Transformers for Time Series Forecasting

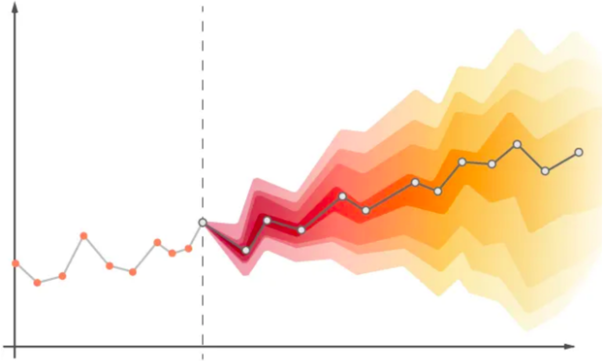
Marco Zanotti

University Milano-Bicocca

Contents

1. The TSF Problem

Time series forecasting (TSF) is the task of predicting future values of a given sequence based on previously observed values.



The TSF problem may be essentially identified by the following aspects:

- **Prediction objective**: point forecasting vs probabilistic forecasting
- **Forecast horizon**: short-term vs long-term forecasting
- **Input-Output dimension**: univariate vs multivariate forecasting
- **Forecasting task**: single-step vs multi-step forecasting

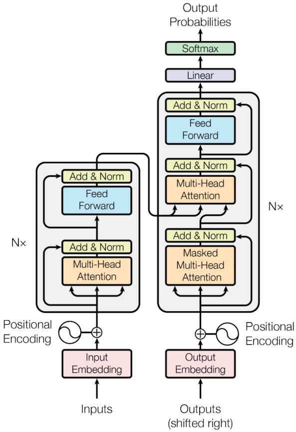
The TSF problem is usually faced with statistical models (ARIMA, ETS) or deep learning models (RNN, LSTM).

The main challenges of the TSF problem are:

- **uncertainty** increases as the forecast horizon increases
- difficulty in capturing **multiple complex patterns** over time
- difficulty in capturing **long-term dependencies** (critical for long-term forecasting)
- difficulty to handle **long input sequences**

2. Vanilla Transformer

- Based on Encoder-Decoder architecture
- Uses self-attention mechanism to access any part of the sequence history
- Positional encoding allows to account for element positions
- Residual connections and layer normalization help to stabilize the learning process
- Each encoder and decoder layer is composed of an attention layer and a feed-forward layer



Can vanilla Transformers be used for TSF?

The TSF problem can be seen as a sequence learning problem such as machine translation.

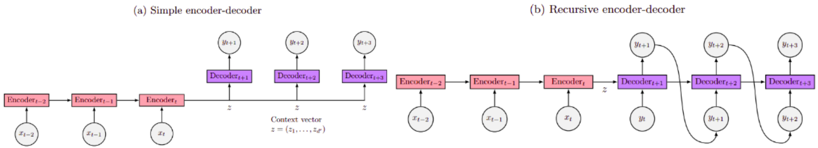
Main ingredients allowing to use vanilla Transformers for TSF:

- **Multi-head Self-attention** mechanism allows to access any part of the sequence history, capturing both short-term and long-term dependencies (but it is invariant to the order of elements in a sequence)
- **Positional encoding** allows to account for the sequence ordering
- **Masked self-attention** allows to avoid information leakage from future

Can vanilla Transformers be used for TSF?

Just few changes are needed to adapt Transformers to TSF:

- **Remove the final activation** function (softmax) from the output layer and set the dimension of the linear layer equal to the forecasting horizon
- **Adapt the structure** to the desired forecasting task (single-step or multi-step)



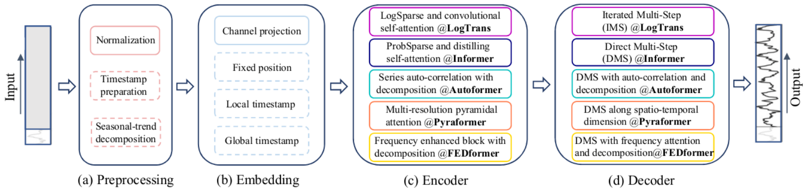
Problems with vanilla Transformers

- **Positional encoding**: only the order in which two elements occur is taken into account, but their temporal distance is not
- **Computational complexity**: given a sequence of length L, the time and memory burden is $O(L^2)$, making it difficult to learn patterns in long time series
- **Simple Architecture**: the architecture does not include any component of typical importance in TSF (e.g. autocorrelation, decomposition, recurrent layers, etc.)

3. TSF Transformers

Transformers are **very appealing for long-term TSF** due to their ability to learn long-range dependencies.

Many solutions have been proposed to adapt Transformers to TSF, mainly in the direction to adopt **more efficient attention** and **expand the architecture**.

