

# Generalized Low Rank Model for Feature Selection

Zhihao Huang

University of Michigan

December, 2018

# Outline

- 1 Introduction
- 2 Algorithm
- 3 Implementation
- 4 Experiment
- 5 Conclusion

# Outline

- 1 Introduction
- 2 Algorithm
- 3 Implementation
- 4 Experiment
- 5 Conclusion

# Motivation

## Principal Component Analysis (PCA)

- Identifies a low rank matrix that minimizes the error based on least-squares

## Principal Component Analysis

Suppose  $A \in \mathbb{R}^{n \times p}$ ,  $X \in \mathbb{R}^{n \times r}$  and  $Y \in \mathbb{R}^{p \times r}$ , the principal component analysis optimize the following object function

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

## Generalized Low Rank Model (GLRM)

- Extension of PCA
- Add regularization on low-dimensional factors
- Change loss function on approximation error

# Motivation

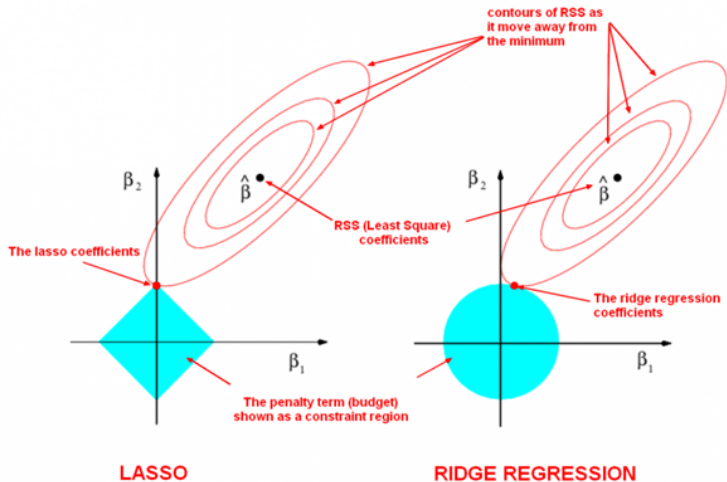
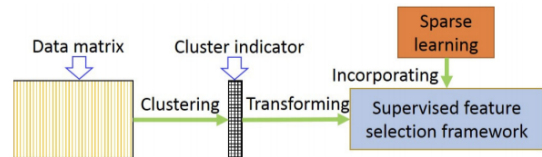
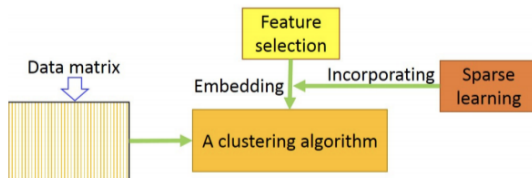


Figure 1: Sparse Learning

# Motivation: Unsupervised Feature Selection



(a) Existing sparse learning based unsupervised feature selection method



(b) Embedded unsupervised feature selection

Figure 2: Difference between existing sparse learning based and embedded unsupervised feature selection

# Goals and Challenges

## Goals

- Dimensional reduction with concentration on feature selection
- Build a C++ program based on a generalized low rank matrix decomposition algorithm along with user-defined generalizations

## Challenges

- Optimization algorithm for sparse learning
- Fast performance
- Generalized code framework

# Outline

- 1 Introduction
- 2 Algorithm**
- 3 Implementation
- 4 Experiment
- 5 Conclusion



Basis framework:

## Generalized Low Rank Model

Suppose  $A \in \mathbb{R}^{n \times p}$ ,  $X \in \mathbb{R}^{n \times r}$  and  $Y \in \mathbb{R}^{p \times r}$ , the glrm optimize the following object function

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

## Quadratic regularized PCA (QRPcaClass)

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

## Quadratic loss feature selection (QSPcaClass)

$$\begin{aligned} \operatorname{argmin}_{X,Y} \|A - XY^T\|_F^2 + \lambda \|Y\|_{2,1} \\ \text{s.t. } X^T X = I \end{aligned}$$

## Robust feature selection (RSPcaClass)

$$\begin{aligned} \operatorname{argmin}_{X,Y} \|A - XY^T\|_{2,1} + \lambda \|Y\|_{2,1} \\ \text{s.t. } X^T X = I \end{aligned}$$

Notations:  $\|A\|_F^2 = \sum_{i=1}^n \sum_{j=1}^p A_{ij}^2$ ,  $\|Y\|_{2,1} = \sum_{i=1}^p \sqrt{\sum_{j=1}^r Y_{ij}^2}$

## Alternating Direction Method of Multiplier (ADMM)

Applied to solve the following optimization problem

$$\min_x f(x) + g(x)$$

$$\Leftrightarrow \min_{x,y} f(x) + g(y), \quad s.t. \quad c := (x - y) = 0$$

- Attack the problem with constrained optimization
- Separate the problem into smaller pieces

