Dimension reduction for hyperspectral imaging using laplacian eigenmaps and randomized principal component analysis: Midyear Report

Yiran Li
yl534@math.umd.edu

Advisor: Wojtek Czaja
wojtek@math.umd.edu

12/16/2014

Abstract

Hyperspectral imaging has attracted researchers’ interests in recent years. Because of its high dimensionality and complexity, the reduction of dimension of hyperspectral data sets has become crucial in processing and categorizing the data. In this project, I will apply two methods of dimension reduction: laplacian eigenmaps and randomized principal component analysis in order to reduce the dimension of hyperspectral imaging data. Laplacian eigenmaps is a method developed in 2002 for dimension reduction. It is widely used and is essential in understanding dimension reduction. Randomized principal component analysis is a method that is developed later in 2008. It takes an approach different from laplacian eigenmaps, but is more efficient in dealing with large matrices. My goal is to implement these two methods to some hyperspectral data sets, and study in order to understand the differences and similarities between these two methods. This project will also assists me in understanding hyperspectral imaging and in analyzing hyperspectral data sets.

1 Introduction

With the development of hyperspectral sensors, hyperspectral imaging has drawn attention to researchers. In order to understand hyper spectral imaging, we need to learn about the notion of spectrum. Lights are generally described in terms of their wavelength. Different materials receive, transmit and reflect lights of different wavelengths. A spectrum of a certain material gives the percentage of light that is reflected by this material measured across a wide range of wavelengths. (Shippert, 2003). A hyperspectral imaging data set contains the spectrum of each pixel on the image. The data sets are presented as the following:

\[ A(x, y, :) = \phi_{xy} \]

where \( x, y \), are the x-coordinates and y-coordinates of the pixel in the image, and \( \phi_{xy} \) is an n-dimensional vector that gives the spectrum of the pixel at \( (x, y) \), assuming that there are n bands of wavelengths at which we measure the reflectance. If we take images of 1000 pixels, and use \( n = 100 \) bands of wavelengths, we would end up with a data set of size \( 10^8 \). Hence it is easy to see that hyperspectral data set can be large and thus difficult to analyze. Because of the large magnitude, there have been great incentives to reduce the dimensionality of hyperspectral data, in order for us to simplify the data complexity to enable calculations.
Dimension reduction may also help reveal some underlying structures in data for the purposes of target or anomaly detection, and give us better classification and segmentation of the data set.

Various algorithms have been developed in recent years for the purpose of dimension reduction. A list of them are: Principal Component Analysis (PCA), Probabilistic PCA, Factor Analysis (FA), Classical multidimensional scaling (MDS), Linear Discriminant Analysis (LDA), Isomap, Local Linear Embedding (LLE), Local Tangent Space Alignment (LTSA), Fast Maximum Variance Unfolding (FastMVU), Kernel PCA. (Delft University) These algorithms may serve for different purposes, and thus they may differ in operation costs yet be of equal interests to different groups of people.

2 Approach

I will implement two algorithms of dimension reduction: laplacian eigenmaps and randomized principal component analysis. Laplacian eigenmaps is a widely used and well-developed method for dimension reduction. Implementing laplacian eigenmaps is an essential step to take in learning dimension reduction algorithms. The second method that I will use is randomized PCA. It is developed after laplacian eigenmaps and the method is more efficient under certain circumstances. The method of Laplacian eigenmaps is presented below, along with a detailed explanation using hyperspectral imaging as example.

2.1 Laplacian eigenmaps

In the paper Laplacian Eigenmaps for Dimensionality Reduction and Data Representation by Mikhail Belkin and Partha Niyogi, they introduced the method of laplacian eigenmaps as the following.

Let $n$ be the number of data points in, in our case the number of pixels in the image, and let $x_i$ denote the $i$th data point.

Step 1: Constructing the Adjacency Graph

We try to construct a weighted graph with $n$ nodes, and a set of edges connecting neighboring points. There are two ways of constructing edges. The first way is to connect the nodes that are within $\epsilon$ of each other, for some real number $\epsilon > 0$. This means: $x_i, x_j$ are connected if $|x_i - x_j|^2 < \epsilon$.

The second way is to connect the $m$ nearest nodes of any node.

