Imperial College London

MSc in Applied Computational Science and Engineering

Independent Research Project

Project Report

A Machine Learning Approach to the Prediction of Tidal elevation

by

Chen Qian

chen.qian19@imperial.ac.uk

Supervisors:

Professor Matt Piggott

August 28th, 2020

本项目报告系钱宸原创，剽窃必究！

Abstract

Accurate tidal prediction is of great significance for the effective generation of energy. This report proposes one prediction model combining periodic analysis and neural network, which is called back propagation (BP). By learning the correlation between training samples and training labels, the BP neural network can determine the fitting weight coefficients of the prediction curve. In the section of performance analysis, consider using different timescales, different tidal observation sites, and different types of BP neural network methods to measure success. To further quantify performance, the correlation coefficient is introduced as the measurement indicator. The closer the correlation coefficient is to 1, the better the prediction accuracy of the trained neural network will be. Experimental results demonstrate that the trained BP neural network model can achieve good prediction accuracy under small timescale and observation points with regular fluctuations.

Keywords: Tidal prediction; Periodic analysis; BP neural network; Correlation coefficient

版权归chen qian 本项目报告仅供学习，禁止复制和商业使用

1. Introduction

Tide is a periodic wave phenomenon produced by the gravitational force of celestial bodies. It is possible to predict in advance when high tide and low tide will occur by analysing the tidal phenomenon. Tidal energy generated during tidal fluctuations is a broad and reliable energy source, one of whose practical applications is to generate electricity (Rourke et al., 2010). Therefore, good tide prediction plays a crucial role in the efficient generation of energy.

Until now, traditional tidal prediction models have been widely put into practical application. For example, harmonic analysis, finite difference method, statistical method, Kalman filters, and so forth. However, each of the methods above has its limitations, which have been stated in the literature. The harmonic analysis could not use short-term recorded observation data to determine the required constituent tidal parameters (Doodson, 1921). It was too complex to deal with nonlinear problems by the algorithm of finite difference method (Weisse et al., 2014). The statistical methods could only deal with discrete problems (Paul et al., 2018). Regarding Kalman filters, since the measured harmonic parameter data was limited, so it could be only used as short-term forecasting (Yen et al., 1996). Besides, as was illustrated in the literature (Consoli et al., 2014), especially for prediction models as harmonic analysis and Kalman filters, these two methods will be invalid for supplementing the missing data when the data of the observation periods were lost or incomplete. Based on this, it is vital to find novel techniques that can accurately forecast the tidal level.

The artificial neural network is a complex nonlinear information processing system developed in the view of brain neural structure. Anderson and McNeill (1992) stated in his report that in 1943, one neurophysiologist called Warren McCulloch and one mathematician called Walter Pitts used circuits to simulate simple neural networks, which demonstrated how neurons work. This marked a milestone in the start of artificial neural networks. Since then, the neural networks have developed rapidly, and have been put into application in many fields such as prediction. Taormina et al. (2012) used neural networks to predict the groundwater level. Nayak et al. (2013) applied neural networks to forecast rainfall. (Elsafi, 2014) applied neural networks for monitoring flood hazards in the River Nile.

Artificial neural networks are also applied to forecast tidal levels. At present, there are several mixed approaches, which are based on traditional neural networks. For example, neural network and wavelet analysis (Chen et al., 2007), neural network and genetic algorithm (Ghorbani et al., 2010), neural network and finite element analysis (Yasseri et al., 2010). In addition to traditional neural networks, there is also a type of neural network called back propagation (BP). Yuan et al. (2015) illustrates that when it is combined with typhoon parameters, this approach can effectively make accurate

predictions and prepare for the reduction of typhoon disasters. Despite the predictive ability of this method is remarkable, it is not easy to obtain non-astronomical components. In some ways, it is difficult to train such a neural network model without adequate parameters. Based on it, from the perspective of simplification, attempt to directly apply BP neural network for prediction, which selects the actual tidal levels as input data. Besides, considering the tidal fluctuation phenomenon, periodic analysis can be used to study the periodicity of tidal data, which helps to train the BP neural network later and improve prediction accuracy.

This report is organized to build a prediction model which combines the periodic analysis with BP neural network, selecting tidal elevation as input data. Software & description, implementation & code, performance analysis, discussion & conclusion will be demonstrated in order from second section to fifth section.

版权归chen qian

2. Software & Description

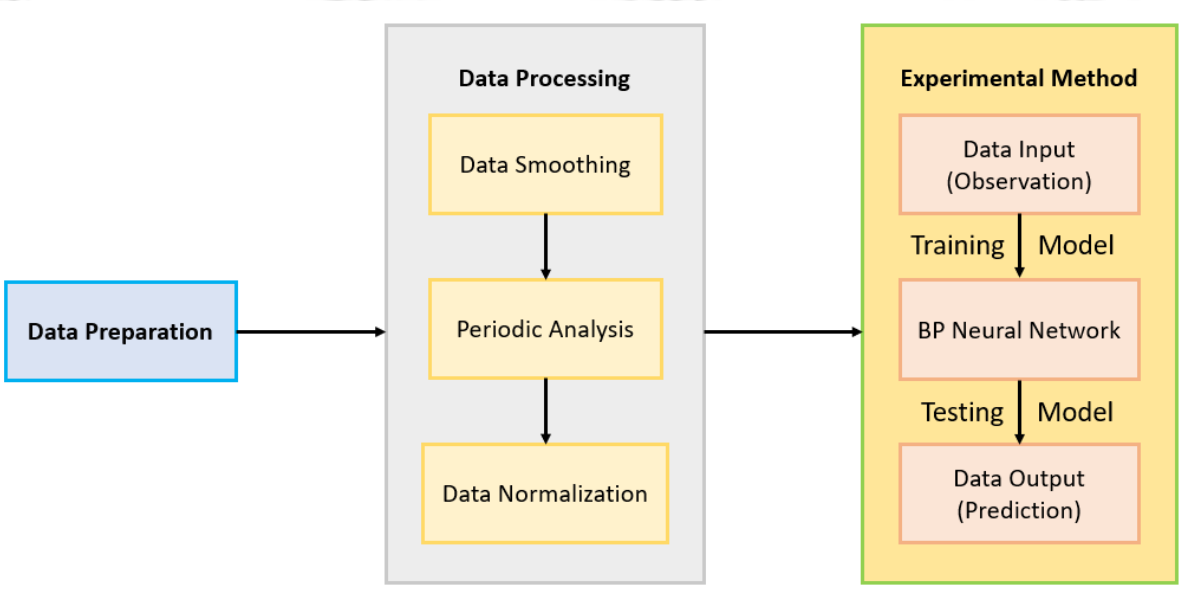

Figure 1: Software Architecture, which consists of data preparation, data processing and experimental method.

