Real-time multi-step predictors from data streams

Yoshito Hirata,^{a,*} Taiji Yamada,^b Jun Takahashi,^b and Hideyuki Suzuki^a

^aInstitute of Industrial Science, The University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505, Japan

^bAihara Electrical Engineering Co. Ltd., 2-16-8 Hama-cho, Funabashi, Chiba 273-0012, Japan

*Corresponding author.

Tel.: +81 3 5452 6697.

Fax: +81 3 5452 6694.

Email address: yoshito@sat.t.u-tokyo.ac.jp (Y. Hirata).

Abstract:

An online multi-step prediction method is constructed based on data streams, for which using all observed data points for prediction is impractical. The proposed method is superior in prediction accuracy and computational time, at various prediction steps, compared to two simple extensions of Kwasniok and Smith (Phys. Rev. Lett. **92**, 164101 (2004)). We apply our proposed prediction method to artificial data sets generated from the Lorenz-63 model and to measured wind speed data streams.

Keywords: multi-step nonlinear predictions, data streams, online algorithms, wind speeds

1. Introduction

The development of measurement techniques has produced many kinds of data streams. These data streams contain very large amounts of data that often need to be analyzed in real time. For example, wind speed measurements are important for predicting the output of wind-based power plants. The variability in wind speed has a destabilizing effect on electric power systems, driving up costs and inefficiencies. Multi-step predictions on the time scale of seconds or minutes can compensate for the variability of wind speed, but require correspondingly large computational costs. Few methods can meet the demands of such multi-step wind speed predictions given very large data streams, where using the entire set of measurements is impractical.

In this letter, we propose an online multi-step predictor using data streams by extending the method of Kwasniok and Smith [1] (for another online single step predictor, please see Richard et al. [2]). The method of Kwasniok and Smith [1] adaptably constructs a finite size

database for a single step prediction given a data stream, although the method has not been used well since its proposal possibly because the method is a little bit complicated. The method of Kwasniok and Smith [1] is great in its computational time and memory usage. Instead of preparing multiple databases for direct multi-step predictions in parallel or iteratively applying a single step prediction to produce multi-step predictions, we construct and update a single database for a data stream by evaluating multi-step predictions collectively.

Our extension of this method to multi-step predictions has an improved accuracy and can be obtained in a shorter computational time. Especially, in the test examples of the Lorenz-63 model and real wind speed, the proposed method yields smaller prediction errors than the compared two simple extensions and global linear model when prediction steps are more than 70 steps. Moreover, the computational times required for the proposed method are in the order of hundredth of those required for the two simple extensions, and they are even about one sixth of the times required for a global linear model.

2. Method of Kwasniok and Smith

First, we outline the method of Kwasniok and Smith [1]. Let s_n be a data stream of scalar values measured at a fixed sampling interval for simplicity. It is straightforward to extend the method for data streams of multivariate time series. In order to give a p step-ahead prediction for s_{n+p} , we construct an m -dimensional delay vector $(s_{n-m+1}, \dots, s_n)$. Then,

we consider the simple nonlinear prediction described in [3], where the prediction $\hat{s}_{n+p}$ is the average of local neighbors measured in the m -dimensional delay vector. These neighbors are selected from a database built until time n . Let $L = \{(x_\alpha, y_\alpha)\}_{\alpha=1}^N$ be a database of size N with $x_\alpha = (s_{n_\alpha-m+1}, \dots, s_{n_\alpha})$ and $y_\alpha = s_{n_\alpha+p}$. We use the first N points from the data stream in order to construct the initial database $L_0 = \{(x_\alpha, y_\alpha)\}_{\alpha=1}^N$,

where $x_\alpha = (s_\alpha, \dots, s_{\alpha+m-1})$ and $y_\alpha = s_{\alpha+m-1+p}$. Setting $L = L_0$ initially, for each time later than $N + m + p$ for which a new point appears in the data stream, we then apply the following algorithm:

