End of Semester Presentation

December 11, 2012

AMSC 663: Advanced Scientific Computing

Nonlinear Dimensionality Reduction Applied to the Classification of Images

Student:
Chae A. Clark (cclark18 [at] math.umd.edu)

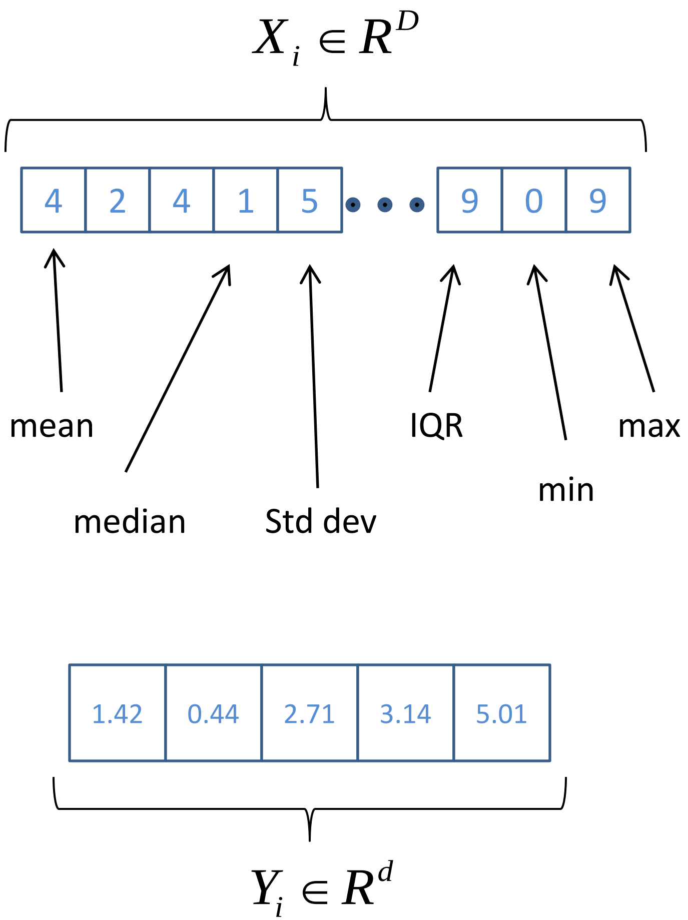
Advisor:
Dr. Kasso A. Okoudjou (kasso [at] math.umd.edu)
Norbert Wiener Center for Harmonic Analysis
Department of Mathematics
University of Maryland

Abstract:

For this project I plan to implement a dimension reduction algorithm entitled "Locally Linear Embeddings" in the programming language MatLab. For a group of images, the dimension reduction algorithm is applied, and the results are used to compare classification accuracies.

Review I

Dimension Reduction

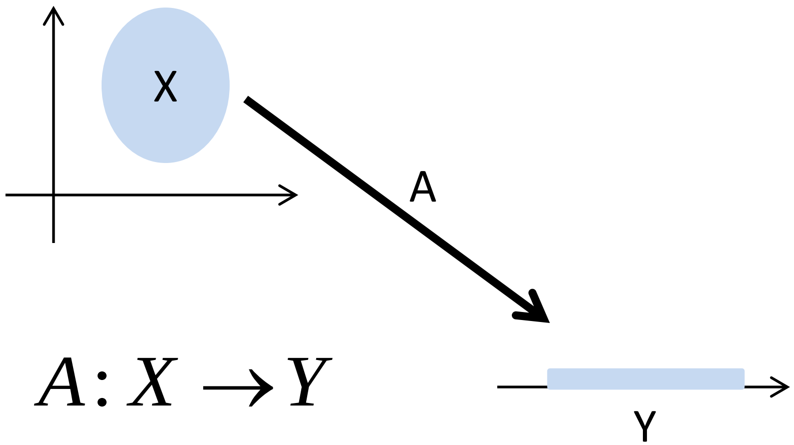


$$d << D$$

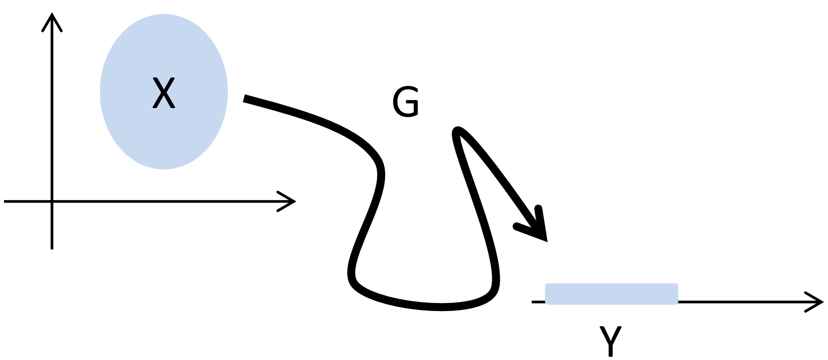
- We start with multiple high-dimensional points (maybe a set of images)
- We map that image to a D dimensional vector
- Lots of elements means the processing of this data is more computationally intensive
- Usually lots of redundant data, or lots of correlation in the elements

