Explicit Non-Linear Dimensionality Reduction

Pierre Visconti

Department of Mathematics, Walla Walla University

03 May 2024

Outline

1. Introduction and Motivation
2. Mathematical Intuition
3. Developing an Algorithm
4. NPPE Algorithm
5. Results and Conclusion

Complexity of Datasets in Machine Learning

Complexity of Datasets in Machine Learning

Image size

Each image: 200x300 pixels = 60,000 pixels per image.

Image ID	**Pixel 0**	**Pixel 1**	**...**	**Pixel (59,999)**
Img_00	120	135	...	90
Img_01	100	110	...	95
Img_02	130	125	...	85
...	...	...	...	...
Img_n-1	115	105	...	100

Time complexity

Number of operations.

- Decision Trees: $O(mn \log(n))$
- Naive Bayes: $O(mn)$

Complexity of Datasets in Machine Learning

Image size

Each image: 200x300 pixels = 60,000 pixels per image.

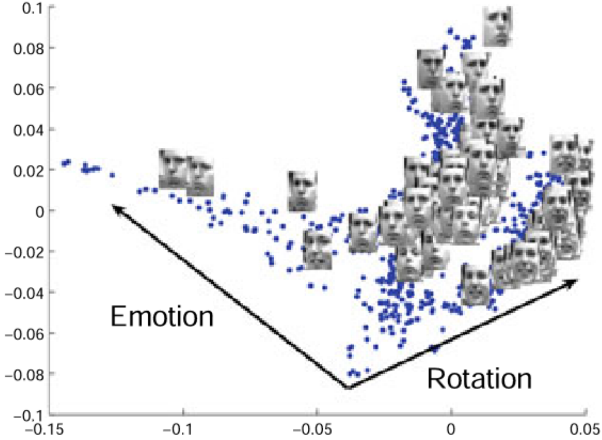
Image ID	**Pixel 0**	**Pixel 1**	**...**	**Pixel (59,999)**
Img_00	120	135	...	90
Img_01	100	110	...	95
Img_02	130	125	...	85
...	...	...	...	...
Img_n-1	115	105	...	100

Time complexity

Number of operations.

- Decision Trees: $O(mn\log(n))$
- Naive Bayes: $O(mn)$

Complexity of Datasets in Machine Learning



Dimensionality Reduction

Dimensionality reduction is the act of reducing high dimensional data to a lower dimension while retaining the intrinsic properties of the data.

Issues with current algorithms

- Linear techniques such as Principal Components Analysis (PCA) cannot handle complex non-linear data.
- Manifold Learning algorithms, can handle non-linear data, however most existing algorithms assume a linear mapping and do not produce an explicit model.
- Goal:

  Explore a manifold learning algorithm that produces an explicit non-linear model.

Dimensionality Reduction

Dimensionality reduction is the act of reducing high dimensional data to a lower dimension while retaining the intrinsic properties of the data.

Issues with current algorithms

- Linear techniques such as Principal Components Analysis (PCA) cannot handle complex non-linear data.
- Manifold Learning algorithms, can handle non-linear data, however most existing algorithms assume a linear mapping and do not produce an explicit model.
- Goal:
  - Explore a manifold learning algorithm that produces an explicit non-linear model.

Dimensionality Reduction

Dimensionality reduction is the act of reducing high dimensional data to a lower dimension while retaining the intrinsic properties of the data.

Issues with current algorithms

- Linear techniques such as Principal Components Analysis (PCA) cannot handle complex non-linear data.
- Manifold Learning algorithms, can handle non-linear data, however most existing algorithms assume a linear mapping and do not produce an explicit model.
- Goal:
  - Explore a manifold learning algorithm that produces an explicit non-linear model.

Dimensionality Reduction

Dimensionality reduction is the act of reducing high dimensional data to a lower dimension while retaining the intrinsic properties of the data.

