Tensor Forecasting

Seyun Kim
Seoul National University
Seoul
seyun0114kim@gmail.com

U Kang
Seoul National University
Seoul
ukang@snu.ac.kr

August 25, 2021

Abstract

Given a tensor with a time dimension, how can we accurately predict its future values? How can we adjust the temporal factor in a form that better represents data seasonality and trend? How can we make one model learn all theses tasks at once? Tensors are useful in storing and representing high dimensional data. For example, weather data can be represented as a 3-order tensor with dimensions corresponding to locations, weather features, and time. In this paper, we will focus on tensors with time mode as one of its dimensions. Predicting future values of a tensor can provide large amount of diverse information ahead of time, benefiting people in areas such as stock price prediction or demand forecasting. Most studies on tensor forecasting focus on smoothing the temporal factor of a tensor using various smoothing techniques. Among others, Gaussian kernel smoothing has showed great contribution in improving forecasting accuracy. However, Gaussian smoothing tends to estimate the target too closely that it learns and reproduces anomalies and noises as well. To resolve this problem, we propose a method that 1) distinguishes anomalies from real data and reduce their effects in smoothing, 2) optimizes smoothing, anomaly reducing, and tensor decomposition altogether at once, and 3) maintains high tensor reconstruction accuracy.

1 Introduction

Tensor decomposition has been used in many applications for its proved efficiency in revealing latent structures in data[2]. Among all others, CP/PARAFAC and Tucker decomposition are the two most widely used tensor decomposition algorithms. CP/PARAFAC, from here referred to as CP, decomposes a tensor into a sum of rank-one tensors which represent latent structures of the tensor. On the other hand, Tucker decomposition factorizes a tensor into a core tensor and matrices for all modes and is effective for dimension reduction. In this paper, CP will be used for its usefulness in revealing latent features in numerous data mining applications.

Tensor, a multi-dimensional array of data, is useful for storing information thas has variety of features. For example, the amounts of pollutants at every location and hour can be represented as a 3-dimensional tensor whose axes are location, type of pollutant, and time. In a specific case as such when one of the dimensions is time, we are often interested in predicting future values of the given tensor. Suppose there is a stock data tensor whose dimensions are *stock X price X time*, it would bring a lot of profit if we could predict tomorrow's stock prices.

[4]To forecast values in tensors, one of the most commonly used tactics is to first perform tensor decomposition and obtain factor matrices. This is to reduce dimensionality and size of the data. In the proposed method, the author used CP decomposition to preserve the features of tensor. Once the factor matrices are given, most existing methods perform smoothing regularization to remove noise and accentuate trends. Smoothing can be done in multiple ways, such as moving average smoothing, exponential smoothing, or [1]Gaussian kernel smoothing. The most commonly used and effective smoothing method is Gaussian

smoothing. However, Gaussian smoothed vector estimates the original vector too closely that it learns noises as well.

Data are often obtained by measuring values using tools or by reports written by humans. These are all prone to errors and sudden changes due to machine malfunction, human mistakes, or unexpected outside forces. For example, COVID confirmed cases can rise or fall due to the person who made a mistake in the COVID report or due to the emergence of new highly contagious mutation of the virus. In both cases, the number of the total confirmed will show a spike or dip at the occurrence of such errors and there needs to be a way to discourage the forecast model from learning such errors. Gaussian kernel smoothing, however, easily learns the noise because it estimates the original sequence without distinguishing between noise and real values. The proposed method aims to reduce sudden spikes and dips that are most likely unreliable information in the data so that the forecast model only learns accurate data to make predictions.

2 Preliminaries

2.1 Notations

In this section, we describe the preliminaries of tensor decomposition and tensor forecasting. Throughout the paper, we use the symbols defined in Table 1.

