MATH 4999: Capstone

Estimating Missing Value on Image Data Using Simple Low Rank Tensor Completion

Supervising Professor: Prof. Cai Jian Feng

Student: Kirsteen Ng Yeong Li

Introduction

Estimation of missing values is often a necessary process in ensuring the completeness of data and this process is frequently used in image processing and machine learning application. In a 2-dimensional matrix, a holistic approach in estimating missing values is by using global information of the matrix to predict the value of the missing entries. The rank of a matrix is a good tool as it captures the global information and low rank approximation is a common estimation method for missing entries in a matrix [3].

Researchers have tried to solve missing values on tensor, which is a higher dimensional matrix, by generalizing the low rank matrix approximation method. However, there are few problems with using low rank approximation in estimating missing values in tensors. 1. Rank function is non-convex and it becomes NP hard as the size of matrix grows. 2. Low-rank approximation is a non-convex optimization problem. 3. Sparsity of a matrix with missing values puts a constrain on determining the rank of the matrix. 4. Prior to Liu et al there is no definition to the rank of a tensor.

So, it can be seen even in a 2-dimensional case, low rank approximation is already a challenging task due to non-convexity of rank function. Fortunately, recent research suggests a simpler way to reduce rank function to a convex optimization problem using trace norm. Recht et al found that the trace norm of a matrix gives a tight convex envelope to the rank of matrix [4]. Thus, the minimum rank solution can be recovered by solving a convex optimization problem involving minimizing trace norm over an affine space.

Now seeing that the non-convexity problem has been solved using trace norm, there is still a challenge of rank definition in tensor. In this regard, Liu et al have proposed to define the rank of a tensor as the convex combination of the trace norm of all the matrices unfolded in a direction along each mode [2]. That has significantly bridged the low rank matrix approximation method in solving missing values in tensors. Further details on the connection between two concepts are included in Section 2 of this paper.

There are a few algorithms in estimating missing values in tensor developed by Liu et al. This project aims to replicate and understand one of the algorithms which is the Simple Low Rank Tensor Completion (SiLRTC) algorithm proposed by Liu et al [2]. SiLRTC will be implemented on a new set of experiment samples in this report. This paper is organized as the followings. Section 2: Background and Algorithm will discuss the background and the important component of SiLRTC. Section 3: Experiment Procedure will describe the process of preparing the sample set and executing the image recovery process. Section 4: Data Analysis will discuss the result and evaluate the error rate of SiLRTC.

Section 2: Background and Algorithm

Before moving on to SiLRTC, this section will first lay out the optimization problem for estimating missing values in tensors. In the low rank approximation method for matrix, the function that is to be minimized is the cost function that measures the fit between a given matrix (the data) and an approximating matrix (the optimization variable), subject to a constraint that the approximating matrix has reduced rank [3]. Essentially the minimization problem is shown below:

$$\begin{aligned} &min_x\colon rank\ (X) \\ &s.t. \mathrm{X}_\Omega = M_\Omega \end{aligned} \tag{1}$$

where X and M are pxq matrices and elements of M in the set Ω are given while the ones outside of Ω set is missing. However, there is an easier way to factorize matrix and rank function can be recovered using trace norm [4]. Therefore, Equation 1 is reformulated into a convex optimization problem in Equation 2.

$$\begin{aligned} &min_\chi\colon ||\mathrm{X}|| * \\ &s.t. \mathrm{X}_\Omega = M_\Omega \end{aligned} \tag{2}$$

where $||\mathrm{X}|| *$ is the trace norm of the matrix X. Researchers generalized the case of low rank matrix approximation to tensor and hence propose to generalize Equation 2 to a tensor. Therefore, in order to estimate missing values in a tensor, we are essentially trying to solve the following optimization problem:

$$\begin{aligned} &min_\chi\colon ||\chi|| * \\ &s.t. \chi_\Omega = \mathcal{T}_\Omega \end{aligned} \tag{3}$$

where $\chi, \mathcal{T}$ are n-mode tensors with identical size in each mode. However, as mentioned in the introduction earlier there is no proper definition of tensor rank and hence we could not directly apply the trace norm matrix factorization on a tensor.

