Multistep-ahead Time Series Prediction

Haibin Cheng[1], Pang-Ning Tan[2], Jing Gao[3], Jerry Scripps[3]

Department of Computer Science and Engineering
Michigan State University
chenghai,ptan,gaojing2,scripps@msu.edu

Abstract. Multistep-ahead prediction is the task of predicting a sequence of values in a time series. A typical approach, known as multi-stage prediction, is to apply a predictive model step-by-step and use the predicted value of the current time step to determine its value in the next time step. This paper examines two alternative approaches known as independent value prediction and parameter prediction. The first approach builds a separate model for each prediction step using the values observed in the past. The second approach fits a parametric function to the time series and builds models to predict the parameters of the function. We perform a comparative study on the three approaches using multiple linear regression, recurrent neural networks, and a hybrid of hidden Markov model with multiple linear regression. The advantages and disadvantages of each approach are analyzed in terms of their error accumulation, smoothness of prediction, and learning difficulty.

1 Introduction

Many time series problems involve the task of predicting a sequence of future values using only the values observed in the past. Examples of this task, which is known as ***multistep-ahead time series prediction*** [1], include predicting the time series for crop yield, stock prices, traffic volume, and electrical power consumption. By knowing the sequence of future values, we may derive interesting properties of the time series such as its projected amplitude, variability, onset period, and frequency of abnormally high or low values. For example, multistep-ahead time series prediction allows us to forecast the growing period of corn for next year, the maximum and minimum temperature for next month, the frequency of El-Nino events in the next decade, etc.

A typical approach to solve this problem is to construct a single model from historical values of the time series and then applies the model step by step to predict its future values. This approach is known as ***multi-stage prediction***. Since it uses the predicted values from the past, it can be shown empirically that multi-stage prediction is susceptible to the error accumulation problem, i.e., errors committed in the past are propagated into future predictions.

This paper considers two alternative approaches for multistep-ahead time series prediction. The first approach, known as ***independent value prediction***, builds a separate model for each prediction step using only its past observations. The second

approach, known as ***parameter prediction***, fits a parametric function to the time series and builds regression models to predict the parameters of the function.

We implement allthree prediction approaches using multiple linear regression [2], recurrent neural networks [3], and a hybrid of hidden Markov model with multiple linear regression [7] as the underlying regression methods. The advantages and disadvantages of each prediction approach are analyzed in terms of their error accumulation, smoothness of prediction, and learning difficulty.

2 Methodology

A time series is a sequence of observations in which each observation x_t is recorded at a particular timestamp t. A time series of length t can be represented as a sequence $X=[x_1,x_2,...,x_t]$. We use the notation X_{t-p}^{t} to denote a segment of the time series $[x_{t-p},x_{t-p+1},...,x_t]$. Multistep-ahead prediction is the task of predicting a sequence of h future values, X_{t+1}^{t+h}, given its p past observations, X_{t-p+1}^{t}.

2.1 Regression Methods

This section presents the regression methods used for modeling the time series.

2.1.1 Multiple Linear Regression (MLR). The MLR model, which is also called the AR model, is given by the equation: $f\left(X_{t-p+1}^{t}\right)=\sum_{i=1}^{p} a_i x_{t-i+1} + \varepsilon_t$, where ε_t corresponds to a random noise term with zero mean and variance ^{2}. The coefficient vector $[a_1, a_2, \ldots, a_p]^T$ is estimated using the least square method by minimizing the sum of squared error, *SSE*, of the training data. The variance is estimated using *SSE*/*h*, where *h* is the size of the prediction window.

2.1.2 Recurrent Neural Networks (RNN) has been successfully applied to noisy and non-stationary time series prediction. In RNN, the temporal relationship of the time series is explicitly modeled using feedback connections [3] to the internal nodes (known as hidden units). An RNN model is trained by presenting the past values of the time series to the input layer of the Elman back propagation network [4]. The weights of the network are then adjusted based on the error between the true output and the output predicted by the network until the algorithm converges. Before the network is trained, the user must specify the number of hidden units in the network and the stopping criteria of the learning algorithm.

