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)

May 18, 2011

Abstract

Today with sensors becoming more complex and cost no longer a deterrent to storing large amounts of data; analysts need methods to reduce the volume of stored data and reveal its important facets. Dimensionality reduction, particularly non-linear dimensionality reduction, is a solution to this problem. In this paper, we will look at two nonlinear dimensionality reduction algorithms, Local Linear Embedding and ISOMAP. These algorithms both have been shown to work well with artificial and real world data sets, but are computationally expensive to execute. We solve this problem for both algorithms by applying landmarks or out of sample extensions which perform computationally expensive calculations on a small subset of the data and then map projection onto non-landmark data . Finally, we will apply these algorithms first to artificial data sets for validation and then to hyperspectral images for the application of classification.

1 Background

Dimensionality reduction is a field of mathematics that deals with the complexities of very large data sets and attempts to reduce the dimensionality of the data while preserving the important characteristics of the data. These algorithms are becoming more important today because the complexity of sensors have increased as well as the ability to store massive amounts of data. For example, hyperspectral sensors, which we will discuss below, record roughly a hundred times the amount of information as a typical optical sensor. With this high number of dimensions being recorded it becomes no longer feasible for analysts to examine the data without the assistance of computer algorithms to reduce the number of dimensions but still keep the intrinsic structure of the data intact.

There are two main branches of dimensionality reduction: linear and non-linear. In this project we will focus on non-linear algorithms as they have been shown to perform at least as well as linear algorithms but in many cases much better. We have chosen to study two of the leading non-linear algorithms, of about fifteen [8], in the field of dimensionality reduction: Local Linear Embedding and ISOMAP. The details of these algorithms will be presented in following sections.

A hyperspectral image (HSI), in general, has hundreds of spectral bands in contrast to a normal digital image which has three spectral bands (blue, red, and green) and thus offers a more complete part of the light spectrum for viewing and analysis [9]. This high dimensionality makes HSI good candidates for the methods of dimensionality reduction. A regular digital image can be viewed as a collection of three-dimensional spectral vectors, each representing the information for one pixel. Similarly a hyperspectral image can be viewed as a collection of D-dimensional spectral vectors, each representing the information for one pixel. Hyperspectral images typically include spectral bands representing the ultraviolet (200-400 nanometers), visible (400-700 nanometers), near infrared (700-1000 nanometers), and short-wave infrared (1000-4000 nanometers). In Figure 1, a representation of the light spectrum is shown with the approximate coverage of a hyperspectral image.

Thus, HSI are favored over regular images for some applications such as forestry [16] and crop analysis [7], mineral exploration [12], and surveillance [10]. The spectrum of vegetation, for example, is quite different from that of man-made objects (around 1100-1600 nanometers) even if painted to camouflage in with local vegetation. In this case, a simple photograph would not be able to pick out the man made objects as well as a hyperspectral image [10]. A hyperspectral image can produce a traditional

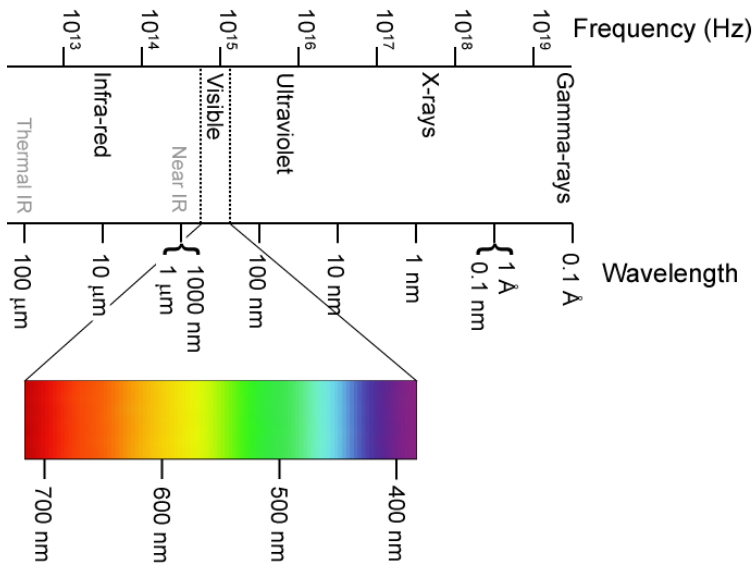

Figure 1: Electromagnetic Spectrum showing the ultra violet, visible, near-infrared, and shortwave infrared

red-blue-green image by resampling the image using the human visual response or any three spectral bands desired.

HSI are collected with special detectors that can be placed on high structures, flown in planes, or contained in satellites. For example, if a plane is used to collect the data, it will record the amount solar radiation reflected back from the ground at specific wavelengths line by line (like a push broom). Later the readings can be assembled, with necessary smoothing done to remove effects from the uneven travel of the plane, into a complete hyperspectral image [14]. The sensor onboard the plane works by collecting the emitted solar radiation that is reflected off the ground or object on the ground. As the solar radiation enters the atmosphere, it is altered by the presence of water molecules and other particulate matter in the atmosphere as shown in Figure 2. The same effect happens once the solar radiation is reflected off the ground or object. The data that are recorded by the sensor are known as the radiance spectrum. The reflectance spectrum for a particular band is the ratio of the reflected radiation at that band to the incident radiation at that band, and can be recovered from the collected radiation spectrum by using atmospheric correction equations. In this paper we will use the Quick Atmospheric Correction (QUAC) algorithm to correct any raw

images[2].

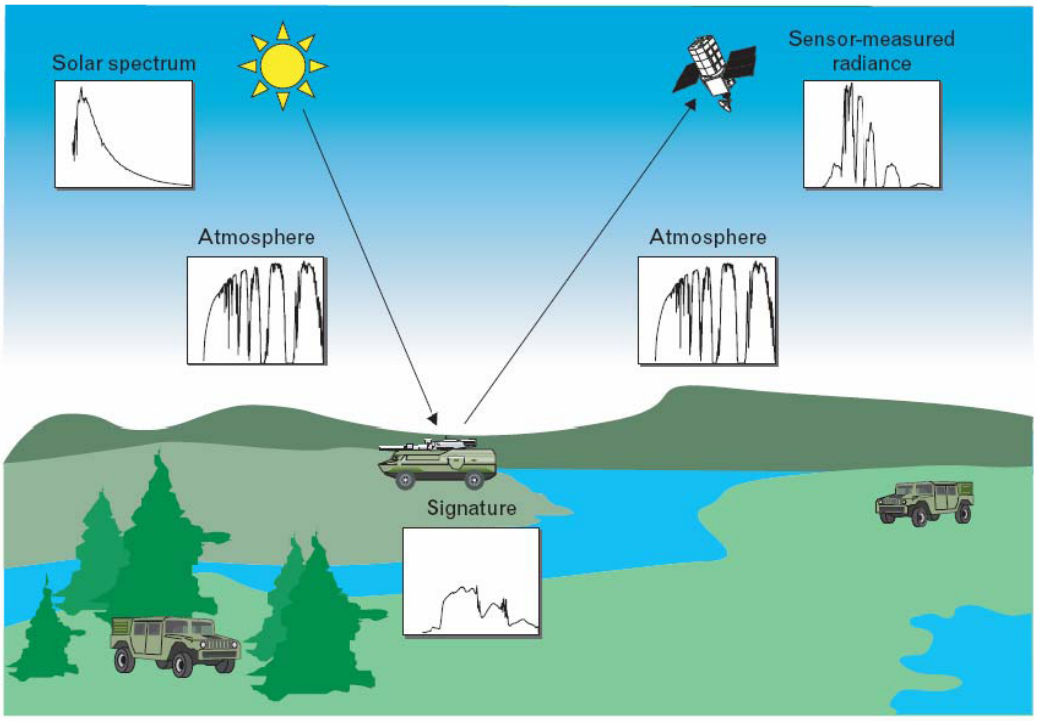

Figure 2: The path of solar radiation from the sun to the hyperspectral sensor (in this case on a satellite) [9]

One of the areas of research into HSI is image classification. The major goal of image classification is to classify the pixels of an image into some number of classes with the use of training data. This allows for example the creation of vegetation maps near wetlands. Bachmann [1] proposes using non-linear dimensionality reduction algorithms to first process the data into a lower dimension before using classification algorithms on the data set. This allows for the similarities and dissimilarities of the data members to become more evident as well as reducing the computation time (though this is greatly offset by the dimensionality reduction algorithm complexities).