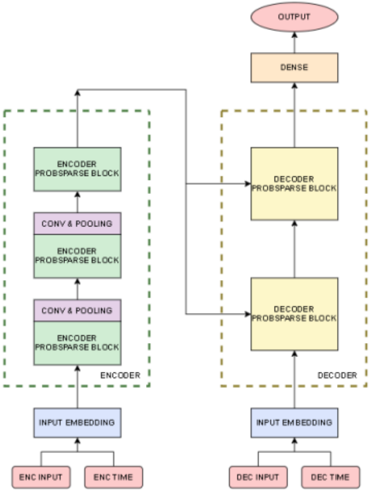


Informer

Informer employs two major improvements:

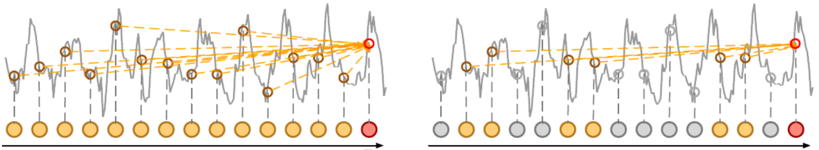
- **ProbSparse Attention Mechanism**, which replaces the standard self-attention used in the vanilla transformer with a sparse attention so to achieve $O(LlogL)$ complexity
- **Distilling Module**, which reduces the input size between encoder layers into its half slice, removing redundancy and reducing the computational burden

Informer - Architecture



Informer - ProbSparse Attention

The main idea of **ProbSparse** is that just a small subset of queries ("active") effectively contribute to the attention mechanism. The ProbSparse attention selects the "active" queries, and creates a reduced query matrix, which is then used to calculate the attention weights, reaching $O(LlogL)$ complexity.



(a) Full Attention (b) Sparse Attention

In practice, the KL divergence is used to define a sparsity measure, since active queries distribution diverges from uniform, and a random sample of K is used to obtain a sparse matrix of keys.

Informer - Convolution Layers

Because of the ProbSparse self-attention, the encoder's feature map has some redundancy that can be removed. The **distilling** operation is used to reduce the input size between encoder layers.

In practice, Informer's distilling operation just adds **1D convolution layers**, along the time dimension, with max pooling between each of the encoder layers.

$$X_{n+1} = MaxPool(\, ELU(\, Conv1d(Xn)\,)\,)$$

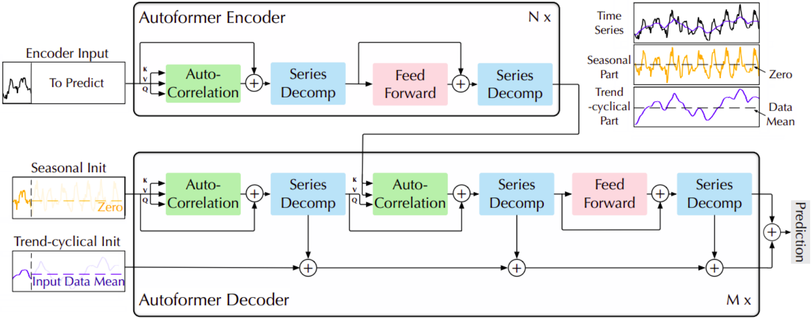
This reduces by half the size of the data between the feature space, improving both memory and computational efficiency.

Autoformer

Autoformer builds upon two traditional time series analysis methods:

- **Decomposition Layer**, which allows to decompose the time series into seasonality and trend-cycle components, enhancing the model's ability to capture these components accurately
- **Autocorrelation Attention Mechanism**, which replaces the standard self-attention used in the vanilla transformer with an autocorrelation mechanism, allowing to capture the temporal dependencies in the frequency domain and achieving $O(LlogL)$ complexity

Autoformer - Architecture



Autoformer - Decomposition Layer

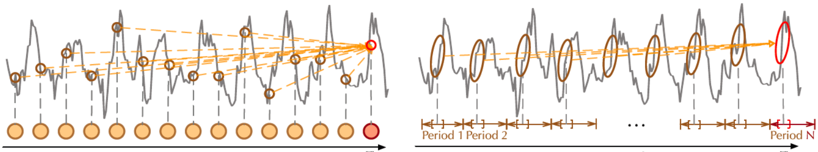
Autoformer incorporates **decomposition blocks** as an inner operation of the model.

Encoder and decoder use decomposition blocks to extract and aggregate the trend-cyclical and seasonal components from the series progressively, so to make raw data easier to predict.

For an input series X_t with length L, the decomposition layer returns X_{trend} and $X_{seasonal}$, both of length L. In practice, X_{trend} is extracted using some form of moving-average and $X_{seasonal}$ is then obtained by difference.