- (1) Let $w = (s_{n'-m+1+p}, \dots, s_{n'+p})$ be the most recent part of the dataset. From this part, we construct a prediction $\hat{s}_{n'+2p}$ for $s_{n'+2p}$ by the following procedure. First, find the k nearest neighbors from the database by calculating and sorting the distances $\|w - x_\alpha\|$ for all $\alpha = 1, \dots, N$. For $i = 1, \dots, k$, let $b_i(w) \in \{1, \dots, N\}$ be an index of the i th nearest neighbor to w . Second, obtain the prediction $\hat{s}_{n'+2p}$ by $\frac{1}{k} \sum_{i=1}^k y_{b_i(w)}$.
- (2) Let $(u, v) = ((s_{n'-m+1}, \dots, s_{n'}), s_{n'+p})$ be a new point from the data stream.
- (3) Based on the current database L , calculate a prediction error, which is used for evaluating whether we update L or not. Similarly as in Step (1), find the k nearest neighbors from the database by calculating and sorting $\|u - x_\alpha\|$. Denoting an index of the i th nearest neighbor to u by $b_i(u) \in \{1, \dots, N\}$ ($i = 1, \dots, k$), obtain the prediction $\hat{s}_{n'+p}$ of $s_{n'+p}$ by $\frac{1}{k} \sum_{i=1}^k y_{b_i(u)}$ and calculate the prediction error $\mathcal{E}' = |s_{n'+p} - \hat{s}_{n'+p}|$.
- (4) Choose a reference point $(x_{\alpha^*}, y_{\alpha^*})$ randomly from L .
- (5) Construct a temporary database $L^* = L \cup \{(u, v)\} \setminus \{(x_{\alpha^*}, y_{\alpha^*})\}$, predict y_{α^*} from x_{α^*} based on L^* as similarly as Step (3), and calculate the prediction error $\mathcal{E}^* = |s_{n_{\alpha^*}+p} - \hat{s}_{n_{\alpha^*}+p}|$.
- (6) If $\mathcal{E}^* < \mathcal{E}'$, then replace L with L^* .

This algorithm of Kwasniok and Smith predicts a single step. If $p = 1$, we may merge Steps (1) and (3) into a single step in order to have a simple algorithm as listed in [1] (see Fig. 1 for the schematic diagram of the method of Kwasniok and Smith for $p = 1$).

The method of Kwasniok and Smith has been extended toward the prediction of phase space in [4]. In order to extend the method of Kwasniok and Smith to a multi-step predictor, we may prepare a database L for each p and apply the above algorithm subsequently from a smaller p . We call this extension the naive extension. The other obvious extension for a multi-step predictor is to prepare a single step prediction and apply

the prediction iteratively. We call this extension the iterative extension.

3. Proposed method

Instead of the naive extension and the iterative extension, we extend the method of Kwasniok and Smith to multi-step predictions in the following way. Suppose we construct

multi-step predictions up to Q steps ahead. Let $L = \{(x_\alpha, y_\alpha)\}_{\alpha=1}^N$ be a database of size

N with $x_\alpha = (s_{n_\alpha-m+1}, \dots, s_{n_\alpha})$ and $y_\alpha = (s_{n_\alpha+1}, \dots, s_{n_\alpha+Q})$. Construct the initial database

$L_0 = \{(x_\alpha, y_\alpha)\}_{\alpha=1}^N$, where $x_\alpha = (s_\alpha, \dots, s_{\alpha+m-1})$ and

$y_\alpha = (y_{\alpha,1}, \dots, y_{\alpha,Q}) = (s_{\alpha+m}, \dots, s_{\alpha+m+Q-1})$. We set $L = L_0$ at the beginning of the

algorithm. Later, for each time when a new point appears in the data stream, we perform the following steps (see Fig. 2 for the schematic diagram):

(1) Let $w = (s_{n'-m+1+Q}, \dots, s_{n'+Q})$ be the most recent part of the dataset. From this part, we

construct a prediction $\hat{s}_{n'+Q+p}$ for $s_{n'+Q+p}$ and each p ($0 < p \leq Q$) by the following

procedure. First, find the k nearest neighbors from the database by calculating and

sorting the distances $\|w - x_\alpha\|$ for all $\alpha = 1, \dots, N$. For $i = 1, \dots, k$, let

$b_i(w) \in \{1, \dots, N\}$ be an index of the i th nearest neighbor to w . Second, for each p ,

obtain the prediction $\hat{s}_{n'+Q+p}$ by $\frac{1}{k} \sum_{i=1}^k y_{b_i(w), p}$.

(2) Let $(u, v) = ((s_{n'-m+1}, \dots, s_{n'}), (s_{n'+1}, \dots, s_{n'+Q}))$ be a new point from the data stream.

(3) We calculate prediction errors based on the current database L for evaluating whether we update L or not. Find the k nearest neighbors from the database, whose indexes are denoted by $b_i(u)$ ($i = 1, \dots, k$) from the smaller ones. For each

p ($0 < p \leq Q$), obtain the prediction $\hat{s}_{n'+p}$ of $s_{n'+p}$ by $\frac{1}{k} \sum_{i=1}^k y_{b_i(u), p}$, and calculate

the prediction error $\varepsilon_p = |s_{n'+p} - \hat{s}_{n'+p}|$.