2 Dimension Reduction Algorithms

We will apply Local Linear Embedding and ISOMAP to our hyperspectral images because due to the similarity among spectral bands that are next to each other, hyperspectral data can be assumed to lie on a lower dimensional manifold.

2.1 Local Linear Embedding

Local Linear Embedding (LLE) [13, 11] developed by Saul and Roweis is a nonlinear manifold-based approach to dimensionality reduction. LLE seeks to preserve the local properties for each data point when the data is projected to a lower dimension.

2.1.1 Algorithm

Step 0: Let $X = \{X_1, X_2, \ldots, X_n\}$ be a set of vectors (in our case the spectrum of each pixel) with $X_i \in \mathbb{R}^D$.

Step 1: Create a directed graph, G_k, for the data set X, where node X_i is connected to node X_j if it is one of the k-nearest-neighbors (KNN) of X_i. Any metric can be used for the KNN calculation but Euclidean (which we will use) and spectral angle (the angle between two pixels when viewed as vectors in R^D) are the most common. For X_i calculate the vector $E = [E_1, E_2, \ldots, E_n]$ where $E_j = ||X_i - X_j||_2$ and let E' be a vector corresponding to the indices of E sorted such that the smallest distances appear first. Let $U_i = [E'_2, E'_3, \ldots, E'_{k+1}]$, we will call U_i the KNN for X_i. Let $U = \{U_i\}_{i=1}^N$.

A directed graph is a collection of objects called nodes with an associated set of ordered pairs of nodes called edges. Each edge has a weight attached to it defining the distance between the nodes it connects. A directed graph differs from a graph in that edges are only traversable in one direction.

Step 2: Calculate the reconstruction weights, W, by minimizing the cost function:

$$E(W) = \sum_{i=1}^{N} |X_i - \sum_{j \neq i} W_{i,j} X_j|^2.$$

To find $W(i, j)$, the cost function is minimized subject to $W(i, l) = 0$ if $X_l \notin U_i$ and

$\sum_{j=1}^{N} W(i,j) = 1$. By forcing the weights to sum to 1 we are removing the effects of translations of points. The use of the cost function ensures that points are not dependent upon rotations and rescaling. Now the set of weights will represent the underlying geometric properties of the data set.

In practice we find the reconstruction weights, W_i, for each X_i, by finding $C = A^T A$, where $A = [X_{U_{i,1}}, X_{U_{i,2}}, \ldots, X_{U_{i,k}}] - [X_i, X_i, \ldots, X_i]$ or the matrix of X_i centered neighbors of X_i. We then solve $CW_i = 1$ and let $W_i = W_i / \sum W_i$. Now we let the sparse matrix $W = [W_1, W_2, \ldots, W_n]$.

Step 3: Now by use of a similar cost function we will map each X_i to a lower dimensional Y_i. The cost function we are minimizing is:

$$\Phi(Y) = \sum_{i=1}^{N} |Y_i - \sum_{j \neq i} W_{i,j} Y_j|^2,$$

and we minimize it by fixing $W(i,j)$ and optimizing Y_j. Saul and Roweis were able to show that minimizing the cost function is equivalent to finding the $d+1$ smallest eigenvalues, $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{L+1}$, and their corresponding eigenvectors, $V_1, V_2, \ldots, V_{d+1}$ of the matrix $(I - W)^T (I - W)$. We reject the smallest eigenvector as it is the unit vector with eigenvalue 0. Now we use the remaining eigenvectors to map X from D dimensions to d dimensions: $X_i \mapsto (V_2(i), V_3(i), \ldots, V_{d+1}(i))$.

2.1.2 Complexity

Local Linear Embedding using naivee implementation and worst case big-oh notation, $O(\cdot)$, requires:

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

2.2 ISOMAP

Isometric Feature Mapping (ISOMAP) [17] developed by Tenenbaum, Silva, and Langford, is a nonlinear manifold based approach to dimensionality reduction like LLE. ISOMAP tries to maintain the geodesic distances between points in the data set when the data is projected down. By focusing on geodesic distance and not the distance in the higher dimensional space ISOMAP is less prone to short circuiting. The algorithm is as follows:

2.2.1 Algorithm

Step 0 and 1: Apply Step 0 and 1 from LLE algorithm.

Step 2: Let G be a graph constructed from the information in G_K. The edge (i, j), the distance from X_i to X_j, will be defined as the pairwise Euclidean distance if $X_j \in U_i$ and if $X_j \notin U_i$ then the distance will be ∞. Now find the shortest geodesic distance matrix S such that $S_{i,j}$ is the minimum geodesic distance between X_i and X_j squared. To find the pairwise shortest path distance we will use either Floyd-Warshall [5] or Dijkstra's algorithm [5].

Step 3: Find the optimal embedding by minimizing the cost function:

$$\Phi(Y) = || - \tfrac{1}{2}H^T SH + \tfrac{1}{2}H^T CH||_F$$

where C is the matrix of square pairwise distances, $C_{ij} = ||Y_i - Y_j||^2$, H=$I - \frac{1}{n}11^T$ is defined as the centering matrix where I is the identity matrix and 1 is a vector of ones and $||A||_F = \sqrt{\sum_{ij} |A_{ij}|^2} = \sqrt{Tr(AA^T)}$ is the Frobenius norm. Tenenebaum showed that minimizing the cost function is equivalent to finding the d principle eigenvalues, $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_d$, and their corresponding eigenvectors, $V_1, V_2, \ldots, V_d$ of $\frac{1}{2}H^T SH$. Now we map X from D dimensions to d dimensions: $X_i \mapsto (\sqrt{\lambda_1}V_1(i), \sqrt{\lambda_2}V_2(i), \ldots, \sqrt{\lambda_d}V_d(i))$.

2.2.2 Floyd-Warshall Algorithm

Floyd-Warshall is a $O(n^3)$ algorithm for finding the shortest pairwise distance in a graph with n nodes. It requires only that the graph contain no negative weights or distances between nodes (or pixels in our case) which will not occur due to our problem definition. The algorithm is as follows:

Step 1: Create a matrix M from a graph G such that $M(i,j)$ corresponds to the weight between node i and node j. If node i and j are not connected in G then $M(i,j) = \infty$; $M(i,i) = 0$.

Step 2: For node $a = 1, 2, 3, \ldots, n$ let $M(i,j) = \min\{M(i,j), M(i,a) + M(a,j)\}$. This step checks to see if it is shorter to travel from node i to node j through an intermediary node a. Here we are looping over all (i,j) pairs, for each a, ordered on i.

The algorithm insures that the shortest path possible from node i to node j using intermediary nodes $1, 2, \ldots, a$ is recorded for $M(i,j)$. When a=n, then the shortest possible pairwise geodesic distances have been found.

2.2.3 Dijkstra's Algorithm

Dijkstra's Algorithm is a $O(nk + n\log n) = O(n(k + \log n)$ for finding the shortest distance from a node i to all other nodes in a graph with n nodes and k nearest neighbors. We will extend the algorithm by iterating it over all nodes in our graph thus increasing the complexity to $O(n^2(k + \log n))$[3]. The algorithm is as follows

Step 1: Create a matrix M from a graph G such that $M(i,j)$ corresponds to the weight between node i and node j. If node i and j are not connected in G then $M(i,j) = \infty$; $M(i,i) = 0$.