As shown in Figure 1, the software consists of several parts, which can be divided into three types: data preparation, data processing, and experimental method.

For data preparation, mainly summarized as reading the data in the form of a text file and storing it in the form of a CSV file.

Regarding data processing, firstly smooth the data which is stored in the form of a CSV file, then obtain the periodicity of data by periodic analysis, afterwards normalize the sample data which is required to be researched later.

Concerning the experimental method, take the normalized sample data as input data, training it under the built BP neural network framework, and then select suitable historical data as the starting data for prediction. As a result, the normalized prediction data will be obtained. Finally, convert the normalized prediction data to normal prediction data.

2.1. Methodology

i. Data Preparation:

The data are provided by UK Tide Gauge Network, which is from the British Oceanographic Data Centre. The frequency of the raw data is 15 minutes. Download relevant data as required, and then read the downloaded data in the form of a text file, afterwards storing it in the form of a CSV file.

ii. Data Processing:

Select required observation data in the CSV file for data smoothing, which means eliminating abnormal values (too large or too small values) in the data series and filling other appropriate values. After that, it will be more obvious to display the trend of the observation data. Meanwhile, the approximate period of the tidal data can be observed and determined by periodic analysis, then select data series sample and normalize it.

$$\bar{Y}_i = \frac{Y_i - Y_{min}}{Y_{max} - Y_{min}}$$

$\bar{Y}_i$: Tidal values after normalization Y_i : Tidal values before normalization

Y_{max} : Maximum tidal value in sample Y_{min} : Minimum tidal value in sample

iii. Experimental Method:

Build a suitable BP neural network model, which is first proposed by Rumelhart et al. (1986). As shown in Figure 2, It consists of three layers by default here: input, hidden and output layer. The characteristic of this method is that it can approximate any nonlinear continuous function (Hornik, 1991).

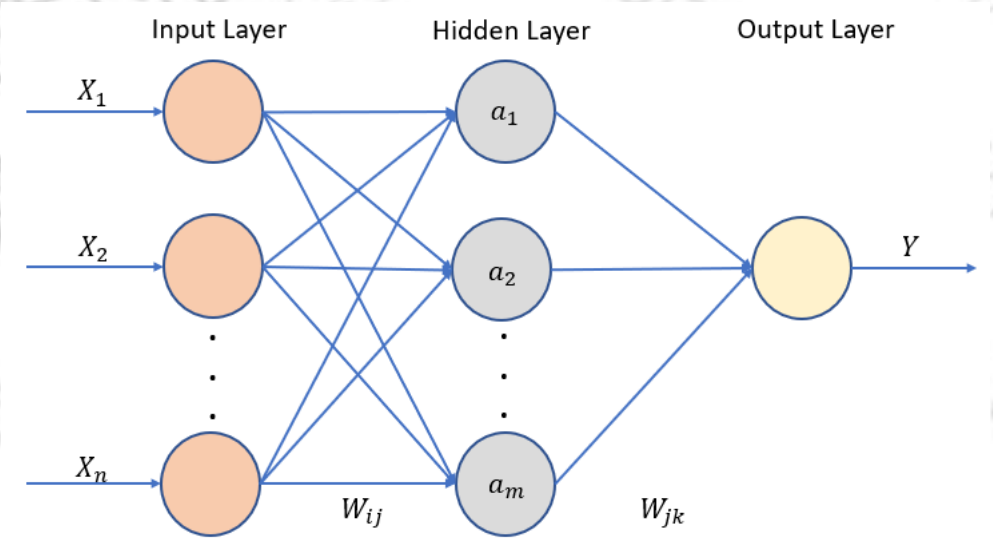

Figure 2: A classical three-layer BP neural network model, which includes n input, m hidden and *1* output neurons.

The following is the procedure of BP neural network implementation:

1. Randomly assign connection matrix weights and define correction matrix weights, which is the same as the connection matrix weights at the beginning.

   W_{ij} / C_{ij} : Connection / Correction matrix weights from input layer i to hidden layer j

   W_{jk} / C_{jk} : Connection / Correction matrix weights from hidden layer j to output layer k

2. Determine the input data as training samples and the actual output data as training sample labels.

   Training samples : $x_n, x_{n-1}, x_{n-2}, \dots, x_{n-m+1}$ (totally m data)

   Training sample labels : x_{n+k} (k steps after data x_n, $k \geq 1$)

3. Calculate the predicted output data and select the sigmoid function as the nonlinear activation function during training the BP neural network.

   Sigmoid function : $f(x) = \frac{1}{1+e^{-x}}$

$$O_j = f\left(\sum_i W_{ij} O_i\right) = \frac{1}{1 + e^{-\sum_i W_{ij} O_i}}$$

$$O_k = f\left(\sum_j W_{jk} O_j\right) = \frac{1}{1 + e^{-\sum_i W_{jk} O_j}}$$

   O_i : Activation vector at input layer i

   O_j : Activation vector at hidden layer j

   O_k : Activation vector at output layer k

4. Adjust the connection matrix weights based on the error function, which is the difference between the training sample labels and the actual output data.

Error function : $E = \frac{1}{2}\sum_k (T_k - O_k)^2$

T_k : Training sample labels O_k : Actual output data

The error of output layer is defined as:

$$\delta_k = f'(O_k)(T_k - O_k) = O_k(1 - O_k)(T_k - O_k)$$

The error of hidden layer is defined as:

$$\delta_j = f'(O_j)\sum_k \delta_k W_{jk} = O_j(1 - O_j)\sum_k \delta_k W_{jk}$$

In order to speed up the efficiency of updating connection matrix weights, one mechanism to correct the matrix weights is introduced, which are defined as correction matrix weights C_{jk} and C_{ij}. C_{jk} and C_{ij} load the values $\delta_k O_j$ and $\delta_j O_i$ respectively in the previous step of back-propagation. The modified connection matrix weights formula is defined as (Rumelhart et al., 1986; Jacobs, 1988):

$$W_{jk}^{t+1} = W_{jk}^{t} + \eta(\delta_k O_j)^t + \alpha C_{jk}^{t} \qquad C_{jk}^{t+1} = (\delta_k O_j)^t$$

$$W_{ij}^{t+1} = W_{ij}^{t} + \eta(\delta_j O_i)^t + \alpha C_{ij}^{t} \qquad C_{ij}^{t+1} = (\delta_j O_i)^t$$

t / t + 1 : Current / Next training steps

η : Learning rate α : Momentum factor

5. Apply BP neural network into training samples and labels, which determines connection matrix weights. Based on it, the BP neural network can be used for predicting tidal elevation. As is shown in Figure 3, traditional prediction methods mainly include one-step and multi-step iterative prediction. The detailed implementation is to firstly select a suitable historical data sample $\{x_n, x_{n-1}, x_{n-2}, \dots, x_{n-m+1}\}$ and then backtrack the results of prediction data $\{x_{n+k}, k \geq 1\}$ to the BP neural network as the input data for the next prediction.