(4) Choose a reference point $(x_{\alpha^*}, y_{\alpha^*})$ randomly from L .

(5) Construct a temporary database $L^* = L \cup \{(u, v)\} \setminus \{(x_{\alpha^*}, y_{\alpha^*})\}$, predict y_{α^*} from

x_{α^*} based on L^* in a similar way to Step (3), and calculate the prediction error

$$\varepsilon_p^* = \left| s_{n_{\alpha^*}+p} - \hat{s}_{n_{\alpha^*}+p} \right| \text{ for each } p.$$

- (6) If the number of $p \in \{1, \dots, Q\}$ satisfying $\varepsilon_p^* < \varepsilon_p'$ is greater than $0.5Q$, then replace L with L^* .

There are two advantages for the proposed extension. First, the proposed extension provides more accurate predictions than the naive and iterative extensions because here a more appropriate database is built by constructing multi-step predictions simultaneously. In the naive and iterative extensions, prediction errors are evaluated separately for each single prediction step p . It is known that evaluating multi-step prediction errors simultaneously improves the reconstruction of attractors [5].

The second advantage is that the computational cost is smaller than the naive and iterative extensions mentioned above since in the present extension, sorting is used three times per observation, while in the naive and iterative extensions sorting is performed $3Q$ and $(Q+2)$ times per observation, respectively. Therefore, the proposed extension enjoys a smaller computational burden when $Q > 1$.

An additional advantage is that the proposed extension needs memory space proportional to $(m+Q)N$, while the naive extension needs memory space proportional to $Q(m+1)N$ and $(m+Q)N < Q(m+1)N$ for $Q > 1$. The iterative extension uses the smallest portion of memory, which is proportional to $(m+1)N$.

4. Method of comparison

In order to compare the present method with the iterative and naive approaches, we used global linear models. Assume that we construct a global linear model predicting p steps ahead. The global linear model can be written as follows [6]:

$$\begin{aligned} \hat{s}_{n'+p} &= c_0(p) + \sum_{i=0}^{m-1} c_{i+1}(p) s_{n'-i} = C(p)S(n'), \\ C(p) &= (c_0(p), c_1(p), \dots, c_m(p)), \\ S(n') &= (1, s_{n'}, s_{n'-1}, \dots, s_{n'-m+1})^T. \end{aligned}$$

We adaptively adjust parameters as follows by making the method of parameter estimation, as described in Ref. [6], online. Suppose that a $(m+1) \times (m+1)$ matrix V and a

$(m+1) \times 1$ matrix $Y(p)$ for each p are given. In addition, assume that we have the current parameters $C(p)$. When a new point appears, we first predict the future value for each p as:

$$\hat{s}_{n'+p} = C(p)S(n').$$

Second, we perform the calculation:

$$V \leftarrow V + \begin{bmatrix} 1 & s(n'-Q) & \cdots & s(n'-m+1-Q) \\ s(n'-Q) & s(n'-Q)s(n'-Q) & \cdots & s(n'-Q)s(n'-m+1-Q) \\ \vdots & \vdots & \ddots & \vdots \\ s(n'-m+1-Q) & s(n'-m+1-Q)s(n'-Q) & \cdots & s(n'-m+1-Q)s(n'-m+1-Q) \end{bmatrix}.$$

Third, for each p we calculate:

$$Y(p) \leftarrow Y(p) + \begin{bmatrix} s(n'-Q+p) \\ s(n'-Q+p)s(n'-Q) \\ \vdots \\ s(n'-Q+p)s(n'-m+1-Q) \end{bmatrix},$$

$$C(p) \leftarrow V^{-1}Y(p).$$

We also use persistence prediction for comparison, where we let the current value be the predicted value of p steps ahead.

5. Examples

The first example we use is for the predictions of the Lorenz-63 model [7]. These model equations can be written as

$$\begin{aligned} \dot{x} &= \sigma(y - x), \\ \dot{y} &= rx - y - xz, \\ \dot{z} &= -bz + xy. \end{aligned}$$

This model is the most appropriate test bed because the model extracts the most dominant mode of the atmospheric currents in the air. We set $(\sigma, r, b) = (10, 28, 8/3)$, and used ode45 in MATLAB for integrating the equations. After discarding the initial transient response, we observe the x variable every 0.01 unit in time in order to have a time series of length 20,000. We set $N = 500$, $k = 25$, and $Q = 240$. We tested m for 10, 15, 20, 25, and 30. We evaluate the prediction errors from the 10001st point until the end.

