Multi-Source Transformer Models for Time Series Forecasting

Zebang Li
Rutgers University
New Brunswick, NJ 08854
`zl326@stat.rutgers.edu`

Zhe Zhang
Rutgers University
New Brunswick, NJ 08854
`zhe.zhang@rutgers.edu`

Abstract

In this report, we propose a new approach to multi-variate time series forecasting based on deep transformer models. Time series prediction plays an important role in machine learning and has many applications in real life. In this work we developed a novel method based on deep transformer model for multi-source time series. This approach works by combining multiple sources of information from other time series to learn complex patterns from a large input matrix. As case study, we show that our model has good performance on stock market data.

1 Introduction

In this paper, we will mainly focus on the prediction of time series through deep learning approach. Time series is a data type where the data points are ordered in time. Time series prediction algorithm is widely-used in the area of weather forecasting, stock value and market prediction in financial industries, customer growth, demand estimation in IT industries and the spread of infectious disease in public health, etc. Therefore, the prediction of time series is an important and influential area of research which arouses broad attention. In the traditional statistical literature, some representative models, such as AR model, MA model, ARMA model and ARIMA model has been well studied, but the assumptions from traditional statistical models are often too strong to mimic the time series data in real life. With the recent era of big data and the trend of deep learning, there has been more attention on the handling of sequence data recently including recurrent neural network(RNN), Transformer, etc, which brings new intuitions and ideas to handle time series prediction.

Time series can be naturally attributed into a specific type of sequential data and can be analyzed through deep learning approaches. Their has been some research on time series prediction through deep learning methods, including Selvin et al. [2017], Zhang and Man [1998], Tokgöz and Ünal [2018]. However, the recent study of time series based on transformer is quite limited. By far, the latest work was Wu et al. [2020], which analyzed time series based on transformer. However, this work fails to consider additional information from other sources of time series and combine such information into the estimation of time series prediction. In our research work, we will design a novel approach of analyzing time series by transformer motivated by Wu et al. [2020]. We will fully utilize additional data or data features into the inputs and further show that we could greatly improve the performance of prediction through our algorithm.

The major design of our research project consists of two parts. In the first part, we will consider to take advantage of additional data and features based on some known or historical time series data and further improve the accuracy of time series prediction. For example, for the stock value data, if we want to make predictions on the future stock value, the common approach is to use the historical stock value data as the input. But intuitively, the stock market is closely related to some other factors including the global policies, the economics, or the trade between countries. Such data could be acquired or quantified and are often ignored in the analysis of time series models. We will

Preprint. Under review.

possible add such features as the inputs and discover if such additional data features can improve the prediction. In the second part, we will also discuss how to use these additional data to benefit the prediction procedure. Below is the major contribution of this work:

- We designed a novel framework to make time series forecasting based on multi-sources time series through transformer. From our knowledge, this is the first work which considers time series forecasting from multi-sources by transformer in the deep learning field.
- We further analysed the stock market value prediction through our algorithm. After evaluation, we have shown that our algorithm benefits the prediction accuracy and greatly improve the results compared with previous work.

The rest of the paper will be organized as follows. In chapter 2, we will formally define the problem formulation and we will introduce the main model and major designs. We will also include the choice of optimizer and regularization. In chapter 3, we will conduct experiments based on our algorithm on real life data set. We will further show that our algorithms outperforms the existing algorithms. In chapter 4, we will make conclusions and also discuss on the possible directions of our future work.

2 Model

2.1 Problem Formulation

Given the target time series data $\boldsymbol{X}_0$ and other k time series $\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_k$, where $\boldsymbol{X}_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,t})$, $i = 0, \cdots, k$. For one-step prediction problem, we are interested in forecasting the unobserved future data points $x_{0,t+1}$ of $\boldsymbol{X}_0$. Each data point $x_{i,t}$ is a scalar and time series $\boldsymbol{X}_0$ is a vector. We pile up these time series together as a matrix time series. We denote this $(k+1) \times t$ matrix as $\boldsymbol{D}_0$, where $\boldsymbol{D}_0 = (\boldsymbol{X}_0, \boldsymbol{X}_1, \cdots, \boldsymbol{X}_k)'$. The input is $\boldsymbol{D}_0$ and the output is $x_{0,t+1}$.

