University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Marta Nowakowska
Student no. 385914

Time series prediction using self-attention models

Master's thesis
in COMPUTER SCIENCE

Supervisor:
dr hab. Piotr Miłoś
Institute of Mathematics, Polish Academy of Sciences

Co-supervisor:
dr hab. Marek Cygan, prof. UW
University of Warsaw

Warsaw, August 2022

Abstract

TODO abstract

Keywords

time series, machine learning, self-attention model

Thesis domain (Socrates-Erasmus subject area codes)

11.4 Sztuczna inteligencja

Subject classification

D. Software
D.127. Blabalgorithms
D.127.6. Numerical blabalysis

Tytuł pracy w języku polskim

Prognozowanie szeregów czasowych przy użyciu modeli z uwagą

Contents

Chapter 1

Introduction

1.1. Time series and their importance

Time series is a sequence of data points measured across time with a fixed frequency. It is one of the most widespread type of data, making its forecasting incredibly useful for vast industries and research areas. Among others they are being used for stock market analysis, weather prediction and resource allocation planning.

It is not hard to imagine that this wide usage brought time series a lot of attention in the past. One of the most widely known method for their forecasting is ARIMA (autoregressive integrated moving average). Despite its popularity, it comes with some disadvantages, like the necessity to fit it separately for every similar time series and for the user to have some expertise in the parameter selection procedure.

This brings us to the next era of the time series forecasting: deep neural networks. The achievements in the natural language processing (NLP) field, including RNN and Transformer self-attention models have almost directly transferred to time series, due to a strikingly similar nature of both problems. However, RNNs and their successors (LSTM[2], GRU[3]) have difficulty with long sequences[4], causing them to struggle with time series that can contain long-term dependencies.

Transformer's[5] self-attention has revolutionized the world of long sequence processing, making it possible for the model to deal with larger-scale forecasting. However, there are several problems with integrating it directly for the time series forecasting usage: its local context-blindness[1] and large space-complexity growing quadratically with the sequence length. One solution was proposed in [1] by introducing convolutional self-attention and LogSparse Transformer.

1.2. Problem statement

Transformer proposed in [1] is centered around the idea of every point in a time series being drawn from some Gaussian distribution. Thus, the model's goal is to predict the parameters of these distributions, ensuring that each prediction is partially random and introducing Gaussian noise to it.

In reality, it's not always the case that every time series' noise comes from a Gaussian distribution. It would be more useful if we could model any continuous distribution instead. One way to approximate it is by using a discrete distribution with categories that are granular enough. Obviously, this comes with a disadvantage of being able to predict only a discretely limited value, which could cost some accuracy. However, the expectation is that this problem can be solved easily by just using a sufficiently large number of categories.

The goal of our project was using the Transformer model to predict time series assuming that each point in the sequences belongs to a categorical distribution.

1.3. Contributions

Work described in this thesis has been done via a collaboration with scientists from Institute of Mathematics of Polish Academy of Sciences.

My most important contributions are:

1. Gaussian distribution-based Transformer
2. Experiments with the Gaussian distribution-based Transformer
3. Base implementation of the dataset feeder
4. Time features as in [1] (on the data feeding side)

Chapter 2

Background

2.1. Notation

For a time series represented as a vector y, $y_t^{(i)}$ denotes t time step of a series number i.

2.2. Time series

We define a **time series** as a sequence of numbers measured across time with a constant **frequency**.

2.3. Causal convolution

Definition 2.3.1 ***Causal convolution*** *is a convolution used on input padded with $k-1$ zeros at the start, where k denotes the size of the kernel.*

This type of convolution is particularly useful for time series, since it enables us to make predictions with only partial input, essentially generating a completely new time series.

2.4. Attention

Attention is the key part to explaining the way Transformer works. It allows us to train an approximated dictionary which encodes the relations between different parts of the input.

One needs to provide three matrices to the attention layer: keys, queries and values. They are all created from the input in a way that depends on the attention function.

2.4.1. Scaled Dot-Product Attention

Scaled dot-product attention is perhaps the most known kind of attention function, which was introduced in [5]. Matrices Q, K, V (respectively queries, keys and values) are created by multiplying the input matrix with three distinct weight matrices W_Q, W_K, W_V.

For given Q, K, V we compute the attention with the following formula:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}} \cdot M)V \text{ (from [1])}$$

where d_k stands for the dimension of Q and K. Dividing by this value is supposed to balance out the growth of Q and K^T dot product with larger values of d_k.

M is a mask matrix with zeros in the upper triangle part and ones in the rest. It prevents the model from attempting to relate the past events to the future ones.

