Nonlinear Dimensionality Reduction for Hyperspectral Image Classification

Final Report

Tim Doster (tdoster@umd.edu)
Advisors:
Dr. John Benedetto (jjb@math.umd.edu)
Dr. Wojciech Czaja (wojtek@math.umd.edu)

University of Maryland - College Park

May 10, 2011
Table of contents

1 Introduction

2 Nonlinear Dimensionality Reduction

3 Implementation

4 Hyperspectral Image Classification
Outline

1. Introduction
2. Nonlinear Dimensionality Reduction
3. Implementation
4. Hyperspectral Image Classification
Multidimensional data has many application fields, to a name a few:

- Image Processing
- Multivariate Analysis
- Sensor Arrays
- Data Mining

The “curse of dimensionality” however dictates as the dimensionality of a data set increases the amount of empty space in the data increases as well. This can make some problems intractable and thus a method to reduce the dimensionality but still maintain the intrinsic qualities of the data is needed.
Hyperspectral Images

- A normal digital photograph contains three spectral bands
- A hyperspectral image contains hundreds of spectral bands and thus a more extensive and continuous part of the light spectrum is represented
  - Bands include the visible, near infrared, and short-wave infrared
Why use Hyperspectral Images?

- A hyperspectral image can be rendered into a RGB image by selecting the corresponding red, blue and green spectral bands.
- Hyperspectral images are favored over regular digital images as they provide more information to the analyst.
  - Camouflaging material vs Vegetation
  - Target detection
  - Classification
How are Hyperspectral Images Collected?

- Hyperspectral sensors work by collecting solar radiation that is reflected off of objects on the earth.
- As the solar radiation passes through the atmosphere, strikes the object, and passes back through the atmosphere it is altered.
- The sensors may be mounted on high platforms, flown in planes, or contained in satellites in earth orbit.
- The data that is recorded by the sensor is known as radiance spectrum.
Outline

1. Introduction

2. Nonlinear Dimensionality Reduction

3. Implementation

4. Hyperspectral Image Classification
Local Linear Embedding

Given: \( X = \{X_1, X_2, \ldots, X_n\}, \ X_i \in \mathbb{R}^D \)
Find: \( Y = \{Y_1, Y_2, \ldots, Y_n\}, \ Y_i \in \mathbb{R}^d \)

- Step 1: \( D_{ij} = \|X_i - X_j\| \)
- Step 2: Find \( U_i \), the set of \( k \) nearest neighbors for \( X_i \). Let \( U = \{U_i\}_{i=1}^n \).
- Step 3: Minimize the cost function:
  \[
  E(W) = \sum_{i=1}^{N} \|X_i - \sum_{j \neq i} W_{i,j}X_j\|^2.
  \]
  Subject to \( W(i, l) = 0 \) if \( X_l \notin U_i \) and \( \sum_{j=1}^{N} W(i, j) = 1 \).
- Step 4: Minimize the cost function
  \[
  \Phi(Y) = \sum_{i=1}^{N} \|Y_i - \sum_{j \neq i} W_{i,j}Y_j\|^2
  \]
- Step 4 (alternative) Find the eigendecomposition for \( (I - W)^T(I - W) \) and order eigenvalues as \( \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \). Let \( Y_i = \{V_2(i), V_3(i), \ldots, V_{d+1}(i)\} \)
  where \( V_i \) is the eigenvector associated with \( \lambda_i \).
LLE Diagram

1. Select neighbors

2. Reconstruct with linear weights

3. Map to embedded coordinates

Tim Doster  NLDR for HSI Classification
ISOMAP

Given: \( X = \{X_1, X_2, \ldots, X_n\}, X_i \in \mathbb{R}^D \)

Find: \( Y = \{Y_1, Y_2, \ldots, Y_n\}, Y_i \in \mathbb{R}^d \)

- Step 1: \( D_{ij} = ||X_i - X_j|| \)
- Step 2: Find \( U_i \), the set of \( k \) nearest neighbors for \( X_i \). Let \( U = \{U_i\}_{i=1}^n \).
- Step 3: Find \( S \), the pairwise geodesic shortest distance matrix using Floyd-Warshall or Dijkstra’s Algorithm, using edges from \( U \) and weights from \( D \).
- Step 4: Minimize the cost function
  \[
  \Phi(Y) = \sum_{i,j}^N |H^T S_{i,j}^2 H - H^T ||Y_i - Y_j||^2 H |
  \]
- Step 4 (alternative) Find the eigendecomposition for \( -\frac{1}{2} H^T S_{ij}^2 H \) and order eigenvalues as \( \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \). Let \( Y_i = \{\sqrt{\lambda_1} V_1(i), \sqrt{\lambda_2} V_2(i), \ldots, \sqrt{\lambda_d} V_d(i)\} \) where \( V_i \) is the eigenvector associated with \( \lambda_i \).
ISOMAP Diagram
# Complexity

