Research on Discovering Time Series Motifs Based on Stacked Auto Encoder

YiHong Gao, XinMing Duan, ZiLiang Chen

Abstract—**Time series are widely used in financial data, streaming media data, weather data, census data, and system log data. It is very important to find frequently repeated patterns (motifs) in the time series. But, finding motifs is a complicated task due to its huge dimensions. In order to fix the dimension problems and reduce the calculation time of time series, researchers have done a lot of research on the dimensions of the time series, but they have not made much breakthrough. Therefore, this paper has carried out related work research to improve the problem: (i) Preprocessing the time series.(ii) Using the more popular neural network-stacked autoencoder to extract features of time series, which can reduce the number of time series calculations. (iii) Running a large number of experiments to verified the accuracy of time series motif search combined with stacked autoencoders. The study found that the method in this paper can not only guarantee the validity of the time series motifs, but also ensure the accuracy (about 88%).**

Index Terms —**Time Series ,Motif discovery,Stacked AutoEncoder,network**

I. INTRODUCTION

Nowadays, data mining in computer science is developing rapidly. The importance of data mining or knowledge discovery in the large amount of data is obtaining from different sources, such as the Internet, biomedical or geographic data processing, finance, industry and so on. Most of this information is stored in a time series. Time series is a sequence of data points arranged in chronological order, which can easily obtain data from applications such as financial data analysis, medical and health monitoring, scientific measurement, weather observation, music and motion capture. Motif discovery is an important subroutine in time series. Motif discovery aims to find frequent unknown patterns in time series without any prior information about its position or shape, as shown in ***Figure 1***. Because Motif can reveal useful potential information, it has been widely used in recent years.[[9],[5],[2],[6],[11],[8],[4]].

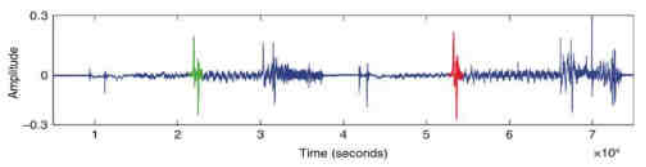

Fig. 1. The green and red pattern are a motif in time series

The emergence of Motif is not accidental. Motif usually contains important information about the system and provides valuable insights for the problem being studied. In addition, Motif provides important information for rule discovery and summary, or specifies prototype patterns for other pattern recognition tasks. Time series Motif discovery can be used for one-dimensional and multidimensional data and univariate or multivariate signals. It can also be applied to different types of sequences, such as time space sequences.

In order to find frequent patterns (Motif) in the time series, distance algorithms need to be used. In time series distance measurement, we found many useful formulas, such as Euclidean distance [10], correlation coefficient distance [2] formula, etc. Due to the problem of large dimensions in time series, when we use the distance formula, as the dimension increases, the number of calculations will increase.

Therefore, the research on reducing the dimension of time series has very important research significance. Although quite a few scholars have conducted research on Motif in the past and have achieved many useful academic results, there is still a lot of research space in the search of Motif in time series. The traditional methods of paa segmentation technology [7], symbolized sax [12], principal component analysis method pca [13] and other methods are all dimensionality reduction methods. Among them, symbolized sax is widely used in time series research, for example, MOEN algorithm [14], Mr. Motif algorithm [11], HIME algorithm [4] and other well-known algorithms all use sax symbolization technology. Although sax symbolization technology and other traditional methods can greatly reduce the large-dimensional problem of time series and reduce the calculation time of time series, they all have some defects and may lose some time series characteristics. Therefore, in view of the powerful dimensionality reduction ability and feature learning ability of deep learning, this article performs dimensionality reduction operations on time series from the direction of deep learning.

In order to develop a powerful deep learning dimensionality reduction model, we first analyze the time series and process it into a data set that can be used in deep learning training. After that, we train a usable neural network dimension- ality reduction model through a large amount of data. Finally, a large number of experiments and data verify the effectiveness of the trained neural network.

In summary, the main contributions of this work are as follows:

1) In order to be suitable for neural network training, we analyzed the data of the time series, and divided the

YiHong Gao, TianGong University, Tianjin, China
ZiLiang Chen, TianGong University, Tianjin, China

time series into sub-sequences through a sliding window, and formed a data matrix suitable for network training.