The dot-product attention does not use the information about input's general shape, which is a property of the classical Transformer that we call **context-blindness**. This means that points with similar values can be assigned a strong attention score, even if they come from dissimilar (differently-shaped) events.

2.4.2. Convolutional Attention

The convolutional attention[1] is calculated in almost the same way as the scaled dot-product attention. It only differs in the way it computes queries and keys. Instead of applying a matrix multiplication with weight matrices, it uses a causal convolution with a stride of 1. When the kernel's size is equal to 1 it is equal to the matrix multiplication.

The usage of convolution is supposed to offset the dot-product attention's context-blindness. Since the convolution layer learns to recognize input's shape, stronger attention score should be assigned to similar events.

2.4.3. Multi-headed attention

Each attention block might contain more than one parallel attention layer, aka attention "head". In theory, this allows for different layers to learn other kind of relationships in the input and is a very successful tool e.g. in natural language processing[5].

The outputs of all attention heads are concatenated before passing them down to the next block.

2.5. Architecture

2.5.1. Overview

The Transformer in [1] is a modification of the original Transformer[5], that uses only a decoder. We follow the same architecture type and therefore whenever we speak about the Transformer we will mean a decoder-only[6] architecture one.

Our Transformer consists of a causal convolution layer, positional encoding layer, N decoder blocks and a layer normalization.

2.5.2. Positional encoding

Positional encoding is a mapping of each position in the input to a vector. A positional encoding layer sums the (embedded) input with its positional encoding matrix.

2.5.3. Decoder

The decoder block consists of a causal attention block and a feed-forward block. Both blocks are residual.

2.6. Metrics

2.6.1. Quantile loss

Quantile loss is a metric designed to evaluate and/or train models designed with quantile prediction in mind. We define ρ-quantile loss as follows:

$$QL_\rho(y, \hat{y}) = max(\rho(\hat{y} - y), (\rho - 1)(\hat{y} - y))$$

, where y denotes the model's prediction for the ρ quantile and $\hat{y}$ is the actual value.

This metric is characterized by a stronger penalization of over-predictions for ρ smaller than 0.5, and under-predictions for ρ greater than 0.5.

2.6.2. Risk metric

$Risk_\rho$ is a type of normalized quantile loss metric defined as follows for $\rho \in (0, 1)$:

$$Risk_\rho(\mathbf{y}, \hat{\mathbf{y}}) = 2\frac{\sum_{i,t} QL_\rho(y_t^{(i)}, \hat{y}_t^{(i)})}{\sum_{i,t} |y_t^{(i)}|}$$

It is very popular in time series prediction papers that use the Transformer (e.g. in [1], [7]), which lead us to choose it as the metric we use in the model's evaluation.

2.7. Training

2.7.1. Covariates

Following [1], we use additional variables as part of the model's input aside the time series itself. There are three kinds of covariates we use:

1. Numerical information about time the input was recorded (year, month, day-of-the-week, hour-of-the-day, minute-of-the-hour). We normalize them and store in one dimension.
2. Embedded ID of the time series with the same dimensionality as positional embeddings.
3. Input's distance from the sequence's start.

The input to our model consists of summed ID embeddings and positional embeddings concatenated with rest of the covariates.

2.7.2. Weighted sampling

We adapt a weighted sampling technique in our training, similarly to [1] and [7]. Its main purpose is to counteract model's tendency to underfit data of a large scale, since it occurs more rarely in the datasets. We test that idea by including the weighted versus uniform sampling in our grid searches.

We calculate the proper weights by normalizing the scale factor v_i:

$$v_i = 1 + \frac{\sum_{t=1}^{t_0} y_t^{(i)}}{t_0}$$

Bibliography

[1] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang and Xifeng Yan. *Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting.* arXiv:1907.00235, 2019.

[2] Sepp Hochreiter and Jürgen Schmidhuber. *Long short-term memory.* Neural computation, 9(8):1735–1780, 1997.

[3] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. *On the properties of neural machine translation: Encoder-decoder approaches.* arXiv preprint arXiv:1409.1259, 2014.

[4] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. *Sharp nearby, fuzzy far away: How neural language models use context.* arXiv preprint arXiv:1805.04623, 2018.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. *Attention is all you need.* NIPS 2017.

[6] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. *Generating wikipedia by summarizing long sequences.* arXiv preprint arXiv:1801.10198, 2018.

[7] Valentin Flunkert, David Salinas, and Jan Gasthaus. *Deepar: Probabilistic forecasting with autoregressive recurrent networks.* arXiv preprint arXiv:1704.04110, 2017.