

UNIVERSITY OF WASHINGTON

CSEP 521 FINAL PROJECT

# Practical Applications of RPCA

*An Analysis by Jonathan Donovan & Si Latt*

March 4, 2017

# 1 Abstract

In this paper we analyze the performance of PCA and robust PCA on datasets that are artificially made noisy. In particular, we use a method[2] which attempts to decompose the corrupted matrix into a low rank and a sparse component. The low rank component can then be used to recover the principal components for the data being analyzed. We attempt this on facial data (to recover eigenfaces) and on video data (as background recovery) and examine the performance of Robust PCA for each of these applications.

## 2 Introduction

### 2.1 PCA

$$PCA(M, k) = \arg \min_{M'} (||M - M'||_F) \mid rank(M') \leq k$$

Principle Component Analysis[4] (or PCA) is the attempt to approximate a matrix  $M$  with a nearby (in the Frobenius norm) low rank matrix  $M'$ . As a reminder, the rank of a matrix  $M$  is the number of independent vectors that are needed to recreate the columns of the matrix, and the Frobenius norm of a matrix is a generalization of the euclidean norm for vectors, where  $||A||_F = \sqrt{\sum_{a_{i,j} \in A} |a_{i,j}|^2}$ .

When the columns of the matrix we are performing PCA on represent a series of measurements (usually with the average subtracted out), with PCA we are essentially trying to decompose the measurements into linear independent 'hidden variables'. If we have reason to believe that our measurements are the result of a smaller subset of independent variables, PCA gives us the tools needed to find the best set of these variables.

PCA would not be useful if it were costly to compute, but thankfully, the problem formulation listed above is intimately related to a common operation in the analysis of matrices, that is: the Singular Value Decomposition. The Singular Value Decomposition (SVD) problem is a problem with the goal of decomposing a matrix into  $M = U\Sigma V^*$ , where  $U$  and  $V$  are unitary matrices ( $UU^* = I$  and  $V^*V = I$ ) and  $\Sigma$  is a real diagonal matrix with the entries sorted. While the factorization may not be unique, the diagonal matrix is, and it can be shown that the best  $k$ -rank approximation of a matrix  $M$  can be found as  $M' = U\Sigma'V^*$  where  $\Sigma'$  is the diagonal matrix  $\Sigma$  with all entries  $\sigma_{i,i} = 0$  for  $i > k$ . It suffices here to say that the SVD problem is well studied, and that solutions

can be found quite quickly in  $O(mn^2)$  time for an  $m \times n$  matrix. Probabilistic methods exist which can perform even faster when  $k \ll m, n$ .

## 2.2 Robust PCA

Robust PCA[2] is an extension of PCA to the domain of corrupted data. PCA can quite easily recognize data where redundancy in the information of the data points exists, but it does not properly account for the fact that the collection of data itself may be lossy. One circumstance in which PCA fails is where significant outliers exist in the data, where the inclusion of a single dramatic outlier may have a dramatic impact on the estimated principal components. In a world where massive amounts of data exist to be sample, but where the quality of the data may vary significantly, a variation on PCA which incorporates robustness to these problems adds significant utility when analyzing data with the intent of discovering low rank approximations.

In the paper "Robust principal component analysis?"[2] such a method is outlined, called Principal Component Pursuit (PCP). PCP breaks down the finding of the solution into decomposing a given matrix  $M$  into two components  $L$  and  $S$  via the equation:

$$L, S = \arg \min_{L, S} (\|L\|_* + \lambda \|S\|_1) \mid L + S = M$$

where  $\|A\|_*$  is the nuclear norm, defined as the sum of the singular values of  $M$ , or  $\sum_{\sigma \in \text{SVD}(M)} \sigma$ , and  $\|A\|_1$  is the sum of the absolute values of the entries of  $A$ , or  $\sum_{a_{i,j} \in A} |a_{i,j}|$ . Amazingly, as part of the analysis the authors determine that there is a choice of parameter  $\lambda = \frac{1}{\sqrt{n}}$  that succeeds with fairly high probability. To be specific this algorithm succeeds to capture the to perfectly capture the original matrices  $L$  and  $S$  with  $P \geq 1 - \frac{c}{n^{10}}$  provided the conditions that (for an  $n \times n$  matrix)

