The Need for Benchmarks with Data from Stochastic Processes and Meta-Models in Evolving Systems

Katharina Tschumitschew
Department of Computer Science
Ostfalia University of Applied Sciences
Salzdahlumer Str. 45/48
D-38302 Wolfenbuettel, Germany
Email: katharina.tschumitschew@ostfalia.de

Frank Klawonn
Department of Computer Science
Ostfalia University of Applied Sciences
Salzdahlumer Str. 45/48
D-38302 Wolfenbuettel, Germany
Email: f.klawonn@ostfalia.de
and
Bioinformatics and Statistics
Helmholtz Centre for Infection Research
Inhoffenstrasse 7
D-38124 Braunschweig, Germany
Email: frank.klawonn@helmholtz-hzi.de

***Abstract*—The problem with real data is often that noise and forms of randomness are involved. This causes a specific problem for evolving systems. They must be able to distinguish between changes according to noise and changes of the underlying data generating process or its parameters. In the worst case, an evolving system might just try to track the noise in the data and is unable to learn the actual relationship inherent in the data. But how can we make sure that this will not be the case for a given evolving system? We propose some simple benchmark tests in our paper that can give an idea of how much an evolving system might be misled by noise.**

I. Introduction

Evolving systems are designed to cope with streaming data. There are two important aspects that are usually considered for evolving systems. It is required that the evolving system can react in real time so that efficient computations – usually recursive ones – are needed without analyzing the whole data that have been collected so far again. The second aspect is the change of the underlying process that generates the data, so that the evolving system must adapt itself on-line.

The problem with real data is often that noise and forms of randomness are involved. This causes a specific problem for evolving systems. They must be able to distinguish between changes according to noise and changes of the underlying data generating process or its parameters.

Statisticians categorize models that do not make explicit assumptions about the data generating process as exploratory data analysis techniques. Such methods, like decision or regression trees, neural networks or support vector machines are very successful in classification and regression tasks, but there is a need to evaluate the performance of such models, since they cannot be analyzed in the classical statistical sense where the underlying assumptions about the data generating process must be made explicit. A very common way to evaluate the performance of such models are methods that divide the data into training and test data like, for instance 10-fold cross-validation. The training data are used to construct the model and the performance is evaluated based on the test data.

For evolving systems, the partition into training and test data is more complicated. We cannot simply take out some data for testing, since the data generating process is assumed to change and the evolving system is intended to track these changes. What could be done is to take out single data records of a time series and see how the evolving system performs with respect to this records. However, to be representative, we would need a much larger test data set due to the fact that the data generating process changes and the test data must reflect situations with changes and without changes accordingly. Even then it is not clear how to judge the performance of an evolving system. A large error can be due to overfitting – i.e. erroneously tracking the noise in the data – or because there is so much noise in the data that the performance cannot be better. A rough indication of how much an evolving system tries to track the noise, is the difference between the error on the training and the test data. However, if an evolving system has already a bad performance on the training data, the performance on the test data cannot be much worse and one might assume that the error comes from the noise in the data.

In this paper, we propose to set up simple theoretical models for the data generating process for which we can give at least a rough answer to the question, how well a model that takes assumptions on the data generating process into account could perform. We can then compare how much worse the evolving system performs. Of course, we cannot expect the evolving system to have a similar performance, since it is not allowed to make specific assumptions about the data generating process. But this comparison will give us an idea, how much an evolving system can deviate from an optimal model and whether a simpler model that would not track the noise might not have a better performance.

We consider simple prediction tasks like classification and regression in this paper. Evolving system have been proposed for such problems for instance in [1], [2], [4], [5], [6].

In section II we carry out a theoretical analysis between an extremely simplified evolving system based on a windowing technique and a model tailored to the known assumptions of the data generating process. This can be considered as a regression problem with the identity function as the regression function.

Section III introduces a simple switching model which can be interpreted as a regression or a classification problem. Here we carry out an experimental comparison between a maximum likelihood estimator exploiting the assumptions on the underlying data generating process and an evolving system without specific assumptions on the data generating process.

We conclude our paper with remarks on more complicated prediction tasks.

