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

Outline

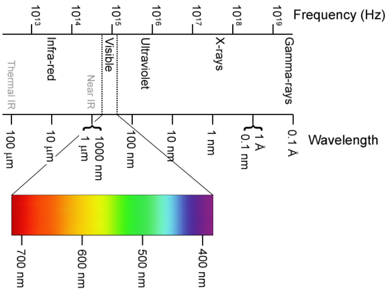
Multidimensional data has many application fields, to a name a few:

- 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

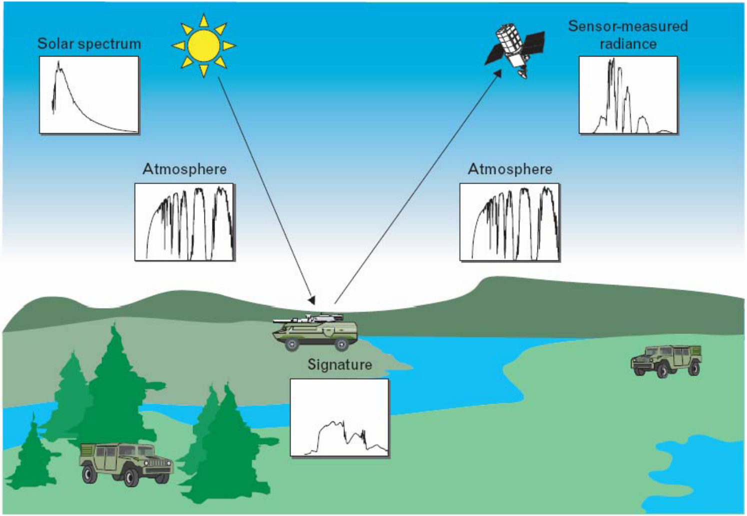


Why use Hyperspectral Images?

- 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
  - Classifcation

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


Solar spectrum
Sensor-measured radiance
Atmosphere
Atmosphere
Signature

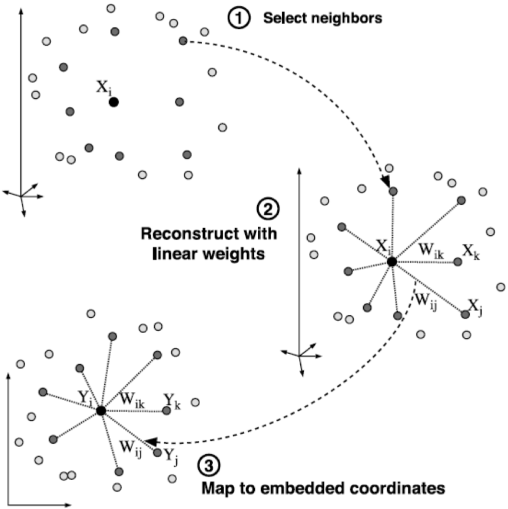
Outline

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 λ_i.

LLE Diagram


1 Select neighbors
X_i
2 Reconstruct with linear weights
X_i
W_ik
X_k
W_ij
X_j
Y_i
W_ik
Y_k
W_ij
Y_j
3 Map to embedded coordinates

ISOMAP

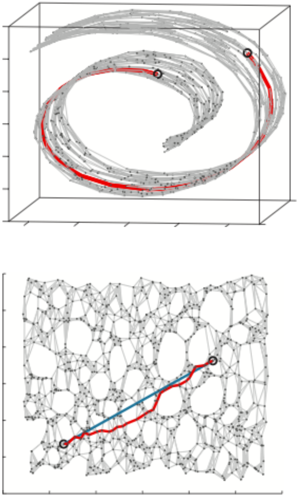
Given: $X = \{X_1, X_2, \ldots, X_n\}$, $X_i \in \mathbb{R}^D$
Find: $Y = \{Y_1, Y_2, \ldots, Y_n\}$, $Y_i \in \mathbb{R}^d$

- 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 λ_i.

ISOMAP Diagram



Complexity

Local Linear Embedding

Distance Matrix	$O(n^2)$
KNN Selection (quick sort)	$O(n^3)$
Reconstruction Weights (LU factorization)	$O(\frac{2}{3}k^3n)$
Eigendecompostion (QR algorithm)	$O(n^3)$
Total	$O(2n^3 + n^2 + \frac{2}{3}k^3n)$

ISOMAP

Euclidean Distance Matrix	$O(n^2)$
KNN Selection (quick sort)	$O(n^3)$
Geometric Distance (Dijkstra's Algorithm)	$O(n^2 \log n + n^2k)$
Eigendecompostion (QR algorithm)	$O(n^3)$
Total	$O(2n^3+$ $n^2(1 + \log n + k))$

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.