Issues with current algorithms

- Linear techniques such as Principal Components Analysis (PCA) cannot handle complex non-linear data.
- Manifold Learning algorithms, can handle non-linear data, however most existing algorithms assume a linear mapping and do not produce an explicit model.
- Goal:
  - Explore a manifold learning algorithm that produces an explicit non-linear model.

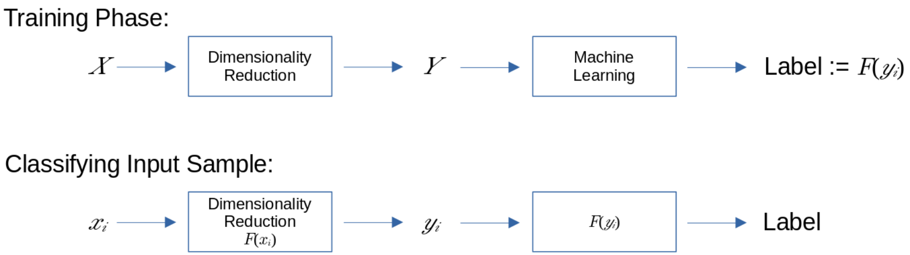
Dimensionality Reduction

Dimensionality reduction is the act of reducing high dimensional data to a lower dimension while retaining the intrinsic properties of the data.

Issues with current algorithms

- Linear techniques such as Principal Components Analysis (PCA) cannot handle complex non-linear data.
- Manifold Learning algorithms, can handle non-linear data, however most existing algorithms assume a linear mapping and do not produce an explicit model.
- Goal:
  - Explore a manifold learning algorithm that produces an explicit non-linear model.

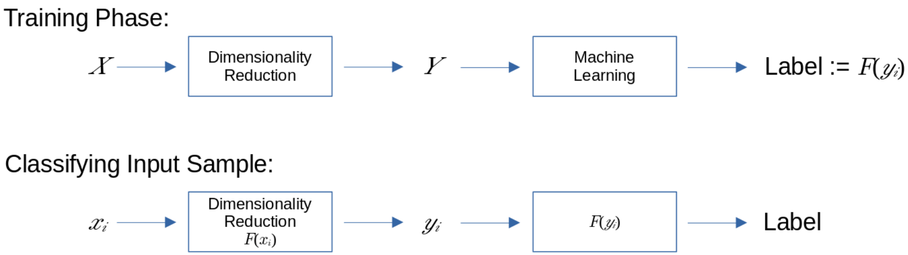
Importance of Producing an Explicit Model



Explicit

- Closed form expression.
- Independent of training samples.

Importance of Producing an Explicit Model



Explicit

- Closed form expression.
- Independent of training samples.

Outline

1. Introduction and Motivation
2. Mathematical Intuition
3. Developing an Algorithm
4. NPPE Algorithm
5. Results and Conclusion

Manifolds and Embeddings

Definition: smooth manifold

M is a smooth manifold of dimension d if it is a topological space where,

- For every $p, q \in M$ there exists disjoint open subsets $U, V \subseteq M$, such that $p \in U$ and $q \in V$.
- There exists a countable basis for the topology of M.
- For every $p \in M$, there exists
  - an open subset $U \subseteq M$ containing p,
  - an open subset $\hat{U} \subseteq \mathbf{R}^d$, and
  - a homeomorphism $\phi : U \rightarrow \hat{U}$

A topological space that that has a differentiable structure and locally resembles Euclidean space near every point. The dimension d is called the intrinsic dimension.

Manifolds and Embeddings

Definition: smooth manifold

M is a smooth manifold of dimension d if it is a topological space where,

- For every $p, q \in M$ there exists disjoint open subsets $U, V \subseteq M$, such that $p \in U$ and $q \in V$.
- There exists a countable basis for the topology of M.
- For every $p \in M$, there exists
  - an open subset $U \subseteq M$ containing p,
  - an open subset $\hat{U} \subseteq \mathbf{R}^d$, and
  - a homeomorphism $\phi : U \to \hat{U}$

A topological space that that has a differentiable structure and locally resembles Euclidean space near every point. The dimension d is called the intrinsic dimension.