2.2 Model Architecture

In this work, we add a connected layer first and then follow the design from the paper for the original Transformer architectures Vaswani et al. [2017], which consists of the encoder layer and the decoder layer.

Sampling: Each time for the training, we will sample the data from the data set $\boldsymbol{D}_0$. The data points sampled each time are consecutive data points with length $t' + l_0$, where $t' + l_0$ should be less than t. Notice that t' and l_0 are two pre-assigned hyper-parameters. We denote the first length of t' data points as $\boldsymbol{D} \in \mathbb{R}^{(k+1)\times t'}$. We also take the rest l_0 data points in $\boldsymbol{X}_0$ as the output for the decoder in the training process. See Figure 1.

Connected layer: We add a fully connected layer before the input for the encoder and the decoder to combine multiple sources of information from the data set $\boldsymbol{D}$. In this layer the input data matrix $\boldsymbol{D} \in \mathbb{R}^{(k+1)\times t'}$ is converted to a vector $\boldsymbol{Z} \in \mathbb{R}^{1\times t'}$. Such pre-processing will utilize the other source of time series data, which can help improve the accuracy of the prediction. Specifically, we will define a vector of length $k+1$ as $\boldsymbol{a} = (a_0, a_1, ...a_{k+1})$, where each element as trainable parameters. Then the output of this connected layer would be $\boldsymbol{Z}$ defined as follows:

$$\boldsymbol{Z} = \boldsymbol{a}\boldsymbol{D} \tag{1}$$

Encoder: We will feed the first l_1 elements of output in the previous connected layer, denoted as $\boldsymbol{Z}_{[1:l_1]}$, into the encoder, where the hyper-parameter l_1 is the length of input of the encoder. The encoder is composed of an input embedding layer and a positional encoding layer, and a stack of $N = 6$ identical layers. Follow the original design, each identical layers has two sub-layers, a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. For these two sub-layers, both of them take the residual connection around the layers and also are followed by a normalization layer.

Decoder: Similar as the encoder, the input of the decoder is the last l_2 elements of $\boldsymbol{Z}$, denoted as $\boldsymbol{Z}_{[-l_2:t']}$, where l_2 is the length of decoder input. Then it is also followed by an input embedding layer and a positional encoding layer. The decoder is also composed of a stack of $N = 6$ identical layers. However, the major difference is that the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. These three sub-layers also employ residual connections around each sub-layer and also followed by layer normalization. Same as the original design, we also implement the masking, which could prevent positions from attending to subsequent positions.