Step 2: Choosing the weights

There are two ways of choosing the weight on the edge. Let $W_{ij}$ denote the weight function between points $x_i, x_j$. The first way is to define the weight on edge connecting point $x_i, x_j$ to be the heat kernel. Indeed, let

$$W_{ij} = e^{-|x_i - x_j|^2/t}$$

where $t$ is a parameter to be specified by the user. The second way is the simple minded way of choosing the weights: let

$$W_{ij} = 1$$

if two nodes are connected and

$$W_{ij} = 0$$

otherwise.
Step 3: Compute eigenvalues and eigenvectors

We will compute the eigenvalues and eigenvectors for the generalized eigenvector problem:

\[ Lf = \lambda Df \]

where \( W \) is the weight matrix defined in step 2, \( D \) is diagonal weight matrix, with \( D_{ii} = \sum_{j=1}^{n} W_{ji} \), and \( L = D - W \). Let \( f_0, f_1 \ldots f_{n-1} \) be the solutions of equation \( Lf = \lambda Df \), ordered such that

\[ 0 \leq \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda(n-1) \]

Then the first \( m \) eigenvectors (excluding \( f_0 \)), \( f_1, f_2, \ldots, f_m \) are the desired vectors for embedding our high dimensional data (n-dimensional) in m-dimensional Euclidean space.

2.2 Randomized Principal Component Analysis

In the paper A Randomized Algorithm for Principal Component Analysis by Vladimir Rokhlin, Arthur Szlam, and Mark Tygert, the method randomized PCA was introduced. In our case, we assume that there are \( m \) pixels in the image. Ordering the pixels as \( x_1, x_2, \ldots, x_m \), we form a matrix \( A \), such that the \( i \)-th row of \( A \),

\[ A_i = \phi_i, \]

where \( \phi_i \) is the spectrum (n-dimensional vector) of pixel \( x_i \). Assume that we want to reduce the dimension of \( A \) from \( m \) to \( k \).

The algorithm in Rokhlin’s paper is presented as the following: A general way of constructing the best rank-\( k \) approximation to a real \( m \times n \) matrix \( A \) uses single value decomposition:

\[ A \approx \tilde{U} \tilde{\Sigma} \tilde{V}^T, \]

where \( U \) real unitary \( m \times m \) matrix, \( V \) is real unitary \( n \times n \) matrix, and \( \Sigma \) is real \( m \times n \) diagonal matrix with nonnegative, non increasing diagonal entries. The best rank-\( k \) approximation of \( A \) is given by

\[ A \approx \tilde{U} \Sigma_k \tilde{V}^T, \]

where \( \tilde{U} \) is the leftmost \( m \times k \) block of \( U \), \( \Sigma_k \) is the \( k \times k \) upper left block of \( \Sigma \), \( \tilde{V} \) leftmost \( n \times k \) block of \( V \). This approximation is considered the best because it minimizes the spectral norm \( ||A - B|| \) for a rank-\( k \)-matrix \( B = \tilde{U} \Sigma_k \tilde{V}^T \). In fact,

\[ ||A - \tilde{U} \Sigma_k \tilde{V}^T|| = \sigma_{(k+1)}, \]

where \( \sigma_{(k+1)} \) is the \((k+1)\)th greatest singular value of \( A \). The randomized PCA generates \( B \) such that

\[ ||A - B|| \leq C m^{1/(4i+2)} \sigma_{(k+1)} \]

with high probability \((1 - 10^i - 15)\), where \( i \) is specified by user, and \( C \) depends on parameters of algorithm. Let us choose \( l > k \) such that \( l \leq m - k \).

Step 1

Generate a real \( l \times m \) matrix \( G \) whose entries are independently, identically distributed normal Gaussian random variables, and compute

\[ R = G(AA^T)^\dagger A. \]
Step 2
Using SVD, form a real nxk matrix Q whose columns are orthonormal, such that
\[ ||QS - R^T|| \leq \rho_{(k+1)} \]
for some kxl matrix S, where \( \rho_{(k+1)} \) is the \((k + 1)^{th}\) greatest singular value of R.