Step 2: For node $a = 1, 2, 3, \ldots, n$ complete steps 3-6

Step 3: Create an empty set S which will hold nodes that have been visited. Set the current node $z = a$. Let $S = S \cup \{z\}$. Create the vector B where $B(i) = \infty$ for $i \neq a$ and $B(a) = 0$.

Step 4: For $i \notin S$ let $B(i) = \min\{B(i), B(z) + M(z,i)\}$.

Step 5: Let $z = i$, where $i = \min B(i)$ such that $i \notin S$. Let $S = S \cup \{z\}$. If $S = \{1, 2, \ldots, n\}$ then go to Step 6 otherwise go to Step 4.

Step 6: Let M(i,:) = B.

2.2.4 Comparison

Dijkstra's algorithm will take less operations than the Floyd-Warshall algorithm but requires more memory and initialization time. For small data sets we will use Floyd-Warshall.

2.2.5 Complexity

ISOMAP using naivee implementation, Dijkstra's Algorithm and worst case big-oh notation, $O(\cdot)$, requires:

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

2.3 Landmarks

Landmarks work by doing the computationally complex work on a small subset of the data points of size δ, which are either chosen at random or intelligently, and then applying their mapping to the other non-landmark points in such a way as to minimize the error between the normal embedding and the landmark embedding. By using landmarks we reduce the complexity of finding KNN to $O(\delta \log \delta)$ and solving the eigensystem in $O(\delta^2)$. Applying the mappings to the other data points will have much less complexity than finding the KNN and solving the eigensystem for all the points. The tradeoff is of course accuracy in the final embedding.

There are many methods to select sets of landmark points but we have chosen to implement three of the more popular methods. The first method, which we call random, is by taking a random selection of the points. The second method, which we call the grid method, is by picking a random seed pixel then moving to the right a set number pixels and then adding adding that pixel to the landmark set. We then repeat the process until we have selected enough pixels for our landmark sets. As opposed to the random method this method with just slightly more overhead will give a more uniform sampling of the image. The third method, which we will call the max-min method, seeks to create an optimal set of landmark points where the addition of each landmark point maximizes the minimum distance from the set of landmarks to all other non-landmark points. The algorithm for the max-min method is presented below for a set of X data points, a set of landmark points $L = \emptyset$, a desired number of landmark points l, and an initial number of seed points s.

2.3.1 Max-Min Algorithm

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.

The max-min method has additional complexity of $O((l - s) * n)$ over the random methods, where n is the number of data points in X but has the advantage of creating a much smaller set of landmark points then that needed by the random methods to get approximately the same results. The method also has the advantage that it creates a repeatable set of points each time. It is customary when using the max-min method to set l equal to the intrinsic dimension of the data plus some small padding for safety.

2.4 Landmark Local Linear Embedding

Landmark Local Linear Embedding (LLLE) is an extension of LLE which uses a small subset of points, called landmarks, to perform the costly embedding operations

on and then extends this embedding to non-landmarkpoints. The landmark set is embedded according to regular LLE which entails

2.4.1 Algorithm

Step 1: Choose $L \subset X$ by either using random, grid, max-min, or some other landmark selection criterion to choose l landmark points. Let $\hat{X} = X \setminus L$.

Step 2: Perform LLE steps 1,2 and 3 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 $\sum_{i=1}^{k} = 1$.

$$E(W) = |x - \textstyle\sum_{i=1}^{k} w_i l_i|^2.$$

Step 5: Let the embedding for x be given by $w_1 l_1 + w_2 l_2 + \cdots + w_k l_k$.

2.4.2 Complexity

Comparing the complexity of LLE and LLLE (with random landmarking method) we see that LLLE has a large computational savings when $l << n$:

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

If we instead used the max-min method for landmark selection we would have an additional complexity of $O((l-s)n)$.

2.5 Landmark ISOMAP

Landmark ISOMAP (LISOMAP), similar to LLLE, uses landmark points to lessen the computational cost of ISOMAP

2.5.1 Algorithm

Step 1: Choose $L \subset X$ by either using random, grid, max-min, or some other landmark selection criterion to choose l landmark points. Let $l = |L|$ and $\hat{X} = X \setminus L$.

Step 2: Find the pairwise minimum geodesic squared distance matrix S from the set of landmark points L to the set of points X.

Step 3: Find the optimal embedding for the landmark points by minimizing the cost function:

$$\Phi(Y) = || - \tfrac{1}{2}H^T SH + \tfrac{1}{2}H^T CH||_F$$

where C is the matrix of square pairwise distances, $C_{ij} = ||Y_i - Y_j||^2$, H=$I - \frac{1}{l}11^T$ is defined as the centering matrix where I is the identity matrix and 1 is a vector of ones and $||A||_F = \sqrt{\sum_{ij} |A_{ij}|^2} = \sqrt{Tr(AA^T)}$ is the Frobenius norm. Tenenebaum showed that minimizing the cost function is equivalent to finding the d principle eigenvalues, $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_d$, and their corresponding eigenvectors, $V_1, V_2, \ldots, V_d$ of $\frac{1}{2}H^T SH$.
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 vector between all landmarks and δ_i is the

distance from X_i to all the landmarks. Note that Tenenebaum showed that this triagonalization technique preserved the original mapping of the landmark points.

2.5.2 Complexity

Procedure	ISOMAP	LISOMAP
Euclidean Distance Matrix	$O(n^2)$	$O(nl)$
KNN Selection (quick sort)	$O(n^3)$	$O(nl^2)$
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	$\cdot$	$O(ld^2 + (n-l)l)$
Total	$O(2n^3 + n^2(1 + \log n + k))$	$O(n^2 k + nl^2 ln(\log n + 1) + n + l^3$

2.6 Correlation Dimension

The Correlation Dimension [6, 4] is an intrinsic dimensionality estimator that can be used to compute the approximate intrinsic dimension of a data set without knowing the true nature of the manifold it lies on. The Correlation integral, C_2 is defined below

$$\begin{aligned} C_2(\varepsilon) &= \lim_{N\to\infty} \frac{1}{N(N-1)} \sum_{i<j}^{N} H(\varepsilon - ||y(i) - y(j)||_2) \\ &= P(||y(i) - y(j)||_2 \leq \varepsilon) \end{aligned}$$

where H is the Heaviside function and P is the proportion of distances less than ε. $C_2(\varepsilon)$ basically tallies all nodes that are within ε of another node. If we look at $C_2(\varepsilon)$ for a single node only, and increase ε, $C(\varepsilon)$ will increase a hypervolume in d-dimensional space:

$$C_2(\varepsilon) \sim \varepsilon^d$$
$$d = \lim_{\varepsilon\to 0} \frac{log(C_2(\varepsilon))}{log(\varepsilon)}$$

2.6.1 Algorithm

Step 1: Find the matrix $D_{ij} = ||X_i - X_j||^2$, which holds all the pairwise distances.

Step 2: Choose $r \in Z$. Let $\varepsilon_1 = \min\{D\}$ and $\varepsilon_r = \max\{D\}$. Create a set, $\varepsilon = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_r\}$ with equal spacing.

Step 3: For each ε_i let C_i be the number of distances that are less than ε_i. Let $C = \{C_i\}$

Step 4: Calculate the slope of $\log \varepsilon$ vs $\log C$ where the slope is approximately constant.

3 Implementation

To complete the objectives of this project we have implemented LLE and ISOMAP in C++. Both algorithms were modeled after the founding papers and the implementation in the Matlab Dimensionality Toolbox. We have chosen to implement these algorithms in C++ to improve their speed, memory management, and make them more open. The implementation of LLE and ISOMAP were given the same inputs for simplicity; the number of nearest neighbors to use, the dimension to project down to, and the location of the data file.