- We want a vector of a reduced size that retains important characteristics of the data
- We also want the new vector's elements to be un-correlated

Dimension Reduction



$$A: X \rightarrow Y$$



$$G: X \rightarrow Y$$

There are a number of techniques to perform this operation under the field Dimension Reduction

Linear Reduction Methods

- Search for a matrix A (or matrix operation) that maps your high-dimensional data into a lower dimensional space
- Preserves key characteristics of data

Nonlinear Reduction Methods

- Use a nonlinear mapping that reduces your dimension
- Preserves key characteristics of data

LLE Overview

Locally Linear Embeddings (LLE)

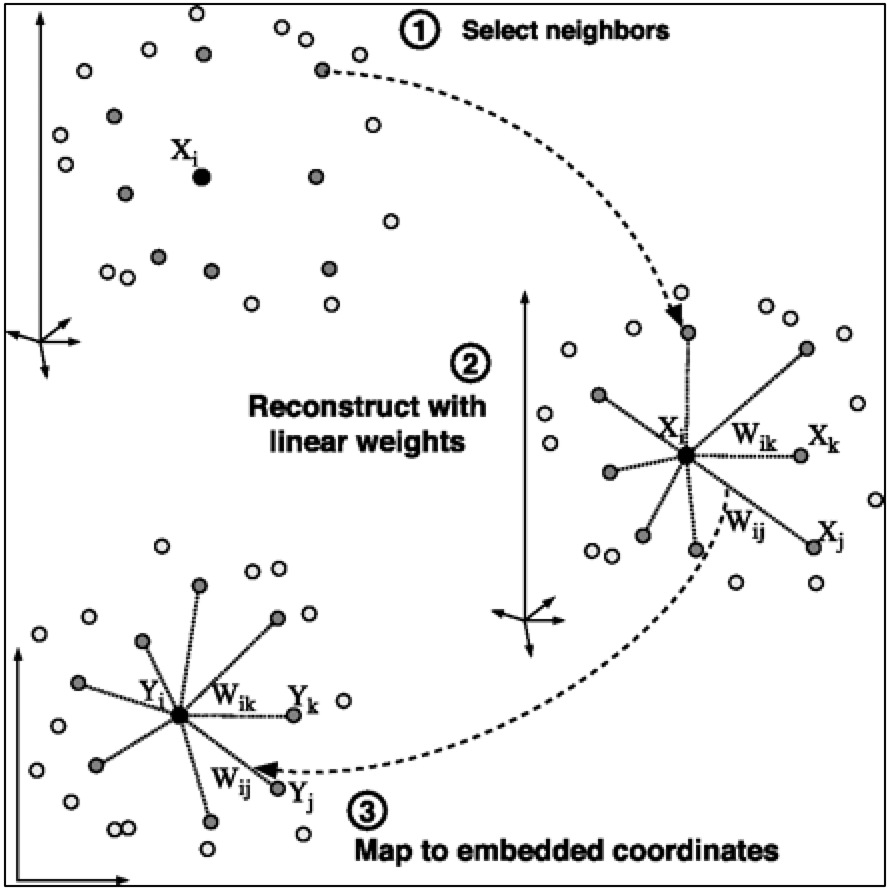


Figure 1: Obtained from LLE website [1]

- Nonlinear dimension reduction method
- Developed by Dr. Sam Roweis and Dr. Lawrence Saul
- Takes a high-dimensional point X and maps it a lower dimensional point Y
- Preserves local geometry (local distances between points)
- This is done by solving a series (two) constrained optimization problems

[1] http://www.cs.nyu.edu/~roweis/lle/algorithm.html

LLE Overview

Optimization Problem

$$\arg\min : E(W) = \sum_i \left\| X_i - \sum_j W_{ij} X_j \right\|^2$$

$$\sum_j W_{ij} = 1$$

$$W_{ij} = 0$$

For points X_j that are not neighbors of X_i

Optimization Problem

$$\arg\min : e(Y) = \sum_i \left\| Y_i - \sum_j W_{ij} Y_j \right\|^2$$

$$\sum_j Y_i = 0$$

$$\frac{1}{N} \sum_i Y_i^T \cdot Y_i = I$$

- Find the nearest neighbors of each point in our set
- Try to find a linear (almost convex) combination of the nearest neighbors that best represents the point
- Use the found weights as the contribution of each neighbor point

- Find the reduced dimension points that retain the weight spacing determined in Step 1
- In essence, we are preserving pair wise distances between neighbors
- Try to find a linear (almost convex) combination of the nearest neighbors that best represents the point
- Use the found weights as the contribution of each neighbor point

Implementation

Software

Algorithms implemented in the programming language MatLab

This is due to:

- Flexibility in syntax
- Ubiquitous use by the scientific community
- Wide availability of support

Hardware

Currently using a personal computer for development, validation, and testing

If this becomes computationally infeasible, I will also use the computers in the Norbert Weiner Center for testing

Nearest Neighbor Search

Process & Algorithm

- There are a number of ways to find the K-nearest neighbors

- In this project, 3 different methods are implemented, one through binary programming, another through a heuristic method, and the one presented below

