Robust Principle Component Analysis

School Of Engineering And Applied Science, Ahmedabad University
Report By Harsh Mehta - 1401086
harshmehta87.1997.hm@gmail.com

***Abstract*—This report is about a Principle component analysis.Principle component analysis is a fundamental operation in computational data analysis, with application ranging from web search to bioinformatics to computer vision. This report is based on idealized "robust principal component analysis" problem of recovering a low-rank matrix from corrupted matrix. Suppose we have a data matrix, which is the superposition of a low-rank component and a sparse component. Under the specific assumption we can prove that we can recover the low-rank and sparse matrix exactly by solving a very convenient convex program called Principal Component Pursuit. Results of this paper assert that one can recover the principal components of a data matrix even though a positive fraction of its entries are arbitrarily corrupted. It extends to the situation where a fraction of the entries are missing as well. We can use this algorithm for detection of objects in a cluttered background, and in the area of video surveillance, where it offers a principled way of removing shadows and specularities in images of faces.**

I. Introduction

We are given a data matrix M and it is decomposed into two matrices, the first is low rank matrix and the other is sparse matrix. Matrices are represented as

$$M = L_0 + S_0$$

where L_0 is a low-rank matrix and S_0 is a sparse matrix. Here, we do not know the row space and dimensionality of L_0. Likewise, we do not know the sparsity of the S_0 matrix. Our objective is to recover both low-rank matrix and sparse matrix accurately and efficiently. In many application data lie in high dimensional space. To reduce the effect of dimension, the most useful assumption is that the data lie on low-dimensional subspace that is data matrix should have low-rank. We can represent this in a mathematical term as

$$M = L_0 + N_0$$

where L_0 is low-rank matrix and N_0 is Independent and Identically Distributed Gaussian Noise. Classical Principal Component Analysis finds the best rank-k estimation of L_0 by solving

$$\begin{aligned} &\text{minimize} \quad ||M - L|| \\ &\text{subject to } rank(L) \leq k \end{aligned}$$

PCA is widely used statistical tool for data analysis and dimension reduction. However, PCA fails when large number of data are corrupted. Practically, errors are present everywhere in todays application. The problem we are given here can be considered an idealized version of Robust PCA, in which our goal is to recover low-rank matrix L_0. Unlike the small noise term N_0 in classical PCA, the entries in S_0 can have arbitrarily large magnitude, and their support is assumed to be sparse but unknown. Under the weak assumptions, the PCP exactly recovers the L_0 and S_0 by solving

$$\begin{aligned} &\text{minimize } ||L||_* + \lambda||S||_1 \\ &\text{subject to} \quad L + S = M \end{aligned}$$

II. Application

There are many important applications in which the data can naturally be modeled as a low-rank and sparse component and depending on the application either the low rank or sparse component could be the object of interest.

A. Video Surveillance

Given a sequence of a video frames and we often need to find out activities that stand out from the background. If we stack the video frames as the column of the matrix M, then we can represent the stationary background as a low-rank component L_0 and the moving object in the foreground as sparse component S_0.

B. Latent Semantic Indexing

Web search engines often need to analyse and index the content of large collection of documents. The basic idea is to gather a document-versus-term matrix M whose entries typically encode the relevance of a term or a word to a document such as the frequency it appears in the document. If we were able to decompose M as a sum of a low-rank component L_0 and a sparse component S_0, then L_0 could capture common words used in all the documents while S_0 captures the few key words that best distinguish each documents.

C. Netflix

Netflix is an application which provides movies on demand. Users of this application will rate the movies based on their likes and dislikes. PCP is used in this application to provide good suggestion to their users.Based on the given ratings PCP will predict the ratings for other movies and provide suggestion to their users.

III. Organization of the report

The report is organized as follows. In section 4 we are justifying and introducing our assumptions for the algorithm. Section 5 is about notations used in this report and its brief explanation. In Section 6 we are explaining Theorem 1.1 and Algorithm to find Principal Component Pursuit solution when M is of very large scale. We conclude the report in Section 6.

IV. Assumptions

We are given a Data Matrix M which is both sparse and low-rank. Our aim is to identify whether it is sparse or low-rank. So we assume that low-rank component is not sparse. Under the assumption that rank of the low-rank component is not too large and the sparse component is sensibly sparse, PCP can perfectly recovers the low-rank component. Here, we will assume that sparsity of a sparse matrix is uniformly random. Singular value decomposition of L_0 as

$$L_0 = U \sum V^* = \sum_{i=1}^{r} \sigma_i u_i v_i^*$$

Our first assumption that low-rank matrix is not sparse will true if

$$\max_i ||U_* e_i||^2 \leq \frac{\mu_r}{n_1}, \max_i ||V_* e_i||^2 \leq \frac{\mu_r}{n_2} \tag{1}$$

$$||UV^*||_\infty \leq \sqrt{\frac{\mu_r}{n_1 n_2}} \tag{2}$$

L_0 obeys equation (1) and (2).