Manifolds and Embeddings

Definition: embedding

A smooth map between two manifolds whose inverse exists and is also smooth.

- Maps elements from one space to another.

Properties of embeddings

- Injective
- Preserves the structure of the object being embedded.
- Manifold learning:
  - The points close to each other in the high dimensional space remain close in the low dimensional space.

Manifolds and Embeddings

Definition: embedding

A smooth map between two manifolds whose inverse exists and is also smooth.

- Maps elements from one space to another.

Properties of embeddings

- Injective
- Preserves the structure of the object being embedded.
- Manifold learning:
  - The points close to each other in the high dimensional space remain close in the low dimensional space.

The Manifold Assumption and Manifold Learning

The manifold assumption

The data is sampled from a distribution that is close to or supported on, a smooth manifold of dimension d, embedded in $\mathbf{R}^n$.

Application to manifold learning

- The manifold learning algorithm finds a mapping F of a high dimensional point in $\mathbf{R}^n$, to a lower dimensional point in $\mathbf{R}^m$, denoted $\mathbf{R}^n \rightarrow \mathbf{R}^m$.
- Assume that $n > d$.
- Desire that $n > m$ and $m \geq d$.
- Desire that the embedding is found explicitly and is non-linear in nature.

  "Explore a manifold learning algorithm that produces an explicit non-linear model."

The Manifold Assumption and Manifold Learning

The manifold assumption

The data is sampled from a distribution that is close to or supported on, a smooth manifold of dimension d, embedded in $\mathbf{R}^n$.

Application to manifold learning

- The manifold learning algorithm finds a mapping F of a high dimensional point in $\mathbf{R}^n$, to a lower dimensional point in $\mathbf{R}^m$, denoted $\mathbf{R}^n \rightarrow \mathbf{R}^m$.
- Assume that $n > d$.
- Desire that $n > m$ and $m \geq d$.
- Desire that the embedding is found explicitly and is non-linear in nature.
  - "Explore a manifold learning algorithm that produces an explicit non-linear model."

The Manifold Assumption and Manifold Learning

The manifold assumption

The data is sampled from a distribution that is close to or supported on, a smooth manifold of dimension d, embedded in $\mathbf{R}^n$.

Application to manifold learning

- The manifold learning algorithm finds a mapping F of a high dimensional point in $\mathbf{R}^n$, to a lower dimensional point in $\mathbf{R}^m$, denoted $\mathbf{R}^n \rightarrow \mathbf{R}^m$.
- Assume that $n > d$.
- Desire that $n > m$ and $m \geq d$.
- Desire that the embedding is found explicitly and is non-linear in nature.
  - "Explore a manifold learning algorithm that produces an explicit non-linear model."

The Manifold Assumption and Manifold Learning

The manifold assumption

The data is sampled from a distribution that is close to or supported on, a smooth manifold of dimension d, embedded in $\mathbf{R}^n$.

Application to manifold learning

- The manifold learning algorithm finds a mapping F of a high dimensional point in $\mathbf{R}^n$, to a lower dimensional point in $\mathbf{R}^m$, denoted $\mathbf{R}^n \rightarrow \mathbf{R}^m$.
- Assume that $n > d$.
- Desire that $n > m$ and $m \geq d$.
- Desire that the embedding is found explicitly and is non-linear in nature.
  - "Explore a manifold learning algorithm that produces an explicit non-linear model."

The Manifold Assumption and Manifold Learning

The manifold assumption

The data is sampled from a distribution that is close to or supported on, a smooth manifold of dimension d, embedded in $\mathbf{R}^n$.

Application to manifold learning

- The manifold learning algorithm finds a mapping F of a high dimensional point in $\mathbf{R}^n$, to a lower dimensional point in $\mathbf{R}^m$, denoted $\mathbf{R}^n \rightarrow \mathbf{R}^m$.
- Assume that $n > d$.
- Desire that $n > m$ and $m \geq d$.
- Desire that the embedding is found explicitly and is non-linear in nature.
    - "Explore a manifold learning algorithm that produces an explicit non-linear model."