Parallelization was added through OpenMP to make use of non-dependent calculations in the algorithms. In LLE we were able to improve performance by parallelizing the computation of the KNN, the graph weights, and the extensions from landmark to non-landmark points. In ISOMAP again we were able to parallelize the the computation of the KNN, the calculation of graph weights (Floyd-Warshall and Dijkstra's algorithms), and the extension from landmark to non-landmark points. For the addition of a reasonable number of processor we saw approximately linear improvement in the speed of computation. We were unable to parallelize the eigensolvers as LAPACK does not support OpenMP parallelization like the Math Kernel Library (MKL).

3.1 Libraries

We have made use of the Boost Ublas and LAPACK. Boost UBLAS contains containers for matrices and vectors and has access to the Basic Linear Algebra levels I,II,III Fortran code base. This allows use to efficiently do vector-vector products, matrix-vector products, matrix-matrix products, calculate norms, and to solve a system of linear equations (Ax=b). We also made use LAPACK function DGEEV and DYSEV which allowed us to find the eigenvalues and eigenvectors of a general real matrix and a general real symmetric matrix respectively. As we choose to program in C++ we made use of wrapper functions dgeev.h and dsyev.h for LAPACK created by Scot Shaw [15]. These functions simply create a clean C++ type method and convert any data types needed for calling the LAPACK libraries.

Installation of the codebases was handled through Macports which optimized them automatically for our current architecture.

3.2 Software

We used MATLAB 2010B and in particular the Statistics and Dimensionality Reduction Toolboxes. We also used ENVI 4.8 to perform operations on the data cube.

3.3 File IO

The data file format we have chosen in comma separated values (csv) with a header line. The header line contains: [number of rows], [number of columns], [dimensionality of data], where all numbers are integers greater than 0. We choose to use a header line because it makes it quicker and simpler to read the data file into our program. Knowing the number of rows and columns also gives us the opportunity to correctly reconstruct an image that is passed in. Other than the first line of the data file each line corresponds to the spectrum for an individual pixel.

When we write out output, the lower dimensional mapping, we create the file filename.[lle or isomap].out. We first write a header line: [number of rows], [number of columns], [dimensionality of data], where all numbers are integers greater than 0. After the first line we write the embedding for each pixel using csv.

3.4 Compiling

The code can be compiled, assuming the libraries are installed correctly, using g++ as such:

```
>>g++ -Wno-write-strings -fopenmp lle.cpp /path/to/liblapack.dylib
-I /path/to/boost_1_44_0/ -o lle
```

```
>>g++ -Wno-write-strings -fopenmp isomap.cpp /path/to/liblapack.dylib
-I /path/to/boost_1_44_0/ -o isomap
```

```
>>g++ -Wno-write-strings -fopenmp llle.cpp /path/to/liblapack.dylib
-I /path/to/boost_1_44_0/ -o llle
```

```
>>g++ -Wno-write-strings -fopenmp lisomap.cpp /path/to/liblapack.dylib
-I /path/to/boost_1_44_0/ -o lisomap
```

```
>>g++ -Wno-write-strings -fopenmp cordim.cpp
-I /path/to/boost_1_44_0/ -o corrdim
```

3.5 Execution

The executables can be run as such assuming they were correctly compilied. Only the path to datafile and the intrinsic dimension are required for the program to run as the the number of nearest neighbors defaults to 12, the percent of landmark points defaults to 5, and the landmark method defaults to max-min. The landmark selection method is chosen by inputting 1 for random, 2 for grid and anything else for max-min.

```
>>./lle [./path/to/datafile] [number of nearest neighbors]
[intrinsic dimensionality]
```

```
>>./isomap [./path/to/datafile] [number of nearest neighbors]
[intrinsic dimensionality]
```

```
>>./lle [./path/to/datafile] [number of nearest neighbors]
[intrinsic dimensionality] [percent landmarks] [landmark selection method]


>>./isomap [./path/to/datafile] [number of nearest neighbors]
[intrinsic dimensionality] [percent landmarks] [landmark selection method]


>>./corrdim [./path/to/datafile]
```

4 Validating Implementation

To validate out C++ implementation we compared results from C++ to the Matlab Dimension Reduction Toolbox for the Swiss Roll, Broken Swiss Roll, Twin Peaks, and Helix, which are defined in three dimensions but are known to lie on a two dimensional manifold. Due to the topological nature of these structures, they are perfect for dimensionality reduction testing since we can be sure that the data actually lies in the plane.

We created data sets using the Matlab Dimensionality Reduction Toolbox function generate_data for the Swiss Roll, Broken Swiss Roll, Helix, and Twin Peaks for 100,250,500,750, and 1000 data points. This data was written to a csv file with a proper header line for use with the C++ code (the Matlab algorithms are also able to read these data files so we be assured that they both have the same data).

To compare the Matlab embedding, $V = [v_1, v_2]$, with the C++ embedding, $W = [w_1, w_2]$, we used the L2 norm difference, $||w_1 - v_1|| + ||w_2 - v_2||$, and max element difference, $\max\{\max\{|w_1 - v_1|\}, \max\{|w_2 - v_2|\}\}$. Since different eigensolver implementations normalize eigenvectors in slightly different ways we normalize the C++ embedding to the Matlab embedding by finding the ratio of the first elements in each embedding dimension.

Due to the some additional code in the Matlab Dimensionality Reduction Toolbox which removes outlier points (something we are not interested in doing) and the fact that some of the data sets need more nearest neighbors to construct a good embedding we had to vary the number of nearest neighbors supplied to the code slightly as seen in the tables in the subsequent subsections.

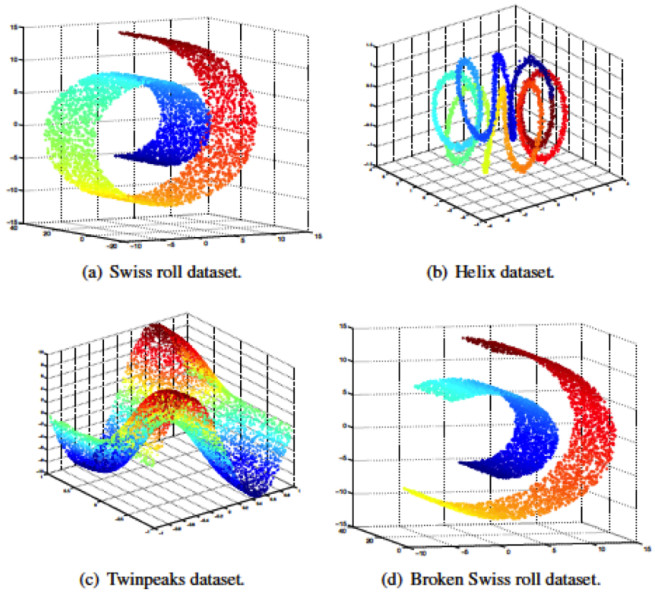

Figure 3: Clockwise from top right: Helix, Broken Swiss Roll, Twin Peaks, Swiss Roll[8].

4.1 LLE

For LLE we had to use a slightly different eigensolver than that used in MATLAB so some additional error between the implementations was expected. Despite these set backs we were able to show good correspondence between the two implementations (see Table 1 and Figure 4, especially for the Swiss Roll and Helix data sets, the sets which are easier to embed. When we consider that our data files had 1E-3 accuracy, the two embeddings corresponded well below this for most of the test data sets. Considering these results we can say that lle.cpp is working properly.