Prediction Method	Historical Data Sample	Prediction data
One-step iterative prediction	$\{x_n, x_{n-1}, x_{n-2}, \dots, x_{n-m+1}\}$ (totally m data)	$\{x_{n+k} \mid k = 1\}$
Multi-step iterative prediction	$\{x_n, x_{n-1}, x_{n-2}, \dots, x_{n-m+1}\}$ (totally m data)	$\{x_{n+k} \mid k \geq 2\}$

Figure 3: Two types of traditional prediction methods, which are One-step and Multi-step iterative prediction. Select appropriate historical data to forecast data $\{x_{n+k} \mid k \geq 1\}$.

Concerning one-step iterative prediction, one new improved prediction method is introduced based on it. As is illustrated in (Box et al., 2015), the model's name is autoregressive moving average model (ARMA), which can be defined as:

$$x_t = \beta_1 x_{t-1} + \beta_2 x_{t-2} + \dots + \beta_p x_{t-p} + \varepsilon_t + \alpha_1 \varepsilon_{t-1} + \alpha_2 \varepsilon_{t-2} + \dots + \alpha_q \varepsilon_{t-q}$$

x_t : Observation value ε_t : Residual value

β_t / α_t : Weight parameters of observation / residual value

p / q : The order of the autoregressive / moving average model

2.2. Code Metadata

2.2.1. Platform

Python 3 is the main language for software development. It is a high-level scripting language that combines interpretation, compilation, interactivity and object-oriented. In addition, the Jupyter Notebook is also introduced. This is a web application that permits users to combine text, mathematical equations, codes and visual content into an easy-to-share document, which means that users can concentrate on explaining the entire analysis process to others. Based on it, the Jupyter Notebook has rapidly become an essential tool for data analysis and machine learning.

2.2.2. Libraries

1. NumPy
   NumPy is an open-source numerical computing toolkit for python. It includes several aspects: Powerful N-dimensional array objects; Full-featured function libraries; Practical linear algebra, Fourier transforms and random number generation functions.

2. Pandas
   Pandas is a data analysis package of python, which is a prevailing tool for analysing structured data. This tool is based on NumPy (supporting high-performance matrix operations). The main data structures of Pandas are ***Series*** (1-dimensional data) and ***DataFrame*** (2-dimensional data).

3. Matplotlib
   Matplotlib is a plotting library of python, which can be regarded as a visual operation interface of NumPy. It generates high-quality graphics in various hardcopy formats and a cross-platform interactive environment. Matplotlib supports a variety of graphics. For example, histograms, pie charts, scatter plots, polar plots and 3-dimensional plots.

版权归chen qian

3. Implementation & Code

3.1. Data Preparation

```
def read_data ( ):
    Read data from text file into lists.
    Create three lists.
    Open text file.
    Read the required data and append it to the created lists.
    return list
```

```
def list_transform_csv ( ):
    Transform lists into CSV.
    Slice three lists according to selected timescale.
    Define name and data and append lists to the DataFrame.
    return CSV
```

	Cycle	Hour	Elevation
0	7.0	1.5	6.266
1	13.0	3.0	3.809
2	19.0	4.5	3.072
3	25.0	6.0	4.245
4	31.0	7.5	6.426
...	...	...	...
444	2671.0	667.5	9.001
445	2677.0	669.0	11.012
446	2683.0	670.5	10.344
447	2689.0	672.0	7.608
448	2695.0	673.5	4.457

449 rows × 3 columns

	Cycle	Hour	Elevation
0	7.0	1.5	5.584
1	13.0	3.0	3.400
2	19.0	4.5	2.885
3	25.0	6.0	4.534
4	31.0	7.5	7.186
...	...	...	...
444	2671.0	667.5	9.273
445	2677.0	669.0	10.749
446	2683.0	670.5	9.163
447	2689.0	672.0	5.945
448	2695.0	673.5	2.883

449 rows × 3 columns

Figure 4: Tidal elevation data are in February and March 2020 respectively. The frequency of the raw data are 15 minutes. In this sample, their timescales are all 1.5 hours.

The purpose of the above ***read_data*** and ***list_transform_csv*** function together is to convert the data read from the text file into ***DataFrame*** format. Figure 4 displays the tidal elevation data measured by one tidal observation site in February and March 2020 respectively, whose timescales are all selected as 1.5h.

This type of data is a time series array of three columns. The first and second columns are the index and hour from the start date of the data respectively. The third column is the elevation data in meters relative to the average water level.

3.2. Data Processing

```
def data_smooth ( ):
    Make the data curve smooth.
    Define the difference series.
    Define the maximum and minimum normal value.
    Define a value which is greater than maximum normal value or
        less than minimum normal value as an abnormal value.
    Use loops and conditional statements to find all the abnormal
        values, then fill abnormal values with appropriate values.
    return data
```

```
def data_normalization ( ):
    Normalize the input data.
    Create normalization data list.
    Define normalization formula.
        Normalized data = (Raw data – Minimum raw data) /
                          (Maximum raw data – Minimum raw data)
    return normalized data
```

According to the above ***data_smmoth*** function, the detailed procedure is to read the raw data firstly, and then take the difference value of each current and previous point on the raw data as a new series, afterwards find all the abnormal points which indicate outrageous changes and delete them, subsequently fill abnormal values with appropriate values. After data smoothing, it will be more obvious to display the trend

of data sample changing over time, then use periodic analysis to obtain the periodicity of the observation data.

As shown in Figure 5 below, it illustrates the fluctuation of the selected testing sample data is regular, which can be regarded as a combination of several sine or cosine functions. Given this type of observation data, sub-intervals are chosen, which can be divided into period 1 and period 2. The data from the first period can be approximately considered the same as the second period by observation. Based on it, it is estimated that the periodicity of this testing sample data is 15 to 16 days (around half a month). Following this, select appropriate data for normalization by ***data_normalization*** function. The above data processing procedures are fully prepared for training BP neural networks later.

