

Project Assignment – Phase II

Big Data and Scalable Analytics

**INFO - H - 515**

2018 - 2019

10 June 2019

**Introduction**

The second phase of the assignment for the course on Big Data and Scalable Analytics requires the students to implement a scalable distributed online forecasting system for missing sensor measurements.

In a real-world smart city project, data emitted by sensors can be subject to multiple problems. Indeed, malfunctions of sensors could lead to erroneous, noisy or even missing data. By training machine learning models on incoming streams of data, these problems can drastically be reduced, if not eliminated, through replacement of problematic values by realistic predictions.

Such a system is materialized through implementation of three different models to predict missing (or future) values. The assignment specifies the first model is a simple persistence model. Two other approaches need to improve on this very simple model through the use of machine learning. A precondition is that one of these additional implementations is based on the RLS model. Consequently, the last required model is fully contingent on the students’ choice. In our case, we have opted for a simple, but efficient method in streaming contexts, namely stochastic gradient descent.

**Streaming architecture**

Short description of streaming architecture, how **parallel** operations are handled, **states** that one keeps in mind, etc. to what extent **scalable**.

**Prediction methods**

First, recall some conventions that were defined in the assignment. Let denote the temperature of a sensor at time and let be the prediction of using a prediction model with input data and parameters .

1. Persistence

As described in the assignment, the persistence model predicts the value of sensor solely based on the last observation:

With being and the last instant for which data for sensor was collected. It comes without surprise that the persistence model is not well-suited for the considered problem. Indeed, temperature values are expected to change numerous times during a span of 24 hours. The predictive quality of this model being poor, the goal of the assignment is aimed towards implementing models that are better-suited for predictions corresponding to our smart-city problem.

1. Recursive Least Squares

Recursive Least Squares (RLS)

1. Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an approach that aims to identify the objective function that, in our case, defines temperature distributions through an iterative process. Using our formalism, the predictive model would be described as such:

where is the observed temperature of the slot of sensor (or neighboring sensors) the previous day and is the weight associated to that feature. However, our approach poses the hypothesis that predictions are made for a 24-hour span, thus, the model can be updated for all slots simultaneously.

As mentioned in the assignment, the first day serves to collect data for the initial model with hopes to predict the next day. Evidently, it is impossible to predict very accurate values as no feed-back has been provided by measurements of the next day. For this purpose, it is important to initialize weights adequately. Indeed, in most cases weights can be initialized to zero as these will be subject to training before any prediction. In our approach, a first prediction is to be emitted without preceding feedback. By initializing weights to 1, we obtain a first prediction not too far off the real value. When considering multiple features, the initial weight is set to .

Weights form the core of the learning method. It is precisely the update of these weights that permit to narrow the scope of predictions and steadily approach the objective function. More precisely, when feedback is provided (actual observations) weights are updated as follows:

Where the second term represents the derivative of loss with respect to the selected features. There are many possibilities of choice for a loss function. In our case, loss is the negative difference between observations and the hypothesis nuanced by a small regularization term. By subsequently deriving with respect to the features (temperature and bias) and dividing by the number of observations (slots in a day), the second term can be evaluated and weights updated adequately. Additional research provides support for weight normalization: weight normalization speeds up convergence of stochastic gradient descent [1], resulting in fewer iterations required to obtain realistic predictions.

It is important to know that, given current number of streamed values and architecture, all slot observations are used to update weights. In that perspective, stochastics of the method does not lie in the random selection of instances to update weights, but rather the one-pass training that is exercised on all provided instances.

**Scalability**

Discuss scalability of both the architecture, as well as predictive methods based on the architecture’s characteristics.

The level of scalability of the system is defined by the number of sensors it can accommodate without requiring vast amounts of changes in implementation. As mentioned in the assignment, predictions for sensors 1 and 24 are expected. For this purpose, we have chosen to lighten the load that is sent by the Kafka producer and restrict it to measures of the considered sensors, as well as measures from their ten nearest neighbors.

