“Federated Averaging” lab guide

Course Assistant Alexander Pavlyuk

April 3, 2023

Motivation

This lab is aimed to help the students to visualize the actual federated learning processes. It is based on the real-world example to demonstrate the relevance of this machine learning (ML) subfield.

It is impossible to perform meteorological measurements at every point in the space. Therefore, the network of weather stations must share meteorological data in order to approximate the desired parameter at the specific point in the space.

Specifically, the lab simulates the network of the Finnish weather stations (see Figure 1). The corresponding data sets and ML algorithms are described in the following section of the guide.

The network consists of the following weather stations:

1. Helsinki Kaisaniemi
2. Espoo Tapiola
3. Hyvinkää Hyvinkäänkylä
4. Mäntsälä Hirvihaara
5. Porvoo Harabacka
6. Loviisa Orrengrund
7. Pyhtää lentokenttä
8. Kouvola Utti Lentoportintie
9. Lahti Sopenkorpi
10. Heinola Asemantaus

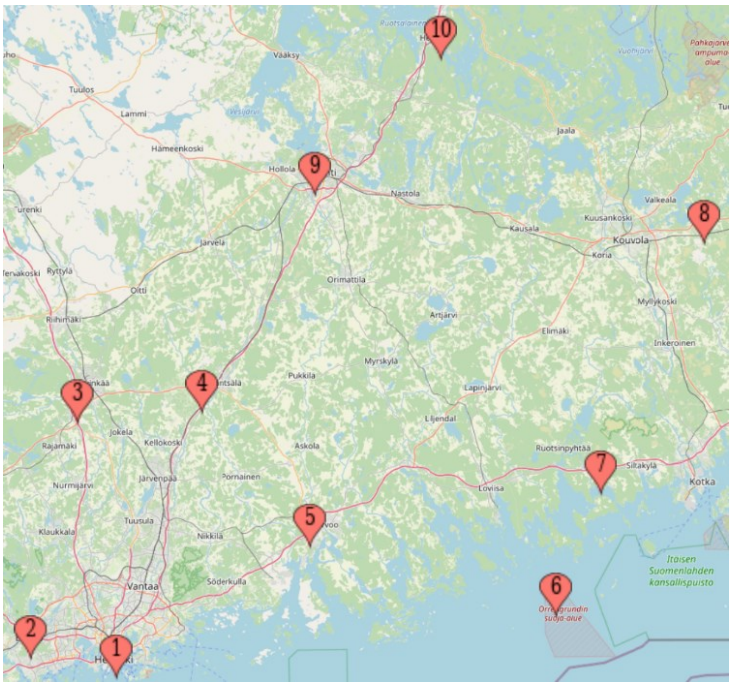

Figure 1. Weather stations network.

How it works?

In abstract, the clients train local models on their data sets and send the obtained weight vectors to the server. When all the weight vectors are received, the federated averaging (see Section 9.3 at the Lecture Notes) algorithm is applied to them. The obtained global weight vector is then sent back to all the clients. The iteration of the described actions is performed before the stopping criteria is satisfied.

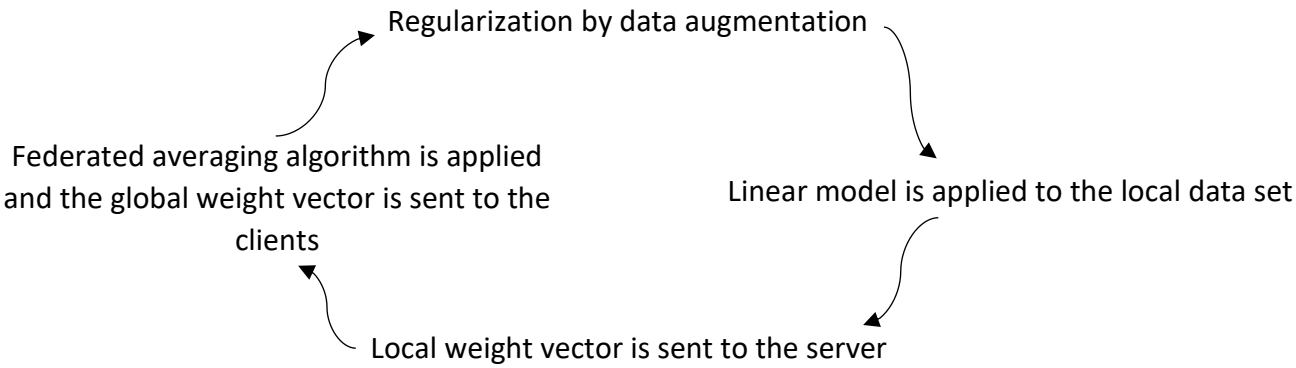

Figure 2. A single iteration of the federated averaging algorithm [1].