Table 1: Table of symbols

Symbol	Definition	Symbol	Definition
$\mathcal{X}$	tensor $\in R^{I_1 \times \ldots \times I_N}$	α	index $(i_1, ..., i_N)$
$A^{(n)}$	nth factor matrix ($\in R^{I_n \times K}$)	x_α	entry of $\mathcal{X}$ with index α
$\mathbf{a}^{(n)}_{i_n}$	i_nth row of $A^{(n)}$	N	mode
$a^{(n)}_{i_n r}$	(i_n, r)th entry of $A^{(n)}$	R	rank
t	time mode of $\mathcal{X}$	I_n	length of the nth mode
$\circ$	outer product	$L_{i_t,r}$	level at (i_t, r)th entry of $A^{(t)}$
$b_{i_t,r}$	trend at (i_t, r)th entry of $A^{(t)}$	$S_{i_t,r}$	seasonality at (i_t, r)th entry of $A^{(t)}$
s	seasonality period	$\mathcal{X}\|_F$	Frobenius norm of tensor $\mathcal{X}$

2.2 Tensor Decomposition

Tensors are usually high dimensional and large in size, making it difficult to analyze them in their raw form. Tensor decomposition breaks a tensor down to lower dimension and provides dimension and size reduction. [3]One of the most widely used tensor decomposition methods are CP/PARAFAC (CP) and Tucker Decomposition. For the proposed method, we used CP Decomposition.

CP Decomposition aims to decompose a given tensor into latent matrices that closely approximates the tensor when reconstructed. Those latent matrices are also called as factor matrices and they are obtained by minimizing the following loss

$$\mathcal{L} = \|\mathcal{X} - \sum_{r=1}^{R} \mathbf{a}^{(1)}_r \circ \cdots \circ \mathbf{a}^{(N)}_r\|_F^2 = \sum_{\forall \omega \in \Omega} (x_\omega - \sum_{r=1}^{R} \prod_{n=1}^{N} a^{(n)}_{i_n,r})^2 \tag{1}$$

where Ω indicates a set of indices of observed entries and x_ω indicates $\omega = (i_1, \ldots, i_N)$th entry of $\mathcal{X}$. In the proposed method, we use the traditional CP decomposition to obtain factor matrices. One of the research attempts was to investigate effectiveness of using non-negative CP decomposition in comparison to original CP decomposition.

2.3 Holt-Winters Exponential Smoothing

Our forecasting method is based on Holt-Winters Exponential Smoothing (HWES). HWES allows for simple but powerful time series analysis by using three coefficients: level, trend, and seasonality. They can be

described in the following recursive equations.

$$L_{i_t,r} = \alpha(a_{i_t,r} - S_{i_t-s,r}) + (1-\alpha_r)(L_{i_t-1,r} + b_{i_t-1,r}) \tag{2}$$

$$b_{i_t,r} = \beta(L_{i_t,r} - L_{i_t-1,r}) + (1-\beta)b_{i_t-1,r} \tag{3}$$

$$S_{i_t,r} = \gamma(a_{i_t,r} - L_{i_t,r}) + (1-\gamma)S_{i_t-s,r} \tag{4}$$

Level can be thought of as what is left after trend and seasonality are removed from the data. Trend is defined as change in level, and seasonality says temporal patterns that appear every specific time period. Note that current level, trend, and seasonality are all weighted by smoothing parameters $\alpha, \beta,$, and γ that range 0-1. Note that because HWES coefficients are recursive, values in the past are weighted by an amount exponentially less by a factor of $(1-\beta)$ than the most recent coefficients. This indicates that the smoothing parameters determine the importance of past values of HWES coefficients.

Another strength of HWES is that it can combine information about level, trend, and seasonality to predict values in the future using the following equation where k is the number of time steps in the future to forecast.

$$F_{i_t+k,r} = (L_{i_t,r} + kb_{i_t,})S_{i_t+k-s} \tag{5}$$

[2]HWES forecasting can be done by either adding or multiplying HWES coefficients but we decided to use multiplication since real life data are rarely linear.

3 Proposed Method

The main motivation behind our method is to reduce sudden spikes and dips in the data so that the forecast model does not learn unreliable information. Sudden spikes and dips can be defined as points that show a high rate of change above threshold in a short period of time.