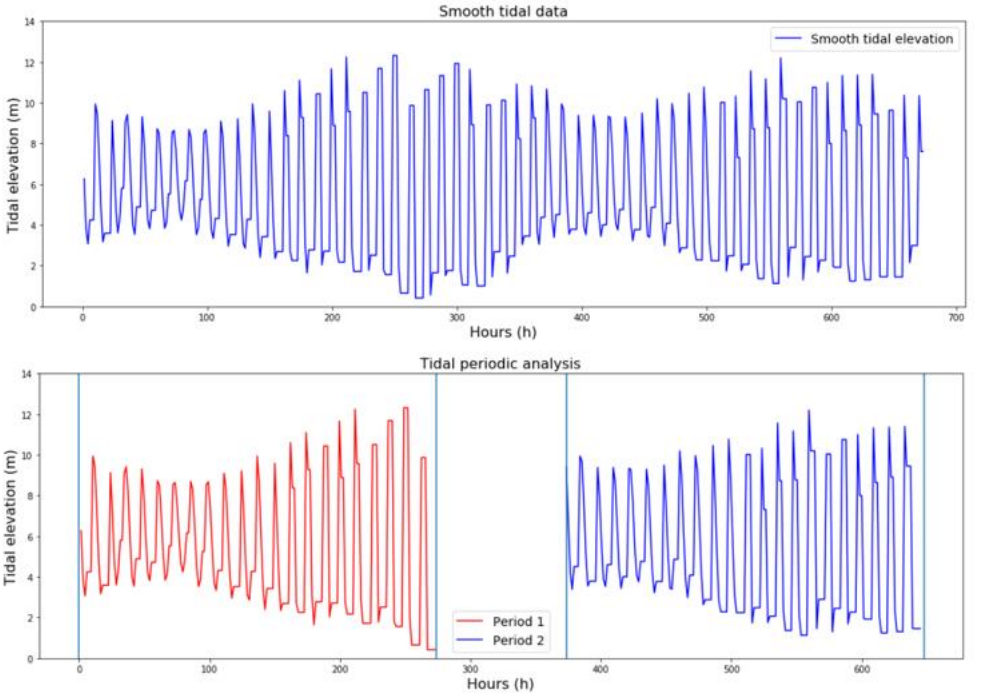

Figure 5: Data smooth and Periodic analysis, which determines the periodicity of this sample data is around half a month.

3.3. Experimental Method

```
def create_sample_and_label ( ):
    Create training samples and sample labels.
    Read normalized data.
    Define training samples and sample labels.
    return training samples, sample labels
```

```
class BP_Neural_Network:
    Back propagation neural network algorithm.
    def __init__ (self):
        Create activation vectors and layer neurons.
        Create connection and correction matrix weights.
    def sigmoid (self):
        Define sigmoid function: 1 / (1 + e^(-x)).
    def sigmoid_derivative (self):
        Define sigmoid derivative function: sigmoid * (1 - sigmoid).
    def rand_interval (self):
        Define random number.
    def weights_matrix (self):
        Create weights matrix.
    def initial_setup (self):
        Define initial connection and correction matrix weights.
    def predict (self):
        Define feed forward neural network.
    def back_propagate (self):
        Call predict function.
        Backtrack and adjust connection matrix weights layer by layer
    def training_set (self):
        Call back_propagate function.
        Calculate the global error of the training set.
        Stop when the convergence condition is met.
    def test_set (self):
        Call initial_setup and training_set function.
        Use learning samples for testing.
```

Follow the ***create_sample_and_label*** function, training samples and sample labels are created, which can be used for training BP neural network later. Afterwards, it is time to start building the BP neural network, whose detailed pseudo-code flow chart is shown above.

Firstly, define ***BP_Neural_Network*** as a class. Secondly, declare one special function (construction method) and the other nine general methods. The basic framework of BP neural networks has been created, which contains neurons, layers, activation vectors, activation function and matrix weights.

In general, training BP neural network consists of two steps: The first is feed-forward prediction, which means forwarding the training samples as input information layer by layer. The second is back propagation. According to the error between the training sample labels and the actual output data, the connection matrix weights are modified from the output layer to the input layer.

```
def iteration_prediction_list ( ):
    Use historical samples to do prediction.
    Read historical data.
    Use the trained BP neural network to iterate feed-forward prediction
        and append the prediction values to the historical data.
    return prediction data, length of prediction data
```

```
def data_transfer ( ):
    Transfer normalized data into actual data.
    Create actual data list.
    Define transfer formula.
        Actual data = (Normalized data * (Maximum normalized data -
                      Minimum normalized data) + Minimum
                      normalized data)
    return actual data
```

After training BP neural network, the connection matrix weights will be loaded, which can be used for feed-forward prediction later. According to ***iteration_prediction_list*** function, firstly select the appropriate historical data, and then call the trained BP neural network to use iteration for prediction. There are two main forecasting methods, named one-step and multi-step iterative prediction.

Given that the prediction data obtained is based on the normalization system, which means that these values are in the range of zero to one. Therefore, it is essential to convert the obtained prediction values to the original system. This can be carried out by ***data_transfer*** function.

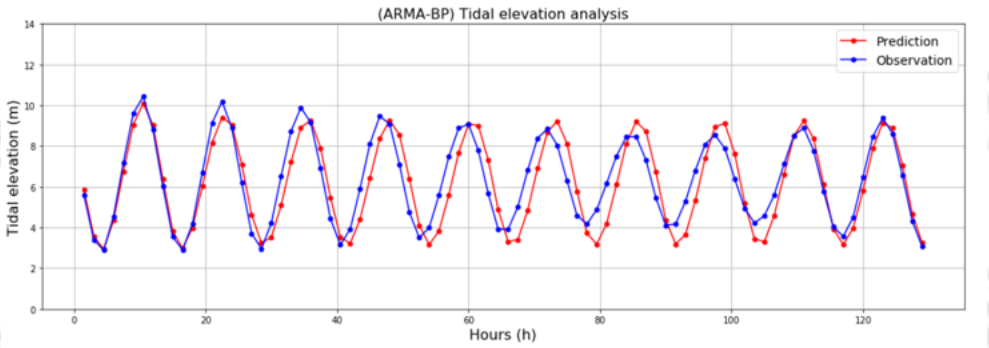

Figure 6: ARMA-BP neural network prediction sample, which selects tidal data in February as the training samples and labels and then apply it for prediction in March.

Figure 6 demonstrates the experimental result which combines the autoregressive moving average model (ARMA) and BP neural network method for prediction. Given the previous periodic analysis, it is estimated that the periodicity of this testing sample data is around half a month. In this testing sample, tidal elevation data measured by one tidal observation site in February is selected as the training samples and training sample labels. Following this, apply suitable historical data into the trained BP neural network for prediction in March. Finally, make a comparison between the prediction value and the observation value.

As far as training BP neural network is concerned, it is crucial to choose appropriate parameters. In the above testing sample, the selected parameters are as follows: twelve input neurons, six hidden neurons and one output neuron. The learning rate η is equal to 0.1 and the momentum factor α is equal to 0.85. Besides, the residual value ε_t of the ARMA model is selected as 0.001.