4.2 ISOMAP

For ISOMAP we were able to show near perfect correspondence between the C++ and established Matlab implementation using the normed difference and max difference metric. The difference between the two embeddings were very low (and close to machine precision, E-16), as seen in Table 2 and Figure 5, so we can claim that the

LLE				
Data Set	# points	# neighbors	Normed Difference	Max Difference
Swiss Roll	100	12	1.76E-5	3.67E-6
	250	12	7.31E-4	9.50E-5
	500	12	1.62E-4	1.66E-5
	750	12	6.79E-4	4.72E-5
	1000	12	1.21E-2	8.13E-4
Broken Swiss Roll	100	12	4.35E-4	9.39E-5
	250	12	1.58E-4	1.74E-5
	500	14	4.77E-4	4.58E-5
	750	18	8.46E-1	6.74E-2
	1000	29	3.74E-5	2.87E-6
Helix	100	12	1.02E-5	1.82E-6
	250	12	5.29E-5	7.35E-6
	500	12	8.32E-6	7.57E-7
	750	12	7.05E-5	5.22E-6
	1000	12	1.79E-3	1.09E-4
Twin Peaks	100	12	5.47E-5	8.09E-6
	250	12	3.01E-1	4.54E-2
	500	20	8.03E-4	7.36E-5
	750	30	4.23E-3	3.42E-4
	1000	40	2.59E-3	1.57E-4

Table 1: Normed Difference and Max Difference for LLE validation.

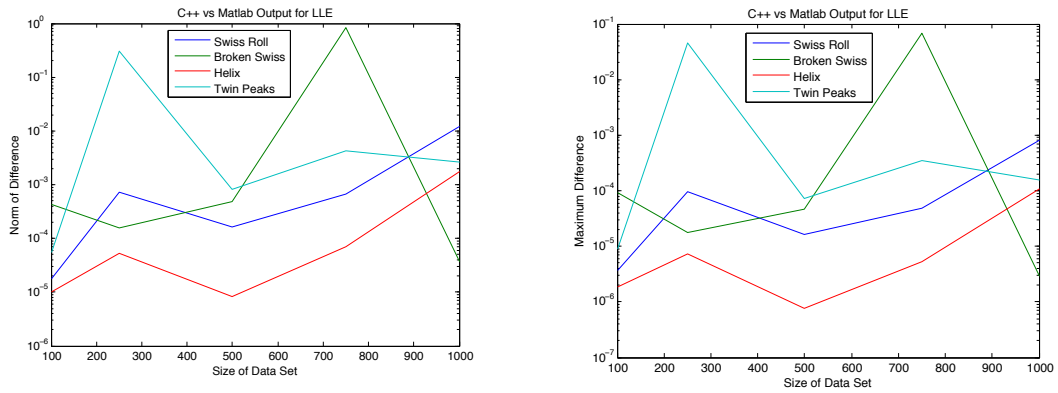

Figure 4: Results for validation test for LLE.

isomap.cpp program is working correctly.

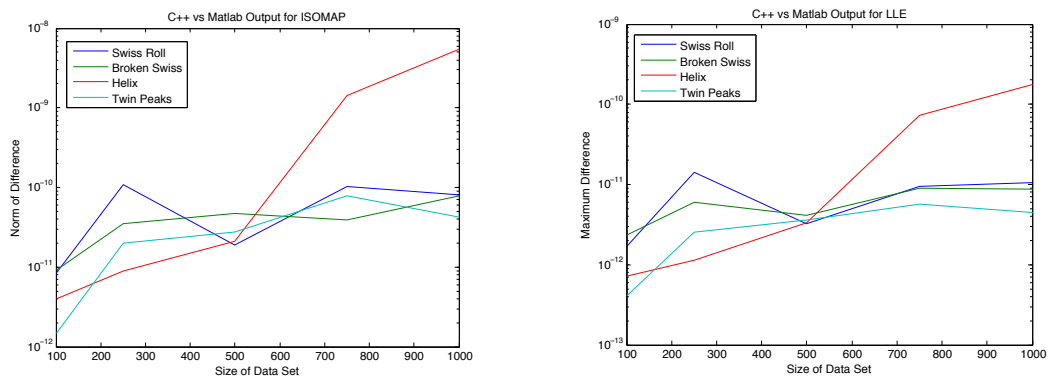

Figure 5: Results for validation test for ISOMAP.

4.3 LLLE

The Dimension Reduction Toolbox does not have an implementation of LLLE so we can not validate against the toolbox as we did with LLE and ISOMAP; instead we look at the results of the image classification presented later. These results show that, with some error produced by the randomness of their selection, that as the percentage

ISOMAP				
Data Set	# points	# neighbors	Normed Difference	Max Difference
Swiss Roll	100	12	8.44E-12	1.71E-12
	250	12	1.09E-10	1.43E-11
	500	12	1.87E-11	3.20E-12
	750	12	1.02E-10	9.43E-12
	1000	12	8.17E-11	1.05E-12
Broken Swiss Roll	100	12	9.27E-12	2.32E-12
	250	12	3.51E-11	6.00E-12
	500	14	4.66E-11	4.15E-12
	750	18	3.89E-11	8.92E-12
	1000	29	7.87E-11	8.77E-12
Helix	100	12	3.98E-12	7.28E-13
	250	12	8.91E-12	1.14E-12
	500	12	2.10E-11	3.35E-12
	750	12	1.41E-9	7.22E-11
	1000	12	5.48E-9	1.76E-10
Twin Peaks	100	12	1.48E-12	4.07E-13
	250	12	2.02E-11	2.54E-12
	500	20	2.74E-11	3.65E-12
	750	30	7.91E-11	5.66E-12
	1000	40	4.19E-11	4.51E-12

Table 2: Normed Difference and Max Difference for ISOMAP validation.

of landmarks approaches 100% the classification accuracy of LLLE approaches that of LLE.

4.4 L-Isomap

The Dimension Reduction Toolbox does not have an implementation of LISOMAP so we can not validate against the toolbox as we did with LLE and ISOMAP; instead we look at the results of the image classification presented later. These results show that, with some error produced by the randomness of their selection, that as the percentage of landmarks approaches 100% the classification accuracy of LISOMAP approaches that of ISOMAP.

4.5 Floyd-Warshall

Given that the Floyd-Warshall algorithm is integral to ISOMAP, and we validated ISOMAP, by extension we have validated the Floyd-Warshall algorithm.

4.6 Dijkstra

Given that Dijkstra's algorithm is integral to ISOMAP, and we validated ISOMAP, by extension we have validated Dijkstra's algorithm.

4.7 Correlation Dimension

To validate the correlation dimension algorithm we use a data described in [6] made by finding the distances to ten random points in $[-1, -1]^3$ from a set of N random points. This will create a ten dimensional data set which has intrinsic dimension of three. The results presented below show that the correlation dimension algorithm coded gives approximately the correct intrinsic dimension.

# points	Intrinsic Dimension
100	3.194
500	2.827
1000	2.788
2000	2.791
5000	2.799

Table 3: Validation results for Correlation Dimension.

5 Hyperspectral Image Classification

5.1 Classifier

A supervised learning algorithm is an algorithm which utilizes a subset of classified data points, called training points, to classify the complete data set. Classification

decisions are made based upon the specified type of classifier. For this project we have chosen to use a naive bayes quadratic classifier. This type of classifier is based upon the data having strong independence among the individual features and builds decision surface between classes that are quadratic. Once the data has been classified we can use another subset of the data that has been identified, just as the training set, to check how good the classification was. Other than looking at a classification accuracy percentage we can use confusion matrices, which for n classes, are an n×n matrix where the columns and rows correspond to the predicated and actual classes respectively. For perfect classification we expect to see all the entries lie on the main diagonal. This type of analysis indicates which classes are being 'confused' by the classifier. We can also display the classification results, with a proper colormap, to see on a macro scale how the classifier performed.

5.2 Image

The hyperspectral image we are using is called Urban (Figure 6) in the literature and is of Copperas Cove, Texas. The image was acquired using a HYDICE sensor and is 310 × 310 pixels with 210 spectral bands and 3 meter resolution. Using the ENVI software we first removed the bad bands (spectral channels where water in the atmosphere caused no data to be received) and then we removed in-scene atmospheric effects to a lab standard atmosphere via the built-in QUAC algorithm. Using a combination of ground truth and higher spatial lower spectral resolution images we created a set of training pixels for seven classes (grass, walmart roof, road, light roof, tan roof, metal rood, and asphalt parking lot.) which we will use for training and verification. The datafile and training pixels were then saved in a CSV file to be used by MATLAB.