V. Notation

In this article we used five norms of a matrix. (1) Nuclear norm denoted by$||M||_*$,that is sum of all the singular values of M. So,$||M||_* = \sum_i \sigma(M)$. (2) Operator norm or 2-norm is a square root of a largest singular value of M which is denoted by $||M||$. (3) Frobenius norm is a matrix norm defined as the square root of the sum of the absolute squares of its element, that is denoted by $||M||_F$. (4) l_1 norm is the sum of the absolute values of its element. (5) l_∞ norm is the maximum absolute value of a matrix.

VI. Algorithm

Theorem 1.1 says that, suppose L_0 is a matrix of size $n_1 \times n_2$, obeys (1) and (2) then there is a numerical constant c such that with probability at least $1 - cn_1^{-10}$, PCP with $\lambda = 1/\sqrt{n_1}$ is exact, that is, $\hat{L} = L_0$ and $\hat{S} = S_0$ provided that

$$rank(L_0) \leq \rho_r n_2 \mu^{-1} (log n_1)^{-2} \; and \; m \leq \rho_s n_1 n_2$$

Under the assumption, minimizing

$$||L||_* + \frac{1}{n_1}||S||_1, \quad n_1 = max(n_1, n_2).$$

always return the correct answer. Our algorithm for solving the convex PCP problem works on augmented Lagrange multiplier (ALM) introduced in Lin et al. [2009a] and Yuan and Yang [2009]. The ALM method is based on the augmented Lagrangian

$$l(L, S, Y) = ||L||_* + \lambda||S||_1 + \langle Y, M - L - S\rangle + \frac{\mu}{2}||M - L - S||_F^2 \tag{3}$$

This algorithm will solve the PCP by updating $(L_k, S_k) = argmin_{L,S} l(L, S, Y_k)$ and then updating the lagrange multiplier via $Y_{k+1} = Y_k + \mu(M - L_k - S_k)$. For our low-rank and sparse decomposition, we will find $min_L l(L, S, Y)$ and $min_S l(L, S, Y)$ instead of solving sequence of convex programs. In our algorithm, first we are minimizing l with respect to L (fixing S)

$$\arg\min_L l(L, S, Y) = D_{\frac{1}{\mu}}(M - S + \mu^{-1}Y) \tag{4}$$

and then we are minimizing l with respect to S (fixing L)

$$\arg\min_S l(L, S, Y) = D_{\frac{1}{\mu}}(M - S + \mu^{-1}Y) \tag{5}$$

and then finally we are updating the lagrange multiplier matrix Y based on the residual M-L-S. So outline of this strategy is described as Algorithm 1.

Algorithm 1 Principal Component Pursuit by Alternating Directions [Lin et al. 2009a;Yuan and Yang 2009]

1: **Initialize**: $S_0 = Y_0 = 0, \mu > 0$
2: **while** not converged **do**
3: compute $L_{k+1} = D_{\frac{1}{\mu}}(M - S_k + \mu^{-1}Y_k)$;
4: compute $S_{k+1} = S_{\frac{\lambda}{\mu}}(M - L_{k+1} + \mu^{-1}Y_k)$;
5: compute $Y_{k+1} = Y_k + \mu(M - L_{k+1} - S_{k+1})$;
6: **end while**
7: **Output:** *L,S.*

In this algorithm, we simply choose $\mu = n_1 n_2 / 4||M||_1$, as suggested in Yuan and Yang [2009]. And the termination condition for our algorithm is $||M - L - S||_F \leq \delta||M||_F$, with $\delta = 10^{-7}$

VII. Conclusion

The paper shows that for incoherent L_0, correct recovery occurs with high probability for rank(L_0) on the order of $n/[\mu log^2 \, n]$ and a number of nonzero entries in S_0 on order of n^2. That is, matrices of large rank can be recovered from non-vanishing fraction of sparse errors. We can extract the low-rank and sparse components exactly by convex programming. This article is showing close relationship between matrix completion and matrix recovery. In general, we can recover low-rank and sparse component when there are both incomplete and corrupted entries.

References

[1] Cands, E. J., & Recht, B. (2009). Exact Matrix Completion via Convex Optimization. Foundations of Computational Mathematics, 9(6), 717-772. doi:10.1007/s10208-009-9045-5

[2] Cands, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis? Journal of the ACM, 58(3), 1-37. doi:10.1145/1970392.1970395