L-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: Choose $L \subset X$. Let $\hat{l} = |L|$ 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 eigendecoposition 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=$I - \frac{1}{\hat{l}} 1^T 1$ 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 δ_μ is the average distance squared vector between all landmarks and δ_i is the distance squared from X_i to all the landmarks.

Complexity ISOMAP and LISOMAP

Procedure	ISOMAP	LISOMAP
Euclidean Distance Matrix	$O(n^2)$	$O(n^2)$
KNN Selection (quick sort)	$O(n^3)$	$O(n^3)$
Geometric Distance (Dijkstra's Algorithm)	$O(n^2 \log n + n^2 k)$	$O(ln \log n + n^2 k)$
Eigendecompostion (QR algorithm)	$O(n^3)$	$O(l^3)$
Pseudo Inverse (SVD) and Mapping	·	$O(ld^2 + (n-l)l)$
Total	$O(2n^3 + n^2(1 + \log n + k))$	$O(n^3 + n^2 k$ $ln \log n + n + l^3$

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, $W = \{w_1, w_2, \ldots, w_k\}$, such that the cost function, $E(W)$, is minimized subject to $\sum_{i=1}^{k} w_i = 1$.
$$E(W) = |x - \sum_{i=1}^{k} w_i l_i|^2.$$
- 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

Function	LLE	LLLE
Distance Matrix	$O(n^2)$	$O(l^2)$
KNN Selection (quick sort)	$O(n^3)$	$O(l^3)$
Reconstruction Weights (LU factorization)	$O(\frac{2}{3}k^3n)$	$O(\frac{2}{3}k^3l)$
Eigendecompostion (QR algorithm)	$O(n^3)$	$O(l^3)$
Distance Matrix	·	$O(ln - l^2))$
KNN Selection (Quick Sort)	·	$O((n-l)^3)$
Reconstruction Weights (LU factorization)	·	$O(\frac{2}{3}k^3(n-l)$
Total	$O(2n^3 + n^2 + \frac{2}{3}k^3n)$	$O((n-l)^3 + 2l^3 + l^2 + \frac{2}{3}k^3n)$

Outline

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
- llle.cpp and lisomap.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]

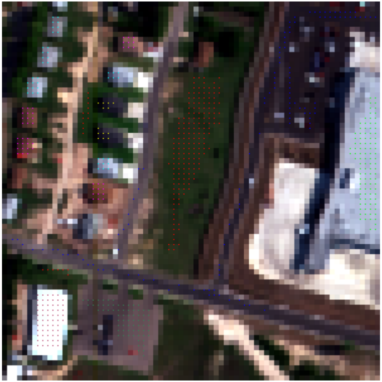
>> ./lle [k] [d] [filename]