2.1.3 Hybrid HMM/MLR Model is an extension of traditional hidden Markov model applied to regression analysis [7]. This method is an effective way for modeling piecewise stationary time series, where the observed values are assumed to be generated by a finite number of hidden states. Let (Z_t) denote the Markov chain

on the state space $S = \{s_1, s_2, \ldots, s_N\}$. The initial probability for a given state s is denoted as π_s, while the transition from one state to another is characterized by the transition matrix $A = (a_{ij})$, where $P(Z_{t+1} = s_j \mid Z_t = s_j) = a_{ij}$. At time t, the observed value x_t depends only on the current state Z_t : $x_t = f_{z_t}(X_{t-p}^{t-1}) + e(0, \sigma_{z_t})$ where $f_{z_t} \in \{f_{s_1}, f_{s_2}, \ldots, f_{s_N}\}$ is the corresponding regression function and $e(0, \sigma_s)$ is a noise term with mean zero and a variance σ_s^2 that depends on the current state, s. We use the regression function produced by MLR in our experiments. The hybrid HMM/MLR model is trained by maximizing the following likelihood function:

$$L_\theta(X_1^t) = \sum_Z P(X_1^t; Z) = \sum_Z \pi_{z_1} \prod_{i=2}^{t} P(z_{i+1} \mid z_i) \Phi\left(X_i - f_{z_i}(X_{i-p}^{i-1})\right) \quad \textbf{(1)}$$

A brute force method for maximizing the likelihood function requires a complexity of $O(N^T)$ operations. However, an efficient approach called the forward-backward procedure can reduce the complexity of the computation down to $O(N^2 T)$. This procedure is based on the well-known expectation-maximization (EM) algorithm.

2.2 Prediction Approaches

We investigate three approaches for predicting the sequence of future values X_{t+1}^{t+h} from a given time series X_1^t. A training set D is initially created from the time series using a sliding window of length $p+h$ (see Figure 1). Each instance of the sliding window corresponds to a record in the training set, as shown in Table 1. The input X corresponds to the first p values of the window while the output Y corresponds to the remaining h values of the window. For example, the first record of the training set D contains $X = [x_1, x_2, \ldots, x_p]$ as its input variables and $Y = [x_{p+1}, x_{p+2}, \ldots, x_{p+h}]$ as its output variables. Similarly, the second record contains $X = [x_2, x_3, \ldots, x_{p+1}]$ as its input variables and $Y = [x_{p+2}, x_{p+3}, \ldots, x_{p+h+1}]$ as its output variables, while the last record contains $X = [x_{t-h-p+1}, x_{t-h-p+2}, \ldots, x_{t-h}]$ as its input variables and $Y = [x_{t-h+1}, x_{t-h+2}, \ldots, x_t]$ as its output variables. For notational convenience, we use $Y(i)$ to refer to all the values in the i^{th} column of Y in D. For example, $Y(3) = [x_{p+3}, x_{p+4}, \ldots, x_{t-h+3}]^T$.

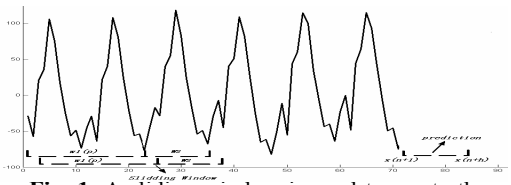

Fig. 1. A sliding window is used to create the regression training set D=X'+Y.

Table 1. Traning Set $D = X \times Y$

$X = [X(1), \ldots, X(p)]$	$Y = [Y(1), \ldots, Y(h)]$
$[x_1, x_2, \ldots, x_p]$	$[x_{p+1}, x_{p+2}, \ldots, x_{p+h}]$
$[x_2, x_3, \ldots, x_{p+1}]$	$[x_{p+2}, x_{p+3}, \ldots, x_{p+h+1}]$
$\vdots \quad \cdot$	$\vdots \quad \cdot$

2.2.1 Multi-stage Prediction predicts the future values of a time series in a step by step manner. We first predict x_{t+1} using the previous p values, $x_{t+1-p},\ldots,x_{t-1},x_t$. We then predict x_{t+2} based on its previous p values, which includes the predicted value for x_{t+1}. The procedure is repeated until the last value, x_{t+h}, has been estimated. In this approach, it is sufficient to construct a single model for making the prediction.