5.3 Experimental Setup

To get an initial idea of how many dimensions to use we ran our implementation of the correlation dimension estimator as well as the MLE and PCA estimators from the Dimensionality Reduction Toolbox on the hyperspectral image and came up with a range of 3-6 dimensions. Due to the fact that intrinsic dimension estimation is an imprecise science we have chosen to look at projecting the image from 1 to 50 dimensions. We were not able to implement or optional k-nearest neighbor parameter selector so by trial and error we found that $k = 12$ was optimal for ISOMAP and

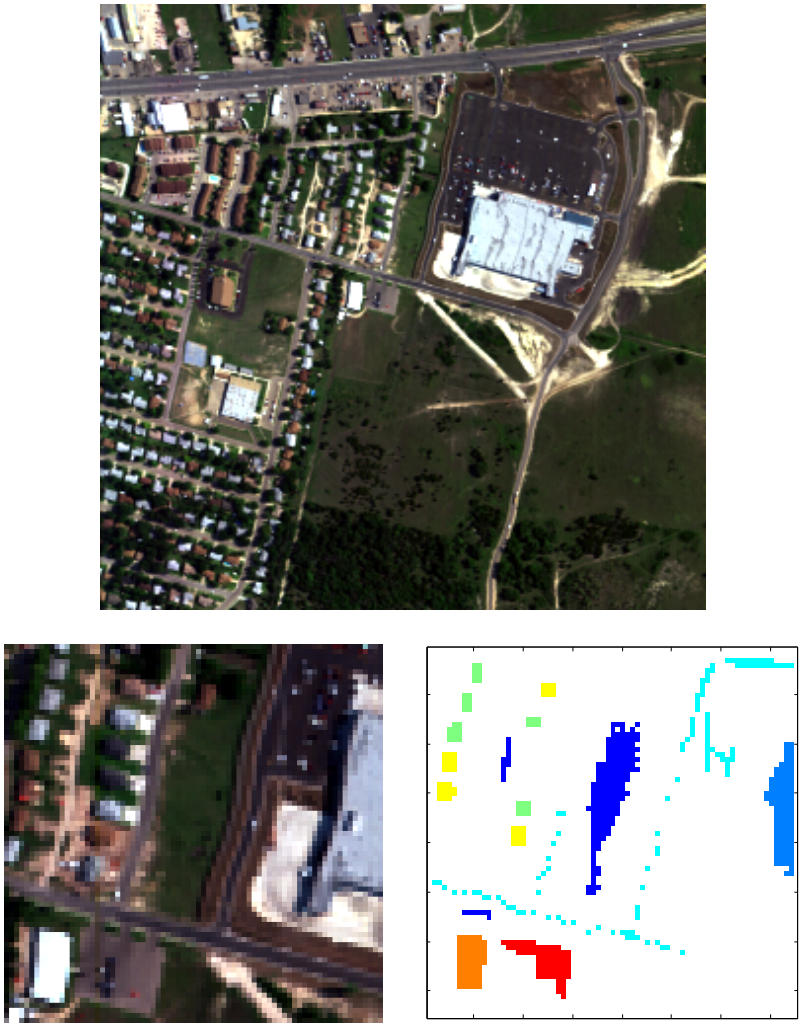

Figure 6: Top is the Urban image with approximate red, blue and green bands selected. Bottom left is the sub-image we are analyzing. Bottom right shows the training pixels we are using.

$k = 12$ was optimal for LLE. Using the MATLAB classifier we trained on 10% of the training data and used the other 90% for classification verification. Averaging over 10 random selections of the training data we calculated the percent accuracy of the classification for the original scene, LLE, ISOMAP, LLLE, and LISOMAP. It should be noted, that because the LLE and ISOMAP experiments were conducted in series that classification accuracy for the original data cube changes because of the random selection of the training and validation sets. This does not invalidate

the results because we are using the original data cube classification percentage as a benchmark. Also note that due to the random nature of the landmark selection(even with max-min as it has a random seed) it is not guaranteed that the results will improve as the percent landmarks increases.

5.4 ISOMAP Results

In Figure 7 we can see the various classification results for the ISOMAP algorithm. Here we can see that across the board the algorithms improved classification results from around 84% using a full dimensional hypercube to around 87% for the uniform random landmarks and the max-min landmarks and 88% for the grid landmarks. We can see the power of using landmarks in the second column of 7 as with only 1% of the pixels used for landmarks the classification results are very close to the full data set being used for ISOMAP and are an improvement over using the original datacube. The fact that it has been shown that using random landmarks over the more costly max-min produced landmarks gives equal and sometimes better results allows us to reduce the complexity of the algorithms further.

In Figure 8 we can see the classification results, with an applied colormap, for the original data cube and the ISOMAP reduced dimensional data with 5 dimensions. We can see that the ISOMAP embedding leads to better classification of the grass field and the lawns around the houses as well as the roof of the building in the lower left. We also can see some improvement in the classification of the roads.

In Table 4 we can see the confusions associated with the classification images in Figure 8. The confusion matrices supports the claims inferred by the images and shows that while it improves classification of the grass, road, and parking lot; it does not improve the classification of the roofs (these remain approximately the same).

5.5 LLE Results

Like we saw with ISOMAP, in Figure 10 the LLE algorithms showed improvement over the original data cube classification results and LLLE with just 1% landmarks was comparable to LLE.

In Figure 9 we can see the classification results, with an applied colormap, for the