2) Look for a neural network suitable for univariate time series-stack autoencoder, and use the generated training data set to train the corresponding neural network model.
3) A large number of experiments have proved the feasibility and accuracy of neural network dimensionality reduction.

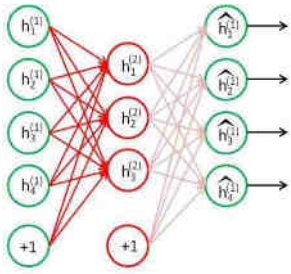

Fig. 2. The green and red pattern are a motif in time series

II. PROBLEM DEFINITIONS

A. *Definitions*

Definition 1. (*Time Series*) Time series T is a sequence of data points arranged in chronological order and usually formalized as $T = [(t_1, d_1), (t_2, d_2), ..., (t_n,d_n)]$, where d is the data item at time t, and n is the length of the time series.

Definition 2. (*Subsequence*) The subsequence $T[j : j+m] = [d_j,d_{j+1},...,d_{j+m}]$ is a set of consecutive points in T starting at position j and length m. This article only considers the data values of the time series.

Definition 3. (*Time Series Motif*) The time series motif is a frequent pattern in a time series. It also often described as repeated patterns, frequent trends, approximately repeated sequences, or frequent subsequences.

Definition 4.(*Top-K Motifs*) Motif is described as repetitive patterns, frequent trends, approximate repetitive sequences, shapes, events, or frequent subsequences, all of which have the same spatial structure. Motif aims to find the fragments with the largest number of subsequence matches. Through a predefined threshold ε, when the distance between the found subsequence fragments is less than ε, these subsequence fragments are considered to be a Motif.

B. *Stacked-AutoEncoder*

It is an unsupervised machine learning algorithm. Its model structure is simple with only one hidden layer. The network layer neurons are fully connected, as shown in ***Figure 2.*** Researchers have done a lot of Optimization Research on the classic automatic encoder, and put forward a variety of improved versions of the automatic encoder. For example, the neural units in the hidden layer of the original automatic encoder are sparsely limited. This network is called sparse automatic encoder; the stack type automatic encoder is composed of the superposition of the classic automatic encoder.

Stacked autoencoder[3] is a multilayer forward neural network that can be used for feature extraction of data, and it can also be used for feature extraction of time series. Autoencoders have been widely used in deep learning. For feature extraction, this paper uses a neural network. Take the first hidden layer as an example. The input is a 256-dimensional time series sub-segment vector x, and the output layer is set to a 128-dimensional vector z. This layer z is the feature vector after dimensionality reduction from the encoder, and then output The layer is a 256-dimensional vector reconstructed from the encoder according to the feature vector. The activation formula of the hidden layer and the output layer is

$$z = sigmoid(Wx + b) \quad (1)$$

$$\hat{x} = sigmoid(W^T y + b') \quad (2)$$

Among them, $sigmoid(x) = 1/(1 - e^{-x})$ is the Sigmoid function, where W is the weight matrix of the self-encoder, W^T is the weight matrix of the decoder, b' and b are the bias.

As unsupervised learning, autoencoder is different from supervised learning. The goal of autoencoder is to reconstruct the state of the input layer as much as possible in the output layer. Ideally, the features of the hidden layer can correctly reconstruct the same time as the input layer. Sequence fragments, so the feature vector obtained by the hidden layer is a good representation of the original time series, but the reconstruction cannot guarantee 100% accuracy. Therefore, there is still a little error in the hidden layer as the extracted feature vector. During training, the encoder should be made to reduce the reconstruction error as much as possible. Therefore, the objective function reconstructed by the autoencoder can be obtained, and the formula is

$$\{W, b, W^T, b'\} = argminLoss(x, \hat{x}) \quad (3)$$

Among them, $Loss(x, \hat{x})$ is the loss function, also called reconstruction error. Its function is to measure the reconstruction of the final output layer and input layer. What we are looking for in the autoencoder is the minimum value of this error.

The dimension of the hidden layer z is smaller than the dimension of the input layer x, so the hidden layer z can learn the low-dimensional representation of the input sample, and can contain the same information as the high-dimensional representation through decoding. Use the unlabeled data set X to perform unsupervised learning of automatic coding. Finally, for any input vector x, calculate through the trained model to obtain the hidden layer vector z, which is a low-dimensional code of the input vector. The weight training of the autoencoder adopts the stochastic gradient descent algorithm, and the formula is

$$W \leftarrow W - \eta * \frac{\partial Loss(x, \hat{x})}{\partial W} \quad (4)$$

This formula is a gradient descent formula used to update the weight matrix, where η is the update step size; other

parameters b, W^T , b' are updated in the same way. The step size of the update in this article is set to 0.001.