Algorithm: Nearest Neighbors through Full Enumeration

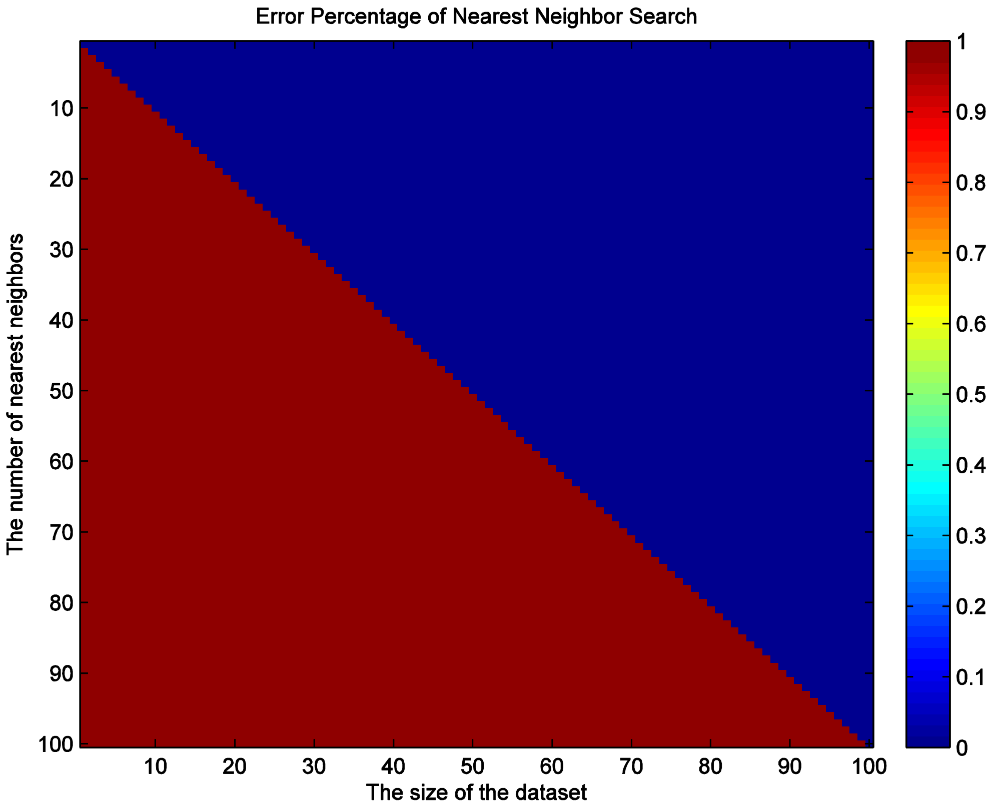
- **for each** $X_i \in X$ (for each data point in our data set)
- compute the pair-wise distance between X_i and every point in the dataset
- arrange these distances as a sorted array, keeping track of the indices of the K+1 smallest distances
- remove the index of the 0 distance
- the remaining indices are the K nearest neighbors
- **end for**

Validation

- Using MatLab's built-in nearest neighbor search (knnsearch.m), we can validate our search code.

- The graph below shows the difference in nearest neighbors found between our implementation and MatLab's

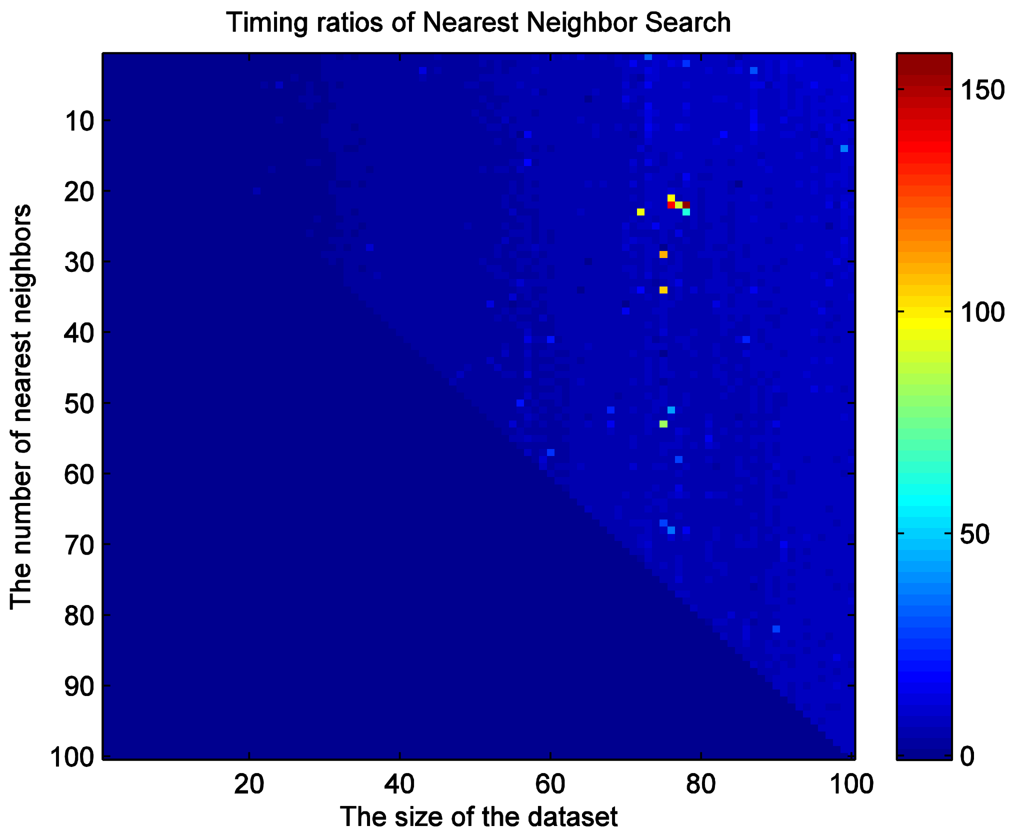
- This test was performed on random matrices
- Filename: knn_validation.m



