# Generalized Low Rank Model for Feature Selection

Zhihao Huang

January 21, 2019

**Abstract**

Principal Component Analysis (PCA) has been proven to be a very powerful technique in dimensional reduction, which allows user to conduct further analysis in a lower dimension while keeping the characteristics of the original data as much as possible. An enhanced version of PCA, the Generalized Low Rank Model (GLRM), has been proposed to accomplish additional tasks while conducting dimensional reduction. In this report, we follow the framework of the GLRM and implement three models with concentration in unsupervised feature selection. The Alternating Direction Method of Multipliers is used to address the optimization problem and C++ is used to create a command line tool. Experimental results demonstrate faster performances and the effectiveness in feature selection.

## 1 Introduction

In application of machine learning and data mining, one frequently encounters high dimensional data sets. The curse of dimensionality creates difficulties in implementation, analysis and computational viability. Therefore, the dimensional reduction techniques have been proposed to address this problem. Principle Component Analysis (PCA), an unsupervised learning algorithm, was proposed to reduce the dimension by projecting the original data onto a lower dimension. However, in the lower dimension, the features are the linear combinations of the original features, thus leading to problems with interpretation.

Therefore, another approach, feature selection [1], has been a very popular topic in recent years. It reduces the dimension by choosing a subset of the most relevant features. The features of the original data are ranked and selected as opposed to being transformed. Among all of the feature selection algorithms, sparse learning [2] is the most common and basic one for the feature selection without labels, i.e. unsupervised feature selection. Some of the sparse learning algorithms follow a similar framework, the generalized low rank model. It encourages certain structures in the low rank representations and approximates representation errors appropriately for a given data set.

In this project, we follow the framework of the generalized low rank model and extend the traditional PCA with different loss functions and penalty terms. We heavily focus on the functionality of unsupervised feature selection. The experiments show that the feature selection from our algorithm is effective and the optimization algorithm is efficient.

## 2 Algorithm

Throughout this report, matrices are written as capital letters and vectors are denoted as lowercase letters. For an arbitrary matrix $M \in \mathbb{R}^{m\times n}$, $M_{ij}$ denotes the $(i,j)$-th element of $M$. $m_i$ and $m_j$ denotes the $i$-th row and $j$-th column respectively. For the models in this report, $\|M\|_F =$

$\sqrt{\sum_{i=i}^{m}\sum_{j=1}^{n} M_{ij}^2}$ is the Frobenius norm of $M$, and $\|M\|_{2,1} = \sum_{i=1}^{m}\sqrt{\sum_{j=1}^{n} M_{ij}^2}$ is the $l_{2,1}$-norm. $\mathrm{Tr}(M)$ is the trace of $M$. $< A, B >= \mathrm{Tr}(A^T B)$ is the inner product of $A$ and $B$. Let $A \in \mathbb{R}^{n\times p}$ be the original data set with each row $a_i \in \mathbb{R}^{1\times p}$ being a data instance. $p$ is the number of features.

### 2.1 Generalized Low Rank Model

PCA finds a low rank matrix that minimizes the approximation error, in the least-squares sense, to the original data set. It takes the following optimization problem

$$\mathrm{argmin}_{X,Y} \|A - XY^T\|_F^2 \tag{1}$$

where $X \in \mathbb{R}^{n\times r}$ is the low dimensional representation of the original data and $Y \in \mathbb{R}^{p\times r}$ is the feature projection matrix (loading).

The Generalized Low Rank Model (GLRM) [3] is an extension of PCA. The least-squares in PCA is replaced by a loss function that is appropriate for the provided data type. Another extension beyond PCA is to add regularization on the low dimensional factors to impose or encourage some structures in the low dimensional factors. It solves the following optimization problem

$$\mathrm{argmin}_{X,Y} \mathrm{loss}(A, X, Y) + \lambda_1 \mathrm{reg}_1(X) + \lambda_2 \mathrm{reg}_2(Y) \tag{2}$$

### 2.2 Proposed Models

In this project, three models are proposed and implemented based on the GLRM framework. They have different object functions and optimization algorithms.

