18660 Project Progress Report **22 Apr 2018**

Comparison of robust PCA techniques in image and video processing

Joel Loo, Wang Xueqiang

1 Introduction

Principal components analysis (PCA) and other dimensionality reduction techniques have gained wide traction in the scientific community because of their utility in extracting salient data from large and complex datasets. However an implicit assumption of PCA is that the noise in the data is small and variance from the mean is bounded. Hence, PCA works well on data that is corrupted by small amounts of noise (e.g. data corrupted by Gaussian noise), but can perform poorly on grossly corrupted data and data containing significant outliers.

While there have been many attempts to robustify PCA, many of these methods are heuristics without any optimality guarantees. However, recent advances in low-rank matrix recovery and completion by Candes et al [1], have led to the formulation of robust PCA (RPCA). In RPCA, the problem is reformulated as the recovery of the matrices L, S such that $M = L + S$, where M is the original matrix, L is a low-rank matrix and S is a sparse matrix. Candes et al made the surprising observation that it is possible to perform this decomposition through a tractable convex optimization, if we assume that the low-rank matrix is not simultaneously sparse [1]. In particular, this can be achieved using the Principal Component Pursuit (PCP) estimate which solves the problem

$$\underset{L,S}{\operatorname{argmin}} \quad \|L\|_* + \lambda\|S\|_1 \quad \text{subject to} \quad L + S = M$$

Since the sparse matrix can have elements of arbitrarily large magnitude, robust PCA thus improves over the PCA technique by allowing the extraction of low-dimensional structure from grossly corrupted data with unbounded outliers. This technique has many important applications in general, and in particular is of great utility in computer vision, where it is used in tackling problems such as background recovery/subtraction (which can be used in video surveillance software), removal of non-Lambertian distortions/specularities/shadows from images (akin to performing inpainting) and even recovery of 3D geometry from low-rank textures in 2D images.

In order to make RPCA practicable on real-world vision systems, a number of reformulations of the problem and improvements have been proposed. Some of these improvements (e.g. Fast PCP [3]) provide significant speed improvements over the method proposed by Candes et al, while others reformulate RPCA as an online algorithm (e.g. online RPCA via stochastic optimization [2], iFrALM).

2 Problem Statement

In this project, we aim to survey several RPCA algorithms - in particular we will implement these algorithms and compare their performance, in terms of both speed and ability to separate low-rank structure from sparse noise. We are primarily interested in applying these algorithms to, and testing them on vision applications. Two applications that we are particularly interested in are the removal of specularities/non-Lambertian distortions from images, as well as background recovery/subtraction in videos. Background recovery/subtraction in video streams provides an interesting case study for RPCA as well, since it is desirable to have both algorithms that can separate backgrounds in post-processing as well as on the fly. To this end, we will use this particular application to benchmark the performance of both RPCA algorithms that offer significant speed-ups, as well as RPCA algorithms formulated to be online in nature. We will perform the implementations and benchmarking described above in Python, with code available on the private Github repository https://github.com/joelloo/18660_project

3 Algorithm implementation and comparison

3.1 Standard RPCA algorithms

The formulation of RPCA by Candes et al aims to solve the convex problem

$$\underset{L,S}{\operatorname{argmin}} \quad \|L\|_* + \lambda\|S\|_1 \quad \text{subject to} \quad L + S = M$$

There have been several approaches taken to solving this original formulation as summarized in [4]. These include a principal component pursuit that is effectively an exact Augmented Lagrangian Multiplier (ALM) method, an inexact ALM method, and an accelerated proximal gradient approach [5].

3.2 Fast PCP

The Fast PCP method [3] reformulates the original convex PCP problem. The original formulation (solved with standard RPCA methods) tends to be rather computationally expensive as we have observed. The Fast PCP method on the other hand changes the PCP constraint $M = L + S$ to the Frobenius-norm penalty $\|L + S - M\|_F$, and changes the nuclear norm penalty to a constraint, i.e. $\|L\|_* \leq t$, for some t that is small compared to the dimensions of M. The nuclear norm constraint is reformulated as a constraint on the rank, yielding

$$\underset{L,S}{\operatorname{argmin}} \quad \|L + S - M\|_F + \lambda\|S\|_1 \quad \text{subject to} \quad \operatorname{rank}(L) = t$$

This optimization problem over two variables is easily amenable to an alternating minimization procedure $L_{k+1} = \underset{L,S}{\operatorname{argmin}} \quad \|L_k + S_k - M\|_F \quad \text{subject to} \quad \operatorname{rank}(L_k) = t$ and $S_{k+1} = \underset{L,S}{\operatorname{argmin}} \quad \|L_{k+1} + S_k - M\|_F + \lambda\|S\|_1$.
The first minimization is recognizable as a rank-constrained least squares problem, for which the canonical solution is to compute the partial SVD, i.e. $L_{k+1} = u * s * vt$, where u is the first t columns of U, s is the first square submatrix of Σ of dimension t, vt is the first t rows of V^T and $M - S_k = U\Sigma V^T$. The second minimization can be solved by direct application of the soft thresholding operator.