III. SATHTACKED-AUTOENCODER REPRESENTATION LEARNING

In this paper, the basic idea of using stacked autoencoders[1] is: For a large-scale high-dimensional time series, the time series is divided into large-scale small fragment time series vectors according to subsequences, and a matrix composed of large-scale time series fragment vectors For deep learning, use neural networks to learn the correlation between time points *ti* and t_m (i and m are any moments in the time series), such as a special correlation between t_5 and t_{10} in the time series, and complete these two points Feature learning, further feature extraction, to understand the correlation between multiple points, and finally learn the correlation features of this time series. After network training, we can get a model. The input of this model is a high-dimensional time series segment, and the output is a low-dimensional time series feature vector. The model is as follows, the input is $T[j : j + m]$, the output is $S[i : i + l]$, where $m > l$.

$$S[i : i+l] = f(T[j : j+m]) \quad (5)$$

A. Data Preprocessing

In this step, we focus on building a data set suitable for neural network learning, and building a vector matrix suitable for neural network input. The goal is feature extraction. Of course, we can't directly input the entire time series and perform feature extraction all at once. We start with the subsequence. In this step, we first read the prepared time series text. The data set contains 40mb, and it is preliminary estimated that there are several million time points. The first step of preprocessing is to normalize the minimum and maximum values, and then split according to the length m of 256 and the sliding window of 64. The time series after splitting should be a subsequence with a length of 256, T_0, T_{1*64}, ...T_{i*64}, and then synthesize the matrix according to each k subsequences, and then merge these matrices into a 3-dimensional data set D, which completes the arrangement of the data set, and the process is shown ***algorithm 1***: In ***Algorithm 1***, the input path is the storage path of the time series file, k is the number of subsequences of a matrix, m is the length of the subsequence, and s is the step length of the window sliding. Among them, the getTimeSeries function is to read time series files to form array data. The *MaxMinScoreAndSplitTimeSeries* function is the maximum and minimum normalized time series and divides the subsequences in the form of *Ti*. The createMatrix function is to initialize a matrix. The function of the whole algorithm is to get a data set suitable for neural network training.

B. Stack-AutoEncoder Learning model

The implementation of the stacked autoencoder is the core step of this article. First of all, our goal is to input a 256-dimensional time series subsequence, through which the corresponding feature vector can be extracted. We have to set some neural network parameters. In the implementation process, we set an input layer, two hidden

Algorithm 1: *Preprocessing(path)*

Input : *path*: The filePath of the time series
k: the number of sample
m: the length of subsequence
s: the stride of the window
Output: *D*: The dataset suitable for the Auto-Encoder

```
t ← getTimeSeries(path);
T ← MaxMinScoreAndSplitTimeSeries(path,t,m,s);
l = 0
n = 0
D_n ← createMatrix();
foreach subsequence T_i ∈ T do
    if l < k:
        l++
        D_n.add(T_i)
    else:
        D.add(D_n)
        l = 0
        n++
        D_n ← createMatrix();
```

layers, and an output layer. The feature vector extracted by the stacked self-encoding network is the second hidden layer, our input layer Set to 256, the output of the first hidden layer is set to 128, and then the activation function uses the sigmoid function. Therefore, the first layer input is X, the first layer self-encoder weight matrix W_{1e} should be [256,128], the corresponding first layer decoder weight matrix W_{1d} should be [128,256], the first layer offset b_{1e} should be [128], the decoder offset b_{1d} of the first layer should be [256], the output of the first layer is z_1, the first layer is realized according to the principle of ***formula (1) (2)***, and it is written according to ***formula (3)*** For the reconstruction error of this layer, use the *BP* gradient descent ***formula (4).***for network learning and training of this layer. The specific ***algorithm 2*** is as follows:

Algorithm 2: $firstHidden(D)$

Input : D: the time series dataset
Output: W_{1e}: the weight of encoder
W_{1d}: the weight of decoder
b_{1e}: the bias of the encoder
b_{1d}: the bias of the decoder

```
(W_1e, W_1d, b_1e, b_1d) ← WeightInitialization()
foreach subsequence D_i ∈ D do
    z_1 ← Encoder1(W_1e, b_1e, D_i);
    D̂_i ← Decoder1(W_1d, b_1d, z_1);
    (W_1e, W_1d, b_1e, b_1d) ← updateByLoss(D_i, D̂_i);
```

The WeightInitialization function in the above algorithm 2 initializes the values of W_{1e},W_{1d},b_{1e} and gives these weights inital values according to the mean value of 0 and the variance of 0.01. The Encoder function is the encoder that implements the ***formula (1)***, and the return is the input Feature vector after feature extraction. The Decoder function is the decoder that implements the ***formula (2)*** and returns the input time series segment reconstructed by the feature vector. The *updateByLoss* function updates the weight through the BP gradient descent ***formula (3) (4).*** Algorithm 2 returns some trained weight parameters that can be used for training of the second layer.

The input of the second hidden layer comes from the output of the first hidden layer, and the input of the second layer is *z1*, which is obtained by the self-encoder of the first layer. In the same way, the weight matrix W_{2e} of the second layer of the self-encoder should be [128,64], the corresponding decoder of the second layer is [64,128], and the offset b2e of the first layer should be [64], The decoder bias b2d of the first layer should be [128], the output of the first layer is z2, the second layer is realized according to the principle of ***formula (1) (2)***, and the repetition of this layer is written according to ***formula (3)***. To construct error, use BP gradient descent ***formula (4)*** for network learning and training of this layer. The specific algorithm is similar to

Algorithm 2, except that the input is the output z_1 of the first layer.

C. Model Validation

Algorithm 3 is the verification method of the model. The input Dt of the algorithm is the test data set, and w is the weight of the stacked autoencoder and a threshold. The FeatureLearning function in the algorithm is a dimensionality reduction function of the stacked self-encoding model. Lines 2 to 5 calculate the number of Motifs found in the original time series, and lines 6 to 9 are the number of Motifs in the time series after dimensionality reduction. The p in the 12th line is the accuracy of the trained neural network to find Motif.

Algorithm 3: Model Validation Algorithm

Input : D_t: the subsequence set used for test,
w: parameters of stacked autoencoder model,
ε: the threshold of the motifs
Output: p: the accuracy of the model

1 $D_t^{'} \leftarrow \text{FeatureLearning}(D_t, w)$;
2 **foreach** *subsequence* $T_i \in D_t$ **do**
3 **foreach** *subsequence* $T_j \in D_t$ **do**
4 $d_{ij} \leftarrow \text{EuclideanDistance}\ (T_i, T_j)$;
5 $M_i \leftarrow \text{findMotifs}\ (d_i, \varepsilon)$;
6 **foreach** *subsequence* $T_i^{'} \in D_t^{'}$ **do**
7 **foreach** *subsequence* $T_j^{'} \in D_t^{'}$ **do**
8 $d_{ij}^{'} \leftarrow \text{EuclideanDistance}\ (T_i^{'}, T_j^{'})$;
9 $M_i^{'} \leftarrow \text{findMotifs}\ (d_i^{'}, \varepsilon)$;
10 $n = k * |S|$;
11 $m \leftarrow compared(M, M^{'})$;
12 $p = m/n$;

IV. Experimental Evaluation

A. Experimental Evaluation

Our experiment uses the Linux system, and the neural network is written with *tensorflow*. *TensorFlow*^{TM} is a symbolic mathematics system based on dataflow programming, which is widely used in various machine learning and algorithmic. The programming implementation compilation tool is *pycharm*, which is a mainstream compilation tool. The programming language is python.

B. DataSet

In the experiment, the data set we used was a 40m text file. It is conservatively estimated that there are several million data points in this data set. The data set for neural network training in this paper is obtained by processing the text, and the preprocessing method can refer to ***Algorithm 1***. We extracted the first 69,000 data points of the time series as the test data set.

