Poster: Mobile Gossip Learning for Trajectory Prediction

Mina Aghaei Dinani, Adrian Holzer
University of Neuchatel & HES-SO Valais, Switzerland
name.surname@unine.ch

Hung Nguyen
The University of Adelaide, Australia
hung.nguyen@adelaide.edu.au

Marco Ajmone Marsan
Politecnico di Torino, Italy and Institute Imdea Networks, Spain
ajmone@polito.it

Gianluca Rizzo
HES-SO Valais, Switzerland,
gianluca.rizzo@hevs.ch

***Abstract*—Gossip Learning (GL) is a fully decentralized machine learning paradigm with the potential to enable highly scalability and to preserve user privacy. The majority of existing results however consider scenarios in which either each node communicates with all other nodes, or in which the connectivity graph is static, and they are therefore inapplicable in dynamic setups such as in VANETs. This work is a first attempt at designing and assessing GL schemes suited for scenarios with moving nodes with the application of predicting the trajectory of moving cars.**

I. Introduction

Time series data has become ubiquitous, thanks to affordable edge devices and sensors. Much of this data is valuable for decision making. An example of this is predicting the online trajectory of moving nodes to manage traffic, or proactive resource allocation in vehicular networks. To use this data for forecasting, the conventional centralized approach has shown deficiencies regarding extensive data communication and data privacy issues. Federated learning (FL) allows learning from decentralized data without the need to store it centrally. The data remains where it was generated, which guarantees privacy and reduces communication costs [1]. The majority of existing FL approaches, however, consider scenarios in which either each node communicates with all other nodes, or in which the connectivity graph is static, and they are therefore not applicable in dynamic setups such as in VANETs.

In this work, to tackle these problems, we use personalized distributed FL (that is gossip learning) which is online, peer-to-peer and provides asynchronous communications. Each node in this network is a client for other existing nodes and use its local dataset to improve their models. At the same time, it acts as a coordinating server that merges received models and personalized the model for itself. There can be as many models as the number of clients. Our intended application is a scenario in which a Mobile Network Operator (MNO) receives regularly predictions of car trajectory in order to implement proactive strategies for resource allocation, e.g. for Mobile Edge Computing (MEC) services.

We present three practical algorithms, called DFed Avg, DFed Pow and DFed Best for the serverless federated learning of deep networks based on iterative model averaging, and an empirical evaluation which considers time series datasets and an LSTM model. The experiments demonstrate that these approaches perform well on vehicles which spend a sufficiently large amount of time (min 20 min) in the scenario even with dynamic, unbalanced and non-IID data distributions.

II. System model

We consider a set of wireless nodes, moving on a finite region of the plane according to an arbitrary *stationary* mobility model, and over a finite time window. The given region is partitioned into *cells*, whose shape and size are typically determined by the specific application scenario. Nodes know exactly their position at any point in time, and they communicate among them using a wireless technology (e.g. WiFi, DSLR, Cellular D2D, among others). We say that two nodes are *in contact* when they are able to exchange information directly. From the beginning of the time window, each vehicle samples its position in space at regular time intervals. The resulting time series constitutes the *local dataset* of each vehicle.

III. Gossip Learning (GL) algorithms

At each time interval, each node has to predict its location h time intervals ahead in time. To this end, each node trains a 2-stages LSTM model, an architecture which presents several advantages (such as larger memory, multi-variant input/output, among others) with respect to other RNN models. The output of the model is a vector which associates to each cell a probability value, corresponding to the probability for the node to be in that cell in h time intervals in the future. Finally, a one-hot encoder selects the most likely class as final output. We focus on a collaborative approach, in which each node minimizes a loss function and maximizes accuracy metric for its own model over its local dataset, by leveraging the availability of other node's models to improve its own model. In general the outcome is a different model for each node.

Specifically, the GL algorithms are structured as follows. The time spent by each node in the scenario is divided into two stages. In the *initialization stage*, a node entering the given region collects data points and builds its local dataset, without performing any prediction. At the end of this phase, the node initializes the model by training it on local data. In the *exploitation stage*, at regular intervals (rounds) every node sends its model to nodes in its range. Similarly to traditional server-based federated learning schemes, every node can have two roles (possibly at the same time). A node is a *client* when it receives a model from another node within its range (called *server*), it trains it using its local dataset, and it sends back the model updates to the server. The server node combines the model updates into a meta-model, and it orchestrates the steps of the learning process. Each node is thus a server for his personal learning task, for which it is building its model, while at the same time it is available as a client for the learning tasks of all other nodes in the scenario.

