Graph-Regularized Generalized Low-Rank Models

Mihir Paradkar
Dept. Biological Engineering
Cornell University
mkp65@cornell.edu

Madeleine Udell
Dept. Operations Research
Cornell University
mru8@cornell.edu

Abstract

Image data is frequently extremely large and oftentimes pixel values are occluded or observed with noise. Additionally, images can be related to each other, as in images of a particular individual. This method augments the recently proposed Generalized Low Rank Model (GLRM) framework with graph regularization, which flexibly models relationships between images. For example, relationships might include images that change slowly over time (as in video or surveillance data), images of the same individual, or diagnostic images which picture the same medical condition. This paper proposes a fast optimization method to solve these graph-regularized GLRMs, which we have released as an open-source software library. We demonstrate that the method outperforms competing methods on a variety of data sets, and show how to use this method to classify and group images.

1. Introduction

Noisy images pose problems for many tasks in computer vision, obscuring features that may be present and the true class structure of the data. The high dimensionality of image data exacerbates the effects of noise and corruption by masking the true distribution of data further. Sufficient statistics can be derived from low-dimensional and graph structure to better estimate noisy values.

One common way to obtain a low-dimensional structure is PCA, which seeks to find the best approximation (in the least-squares sense) of a given rank to a data matrix. The factorization of this approximation embeds the data into a low-dimensional space. Several extensions exist which modify PCA in some way, either by changing the approximation metric as in robust PCA [3], or by imposing additional penalties on the factorization as in some matrix completion algorithms [6] or sparse PCA [4].

The Generalized Low-Rank Model (GLRM) [7] is an emerging framework that extends this idea of a low-rank factorization. It allows mixing and matching of loss functions and various regularization penalties, such as l_1 and l_2 penalties, to be fit over heterogeneous-typed and missing or unknown data.

Spectral Embedding (SE) [1] takes a different approach to reducing the dimension of data, using a graph where edge weights are pairwise similarities. This graph is used to learn a different low-dimensional embedding of data. This method can have the advantage of preserving non-linear structures in data that would be lost in a linear projection such as PCA.

Methods such as Graph-Laplacian PCA (glPCA) [5] seek to combine the low-dimensional data representation of PCA with the non-linear embedding of SE by minimizing a weighted sum of their objective functions. In this paper, the graph-regularized GLRM (GraphGLRM or GGLRM) is introduced as a new way to incorporate graph data in a low-rank data representation. The GraphGLRM framework extends glPCA by providing ways to choose loss functions, regularizers, and non-missing values. We demonstrate that the incorporation of graph information in GLRMs can outperform pure PCA, spectral embedding, and vanilla GLRMs in reconstructing occluded images and in creating a low-dimensional embedding suitable for supervised learning.

2. Previous Work

2.1. Generalized Low-Rank Models

The Generalized Low Rank Model (GLRM) was developed by Udell *et al*. to fit a low-rank representation $X^T W$, $X \in \mathbb{R}^{k \times n}$, $W \in \mathbb{R}^{k \times d}$ to the non-occluded portions of a data matrix $Y \in \mathbb{R}^{n \times d}$. The model aims to minimize the difference between Y_{ij} and imputed $x_i^T w_j$ as measured by l_j for $(i, j) \in \Omega$, where Ω are the indices of known values. The model also can impose regularization penalties r_i and $\tilde{r}_j$ on the rows of representation matrix X^T and columns of embedding matrix W respectively to ensure chosen properties of the low-rank representation. The

objective function can then be written as

$$\min_{X,W} \sum_{(i,j)\in\Omega} l_j(Y_{ij}, x_i^T w_j) + \sum_{i=1}^{n} r_i(x_i) + \sum_{j=1}^{d} \tilde{r}_j(w_j). \quad (1)$$