The algorithm derived from this reformulation of the PCP objective is exceedingly simple, and was easily implemented. To compute the partial SVD, we used scikit-learn's randomized_svd function to efficiently compute the SVD up to the estimated rank. As suggested in the paper, we also implement a check to enforce low rank in L, wherein we compute the contribution of each successive singular value, and stop increasing the rank when the singular value becomes too small. Since the algorithm consists of only two steps, where the main bottleneck is the SVD, this method is very fast in practice because we are only computing a partial SVD for only a small number of columns/rows (if we enforce low rank on L).

3.3 Online RPCA via stochastic optimization (STOC-RPCA)

The previous two methods can only be applied to video streams if we already have the entire video. In contrast, STOC-RPCA is an online algorithm that can take each incoming frame, and incrementally improve the low-rank and sparse separation. STOC-RPCA uses the original formulation of PCP, but combines the $L + S = M$ constraint into the objective as the penalty $\frac{1}{2}\|M - L - S\|_2^2$. In addition, it expresses the nuclear norm as $\inf_{L,R}\{\frac{1}{2}\|L\|_F^2 + \frac{1}{2}\|R\|_F^2 : X = LR^T\}$, where L can be thought of as a fixed basis. Substituting this back into objective expression and converting into a form that takes each column vector of R and S (corresponding to each video frame) as arguments, we have $\frac{1}{t}\sum_{i=1}^{t}(\frac{1}{2}\|\mathbf{m}_i - L\mathbf{r}_i - \mathbf{s}_i\|_2^2 + \frac{\lambda_1}{2}\|\mathbf{r}_i\|_2^2 + \lambda_2\|\mathbf{s}_i\|_1) + \frac{\lambda_1}{2t}\|L\|_F^2$. This can be solved through a alternating minimization type algorithm, where we first minimize the objective with respect to $\mathbf{r}_i, \mathbf{s}_i$, then update the basis L using these vectors.

The paper recommended solving the first minimization to find $\mathbf{r}_i, \mathbf{s}_i$ using standard available solvers. Since the first minimization is essentially a convex problem, we attempted to find a solution using CVXPY. However, this yielded poor results for reasons that we are still exploring, and the overhead from CVXPY made it relatively slower compared to our final solution. Our final solution was to take an alternating minimization method, where we would first update $\mathbf{r}_i$, then update $\mathbf{s}_i$. Since the objective with respect to $\mathbf{r}_i$ is fully differentiable, we can simply use the closed-form partial derivative to compute the minimum for a given $\mathbf{s}_i$. We then minimize $\mathbf{s}_i$ using proximal gradient descent, since it contains the nondifferentiable L1 norm. Once we have computed $\mathbf{r}_i, \mathbf{s}_i$, we can update the basis L using coordinate descent as suggested by the paper. While this might seem like a computationally intensive approach (especially since we have proximal gradient descent nested inside two outer layers of alternating minimization), we have found that in practice this approach seems to converge fast and works close to real-time. This is especially the case since we have noted that large fixed step sizes yield acceptable results and the proximal method requires few iterations to get close to convergence.

3.4 Incremental Fixed-rank RPCA

In a similar vein to STOC-RPCA, Jian et al [6] presented iFrALM (incremental fixed-rank ALM), which allows RPCA to be applied to frames in an initial batch, and also allows the low-rank and sparse separation to be updated incrementally. The algorithm will only save fixed-rank components of initial batch and the incremental batch.

iFrALM is based on an incremental SVD (iSVD) algorithm. iSVD is a method to update the matrix's SVD with newly-added part based on the original matrix's SVD, to update the matrix's SVD with newly-added columns. iFrALM implements the iSVD method presented by Zha [7]. Let QR decomposition of $(I - P_k P_k^T)$ be

$$(I - P_k P_k^T)D = \hat{P}_k R$$

where $\hat{P}_k$ is the orthonormal and the R is upper triangle, then

$$B \equiv [A_k, D] = [P_k, \hat{P}_k]\begin{bmatrix}\Sigma_k & P_k^T D \\ 0 & R\end{bmatrix}\begin{bmatrix}Q_k^T & 0 \\ 0 & I_p\end{bmatrix}$$

Then the SVD of

$$\hat{B} \equiv \begin{bmatrix}\Sigma & P_k^T D \\ 0 & R\end{bmatrix} = [U_k, U_k^{\perp}]\begin{bmatrix}\hat{\Sigma}_k & 0 \\ 0 & \hat{\Sigma}_p\end{bmatrix}[V_k, V_k^{\perp}]$$

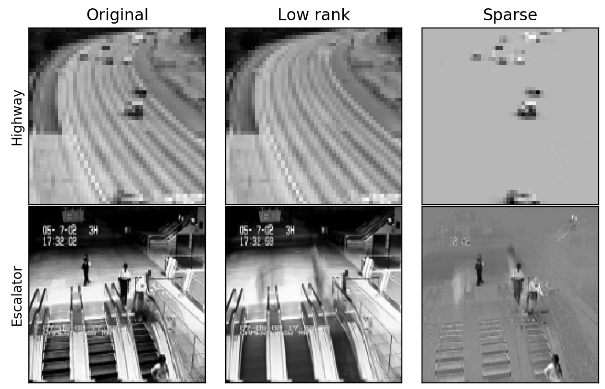
So that we get the updated incremental SVD:

$$B_k \equiv ([P_k, \hat{P}_k]U_k)\hat{\Sigma}_k(\begin{bmatrix}Q^k & 0 \\ 0 & I_p\end{bmatrix}V_k)^T$$