Recently, Liu et al has defined the tensor rank as the convex combination of all matrices unfolded along each direction in the tensor and the tensor rank is described in the following equation.

$$\|\chi\| * := \sum_{i=1}^{N} \alpha_i \left\|\chi_{(i)}\right\| *$$

where, $\alpha_i' s$ are constants which satisfies $\alpha_i \geq 1$ and $\sum_{i=1}^{N} \alpha_i = 1$ and $\chi_{(i)}$ is the combination of all matrices unfolded along the i-th mode. By combining the definition of tensor rank and Equation 3 we get the following optimization problem:

$$\begin{aligned} &min_{\chi}: \textstyle\sum_{i=1}^{N} \alpha_i \left\|\chi_{(i)}\right\| * \\ &s.t. \chi_{\Omega} = \mathcal{T}_{\Omega} \end{aligned} \tag{4}$$

However, one problem during optimizing Equation 4 is that the trace norm along each direction is interdependent. As we are trying to optimize the sum of the trace norm, most of the matrix have same entries and therefore cannot be optimized independently.

Liu et al proposes to adopt block coordinate descent to locally optimize a block of matrices while keeping the rest constant. Essentially the tensor χ is divided into n+1 blocks: $\chi, \mathrm{M}_1, \mathrm{M}_2, \ldots, \mathrm{M}_n$. Now we can represent Equation 4 as the followings:

$$\chi_{i_1, i_2, \ldots, i_n} \begin{cases} \left(\dfrac{\sum i\, \beta_i\, fold(M_i)}{\sum i\, \beta_i}\right)_{i_1, i_2, \ldots, i_n} & (i_1, i_2, \ldots, i_n) \notin \Omega \\ \mathcal{T}_{i_1, i_2, \ldots, i_n} & (i_1, i_2, \ldots, i_n) \in \Omega \end{cases} \tag{5}$$

where β_i is certain positive value. Now we come to the question of how to compute M_i. This problem has been to appear in closed form in [1] and thus the optimal M_i can be computed using $D_\tau\big(\chi_{(i)}\big)$ where $\tau = \frac{\alpha_i}{\beta_i}$.

$D_\tau(\chi_{(i)})$ is the singular value shrinkage operator that applies the extension of soft-thresholding rule for vectors onto the singular value matrix of $\chi_{(i)}$. The singular value matrix of $\chi_{(i)}$ is obtained through singular value decomposition (SVD). Consider the SVD of $\chi_{(i)}$ of rank r.

$$\chi_{(i)} := U\Sigma V^*$$

where $\Sigma = diag(\{\sigma_i\}_{1\le i\le r})$. The singular shrinkage operator applies the soft thresholding rule onto Σ:

$$D_\tau(\chi_{(i)}) := UD_\tau(\Sigma)V^*, \tag{6}$$

where $D_\tau(\Sigma) = diag(\{(\sigma_i - \tau)_+\}), \tau \ge 0$.

Therefore, the shrinkage operator is reducing the singular values of $\chi_{(i)}$,in the case of tensor, and essentially reducing the rank of the combined matrices along the i-th node of the tensor. If many singular values of $\chi_{(i)}$ is below the threshold τ, then the rank of $D_\tau(\chi_{(i)})$ will be lower than $\chi_{(i)}$. This in turns allows researchers to define M_i using the singular value shrinkage operator. Now that we have the components of SiLRTC and the pseudocode is being laid out in Algorithm 1.

Algorithm 1 SiLRTC: Simple Low Rank Tensor Completion

```
Input: $\chi\ with\ \chi_\Omega = \mathcal{T}_\Omega, \alpha_i, \beta_i' s, and\ K$
Output: $\chi$
    **for** k =1 to K do
  **for** i = 1 to n do
        $M_i = D_{\frac{\alpha_i}{\beta_i}}(\chi_{(i)})$
  **end for**
  update $\chi$ in Equation 5
**end for**
```

Section 3: Experiment Procedures

In this project, a new set of sample images is being used to recover using Algorithm 1. The sample images are located at *Report/Data/ Original*. This process is done by dividing a video into 50 Frames per second using the code in *Report/Code/gen_images.m*. Consequently, more than 350 images have been generated from a 1 minute video. As the recovery process is computationally expensive, selected 20 images that can represent the flow of the video were selected and these images are located at *Report/Data/Images/Experiment*. Some pixels from the 20 images were removed by adding the words 'New York 2015' using the code in *Report/Code/remove.m* and the following illustrates the effect before and after the removal. The 20 edited images are stored in the folder *Report/Data/Images/Edited*.