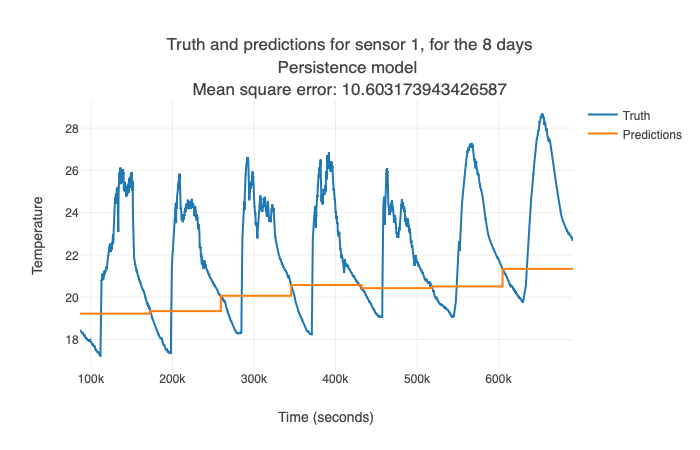
Evidently, a first level of scalability would revolve around the inclusion of all other sensors. To enable this, only few changes have to be applied to the sender’s procedure to avoid restrictions on the information that is being sent. [WHAT CHANGES?]. On the receiver’s side, implementation would remain the same. [MORE MODELS?]

Of course, one could also imagine that additional sensors are added to the project. This would translate in potentially much heavier loads being sent and received. As the current architecture is able to perform relatively fast under speed-up conditions, it safe to assume it could handle a higher load of information in normal speed conditions. However, if growth of sensors is such that the system becomes overloaded, Spark Streaming permits to handle data in parallel and, hence, reduce latencies introduced by these new readings.

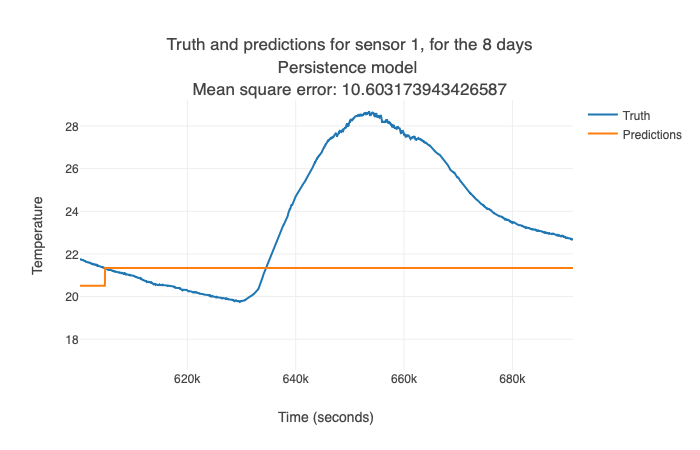
**Sensor Predictions**

The expectation of this assignment is to obtain realistic predictions

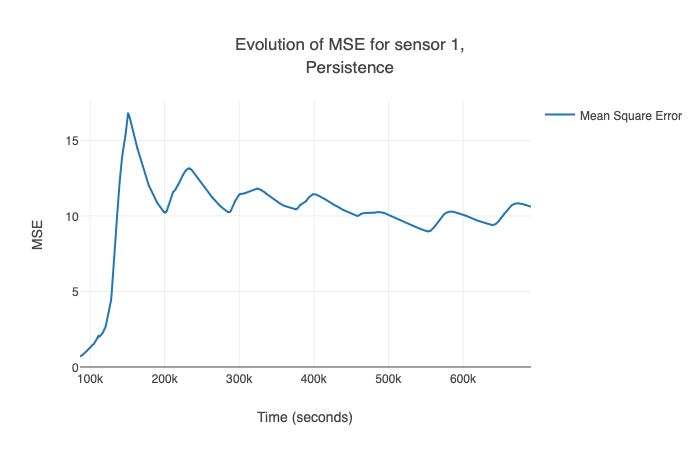
1. Persistence
   1. Sensor 1
      1. Week