This framework recovers PCA by setting $l_j(Y_{ij}, x_i^T w_j) = \|Y_{ij} - x_i^T w_j\|_2^2$, $r_i(x_i) = r_j(w_j) = 0$, and letting $\Omega = \{(i,j)|i \in 1..m,\ j \in 1..n\}$. In this form, the solution is not unique, so the constraint that $WW^T = I$ enforces uniqueness. This produces the well-known objective function

$$\min_{X,W} \|Y - X^T W\|_F^2 \quad s.t.\ WW^T = I. \quad (2)$$

However, the use of other loss functions besides least-squares can be beneficial. For example, the use of the absolute-value loss function can produce a solution more robust to outliers. The GLRM framework also exploits column-wise separability of regularizers to utilize the properties of different features. In this paper, the GLRM with column-wise regularization is referred to as 'vanilla' to distinguish it from the newly-developed model presented.

2.2. Spectral Embedding

The objective of PCA is to find an optimal linear projection to minimize euclidean distance between the original reconstructed data. In contrast to PCA, spectral embedding seeks to preserve the similarity on edges specified in a graph. Let A and D be the adjacency and degree matrix, respectively, of the graph. The aim of spectral embedding is to find a matrix X^T with one row for every node in the graph, such that the sum of euclidean distances between connected records is minimized. Letting E be the edge set, compute the spectral embedding by minimizing the objective function

$$\min_{X} \sum_{(i,j)\in E} \|x_i - x_j\|_2^2. \quad (3)$$

This objective can be written in matrix form as

$$\min_{X} \text{tr}(XLX^T) \ \ s.t.\ XX^T = I. \quad (4)$$

The solution to this problem can be found by computing the eigenvectors corresponding to the smallest eigenvalues of matrix L. The rows of X^T are vector representations of every record in the graph.

2.3. Graph-Laplacian PCA

Graph-Laplacian PCA (glPCA) incorporates both linear projection and graph structure in a low-dimensional embedding [5]. This method demonstrates increased robustness over PCA because graph structure penalizes fitting to outliers. GLPCA incorporates graph structure by minimizing a weighted sum of the objective functions of PCA and Spectral Embedding. The objective function can be expressed as

$$\min_{X,W} \|Y - X^T W\|_F^2 + \alpha \text{tr}(XLX^T) \ \ s.t.\ XX^T = I. \quad (5)$$

By adjusting α, the effect of graph structure can be adjusted to appropriately fit the given data.

3. Graph Regularizer

3.1. Objective Function

This paper combines the flexible losses of GLRMs with graph structure to allow fitting flexible models using both types of structure. In contrast to glPCA, the graph regularizer can be applied to a model with a non-quadratic loss function and arbitrary regularization on the other factor. Let Y be a data matrix where the ith row y_i is an example and the jth column y_j is a feature. Also let L be the Laplacian matrix for a fixed graph on the examples which tells us the similarity between the examples. The form of the GGLRM is

$$\min_{X,W} \sum_{(i,j)\in\Omega} l_j(Y_{ij}, x_i^T w_j) + \alpha \text{tr}(XLX^T) + \tilde{r}(W). \quad (6)$$

In this objective function, $\tilde{r}(W)$ can be any convex penalty. In this paper, we shift L by λI where λ is a small positive constant. This shifting helps guarantee convergence and reduce over-fitting, which is explained in section 3.2. The penalty on the right factor $\tilde{r}(W)$ can also be chosen to be a graph regularizer,

$$\tilde{r}(W) = \text{trace}(WL'W^T). \quad (7)$$

Graph regularization can be used to help learn the values of missing entries in a matrix. Intuitively, graph regularization on X encourages the values of a missing entry in a row to be close to a corresponding known entry in another row; similarly, graph-regularization on W encourages the feature embedding of a missing column to be close to that of a more complete column. Specifically, graph regularization on X encourages the representations $x_i, x_{i'}$ to be similar for related rows i and i', encouraging the values $x_i^T w_j, x_{i'}^T w_j$ to be similar. Graph regularization on W has the same effect for related columns j and j'. Jiang *et al.* have demonstrated that using this regularization with PCA yields improved results in various supervised and unsupervised tasks on data that includes partially-occluded pictures [5].