According to Fig. 3, the predictions using the proposed method are better than the persistence prediction when the prediction steps are more than 0.01. In addition, the proposed predictions are mostly better than the global linear prediction, the naive extension, and the iterative extension for prediction time greater than 0.16, 0.51, and 0.66, respectively. In particular, the proposed predictions are the best for longer prediction steps. The proposed predictions do not largely depend on m . In addition, the proposed method is faster than the other methods, including the global linear model (Table 1). We used a machine with an Intel Xeon 1.87GHz CPU for this numerical experiment.

An example of predicted time series compared with the actual realization is presented in Fig. 4. We can see a tendency that when the prediction period gets larger, the prediction error gets larger.

We also examined a case where 5% observational noise was added to the dataset of Lorenz-63 model. Figure 5 demonstrates that the proposed method still produced efficient predictions for the case under the influence of the observational noise.

Moreover, we investigated a case where dynamical noise of Gaussian distribution with mean 0 and standard deviation 0.01h was added to each variable at each step of the integration by the fourth order Runge-Kutta method, where h is the integration step and we set $h = 0.01$ here. The results are shown in Fig. 6. Even if there was dynamical noise, the proposed method produced the reasonable multi-step predictions compared with the global linear and persistence predictions.

As a second example, we tested our proposed method on the prediction of wind speeds. We used the measured dataset from Ref. [8]. We have so far learned that there are a serial dependence [9,10] and laminarity [10] for this dataset. This data set can also sometimes show nonlinearity [8]. Therefore, we examine here whether or not we can construct online multi-step predictions by the proposed method superior to persistence and global linear predictions. The length of the time series is for a day. The original time series was sampled at 50 Hz. We took the average of the wind speed for the north-and-south direction every 1 second. The averaged time series then contains 86,992 points. Here, we used $N = 500$, $k = 25$, $Q = 240$, and $m = 20, 25, 30$. We evaluated the prediction errors from the 10001st point until the end of the time series.

According to Fig. 7, the proposed method led to smaller prediction errors when the prediction steps are more than 28, 64, 8, and 6 seconds compared with the persistence predictions, the global linear models, the naive extension, and the iterative extension, respectively. This observation held for various m . In addition, the computation time required for the proposed method is shorter than that for the global linear model, the naive extension, and the iterative extension (Table 2).

In this example, we also shows a sample of predicted time series, which is compared with the actual data (see Fig. 8). We could also see the tendency that the prediction error is larger when the prediction period is longer.

6. Discussions

The parameter m should be chosen large enough so that delay vectors become overembedding in order to cope with the nonstationarity of given data streams [11]. The results presented in Figs. 3, 5, 6, and 7 did not depend on m much. How to choose m optimally should be an interesting future research topic.

If we make the size N of the database larger, then we may be able to produce more accurate predictions (Fig. 9). However, a larger database will need more computational time. Therefore, there is a trade-off between the size N of database and the computational time. There is also an appropriate number k of neighbors that minimizes the prediction errors (Fig. 10) due to the trade-off between the accuracy and the observational noise. Hence, the size N of database and the size k of neighbors might be chosen depending on the desired application. But, we feel that the proposed method is relatively robust against the choice of k because the prediction error varied by about 40% for the tested k 's.

The proposed method may be further modified in order to deal with point process data such as trades in foreign exchange markets [12]. In such an application, constructing a set of databases of fixed size from given data streams and predicting their multi-step futures is required.

7. Conclusions

In conclusion, we have extended the method of Kwasniok and Smith [1] towards multi-step predictions. Instead of comparing the single step prediction error obtained by the current database with that obtained by a temporary database for each prediction step, we compare

the database (current or temporary) that provides smaller prediction errors for more prediction steps. If the temporary database provides smaller prediction errors for more prediction steps, then we replace the current database with the temporary database. We demonstrated the proposed extension for predicting the future behavior of the Lorenz-63 system and wind speeds. We believe that the proposed real-time prediction method as applied to wind speeds may help introduce large amounts of renewable energy resources into an electric power system without causing instabilities.

Acknowledgements

We thank Kazuyuki Aihara for discussions. This study was supported by Japan Science and Technology Agency through "Adaptable and Seamless Technology transfer Program through target-driven R&D (A-STEP)" Feasibility Study (Seeds Validation, Grant No. AS2311244A).