Outline

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

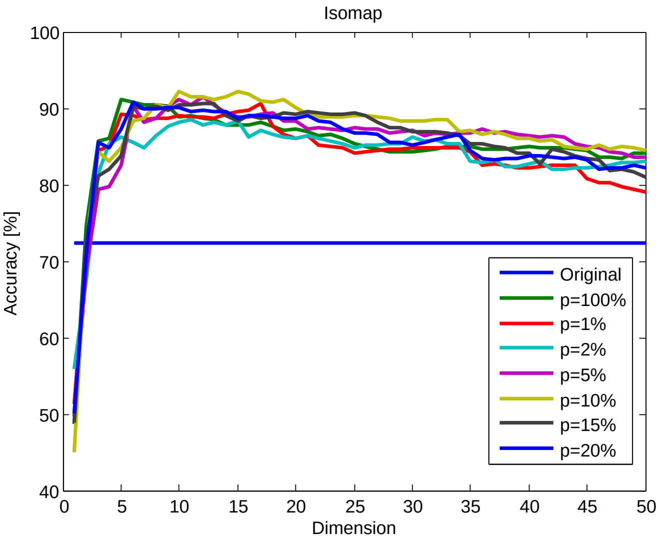
Copperas Cove Image a.k.a. Urban

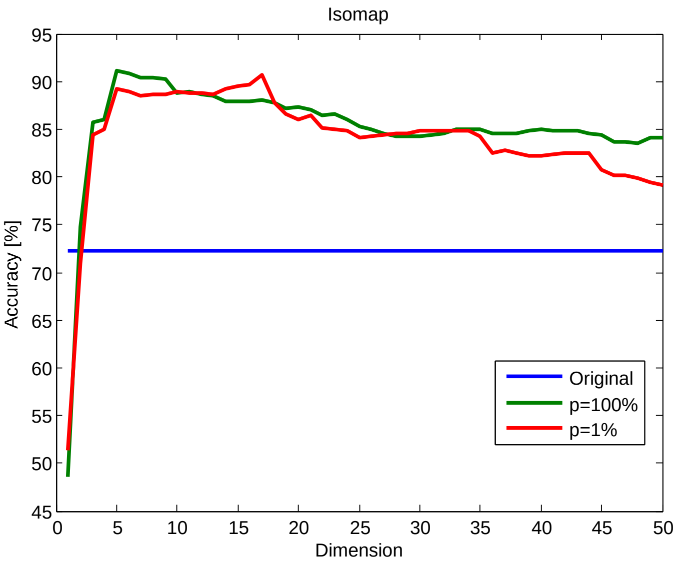
- Copperas Cove, Texas
- HYDICE sensor
- 210 spectral bands, 310×310 pixels, with 3 meter resolution
- Bad bands were removed resulting in 162 spectral bands
- Atmospheric Correction done with QUAC Algorithm
- Image subset is full spectral bands and 75×75



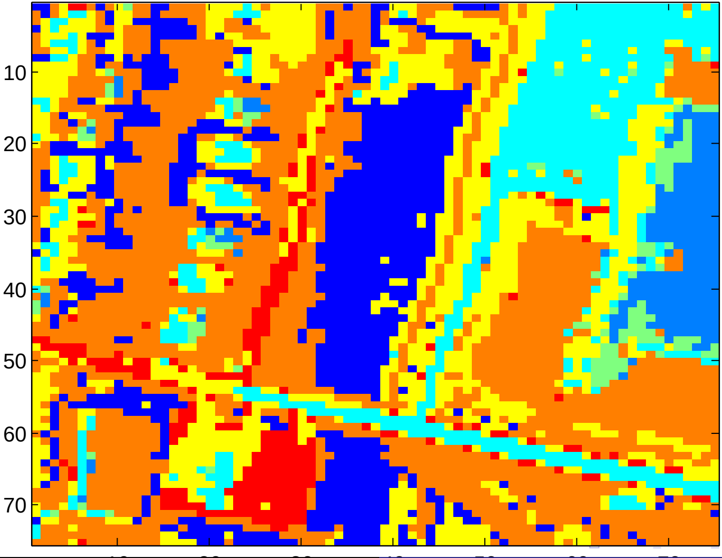
Classification

- 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 form Matlab
- Quadratic: Uses quadratic decision surfaces to differentiate classes

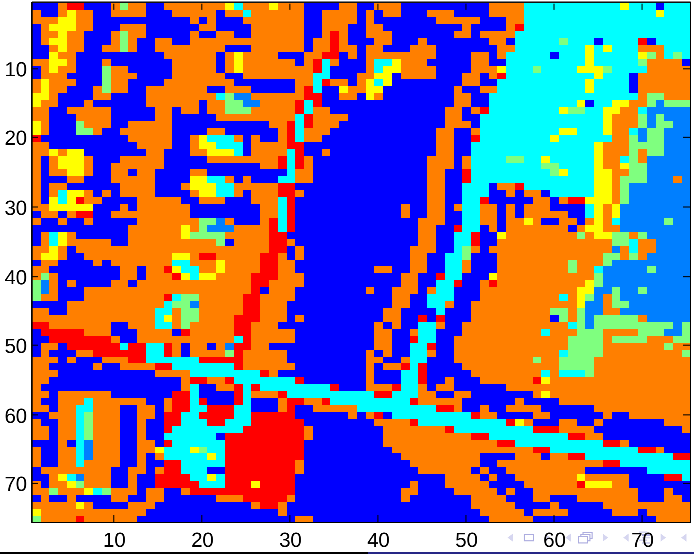

Isomap
100
90
80
70
60
50
40
Accuracy [%]
0
5
10
15
20
25
30
35
40
45
50
Dimension
Original
p=100%
p=1%
p=2%
p=5%
p=10%
p=15%
p=20%


Isomap
95
90
85
80
75
70
65
60
55
50
45
Accuracy [%]
0
5
10
15
20
25
30
35
40
45
50
Dimension
Original
p=100%
p=1%

Classification with Raw Data: 72%



Best Classification: ISOMAP with 5 Dimensions: 92%



Confusion Matrix

Original :

161	0	0	1	0	0	0
0	90	0	3	0	2	0
0	0	74	4	19	0	0
0	0	0	8	0	1	0
1	0	18	10	22	0	27
2	2	4	12	0	35	0
0	0	11	0	1	0	37

ISOMAP:

164	0	3	7	0	0	0
0	88	1	1	0	3	0
0	0	84	0	6	0	14
0	1	0	14	0	4	0
0	0	1	0	19	0	0
0	3	3	16	15	31	0
0	0	15	0	2	0	50

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

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?