- This graph is uninteresting, but it validates the nearest neighbor code

Timing Results

- Here are some timing results as well
- They are the ratio of the time required to process the same dataset
- Here the time for my algorithm is divided by the timing for MatLab's implementation



Weights Construction

Process & Algorithm

• Given the K-nearest neighbors for each data point X_i in the dataset X_i
(Denote these $\{Z_{ij}\}_{j=1}^{K}$)

• We want to find the weights that reduce reconstruction error for each data point

•2 different methods implemented in the project, one through matrix inversion and the other presented here

Algorithm: Weight Construction II

- **for each** $X_i \in X$ (for each data point in our data set)
- center the points about X_i , $\tilde{Z}_{ij} = Z_{ij} - X_i$
- create the matrix $Z_i = [\tilde{Z}_{i1} \cdots \tilde{Z}_{iK}]$
- for the Gram matrix $G_i = Z_i^T \cdot Z_i$
- solve the system:

$$G_i \cdot \dot{w}_i = 0.5 \cdot \vec{1}_K$$

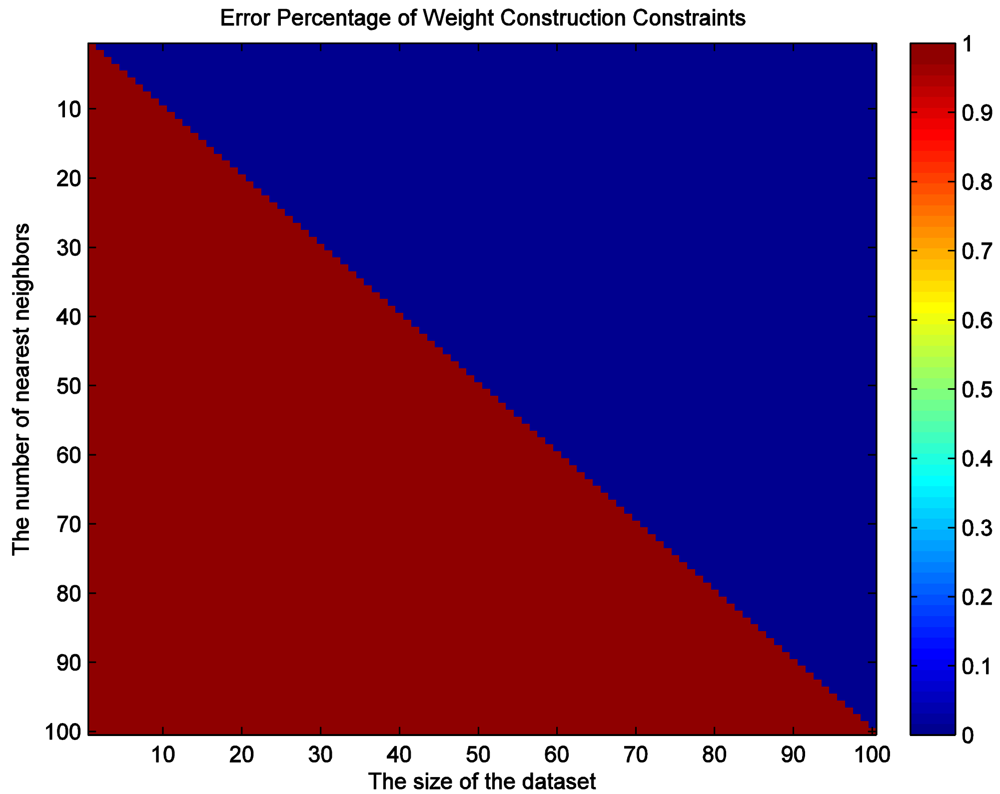
- compute the Lagrange multiplier to enforce the constraints $\lambda_i = 1 / \sum_{j=1}^{K} \dot{w}_j$
- compute the reconstruction weights $w_i = \lambda_i \cdot \dot{w}_i$
- **end for**

Validation

- Here, validation of the algorithm will be viewed as the correctness of its constraint requirements

- Optimality of the method will be present in the appendix of the report

- Using random matrices, validation results are presented
- Filename: weights_validation.m



- This colormap is as interesting as the last, but it validates the implementation