C. Validate the accuracy of the model

Since the threshold value selected by neural network feature extraction cannot be consistent with the original sequence, we should first obtain the distance matrix when performing neural network verification, and use ***algorithms 1 and 2***, on the trained neural network model, through various The parameters are tried to get a model with the best effect. The training step of the model is 0.001, the initial value of the weight obeys the normal distribution, the mean is 0, the variance is 0.01, and the value of k is 20. The input layer is 256, the output of the first hidden layer is 128, and the second is 64. The sliding window is set to 64. Since the Euclidean distance between the feature vectors of the neural network becomes quite small, we find the threshold ε by observing the distance matrix, and then find the corresponding Motif using this threshold ε as the standard. The Motif results are as follows:

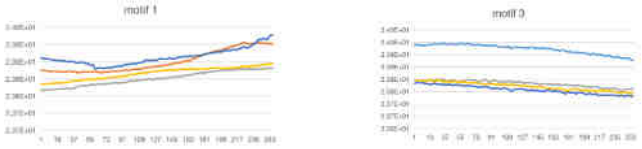

Fig. 3. The motif find by the AutoEncoder learning feature model

We mentioned above that the selection of the threshold after the feature extraction of the neural network cannot be consistent with the original sequence, so how can we estimate the accuracy? We can start from the distance matrix obtained by the original method and the distance matrix obtained by the neural network. Through ***Algorithm 3***, we can obtain the distance matrix obtained by the original method. Similarly, in ***Algorithm 3***, we can obtain the distance matrix of the neural network. But the values in these two matrices are quite different, and the thresholds are not synchronized. What kind of method can be used to ensure that the similarity of the two is the same? We can normalize the two distance matrices to change the range Specify between [0,1], and then find the value of the threshold less than 0.2 after the two matrices are normalized, so that we can maintain their consistency, then let n be the number of Motif found by the original method, this n is the same as the value of Motif found by the original method. Through the experimental results, we get: m is 6632, n is 7536, so we get the result: Experiment with 69000 points, the accuracy of the experiment is *88%*.

V. CONCLUSION

The use of autoencoders for feature extraction of time series is a new dimensionality reduction method, which provides researchers with a new way of thinking when studying Motif, and has a more and more extensive role in artificial intelligence and deep learning. However, the method proposed in this paper still needs to be improved a lot, and the self-encoding method can be applied to many fields, and further research is still needed.

References

[1] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer- wise training of deep networks. In NIPS, pages 153–160, 2006.
[2] Jeremy Buhler and Martin Tompa. Finding motifs using random projections. Journal of Computational Biology, 9(2):225–242, 2002.
[3] Wenqing Chu and Deng Cai. Stacked similarity-aware autoencoders. In IJCAI, pages 1561–1567, 2017.
[4] Yifeng Gao and Jessica Lin. Efficient discovery of variable-length time series motifs with large length range in million scale time series. CoRR, abs/1802.04883, 2018.
[5] Eamonn J. Keogh, Jessica Lin, and Ada Wai-Chee Fu. HOT SAX: efficiently finding the most unusual time series subsequence. In ICDM, pages 226–233, 2005.
[6] Hoang Thanh Lam, Toon Calders, and Ninh Pham. Online discovery of top-k similar motifs in time series data. In ICDM, pages 1004–1015, 2010.
[7] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. Finding motifs in time series. In Proc. of 2nd Workshop on Temporal Data Mining at KDD, pages 53–68, 2002.

[8] Wenqing Lin, Xiaokui Xiao, Xing Xie, and Xiaoli Li. Network motif discovery: A GPU approach. IEEE Trans. Knowl. Data Eng., pages 513–528, 2017.
[9] Abdullah Mueen. Enumeration of time series motifs of all lengths. In ICDM, pages 547–556, 2013.
[10] Abdullah Mueen and Eamonn J. Keogh. Online discovery and maintenance of time series motifs. In KDD, pages 1089–1098, 2010.
[11] Abdullah Mueen, Eamonn J. Keogh, Qiang Zhu, Sydney Cash, and M. Brandon Westover. Exact discovery of time series motifs. In SDM, pages 473–484, 2009.
[12] Pavel Senin, Jessica Lin, Xing Wang, and Tim Oates etc. Grammarviz 2.0: A tool for grammar-based pattern discovery in time series. In ECML PKDD, pages 468–472, 2014.
[13] Dragomir Yankov, Eamonn J. Keogh, Jose Medina, Yuan Chi Chiu, and Victor B. Zordan. Detecting time series motifs under uniform scaling. In KDD, pages 844– 853, 2007.
[14] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Zachary Zimmerman, Diego Furtado Silva, Abdullah Mueen, and Eamonn J. Keogh. Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. Data Min. Knowl. Discov., 32(1):83–123, 2018.