The Manifold Assumption and Manifold Learning

The manifold assumption

The data is sampled from a distribution that is close to or supported on, a smooth manifold of dimension d, embedded in $\mathbf{R}^n$.

Application to manifold learning

- The manifold learning algorithm finds a mapping F of a high dimensional point in $\mathbf{R}^n$, to a lower dimensional point in $\mathbf{R}^m$, denoted $\mathbf{R}^n \to \mathbf{R}^m$.
- Assume that $n > d$.
- Desire that $n > m$ and $m \geq d$.
- Desire that the embedding is found explicitly and is non-linear in nature.
  - "Explore a manifold learning algorithm that produces an explicit non-linear model."

Outline

1 Introduction and Motivation

2 Mathematical Intuition

3 Developing an Algorithm

4 NPPE Algorithm

5 Results and Conclusion

Locally Linear Embedding (LLE)

LLE aims to preserve local relationships between data points.

First Step

- Find the linear coefficients that best reconstructs each data point x_i by its k-nearest neighbors.
- Using Euclidean Distance, the linear reconstruction weights $R_{ij}, i, j = 1, 2, ..., N$ are given by minimizing the sum of the squared distances between all the data points and their reconstructions.
  - x_i is only reconstructed from its neighbors
  - the weights for x_i must sum to 1.

$$R_{ij} = \underset{\sum_{j=1}^{N} R_{ij}=1}{\arg\min} \sum_{i=1}^{N} \left\| x_i - \sum_{j=1}^{N} R_{ij} x_j \right\|_2^2 \qquad \text{(i)}$$

Locally Linear Embedding (LLE)

LLE aims to preserve local relationships between data points.

First Step

- Find the linear coefficients that best reconstructs each data point x_i by its k-nearest neighbors.
- Using Euclidean Distance, the linear reconstruction weights $R_{ij}, i, j = 1, 2, ..., N$ are given by minimizing the sum of the squared distances between all the data points and their reconstructions.
  - x_i is only reconstructed from its neighbors
  - the weights for x_i must sum to 1.

$$R_{ij} = \underset{\sum_{j=1}^{N} R_{ij}=1}{\arg\min} \sum_{i=1}^{N} \left\| x_i - \sum_{j=1}^{N} R_{ij} x_j \right\|_2^2 \qquad \text{(i)}$$

Locally Linear Embedding (LLE)

LLE aims to preserve local relationships between data points.

First Step

- Find the linear coefficients that best reconstructs each data point x_i by its k-nearest neighbors.
- Using Euclidean Distance, the linear reconstruction weights $R_{ij}, i, j = 1, 2, ..., N$ are given by minimizing the sum of the squared distances between all the data points and their reconstructions.
  - x_i is only reconstructed from its neighbors
  - the weights for x_i must sum to 1.

$$R_{ij} = \underset{\sum_{j=1}^{N} R_{ij}=1}{\arg\min} \sum_{i=1}^{N} \left\| x_i - \sum_{j=1}^{N} R_{ij} x_j \right\|_2^2 \quad \text{(i)}$$

Locally Linear Embedding (LLE)

LLE aims to preserve local relationships between data points.

First Step

- Find the linear coefficients that best reconstructs each data point x_i by its k-nearest neighbors.
- Using Euclidean Distance, the linear reconstruction weights $R_{ij}, i, j = 1, 2, ..., N$ are given by minimizing the sum of the squared distances between all the data points and their reconstructions.
  - x_i is only reconstructed from its neighbors
  - the weights for x_i must sum to 1.

$$R_{ij} = \underset{\sum_{j=1}^{N} R_{ij}=1}{\arg\min} \sum_{i=1}^{N} \left\| x_i - \sum_{j=1}^{N} R_{ij} x_j \right\|_2^2 \quad \text{(i)}$$

Locally Linear Embedding (LLE)

LLE aims to preserve local relationships between data points.