Embedding Construction

Process

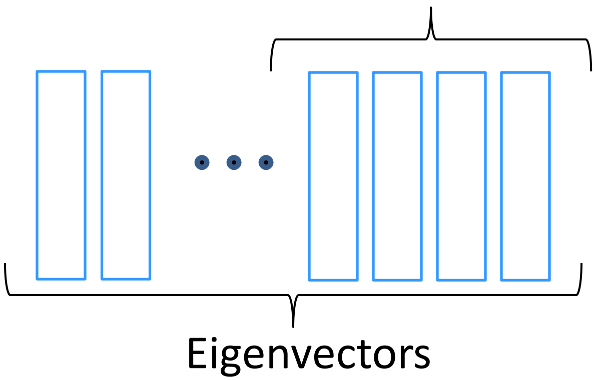
Minimizing e(Y)

$$\arg\min : e(Y) = \sum_i \left\| Y_i - \sum_j W_{ij} Y_j \right\|^2$$

$$\sum_j Y_j = 0$$

$$\frac{1}{N} \sum_i Y_i^T \cdot Y_i = I$$

$$Y \in R^{N \times d}$$



- It has been proven that minimizing this function is equivalent to performing an eigen-decomposition [1]
- We find the eigenvalues and eigenvectors of

  $$(I - W)^T (I - W)$$

- Taking the eigenvectors that correspond to the smallest eigenvalues, we now have Y
- The rows of the eigenvector matrix are the reduced dimension dataset Y

[1] Sam Roweis and Lawrence Saul, Nonlinear Dimensionality Reduction by Locally Linear Embeddings, Science v.290 no.5500, Dec.22, 2000. pp.2323--2326.

Algorithm & Validation

•Here, built-in MatLab functions are used exclusively, so a validation step would be to check the function against itself (so it's not included here)

Why a 0-Eigenvalue?

$$W \cdot \vec{1} = \vec{1}$$

$$(Id - W)\vec{1} = 0 \cdot \vec{1}$$

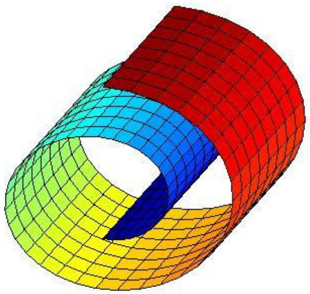
$$(Id - W)^T (Id - W)\vec{1} = 0 \cdot \vec{1}$$

Algorithm: Nearest Neighbors through Full Enumeration

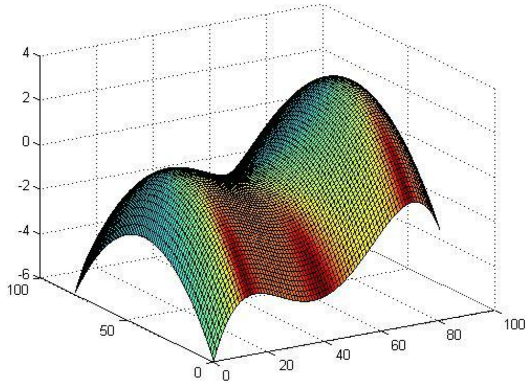
- form the matrix $\tilde{W} = (I - W)^T (I - W)$
- compute the eigenvalues and eigenvectors of $\tilde{W}$
- discard the eigenvector corresponding to the eigenvalue of 0
- let Q denote the matrix of the smallest d eigenvectors ($Q = [q_1 \cdots q_d]$)
- Return Q as the lower-dimensional embedding

Algorithm Validation

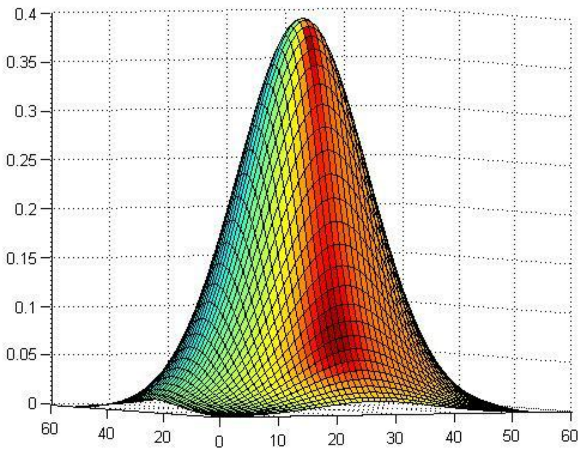
Validation Surfaces



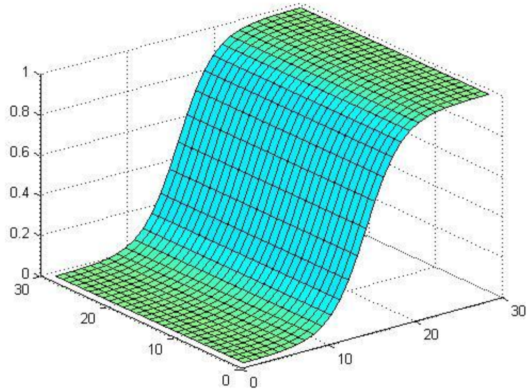
Swiss Roll Mapping



Twin Peaks Function



Gaussian Function



Logistic Function

Author’s Implementation

•On the co-author’s site there is a full implementation of the LLE algorithm

•It is free to use and open to the public

•Using the random data sets and the surfaces in the previous slide, we can compare the output to ensure a correct implementation of our LLE algorithm