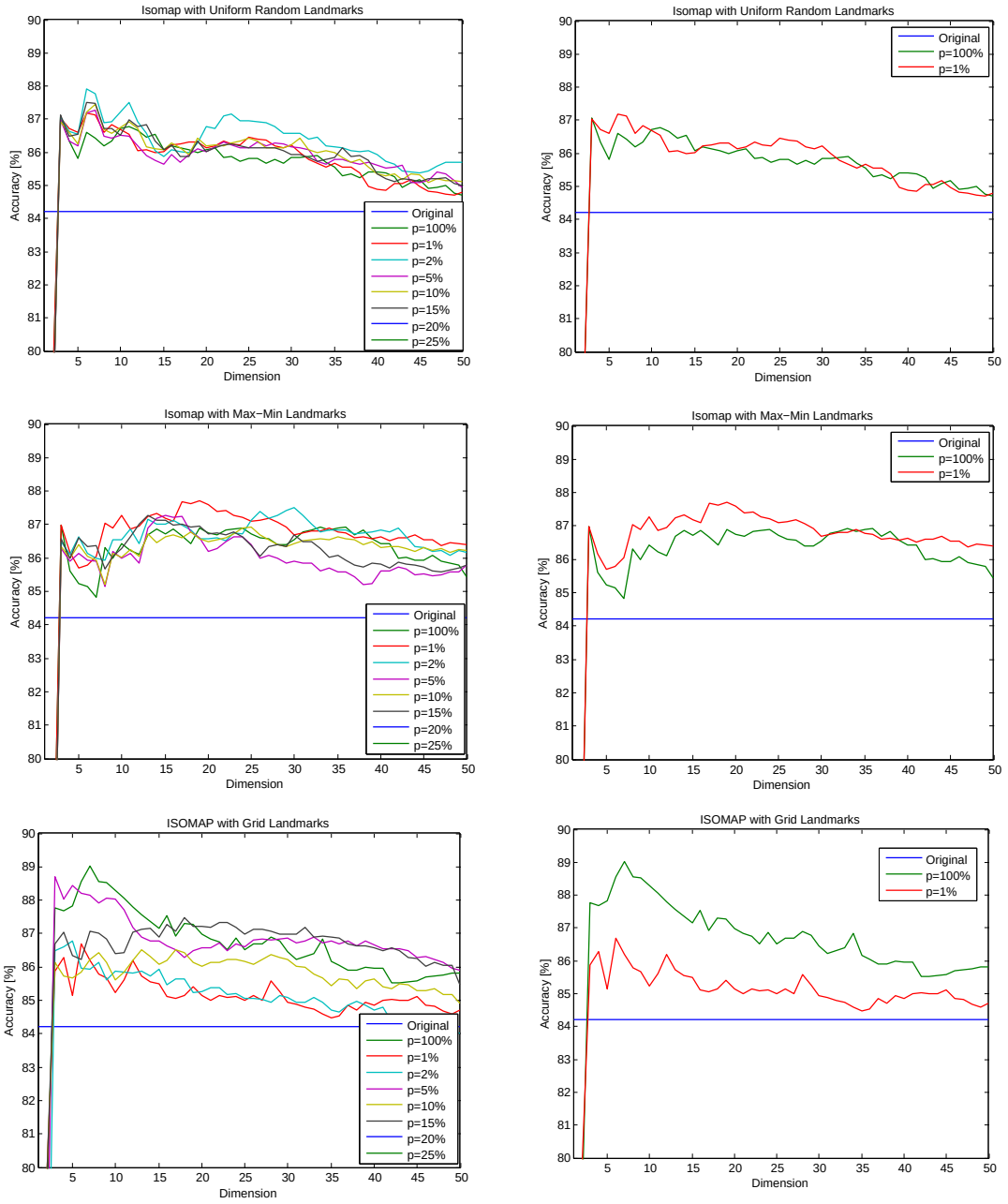

Figure 7: Results for ISOMAP classification. Top to bottom is random, max-min, and grid landmarks. Left is full results and right is ISOMAP vs LISOMAP with 1% landmarks. Horizontal line represents original data classification.

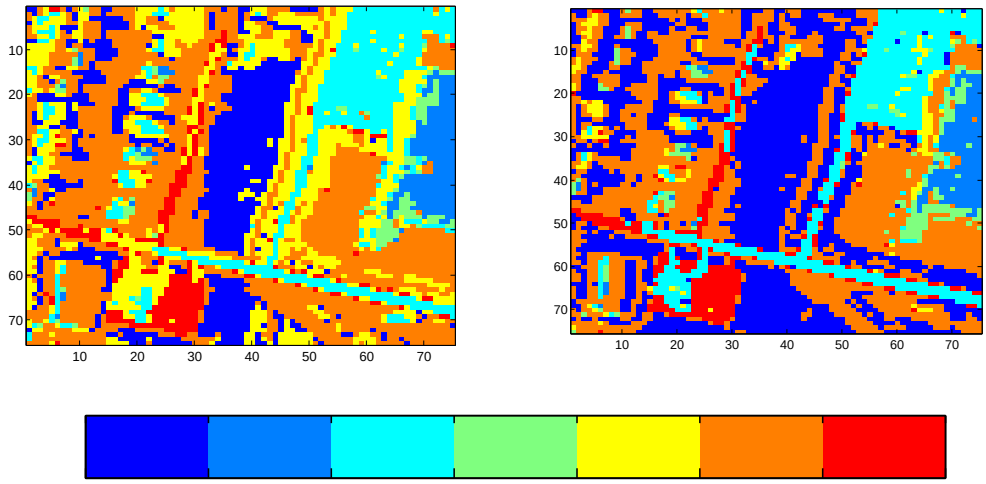

Figure 8: Left is classification using raw data cube. Right is classification using ISOMAP with 5 dimensions. The colors from left to right, represented in the colorbar, are grass, walmart roof, road, light roof, tan roof, metal rood, and asphalt parking lot.

Original 162 Bands

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 with 5 dimensions

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

Table 4: Confusion matrices for Original and ISOMAP reduced. Rows from top to bottom and columns from left to right are grass, walmart roof, road, light roof, tan roof, metal rood, and asphalt parking lot.

original data cube and the LLLE(1%) reduced dimensional data with 22 dimensions. We can see that the LLLE embedding leads to better classification of the grass field and the lawns around the houses. We also can see some improvement in the differentiation between the classification of the roads and the asphalt parking lots..

Original 162 Bands

161	0	0	0	0	0	0
0	85	0	0	0	0	0
3	0	32	9	12	1	4
0	4	1	27	10	5	0
0	0	47	2	19	0	0
0	3	2	0	0	48	0
0	0	25	0	2	0	60

LLLE(1%) with 22 dimensions

164	0	0	2	0	0	0
0	89	0	0	0	0	0
0	2	98	8	0	3	1
0	1	2	19	0	0	0
0	0	1	9	43	3	1
0	0	0	0	0	48	0
0	0	6	0	0	0	62

Table 5: Confusion matrices for Original and LLLE reduced.Rows from top to bottom and columns from left to right are grass, walmart roof, road, light roof, tan roof, metal rood, and asphalt parking lot.

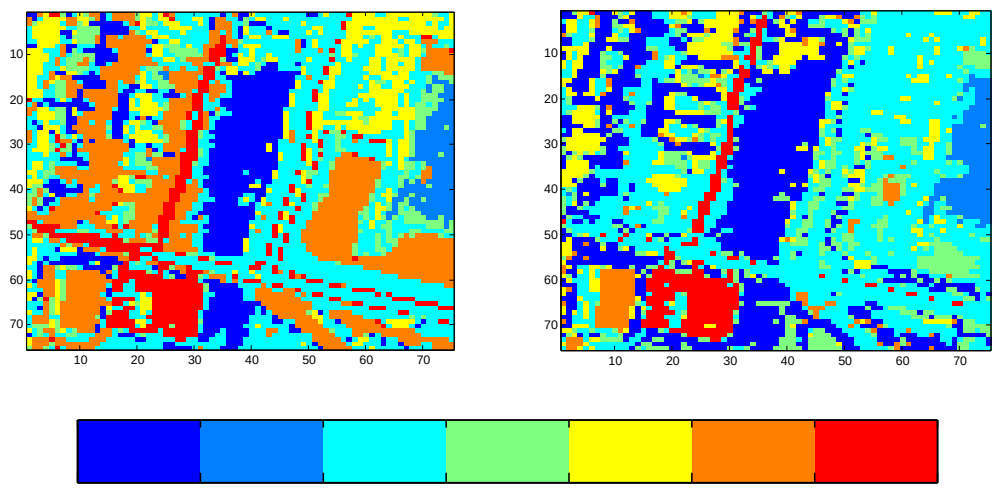

Figure 9: Left is classification using raw data cube. Right is classification using LLE(1%) with 22 dimensions. The colors from left to right, represented in the colorbar, are grass, walmart roof, road, light roof, tan roof, metal rood, and asphalt parking lot.

In Table 5 we can see the confusions associated with the classification images in Figure 9. LLLE can be seen to improve the differentiation between the 3 types of roofs here more clearly then in the images. The confusion matrix supports the claims inferred from the images as well.

