INFO-H515 - Big Data Scalable Analytics
Project Assignment - Phase II

Yann-Aël Le Borgne, Jacopo De Stefani and Gianluca Bontempi

2017-2018

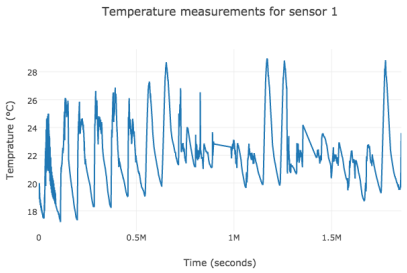
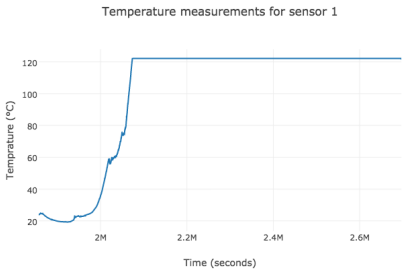
1 Objective

The goal of the assignment is to design a scalable distributed online forecasting system for missing sensor measurements.

1.1 Data overview and problem motivation

The dataset is the same as the one used for Phase I, i.e., temperature, humidity and light measurements, taken at 54 different locations in an indoor environment over a one month period. As for most real-world sensor datasets, the dataset contains many noisy/missing/erroneous values.

For example, the figure below displays the temperature collected by sensor 1, as a function of time (in seconds, from the first measurement on the 28/02/2017, 0am).

Temperature measurements before day 21	Temperature measurements after day 21
	

One can see that:

- The first day (the first 86400 seconds) contains a lot of noisy temperature data
- Day 9 to 10 measurements (around 900k to 1M seconds) are missing
- After day 21 (after 1.85M seconds), the sensor started to ran out of battery and data is erroneous.

Dealing with noisy, missing and erroneous values is one of the hardest challenge in Internet of Things data. An efficient way to handle these issues is to rely on machine learning prediction models.

1.2 Problem statement

In order to simplify the design of your distributed online forecasting system, the project focuses on the daily predictions of sensors 1 and 24, temperature data. The period is restricted to the first 8 days of data (from 28/2/2017 to 7/3/2017), for which most of the temperature data from these 2 sensors are available. This will help to assess the prediction accuracy of the models.

For each day from day 2 to day 8 (1/3/2017 to 7/3/2017), your system should provide daily predictions for all measurements collected by these two sensors, as well as compute the prediction error for the day in terms of MSE (mean squared error). For example, on 1/3/2017, there are 3045 temperature measurements for sensor 1. Your system should be able to build a prediction model that provides 3045 predictions, using only past measurements from sensor 1 (before day 2), or measurements from neighboring sensors (e.g., 2, 33, 35) during day 2 or before.

The overall goal is that, after day 8, your prediction models should be able to provide predictions for sensor measurements even they happen to not send data for a whole day.

Formally, let us denote by $s_i(t)$ the temperature of sensor i at time t, and let $\hat{s}_i(t) = h(x_i, \theta_i)$ be the prediction of $s_i(t)$ using a prediction model h with input data x_i and parameters θ_i.

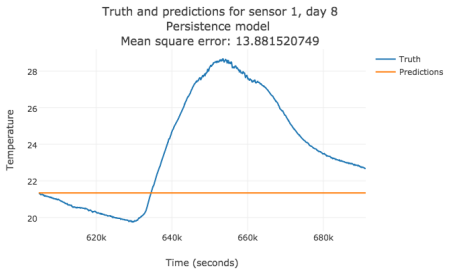
A very basic prediction model is the **persistence** model, where the predicted value of a sensor stream is the last observation, i.e.

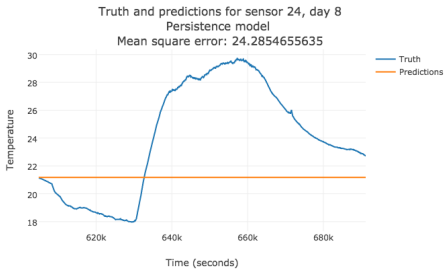
$$\hat{s}_i(t) = \theta_i$$

with $\theta_i = s_i(t_{last_i})$, and t_{last_i} was the last time instant for which a measurement was collected from sensor i.

The figure below illustrates the predictions provided by the persistence model on day 8, for sensors 1 and 24.

Predicted measurements - Day 8 - Sensor 1	Predicted measurements - Day 8 - Sensor 24





The predictions are constant, and qualitatively quite poor. The MSE for sensor 1 is 13.88, and 24.28 for sensor 24. Much better predictions could be obtained using measurements from the past day (there are clear daily patterns, and temporal correlations), or from neighboring sensors (spatial correlations).

Your goal in this assignment will be to implement better predictive models, that take advantage of the existing spatiotemporal correlations between sensor measurements. Your implementation should work in a distributed and streaming fashion (i.e., be scalable, and assume that data is only since once), and rely on Kafka for sending data, and Spark Streaming for building your prediction models and computing predictions.

2 Simulation environment

In order to speed up the design/testing of the prediction system, it is necessary to speed-up the streaming of measurements (testing in real-time the system would take 8 days...). You may adopt an approch similar to the example given in `KafkaSparkStreamingPersistence`:

- Every X seconds, the whole set of measurements for a given day are sent to Kafka
- On the receiver side (Spark streaming), the mini-batch interval is fixed at X seconds, allowing to retrieve a whole day of measurements in one batch.