2.3 Training

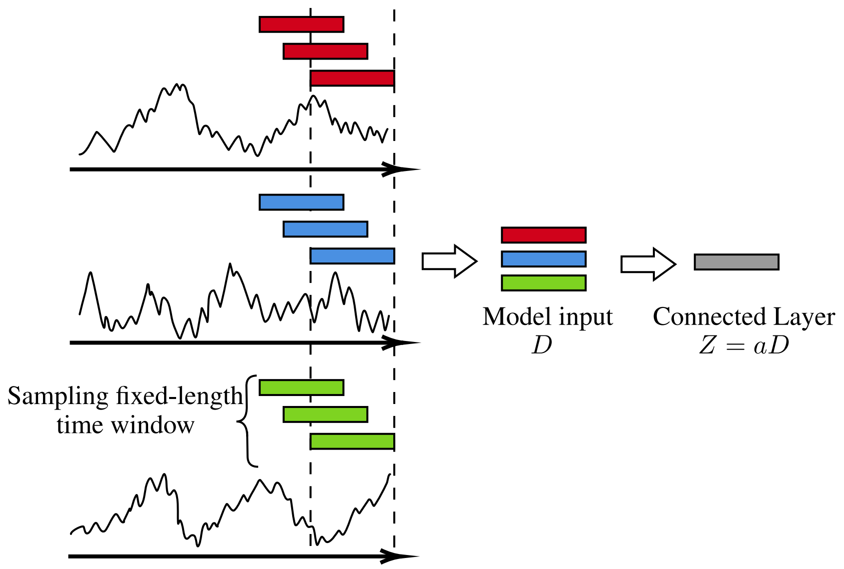

Figure 1: Linear combinations of horizontal slices

Training Data and Batching: We train the model to predict one-step ahead future data $x_{0,t+1}$ from previous t data points of $k+1$ time series, the key design of our model is shown as Figure 1. The generation of training samples has been discussed in the previous sub-section, where the choice of t' normally equals $l_1 + l_2$. A look-ahead mask is applied to ensure that attention will only be applied to data points prior to target data by the model.

Optimizer: We have done many experiments that tried different optimization methods, such as SGDs [Bottou and Bousquet, 2011, Zinkevich et al., 2010], Adaptive Gradient (Duchi et al. [2011]), Adam (Kingma and Ba [2014]) and AdamW variant (Loshchilov and Hutter [2017]), shown as Figure 2 (data and model detail describes in Section 3). It can be seen that Adam is best, a common and widely-used optimizer. Thus, we choose the Adam optimizer, with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. The learning rate also varies over the training procedure according to the formula below:

$$lrate = d_{model}^{0.5} \times \min(\text{step\_num}^{-0.5}, \text{step\_num} \times \text{warmup\_steps}^{-1.5}), \tag{2}$$

where d_{model} is the dimension of the input after the input embedding layer and warmup_steps $= 5000$.

Regularization: We apply dropout techniques with dropout rate 0.2 in each sub-layer in the encoder and decoder, including the self-attention layer, the feed-forward layer and the normalization layer.

Evaluation: For the evaluation, each time, we will take the one-step prediction as the evaluation. To be more exact, for the data set, we will first divide it into two parts: the training set and the test set. We will first use all the training data to perform training for the transformer. Then, for the prediction, we will make one-step forecasting each time from all data before the time stump of the target prediction. The evaluation will be based mean squared error.

3 Experiments

In this section, we will first introduce the data set we are using. Then, we will implement our algorithm on this data set. Results and conclusions will be shown accordingly.

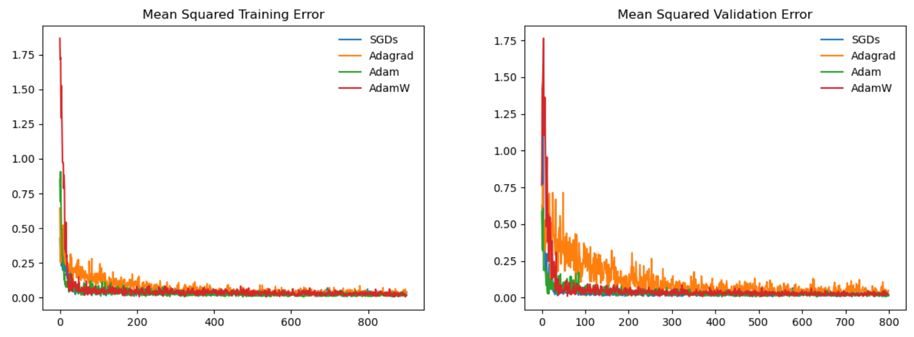

Figure 2: Loss curve of training error and validation error using SGDs, Adagrad, Adam, AdamW.

3.1 Data

The data set we are using in this experiment is the stock market data for Apple Inc., which can be naturally seen as time series data. The index we are interested in is the stock market value forecasting. We also notice that the transaction volume data, another source of time series, could be naturally correlated with the stock market value, which can be potentially improve the prediction accuracy. Thus, we would like to try to bring the information of the transaction volume into the prediction as well as the additional data. In our experiment, we use the stock market data from July 30th, 2020 to April 30th, 2021. We normalize the data by subtracting the mean and divide by standard deviation.