## Local Linear Embedding

<table>
<thead>
<tr>
<th>Step</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance Matrix</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>KNN Selection (quick sort)</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>Reconstruction Weights (LU factorization)</td>
<td>$O(\frac{2}{3}k^3n)$</td>
</tr>
<tr>
<td>Eigendecomposition (QR algorithm)</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$O(2n^3 + n^2 + \frac{2}{3}k^3n)$</td>
</tr>
</tbody>
</table>

## ISOMAP

<table>
<thead>
<tr>
<th>Step</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean Distance Matrix</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>KNN Selection (quick sort)</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>Geometric Distance (Dijkstra’s Algorithm)</td>
<td>$O(n^2 \log n + n^2k)$</td>
</tr>
<tr>
<td>Eigendecomposition (QR algorithm)</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$O(2n^3 + n^2(1 + \log n + k))$</td>
</tr>
</tbody>
</table>
Landmarks

Landmarks seek to lower the complexity of NLDR by only performing the computationally costly parts of LLE and ISOMAP (eigenproblem, pairwise minimum geodesic distance) on a small subset of the points in the data set.

Landmarks points can be created in 3 ways:

- Random
- Uniform Random or Grid
- Max-Min
Max-Min Landmark Selection

- Step 1: Choose $1 \leq s < l$ seed points at random, adding them to $S$ and removing them from $X$.
- Step 2: For $X_i \in X$ and $S_j \in S$, let $d_i = \min_{j=1:||S||}\{\text{dist}(X_i, S_j)\}$.
- Step 3: Let $d_k$ be the maximum of $\{d_i\}$. Add $X_k$ to the set of landmark points $S$ and remove it from the set of data points $X$. If $||S|| = l$ then done, otherwise go to Step 2.

This method adds additional complexity of $O((l - s) * n)$ but can provide the same results with a much smaller set of landmarks. In the literature it is suggested that the number of landmarks, when using max-min should be the intrinsic dimension plus some small integer.
Given: \( X = \{X_1, X_2, \ldots, X_n\}, X_i \in \mathbb{R}^D \)
Find: \( Y = \{Y_1, Y_2, \ldots, Y_n\}, Y_i \in \mathbb{R}^d \)

- **Step 1:** Choose \( L \subset X \). Let \( \hat{I} = |L| \) and \( \hat{X} = X \setminus L \).

- **Step 2:** Find the pairwise minimum geodesic distance matrix \( S \) for the set of landmark points \( L \).

- **Step 3:** Find the eigendecomposition for \( \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_d \), and their corresponding eigenvectors, \( V_1, V_2, \ldots, V_d \) of \(-\frac{1}{2}(H^T S_{ij}^2 H)\), where \( H = I - \frac{1}{l} 1^T 1 \) is defined as the centering matrix where \( I \) is the identity matrix and \( 1 \) is a vector of ones.
LISOMAP continued

- Step 3 (cont): Let \( B = \begin{bmatrix} \sqrt{\lambda_1}V_1 \\ \sqrt{\lambda_2}V_2 \\ \vdots \\ \sqrt{\lambda_d}V_d \end{bmatrix} \)

- Step 4: Let \( B^\# \) be the pseudo inverse of \( B \).
  Using triagonalization: \( Y_i = B^\#(\delta_i - \delta_\mu) \), where \( \delta_\mu \) is the average distance squared vector between all landmarks and \( \delta_i \) is the distance squared from \( X_i \) to all the landmarks.
## Complexity ISOMAP and LISOMAP

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ISOMAP</th>
<th>LISOMAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean Distance Matrix</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>KNN Selection (quick sort)</td>
<td>$O(n^3)$</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>Geometric Distance (Dijkstra’s Algorithm)</td>
<td>$O(n^2 \log n + n^2 k)$</td>
<td>$O(\ln \log n + n^2 k)$</td>
</tr>
<tr>
<td>Eigendecomposition (QR algorithm)</td>
<td>$O(n^3)$</td>
<td>$O(l^3)$</td>
</tr>
<tr>
<td>Pseudo Inverse (SVD) and Mapping</td>
<td>$\cdot$</td>
<td>$O(ld^2 + (n - l)l)$</td>
</tr>
<tr>
<td>Total</td>
<td>$O(2n^3 + n^2(1 + \log n + k))$</td>
<td>$O(n^3 + n^2 k + \ln \log n + n + l^3)$</td>
</tr>
</tbody>
</table>
L-LLE

Given: $X = \{X_1, X_2, \ldots, X_n\}$, $X_i \in \mathbb{R}^D$
Find: $Y = \{Y_1, Y_2, \ldots, Y_n\}$, $Y_i \in \mathbb{R}^d$

