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.
0. Introduction

Dataset – collection of vectors

\[ X_i \in \mathbb{R}^D \]

\[ \begin{bmatrix} 4 & 2 & 4 & 1 & 5 & \cdots & 9 & 0 & 9 \end{bmatrix} \]

\[ X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \]

\[ X \in \mathbb{R}^{N \times D} \]
1. Introduction (cont.)

- 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

\[ X_i \in \mathbb{R}^D \]

\[
\begin{array}{cccccc}
4 & 2 & 4 & 1 & 5 & \ldots & 9 & 0 & 9
\end{array}
\]

\[
\begin{array}{cccccc}
\text{mean} & \text{median} & \text{Std dev} & \text{IQR} & \text{max} & \text{min}
\end{array}
\]

\[
\begin{array}{cccccc}
1.42 & 0.44 & 2.71 & 3.14 & 5.01
\end{array}
\]

\[ Y_i \in \mathbb{R}^d \]

\[
\begin{array}{cccccc}
\end{array}
\]

\[
\begin{array}{cccccc}
\end{array}
\]

\[ d \ll D \]
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
3. Approach

Locally Linear Embeddings (LLE)

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

Figure 1: Obtained from LLE website [1]

4. Approach (cont.)

Step 1

Optimization Problem

\[
\arg \min_e E(W) = \sum_{i=1}^{N} \left| X_i - \sum_{j=1}^{N} W_{ij} X_j \right|^2
\]

• Find the k 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

Constraints

\[
\sum_{j=1}^{N} W_{ij} = 1
\]

• First constraint makes the embedding invariant to data scaling and translations

\[
W_{ij} = 0
\]

• Second constraint ensures that the weight of non-neighbors is zero
5. Approach (cont.)

Step 2

Optimization Problem

\[
\arg \min : e(Y) = \sum_{i=1}^{N} \left\| Y_i - \sum_{j=1}^{N} W_{ij} Y_j \right\|^2
\]

• Find the reduced dimension points that retain the weight spacing determined in Step 1

• In essence, we are preserving pair wise distances between our k neighbors

• Use the found weights as the contribution of each neighbor point

Constraints

\[
\sum_{i=1}^{N} Y_i = 0
\]

• First constraint centers the points around the origin

\[
\frac{1}{N} \sum_{i=1}^{N} Y_i^T \cdot Y_i = I
\]

• Second constraint ensures the outer products sum to the identity matrix
6. Implementation

Minimizing $E(W)$

$$\text{arg min : } E(W) = \sum_{i=1}^{N} \left\| X_i - \sum_{j=1}^{N} W_{ij} X_j \right\|^2$$

$$\sum_{j=1}^{N} W_{ij} = 1$$

$$W_{ij} = 0$$

- We make a set $S_i = \{X_j\}$ which contains the closest k neighbors $X_j$ of point $X_i$
- We then compute the neighborhood correlation matrix $C$
- The elements $C_{jk}$ are the pairwise inner products of the nearest neighbors
- From this correlation matrix, we compute the inverse $C^{-1}$
- This is done for each point in the dataset

Now we can construct our weights $W_{ij}$ with the following formula

$$W_{ij} = \sum_{k=1}^{N} C_{jk}^{-1} [(X_i \cdot X_j) + \lambda]$$

Here, $\lambda$ is the Lagrange multiplier as specified in the paper by Saul and Roweis [1]

Minimizing $e(Y)$

$$\text{arg min} : e(Y) = \sum_{i=1}^{N} \left\| Y_i - \sum_{j=1}^{N} W_{ij} Y_j \right\|^2$$

$$\sum_{i=1}^{N} Y_i = 0$$

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

Taking the eigenvectors that correspond to the smallest eigenvalues, we now have $Y$

The rows of the eigenvector matrix are the reduced dimension dataset $Y$

---

Computational Costs

Minimizing $E(W)$ is of complexity 

$O[Nk^3]$ 

$N$ – size of data set (number of vectors)  
$k$ – number of nearest neighbors used

Minimizing $e(Y)$ can be solved with subquadratic complexity due to the Sparsity of the weight vector
Algorithm Extension 1

There are two parameters with LLE, k and d

In a paper by Kouropteva, Okun, and Pietikainen, a method is presented for determining the optimal number of nearest neighbors [1]

Algorithm Extension 2

In another paper by Kouropteva, Okun, and Pietikainen, an incremental LLE implementation is described [2]

The advantage here is that we must solve smaller optimization problems.


10. Implementation (cont.)

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 plan to use personal computer for development and testing

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

Standard Topological Manifolds (Surfaces)

Swiss Roll Mapping

Twin Peaks Function

Gaussian Function

Logistic Function
11. Validation (cont.)

MatLab Dimension Reduction Toolbox

• The Dimension Reduction Toolbox is implemented in MatLab

• It is free to use and open to the public

• It contains a wide range of Dimension reduction methods

• This includes an implementation of LLE

• Using the test functions from the previous slide we can compare the output to ensure a correct implementation of our LLE algorithm

Available at: http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html
13. Testing

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).
14. 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

15. 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 results of the LLE method present in the Dimension Reduction Toolbox.
- Test the LLE algorithm on a dataset from a publicly available database.

**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.

**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 and report.
- Make any last minute adjustments to code that are required.
- Package deliverables.
- Ensure the safe delivery of source code and other project materials.
16. Deliverables

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