3.2. Fitting

The fitting algorithm is a special case of the Proximal Alternating Linear Minimization PALM algorithm [2]. In each iteration, a gradient step is followed by a proximal

step for each of X and W. In the gradient step, a step of magnitude α_X is taken in the direction $\nabla_X(l_j)$. Next, the proximal mapping $\text{prox}_{\alpha_X r_x}(X - \alpha_X \nabla_X(l_j))$ is used to update X given regularizer r_x and stepsize α_X, and likewise for W. These alternating steps are in contrast to an alternating minimization approach, where X and W are alternatively minimized while holding the other constant. We use the PALM algorithm because the loss functions we use harder to minimize than the quadratic loss, so alternating gradient and proximal steps lead to faster convergence.

Algorithm 1 Fitting Algorithm

initialize X_1, Y_1, gradX_1, gradY_1
for t in 1 to max_iterations **do**
  gradX_t = $\nabla_X(l_j)$
  determine value of stepsize α_X via backtracking linesearch
  X_{t+1} = $\text{prox}_{\alpha_X r_x}(X_t - \alpha_X \text{grad}X_t)$
  gradW_t = $\nabla_W(l_j)$
  determine value of α_W via backtracking linesearch
  W_{t+1} = $\text{prox}_{\alpha_W r_w}(W_t - \alpha_W \text{grad}W_t)$
end for

This algorithm uses the proximal operator (prox), defined as

$$\text{prox}_r(Z) = \underset{U}{\text{argmin}}(r(U) + \frac{1}{2}\|U - Z\|_F^2).$$

We can compute the prox operator for the graph-Laplacian regularizer $r_x(U) = \text{tr}(ULU^T)$. We use the fact that $\nabla_U r_x(U) = 2LU$ to set the gradient of $r_x(U) + \frac{1}{2}\|U - Z\|_F^2$ to zero. Solving for U gives

$$\text{prox}_{r_x}(Z) = (2L + I)^{-1}Z. \tag{8}$$

However, using the Laplacian regularizer on X with another regularizer on W is not sufficient to ensure convergence. Convergence requires the proximal operator to be contractive. We ensure that the operator is contractive by adding a small positive diagonal matrix λI to L so that $\lambda I + L$ is positive definite. Adding a positive diagonal matrix λI to L is equivalent to quadratic regularization by λ, so a more stable $rx(X)$ can be expressed as

$$\begin{aligned} r(X) &= \text{trace}(XLX^T) + \lambda\|X\|_F^2 \\ &= \text{trace}(XLX^T) + \lambda(\text{trace}(XX^T)) \\ &= \text{trace}(X(L + \lambda I)X^T). \end{aligned} \tag{9}$$

One case in which this quadratic regularization is necessary is for an unconnected graph. For example, $\mathbf{0}_{n,n}$ is the Laplacian matrix of an unconnected graph with n nodes. With objective function

$$\underset{X,W}{min} \sum_{(i,j)\in\Omega} l_j(Y_{ij}, x_i^T w_j) + \text{trace}(X\mathbf{0}_{n,n}X^T) + \|W\|_F^2,$$

the objective value can be reduced by setting $X' = \beta X$ and $W' = \beta^{-1}W$ for any $\beta > 1$. Therefore, the optimal solution is achieved as $X \to \infty$ and $W \to 0$ along some path such that $X^T W$ is constant and equal to the best rank-k approximation to Y. Adding quadratic regularization by λ prevents this divergence by penalizing large values of X.

Additional l_2 regularization also reduces overfitting. The combination of graph-regularization and l_2 penalty not only serves to improve the quality of imputed values, but also increases the number of regularization options available within the GLRM framework.

Although computing the prox operator for the graph regularizer requires a matrix inversion, performance bottlenecks usually stem from gradient and objective computation in practice. This is because the sparse Cholesky decomposition that we use to invert the matrix is extremely fast and Laplacian matrices are usually sparse. However, in the case of a very large graph, the conjugate gradient method is a computationally cheap way to approximate the inverse of a matrix in finite iterations.