First Step

- Find the linear coefficients that best reconstructs each data point x_i by its k-nearest neighbors.
- Using Euclidean Distance, the linear reconstruction weights $R_{ij}, i, j = 1, 2, ..., N$ are given by minimizing the sum of the squared distances between all the data points and their reconstructions.
  - x_i is only reconstructed from its neighbors
  - the weights for x_i must sum to 1.

$$R_{ij} = \underset{\sum_{j=1}^{N} R_{ij}=1}{\arg\min} \sum_{i=1}^{N} \left\| x_i - \sum_{j=1}^{N} R_{ij} x_j \right\|_2^2 \qquad \text{(i)}$$

Locally Linear Embedding (LLE)

LLE aims to preserve local relationships between data points.

First Step

- Find the linear coefficients that best reconstructs each data point x_i by its k-nearest neighbors.
- Using Euclidean Distance, the linear reconstruction weights $R_{ij}, i, j = 1, 2, ..., N$ are given by minimizing the sum of the squared distances between all the data points and their reconstructions.
    - x_i is only reconstructed from its neighbors
    - the weights for x_i must sum to 1.

$$R_{ij} = \underset{\sum_{j=1}^{N} R_{ij}=1}{\arg\min} \sum_{i=1}^{N} \left\| x_i - \sum_{j=1}^{N} R_{ij} x_j \right\|_2^2 \qquad \text{(i)}$$

Locally Linear Embedding

Second Step

LLE then constructs an embedding using the weights R, while optimizing the coordinates y_i, to minimize the error.

- Unit Covariance
- Zero Mean

$$\min \quad \sum_{i=1}^{N} \left\| y_i - \sum_{j=1}^{N} R_{ij} y_j \right\|_2^2 \tag{ii}$$

$$\text{s.t.} \quad \frac{1}{N} \sum_{i=1}^{N} y_i {y_i}^T = I_m$$

$$\sum_{i=0}^{N} y_i = 0$$

Polynomial Mapping Assumption

Given a data set $\mathcal{X} := \{x_1, x_2, ..., x_N\}$ in the high dimensional space $\mathbf{R}^n$, assume there exists an explicit polynomial mapping from $\mathcal{X}$ to its low dimensional representation $\mathcal{Y} := \{y_1, y_2, ..., y_N\}$ in $\mathbf{R}^m$.

The k-th component of $y_i \in \mathcal{Y}$ is defined as a polynomial of degree p with respect to x_i, such that

$$y_i^{(k)} = v_k^T \phi(x_i), \text{ where } v_k \in \mathbf{R}^{pn}$$

Mapping function $\phi(x)$

For a given data vector $x_i \in \mathcal{X}$, define the mapping $\phi : \mathbf{R}^n \to \mathbf{R}^{pn}$ as

$$\phi(x_i) = \begin{bmatrix} \overbrace{x_i \odot x_i \odot \cdots \odot x_i}^{p \text{ times}} \\ \vdots \\ x_i \odot x_i \\ x_i \end{bmatrix}.$$

Polynomial Mapping Assumption

Given a data set $\mathcal{X} := \{x_1, x_2, ..., x_N\}$ in the high dimensional space $\mathbf{R}^n$, assume there exists an explicit polynomial mapping from $\mathcal{X}$ to its low dimensional representation $\mathcal{Y} := \{y_1, y_2, ..., y_N\}$ in $\mathbf{R}^m$.

The k-th component of $y_i \in \mathcal{Y}$ is defined as a polynomial of degree p with respect to x_i, such that

$$y_i^{(k)} = v_k^T \phi(x_i), \text{ where } v_k \in \mathbf{R}^{pn}$$

Mapping function $\phi(x)$

For a given data vector $x_i \in \mathcal{X}$, define the mapping $\phi : \mathbf{R}^n \to \mathbf{R}^{pn}$ as

$$\phi(x_i) = \begin{bmatrix} \overbrace{x_i \odot x_i \odot \cdots \odot x_i}^{p \text{ times}} \\ \vdots \\ x_i \odot x_i \\ x_i \end{bmatrix}.$$

Polynomial Mapping Assumption

Given a data set $\mathcal{X} := \{x_1, x_2, ..., x_N\}$ in the high dimensional space $\mathbf{R}^n$, assume there exists an explicit polynomial mapping from $\mathcal{X}$ to its low dimensional representation $\mathcal{Y} := \{y_1, y_2, ..., y_N\}$ in $\mathbf{R}^m$.