$$\text{rank}(L) \leq \frac{p_r n}{\mu (\log n)^2} \text{ and } m \leq p_s n^2$$

Here  $p_r$  and  $p_s$  are positive constants,  $\mu$  is a measure of incoherence[3] (with small  $\mu$  it is required that the significant values are not sparse), and  $m$  is the support set of the matrix  $S_0$  (or equivalently, the number of nonzero entries in  $S_0$ ). Interestingly, this shows that the number of corrupted entries can scale linearly with the size of the matrix, and that with a fixed  $\mu$ , the rank of the low rank component  $L_0$  can grow as  $\frac{n}{\log(n)^2}$ . Another interesting point to note is that while we have constraints on the number of corrupted

entries (the support set of  $S_0$ ) we have no such constraints on how the data is corrupted (that is, it could be corrupted with arbitrarily large absolute value errors). When in fact our goal in many circumstances is to recover a fixed low rank approximation from a large amount of (probabilistically corrupted) data, these constraints work perfectly!

However one difficulty with this algorithm is that, while provably able to separate low rank from sparse data, it has not performed as successfully on real world data, frequently taking a large number of iterations to complete. An alternative algorithm is described in [6], the Inexact Augment Lagrange Multiplier method. This algorithm has no proof of correctness, but uses a relaxed form of the constraints above. by solving the lagrangian problem:

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

In addition, it uses an iterative approach towards updating the  $S$  and  $Y$  matrices that limits the number of SVD operations necessary for the algorithm to succeed. While this algorithm has not been proven correct, the fact that it is much faster in practice, and has been shown to converge in many practical demonstrations, we will use this algorithm when analyzing the performance of Robust PCA on our datasets. One important feature to note, is that it is recommended that  $\mu_0$  be set to  $\frac{1.25}{||M||_2}$ .

During the analysis section of this paper, we primarily used code obtained from [5], which is a translation of the MATLAB implementation of RPCA into numpy created by Kyle Kastner.

## 3 Potential Uses for Robust PCA

### 3.1 Facial Identification

One area we would like to investigate more thoroughly here, is the impact that Robust PCA has on the use of eigenfaces. Eigenfaces are the result of an attempt to use the low dimensional structure of facial images to create a small number of 'basis faces' that can be used to segment pictures of individuals, allowing identification of individuals based on a small number of projections. One property of image data in general that makes it quite amenable to PCA analysis is that the dimensions are naturally bounded (by 0 and 255 in most formats), meaning that no renormalization of particular dimensions needs to take place in order for PCA to find reasonable latent variables.

There is another property of facial data that makes it an attractive target for PCA: it has been shown that an idealized Lambertian convex surface (that is, a convex surface that has an ideal matte appearance) when displayed with arbitrary illuminations, lies on a 9 dimensional subspace[1]. While faces are neither perfectly Lambertian nor perfectly convex, the fact that they may be approximated as such lends credence to the idea that analyzing a particular face (or group of faces) may involve looking for low dimensional data underlying the apparently complex representation of the images themselves (which can be on the order of thousands or even millions of pixels).

Robust PCA has been shown to be capable of correctly identifying specularities in facial image data of a particular individual in the past[8]. As part of our research, we will attempt to determine whether Robust PCA lends itself to analyzing large data sets of facial data with many different lighting circumstances. If the promises of the approach are to be believed, it should be capable of recognizing the irregularities caused by these circumstances and correcting for them, allowing for more accurate eigenfaces to be collected once the sparse error component is eliminated.

### 3.2 Background / Foreground Segmentation of Videos

## 4 Metrics

The remainder of our paper will be spent analyzing the performance of Robust PCA relative to PCA on various data sets. In order to do so we would like to define a few metrics that would be useful when discussing the accuracy of Robust PCA in reproducing the original low rank matrix.

### 4.1 Frobenius error

One useful metric, and the one used by the original paper, was the metric  $\frac{\|L_0 - L\|_F}{\|L_0\|_F}$ , where  $L_0$  is the true low rank matrix, and  $L$  is the approximation that results from applying RPCA. This is a reasonable metric of the error of the approximation, as it is the sum of square errors of the approximation over the sum of squares of the values of the original matrix, giving a natural sense of 'how far' the resulting approximation lies from the true low rank matrix.