The case of an unconnected graph Laplacian with offset αI reduces to quadratic regularization. In this paper, we use this quadratic regularization on the other factor W. The update algorithm for quadratically (l_2) regularized W with regularization parameter α uses the shrinkage operator

$$W_{t+1} = \frac{V}{1 + 2\alpha_W}$$

where

$$V = W_t - \alpha_W \nabla_W (\sum_{(i,j)\in\Omega} l_j(Y_{ij}, x_i^T w_j)).$$

Since both regularizers are smooth, differentiable functions, the proximal update sequence converges [2].

4. Software Implementation

GraphGLRM is available as a software package in the Julia language. The implementation is freely available online at `github.com/mihirparadkar/GraphGLRM.jl`. Features and usage of the software are described here.

4.1. Usage

Modeling data with GraphGLRM is a two-step process. A user first specifies a model, and then fits the model. The GraphGLRM API is inspired by and builds on the LowRankModels framework [7], using available loss functions and most regularizers from LowRankModels. The model is built around the `GGLRM` type, which stores model parameters and fitted factors. To construct a `GGLRM`, the user specifies, in order:

- `A`, a dense matrix, a sparse matrix, a DataFrame, or any tabular data structure which supports row and column indexing
- `losses`, a loss function or list of loss functions where there is one loss function for each column
- `rx`, a regularizer for the embedded examples
- `ry`, a regularizer for the embedded features
- `k`, the rank of the low-rank factorization.

Optional arguments include:

- `obs`, a list of indices where the data value is known, i.e. not missing or occluded
- `sparse_na`, whether or not to treat the zeros in a sparse matrix as missing values.

4.2. Performance

GraphGLRM is tuned for fast fitting speeds on missing data. These fitting speeds are achieved by copying columns with missing values into memory-contiguous arrays. Gradients and objective values can be computed much faster on such arrays, more than offsetting the performance penalty of copying data.

GraphGLRM also enables multi-threaded fitting using Julia's experimental (at the time of writing) threading support. Multi-threaded fitting achieves modest speedup by computing the gradient matrix and objective function in parallel over each loss function. This can be easily achieved by calling `fit_multithread!` instead of `fit!` after initializing a `GGLRM`.

It is also beneficial not to form $X^T W$ explicitly for highly sparse data, especially for large sparse matrices. This is because a dense representation of $X^T W$ may be orders of magnitude larger than the original sparse data. GraphGLRM provides `fit_sparse!` for these cases, which computes gradient values only for observed entries.

5. Experiment Design

We demonstrate the effectiveness of GGLRM by comparing it to PCA, Spectral Embedding, and a vanilla GLRM (without graph regularization). The GLRM and graph-regularized GLRM were constructed with the Huber loss function,

$$\text{Huber}(z) = \begin{cases} \frac{1}{2}z^2, & \text{if } |z| \leq 1 \\ |z| - \frac{1}{2}, & \text{if } |z| > 1, \end{cases}$$

where $z = Y_{ij} - x_i^T w_j$.

We consider two types of regularization: quadratic regularization on the latent factors and graph regularization.

Algorithm	Loss	Quad. Reg.	Graph Reg.
PCA	Quadratic		
SE	None		✓
Vanilla GLRM	Huber	✓	
Graph GLRM	Huber	✓	✓
glPCA	Quadratic		✓

Table 1. Components of the objective functions for each of PCA, Spectral Embedding, Vanilla GLRM, GraphGLRM, and glPCA.

Figure 1. Examples of images of a given individual in the Olivetti dataset. We construct a graph that exploits these similarities to improve low-dimensional embedding.

Table 1 illustrates the types of loss function and regularizer present on each of 5 algorithms.

We use very small quadratic regularization ($\lambda = 0.01$) to stabilize the solution. The GGLRM uses the same regularizer as the vanilla GLRM, and adds a graph regularizer.