References

- [1] F. Kwasniok, L. A. Smith, Phys. Rev. Lett. 92 (2004) 164101.
- [2] C. Richard, J. C. M. Bermudez, P. Honeine, IEEE T. Signal Proces. 57 (2009) 1058-1067.
- [3] H. Kantz, T. Schreiber, Nonlinear Time Series Analysis, Cambridge University Press, Cambridge, UK, 2004.
- [4] T. Yamada, J. Takahashi, K. Aihara, Reliability Engineering Association of Japan 28 (2006) 489-496 (in Japanese).
- [5] L. C. Uzal, G. L. Grinblat, P. F. Verdes, Phys. Rev. E 84 (2011) 016223.
- [6] J. D. Hamilton, Time Series Analysis, Princeton University Press, Princeton, NJ, 1994.
- [7] E. N. Lorenz, J. Atmos. Sci. 20 (1963) 130-141.
- [8] Y. Hirata, D. P. Mandic, H. Suzuki, K. Aihara, Philos. T. R. Soc. A 366 (2008) 591-607.
- [9] Y. Hirata, S. Horai, H. Suzuki, K. Aihara, Phys. Lett. A 370 (2007) 265-174.
- [10] Y. Hirata, K. Aihara, Int. J. Bifurcat. Chaos 21 (2011) 1077-1084.
- [11] R. Hegger, H. Kantz, L. Matassini, T. Schreiber, Phys. Rev. Lett. 84 (2000) 4092-4095.
- [12] Y. Hirata, K. Aihara, Physica A 391 (2012) 760-766.

Figure captions:

Fig. 1. Schematic diagram for the original Kwasniok and Smith method for $p = 1$.

Fig. 2. Schematic diagram for the extended Kwasniok and Smith method (the proposed method).

Fig. 3. Prediction steps vs. prediction errors for the Lorenz-63 model. The solid black lines, dotted black lines, dash-dotted black lines, dash-dotted gray lines, and dashed gray line represent the proposed method, the naive extension, the iterative extension, the global linear model, and the persistence prediction, respectively. For each method, the results for $m = 10, 15, 20, 25, 30$ are overlaid.

Fig. 4. An example of predicted time series (black solid line) compared with the actual time series (gray dashed line), in the case of noise-free Lorenz-63 model. Here we use $m = 20$.

Fig. 5. Prediction steps vs. prediction errors for the Lorenz-63 model with 5% observational noise. The solid black lines, dashed-dotted gray lines, and dashed gray line, represent the the proposed method, the global linear model, and the persistence prediction, respectively. For each method, the results for $m = 10, 15, 20, 25, 30$ are overlaid.

Fig. 6. Prediction steps vs. prediction errors for the Lorenz-63 model with dynamical noise. See the caption of Fig. 5 to interpret this figure.

Fig. 7. Prediction steps vs. prediction errors for wind speed predictions. The solid red lines, dotted red lines, dash-dotted red lines, dash-dotted blue lines, and dashed black line represent the proposed method, the naive extension, the iterative extension, the global linear model, and the persistence prediction, respectively. For each prediction method, the results for $m = 20, 25, 30$ are overlaid.

Fig. 8. An example of predicted time series (black solid line) compared with the actual data (gray dashed line), in the case of real wind speed. Here we use $m = 20$.

Fig. 9. The size N of database vs. prediction errors at 0.6 unit times ahead, in the case of Lorenz-63 model with the observational noise.

Fig. 10. The number k of neighbors vs. prediction errors at 0.6 unit times ahead, in the case of Lorenz-63 model with the observational noise.

Table 1

The computational time required for each method for the Lorenz-63 model ($m = 20$).

Method	Time (seconds)
The proposed method	53
The naive extension	6410
The iterative extension	2897
Global linear model	342

Table 2

The computational time required for each method for wind speed predictions ($m = 20$).

Method	Time (seconds)
The proposed method	236
The naive extension	29059
The iterative extension	12687
Global linear model	1573

(3) Evaluate the prediction error.

(6) If the prediction error for y_{α^*} based on L^* is smaller than that for v based on L , then replace L with L^* .

(6) If more than a half prediction steps satisfy that the mean prediction error for y_{a^*} based on L^* is smaller than that for v based on L , then replace L with L^* .

Figure3

Figure4

Figure5

Figure6

Figure7

Figure8

Figure9

Figure10