The k-th component of $y_i \in \mathcal{Y}$ is defined as a polynomial of degree p with respect to x_i, such that

$$y_i^{(k)} = v_k^T \phi(x_i), \text{ where } v_k \in \mathbf{R}^{pn}$$

Mapping function $\phi(x)$

For a given data vector $x_i \in \mathcal{X}$, define the mapping $\phi : \mathbf{R}^n \to \mathbf{R}^{pn}$ as

$$\phi(x_i) = \begin{bmatrix} \overbrace{x_i \odot x_i \odot \cdots \odot x_i}^{p \text{ times}} \\ \vdots \\ x_i \odot x_i \\ x_i \end{bmatrix}.$$

Outline

1. Introduction and Motivation
2. Mathematical Intuition
3. Developing an Algorithm
4. NPPE Algorithm
5. Results and Conclusion

NPPE Algorithm Overview

Neighborhood Preserving Polynomial Embedding (NPPE)

Given a data set $\mathcal{X} := \{x_1, x_2, ..., x_N\}$ in the high dimensional space $\mathbf{R}^n$, the NPPE algorithm finds an explicit polynomial mapping from $\mathcal{X}$ to its low dimensional representation $\mathcal{Y} := \{y_1, y_2, ..., y_N\}$ in $\mathbf{R}^m$.

NPPE Algorithm

Inputs: Data matrix $X = [x_1 \quad x_2 \quad \cdots \quad x_N]$ of size $n \times N$, the number k of nearest neighbors, the polynomial degree p, and the low dimensional space m.

1. Compute the linear weights R.
2. Compute the non-linear weights W.
3. Solve the generalized eigenvalue problem to get the eigenvectors $v_i, i = 1, 2, ..., m$.
4. Map the high-dimensional data to the low-dimensional embedding space.

NPPE Algorithm Overview

Neighborhood Preserving Polynomial Embedding (NPPE)

Given a data set $\mathcal{X} := \{x_1, x_2, ..., x_N\}$ in the high dimensional space $\mathbf{R}^n$, the NPPE algorithm finds an explicit polynomial mapping from $\mathcal{X}$ to its low dimensional representation $\mathcal{Y} := \{y_1, y_2, ..., y_N\}$ in $\mathbf{R}^m$.

NPPE Algorithm

Inputs: Data matrix $X = [x_1 \quad x_2 \quad \cdots \quad x_N]$ of size $n \times N$, the number k of nearest neighbors, the polynomial degree p, and the low dimensional space m.

1. Compute the linear weights R.
2. Compute the non-linear weights W.
3. Solve the generalized eigenvalue problem to get the eigenvectors $v_i, i = 1, 2, ..., m$.
4. Map the high-dimensional data to the low-dimensional embedding space.

NPPE Algorithm Overview

Neighborhood Preserving Polynomial Embedding (NPPE)

Given a data set $\mathcal{X} := \{x_1, x_2, ..., x_N\}$ in the high dimensional space $\mathbf{R}^n$, the NPPE algorithm finds an explicit polynomial mapping from $\mathcal{X}$ to its low dimensional representation $\mathcal{Y} := \{y_1, y_2, ..., y_N\}$ in $\mathbf{R}^m$.

NPPE Algorithm

Inputs: Data matrix $X = [x_1 \quad x_2 \quad \cdots \quad x_N]$ of size $n \times N$, the number k of nearest neighbors, the polynomial degree p, and the low dimensional space m.