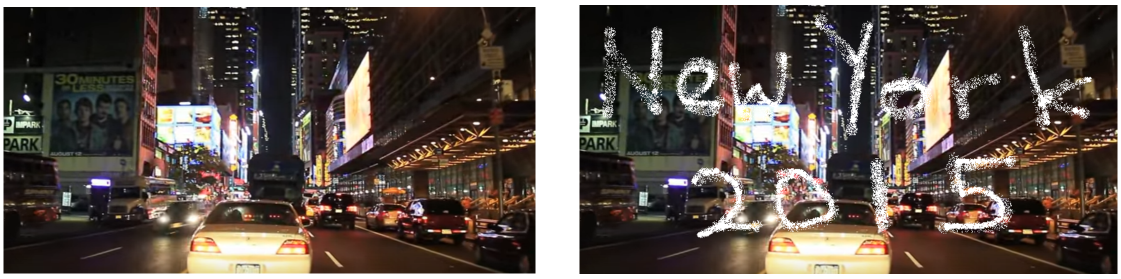

Figure 1: The effect of before and after adding the words 'New York 2015'

These 20 images are being recovered using the SiLRTC MATLAB function on a 4GB Ram 64-bit Windows 10 Enterprise Desktop. The SiLRTC function is in *Report/Code/SiLRTC.m* while the main code that runs the experiment is one *Report/Code/mytest.m*. The recovered images are stored in *Report/Data/Rec.*

Throughout the process of recovering the images, different approaches have been tried to improve the computation time. Firstly, SiLRTC were being tried on different dimensions of the image. Initially SiLRTC was applied onto the entire image of size 480x854x3. However, the process of recovering one image took approximately **3,660 seconds**. Therefore, the image was divided into various dimensions. It is found that dividing the image into **eight 120 x 427 x 3** blocks gives the most optimum result in terms of computation time and error rate. The details will be discussed in Section 4. Secondly, different values of α, β have been tried to find the most optimum combination to estimate missing values in tensors. It is found that the combination of $\alpha = [2,2,2], \beta = [0.1,0.1,0.1]$ gives the most optimum results and the details will be discussed in the following section.

Section 4: Data Analysis

This section will discuss the error rate of SiLRTC by comparing the recovered images from both approach (applying on whole block or individual blocks) with the original image. The computation time was measured using the *tic* and *toc* function before and after the main loop of SiLRTC in MATLAB (*Path: Report/Code/mytest.m*). The error rate was computed by finding the norm of the difference tensor between the original and recovered image. The detailed procedure is listed on line 63-68 in *mytest.m*.

Before applying SiLRTC on all 20 images, 5 images were being experimented on with different dimensions to decide on the most optimum dimension and α, β values. In the first attempt SiLRTC was applied directly on a 480x854x3 size image with $\alpha = [1,1,1], \beta = [0.1,0.1,0.1]$. The average computation time took **3660 seconds** per image and the error rate is **4.2335**. The second approach to this problem was by dividing the 480x854x3 image into various dimensions while keeping other parameters constant. The following table shows the average computation time and error rate using different dimensions.

Dimension	Average Computation time(sec)	Average Error rate
[480x854x3]	3,660	4.2335
[240x854x3]	2,092	6.0204
[120x854x3]	1,599	6.5476
[480x427x3]	2,127	5.0918
[240x427x3]	1,221	5.3366
[120x427x3]	**455**	**5.6923**
[120x61x3]	869	5.3867

Table 1: Various computation time and error rate on different dimensions

Eventually it is decided to divide the image into eight **120 x 427 x 3** blocks as this shows the most optimum result. This approach has shown a substantial decrease in computation time. On average the computation time is **455** seconds per image and the error rate is **5.6923**. Next, by keeping the dimension of the image constant at **120 x 427 x 3** different values of α, β were applied to decrease the error rate.

α	β	Average Computation time(sec)	Average Error rate
1,1,1	0.1,0.1,0.1	455	5.6923
1,1,1	0.5,0.5,0.5	484	29.4629
0.5,0.5,0.5	1,1,1	488	56.9765
2,2,2	0.5,0.5,0.5	467	12.6025
2,2,2	0.2,0.2,0.2	464	5.6923
2,2,2	**0.1,0.1,0.1**	**457**	**5.4943**
3,3,3	0.1,0.1,0.1	450	5.5417

Table 2: Various computation time and error rate on **120 x 427 x 3** with different α, β values.