Augmented Lagrangian

$$\Leftrightarrow \min_{x,y} f(x) + g(y) + \frac{\mu}{2}c^2 + \lambda_l \cdot c$$

- Penalty coefficient updates:  $\mu^{(k+1)} := \rho \cdot \mu^{(k)} \quad (\rho > 1)$
- Lagrangian multiplier updates:  $\lambda_l^{(k+1)} := \lambda_l^{(k)} + \mu^{(k)} * c$

## Example: algorithm for robust feature selection

### Quadratic loss feature selection

$$\begin{aligned} \operatorname{argmin}_{X,Y} \quad & \|A - XY^T\|_{2,1} + \lambda \|Y\|_{2,1} \\ \text{s.t.} \quad & X^T X = I \end{aligned}$$

Equivalent optimization problem:

$$\begin{aligned} \operatorname{argmin}_{E,X,Y} \quad & \|E\|_{2,1} + \lambda \|Y\|_{2,1} \\ \text{s.t.} \quad & X^T X = I, E = A - XY^T \end{aligned}$$

Augmented Lagrangian (object function to optimize):

$$\begin{aligned} \operatorname{argmin}_{E,X,Y} \quad & \|E\|_{2,1} + \lambda \|Y\|_{2,1} + \langle L, A - E - XY^T \rangle + \frac{\mu}{2} \|A - E - XY^T\|_F^2 \\ \text{s.t.} \quad & X^T X = I \end{aligned}$$

where  $L$  is the lagrangian multiplier,  $\mu$  is the penalty coefficient and  $\langle \cdot, \cdot \rangle$  is the matrix inner product. Specifically,  $\langle A, B \rangle = \operatorname{tr}(A^T B)$ .

## Example: algorithm for robust feature selection

**Data:**  $A, \lambda, r$

Random initialization  $X$  and  $Y$ ;

**while** *not converge* **do**

**while** *not converge* **do**

$X = \operatorname{argmin}_X \langle L, A - E - XY^T \rangle + \frac{\mu}{2} \|A - E - XY^T\|_F^2;$

$Y = \operatorname{argmin}_Y \lambda \|Y\|_{2,1} + \langle L, A - E - XY^T \rangle$   
         $+ \frac{\mu}{2} \|A - E - XY^T\|_F^2;$

$E = \operatorname{argmin}_E \|E\|_{2,1} + \langle L, A - E - XY^T \rangle + \frac{\mu}{2} \|A - E - XY^T\|_F^2;$

**end**

$\mu = \rho\mu, L = L + \mu \|A - E - XY^T\|_F^2;$

**end**

**Result:**  $X, Y$

Convergence condition:

- Object function decreases slowly
- Reaches largest iteration times

# Outline

- 1 Introduction
- 2 Algorithm
- 3 Implementation**
- 4 Experiment
- 5 Conclusion