One difficulty with using this form of error, while it is the most common type used in papers discussing the Robust PCA problem, is that when using real world data, the

true low rank matrix is not known, only the original matrix  $M$ . Replacing  $L_0$  with  $M$  above results in the metric becoming  $\frac{\|E\|_F}{\|M\|_F}$ , or to restate another way, measures that the sum of the squares of the error matrix derived relative to the original matrix. This can be used to determine whether the Robust PCA algorithm has successfully found a relatively small error in it's decomposition, but does not give us much indication into how well it performed (that is, whether the low rank matrix obtained is truly indicative of the data underlying the metrics).

## 4.2 Clustering Metric

In the particular case of eigenfaces, one stated goal of the creation of the eigenfaces involved is to allow for easy recognition of particular people. As such, we could evaluate how well our eigenfaces perform based on how well they correctly cause images of the same person to be clustered close together. Let us call  $v_1..v_k$  to be the projection of a particular image onto the first to the  $k$ th eigenface generated either Robust or standard PCA, which is a natural dimensionality reduction often used to analyze PCA performance.

We can evaluate how well our data is clustered within the projection by using the silhouette width[7] for the clusters in the new projected space. Assuming we have mapped each metric into the basis provided by the  $k$  eigenfaces (let us call these mapped vectors objects  $o_1..o_m$ ), we can define our clusters  $C_1..C_n$  to be the natural clusters formed when clustering all images of the same individual.

We can define  $a(o_i)$  to be the average distance of the object  $o_i$  within it's cluster,  $b(o_i)$  to be the minimum over all other clusters of average distance to that cluster, and the silhouette distance for that object to be the difference of those measures over the maximum.

$$a(o_i) = \frac{1}{|C_{o_i}| - 1} \sum_{o_j \in C_{o_i}, o_j \neq o_i} d(o_i, o_j)$$

$$b(o_i) = \min_{C_B \neq C_{o_i}} \frac{1}{|C_B|} \sum_{o_j \in C_B} d(o_i, o_j)$$

$$sil(o_i) = \frac{b(o_i) - a(o_i)}{\max a(o_i), b(o_i)}$$

Then, the silhouette width of the a cluster is naturally defined as the average silhouette width of the objects within it, and the width of the entire clustering to be the average over its clusters.

$$sil(C_i) = \frac{1}{|C_i|} \sum_{o_j \in C_i} sil(o_j)$$

$$sil(C) = \frac{1}{n} \sum_{i=1}^n sil(C_i)$$

We can plot this coefficient over the choice of number of eigenfaces for each process (PCA and Robust PCA, varying  $\lambda$ ) in order to determine what method leads to the most natural clustering of faces into the correct groups.

## 5 Video Segmentation

One area where RPCA is commonly used is in video surveillance. In surveillance videos, each frame can be considered as having two major components- the static background and the moving, sparse foreground objects. Recall that RPCA decomposes the original matrix  $M$  into two parts, the low rank part  $L$  and the sparse part,  $S$ . Therefore, for video surveillance, background can be modeled as the low rank component  $L$  and the foreground objects can be modeled as the sparse component,  $S$ .
















Turns out, naïve PCA can also be used for separating video background and foreground objects since background remains largely the same across frames and PCA can provide a robust probability distribution of the background. However, PCA is known to be susceptible to noise and we think it is interesting to explore how this naïve PCA modeling of background and foreground compare against RPCA and in the process, gain more insights into how RPCA works and how it complements PCA.

## 6 Video Segmentation Experiment

We used two different surveillance videos for experimenting background and foreground segmentation. The first video consists of a scene where a few birds move around in front of a largely white background. We cut a sequence of 36 frames from this video, each 1 second apart. The second video displays a steady stream of cars moving down a highway.

For traffic video, we cut a sequence of 31 frames from this video, each 1 second apart also.

Background Truth for each video was constructed by manually removing all the sparse noise from a carefully selected frame. Due to slight variations of lighting and video quality across frames, this method will not give us the exact Ground Truth. However, for the purpose of our analysis, this manually constructed Ground Truth is sufficient.