In the lab, the clients and the server communicate with each other with the help of the flask library (https://flask.palletsprojects.com/en/2.2.x/). The server code is located on the secure Aalto server – fljung.cs.aalto.fi. The code accepts commands from the students listed in the CSV file on the same server.

Read code comments for more details.

Preparation

Data sets

Enter the webpage of the Finnish Meteorological Institute (FMI). Each lab participant must download the observations with the parameters presented in Figures 3-4. Figure 5 shows the last parameter – observation station. Each lab participant has a different assigned observation station, that can be found in the excel file "Observation_Stations_XX_XX.xlsx" on MyCourses in the "Lab 2 – Materials" folder. The file must be .csv type.

Additionally, download "Porvoo_Kilpilahti_satama_test.csv" data set and put it into the same folder with the training data set described above. This data set will be utilized during the testing phase.

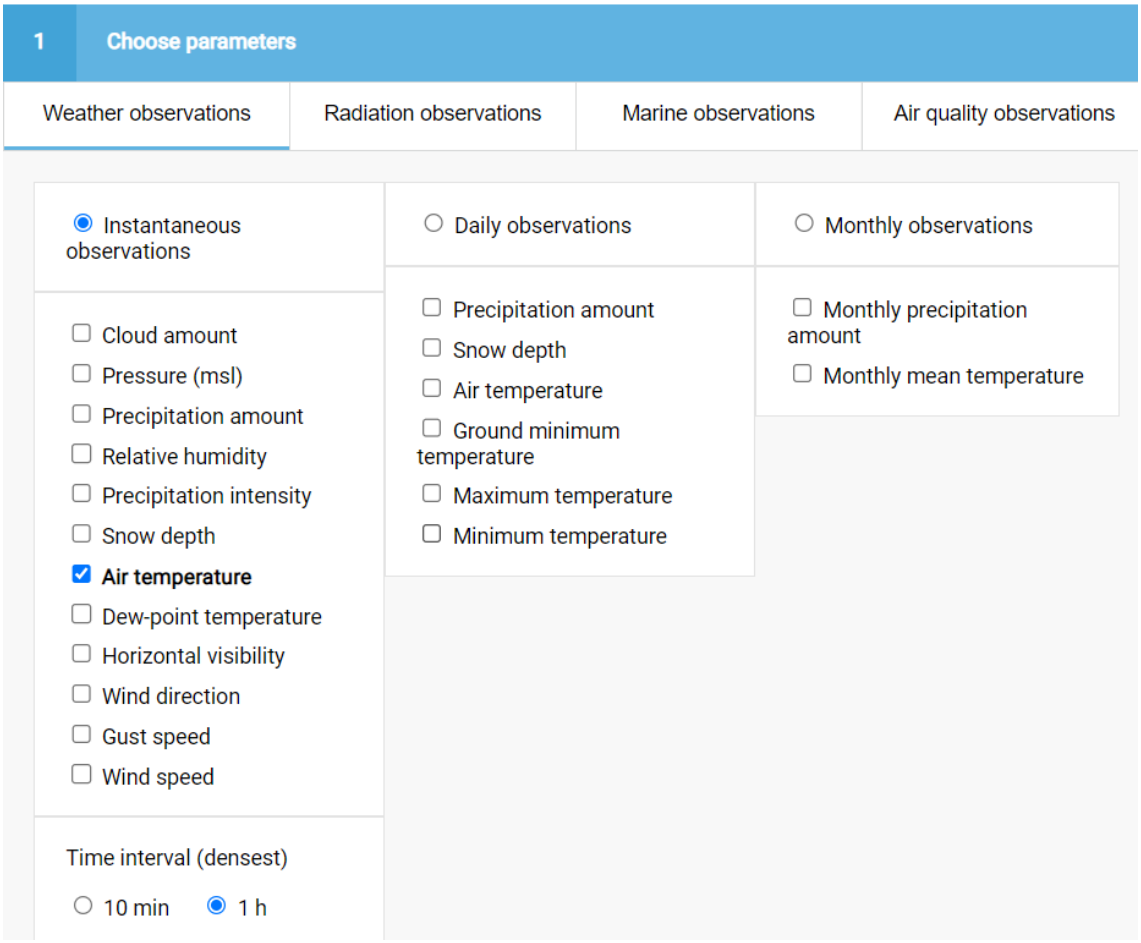

Figure 3. Parameters: instantaneous observations, air temperature, 1 hour time interval [2].

