Machine Learning Heuristics for Short Time Series Forecasting with Quantified-Self Data

Yves Caseau¹

¹ Académie des technologies, Paris, France
yves@caseau.com

Keywords: Machine Learning, Time series, Forecasting, Quantified-self, Generative methods.

1 Introduction

This paper deals with forecasting algorithms for self-tracking in the context of a mobile application. Self-tracking produces short time series that describe the value of variables (“trackers”) that are collected by the user in a relatively short period of time [1]. The main characteristic of “self-tracking” is that the data collected by the user is often biased, quite often short and makes a forecasting challenge which is difficult, when it is not groundless.

The methods proposed in this paper have been developed for the Knomee mobile app (iPhone application available of the App Store), which is based on the “quest” concept. A quest is a collection of trackers which the user expects to make sense as a group. More precisely, one tracker is a target that the user would like to improve or control, and the other trackers are “factors” that may have a causation or correlation relationship with the target.

This paper explores some heuristics to propose robust forecasting for short-time series, that will not be fooled by random time series. Most self-tracking time series have a strong “random noise” component and depend also on other factors that are not tracked or available at the time of the analysis. The challenge is to find a robust heuristic that can extract patterns from data when available, without falling into overfitting.

The paper is organized as follows. Section 2 describes the domain of short-time series forecasting in the context of mobile application self-tracking. Section 3 proposes a toolbox of heuristic algorithms organized into a term algebra. We show how to use random term generation and regularization to produce robust forecasting heuristics. Section 4 presents computational results and shows how these techniques may be used to give some feedback to a mobile self-tracking user. We conclude with the dual observation that short time-series forecasting may be considered as an oxymoron, although some heuristics do provide a moderate but interesting feedback to the user.

2 Short Time-Series Forecast

We have used a set of 20 groups of 4 time series collected by users over two years. The size varies from 40 to 200 time-samples, with values that are both declarative (self-measure using a mobile app) and automatic (using connected devices such as the Apple watch). Because forecasting with such series is very difficult, forecasting plays two roles: it is used to provide for a faster and more playful user interface, and – following the principles of Granger Causality [2] – to use the relative value of forecasting precision as an insight for causality.

The evaluation protocol for machine learning is quite simple: if we have N time-samples, we try to predict the i-th value (i ranges from N/2 to N) using the previous samples (1 to i-1). Our goal is to minimize the normalized difference between the actual value and the predicted one. For most physical phenomenon that are described with short time series, AR(1)MA (AutoRegressive Moving Average) is a method of choice [3]. In the case of our data (biorhythm data with intense fluctuation and stable average), the (integrated) normalization of ARIMA is not necessary. We implemented two other simple classical methods as “gauges”: linear regression using the provided factors as well as time features (time of day and day of the week) and k-means clustering using the same data model.

A quest is a “causal diagram” as defined by Pearl in [4], that is a causality hypothesis. The goal of the application is to evaluate P(target | do(factors)), a notation from [4] that underlines the user’s role. However, human intuition is very easily fooled when self-tracking [2], hence ML training must include a realistic sample of quests that include “patternless” (i.e., mostly random) data. The ability of the forecasting algorithm not to be confused by negative samples is what we defined as robustness, and what makes this problem hard.
3 Randomized Evolutionary Search

We use the following algebra of terms to represent combinations of heuristics:

\[ T :: \text{MovingAverage} | \text{Trend} | \text{Hourly} | \text{Weekly} | \text{Mix(T,T)} | \text{Sum(T,FactorRegression} | \text{CummulativeRegression} | \text{ThresholdRegression} \]

Each term of the algebra represents a heuristic that captures the target time series. The node functions MovingAverage, Trend, Hourly and Weekly are simple heuristics defined by a handful of numeric parameters (i.e., extracting a simple pattern such as moving average or an hourly pattern). Mix() is a linear combination of other terms while Sum(T,_) combines a regular term and a “correction” that is extracted from one of the 3 available factors, using linear regression optionally coupled with integration and threshold mechanisms.

Our toolbox is a combination of three components: (1) a set of heuristics that creates a “lift term” for each the algebra constructor (lift(C,t) produces a term C( _) from a time series t); (2) a generative algorithm that randomly construct terms, that are combinations of these various heuristics, similar to the approach described in [5]; (3) a local optimization algorithm that takes a term and optimizes each numeric parameter within the scope of the term so that the distance to \( t \) is minimized.

The RandOpt(\( n,m \)) algorithm is then simply expressed as randomly producing \( n \) algebra terms with a bounded complexity, applying \( m \) loops of local optimization and selecting the terms with the shortest distance to the original time series. The distance is the sum of a simple Euclidean distance and a regularization penalty to reduce standard deviation [6].

4 Computational Results

Randomized optimization, with \( n=100 \) and \( m=3 \), produces heuristic terms that deliver a performance of 17.4% where the control value is 18% (what we get if we use the average as a predictor). This is to be compared with the individual contributions of the unit heuristics from the algebra that are each over 18% or with the control algorithms (regression, k-mean and ARMA) whose performance are between 19% and 21%.

I will give more details regarding more sophisticated methods in the full paper. There are many ways to use classical techniques to produce better fit: using stronger local optimization (two-opt versus one-opt), genetic algorithms, etc. It is also possible to look for more complex terms and to use more optimization cycles. However, none of the “better methods” have shown to deliver better robustness. The method presented here is the only one, among a large set of approaches tested over two years, that delivers a stable and robust improvement over using the average value for forecasting. “Stable” here means that our preliminary findings with the first-year data (10 data sets) were confirmed with the next year data.

5 Conclusion

Machine learning with small data sets is hard, and robust forecasting from short time series is especially hard. One could think that 0.6% improvement from 18% is not significant. However, this performance is the average of those data sets where something may be found and those where nothing of predictive value is present. Besides, recall that in the best case, only a fraction of the target data can be “explained” or “predicted” from the associated factors.

There are two interesting findings: none of the more sophisticated approaches that we tried were robust enough to beat this simple “RandOpt” approach, while the performance of this simple approach appears to be quite stable even if the results are not impressive.

References