2.2.2 Independent Value Prediction predicts the value at each time step using a separate model. Given the initial data set shown in Table 1, we first create *h* training sets, each of which has the same input *X*, but different output *Y*. We use Y(1) as the output variable for the first training set, Y(2) as the output variable for the second training set, and so on. By learning each training set independently, we obtain *h* regression models f_i (i = 1,2,…, h). The models are then used to predict the next *h* values as follows: $x_{t+i} = f_i(X),\ \ i = 1,2,...,h$.

2.2.3 Parameter Prediction transforms the problem of predicting *h* output values into an equivalent problem of predicting (*d*+1) parameters. For each record in Table 1, we fit a parametric function *g* to the output vector Y. Let (c_0, c_1,…, c_d) denote the parameters of the function *g*. We then replace the original output vector Y=[Y(1),Y(2),…,Y(h)] with a modified output vector Y'=[c_0, c_1, …, c_d]. We now construct *(d+1)* regression models f_i (i = 0,1,2,…,*d*), one for each output column Y'. The models are then applied to predict the (*d+1)* parameters of a test sequence. The test sequence is reconstructed by substituting the predicted parameters into the parametric function *g*. While this methodology is generally applicable to any family of parametric functions, we use polynomial functions in our experiments.

2.3 Model Selection

The parameters for our prediction approaches include the order of regression model *p*, the size of prediction window *h*, and the degree of polynomial fit *d* (for parameter prediction). The size of the prediction window *h* is domain dependent and depends on the nature of the application. We use Akaike's final prediction error (FPE) [8] as our criterion for determining the right order for *p* in the MLR model.

$$FPE = \hat{\delta}^2 \frac{t+p}{t-p} \quad \text{where} \quad \hat{\delta}^2 = \frac{\sum_{j=1}^{t-p-h} (y_{j1} - \hat{y}_{j1})^2}{t-p-h} \qquad \textbf{(2)}$$

The same criterion is applicable to estimate the degree of the polynomial function used in parameter prediction. To determine the correct order for RNN, we employ the method described by Kennel in [5]. Let X_p denote as an instance of the training data and $X_p^{(n)}$ denote its nearest neighbor. The pair is declared as false nearest neighbors if $\left|\frac{d(X_p, X_p^{(n)}) - d(X_{p+1}, X_{p+1}^{(n)})}{d(X_p, X_p^{(n)})}\right|$ exceeds a user-specified threshold (where *d* refers to the

distance between a pair of observations). Our goal is to choose a value for p such that the number of false nearest neighbors is close to zero.

3 Experiments and Discussions

We perform a comparative study on the three prediction approaches using both real and synthetic datasets. The real datasets are obtained from the UCI Machine Learning Repository [9] and the Time Series Data Library [6]. Our experiments were conducted on a Pentium 4 machine with 3GHz CPU and 1GB of RAM.

3.1 Evaluation Metric

The estimation error of a prediction approach is evaluated based on the following measure: $RMSE = \sum_i (y_i - \hat{y}_i)^2 / \sum_i (y_i - \bar{y})^2$, where y_i is the true value, $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the average value of the time series. The RMSE values recorded in our experimental results are obtained using ten-fold cross validation.

A **Win-Draw-Loss Table** is created to compare the relative performance between two prediction approaches when applied to n data sets. We use the criterion of 0.01 difference in RMSE to determine whether one approach wins or loses against another approach. For a stricter evaluation, we also apply the **paired t significance test** to determine whether the observed difference in RMSE is statistically significant. To do this, we first calculate the difference (d) in the RMSE obtained from two prediction approaches on each data set. The mean $\bar{d}$ and standard deviation s_d of the observed differences are also calculated. To determine whether the differences are significant, we compute their T-statistic: $t = \bar{d}\sqrt{n}/s_d$ which follows a t-distribution with n-1 degrees of freedom. Under the null hypothesis that the two prediction approaches are comparable in performance, we expect the value of t should be close to zero. From the computed value for t, we estimate the p-value of the difference, which corresponds to the probability of rejecting the null hypothesis. We say the difference in performance is statistically significant if $p<0.05$ and highly statistically significant if $p < 0.001$.