# Implementation

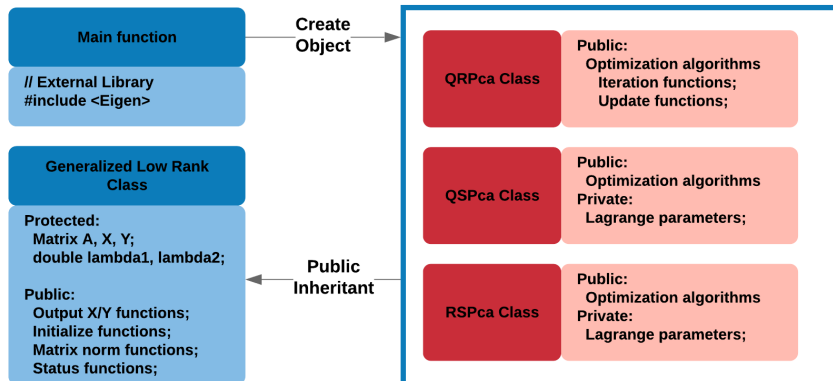


Figure 3: Cpp code structure

# Implementation

```
0587372218:project zhhhwang$ make
g++ -c main.cpp -I ~/Downloads/Eigen -I~/Downloads/boost -o main.o
g++ -c GLrmClass.cpp -I ~/Downloads/Eigen -I~/Downloads/boost -o GLrmClass.o
g++ -c QRPCaClass.cpp -I ~/Downloads/Eigen -I~/Downloads/boost -o QRPCaClass.o
g++ -c QSPcaClass.cpp -I ~/Downloads/Eigen -I~/Downloads/boost -o QSPcaClass.o
g++ -c RSPcaClass.cpp -I ~/Downloads/Eigen -I~/Downloads/boost -o RSPcaClass.o
g++ main.o GLrmClass.o QRPCaClass.o QSPcaClass.o RSPcaClass.o -o proj.exe
0587372218:project zhhhwang$ ./proj.exe

#####
# Generalized Low Rank Model for Feature Selection #
#####

argmin_X, Y} loss(A, XY^T) + lambda1 * reg(X) + lambda2 * reg(Y)

This is the GLRM for Feature Selection program.
You may choose your own specification to run a generalized low rank model with concentration on feature selection.
Please follow the instructions as printed.

Please specify the input data directory: test.csv

Please specify your model. Choose the listed options.
1. Quadratic loss (F norm), quadratic regularizer (F norm).
2. Quadratic loss (F norm), sparse regularizer (l-2,1 norm).
3. Robust loss (l-2,1 norm), sparse regularizer (l-2,1 norm).
Your option is: 3

Please follow the instructions to complete the model specification.
Please specify the low rank: 50
Please specify the lambda for X: 0
Please specify the lambda for Y: 0.001
Please specify the largest iteration times: 50
Please specify the constraints error threshold level: 0.00001
Do you want to see the update process? Type 0 for NO and other YES: 1
===== 100% Current iteration is #1, and the value of object function is 912281.
===== 100% Current iteration is #2, and the value of object function is 255371.
===== 100% Current iteration is #3, and the value of object function is 257948.

Iteration complete!

The trained X and Y are output as csv file in your code directory.
Thank you for using this program! :)
```

Makefile one-step compiling

Execute Program

User Specifies input file

User chooses models

User inputs mode specification

Optimization starts

Write out file automatically

Figure 4: Program Demo



# Outline

- 1 Introduction
- 2 Algorithm
- 3 Implementation
- 4 Experiment**
- 5 Conclusion

# Experiment



Figure 5: Original Data

## warpAR10P Human Face Data Set

- Unstructured data (0 - 255)
- Class size: 10
- Sample size: 130
- Feature size: 2400 (60\*40 pixels)
- More details in the report ...

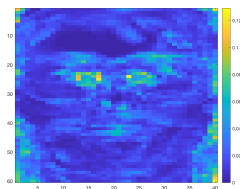
# Experiment

- Used the same tuning and optimization parameters
- Fitted the models
- Ranked the feature importance
- Chose top-ranked features for evaluations
- Averaged performance on different choices of #features

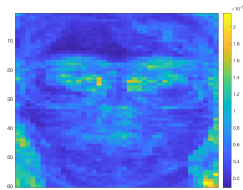
Metric	R: Sparse PCA	QRPcaClass	QSPcaClass	RSPcaClass
Acc (KNN)	0.6606	0.6457	0.7138	0.7211
NMI	0.3008	0.3317	0.4223	0.4216
Acc (Clustering)	0.3178	0.3054	0.4068	0.4070

Table 1: Feature Selection Evaluation

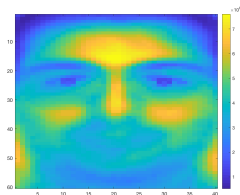
Figure 6: Feature Selection Result



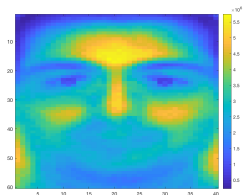
(a) R: Sparse PCA



(b) QRPcaClass

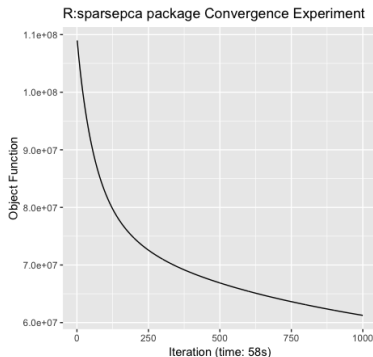


(c) QSPcaClass

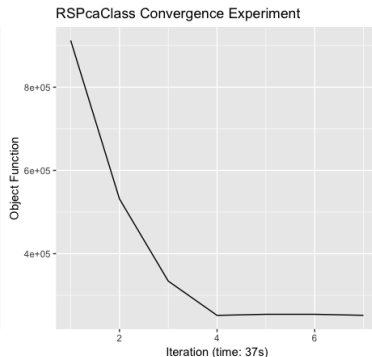


(d) RSPcaClass

Figure 7: Convergence Experiment



(a) R: Sparse PCA



(b) RSPcaClass

# Outline

- 1 Introduction
- 2 Algorithm
- 3 Implementation
- 4 Experiment
- 5 Conclusion**

# Conclusion

## Achievements

- Completed a C++ program. Users can ...
  - Specify the models
  - Add user-defined classes (other glrm models)
  - Obtain outputs of low rank representation and feature importance
- Conducted experiment shows effectiveness in...
  - The optimization algorithm
  - The feature selection functionality
- Typed up a comprehensive program instruction

## Potential Improvements in ...

- Algorithm speed: parallel computing (CUDA)
- Include more generalized models