Our proposed method is as follows: We will use CP decomposition to preserve tensor features and obtain factor matrices. Then, we will optimize the temporal factor matrix by minimizing the following loss function

$$\mathcal{L} = \mathcal{L}_t + \mathcal{L}_{sd} + \mathcal{L}_f + \mathcal{L}_g \tag{6}$$

where $\mathcal{L}_t$ is tensor decomposition loss, $\mathcal{L}_{sd}$ is spike-and-dip reducing loss, $\mathcal{L}_f$ is forecasting loss, and $\mathcal{L}_g$ is Gaussian smoothing loss. First, the tensor decomposition loss equation is shown below.

$$\mathcal{L}_t = (\mathcal{X} - \hat{\mathcal{X}})^2 \tag{7}$$

where $\mathcal{X}$ represents the original tensor data and $\hat{\mathcal{X}}$ represents the reconstructed tensor. This loss is to ensure accurate tensor decomposition. Second, the Gaussian kernel smoothing loss $\mathcal{L}_g$ constructs a Gaussian kernel over the original temporal tensor.

$$\mathcal{L}_g = \sum_{i_t=1}^{I_t} (\mathbf{a}_{i_t}^{(t)} - \tilde{\mathbf{a}}_{i_t}^{(t)})^2 \tag{8}$$

where

$$\mathbf{a}_{i_t}^{(t)} = \sum_{l \in L} \phi(l)\mathbf{a}_{i_t+l}^{(t)} \tag{9}$$

and $\phi(l)$ is Gaussian kernel function that returns the weight of each data point. It is defined as

$$\phi(l) = e^{\frac{-l^2}{2\sigma^2}} \tag{10}$$

In equations 3,4, and 5, $\mathcal{L}$ is a set of indices, l, centered around zero, that decides the kernel size. The intuition behind Gaussian smoothing is that a point has higher relevance to points nearby than those far away. Gaussian kernel function will return Gaussian weights at $2l-1$ indices around the center. At point i_t, $2l-1$ points around i_t will be multiplied by the Gaussian weights, resulting in the final smoothed vector $\tilde{\mathbf{a}}_{i_t}^{(t)}$. This loss reduces noise and smooths the data.

However, as discussed in the earlier sections, Gaussian smoothing does not distinguish between anomalies and real data, and gives same weights to both of them. However, highly noisy data with lots of anomalies results in poor performance in forecasting. Therefore, the spikes-and-dips reducing loss intends to detect anomalies from data and reduce their effects so that they are dampened in the final smoothed data. The intuition behind this loss is that points that show high rate of change at a short period of time is likely to be an anomaly, or unreliable information. The spikes-and-dips reducing loss is shown below.

$$\mathcal{L}_{sp} = \sum_{r=1}^{R} \sum_{i_t=1}^{I_t-1} [a_{i_t,r}^{(t)} - (1 - \mathcal{M}(i_{t+1,r}, i_{t,r}))^2 a_{i_t,r}^{(t)}]^2 \tag{11}$$

where

$$\mathcal{M}(i_{t+1,r}, i_{t,r}) = \frac{(m_{i_t+1,r} - m_{i_t,r})}{\Sigma m} \tag{12}$$

In equations 6 and 7, $m_{i_t,r}$ indicates moving average at point i_t of column r. In summary, $\mathcal{M}(i_t, i_{t+1})$ calculates rate of change of values between i_t and $i_t + 1$. The higher the value of $\mathcal{M}(i_t, i_{t+1})$ is, the greater the rate of change is, meaning that the points are highly likely to be anomalies. Once the rates of changes are determined in equation 7, we give small weights to the point that shows high rate of change from the previous point.