II. Random walk

A very simple example for systems with a random change over time is one dimensional random walks [7]. A random

walk $(X_t)_{t\in\mathbb{N}}$ is obtained by adding up values from independent and identically-distributed (i.i.d.) random variables Y_i with expected value zero, i.e. $E(Y_i) = 0$ and variance $\sigma^2 = Var(Y_i) = Var(Y)$.

$$X_t = \sum_{i=1}^{t} Y_i \tag{1}$$

The expected value for the random walk is

$$E\left(X_t\right) = E\left(\sum_{i=1}^{t} Y_i\right) = \sum_{i=1}^{t} E(Y_i) = 0 \tag{2}$$

independent of t, whereas the variance of random walk increases linearly with t.

$$Var\left(X_t\right) = Var\left(\sum_{i=1}^{t} Y_i\right) = t \cdot Var\left(Y\right). \tag{3}$$

The covariance of a random walk is given by

$$Cov\left(X_t, X_s\right) = s \cdot Var\left(Y\right), \ (s < t) \tag{4}$$

and tends to infinity as well with increasing difference between the time points t and s.

The best prediction that one can do for a random walk is the naïve approach, simply using the last value is a prediction for the next value. $\hat{x}_{t+1} = x_t$. Therefore, the difference between the true value and the predicted value is $X_{t+1} - X_t = Y_{t+1}$. Thus, the expected quadratic error is

$$E\left((X_{t+1} - X_t)^2\right) = E\left((Y_{t+1})^2\right) = Var(Y). \tag{5}$$

Now assume, we do not know that we deal with a random walk and try an extremely simple "evolving system" using a windowing technique with a window size of T, using the mean of the last T values as a prediction for the next value. The expected quadratic error for the prediction can then be computed as follows, where $t_0 = t - T + 1$ is the start of the window:

$$E\left(\left(X_{t+1} - \tfrac{1}{T}\textstyle\sum_{i=t_0}^{t} X_i\right)^2\right) =$$

$$E\left(X_{t+1}^2 - \tfrac{2}{T}\textstyle\sum_{i=t_0}^{t} X_i \cdot X_{t+1} + \tfrac{1}{T^2}\left(\sum_{i=t_0}^{t} X_i\right)^2\right) =$$

$$E\left(X_{t+1}^2\right) - \tfrac{2}{T}\textstyle\sum_{i=t_0}^{t} E\left(X_i \cdot X_{t+1}\right) + \tfrac{1}{T^2} E\left(\left(\sum_{i=t_0}^{t} X_i\right)^2\right) =$$

$$Var\left(X_{t+1}\right) - \tfrac{2}{T}\textstyle\sum_{i=t_0}^{t} Cov\left(X_i, X_{t+1}\right) +$$

$$\tfrac{1}{T^2} E\left(\textstyle\sum_{i=t_0}^{t} X_i^2 + 2\sum_{i=t_0+1}^{t}\sum_{j=t_0}^{i-1} X_i \cdot X_j\right) =$$

$$Var\left(X_{t+1}\right) - \tfrac{2}{T}\textstyle\sum_{i=t_0}^{t} Cov\left(X_i, X_{t+1}\right) +$$

$$\tfrac{1}{T^2}\left(\textstyle\sum_{i=t_0}^{t} Var(X_i) + 2\sum_{i=t_0+1}^{t}\sum_{j=t_0}^{i-1} Cov(X_i, X_j)\right) =$$

$$(t+1) \cdot Var\left(Y\right) - \tfrac{2}{T}\textstyle\sum_{i=t_0}^{t} i \cdot Var\left(Y\right) +$$

$$\tfrac{1}{T^2}\left(\textstyle\sum_{i=t_0}^{t} i \cdot Var(Y) + 2\sum_{i=t_0+1}^{t}\sum_{j=t_0}^{i-1} j \cdot Var(Y)\right) =$$

$$Var\left(Y\right)\left(t + 1 - \tfrac{2}{T}\textstyle\sum_{i=t_0}^{t} i + \tfrac{1}{T^2}\sum_{i=t_0}^{t} i + 2\sum_{i=t_0+1}^{t}\sum_{j=t_0}^{i-1} j\right) =$$

$$\frac{(2t - 2t_0 + 3)\,(t - t_0 + 2)}{6\,(t - t_0 + 1)} Var(Y) \tag{6}$$

It is easy to show that (6) has its minimum at $t_0 = t$, i.e. for a window of size one, where we simply use the last value as a prediction for the next value. The worst case is to use all values, i.e. $t_0 = 0$. In this case, the expected quadratic error tends to infinity with increasing t. Figure 1 shows(6) as a function in t for $t_0 = 0$ and $Var(Y) = 1$. In this case, t can be interpreted as the window size. The expected quadratic error increases with $O(T)$ in terms of the window size T.