Step 3
Compute
\[ T = AQ. \]

Step 4
Form an SVD of T:
\[ T = U\Sigma W^T, \]
where U is a real mxk matrix whose columns are orthonormal, W is a real kxk matrix whose columns are orthonormal, \( \Sigma \) is a real diagonal kxk matrix with nonnegative diagonal entries.

Step 5
Compute
\[ V = QW. \]
In this way, we get U, \( \Sigma \), V as desired.

In implementing laplacian eigenmaps, we could end up with disconnected graphs because of the way we construct edges. If that is the case, I will run the algorithm for each disconnected part and combine the results for each disconnected graph to get final results. It is also possible that we get different results when using different weight functions and connecting the graph in different ways, so I plan to experiment with two weights functions as described above, and choose a reasonable range of \( \epsilon \)'s to connect the graph based on the distances between data points in my particular data sets.

2.3 Discussion on Laplacian Eigenmaps

In constructing the adjacency graph, we view each pixel as a node on the graph, with the position of the nodes in the n-dimensional space (n is the length of each spectrum) indicated by the spectrum of each pixel. The distance between two nodes, \( x_i, x_j \), is calculated using Euclidean norm:
\[ \text{Distance} = |x_i - x_j|. \]
When choosing the \( \epsilon \) and number of neighboring points \( k \), we compared the distance between different nodes using this norm.
In dimension reduction for hyperspectral imaging, our goal is to reduce the dimension of the spectrum. After applying Laplacian eigenmaps, each resulting eigenfunction \( f_0, f_1, ..., f_{n-1} \) represent one layer of the image, i.e., one image that is mapped onto a one dimensional space. We pick the first m eigenvectors accordingly with the first m smallest eigenvalues so that the key structure of the original graph is preserved as much as possible. The new graph that we reconstructed is the sum of the m layers of images that we get from the eigenvectors. Below is a graph demonstration.
3 Implementation and verification

I implemented laplacian eigenmaps in Matlab, on my personal laptop and on the computers in the computer lab in math department. When constructing the neighboring graph, I implemented two different methods: $\epsilon$ method and k-nearest point method.

There is a matlab toolbox developed by people from Delft University for dimension reduction. It contains numerous methods for dimension reduction, and is publically available. Hence I can use the toolbox to reduce dimension of the same data sets as the ones I tested my algorithms on, and compare the results I get using the toolbox with the result I get implementing my algorithms. Comparing the results I get using two different algorithms on the same data sets might also help me verify my method.

In order to verify my code, I downloaded the Dimension reduction toolbox for matlab from Delft University, which has a function for laplacian eigenmaps. I noticed that they used k nearest point method, so I tested several data sets using my code with k nearest point method and their function. I tested my code on 3 different data sets, and was able to get the same set of eigenvalues for all 3 of them (difference is less than $10^{-12}$), and the percentage of the eigenvectors that differ is small. Below is a table of the comparison of the results between my laplacian eigenmaps and the laplacian eigenmaps function from the DR toolbox when using different number of neighboring points, tested on salinasA. The reduced dimension is set to 100.

<table>
<thead>
<tr>
<th>number of eigenvectors that differ:</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\geq 0.00001$</td>
<td>0.1803</td>
<td>0.1779</td>
<td>0.2292</td>
<td>0.2277</td>
<td>0.1861</td>
</tr>
<tr>
<td>$\geq 0.0001$</td>
<td>0.1669</td>
<td>0.1669</td>
<td>0.2126</td>
<td>0.2102</td>
<td>0.1671</td>
</tr>
<tr>
<td>$\geq 0.001$</td>
<td>0.1068</td>
<td>0.0998</td>
<td>0.1203</td>
<td>0.1032</td>
<td>0.0732</td>
</tr>
<tr>
<td>$\geq 0.01$</td>
<td>0.0050</td>
<td>0.0008</td>
<td>0.0004</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error magnitude between eigenvalues:</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>magnitude of error</td>
<td>$10^{-14}$</td>
<td>$10^{-13}$</td>
<td>$10^{-13}$</td>
<td>$10^{-13}$</td>
<td>$10^{-13}$</td>
</tr>
</tbody>
</table>

4 Data base

Here is a list of data bases that I used:
These data sets are the hyperspectral imaging taken across the United States, and the result of dimension reduction of these data sets are convenient to validate. These data sets are publicly available online, and they are of reasonable size to test my algorithms.