We use the Olivetti faces dataset from AT&T Cambridge Labs to compare results between algorithms. This dataset consists of 400 images, with 10 images each of 40 different people. The images of a given individual differ in facial expression, lighting, and slightly in angle.

The graph was constructed as a union of 40 cliques, each of which links every pair of images of the same individual in the dataset. This graph was used for the spectral embedding algorithm, and as the additional graph for regularization in the graph GLRM.

Algorithms were compared in their performance on two experiments involving random occlusions on the dataset. 6.25% of each image was replaced with a black rectangle at a random location. The locations of nonzero pixel values were used as the observation set for both GLRM varieties.

In the first experiment, we use each method to impute the occluded pixels. We measure and report the mean-squared error in our reconstruction.

In the second task, we classify faces as male or female. We use the low-rank representations of the images as features in a linear-kernel Support Vector Machine (SVM). We use precision and recall in identifying females since there are only four females in 40 individuals. Training and testing sets are split so that they consist of distinct individuals.

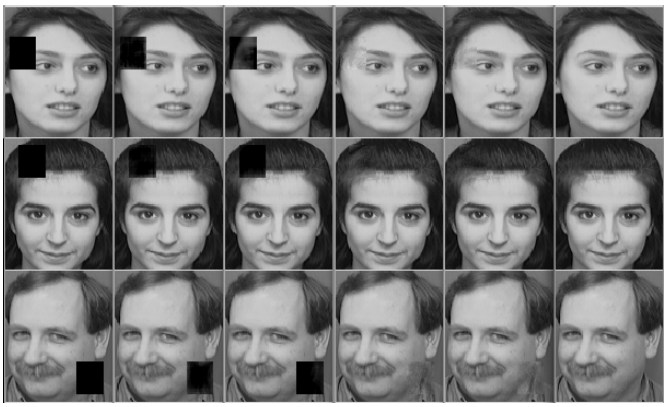

Figure 2. From left to right: The original corrupted image, the reconstructions produced by PCA, spectral embedding, vanilla GLRM, graph-regularized GLRM, and the original uncorrupted image.

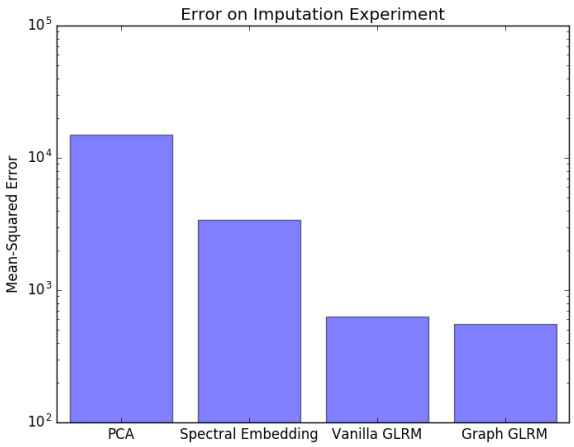

Figure 3. Mean-squared error over the occluded pixels for PCA, spectral embedding, GLRM, and Graph GLRM

Embedding Method	MSE
PCA	15032
SE	3415.4
Vanilla GLRM	634.63
Graph GLRM	**554.48**

Table 2. Precision, recall, and F1-scores for original occluded data, PCA, SE, vanilla GLRM, and graph GLRM

6. Results

6.1. Imputation Experiment

We consider the sensitivity of results to the choice of parameters in the model. We use cross-validation on several values of rank and graph regularization parameter to determine the optimal rank and regularization parameter. The minimal test error was achieved using a rank of 100 and graph regularization parameter of 10.