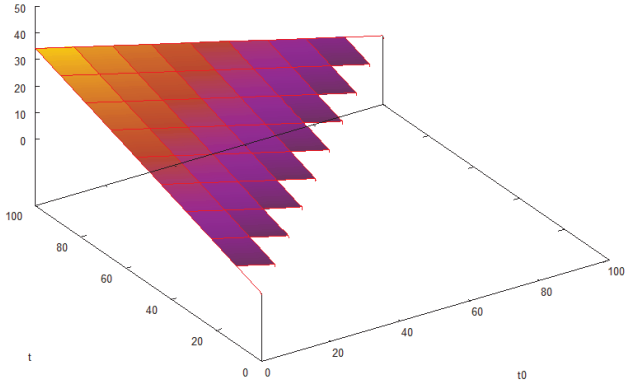

Fig. 1. The expected quadratic error for a random walk prediction depending on the window size.

One might criticize that the mean over a window is a much too simple prediction. But actually, any more complicated function will tend to make the expected quadratic error even worse.

The next example is a random walk to which we add noise in the form of a random variable Z with $E(Z) = 0$.

$$X_t = \sum_{i=1}^{t} Y_i + Z_t \tag{7}$$

where the Y_i are i.i.d. as Y and the Z_t are i.i.d. as Z. We assume also that the random variables Y_i and Z_j are all independent. Since $E\left(Y\right) = 0$ and $E\left(Z\right) = 0$ holds, we have again $E\left(X_t\right) = 0$.

As in the previous example of the random walk without noise, the best prediction for the next value we can choose is the previous value: $\hat{x}_{t+1} = x_t$. The difference between the true and the predicted value is then $X_{t+1} - X_t = Y_{t+1} + Z_{t+1} - Z_t$.

The variance and covariance of a random walk with noise are

$$Var\left(X_t\right) = t \cdot Var\left(Y\right) + Var\left(Z\right) \tag{8}$$

and

$$Cov\left(X_t, X_s\right) = s \cdot Var\left(Y\right), \ (s < t), \tag{9}$$

respectively.

The expected quadratic error can be computed as follows:

$$\begin{aligned} E\left((X_{t+1}-X_t)^2\right) &= E\left((Y_{t+1}+Z_{t+1}-Z_t)^2\right) \\ &= Var\left(Y\right)+2Var\left(Z\right) \end{aligned} \quad (10)$$

Based on (8) and (9) we can calculate the expected quadratic error for the prediction based on a windowing technique.

$$E\left(\left(X_{t+1}-\tfrac{1}{T}\sum_{i=t_0}^{t}X_i\right)^2\right)=$$
$$E\left(X_{t+1}^2-\tfrac{2}{T}\sum_{i=t_0}^{t}X_i\cdot X_{t+1}+\tfrac{1}{T^2}\left(\sum_{i=t_0}^{t}X_i\right)^2\right)=$$
$$E\left(X_{t+1}^2\right)-\tfrac{2}{T}\sum_{i=t_0}^{t}E\left(X_i\cdot X_{t+1}\right)+\tfrac{1}{T^2}E\left(\left(\sum_{i=t_0}^{t}X_i\right)^2\right)=$$

$$\frac{(2t-2t_0+3)\,(t-t_0+2)}{6\,(t-t_0+1)}Var(Y)+2\,(t-t_0+1)\,Var\left(Z\right) \quad (11)$$

As before, this function has its minimum at $t_0 = t$ (window size $T = 1$) and is largest for the highest window size. The expected quadratic error is also increasing in $O(T)$.

III. Switch model

The two random walk examples described in the previous section can be understood as data generating models that change continuously, representing very simple examples for shift. In this section, we discuss an example for a switching model with sudden jumps in the change of the data generating process. To keep the model as simple as possible, we consider a data generating process that switches between two normal distributions with the same variance σ^2, but one with expected value $\mu_1 = 0$ and the other with expected value $\mu_2 = 1$. The probability to switch from one normal distribution to the other is p in each step and the probability to stay with the same normal distribution is $(1-p)$. The random switching between the two normal distributions is carried out independently in each time step. This model for the data generating process is illustrated as an automaton with two states in figure 2.