4. Performance Analysis:

In order to measure the success, consider the following aspects where the factors affecting the accuracy of the BP neural network.

1. Different timescales
2. Different tidal observation sites
3. Different types of BP neural network methods

In terms of detailed implementation procedure, three different tidal observation sites are selected, namely *Hinkley Point*, *St. Helier* and *Bournemouth* respectively. The former two have relatively large peaks and trough tidal elevation while the latter one is the opposite. Regarding timescale, 1h and 1.5h are chosen for comparison. Besides, two types of BP neural networks are introduced for prediction, mainly called one-step and multi-step iterative prediction. Concerning the former one, a representative ARMA model is introduced.

As is shown below, the prediction value obtained by the trained BP neural network in Figure 7 and Figure 8 is relatively close to the observation value, regardless of the *ARMA-BP* or the *Non-ARMA-BP* method. However, the prediction accuracy of the trained BP neural network displayed in Figure 9 is much worse, especially for the *Non-ARMA-BP* method. When the timescale is selected as 1.5h, the trend of the prediction value is relatively consistent with the observation value at first, but then the prediction value gradually turns to be flat and hardly fits the observation value eventually.

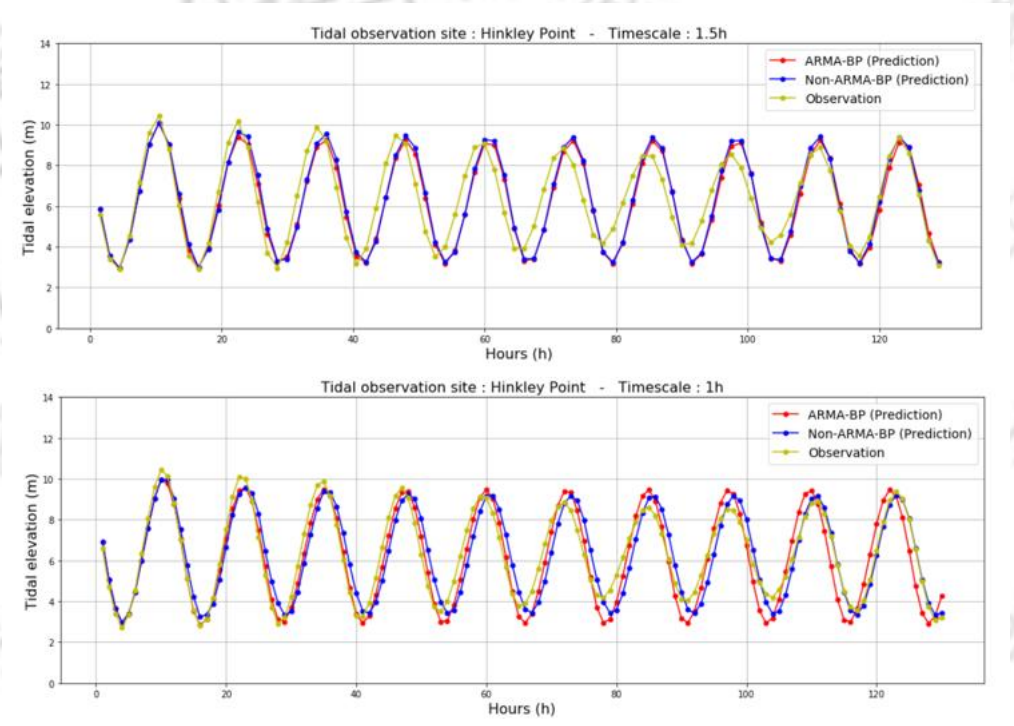

Figure 7: Tidal prediction using different timescales (1.5h / 1h) and different prediction methods (ARMA-BP / Non-ARMA-BP) in tidal observation site (Hinkley Point).

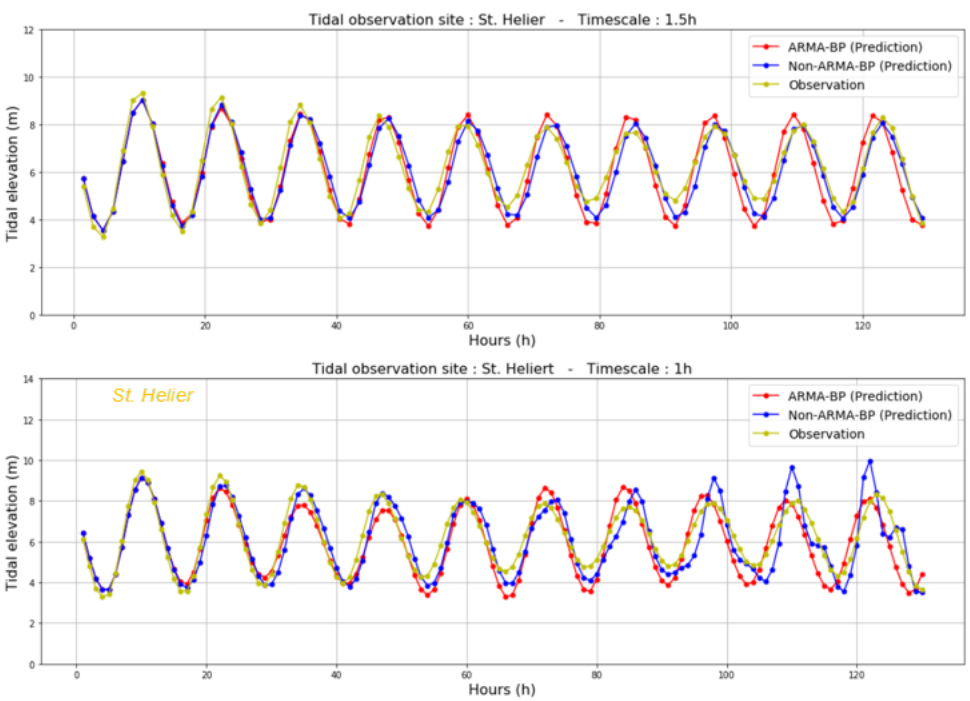

Figure 8: Tidal prediction using different timescales (1.5h / 1h) and different prediction methods (ARMA-BP / Non-ARMA-BP) in tidal observation site (St. Helier).

