Time series analysis with a Convolutional Neural Network

Ali Khudiyev

December 2020

1 Introduction

Time series can be used to analyze a sequence of data which has been measured with the equal distance in time. In this particular example that we have been provided, it is obvious that the time difference is only a second. There are 5 features which can be used to predict the state of system. However, the time series data set can be different from a regular data set in a sense of correlation between different system states. Although there is usually no relationship among different outputs in a regular data set, this might not be the case for the time series. The current state of the system may depend on the previous states in time.

2018-05-17 22:56:06	27897490.0	19824230.0	125.6859	4059666.0	97.55283	0
2018-05-17 22:56:07	27897450.0	19824230.0	125.6859	4059666.0	97.55283	0
2018-05-17 22:56:08	27897360.0	19824230.0	125.6859	4059666.0	97.55283	0
2018-05-17 22:56:09	27897430.0	19824230.0	125.6859	4059666.0	97.55282	0
2018-05-17 22:56:10	27897500.0	19824230.0	125.6859	4059666.0	97.55282	0

Figure 1: Time series data set example

2 Relevance of CNN

Convolutional Neural Networks use convolutional layers to filter the data and extract some meaning or more abstract concept out of it. That's why, after each convolution we get a feature map related to the data. What feature(s) we obtain or how good(representative) they are depends on the applied filter(s). The filters, that CNNs apply, has two main benefits:

- Obtain higher abstraction level on the data
- Compress the data by representing it with less but more abstract information

To predict the current state of the system, we may need to know several previous states. The filters can be very helpful to pack several successive timestamps together and understand the general behaviour of the system, or in other words, to obtain higher abstraction level on the given data. This is not usually done with the regular artificial neural networks, since they do not usually consider the dependence in time series(they are not even trained with time series). Thus, CNNs have an advantage over the regular ANNs in this case.

Since the time series has usually long dimensionality, data compression may be useful to reduce the number of total parameters(i.e. weights) in the training stage of our model. That is not the case with regular ANNs since we do not have any convolution+pooling layers in those architectures.

2.1 1D convolutions

Since the time series is a one dimensional space, we can take m successive timestamps and apply a filter with the help of 1D convolutional layer. After filtering (and probably pooling), we can try to predict the last state of the system which is also considered as the current state.

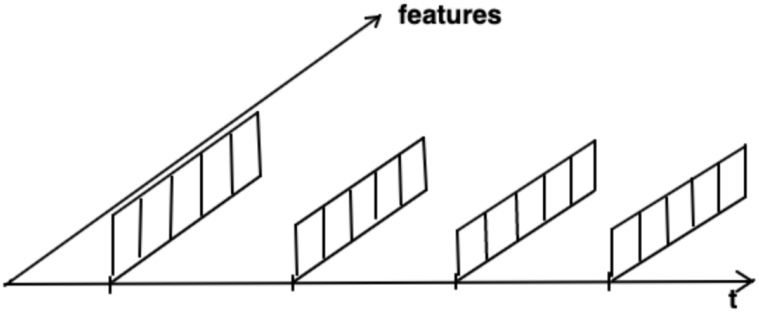

Figure 2: 1D space of inputs (with 5 features)

3 Preprocessing data

The preprocessing can be generalized as below:

- Cleaning the data set(removing or possibly modifying all invalid cells)
- Splitting for training, testing and validation
- Obtaining input vectors as well as the output values for each training/testing sample

After removing(or possibly modifying) all invalid cells in our data set, we need to create input vectors and output values for both training and testing stages. To do that, we can split our data set for training, testing and probably validation stages.

An input vector has 5 values since there are only 5 features in the data set. A training sample has m input vectors obtained from m successive timestamps and since we have n timestamps in our data set, we are going to have $n-m+1$ training samples. Finally, the output is just a single value which is just the last system state in time. So, the shape of the inputs and outputs are going to look like this:

$$\text{input: } (\underbrace{n-m+1}_{\text{training samples}}, \underbrace{m}_{\text{timestamps in each sample}}, \underbrace{5}_{\text{features}}), \qquad \text{output: } (n-m+1)$$

For example, given the data set (1), one could obtain input-output pairs as shown below:

Output	**Input**
	27897490.0 19824230.0 125.6859 4059666.0 97.55283
	27897450.0 19824230.0 125.6859 4059666.0 97.55283
0	27897360.0 19824230.0 125.6859 4059666.0 97.55283
	27897450.0 19824230.0 125.6859 4059666.0 97.55283
	27897360.0 19824230.0 125.6859 4059666.0 97.55283
0	27897430.0 19824230.0 125.6859 4059666.0 97.55282
	27897360.0 19824230.0 125.6859 4059666.0 97.55283
	27897430.0 19824230.0 125.6859 4059666.0 97.55282
0	27897400.0 19824230.0 125.6859 4059666.0 97.55282

Table 1: Example input-output pairs (n=5, m=3)