In iFrRPCA, D is seperated into initial batch D_0 and incremental batches $D_1, D_2, D_i, \cdots$. Then following the FrALM, we can solve each batch size and get the principle component of A: $A_{i-1} = U_{i-1}S_{i-1}V_{i-1}^T$, pass the A_{i-1} into next problem, that is to minimize E with $D = [A_{i-1}, D_i]$. In the new problem, iSVD will help us to get the new SVD update with both efficient computation and memory.

However, during the implementation, because the author did not give an explict convergence condition along with the parameters used in the algorithm, although we have built the programming structure, it takes time to test out proper parameters. Another problem during the implementation is that video frames usually have a large size.

3.5 Preliminary comparisons



4 Next steps

We have two primary goals going forward. Firstly, we want to perform more detailed benchmarking, analysis and comparison of the algorithms we have presently implemented. In particular, we want to test our algorithms on a wider variety of video streams (e.g. videos with larger images, longer videos etc.) to observe how the algorithms scale and how close the algorithms can get to real-time performance. Additionally, we could improve on the way in which we currently benchmark the quality of the low-rank and sparse separation. This is currently done by visual inspection of the results of our implementations when applied to specularity removal or background subtraction. While we are able to get a general sense of how performant our algorithm is from this, we would like to take a more quantifiable approach. We could do this by generating low-rank data and corrupting it with arbitrary sparse data and quantifying the separation by seeing if the algorithm is able to return a rank close to that of our artificial low-rank data, and also by computing the distance of the computed low-rank and sparse matrices from our artificially created low-rank and sparse matrices using the Frobenius norm or some other metric.

Secondly, we would like to explore how close we can get to real-time performance when implementing background recovery/subtraction with RPCA. While Fast PCP is fast and highly performant, it is not adapted for real-time continuous background recovery/subtraction such as in video surveillance. This is because it ultimately uses all the frames in the video to perform the low-rank extraction, which might be slow and unwieldy for the large numbers of frames present in a surveillance video. For this purpose, we might desire an online algorithm that incorporates frames as they come in, such as we have with STOC-RPCA (but which is slower than Fast PCP, because it performs more computations). One idea we have to do this would be to adapt Fast PCP such that it operates on a fixed number of frames, i.e. Fast PCP would be continuously applied to a moving window of frames as more frames comes in, giving it a continuous real-time ability, and possibly giving it the ability to adapt better to permanent changes to the background.

However, we also would like to explore the possibility of combining some aspects of Fast PCP and STOC-RPCA. Both Fast PCP and STOC-RPCA reformulate the original PCP optimization problem. The main difference between the formulations is that the nuclear norm penalty in STOC-RPCA is a rank constraint in Fast PCP. We are curious to see if we are able to reconcile these differences in the formulation and hence the algorithms, and try to reformulate the Fast PCP algorithm such that the low-rank and sparse matrix can be updated frame by frame. Indeed, Rodriguez and Wohlberg have proposed an extension to Fast PCP to enable incremental updating, that suggests the use of the incremental SVD found in iFrALM, although they have not provided implementation details or an algorithm for this. We would also like to take some time to explore the ways in which we might be able to apply the iSVD in the background recovery/subtraction algorithm to transform the Fast PCP algorithm into an online algorithm.

5 References

[1] E. Candes, X. Li, Y. Ma, and J. Wright, ”Robust principal component analysis?,” Journal of the ACM, vol. 58, no. 3, May 2011.

[2] Online Robust PCA via Stochastic Optimization (Feng, Xu and Yan, 2013). Reference: Feng, Jiashi, Huan Xu, and Shuicheng Yan. ”Online robust pca via stochastic optimization.” Advances in Neural Information Processing Systems. 2013.

[3] P. Rodriguez and B. Wohlberg, ”Fast principal component pursuit via alternating minimization,” in IEEE ICIP, Melbourne, Australia, Sep. 2013, pp. 6973.

[4] Guyon, C., Bouwmans, T., and Zahzah, E. (2012). Robust Principal Component Analysis for Background Subtraction: Systematic Evaluation and Comparative Analysis.

[5] Lin, Z., Ganesh, A., Wright, J., Wu, L., Chen, M., and Ma, Y. (2009). Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. Intl. Workshop on Comp. Adv. in Multi-Sensor Adapt. Processing, Aruba, Dutch Antilles.

[6] Jian Lai, Wee Kheng Leow, Terence Sim (2015). Incremental Fixed-Rank Robust PCA for Video Backgroud Recovery

[7] Hongyuan Zha, Horst D. Simon (1999). On Updating Problems in Latent Semantic Indexing