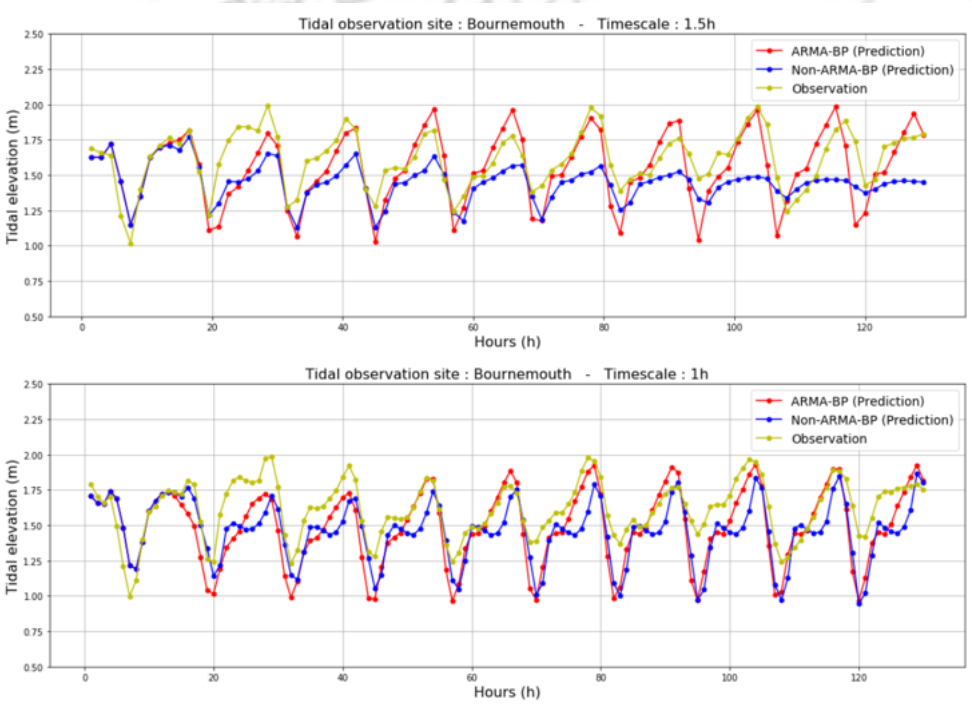

Figure 9: Tidal prediction using different timescales (1.5h / 1h) and different prediction methods (ARMA-BP / Non-ARMA-BP) in tidal observation site (Bournemouth).

To further quantifying performance, one new measurement indicator is introduced, calling the ***correlation coefficient***.

$$r(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var[X]\,Var[Y]}} = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})\cdot\sum_{i=1}^{n}(Y_i-\bar{Y})}}$$

X_i : Prediction value Y_i : Observation value

$\bar{X}$: Mean prediction value $\bar{Y}$: Mean observation value

r : Correlation coefficient Cov : Covariation Var : Variance

$r(X,Y)$ indicates the closeness of a linear relationship between X and Y. The larger $r(X,Y)$ is, the more interrelated X and Y are. See the ***correlation_coefficient*** function below for implementation details.

```
def correlation_coefficient ( ):
    Calculate correlation coefficient between prediction and observation.
    Define Cov(x,y) / Var[x] / Var[y].
    Define correlation coefficient formula.
        correlation coefficient = Cov(x,y) / sqrt( Var[x] * Var[y] )
    return correlation coefficient
```

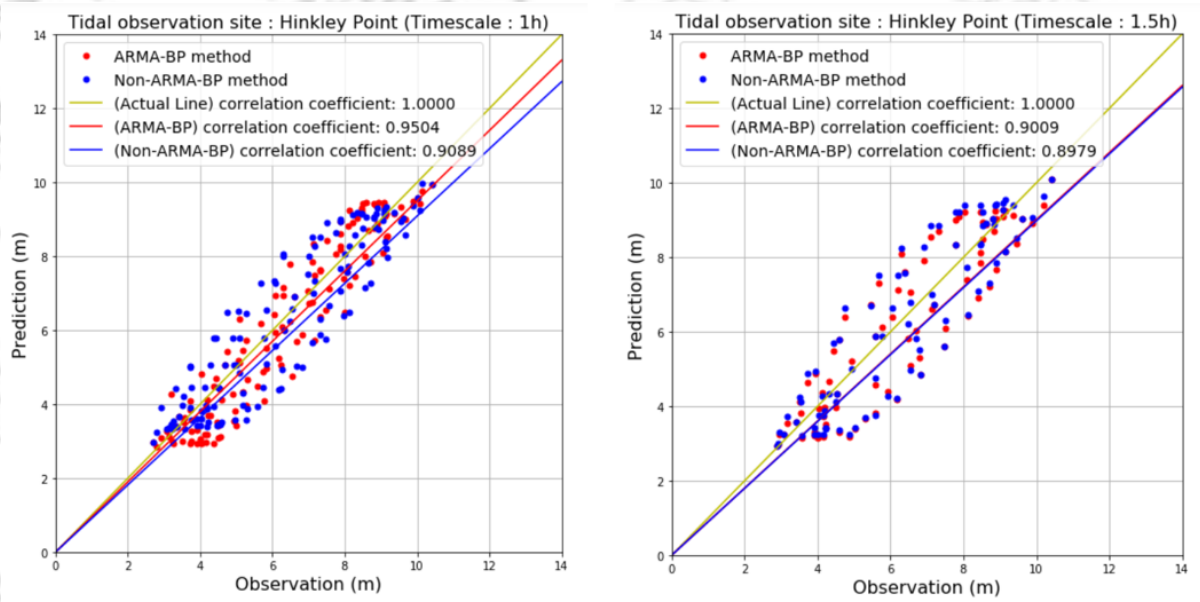

Figure 10: Comparison of correlation coefficients using different timescales (1.5h / 1h) and different prediction methods (ARMA-BP / Non-ARMA-BP) in tidal observation site (Hinkley Point).