3.2 Error Accumulation

Error accumulation refers to the propagation of past prediction errors into future predictions. To gain a better insight into this problem, we employ the bias-variance decomposition for squared loss functions. Consider a time series generated by the model $x_{t+1} = f(X_{t-p}^t) + e(0,\sigma^2)$. Let $(y_1, y_2, \cdots, y_h)$ denote the observed values of the time series in a prediction window of length h, i.e., $y_1 = x_{t+1}, y_2 = x_{t+2}, \ldots, y_h = x_{t+h}$. Furthermore, let $(y_1^*, y_2^*, \cdots, y_h^*)$ be the corresponding values generated by the deterministic model f. In other words, $y_i = y_i^* + e(0,\sigma^2)$. We use the notation

$(v_1, v_2, \cdots, v_h)$ to denote the values predicted by a regression model, g. The mean squared error (MSE) at each prediction step j is defined as: $MSE(j) = E[(y_j - v_j)^2]$. The MSE at each step can be decomposed into the following three components [10]:

$$MSE(j) = (E(v_j) - y_j^*)^2 + E[(v_j - E(v_j))^2] + E[(y_j - y_j^*)^2] \quad \textbf{(3)}$$

The first term represents the squared bias of the model; the second term represents the variance of the model; while the third term represents the variability due to noise. The next example illustrates the error accumulation problem for the noise term.

Example 1: *Consider the following AR(2) model:* $x_{t+1} = a_1 x_t + a_2 x_{t-1} + \varepsilon$, *where* ε *has mean zero and variance* σ^2. *Suppose we were able to model accurately the coefficients* a_1 *and* a_2 *using MLR. For the multi-stage approach, we can show that:*

$$y_1 = a_1 x_t + a_2 x_{t-1} + \varepsilon_1 = v_1 + \varepsilon_1$$
$$\therefore MSE(1) = E[(y_1 - v_1)^2] = E(\varepsilon_1^2) \propto \sigma^2.$$
$$y_2 = a_1 y_1 + a_2 x_t + \varepsilon_2 = a_1 v_1 + a_2 x_t + (a_1 \varepsilon_1 + \varepsilon_2) = v_2 + (a_1 \varepsilon_1 + \varepsilon_2)$$
$$\therefore MSE(2) \propto (a_1^2 + 1)\sigma^2.$$
$$y_3 = a_1 y_2 + a_2 y_1 + \varepsilon_3 = a_1 v_2 + a_2 v_1 + (a_1^2 \varepsilon_1 + a_1 \varepsilon_2 + a_2 \varepsilon_1 + \varepsilon_3)$$
$$= v_3 + (a_1^2 \varepsilon_1 + a_1 \varepsilon_2 + a_2 \varepsilon_1 + \varepsilon_3)$$
$$\therefore MSE(3) \propto (a_1^4 + a_1^2 + a_2^2 + 1)\sigma^2.$$

The preceding formula shows the accumulation of errors due to noise for multi-stage prediction as the prediction step increases. For independent value prediction:

$$y_1 = a_1 x_t + a_2 x_{t-1} + \varepsilon_1$$
$$y_2 = (a_1^2 + a_2) x_t + (a_1 a_2) x_{t-1} + (a_1 \varepsilon_1 + \varepsilon_2)$$
$$y_3 = (a_1^3 + 2 a_1 a_2) x_t + (a_2^2 + a_1^2 a_2) x_{t-1} + (a_1^2 \varepsilon_1 + a_1 \varepsilon_2 + a_2 \varepsilon_1 + \varepsilon_3)$$

Assuming that the coefficients for x_t *and* x_{t-1} *can be accurately estimated by MLR, the noise terms for multi-stage and independent value prediction are identical.*

The preceding example illustrates that error accumulation due to noise is unavoidable, regardless of the prediction approach. To analyze the error accumulation due to the bias and variance of a model, we generate the following time series: $X_t = 0.418 X_{t-1} + 0.634 X_{t-2} + \varepsilon$, where ε is a Gaussian noise with mean zero and variance σ^2=0.1. The length of the time series is set to 1000 and the prediction window is h = 50. To ensure there is sufficient bias in the model, we set p = 1. The bias and variance of the models are estimated by generating 500 bootstrap replicates of the training set D and inducing a model g from each bootstrap replicate. The models are then applied to the test sequence to obtain 500 estimated values (v_j) for each prediction step j. The empirical bias is computed by taking the average value of the 500 predictions ($\bar{v}_j$) and subtracting it from the value predicted using the deterministic model. The variance of the models can also be estimated as follows: $\text{var}(j) = E[(v_j - \bar{v}_j)^2]$. Figures 2 and 3 illustrate the bias and variance for multi-stage and independent value predictions (using MLR and a hybrid HMM/MLR as the underlying regression methods). Both figures show that the bias and variance for

multi-stage prediction grows steadily with increasing time steps, whereas the bias and variance for independent value prediction do not appear to be propagated into future predictions.