5 Validation and Testing

Implementing two different algorithms on the new databases would lead to interesting results to be compared. Plotting the reduced dimensional space can help us visualize and understand the data with reduced dimension. We expect to get similar results for two different methods, but it is possible that running time differs, and that one algorithm works better than the other under certain circumstances.

After the dimensionality of hyperspectral imaging is reduced, we get an image with each pixel indicated by a new vector. Now our goal becomes to classify these vectors into categories, and compare the result with ground truth data. Ground truth data is the classification of the hyperspectral imaging based on the real objects in the image. Each pixel is classified as some category, for instance, woods, crops, etc, in the ground truth image. Figure 2 shows an example of the original hyperspectral image and the ground truth image. It is the Indian pines data set. I will use a method of classification in order to recover the classification with my data of reduced dimension. If this is done successfully, it shows that my dimension reduction algorithm help preserve the basic structure of the original data set.

5.1 classification method: 1NN classifier

The method that I used to classify each pixel using data from reduced dimensionality is described below. First, pick \( l \) pixels on the image uniformly as the training data set, and label
each pixel according to the ground truth image. Second, find the \( k \) nearest vectors to each training vector, and label them as in the same class as the training vector that they are closest to. This way of labeling is the special case of the kNN classifier, i.e, when \( k = 1 \), we have 1NN classifier.

6 Results

After classification, we can picture the resulting 'ground truth' that I find from the image of reduced dimension and compare it with the real ground truth image. Below are some images I produced from the data sets salinas a scene, and Indian Pines.

To measure the error quantitatively, I calculated the percentage of pixels of my categorization that disagree with the ground truth data. Error percentage is calculated as the following:
Figure 4: Comparison between recovered ground truth image and real ground truth image: SalinasA when dimension is 15
Figure 5: Comparison between recovered ground truth image and real ground truth image: Indian pines(part) when dimension is 50
Figure 6: Comparison between recovered ground truth image and real ground truth image: Indian Pines(part) when dimension is 15
Error percentage = number of pixels that disagree with ground truth image / total number of pixels

Below is a table of the error percentage computed when the reduced dimension is 15 and 50.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>SalinasA</th>
<th>Indian Pines(part)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension=15</td>
<td>0.2592</td>
<td>0.4822</td>
</tr>
<tr>
<td>Dimension=50</td>
<td>0.1953</td>
<td>0.4324</td>
</tr>
</tbody>
</table>

For both cases, I used 300 neighboring points. Based on the paper Dimension reduction on hyperspectral images by Fong, Meiching, the classify rate using laplacian eigenmaps and kmeans classifier is 72.470 when dimension is reduced to 4. Since I’m using a different classification method, 1NN classifier, which is less complicated in terms of construction, it is reasonable to see the size of the difference in classy rate.

There are some possible explanations for the error. It is possible that my training vectors are not good representatives. Since the eigenvalues are almost identical to the results that is generated by the function from toolbox, improving the classification method might be a good solution to reducing error size. For instance, I could try kNN classifier with $k > 1$, and kmeans classifier.

7 Project Schedules and milestones

The timeline for my project and the corresponding milestones in each period are the following.

- October 17th: finish and revise project proposal. (Done)
- October to November, 2014: Implement and test laplacian eigenmaps on databases. Get results for the reduced dimensions of images in databases using laplacian eigenmaps. (Done) prepare for implementation of randomized PCA.
- December, 2014: finish midyear report and presentation. (Done)
- January to March: Implement and test randomized PCA, compare two methods in various situations. Get results for the reduced dimensions of databases using randomized PCA.
- April to May: finish final presentation and final report deliverables.

I will be able to deliver the following items of my project by the end of next spring semester.

- Presentation of data sets with reduced dimensions of both algorithms; (Done)
- Comparison charts in terms of running time and accuracy of two different methods;
- Comparison charts with other methods that are available from the DR matlab toolbox; Databases;
- Matlab codes for both algorithms; (Done for Laplacian eigenmaps)
- Presentations; (Done for Laplacian eigenmaps)
- Proposal; (Done)
- Mid-year report; (Done)
- Final report.
References