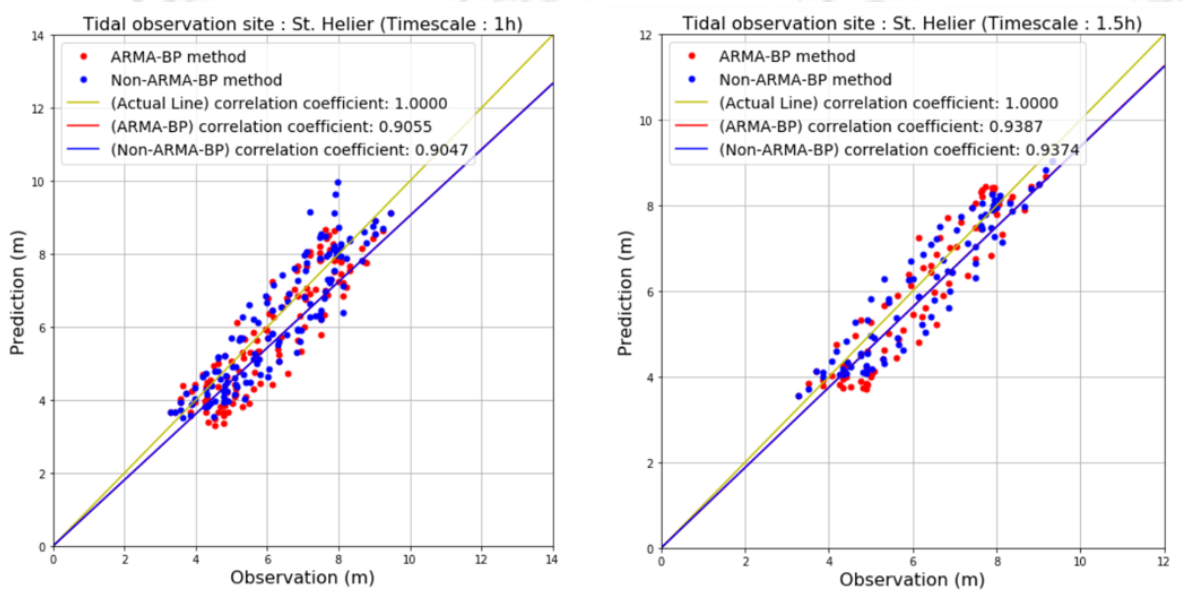

Figure 11: Comparison of correlation coefficients using different timescales (1.5h / 1h) and different prediction methods (ARMA-BP / Non-ARMA-BP) in tidal observation site (St. Helier).

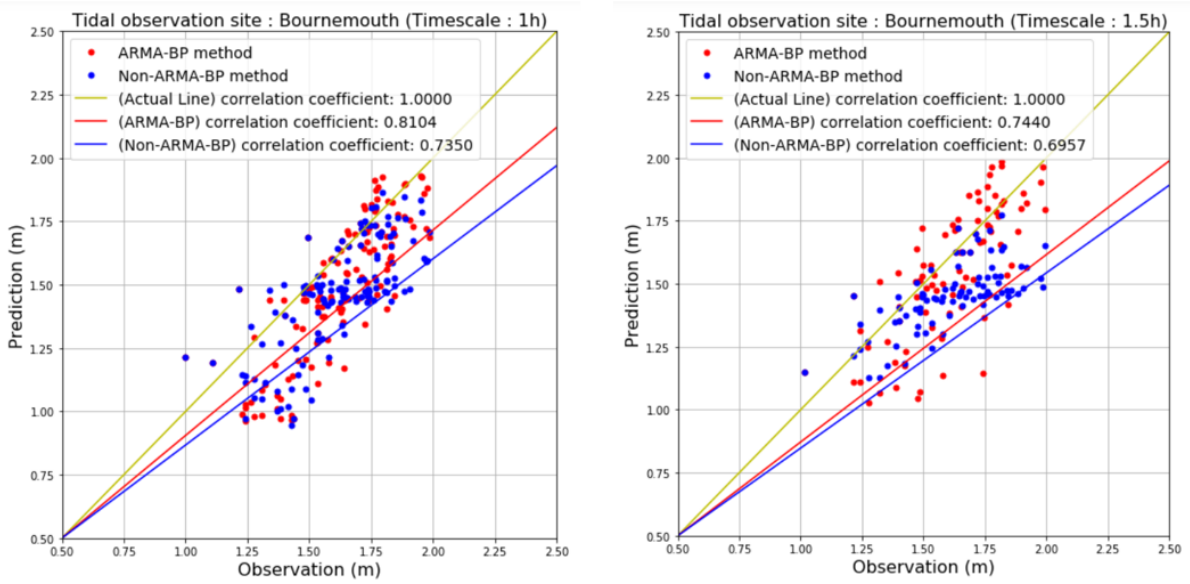

Figure 12: Comparison of correlation coefficients using different timescales (1.5h / 1h) and different prediction methods (ARMA-BP / Non-ARMA-BP) in tidal observation site (Bournemouth).

In general, *ARMA-BP* (one-step iterative prediction) method dominates over the *Non-ARMA-BP* (multi-step iterative prediction) method, either on different tidal observation sites or timescales.

As shown in Figure 10, for a smaller timescale, the trained BP neural network can make a more accurate prediction. In terms of the correlation coefficient, by adjusting the timescale (from 1.5h to 1h), the accuracy of the *ARMA-BP* method has been

significantly improved, from 0.9009 to 0.9504. While the accuracy of the *Non-ARMA-BP* method does not change much, which remains at around 0.9.

Figure 11 illustrates in one tidal observation site named *St. Helier*, reducing the timescale might not enhance the predictive ability of the trained BP neural network. During the timescale adjustment, the prediction accuracy of the *ARMA-BP* method is almost the same as the *Non-ARMA-BP* method. When the timescale is 1.5h, the correlation coefficients of both methods are around 0.94. Similarly, when the timescale is 1h, the correlation coefficients of both methods are approximately 0.9.

Concerning Figure 12, to be honest, in this tidal observation site, the predictive ability of the trained BP neural network is quite terrible, regardless of different timescales or prediction methods. The accuracy displayed by the figure is far inferior to the previous two tidal observation sites. Besides, by adjusting the time scale (from 1.5h to 1h), although the prediction accuracy is still not very ideal, whose correlation coefficient is far below 0.9, the predictive ability of both methods has been greatly enhanced. Especially for the *ARMA-BP* method, when the timescale is 1h, its correlation coefficient has exceeded 0.8.

版权归chen qian

5. Discussion & Conclusion

5.1. Discussion

Take the three samples in the section of performance analysis into consideration, then discuss why the prediction accuracy brought by the same method (*ARMA-BP* or *Non-ARMA-BP*) will be diverse when it is applied to different tidal observation sites or timescales. Overall, the reason why the *ARMA-BP* method dominates over the *Non-ARMA-BP* method is that the residual value ε_t is introduced. In some ways, the additional fitting weight parameters provided by the smaller residual values help correct the curve fitting rate.

Firstly, in terms of the tidal observation site named *Hinkley Point*, the fluctuation of the selected training sample is regular and can be regarded as a function of sine or cosine. Besides, there are larger peak values and lower trough values, which means larger amplitudes. Meanwhile, the distribution of observation points is relatively uniform, neither too dense nor too sparse. By adjusting the timescale (from 1.5h to 1h), under the same training and prediction time, although the number of the observation points increases, the data distribution has not been affected. The relatively uniform data distribution means that when using a trained BP neural network for iterative prediction, every observation point can contribute to the fitting weight parameters. The more observation points are, the more effective fitting weight parameters will be. Hence, based on it, the prediction accuracy is greatly improved.