Available at: http://www.cs.nyu.edu/~roweis/lle/

Presence of Nearest Neighbors

- In this test, the two implementations are compared for the presence of nearest neighbors in the embedding

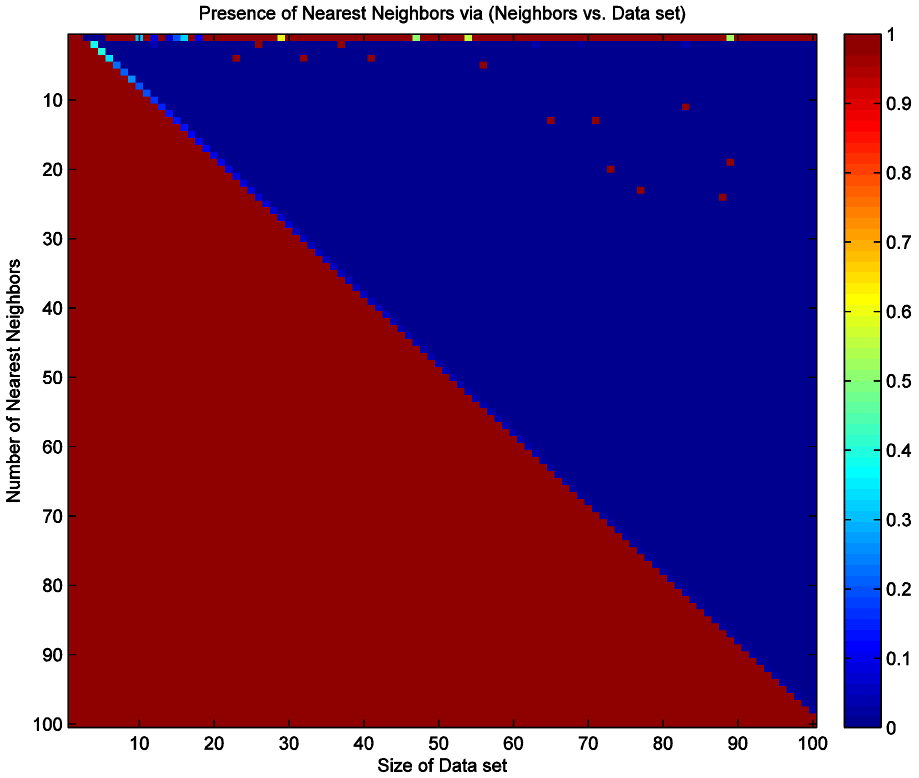
- Order of the nearest neighbors doesn't matter

- This is done on random datasets

- The values are percentage errors in correspondence

Neighbors (Same)

Our Algorithm	Authors Algorithm
3	2
2	3

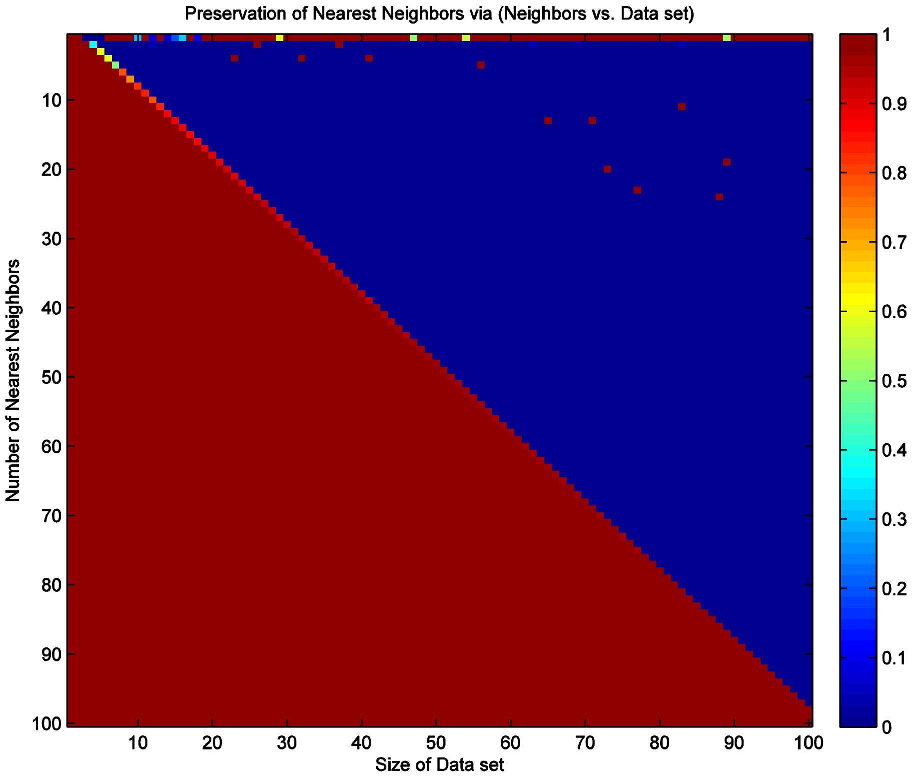


Preservation of Nearest Neighbors

- In this test, the same process is undertaken, with the exception that the order of the nearest neighbors is compared