**Quadratic regularized PCA**

$$\mathrm{argmin}_{X,Y} \|A - XY^T\|_F^2 + \lambda_1\|X\|_F^2 + \lambda_2\|Y\|_F^2 \tag{3}$$

**Quadratic loss feature selection**

$$\mathrm{argmin}_{X,Y} \|A - XY^T\|_F^2 + \lambda\|Y\|_{2,1} \tag{4}$$

$$s.t. \quad X^T X = I$$

**Robust feature selection**

$$\mathrm{argmin}_{X,Y} \|A - XY^T\|_{2,1} + \lambda\|Y\|_{2,1} \tag{5}$$

$$s.t. \quad X^T X = I$$

Specifically, the $\|\cdot\|_F$ is good for preserving characteristics in the original data but sensitive to outliers. The $\|\cdot\|_{2,1}$ is less sensitive to ourliers. It also forces feature $Y$ to be sparse, hence selecting the features. The solutions for $X$ and $Y$ are not unique. Constraint on $X$ allows us to perform feature selection via $Y$.

### 2.3 Optimization Algorithm

For Quadratic regularized PCA model, there exists close form solution via singular value decomposition. We implement it as described and will not discuss its optimization algorithm here. The Quadratic loss feature selection model and Robust feature selection model have no close form solution and they are not convex in both $X$ and $Y$. Therefore, we use Alternating Direction Method of

Multiplier (ADMM) [4] to address the problem by introducing additional constraints and updating $X$ and $Y$ alternatively. Specifically, we rewrite the optimization problems as follows

**Quadratic loss feature selection**

$$\text{argmin}_{E,X,Y} \|A - E\|_F^2 + \lambda\|Y\|_{2,1} \tag{6}$$
$$s.t. \quad X^T X = I, E = XY^T$$

**Robust feature selection**

$$\text{argmin}_{E,X,Y} \|E\|_{2,1} + \lambda\|Y\|_{2,1} \tag{7}$$
$$s.t. \quad X^T X = I, E = A - XY^T$$

### 2.3.1 Augmented Lagrangian

Solving the constrained optimization problem (6) and (7) requires us to transform them back to unconstrained optimization problems [5]. As both problems can be simplified almost identically, we focus on problem (7) in this report and problem (6) can be solved in a similar manner. The augmented Lagrangian of problem (7) is

$$\text{argmin}_{E,X,Y} \|E\|_{2,1} + \lambda\|Y\|_{2,1} + < L, A - E - XY^T > + \frac{\mu}{2}\|A - E - XY^T\|_F^2 \tag{8}$$
$$s.t. \quad X^T X = I$$

where $L$ is the Lagrangian Multiplier and $\mu$ is the penalty parameter for the constraint $E = XY^T$. The alternating updates of the arguments solve the optimization problem.

**Update** $X$ To update $X$, we keep all of the variables other than X as a constant.

$$\text{argmin}_X < L, A - E - XY^T > + \frac{\mu}{2}\|A - E - XY^T\|_F^2 \tag{9}$$
$$s.t. \quad X^T X = I$$

We transform the trace to $\|\cdot\|_F$ and (9) is a Orthogonal Procrustes problem [6]. Therefore,

$$X = UV^T \tag{10}$$

where $[U, \Sigma, V] = \text{svd}((A - E + \frac{1}{\mu}L)Y^T)$.

**Update** $Y$ To update $Y$, we keep variables other than $Y$ fixed.

$$\text{argmin}_Y \lambda\|Y\|_{2,1} + < L, A - E - XY^T > + \frac{\mu}{2}\|A - E - XY^T\|_F^2 \tag{11}$$

This equation has a close form solution [7]. Let $Q = (A - E + \frac{1}{\mu}L)^T X$, then

$$y_i = \begin{cases} 1 - \frac{\lambda}{\mu\|q_i\|} q_i & \text{if } \|q_i\| > \frac{\lambda}{\mu}; \\ 0 & \text{else.} \end{cases} \tag{12}$$

**Update** $E$ To update $E$, we fix variables other than $E$.

$$\text{argmin}_E \|E\|_{2,1} + < L, A - E - XY^T > + \frac{\mu}{2}\|A - E - XY^T\|_F^2 \tag{13}$$

This is nearly the same as updating $Y$. Let $P = A - XY^T + \frac{1}{\mu}L$, then

$$e_i = \begin{cases} 1 - \frac{1}{\mu\|p_i\|}p_i & \text{if } \|p_i\| > \frac{1}{\mu}; \\ 0 & \text{else.} \end{cases} \tag{14}$$

**Update multipliers** In addition to the arguments, the Lagrangian multipliers and constraint penalty parameters should be updated as well.

$$\mu = \max(\mu\rho, \mu_{\max}) \qquad L = L + \mu * (A - E - XY^T) \tag{15}$$

where $\rho$ is a constant, say 10, that allows $\mu$ to grow exponentially until the maximum allowance.