Autoformer - Autocorrelation Attention

Autoformer uses **autocorrelation within the self-attention** layer, extracting frequency-based dependencies from (Q, K). The autocorrelation block measures the **time-delay similarity** and aggregates the top n similar sub-series to reduce complexity.



(a) Full Attention (b) Auto-Correlation

In practice, autocorrelation of the queries and keys for all lags is calculated at once by Fast Fourier Transform, so to achieve $O(LlogL)$ time complexity (similar to Informer).

4. Conclusions

Conclusions

- Transformers are very appealing for TSF due to their ability to learn long-range dependencies
- Many solutions have been proposed to adapt Transformers to TSF, mainly in the direction to adopt more efficient attention and expand the architecture
- Informer and Autoformer are two of the most successful solutions, and they both achieve $O(LlogL)$ complexity
- Empirical results show that Transformer models are able to reach SOTA performance in TSF, and are among the best methods for long-term forecasting

Bibliografy

Ailing Z., et al., 2023, 'Are Transformers Effective for Time Series Forecasting?', AAAI

Haixu W., et al., 2021, 'Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting', NeurIPS

Haoyi Z., et al., 2021, 'Informer: Beyond efficient transformer for long sequence time-series forecasting', AAAI

Lara-Benitez P., et al., 2021, 'Evaluation of the Transformer Architecture for Univariate Time Series Forecasting', Advances in Artificial Intelligence, CAEPIA

Qingsong W., et al., 2022, 'Transformers in Time Series: A Survey', AAAI

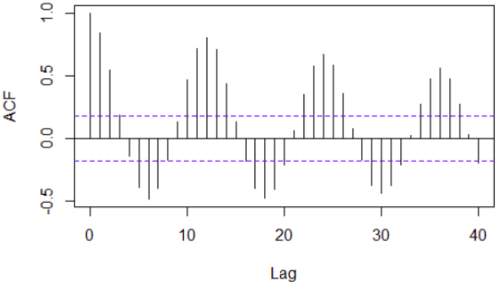
Thank you!

Appendix

Autocorrelation

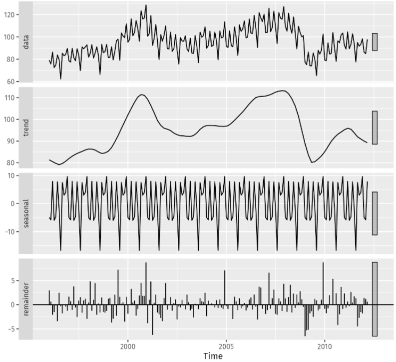
In theory, given a time lag τ, **autocorrelation** for a single discrete variable Y is used to measure the “relationship” between the variable at time t to its past value at time $t-\tau$.

$$Autocorrelation(\tau) = \mathcal{R}(\tau) = Corr(Y_t, Y_{t-\tau})$$



Time Series Decomposition

In time series analysis, **decomposition** is a method of breaking down a time series into three systematic components: **trend-cycle**, **seasonal variations**, and **random fluctuations**.



Time2Vec Embedding

Time2Vec is a method of encoding time information into a vector and has three important properties:

1. **Periodicity**, captures periodic and non-periodic patterns
2. **Invariant to time rescaling**
3. **Model-agnostic**, can be combined with many models

For a given scalar time τ, $t2v$ of τ is defined as:

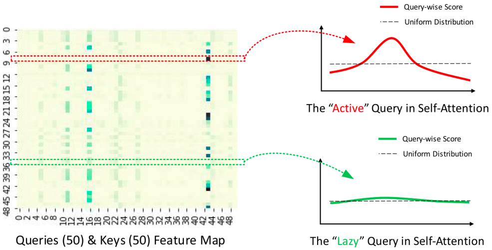
$$t2v(\tau)[i] = \begin{cases} \omega_i \tau + \phi_i, & \text{if } i = 0 \\ \mathcal{F}(\omega_i \tau + \phi_i), & \text{if } 1 \leq i \leq k \end{cases}$$

where $\mathcal{F}$ is a periodic activation function, and ω_i and ϕ_i are learnable parameters.

Informer - ProbSparse Attention

$$Attention(Q,K,V) = softmax(\frac{Q_{reduced}K^T}{\sqrt{d_k}})V$$

$$M(q_i,K) = \max_j \frac{q_i K_j^T}{\sqrt{d_k}} - \frac{1}{L_k}\sum_{j=1}^{L_k}\frac{q_i K_j^T}{\sqrt{d_k}}$$



Queries (50) & Keys (50) Feature Map

Autoformer - Autocorrelation Attention

$$\tau_1, \ldots, \tau_k = arg\ Top - k\ (\mathcal{R}_{Q,K}(\tau))$$

$$Attention(Q, K, V) = \sum_{i=1}^{k} Roll(V, \tau_i) \times softmax(\mathcal{R}_{Q,K}(\tau_i))$$