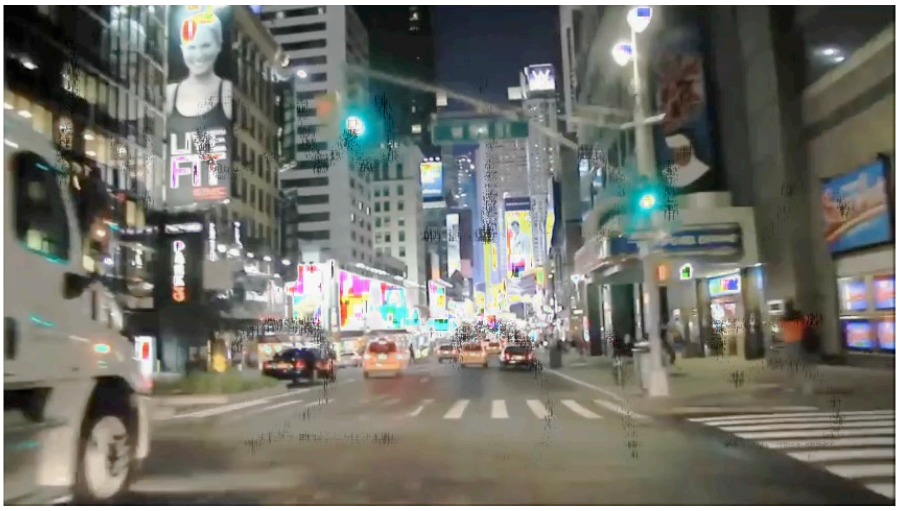

Figure 2: Recovered image at $\alpha = [1,1,1], \beta = [0.5,0.5,0.5]$.

Based on the result from Table 1 and Table 2, it is decided to set the following parameters on the SiLRTC for all images.

- Dimension: **120 x 427 x 3**
- α : [2,2,2]
- β : [0.1,0.1,0.1]

All the images are recovered using the above parameters and are stored at *Report/Data /Rec.*

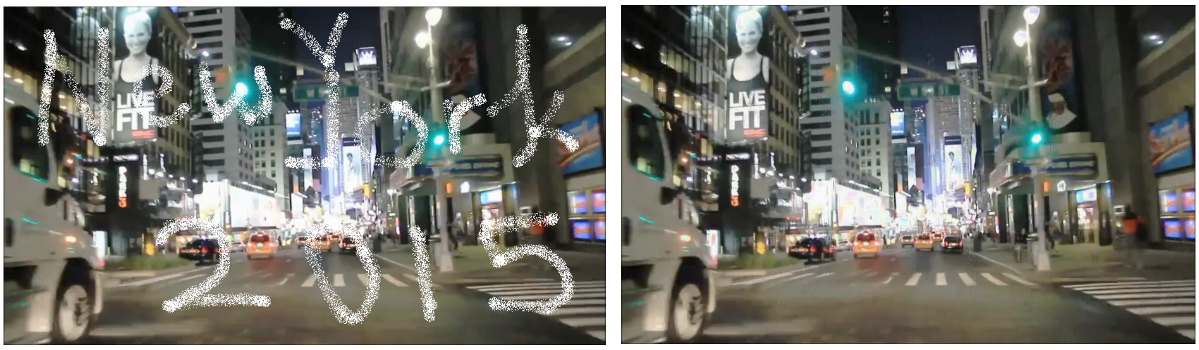

Figure 3: Before and after image recovery

Conclusion

Overall, SiLRTC has been proven to be an effective way to recover missing values in a tensor. One of the biggest breakthroughs that Liu et al have achieved is defining the rank of a tensor and have found an elegant way to generalize low rank matrix approximation onto a tensor.

This project mainly focuses on implementing Simple Low Rank Tensor Completion on a new set of sample images and recovering the images using various dimensions and parameter values. It is found that SiLRTC can be more efficient if the algorithm is being applied on blocks of various dimensions and different values of parameters. Nevertheless, the relationship between the two factors and accuracy and computation time are not well studied and justified. Therefore, in the future, I would extend my work in these areas. At the same time, I will explore other tensor completion algorithm.

References

1. Cai, J.-F., Candes, E. J., & Zuowei, S. (2010). A Singular Value Thresholding Algorithm For Matrix Completion. *SIAM J. Optim, Vol 2, No.4*, 1956-1982.

2. Liu, J., Musialski, P. M., Wonka, P., & Jieping, Y. (2012). Tensor Completion for Estimating Missing Values in Visual Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 35, Issue: 1)*, 208 - 220.

3. Markovsky, I., & Usevich, K. (2013). Structured Low-rank Approximation With Missing Data. *SIAM Journal on Matrix Analysis and Applications, Volumn 34, Issue 2*, 814-830.

4. Recht, B., & Fazel, M. P. (2010). Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Numclear Norm Minimization. *SIAM Review, Volume 52, No.3*, 471-501.