- Step 1: Choose $L \subset X$. $|L| = l$. Let $\hat{X} = X \setminus L$.
- Step 2: Perform LLE steps 1, 2, 3, 4 on $L$ to obtain an embedding $Y$.
- Step 3: For each $x \in \hat{X}$ find the $k$-nearest neighbors of $x$ from $L$ and denote them $l_1, l_2, \ldots, l_k$.
- Step 4: Now find the reconstruction weights, $\mathcal{W} = \{w_1, w_2, \ldots, w_k\}$, such that the cost function, $E(\mathcal{W})$, is minimized subject to $\sum_{i=1}^{k} w_i = 1$.
  \[ E(\mathcal{W}) = |x - \sum_{i=1}^{k} w_i l_i|^2. \]
- Step 5: Let the embedding for $x$ be given by $w_1 l_1 + w_2 l_2 + \cdots + w_k l_k$. 
## Complexity LLE and LLLE

<table>
<thead>
<tr>
<th>Function</th>
<th>LLE</th>
<th>LLLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance Matrix</td>
<td>$O(n^2)$</td>
<td>$O(l^2)$</td>
</tr>
<tr>
<td>KNN Selection (quick sort)</td>
<td>$O(n^3)$</td>
<td>$O(l^3)$</td>
</tr>
<tr>
<td>Reconstruction Weights (LU factorization)</td>
<td>$O(\frac{2}{3}k^3n)$</td>
<td>$O(\frac{2}{3}k^3l)$</td>
</tr>
<tr>
<td>Eigendecomposition (QR algorithm)</td>
<td>$O(n^3)$</td>
<td>$O(l^3)$</td>
</tr>
<tr>
<td>Distance Matrix</td>
<td>·</td>
<td>$O(ln - l^2))$</td>
</tr>
<tr>
<td>KNN Selection (Quick Sort)</td>
<td>·</td>
<td>$O((n - l)^3)$</td>
</tr>
<tr>
<td>Reconstruction Weights (LU factorization)</td>
<td>·</td>
<td>$O(\frac{2}{3}k^3(n - l))$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$O(2n^3 + n^2 + \frac{2}{3}k^3n)$</td>
<td>$O((n - l)^3 + 2l^3 + l^2 + \frac{2}{3}k^3n)$</td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. Nonlinear Dimensionality Reduction
3. Implementation
4. Hyperspectral Image Classification
Implementation

- Code base was developed in C++
- Eigensolver and LU factorization algorithms come from LAPACK and use C wrappers
- OpenMP was used to parallelize most parts of the code
- Sparse data structures used when possible
- Default parameters are made available when possible
- Input and Output done with CSV files with header
Delivered Code

- lle.cpp
- isomap.cpp with Floyd-Warshall and Dijkstra’s Methods
- lle.cpp and isomap.cpp with Random, Uniform-Random, and Max-Min Landmark Selection
- corrdim.cpp - An intrinsic dimension estimator
- Matlab files for classification and displaying embeddings

Example:

```
>> ./lisomap [k] [d] [filename] [p] [m]
>> ./lle [k] [d] [filename]
```
Outline

1. Introduction
2. Nonlinear Dimensionality Reduction
3. Implementation
4. Hyperspectral Image Classification
Copperas Cove Image a.k.a. Urban

- Copperas Cove, Texas
- HYDICE sensor
- 210 spectral bands, $310 \times 310$ pixels, with 3 meter resolution
- Bad bands were removed resulting in 162 spectral bands
- Atmospheric Correction done with QUAC Algorithm
- Image subset is full spectral bands and $75 \times 75$
Introduction
Nonlinear Dimensionality Reduction
Implementation
Hyperspectral Image Classification

Tim Doster
NLDR for HSI Classification
Classification

- Supervised learning takes a set of data with a set of identified training pixels and classifies the rest of data
- I am using the `classify()` method from Matlab
- Quadratic: Uses quadratic decision surfaces to differentiate classes
Isomap

Accuracy [%]

Dimension

Original
p=100%
p=1%
p=2%
p=5%
p=10%
p=15%
p=20%

Isomap

Tim Doster
NLDR for HSI Classification
Introduction Nonlinear Dimensionality Reduction Implementation Hyperspectral Image Classification

Tim Doster NLDR for HSI Classification
Classification with Raw Data: 72%
Best Classification: ISOMAP with 5 Dimensions: 92%
## Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>1: Grass</th>
<th>2: Walmart Roof</th>
<th>3: Road</th>
<th>4-6: Roofs</th>
<th>7: Asphalt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>161</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>74</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>18</td>
<td>10</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>ISOMAP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>164</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>88</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>84</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

1: Grass 2: Walmart Roof 3: Road 4-6: Roofs 7: Asphalt

Tim Doster  | NLDR for HSI Classification
Why use Reduced Dimensional Data?

- Generally better results
- Can have much less training data:
  Most Supervised Learning algorithms require the size of the training set be greater than the dimension of the data
- Classification speed increases (though not enough to offset embedding cost)
Thank you for listening. Any questions?