### 2.3.2 Algorithm

With all the steps, we have the following algorithm for solving the Robust Feature Selection model.

**Algorithm 1** Robust Feature Selection

```
Input Data: A, λ, r
Random Initialization X and Y
while not converge do
    while not converge do
        Update X through (10)
        Update Y through (12)
        Update E through (14)
    end while
    Update μ and L through (15)
end while
Output Data: X, Y
```

The convergence of the algorithm is based on the convergence of the ADMM algorithm. The convergence criterion is set so that the change in object function between two iteration is smaller than a certain threshold, usually $10^{-7}$.

### 2.3.3 Parameter Initialization

There are several ways to Initialize $X$ and $Y$. In our project, the $X$ and $Y$ are initialized with respect to the median of all elements of $A$. Empirically, experiments show that such approach is potentially near the optimal value. Specifically, let $m$ be the median of all $A$ elements, then any arbitrary element of $X$ can be initialized by $X_{ij} = \text{Unif}(0, 1) * \sqrt{(\frac{m}{r})}$. Same method goes with the initialization of $Y$. The constraint parameter $\mu$ usually comes with any arbitrary value, as it is multiplied by $mu$ and constantly increases over the algorithm. $\rho$ is usually set to 10 and the $\mu_{\max}$ is set to $10^{10}$.

### 2.3.4 Time Complexity Analysis

The most time consuming part is the calculation of X, especially the singular value decomposition. It involves solving the exact svd of a $n \times d$ matrix and the time complexity is $O(nd^2)$. The rest of the updates are just simply matrix multiplication. The worst case scenario is $O(ndr)$. As $d \gg n$ and $d \gg r$, the overall time complexity for each iteration is $O(nd^2)$.

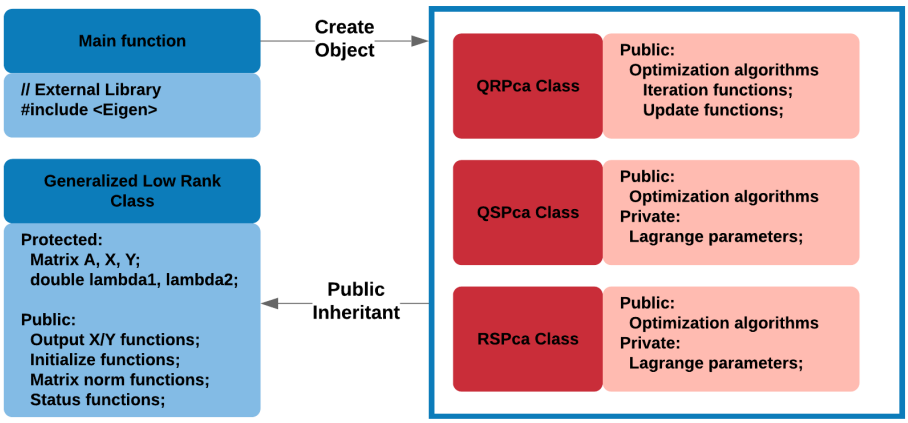



Figure 1: C++ code structure

## 3 Implementation

The project is implemented in C++ as shown in Figure 1. A generalized low rank class is created as basis class. It encapsulates all the necessary elements in the GLRM, including $A$, $X$, $Y$, as well as commonly used matrix norms. All three models are child classes of the generalized low rank class. They inherent the data from the parent class and encapsulate their own optimization algorithm.