Image Type	RPCA (2PC, $\lambda=0.015$ )	PCA (1 PC)	PCA (2 PC)
Original Image			
Ground Truth (GT)			
Extracted Background (EB)			
(GT - EB) Heatmap			
Foreground Noise			

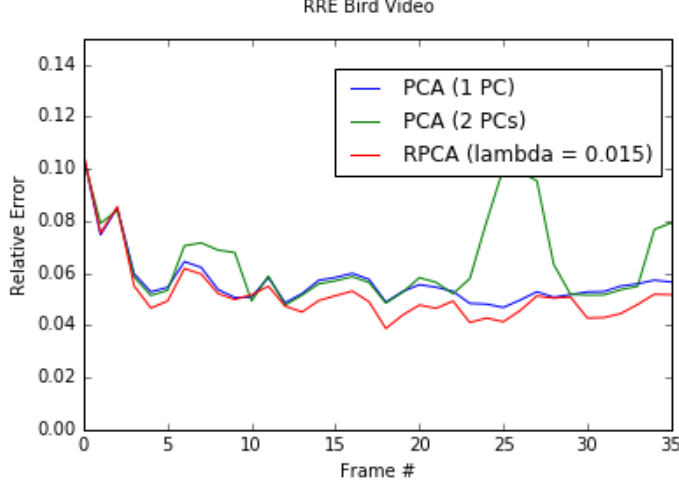
**Figure X. Background and foreground segmentation on Bird Video using RPCA and PCA**

As we can see from Figure X and Figure Y, the extracted background from RPCA for both bird and traffic videos more closely resembles background truth than that of PCA. There is a very minor trace of sparse noise in the extracted background and the differences are uniform throughout the image. Backgrounds extracted from PCA, however, suffers higher “pollution” caused by sparse noise present across the frames. The area with most visible difference is circled in red.

For foreground noise comparison, the Bird Video in Figure X provides a good visual



on the difference between RPCA and PCA. For RPCA, it can identify the exact pixel locations where a bird appears whereas for PCA, it is a mix of actual sparse noise and noise due to slight color variations in the frames. From here, we observe how RPCA prevents actual sparse noise from affecting our principal components without impacting the inherently Gaussian nature of the original dataset.



**Figure Z. Relative Reconstruction Error for Bird Video**

Figure Z shows the RRE for each background frame reconstructed from RPCA and PCA. Albeit a rough measure, we can see that RPCA’s extracted backgrounds are closer to Background Truth than PCA across almost all frames. Another interesting thing to note is that RRE for PCA using 2 components spikes for certain frames. This is because the 2nd principal component is “used up” in explaining the variations cause by sparse noise, causing a big deviation of reconstructed background from the Background Truth. The effect of sparse noise is a bit exaggerated for this data set but it highlights the fact that for real life data using PCA for dimensionality reduction, certain noise/outlier can cause a major problem due to the inability of PCA to handle sparse noise.

Recall the equation for RPCA:

$$\arg \min_{M'} (||A||_* + \lambda ||E||_1), \text{ subject to } D = A + E$$

We think that it is interesting to explore the impact of Lambda on the low rank and spare component.

Figure Y one shows background segmentation of the Bird Video using  $\lambda = 0.2$ , instead of 0.015 as in Figure Y. We can see that with a  $\lambda$ , RPCA’s performance in extracting

background is roughly equivalent to naïve PCA. The impact of sparse noise on the extract background image for RPCA has also become apparent. The area of extracted foreground noise, on the other hand, is now much smaller and concentrated. With a large  $\lambda$ , RPCA is less aggressive in identifying noise and only large enough deviations from Ground Truth are considered noise.

Based on the above experiment, we observed that RPCA is useful for removing irregularities in our dataset and therefore can help PCA obtain a good model governing the data. However, what makes RPCA effective is the regularization parameter  $\lambda$  that controls how much noise to be removed. There are also different types of RPCA that are designed for removing specific types of noise[Link Paper]. Like any other machine learning problems, the first step towards effective dimensionality reduction is to understand the nature of our dataset and apply RPCA accordingly.

## References

- [1] Basri, R., & Jacobs, D. W. (2003). Lambertian reflectance and linear subspaces. *IEEE transactions on pattern analysis and machine intelligence*, 25(2), 218-233.
- [2] Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. *Journal of the ACM (JACM)*, 58(3), 11.
- [3] Candès, E., & Recht, B. (2012). Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6), 111-119.
- [4] Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3), 211-218.
- [5] Kastner, K. (2014) "Robust Matrix Decomposition." Garbage In Garbage Out. <http://kastnerkyle.github.io/posts/robust-matrix-decomposition/>
- [6] Lin, Z., Chen, M., & Ma, Y. (2010). The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*.
- [7] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20, 53-65.

- [8] Wright, J., Ganesh, A., Rao, S., Peng, Y., & Ma, Y. (2009). Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems* (pp. 2080-2088).