Secondly, for the tidal observation site called *St. Helier*, although the fluctuation of the selected training sample is as regular as the previously mentioned tidal observation site, the amplitude of the tidal elevation is smaller. As a result, under the same training and prediction time, the distribution of observation points becomes denser than the previously mentioned tidal observation site. When using a trained BP neural network for prediction, since the observation points with a relatively small scale between each data point are considered as similar points, the tidal elevation data that should have contributed to the fitting weight parameters, making an invalid contribution in the end. Therefore, when fitting the prediction curve, the actual number of effective fitting weight parameters decreases. This explains why the timescale becomes smaller (from 1.5h to 1h), the prediction accuracy also declines.

Finally, for the last tidal observation site, *Bournemouth*, the selected training sample not only has a relatively small amplitude but also has irregular fluctuation. Explaining in detail, the maximum tidal elevation does not exceed 2.5 meters and the trend of fluctuation is inconsistent with functions such as sine or cosine. As far as the distribution of observation points is concerned, some data points are particularly dense while some points are particularly sparse. Besides, there is not a relatively uniformly distributed observation point. The problem is that when the trained BP neural network

is used for prediction, the weight parameters fitted by particularly dense data points cannot be applied to the particularly sparse one, so the same as the opposite. Otherwise, it will bring serious underfitting results. This is the reason why the prediction accuracy is the worst here among three different tidal observation sites.

5.2. Conclusion

A prediction model combining periodic analysis and BP neural network has been successfully developed, which selects tidal data in February as the training samples and labels and then apply it for prediction in March.

In general, due to the residual value ε_t of the *ARMA-BP* method, the additional fitting weight parameters provided by itself are beneficial to correct the fitting rate of the prediction value. Therefore, in most cases, the *ARMA-BP* method is better than the *Non-ARMA-BP* method.

For observation points with regular fluctuations, which can be regarded as a sine or cosine function, after using the trained BP neural network can make a relatively accurate prediction. On the contrary, for observation points with relatively disordered fluctuations, the prediction accuracy brought out by the trained BP neural network is usually very poor.

Furthermore, the amplitude of tidal elevation is also a significant point that requires to be paid attention to. For the training samples selected at the same time, the larger amplitude means that the observation points are more likely to be uniformly distributed, neither too dense nor too sparse. Uniform data distribution plays a crucial role on improving prediction accuracy.

5.3. Future Work

Based on the selected training sample, the currently trained BP neural network can be only used for short-term prediction. If the long-term prediction is required, new appropriate parameters are essential for adjusting. For example, the number of input, hidden and output neurons, learning rate η, momentum factor α and residual value ε_t. However, the problem encountered at present is that there is no scientific and effective theory to help determine how to adjust parameters, which means that it only depends on manual repeated attempts.

Besides, in terms of common sense, large-scale parameters are necessary for long-term prediction since this type of prediction requires more details of observation points,

the fitting weight parameters provided by small-scale parameters are insufficient to accurately fit the observation points. Meanwhile, the new question is that training large-scale parameters is time-consuming. In some ways, it is essential to trade-off between prediction accuracy and training time. In order to solve this problem, there are three preliminary ideas: Firstly, optimize the raw neural network code structure. Secondly, propose a new type of neural network algorithm. Thirdly, apply tensor computation to achieve GPU acceleration, which is from the Pytorch library.

What's more, for the irregularly fluctuating data in some tidal observation sites, at present, there is no appropriate solution to upgrade prediction accuracy. To some extent, the application of neural networks in forecasting might not be valuable for this type of observation points. If continue engaging in research, it is better to consider using other forecasting methods only or combining neural network with other forecasting methods.

版权归chen qian

Bibliography

Rourke, F. O., Boyle, F., & Reynolds, A. (2010). “Tidal energy update 2009”, *Applied energy*, *87*(2), 398-409.

Doodson, A. T. (1921). “The harmonic development of the tide generating potential”, *Proceedings of the Royal Society of London. Series A* pp. 305–329.

Weisse, R., Bellafiore, D., Menéndez, M., Méndez, F., Nicholls, R. J., Umgiesser, G., & Willems, P. (2014). “Changing extreme sea levels along European coasts”, *Coastal Engineering*, *87*, 4-14.

Paul, G. C., Senthilkumar, S., & Pria, R. (2018). “An efficient approach to forecast water levels owing to the interaction of tide and surge associated with a storm along the coast of Bangladesh”, *Ocean Engineering*, *148*, 516-529.

Yen P.-H., Jan C.-D., Lee Y.-P., Lee H.-F. (1996). “Application of Kalman filter to short-term tide level prediction”, *Journal of Waterway, Port, Coastal and Ocean Engineering, vol. 122, num. 5,* pp. 226–231.

Consoli, S., Recupero, D. R., & Zavarella, V. (2014). “A survey on tidal analysis and forecasting methods for Tsunami detection”, *arXiv preprint arXiv:1403.0135*.

Anderson, D., & McNeill, G. (1992). “Artificial neural networks technology”, *Kaman Sciences Corporation*, *258*(6), 1-83.

Taormina, R., Chau, K. W., & Sethi, R. (2012). “Artificial neural network simulation of hourly groundwater levels in a coastal aquifer system of the Venice lagoon”, *Engineering Applications of Artificial Intelligence*, *25*(8), 1670-1676.

Nayak, D. R., Mahapatra, A., & Mishra, P. (2013). “A survey on rainfall prediction using artificial neural network”, *International Journal of Computer Applications*, *72*(16).

Elsafi, S. H. (2014). “Artificial neural networks (ANNs) for flood forecasting at Dongola Station in the River Nile, Sudan”, *Alexandria Engineering Journal*, *53*(3), 655-662.

Chen, B. F., Wang, H. D., & Chu, C. C. (2007). Wavelet and artificial neural network analyses of tide forecasting and supplement of tides around Taiwan and South China Sea. *Ocean Engineering*, 34(16), 2161-2175.

Ghorbani, M. A., Khatibi, R., Aytek, A., Makarynskyy, O., & Shiri, J. (2010). Sea water level forecasting using genetic programming and comparing the performance with artificial neural networks. *Computers & Geosciences*, 36(5), 620-627.

Yasseri, S. F., Bahai, H., Bazargan, H., & Aminzadeh, A. (2010). Prediction of safe sea-state using finite element method and artificial neural networks. *Ocean Engineering*, 37(2-3), 200-207.

Yuan, H., Tan, M., & Wang, W. (2015). Selection of methods for predicting tidal levels with a typhoon surge effect. *Journal of Coastal Research*, (73), 337-341.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536.

Hornik, K. (1991). “Approximation capabilities of multilayer feedforward networks”, *Neural networks*, *4*(2), 251-257.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). “Learning representations by back-propagating errors”, *nature*, *323*(6088), 533-536.

Jacobs, R. A. (1988). “Increased rates of convergence through learning rate adaptation”, *Neural networks*, *1*(4), 295-307.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.