Using only measurements from one sensor, and the persistence model, a mini-batch of 10 seconds is enough to both send, receive, and process one day of data. In this case, running a simulation for 8 days of data can be done in 80 seconds.

Note that the best choice for the mini-batch interval will depend on the execution time of your data processing system, and will have to be adapted as you increase the complexity of your prediction system.

3 Phase II requirements

Your online forecasting system should provide a scalable solution for the prediction of sensor measurements. Generally:

- The system should be scalable in the number of sensors
- For each sensor, the system should allow to run different prediction models, and to assess their MSE (mean squared error) in an online fashion

For the purpose of the project, your system should specifically:

- Provide predictions for sensor 1 and sensor 24, from 1/3/2017 to 7/3/2017 (predictions are not required for the first day - 28/2/2017 - in order to bootstrap the prediction system). For each of these days, the set of predictions for all measurement collected by sensor 1 and sensor 24 should be returned, together with the resulting mean squared error, for each model. That is, if you have three models (for example persistence, regression on past values, and regression with neighboring sensor measurements), you should be able to retrieve after the simulation the predictions and MSE for the three models, for each day, for both sensors.
- Run at least two prediction models, besides the persistence model, for each of the two sensors. At least one of these two models should be based on the RLS (Recursive Least Square) model. Other approaches include for example exponentially weighted mean, mean of neighboring sensor measurements, or you are free to come up with your own ad-hoc approach.
- Provide graphs, after the simulation, of the predictions and MSE obtained for each model/sensor, from 1/3/2017 to 7/3/2017. The graphs should display the prediction and real values for each model/sensor for the 7 days, and the MSE of the last day (7/3/2017) in the title.

Notes:

- We provide two examples of implementation of Kafka/Spark streaming aimed at helping you get started:

  - KafkaSparkStreamingPersistence: Basic implementation of a persistence model for sensor 1. The notebooks provide a basis for i) streaming daily measurements, ii) compute predictions on day 8, iii) retrieve predictions on real measurements on day 8 and plot a graph showing the predictions/real measurements, and MSE. These notebooks can be a basis for your project. You shoud extend them so predictions and real measurements are collected every day from day 2, for other models than the persistence model, and also for temperature measurements of sensor 24.
  - KafkaSparkStreamingRLS: Basic implementation of two RLS model, on a stream of artificial data (output is a noisy linear combination of ten inputs). The notebooks show an example of running two models in parallel using Spark states. They can be used as an inspiration to run several models in parallel for your project.
- It is particularly difficult to debug code in a distributed and streaming environment. **We therefore strongly advise you to 1) First write down your approach in pseudo-code, 2) Then implement an offline simulation of the pseudo-code, and only then, 3) Adapt the code to the streaming setting.**

4 Deliverables

For phase II, you need to deliver both your implementation of the online distributed prediction system and a report that describes the prediction methods you used, explain how your approach is scalable, and demonstrate the ability of your system to improve the persistence model.

You will get the opportunity to demonstrate your project and defend the implementation choices described in your report during an oral presentation session. This session will be scheduled in the exam session.

The report needs to discuss, at a minimum, the following items:

- The overall architecture of your online distributed prediction system (How are sensor streams processed in parallel, what are the states that you need to keep track of, to what extent is the architecture scalable, . . .)
- A description of each of the model used
- An experimental evaluation that illustrates how the MSE of the different prediction models evolve over time
- Experimental justification that your system is scalable (either by means of a scalability analysis on the cluster, or by means of screenshots from the Spark user interface)
- Problems encountered, limits of your solutions, future work

Implementation environment: You are free to implement your system either on the ULB hosted cluster (which we also use during the lab sessions), install all required software in a locally-hosted environment, or use the Docker container.

Presentation: You will have 15 minutes to present this Phase II project (10 minutes presentation, 5 minutes questions). We require a 10 slides presentation (excluding title, reference), addressing the following points:

- Description of the overall architecture, and why it is scalable (2 slides)
- Description of the prediction models (excluding persistence) (2 slides)
- Experimental results in terms of predictions accuracy (2 slides)
- Experimental results in terms of scalability (2 slides)
- Problems encountered, limits of your solutions, future work (2 slides)

5 Modalities

1. The assignment should be solved in the same groups of 2 as Phase I.
2. The assignment will be graded on (1) the implementation itself and (2) the report that you need to write to describe your analysis and motivate your design and implementation.
3. As for Phase I, you will have to create a GIT repository, in the **INFO-H-515/2017-2018-2** repository group at `http://wit-projects.ulb.ac.be/rhodecode/` to submit both your report and your code (using the same convention `project-<student1>-<student2>` where student corresponds to your student number and `<student1>-<student2>` appear in sorted order). This repository must be made private. It is recommended that you create this repository as soon as possible to avoid last minute technical difficulties, and that you use it throughout the project to synchronize your changes.
4. Your solution for phase II should be pushed to the repository no later than the **10th of June 2018**. You get a penalty of -1 points for each day that your solution is delayed. Only the latest commit will be considered as the solution.
5. Sharing of code or reports between groups is not allowed. (Groups may, however, verbally discuss ideas on how to tackle the project).
6. This project counts for 50% of your grade (10 points). This project is a group (pair of two) homework. **It shall be completed independently and it shall represent the sole efforts of the group submitting the assignment**. The result of another group's efforts, or the copy of another student's efforts (current, or past, semester(s)), is considered academic dishonesty. Also, as stated in Phase I, plagiarism, in the sense of copy-pasting from existing reports or books is a serious issue. To avoid plagiarism, be sure to always quote your sources and indicate clearly if something has been copied verbatim.