Forecasting loss is based on Holt-Winters exponential smoothing method. The loss is

$$\mathcal{L}_f = \sum_{r=1}^{R} \sum_{i_t=1}^{I_t-1} [a_{i_t,r}^{(t)} - \mathcal{F}(i_t, r)]^2 \tag{13}$$

where

$$\mathcal{F}(i_t, r) = (L_{i_t,r} + b_{i_t,r}) S_{i_t+1-s,r} \tag{14}$$

In equation 8, $\mathcal{F}(i_t, r)$ is the forecast at time i_t of column r of the temporal factor. The forecast value is determined by using equation 9 where $L_{i_t,r}$ is level and $b_{i_t,r}$ is trend and $S_{i_t+1-s,r}$ is seasonality at time i_t of column r. These coefficients are initialized and calculated for all time steps before training. The initialization of the coefficients are described in detail in Appendix A.1. Once they are initialized, their values for every time step can be calculated recursively using the following equations.

Level

$$L_{i_t,r} = \alpha(a_{i_t,r} - S_{i_t-s,r}) + (1 - \alpha_r)(L_{i_t-1,r} + b_{i_t-1,r}) \tag{15}$$

Trend

$$b_{i_t,r} = \beta(L_{i_t,r} - L_{i_t-1,r}) + (1 - \beta) b_{i_t-1,r} \tag{16}$$

Seasonality

$$S_{i_t,r} = \gamma(a_{i_t,r} - L_{i_t,r}) + (1 - \gamma) S_{i_t-s,r} \tag{17}$$

Forecast

$$F_{i_t+k,r} = (L_{i_t,r} + k b_{i_t,}) S_{i_t+k-s} \tag{18}$$

In equations 10 13, the weights α, β, and γ decides the decay rate of importance of previous values. If they are set high, previous events have more impact on recent ones and if they are set low, they have small impact on recent events. These weights can be optimized iteratively so that $\mathcal{F}(i_t, r)$ gives more accurate forecasts.

In summary, the total loss for the proposed method is as follows :

$$\mathcal{L} = (\mathcal{X}-\hat{\mathcal{X}})^2+\sum_{i_t=1}^{I_t}(\mathbf{a}_{i_t}^{(t)}-\tilde{\mathbf{a}}_{i_t}^{(t)})^2+\sum_{r=1}^{R}\sum_{i_t=1}^{I_t-1}[a_{i_t,r}^{(t)}-(1-\mathcal{M}(i_{t,r},i_{t+1,r}))^2 a_{i_t,r}^{(t)}]^2+\sum_{r=1}^{R}\sum_{i_t=1}^{I_t-1}[a_{i_t,r}^{(t)}-(L_{i_t,r}+b_{i_t,r})S_{i_t+1-s,r})]^2 \tag{19}$$

It is a combination of tensor decomposition loss, Gaussian kernel smoothing loss, spikes-and-dips reducing loss, and Holt-Winter forecasting loss.

4 Experiments

Table 2: Real life tensor data with time dimension.

Name	Size	Time Dimension	Nonzero	Granularity
Beijing Air Quality	$35,064 \times 12 \times 6$	35,064	2,454,305	1 hour
Madrid Air Quality	$2,678 \times 24 \times 14$	2,678	337,759	1 day
Radar Traffic	$17,937 \times 23 \times 5$	17,937	495,685	10 minutes
Indoor Condition	$19,735 \times 9 \times 2$	19,735	241,201	10 minutes
Server Room	$3 \times 3 \times 34 \times 4,157$	4,157	1,009,426	1 second

We implemented our method using Beijing air quality dataset. The data is $35064 \times 12 \times 6$ where each mode is time, locations, and pollutants. Each column of temporal factor was optimized using equation 19. Figure 1 below shows the effect of optimization. It clearly shows that the orange line is spiky compared to the original blue line.

4.1 Evaluation Metric

We evaluate the performance of the proposed method and competitors using RMSE (Root Mean Squared Error).

$$RMSE = \sqrt{\frac{1}{\|\mathcal{X}\|}(\mathcal{X}-\hat{\mathcal{X}})^2} \tag{20}$$

4.2 Experiment Setting

For each row of the given temporal factor, we divide it into several data samples of length w, the number of time steps to consider for forecasting. For example, for a window length w and a row of temporal factor of length K, we divide it into several 1D array from index 0 to index w, index 1 to index $w+1$, and index $K-w$ to index K. Then, with the data samples, we divide them into train, validation, and test sets. To avoid bias among the three sets, we distribute the samples randomly to each of those sets with a ratio of 8:1:1.