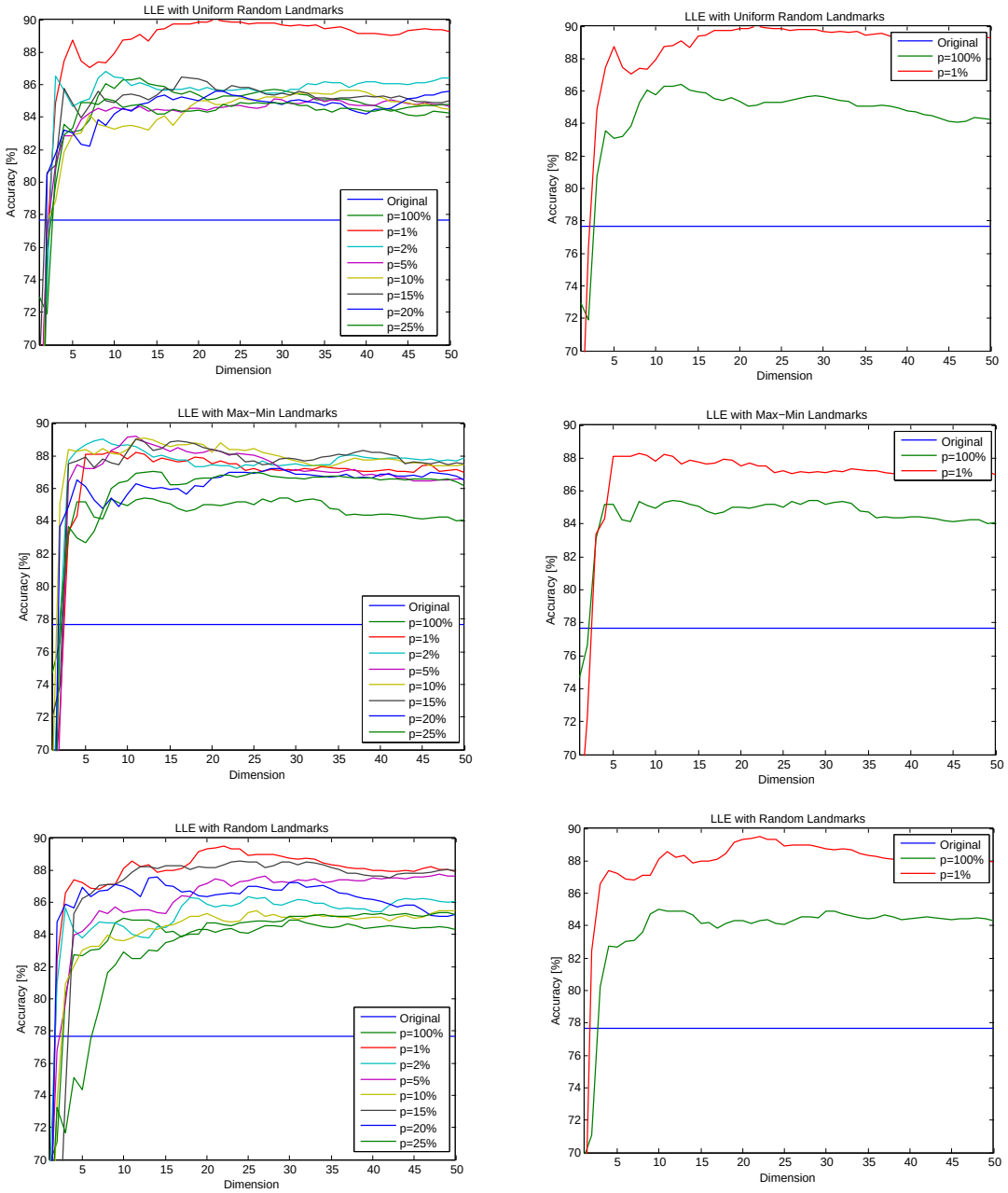

Figure 10: Results for LLE classification. Top to bottom is random, max-min, and grid landmarks. Left is full results and right is LLE vs LLLE with 1% landmarks. Horizontal line represents original data classification.

6 Future Work

To extend this work we would like to add parameter estimators for number of nearest neighbors and the percentage of landmarks similar to the correlation intrinsic dimension estimator. We would also like to add other non-linear dimension reduction algorithms like laplacian eigenmaps or local tangent space alignment. We are also considering other distance metrics like the vector angle which might show more promising results with images.

For computational improvement we would like to explore the use of GPU computing as it is especially adapt at working with images. Also we would like to switch to MKL so as to allow for the parallel computation of eigenvalues and eigenvectors.

7 Code Delivered

• lle.cpp - Performs Local Linear Embedding algorithm given an intrinsic dimensionality, a number of nearest neighbors, and a data file
• isomap.cpp - Performs ISOMAP algorithm given an intrinsic dimensionality, a number of nearest neighbors, and a data file
• llle.cpp - Performs Local Linear Embedding algorithm with landmarks given an intrinsic dimensionality, a number of nearest neighbors, a percentage of landmarks points, and a data file
• lisomap.cpp - Performs ISOMAP algorithm with landmarks given an intrinsic dimensionality, a number of nearest neighbors, a percentage of landmark points and a data file
• corrdim.cpp - Performs the Correlation Dimension intrinsic dimension estimator given a data file
• write.m - Creates the data sets used
• read.m - Reads data sets and embeddings into Matlab
• process_results.m - Automates validation testing and makes graphs
• s####.in, b####.in, t####.in, h####.in - Swiss Roll, Broken Swiss Roll, Twin Peaks, and Helix data sets
• urban.in, urbansm.in - Urban and Urban subset

References

[1] Bachmann, Ainsworth, and Fusina, *Exploting Manifold Geometry in Hyperspectral Imagery*, IEEE Transactions of Geoscience and Remote Sensing, 43 (2005), pp. 441–454.

[2] L. Bernstein, S. Adler-Golden, R. Sundberg, R. Levine, T. Perkins, A. Berk, A. Ratkowski, G. Felde, and M. Hoke, *A New Method for Atmospheric Correction and Aerosol Optical Property Retrieval for VIS-SWIR Multi- and Hyperspectral Imaging Sensors: QUAC (QUick Atmospheric Correction)*, 2006.

[3] M. Fredman and R. Tarjan, *Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms*, Journal of the ACM, 34 (1987).

[4] P. Grassberger and I. Procaccia, *Measuring the strangeness of strange attractors*, Physica, (1983), pp. 189–208.

[5] G. G. J. Bang-Jensen, *Digraphs: Theory, Algorithms and Applications*, Springer, 2010.

[6] J. Lee and M. Verleysen, *Nonlinear Dimensionality Reducation*, Springer, 2007.

[7] C. Lelong, P. Pinet, and H. Poilve, *Hyperspectral Imaging and Stress Mapping in Agriculture: A Case Study on Wheat in Beauce (France)*, Remote Sensing of Environment, 66 (1998), pp. 179 – 191.

[8] Maatan, Postma, and Herik, *Dimensionality Reduction: A Comparative Review*, (2008).

[9] D. Manolakis, D. Marden, and G. Shaw, *Hyperspectral Image Processing for Automatic Target Detection Applications*, Lincoln Laboratory Journal, 14 (2003), pp. 79–116.

[10] R. Mayer, R. Priest, C. Steliman, G. Hazel, and A. Schaum, *Detection of Camouaged Targets in Cluttered Backgrounds Using Fusion of Near Simultaneous Spectral and Polarimetric Imaging*, tech. rep., Naval Research Lab, 2000.

[11] S. Roweis and L. Saul, *An Introduction of Local Linear Embedding.* Unpublished manuscript, 2001.

[12] F. Sabins, *Remote Sensing for Mineral Exploration*, Ore Geology Reviews, 14 (1999), pp. 157 – 183.

[13] L. Saul and S. Roweis, *Reduction by Locally Linear Embedding*, Science, 209 (2000), pp. 2323–2327.

[14] J. Schott, *Remote Sensing: The Image Chain Approach*, Oxford University Press, New York, 1997.

[15] S. Shaw, *LAPACK/ARPACK*. `http://www-heller.harvard.edu/people/shaw/programs/lapack.html`.

[16] M. Smith, S. Ollinger, M. Martin, J. Aber, and R. H. C. Goodale, *Direct Estimation of Aboveground Forest Productivity Through Hyperspectral Remote Sensing of Canopy Nitrogren*, Ecological Applications, 12 (2002), pp. 1286–1302.

[17] Tenenbaum, Silva, and Langford, *A Global Geometric Framework for Nonlinear Dimensionality Reduction*, Science, 209 (2000), pp. 2319–2323.