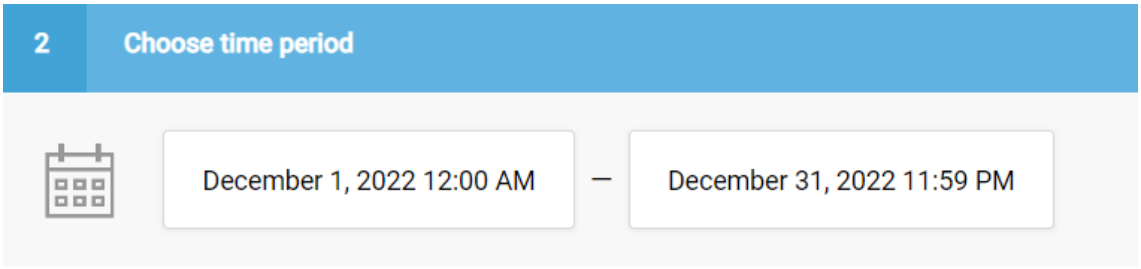

Figure 4. Time period utilized in this guide [2].

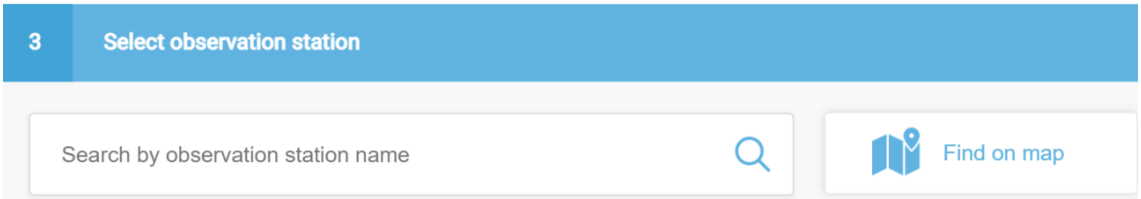

Figure 5. Observation station choice [2].

Code template

The code template called *client_template.py* is in the "Lab 2 – Materials" folder on MyCourses. Download this file and put it to the same directory with the downloaded FMI data set.

Data pre-processing code

The code for data pre-processing is complete, so no modifications are required. It can also be found in the "Lab 2 – Materials" folder on MyCourses. Download this file and put it to the same directory with the code template and the downloaded FMI data set. At this point, everything is ready to start the task.

Task

Data preprocessing

FMI_data.py has already been imported to the *client_template.py* to implement data preprocessing. This step implies the following data modifications:

1. The daytime time interval is defined from 12:00 to 18:00.
2. The daily air temperature inside the daytime interval is averaged.

Finally, each data set contains 31 rows – days. For example, the first five rows of "Espoo Tapiola" data set are presented in the Table 1. Columns "d", "m", and "y" represent the day, the month, and the year respectively.

Average air temperature	d	m	y
-0.442857	1	12	2022
-3.671429	2	12	2022
-3.285714	3	12	2022
-3.042857	4	12	2022
-2.071429	5	12	2022

Table 1. First five rows of "Espoo Tapiola" data set.

Client's code

Open the *client_template.py* and implement the missing parts highlighted with "TODO" comments. There are seven "TODO" comments in total.

Data

To start with, training and testing data sets must be read as pandas.DataFrame type and preprocessed with the imported "daily_avg" function from *FMI_data.py.* Features and labels are constructed with one of the helper functions.

The feature vector consists of five daytime average air temperatures of the consequent days. Therefore, the feature matrix of each data set consists of 26 feature vectors. The shape of the feature matrix is 26 x 5.

The label vector consists of daily daytime average air temperatures started from the sixth day. Therefore, the label vector consists of 26 values.

Next, the initial weight vector and the updated one must be defined with one of the helper functions, which must be also completed according to the "TODO" comments.

Regularization

$$\mathbf{w}^{(i)} := \underset{\mathbf{v}\in\mathbb{R}^d}{\operatorname{argmin}} \left[L_i\left(\mathbf{v}\right) + \lambda \left\| \mathbf{v} - \overline{\mathbf{w}}\left[k\right] \right\|^2 \right]$$

Figure 6. Regularized empirical risk minimization (RERM).