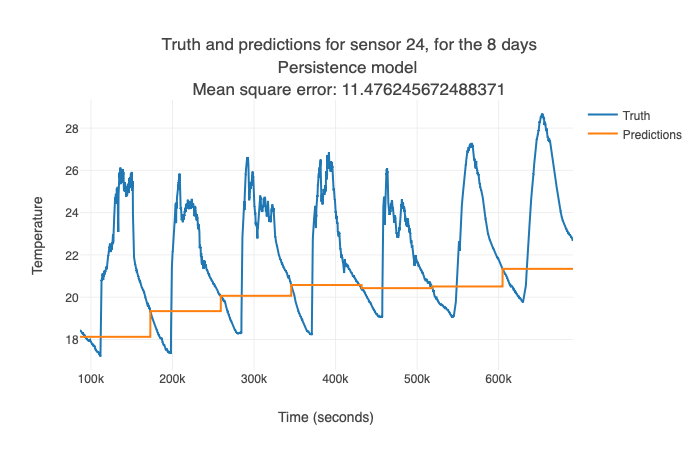
* + 1. Day 8



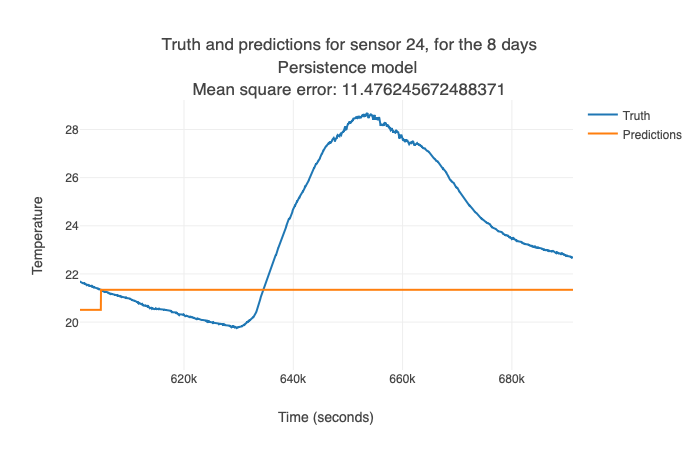
* + 1. MSE



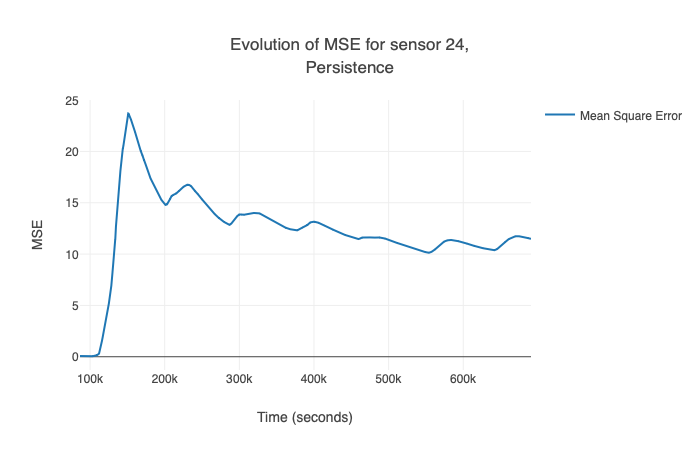
* 1. Sensor 24
     1. Week



* + 1. Day 8



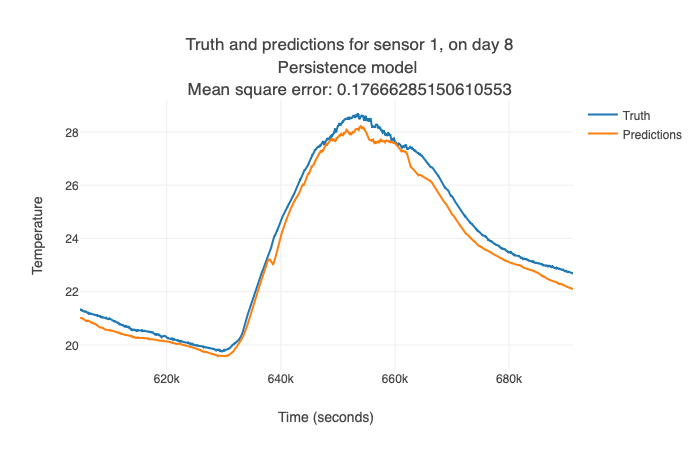
* + 1. MSE



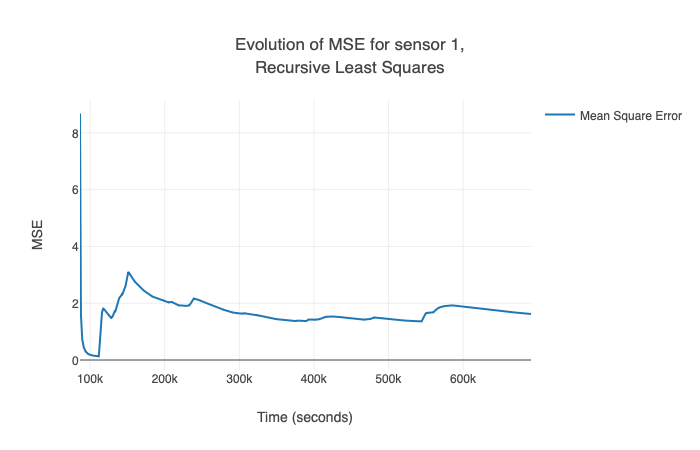
1. Recursive Least Squares
   1. Sensor 1
      1. Week



