Probabilistic PCA and Robust PCA - A Review

Hardil Mehta (1401018), Chintan Saraviya (1401026),
Kashish Shah (1401048), Rupande Shastri (1401102), Kishan Raval (1401117)

***Abstract*—The PCA method is widely used for dimensionality reduction in various applications, but it does not have a probabilistic model. While Probabilistic PCA uses eigenvalue decomposition of covariance matrix to extract the principal axis with maximum likelihood, Robust PCA separates the given matrix into its low-rank and sparse components by using an Augmented Lagrange Multiplier (ALM). The results of the Probabilistic PCA method are most efficient when the Expecation Maximization (E-M) algorithm is used to estimate maximum likelihood.**

I. Introduction and Theory

Principal Component Analysis (PCA) is a statistical method that reduces dimensionality while retaining most of the variance in the data. However, it has its own set of limitations. There is no probabilistic model to parametrize the data. This means less information to work with at the time of predicting data trends. The data is assumed to be along the directions of highest variance in the covariance based algorithm. The low variance directions are said to be noise. This assumption can cause loss of important data during PCA. Also, the data matrix is unrealistically assumed to be of low rank corrupted by normally distributed noise. A single outlier of noise may lead to significant error in the calculation of the low rank matrix.

A. Probabilistic Principal Component Analysis

For a given d-dimensional vector of observed data **t**, the relationship between **t** and a corresponding q-dimensional vector of latent variables **x** is given as **t = Wx +** μ **+** ϵ.

The parameter vector μ allows data with zero mean. The error ϵ is modeled as Gaussian, $\epsilon \; \mathcal{N}(0, \Psi)$ where error covariance Ψ is estimated from observed data. The *d x q* matrix **W** relates **t** to **x**. The computation of **W** gives us the coefficients of the dimensionality-reduced matrix.

We convert the latent variable model into a probabilistic model and obtain a mean μ and covariance matrix **S** of the data via Maximum Likelihood estimation or log-likelihood method in PPCA. The SVD of this covariance matrix **S** is computed to derive **C**. The other part is assumed as Gaussian distributed noise. **W** is given as, $W = C + \sigma^2 I$. This gives a closed-form solution, similar to PCA.

Expectation Maximization uses the probabilistic model in the form $t = \mathcal{N}(\mu, W = CC^T + R)$ to compute the parameter **W** by the following steps:

- **E-step:** Compute the posterior probability **x**|**t**
- **M-step:** Using the posterior, choose **C** and **R** to maximize joint likelihood of **t,x**

TABLE I. Comparing RPCA and PPCA

Percentage of noise or missing values	RPCA		PPCA	
	Missing Values	Corrupt Values	Missing Values	Corrupt Values
5	0.0581	0.0565	0.0535	0.0919
10	0.0693	0.0591	0.0713	0.1218
15	0.1257	0.0624	0.1045	0.1476
20	0.2765	0.0662	0.1528	0.1719
25	0.3773	0.0721	0.2129	0.1933
30	0.4708	0.0798	0.2783	0.2124
35	0.4990	0.0971	0.3408	0.2296

TABLE II. Time taken per iteration (in seconds)

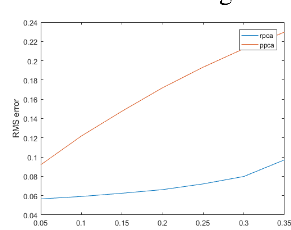
	Missing Values	Corrupted Values
RPCA	42.37	42.32
PPCA	2.44	2.34

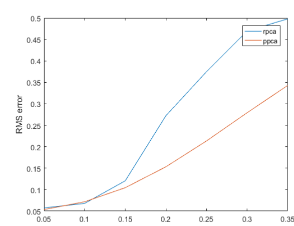
B. Robust Principal Component Analysis

RPCA solves the Principal Component Pursuit Problem of minimizing $||L|| + \lambda||S||^1$ wrt constraint $L + S = M$ using a modified version of the Augmented Lagrangian Multiplier, the ADMM algorithm, which iteratively sets new values for L and S. The algorithm ideally repeats until convergence, but that is not feasible. The step that takes most of the computation time in the algorithm is the calculation for updating L using singular value thresholding. Singular values exceeding the threshold μ are kept along with their corresponding singular vectors. Hence, the accuracy and time taken are determined by the value of threshold μ and the stopping criterion.

II. Implementation and Results

In an experiment with the same image, we manually induced corrupted and missing values of various degrees and applied RPCA and PPCA to them to compare them on the basis of accuracy and time per iteration, as shown in Table I. The results are shown in the figure below and in Table II.





III. Inference and Conclusion

It can be seen that while PPCA fares better than RPCA in recovering from missing values, the latter is more accurate in recovering from corrupted data. In both cases, the PPCA-EM algorithm is much faster than RPCA. We do not get a closed form solution in PPCA-EM as we do in RPCA but for practical purposes it serves well.