assumes that class c_1 is only partially reproducible for training through simulations and lab experiments. This unbalanced scenario mimics an anomaly detection use case, where c_0 represents the *normal* class and c_1 the anomaly [5]. Examples of initial SVMs classifiers for both scenarios are illustrated in Fig. 4a.

Three different strategies are compared: *i*) *no re-training*, i.e., the SVM classifier is never updated; *ii*) *periodic re-training*, i.e., re-training is triggered periodically (we selected a monthly period), augmenting the training dataset with the data generated by the inaccuracies detected during last month, if any; *iii*) *continuous re-training*,

i.e., re-training is triggered upon the detection of every single inaccuracy. The steady-state SVM classifier after one year of operation and the continuous re-training is illustrated in Fig. 4b. Note how the SVM classifier was strongly modified in the unbalanced scenario after processing real anomaly-like measurements. Fig. 4c plots the evolution of the accumulated inaccuracies for one year. In light of the results, it is clear that re-training is crucial to keep the number of inaccuracies descending and low. In particular, a periodic strategy can be used to reduce the amount of re-training loops while keeping similar performance than the continuous one. Nevertheless, in balanced scenarios, the continuous strategy allows speeding up even more obtaining the steady ML model.

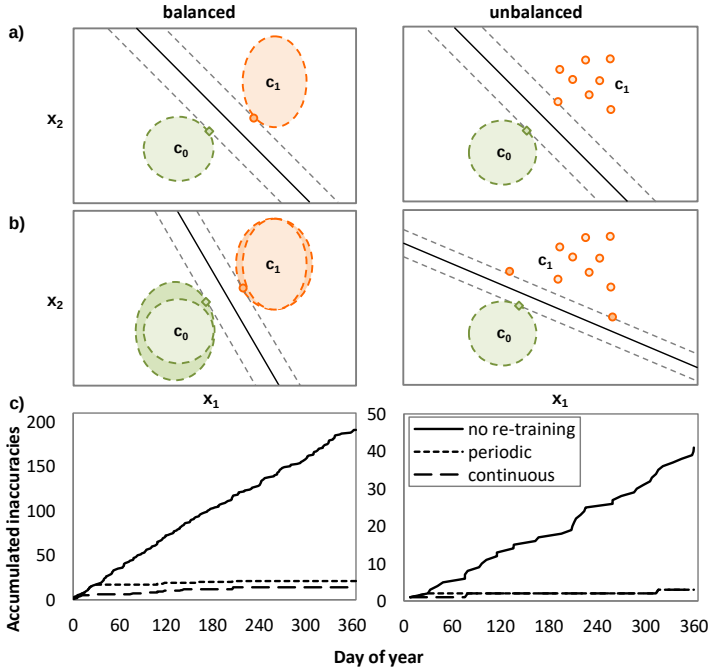

Fig. 4. Initial (a) and the steady (b) training data and SVM classifiers. Accuracy evolution (c).

4. CONCLUSIONS

In this paper an individual self-learning has been proposed looking at deploying highly accurate ML models in and designed to overcome critical obstacles, such as the lack of real measurements to be used for ML training or the extremely large complexity of reproducing realistic heterogeneous fog computing scenarios. The self-learning approach consists of an in-field ML training when inaccuracies are detected; re-training can be performed in a centralized way, i.e., at the cloud, or in a distributed way, i.e., at the fog nodes. Illustrative results for two examples of regression and classification show the potential of self-learning.

ACKNOWLEDGEMENTS

This work was partially supported by the EC through the METRO-HAUL project (G.A. nº 761727), from the AEI/FEDER TWINS project (TEC2017-90097-R), and from the Catalan ICREA Institution.

REFERENCES

[1] R. Vilalta *et al.*, "TelcoFog: A unified flexible fog and cloud computing architecture for 5G networks," IEEE Communications Magazine, vol. 55, pp. 36-43, 2017.

[2] D. Rafique and L. Velasco, "Machine Learning for Optical Network Automation: Overview, Architecture and Applications," (Invited Tutorial) IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 10, pp. D126-D143, 2018.

[3] J. Cohen, *et al.*, "Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences," Third Edition, Routledge, 203.

[4] D. Montgomery, E. Peck, and G. Vining, "Introduction to Linear Regression Analysis," Fourth Edition, John Wiley & Sons, 2012.

[5] X. Chen *et al.*, "On Real-time and Self-taught Anomaly Detection in Optical Networks Using Hybrid Unsupervised/Supervised Learning," in Proc. ECOC, 2018.