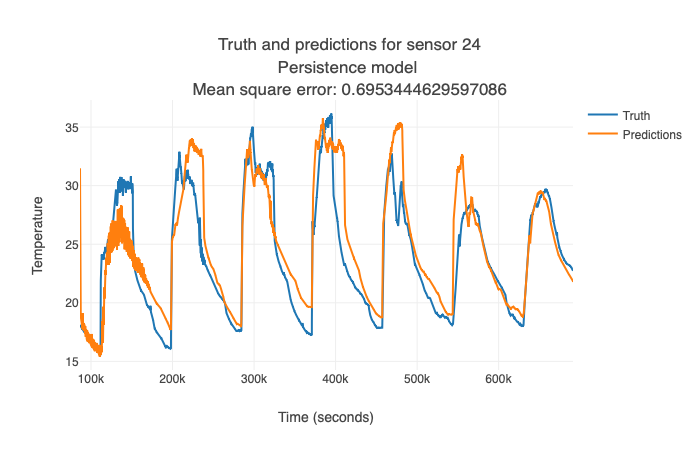
* + 1. Day 8



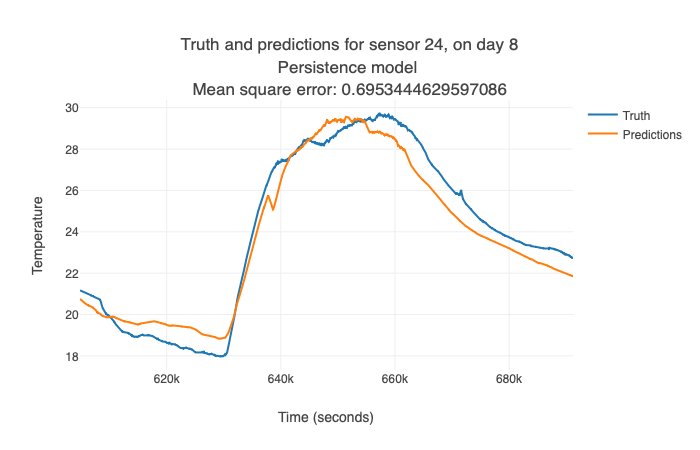
* + 1. MSE



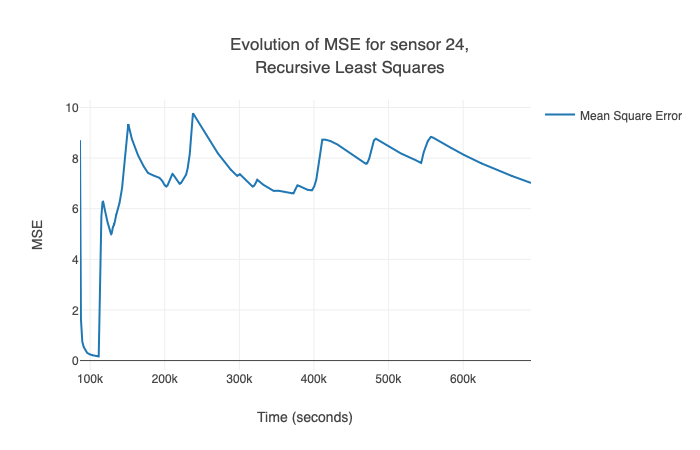
* 1. Sensor 24
     1. Week



* + 1. Day 8



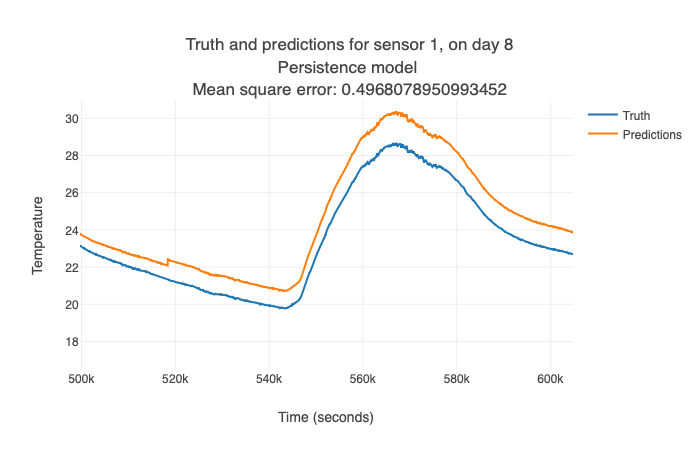
* + 1. MSE



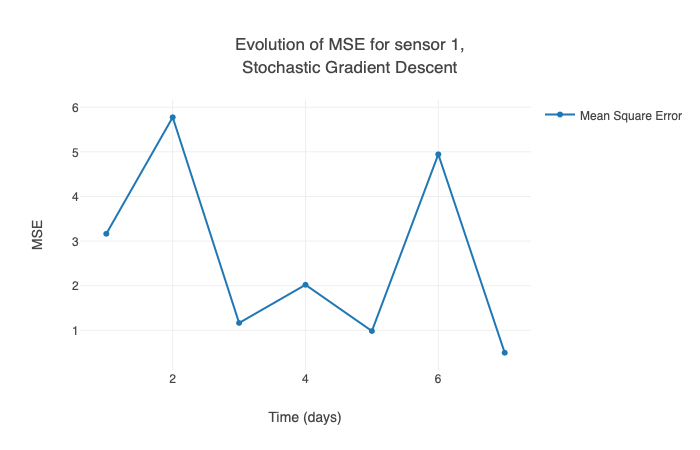
1. Stochastic Gradient Descent