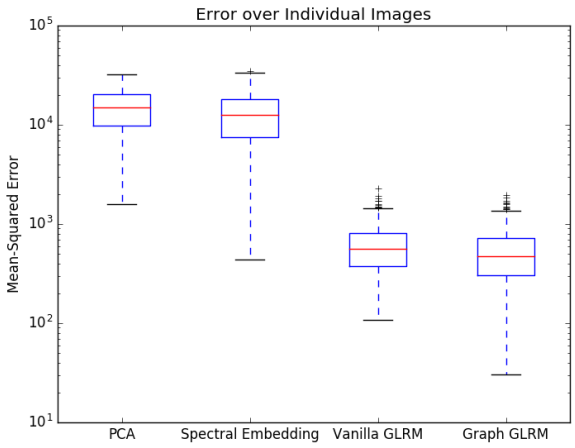

Figure 4. Distribution of average per-pixel error over the occluded pixels of each image for PCA, spectral embedding, GLRM, and Graph GLRM

6.1.1 Comparison with PCA and Spectral Embedding

Figure 3 shows the imputed error we obtained using each method. A visual comparison shows that PCA and Spectral Embedding perform worse and vanilla GLRM and GraphGLRM perform better. We compare the imputed error values quantitatively in table 2 and verify that GraphGLRM indeed performs better. The visual differences between the reconstructions are evident in figure 2.

Specifically, graph regularization induces the smoother transition at the boundaries of the occluded images. This smoothness can be attributed to increased emphasis on learning from images of the same individual.

The reduced error in reconstruction is also apparent in an examination of the mean-squared error of pixel values. Figure 3 shows that the overall mean-squared error was minimized for GGLRM. Furthermore, figure 4 shows that the distribution of errors over each image had a lower median and was more left-skewed than the vanilla GLRM: the graph GLRM was more likely to produce the smallest error on any given image. Additionally, the graph GLRM outperformed PCA on all 400 images, Spectral Embedding on 399 images, and vanilla GLRMs on 307 of the images.

6.2. Classification Experiment

In the classification experiment, rank determination and hyperparameter tuning were both repeated using F1-score (the harmonic mean of precision and recall) as the validation metric. The best F1-score was achieved at a rank of 100 for GraphGLRM and rank 400 for vanilla GLRM. We compare the precision, recall, and F1-score of low-rank embeddings in Table 3.

The graph GLRM attains the highest score in all three

Embedding Method	Precision	Recall	F1-Score
None	0.727	0.4	0.516
PCA	0.381	0.4	0.390
SE	1	0	0
Vanilla GLRM	0.714	0.25	0.370
Graph GLRM	**1**	**0.5**	**0.667**

Table 3. Precision, recall, and F1-scores for original occluded data, PCA, SE, vanilla GLRM, and graph GLRM

metrics. Additionally, the original images (without a low-rank representation) induce better F1-score than all of the other embedding methods.

7. Conclusion

GraphGLRM combines the benefits of graph structure provided by spectral embedding with the flexibility and available robustness in GLRMs. Furthermore, GraphGLRM enables ignoring known occluded data values. This rejection of known occlusions can aid in making a more faithful representation of data for imputation and classification. GraphGLRM also provides extreme flexibility in the choice of loss function and regularization penalties to handle heterogeneous and noisy data. By carefully tuning the parameters associated with the model, performance on imputation and supervised learning tasks can surpass that of the most commonly-used methods.

References

[1] P. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering, 2001. In NIPS 2001. 1

[2] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014. 2, 3

[3] E. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11, 2011. 1

[4] A. d'Aspremont, L. El Ghaoui, I. Jordan, and G. R. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *Advances in Neural Information Processing Systems*, 16:41 – 48, 2004. 1

[5] B. Jiang, C. Ding, B. Luo, and J. Tang. Graph-Laplacian PCA: Closed-form solution and robustness. In CVPR 2011. 1, 2

[6] R. Keshavan and A. Montanari. Regularization for matrix completion., 2010. In 2010 IEEE International Symposium on Information Theory Proceedings (ISIT). 1

[7] M. Udell, C. Horn, R. Zadeh, and S. Boyd. Generalized low-rank models. *Foundations and Trends in Machine Learning*, 9(18):1 – 118, 2016. 1, 3