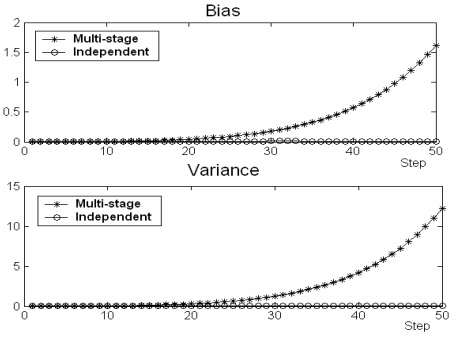

Fig. 2. Bias and variance for MLR

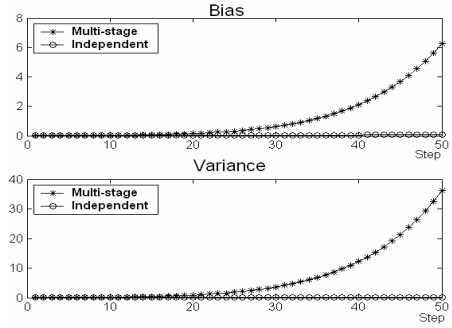

Fig. 3. Bias and variance for HMM/MLR

Therefore, error accumulation is a major problem in multi-stage prediction, irrespective of the choice of regression methods. However, it is not a problem for parameter prediction because the models for predicting different parameters are built independently (similar to independent value prediction).

3.3 Learning Difficulty

Multi-stage prediction builds a single model to fit the entire time series. In contrast, we need to build *h* models for independent value prediction and *(d+1)* models for parameter prediction. Model building is therefore more expensive for independent value and parameter prediction approaches compared to the multi-stage approach.

The model to be learnt by independent value prediction also becomes more complex with increasing time steps. To illustrate this, let *f* denote the true model that generates the data, i.e., $X_t = f(X_{t-p}^{t-1})$. For simplicity, let (x_1, x_2, …, x_p) denote the input variables and $y_i = x_{p+i}$ denote the *h* output variables:

$$
\begin{aligned}
y_1 &= f(x_1, x_2, \ldots, x_p) = f_1(x_1, x_2, \ldots, x_p) \\
y_2 &= f(x_2, \ldots x_p, y_1) = f(x_2, \ldots x_p, f_1) \\
&= f_2(x_1, x_2, \ldots, x_p) \\
&\vdots \\
y_h &= f(\ldots, y_{h-2}, y_{h-1}) = f(\ldots, f_{h-2}, f_{h-1})
\end{aligned}
$$

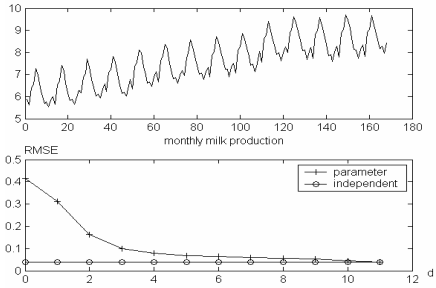

Fig. 4. Prediction Results (p=12,h=12)

If *f* is a linear function, then all the f_k's constructed by the independent value prediction approach are also linear functions. If *f* is non-linear, then the f_k's become increasingly complex functions of the input variables (x_1, x_2, …, x_p). Unless the regression method is very flexible, learning the appropriate model for each time step

can be quite a challenging task. For parameter prediction, the learning difficulty depends on how easy it is to find the appropriate function that fits the output vector.

To compare parameter prediction against independent value prediction, we apply both methods to the monthly *milk production data* (see the top diagram in Figure 4) using *h=12* and *p=12*. For parameter prediction, we use a polynomial function to fit the output vector and vary the degree of the polynomial from 0 to 11. We then employ MLR to predict the parameters of the polynomial. The bottom diagram of Figure 4 shows a comparison between the RMSE of parameter prediction against independent value prediction as the degree of the polynomial function is varied. Observe that the RMSE for parameter prediction drops dramatically when the polynomial degree increases to 3 and decreases slowly thereafter. This result suggests that it is sufficient to fit a polynomial of degree 4 to the output vector and achieves comparable accuracy as independent value prediction (which must construct 12 regression models).