   1. Sensor 1
      1. Week



* + 1. Day 8



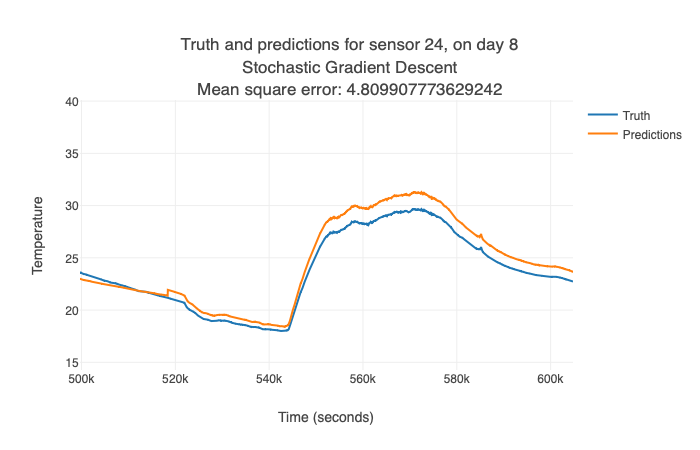
* + 1. MSE



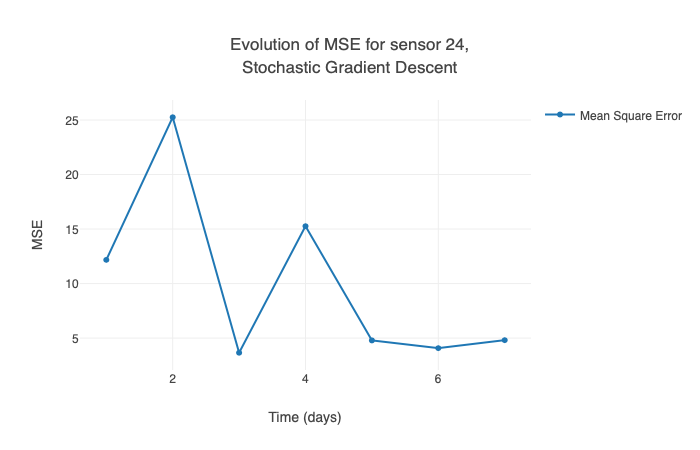
* 1. Sensor 24
     1. Week



* + 1. Day 8



* + 1. MSE



**References**

1. Salimans, T., & Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems* (pp. 901-909).