Our main contribution is in the meta-model for GL. We considered three approaches. In the *DFed Avg* approach, at every round the coefficients of the meta-model are obtained by a weighted sum over all clients. For each client, the weight is the ratio between the number of the samples in its local training set over the total samples of all clients in that round. The *DFed Pow* scheme instead aims at weighting each contribution according to a notion of similarity (in terms of roads and regions of the city covered) among the dataset of the server and of the client nodes. Specifically, the server node uses its local validation set to evaluate the loss value l_k of each of the updates of the model received by client nodes. As a loss function, we considered the well known categorical cross-entropy [2]. In such loss function, in order to have each client contribution proportional to the inverse of loss, we assigned to model update from user k a weight given by $\frac{10^{-l_k}}{\sum_{k'} 10^{-l_{k'}}}$ where the summation at the denominator is over all clients which returned a model in the given round.

Finally, the *DFed Best* strategy, the server selects among the model updates received the one with the lowest loss value (computed as in DFed Pow), and it takes it as its meta-model. This scheme originates from the observation that in DFed Pow the meta-model does not always perform better than the single model updates which compose it.

IV. Numerical Evaluation

In order to assess the performance of our GL schemes, we performed a set of simulations over Omnet++ [3] and Keras [4]. We considered the vehicular traces of the Luxembourg SUMO dataset [5], and the time interval 6:30 AM to 7:30 AM over the whole city. In the given scenario, on average about 300 vehicles are present at every time instant, and every vehicle is in the range of about 60 other vehicles, of which on average only half are in the exploitation phase and are thus able to act as clients. We adopted the most recent 30% of data collected by every car as validation set, a 5 s round duration and a 10 s prediction time. The LSTM configuration which yielded the best performance across our experiments consisted in a 50 neurons LSTM with 24 input and 2 output steps, with 5 s interval among consecutive samples, a 10^{-3} learning rate. Fig. 2 and Fig. 1 show accuracy and loss of our GL schemes, averaged over the 30 cars in the scenario with longest sojourn time.

The plots show that mean loss decreases steadily as the GL scheme progresses for the three schemes considered, and that accuracy increases steadily too. In particular, in the given scenario the DfedAvg scheme, based on relative size of local training set, seems to outperform the other two, based on loss estimation. These results suggest that, despite nodes enter the scenario with an empty dataset, a high level of accuracy can be achieved within a relatively small amount of rounds. Thus, on vehicles which spend a sufficiently large amount of time in the scenario, GL schemes can collectively train high quality models over relatively short timespans.

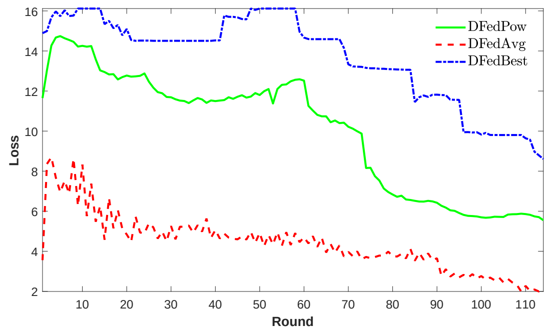

Fig. 1: Mean loss over iterations for our GL algorithm, for the three model merging schemes, in the Luxembourg scenario.

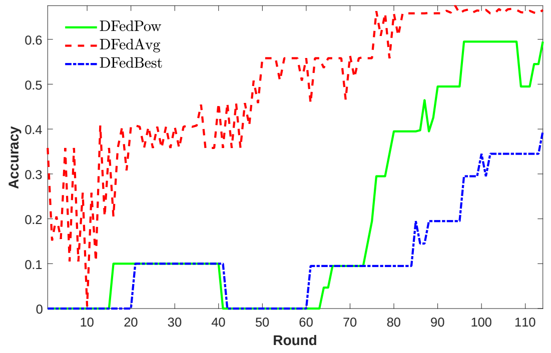

Fig. 2: Mean accuracy over iterations for our GL algorithm, for the three model merging schemes, in the Luxembourg scenario.

V. Future work

These preliminary results suggest that the performance of nodes with too small/poor dataset, or short sojourn times might be improved by having nodes with better models spread them opportunistically, and let other nodes use such received models as starting point in the GL algorithm. We thus plan of expanding this approach with model exchanging among vehicles, which promises to enable vehicles with short sojourn times and small datasets to contribute significantly to the scheme. Moreover, we plan of performing a thorough assessment of our algorithms on a variety of other vehicular scenarios, and to characterize their convergence properties.

References

[1] H. B. e. a. McMahan, "Communication-efficient learning of deep networks from decentralized data," *arXiv: 1602.05629*, Feb 2016.

[2] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[3] A. Varga, *OMNeT++*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 35–59.

[4] F. Chollet, *Deep Learning with Python and Keras: The practical manual from the developer of the Keras library*. MITP-Verlags GmbH & Co., 2018.

[5] L. Codeca, R. Frank, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario," in *IEEE VNC*, Dec 2015, pp. 1–8.