3.4 Smoothness of Prediction

Another factor to consider is the influence of noise on the prediction approaches. To do this, we conduct an experiment using a simple, stationary time series, i.e., white noise, as shown in Figure 5. Figure 6 shows that multi-stage prediction tends to smooth out the time series to its mean value after *p* time steps. Such smoothing effect is not present in independent value prediction, which makes spurious predictions around the mean, because the prediction at each time step is modeled independently. This method may suffer from overfitting as it tries to capture the fluctuations of the noise time series. For parameter prediction, the best fit model of the data is found to be a polynomial of degree zero. Even though the parameters are predicted independently, the smoothness of the time series is guaranteed by the parametric function used to fit the output vector.

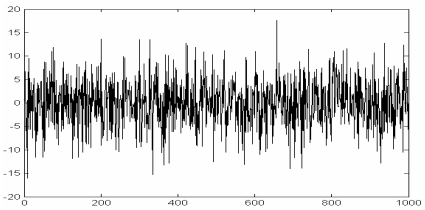

Fig. 5. White Noise WN(0,0.5)

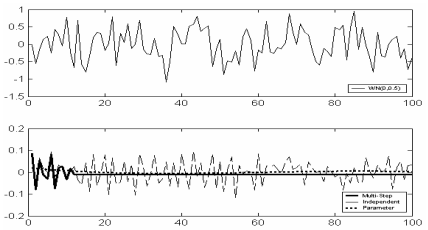

Fig. 6. Predicting Results (p=12,h=100,d=6)

3.5 A General Comparison

Finally, we apply the three prediction approaches to 21 real data sets to compare their relative performance. The RMSE value for each data set is obtained by 10-fold cross validation. The size of the prediction window is set to *h=24.* Table 2 summarizes the RMSE for the three prediction approaches using MLR as the underlying regression method. Their relative performance is summarized in Table 3 in terms of the number of wins, draws and losses. We also test the significance of the difference using paired t-significance test. The result shows that the observed difference between the RMSE

of multi-stage and independent value prediction is not that significant. However, the performance of parameter prediction is significantly worse than independent value prediction. This is because MLR may not be suitable to fit the parameters of the function, which have nonlinear relationships with the time series values.

Table 2. Multiple Linear Regression

	Multi-stage	**Independent**	**Parameter**
milk	0.0733	0.0705	0.0918
Temp.	0.2776	0.2959	0.2936
PET	0.0419	0.0414	0.0619
PREC	0.0310	0.0317	0.0534
Solar	0.1218	0.1236	0.2132
appb	0.2974	0.3804	0.3803
appd	0.2152	0.2395	0.2766
appf	0.3147	0.2445	0.2991
appg	0.8642	0.9343	0.9218
deaths	0.7309	0.5560	0.5633
lead	0.4195	0.4207	0.4206
sales	0.3187	0.3637	0.3637
wine	0.2738	0.2902	0.3209
seriesc	0.9359	0.9845	0.9845
odono	0.4226	0.4731	0.4712
qbirth	0.5450	0.4793	0.5231
Bond2	0.5226	0.5886	0.5884
Daily	0.2006	0.2137	0.2137
food	0.1995	0.1929	0.1950
treerin	0.8929	0.8807	0.8818
pork	0.9462	0.7948	0.7918

Table 4. RNN

	Multi-stage	**Independent**	**Parameter**
Milk	0.0733	0.0705	0.0918
Temp.	0.2776	0.2959	0.2936
PET	0.0419	0.0414	0.0619
PREC	0.0310	0.0317	0.0534
Solar	0.1218	0.1236	0.2132
Appb	0.2974	0.3804	0.3803
Appd	0.2152	0.2395	0.2766
Appf	0.3147	0.2445	0.2991
Appg	0.8642	0.9343	0.9218
Deaths	0.7309	0.5560	0.5633
Lead	0.4195	0.4207	0.4206
Sales	0.3187	0.3637	0.3637
Wine	0.2738	0.2902	0.3209
Seriesc	0.9359	0.9845	0.9845
Odono	0.4226	0.4731	0.4712
Qbirth	0.5450	0.4793	0.5231
Bond2	0.5226	0.5886	0.5884
Daily	0.2006	0.2137	0.2137
Food	0.1995	0.1929	0.1950
Treerin	0.8929	0.8807	0.8818
Pork	0.9462	0.7948	0.7918