5 Related Works

There have been several studies to forecast tensor values effectively. Among them, Ramos de Araujo et al[2] used a context tensor that gives extra information about the target tensor and couple-decomposed them to obtain more insightful factor matrices. Then, Holt-Winters forecasting was applied on temporal factor to obtain future values. To increase accuracy of the forecast, the authors chose only top-k most reliable forecast values. This way, they could avoid having to reconstruct the entire tensor. However, one of its downsides

is that one needs a context tensor to reproduce the performance shown in the paper. Also, applying Holt-Winters on raw temporal factor that contains lots of noise and anomalies leads to poor forecast performance. Yu et al[4] took auto-regressive approach, learning weights of different time steps in the past and predicting future values based on past values and their weights. This method proved to be highly fast and efficient. The method we propose in this paper pre-processes temporal factor to maximize forecast ability and learns Holt-Winter forecasting at the same time.

6 Conclusions

The main objective of the proposed method was the following:

- distinguish between anomaly and real data so that the model can learn reliable information only
- maintain high tensor decomposition accuracy
- combine smoothing and forecasting in one loss function and optimize altogether at once

As shown in the Experiments section, the proposed method was able to reduce spikes and dips, removing anomalies. Also, by optimizing forecasting loss together with temporal factor smoothing loss, the proposed method avoided of having to learn multiple loss functions one after another. Performing Gaussian smoothing and spikes-and-dips reducing on temporal factor improved the overall forecasting accuracy but it needs further research to investigate if the improvement is indeed the result of anomaly control. Potential follow-up research can be done on learning dynamic smoothing parameters of Holt-Winters exponential smoothing to refine forecasting and using non-negative CP decomposition.

References

[1] Dawon Ahn, Jun-Gi Jang, and U Kang. Time-aware tensor decomposition for missing entry prediction. *CoRR*, abs/2012.08855, 2020.

[2] Miguel Ramos de Araujo, Pedro Manuel Pinto Ribeiro, and Christos Faloutsos. Tensorcast: Forecasting with context using coupled tensors (best paper award). In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 71–80, 2017.

[3] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009.

[4] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

A Appendix

A.1 Holt-Winters Forecasting Coefficients Initialization

Holt-Winters Forecasting method can be accomplished either by adding or multiplying its coefficients. We used multiplicative Holt-Winters Forecasting method and initialization equations for it are shown below.

$$\mathcal{L}_{0,r} = (a_{0,r} + a_{s,r} + a_{2,r} + \dots) \tag{21}$$

$$b_{0,r} = e^{\frac{\ln(\frac{1}{s}\Sigma_{i_t=0}^{I_t} a_{i_t,r}) - \ln(\frac{1}{s}\Sigma_{i_t=0}^{I_t-s} a_{i_t,r})}{s}} \tag{22}$$

$$S_{0,r} = [\frac{a_{0,r}}{L_{0,r}}, \frac{a_{1,r}}{L_{0,r}}, \frac{a_{2,r}}{L_{0,r}}, \dots, \frac{a_{s-1,r}}{L_{0,r}}] \tag{23}$$

A.2 Holt-Winters Forecasting Coefficients Equation

Level

$$L_{i_t} = \alpha(a_{i_t} - S_{i_t-s}) + (1-\alpha_r)(L_{i_t-1} + b_{i_t-1}) \tag{24}$$

Trend

$$b_{i_t} = \beta(L_{i_t} - L_{i_t-1}) + (1-\beta)b_{i_t-1} \tag{25}$$

Seasonality

$$S_{i_t} = \gamma(a_{i_t} - L_{i_t}) + (1-\gamma)S_{i_t-s} \tag{26}$$

Forecast

$$F_{i_t+k} = (L_{i_t} + kb_{i_t})S_{i_t+k-s} \tag{27}$$