3.2 Results and Conclusions

We conduct one-step rolling forecast, that is, to predict $x_{0,t}$ we use all data available from $x_{i,0}$ to $x_{i,t-1}$, $i = 0, \cdots, k$ where k is the number of time series. In this case, as described in previous section, $k = 2$.

In our experiment, we apply the algorithm introduced in section 2 to the stock market data. For the hyper-parameters, we pre-assigned $l_1 = 5$, $l_2 = 3$ and $l_0 = 1$. Since we are trying to combine one additional time series into our prediction, then $k = 2$. We also run the algorithm from Wu et al. [2020] as the baseline method. We will compare the results from our algorithm with the existing baseline method. The result is shown below:

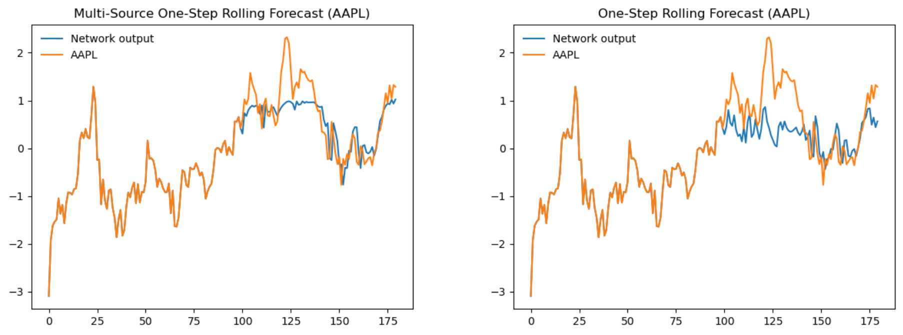

Figure 3: One step rolling forecasting results. The left figure denotes the forecasting from our algorithm and the right figure denotes the baseline methods.

For each figure, the orange line represents the true data and the blue line represents the prediction. We could claim that our algorithm could better mimic the trend of the time series compared with the baseline method. Specifically, Table 1 shows that indeed our multi-source transformer models outperform the single original transformer model.

Model	Training	Test
ARMA	62.67	72.32
Single-source Transformer	30.37	45.56
Multi-source Transformer	34.28	39.35

Table 1: Mean square error of one-step rolling forecast

4 Conclusions

In this work, we have initiated a novel approach to combine multi-sources time series as the input of the transformer model to greatly improve the accuracy of forecasting. We first elaborate the details of our algorithm and also implement our algorithm on the stock market data. From the results, we could conclude that our algorithm outperforms the existing method.

The future research may consist of two main directions. The first potential research field is if we could find better methods to combine the multi-sources time series data. In our approach, we consider to add a fully connected layer before the input. We could potentially find better approaches to dynamically choose the sliding window from the multiple sources to further improve the results. Another possible area of research is that, according to our experience, time series data type is often unstable and fluctuate a lot, so we could seek if other regularization methods, such as noise injection could potentially improve the robustness of the prediction.

5 Contributions

Both Zhe Zhang and Zebang Li brainstormed together, determined the topic, read relevant papers and initialted the ideas. Zebang wrote related codes and conducted the experiments and simulations. Zhe collected the results, started the draft and made presentations in-class.

References

L. Bottou and O. Bousquet. 13 the tradeoffs of large-scale learning. *Optimization for machine learning*, page 351, 2011.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE, 2017.

A. Tokgöz and G. Ünal. A rnn based time series approach for forecasting turkish electricity load. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, 2018.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

N. Wu, B. Green, X. Ben, and S. O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020.

J. Zhang and K. Man. Time series prediction using rnn in multi-dimension embedding phase space. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 2, pages 1868–1873. IEEE, 1998.

M. Zinkevich, M. Weimer, A. J. Smola, and L. Li. Parallelized stochastic gradient descent. In *NIPS*, volume 4, page 4. Citeseer, 2010.