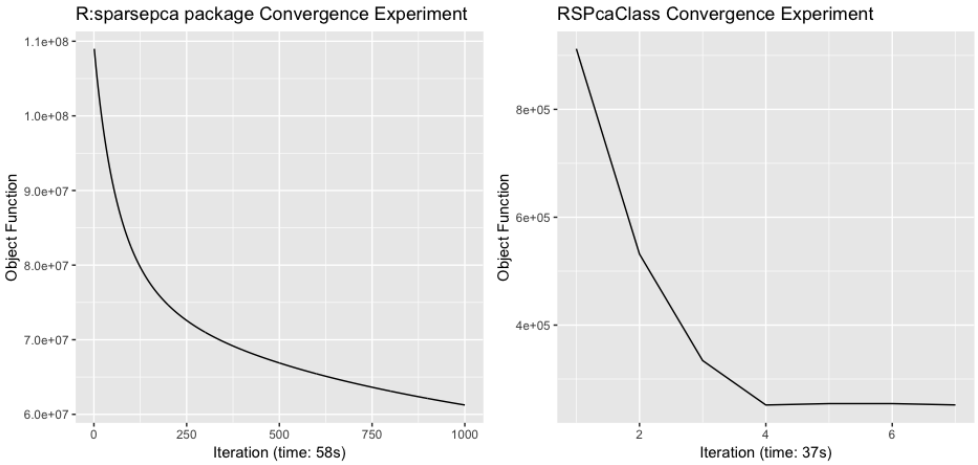
## 4 Experiment

For evaluating the feature selection result, we use a human faces data set to conduct experiments. Several parameters are to be set. The low rank $r$ is set to 50, and the feature selection parameter $\lambda$ is set to $10^{-5}$. Usually the $\lambda$ and $r$ should be tuned with cross validation with a particular goal. For simplicity, all models for comparison use the same parameters in the experiment. Four models are compared, the Sparse PCA in R, QRPcaClass, QSPcaClass and RSPcaClass (with respect to the three models implemented). Top 10 - 150 features are selected and common metrics are used for evaluation, i.e. prediction accuracy (with KNN), normalized mutual information (NMI) and clustering accuracy (with k-means). Each of the metrics refers to the average of all choices of features. We also compare the convergence speed of the algorithm with the R package sparsepca.

| Metric | R: Sparse PCA | QRPcaClass | QSPcaClass | RSPcaClass |
|---|---|---|---|---|
| **Acc (KNN)** | 0.6606 | 0.6457 | 0.7138 | **0.7211** |
| **NMI** | 0.3008 | 0.3317 | **0.4223** | 0.4216 |
| **Acc (Clustering)** | 0.3178 | 0.3054 | 0.4068 | **0.4070** |

Table 1: Feature Selection Evaluation

It is clear that the proposed models have better performances as shown in Table 1. All three metrics for the feature selection models are higher than the ones without such functionality. This proves the effectiveness of feature selection. Also, the optimization algorithm and implementation in C++ are proved to be efficient in Figure 2. The object function decreases over time. The computation time is shorter compared with the R package and the convergence requires less iterations.




(a) R: Sparse PCA (b) RSPcaClass

Figure 2: Convergence Experiment

## 5 Conclusion and Future Development

Following the framework of the GLRM, we proposed, solved and implemented three models that extend the functionality of traditional PCA with concentration in unsupervised feature selection. The experiments show that our algorithm is effective and efficient. As many computations involves large scale matrices, further improvements can be done with parallel computing in GPU (CUDA).

## 6 Acknowledgement

This is an extension of the group project of the course BIOSTAT 615 of the University of Michigan. My teammate, Ji Young Yun and Pin-Yu Chen, helped me to craft some of the slides and this report. Thanks for their support!

## References

[1] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[2] Suhang Wang, Jiliang Tang, and Huan Liu. Embedded unsupervised feature selection. In *AAAI*, pages 470–476, 2015.

[3] Madeleine Udell, Corinne Horn, Reza Zadeh, and Stephen Boyd. Generalized low rank models. *Foundations and Trends in Machine Learning*, 9(1):1–118, 2016.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] Wikipedia contributors. Orthogonal procrustes problem — Wikipedia, the free encyclopedia, 2018. [Online; accessed 11-December-2018].

[7] Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient l 2, 1-norm minimization. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 339–348. AUAI Press, 2009.