- This is done on random datasets

- The values are percentage errors in correspondence



Neighbors (Not the same)

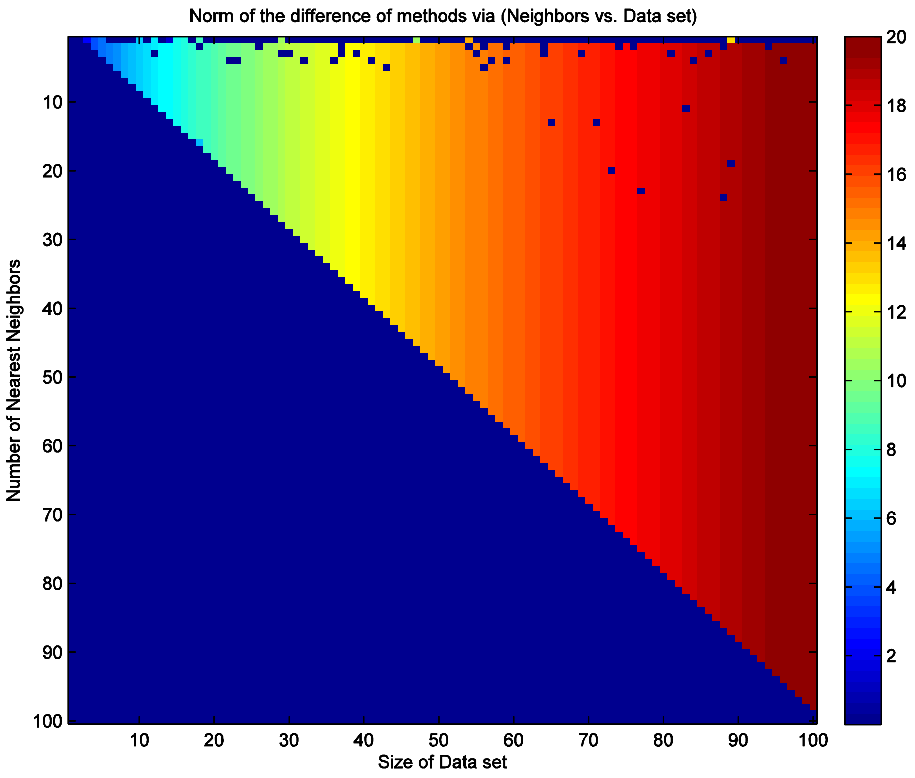
Our Algorithm	Authors Algorithm
3	2
2	3

Norm of the Difference

- Here, the difference between the resulting embeddings is explored

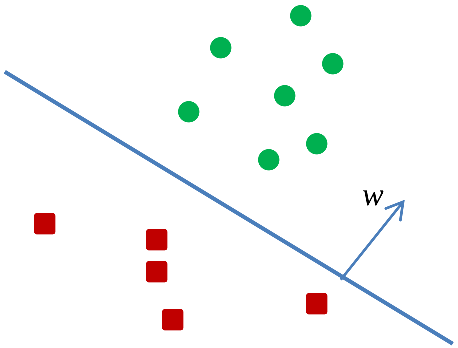
- There is more activity here, but it seems to vary with the size of the dataset and not the number of neighbors

- This would seem to imply that the embeddings differ by some scalar with the dataset



Review II

Testing



$$\arg\min : \frac{1}{2}\|w\|^2$$

Constraints

$$y_i(w^T x_i + w_0) \geq 1$$

Our specific application is in image classification

We want to find a hyper plane that separates different images

This can be done using Support Vector Machines, which finds the optimal hyper plane that separates the data