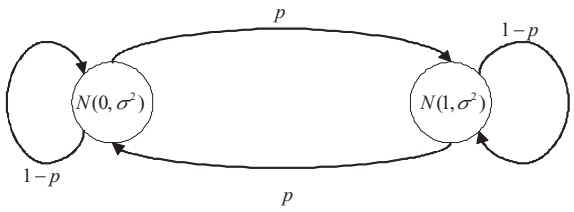

Fig. 2. A switching model.

Depending on the switching probability p, we can distinguish three cases.

- $p \approx \frac{1}{2}$ The data are generated randomly from any of the two normal distributions in each step of the process. It is impossible to make a prediction from which normal distribution the next value will be drawn based on previous data.
- $p \gg \frac{1}{2}$ There is tendency to permanently switch from one normal distribution to the other in each step.
- $p \ll \frac{1}{2}$ The data generating process tends to stay with the same normal distribution in each step and switches only once in a while.

The first case is does not really represent a switching model. It can be understood as drawing independent random samples from a mixture of the two normal distributions. The second case is less interesting from the practical point of view. We normally assume that normally the model will not stay stable and changes happen only once in a while. Therefore, we only consider the last case. There it is interesting to consider the relation between p and σ^2. Small values for p, i.e. changing from one normal distribution to the other rarely happens, and small values for σ^2, i.e. we distinguish between values from the one or the other normal distribution with a higher probability, make the predictions much easier.

Assuming that we know that our data generating process is as described above, we can try to predict the next value based on a maximum likelihood estimation. We can compute the two likelihoods that the previous value was generated by the normal distribution with expected value $\mu_1 = 0$ and that it comes from the normal distribution with expected value $\mu_2 = 1$. We can then use as a simplified prediction, the mean value of the normal distribution with the higher likelihood.

Given a sequence of values $x_1, x_2, \ldots, x_m$ and a sequence of states $\mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_m}$ where $\mu_{i_j} \in \{0, 1\}$ in the automaton in figure 2, the likelihood for this sequence of states or paths is

$$\prod_{i=j}^{m} f(x_j|\mu_{i_j}) \cdot q_{j-1} \quad (12)$$

where $f(x|\mu)$ is the probability density function of the normal distribution with expected value μ and variance σ^2 and

$$q_{j-1} = \begin{cases} (1-p) & \text{if } \mu_{i_j} = \mu_{i_{j-1}}, \\ p & \text{otherwise.} \end{cases} \quad (13)$$

q_0 is the probability for the initial state μ_{i_1}.

The likelihood for a sequence $x_1, x_2, \ldots, x_m$ is then computed as the sum of the likelihoods over all paths.

$$P\left(\mu = c|x_1, \ldots, x_m\right) = \sum_{paths} P\left(path\right) \quad (14)$$

The corresponding path trees for the likelihood computations are shown in figures 3 and 4 for the state corresponding to the normal distribution $N\left(0, \sigma^2\right)$ and the normal distribution $N\left(1, \sigma^2\right)$, respectively.

It is, of course impossible, to compute the exact likelihood as the sum over all possible paths for larger m, since there are 2^m possible paths. Therefore, we restrict the likelihood computation to the last k states, so that the likelihoods are actually computed "backwards" along the paths. In this way, we also do not require the probabilities for the initial states. The error for the differences in the two likelihoods will be quite small with this procedure, since the conditional likelihood for the two states k steps backwards given the final

parameters	Maximum likelihood estimation	TS Evolving Fuzzy Models
$\sigma = 0.5$, $p = 0.25$	0.6390	0.7116
$\sigma = 1.0$, $p = 0.1$	1.2792	2.1124
$\sigma = 1.5$, $p = 0.2$	2.6678	3.7769
$\sigma = 2.0$, $p = 0.1$	4.3743	6.1057
$\sigma = 3.0$, $p = 0.001$	9.1785	12.3010
$\sigma = 6.0$, $p = 0.001$	35.7901	50.3640