Table 3. Win-Draw-Loss results for MLP

	Multi vs Indep	Multi vs Param	Indep vs Param
0.01diff	10-6-5	14-2-5	8-12-1
T value	0.1496	0.8761	2.7299
P value	0.8826	0.3914	0.0129

Table 5. Win-Draw-Loss results for RNN

	Multi vs Indep	Multi vs Param	Indep vs Param
0.01 diff	7-0-14	5-2-14	13-1-7
T value	2.6396	3.3884	0.3012
P value	0.0157	0.0029	0.7664

Tables 4 and 5 show the results using RNN as the underlying regression method. Observe that the independent value and parameter prediction approaches perform significantly better than multi-stage prediction ($p < 0.05$). For multi-stage prediction, the RMSE for RNN is higher than the RMSE of MLR in 10 out of 21 data sets, which suggests the possibility of model overfitting when using a flexible regression method such as RNN. Nevertheless, we still find 17 data sets in which independent value prediction with RNN outperforms all the prediction approaches using MLR and 12 data sets in which parameter prediction with RNN outperforms all the prediction approaches using MLR. This result suggests that, using nonlinear regression methods

such as RNN, the independent value and parameter prediction approaches may achieve better performance than multi-stage prediction. Moreover, for parameter prediction, most of the data sets require $d < 5$, which makes it more efficient to build compared to independent value prediction (which requires building $h = 24$ models).

4 Conclusions

In this paper, we conduct an empirical study on three prediction approaches for solving multistep-ahead time series prediction problems. The tradeoffs among these approaches are studied using both real and synthetic data sets. Our experimental results show that multi-stage prediction tends to suffer from error accumulation problems when the prediction period is long. This is because the bias and variance from previous time steps are propagated into future predictions. Independent value prediction is less susceptible to this problem because its predictions are made independently at each time step. However, it has difficulty in learning the true model because the function to be modeled becomes more complex with increasing time steps. This approach also does not smooth out the effect of noise unlike multi-stage prediction. Parameter prediction handles noisy data by fitting a function over the entire output sequence while alleviating the error accumulation problem by making independent predictions. It also tends to be more efficient than independent value prediction when the number of parameters to be fitted is small. However, finding the appropriate parameter function to fit the time series can be quite a challenging task. Finally, we observe successful applications of both independent value and parameter prediction approaches when applied to real data sets using RNN.

5 References

1. Gershenfeld N. A. and Weigend A. S.: The Future of Time Series. In "Time Series Prediction: Forecasting the Future and Understanding the Past",(1993) 1-70
2. Jones R. H.: Maximum like lihood fitting of ARMA models to time series with missing observations. Technometrics 20, (1980) 389–395.
3. Giles C.L., Lawrence S. and Tsoi A.C.: Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference. Machine Learning, 44(1-2) (2001) 161-183.
4. Elman J.L.: Distributed Representations, Simple Recurrent Networks, and Grammatical Structure. Machine Learning, 7 (2/3) (1991) 195–226.
5. Kennel M. B., Brown R., and Abarbanel H. D. I.: Determining embedding dimension for phase-space reconstruction using a geometrical construction. Phys. Rev. A 45 (1992) 3403
6. Hyndman R., Time Series Data Library. http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/
7. Joseph Rynkiewicz: Hybrid HMM/MLP models for time series prediction. Proc of the European Symposium on Artificial Neural Networks Brugges, Belgium,(1999). 455-462.
8. Akaike H.: Fitting autoregressive models for prediction. Annals of the Institute of Statistical Mathematics, 21 (1969) 243-247
9. UCI Machine Learning Repository http://www.ics.uci.edu/~mlearn/MLRepository.html
10. Y. Le Borgne.: Bias variance trade-off characterization in a classification. What differences with regression? Technical Report N°534, ULB, (2005).