w is the vector normal to the hyper plane and w_0 is the offset from the origin

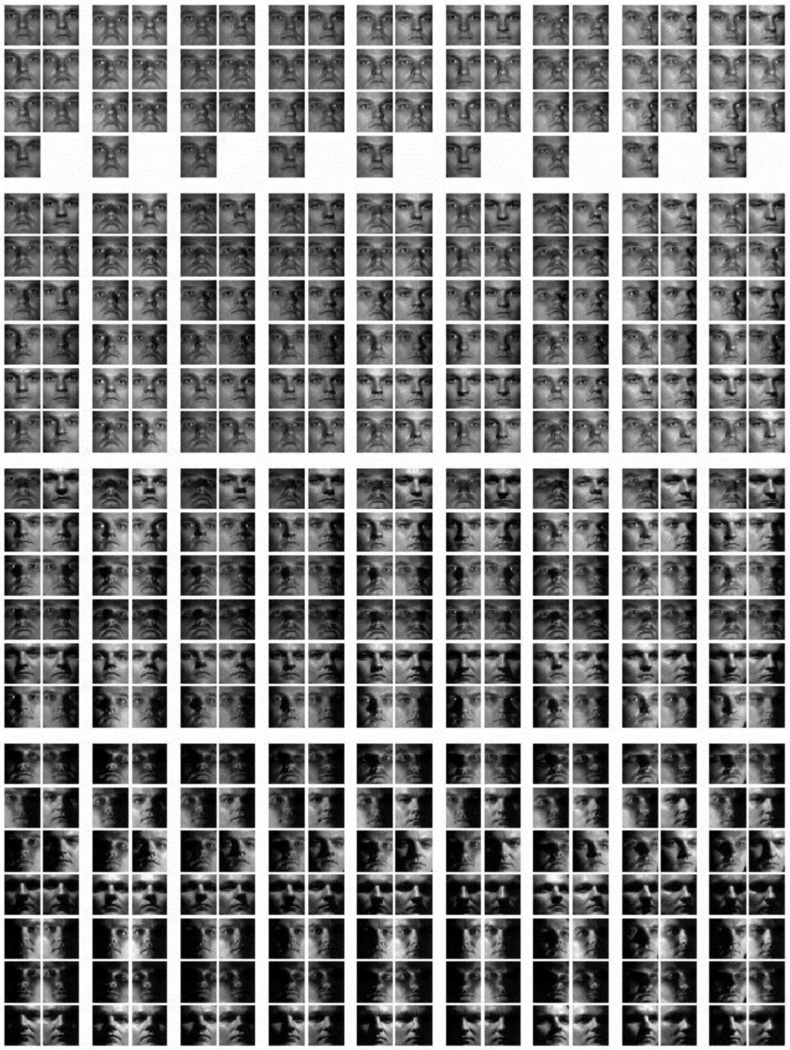
We can find this by solving a constrained optimization problem, or a similar Lagrangian unconstrained problem

Here, x_i are our data points and $y_i \in \{-1,1\}$ are the class labels (which group an image belongs to)

Databases

The Yale Face Database B [1]

- Over 5000 face images
- 10 different subjects (people)
- Over 500 different positions and illuminations
- Using the original dataset (images) and the reduced dataset (LLE), I plan to compare the classification accuracy of the SVM on these sets



[1] http://www.csie.ntu.edu.tw/ cjlin/libsvm/faq.html#f203

Updated Project Schedule

September 2012 - November 2012

- ~~Plan and implement the LLE algorithm in MatLab, efficiently handling storage and memory management issues.~~
- ~~Perform unit tests to correct any bugs present in code.~~
- ~~Validate code on standard topological structures (Swiss Roll, etc.).~~
- ~~Compare results of algorithm output to the LLE algorithm made available by the co-author~~
- ~~Test the LLE algorithm on a randomly distributed dataset~~

November 2012 - December 2012

- ~~Make any necessary preprocessing changes to the image database used.~~
- ~~Prepare the mid-year (end of semester) report and presentation.~~
- Deliver mid-year report and deliverables.

January 2013

- Implement a pre-developed SVM package for MatLab.
- Test classification accuracy of SVM on dimension-reduced dataset.
- Assess effectiveness.

February 2013 - April 2013

- Implement SVM in MatLab (time permitting).
- Implement LLE extensions.
- Compare results of original LLE implementation to extended versions.

April 2013 - May 2013

- Prepare final presentation, report, and deliverables.
- Make any last minute adjustments to code that are required.
- Package and deliver deliverables.

Deliverables

- Implemented LLE MatLab code
- Testing scripts
- Documentation regarding code use and available options
- Final report of algorithm design, testing, and results
- Final presentation

References

[1] Sam Roweis and Lawrence Saul, Nonlinear Dimensionality Reduction by Locally Linear Embeddings, Science v.290 no.5500, Dec.22, 2000. pp.2323--2326.

[2] Sergios Theodoridis and Konstantinos Koutroumbas, Pattern Recognition, Fourth Edition, Academic Press 2008.

[3] Olga Kouropteva and Oleg Okun and Matti Pietikäinen, Selection of the Optimal Parameter Value for the Locally Linear Embedding Algorithm, 1 st International Conference on Fuzzy Systems, 2002, 359--363.

[4] O. Kouropteva and M. Pietikainen. Incremental locally linear embedding. Pattern Recognition, 38:1764–1767, 2005.

[5] Boschetti and Fabio, Dimensionality Reduction and Visualization of Geoscientific Images via Locally Linear Embedding, Comput. Geosci., July, 2005, 31,6, 689--697.

[6] Hong Chang and Dit-yan Yeung, Robust Locally Linear Embedding, 2005.

[7] Chang, Chih-Chung and Lin, Chih-Jen, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology, 2, 3, 2011, 27:1--27:27.

[8] Georghiades, A.S. and Belhumeur, P.N. and Kriegman, D.J., From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose, IEEE Trans. Pattern Anal. Mach. Intelligence, 2001, 23, 6, 643-660.

[9] Zhang Z, Wang J (2007) MLLE: Modified locally linear embedding using multiple weights. Advances in Neural Information Processing Systems (NIPS) 19, eds Scho lkopf B, Platt J, Hofmann T (MIT Press, Cambridge, MA), pp 1593–1600.

QUESTIONS