Regularization term in RERM can be emulated by adding synthetic data points to the data set – data augmentation.

d synthetic data points (x^{[d]]}, y^{[d]}) are added to the local data set (d is the number of features). The feature vectors are the unit vectors multiplied by the squared root of regularization parameter lambda:

x^{(1)} = $\sqrt{\lambda}$ (1,0,0,…,0)^{T}

x^{(2)} = $\sqrt{\lambda}$ (0,1,0,….0)^{T}

…

x^{(d)} = $\sqrt{\lambda}$ (0,0,…,0,1)^{T}

and the labels are y^{(r)} = $\sqrt{\lambda} * w'\_\{r\}$ (the r-th entry of the global weight vector w').

The helper function "data_augmentation" is utilized to augment the initial data. The function is complete, and no modifications are required.

Model

Linear model is applied to the local data sets. The corresponding helper function is called "linear_with_augmentation". It combines the linear model and regularization. It must be completed according to the "TODO" comments.

Training

After all the students complete the *client_template.py*, the training process can start. Open the terminal and go to the directory with the lab files with the following command:

cd <path>

Run the code by calling

python3 <file_name>

and enter the student's aalto email address. Check the "Observation_Stations_XX_XX.xlsx" file to ensure your email is the same.

If an error arises, check .csv file with training data set. Some of the measurements might absent. In that case fill the gaps (not many are expected) with the average of the two neighbor values.

Next, terminal instructions propose the student the available commands:

- POST command allows the user to either send the local weight vector to the server and receive the global weight vector by typing FEDAVG command, or to send the testing feature matrix to receive the predicted labels by typing TEST command.
- GET command allows the user to receive the global weight vector after federated averaging algorithm has been applied to all weight vectors received by the server.

The server cannot send the global weight vector back to the user while not all the weight vectors are received. Similarly, the server cannot accept new weight vectors while not all the users receive the global weight vector.

Implement two iterations with POST and GET commands. Basically, a single iteration implies the following sequence of commands from a single user:

1. POST
2. FEDAVG
3. GET

Note: the number of iterations might seem too small. However, it has been chosen because of the possible confusions during the lab. Such design choices which can affect model performance are not considered during the lab.

Testing

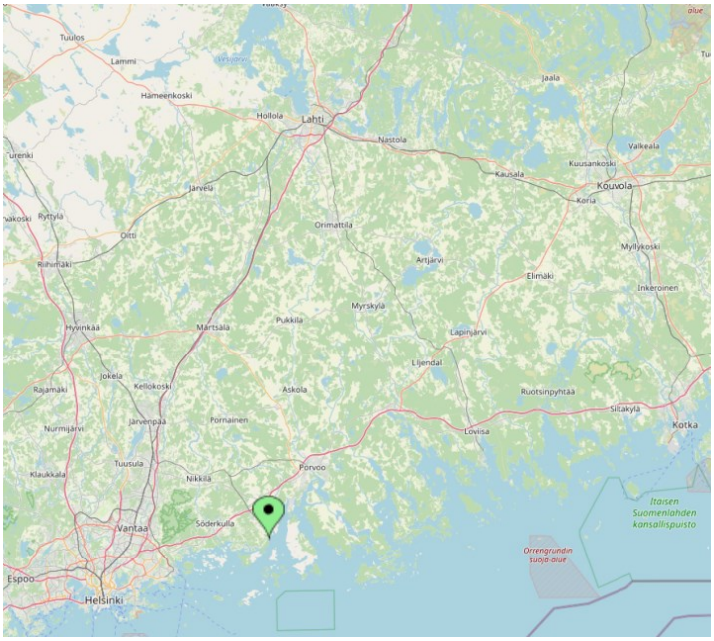

Figure 7. Porvoo Kilpilahti satama observation station.

Porvoo Kilpilahti satama observation station (see Figure 7) is considered as a testing data set. The time period of measurements is the same with the training data sets. This weather station is located inside the network on which the federated learning has been applied.

By calling POST and TEST commands, the server trains the decision tree model on the user-specific data set which is stored on the server. After that the model makes a prediction from the received testing features and send these labels back to the user. Local program calculates the mean squared error (MSE) for the labels predicted by the server and by the local model constructed by federated averaging algorithm. The user can observe MSEs in the terminal.

Implement the testing.

Conclusion

Testing phase reveals that federated averaging over the reasonable nodes of the network shows a great performance. Nevertheless, such improvements as weighted averaging hugely affect the prediction results. However, it is not part of the lab's goal.

References:

[1] Jung, A. *Lecture Notes. CS-E4740 Federated Learning*, *GitHub*. Available at: https://github.com/alexjungaalto/FederatedLearning/blob/main/material/FL_LectureNotes.pdf (Accessed: March 25, 2023).

[2] *Finnish Meteorological Institute*. Available at: https://en.ilmatieteenlaitos.fi/download-observations (Accessed: March 25, 2023).