TABLE I
MSE

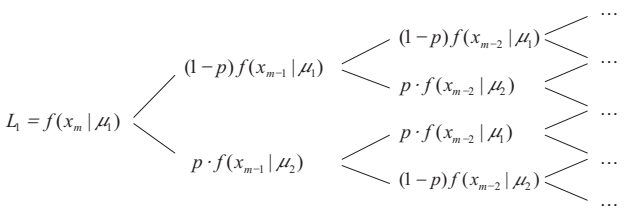

Fig. 3. Likelihood function for the state 1

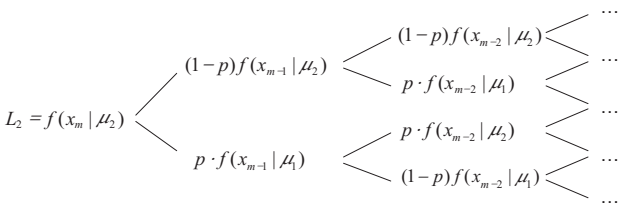

Fig. 4. Likelihood function for the state 2

state will be more or less independent of the final state for larger k according to the theory of Markov chains.

Table I shows a comparison of the described maximum likelihood estimation where we have chosen $k = 7$ with the Takagi Sugeno evolving fuzzy model decribed in [6]. In the case of the maximum likelihood estimation we simply predict the expected value of the normal distribution with the higher likelihood. This prediction could even be improved by choosing a weighted mean between the two expected values.

If we knew from which normal distribution the last value had been sampled, we could base our prediction on the knowledge that we sample from a mixture of two normal distributions. In this case, the expected quadratic error would be $\sigma^2 + p - p^2$. From table I we can see that for small values p, the mean square error of the maximum likelihood-based estimations is very close to theoretical lowest mean square error whereas the evolving system has a much larger error.

IV. Conclusions

We strongly argue in favour to also use benchmarks for evolving systems that are based on well-defined stochastic data generating processes. If only real world data are considered for benchmarks, it is not clear at all, how close an evolving system comes to the *unknown* best solution. With our simple stochastic models, we can explicitly provide or at least estimate the best solution that can be achieved from a theoretical point of view, so that we have a clear measure, how close the evolving system can come to the best solution.

Of course, our comparisons are "unfair", in the sense that we compare techniques from evolving systems, that do not make any specific assumptions about the data generating process, with estimators tailored to the specific problem. The assumption that we know the data generating process in principle and just need to estimate the parameters is unrealistic. Our intention is to give an impression of how much the evolving systems are biased to track the noise or randomness in the data generating process instead of learning the actual dependencies in the data.

Our examples were all restricted to the prediction of the next value. In terms of learning a function, the function was the identity function in the case of the random walks and binary function – providing only two possible outputs – in the case of the switching model. To make the situation more complicated, we could replace the identity function or the binary output by another function. We could also consider a number of our models in parallel to have multiple inputs and aggregate them by a simple function for the output.

We can also use more complicated models for the data generating process. But we should keep in mind that we need to have an idea, how well an optimized model could predict in order to have a comparison with the evolving system.

References

[1] Angelov, P., D. Filev, An Approach to On-line Identification of Evolving Takagi-Sugeno Models. IEEE Trans. Systems, Man, Cybernetics B, vol. 34, no. 1, 2004, pp. 484-498
[2] Angelov, P., E. Lughofer, A Comparative Study of two Approaches for Data-Driven Design of Evolving Fuzzy Systems: eTS and FLEXFIS , Intern. Journal of General Systems, vol.37, issue 1, 2008, pp. 45-67.
[3] Angelov, P., X. Zhou, F. Klawonn, Evolving Fuzzy Rule-based Classifiers, in: Proc. 2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing (CIISP 2007). IEEE, Los Alamitos, 2007, pp. 220-225
[4] Beringer, Hüllermeier. Efficient Instance-Based Learning on Data Streams. Intelligent Data Analysis, vol. 11, 2007, pp. 627-650
[5] Klawonn, F., P. Angelov: Evolving Extended Naïve Bayes Classifiers. In: S. Tsumoto, C.W. Clifton, N. Zhong, X. Wu, J. Liu, B.W. Wah, Y.-M. Cheung: Proc. Sixth IEEE International Conference on Data Mining: Workshops. IEEE, Los Alamitos, 2006, pp. 643-647
[6] Macias-Hernandez, J.J., P. Angelov, X. Zhou, Soft Sensor for Predciting Crude Oil Distillation Side Streams using Takagi Sugeno Evolving Fuzzy Models, Proc. 2007 IEEE International Conference on Systems, Man, and Cybernetics, Montreal, Canada, 2007, pp. 3305-3310.
[7] Spitzer, F. Principles of Random Walk (2nd edition), Springer, Berlin, 2001