1. Compute the linear weights R.
2. Compute the non-linear weights W.
3. Solve the generalized eigenvalue problem to get the eigenvectors $v_i, i = 1, 2, ..., m$.
4. Map the high-dimensional data to the low-dimensional embedding space.

NPPE Algorithm - Step 1

Computing the Linear Weights

$$R_{ij} = \underset{\sum_{j=1}^{N} R_{ij}=1}{\arg\min} \quad \sum_{i=1}^{N} \left\| x_i - \sum_{j=1}^{N} R_{ij} x_j \right\|_2^2 \tag{i}$$

The linear weight matrix $R = [r_1 \quad r_2 \ \cdots \ r_N]$ in $\mathbf{R}^{N \times N}$, is given by computing the following closed formed solution for r_i, where

- e is a column vector of all ones
- $G_{jl} = (x_j - x_i)^T (x_l - x_i)$ where x_j, x_l are in the k-nearest neighbors of x_i.

$$r_i = \frac{G^{-1} e}{e^T G^{-1} e} \tag{1}$$

NPPE Algorithm - Step 2

Computing the Non-linear Weights

The non-linear weight matrix $W \in \mathbf{R}^{N \times N}$, is computed by the following equation.

$$W_{ij} = R_{ij} + R_{ji} - \sum_{k=1}^{N} R_{ik} R_{kj}, \text{ and } \sum_{j=1}^{N} W_{ij} = 1 \tag{2}$$

NPPE Algorithm - Step 3

LLE Eigenvalue Problem

- Define an eigenvalue problem as $Av = \lambda v$.

$$A = (I - R)^T (I - R)$$

Solving the Generalized Eigenvalue Problem for NPPE

- Define a generalized eigenvalue problem as $Av = \lambda Bv$.

$$\phi(D - W)\phi^T v_i = \lambda \phi D \phi^T v_i, \quad v_i^T \phi D \phi^T v_j = \delta_{ij} \tag{3}$$

- Define $\phi = [\phi(x_1) \quad \phi(x_2) \quad \cdots \quad \phi(x_N)]$ to be a $(pn) \times N$ matrix.
- The optimal solutions are found by obtaining the eigenvectors $v_i, i = 1, 2, ..., m$, corresponding to the m smallest eigenvalues.

NPPE Algorithm - Step 3

LLE Eigenvalue Problem

- Define an eigenvalue problem as $Av = \lambda v$.

$$A = (I - R)^T (I - R)$$

Solving the Generalized Eigenvalue Problem for NPPE

- Define a generalized eigenvalue problem as $Av = \lambda Bv$.

$$\phi(D - W)\phi^T v_i = \lambda \phi D \phi^T v_i, \quad v_i^T \phi D \phi^T v_j = \delta_{ij} \tag{3}$$

- Define $\phi = [\phi(x_1) \quad \phi(x_2) \quad \cdots \quad \phi(x_N)]$ to be a $(pn) \times N$ matrix.
- The optimal solutions are found by obtaining the eigenvectors $v_i, i = 1, 2, ..., m$, corresponding to the m smallest eigenvalues.

NPPE Algorithm - Step 3

LLE Eigenvalue Problem

- Define an eigenvalue problem as $Av = \lambda v$.

$$A = (I - R)^T (I - R)$$

Solving the Generalized Eigenvalue Problem for NPPE

- Define a generalized eigenvalue problem as $Av = \lambda Bv$.

$$\phi(D - W)\phi^T v_i = \lambda \phi D \phi^T v_i, \quad v_i^T \phi D \phi^T v_j = \delta_{ij} \tag{3}$$

- Define $\phi = [\phi(x_1) \quad \phi(x_2) \quad \cdots \quad \phi(x_N)]$ to be a $(pn) \times N$ matrix.
- The optimal solutions are found by obtaining the eigenvectors $v_i, i = 1, 2, ..., m$, corresponding to the m smallest eigenvalues.

NPPE Algorithm - Step 3

LLE Eigenvalue Problem

- Define an eigenvalue problem as $Av = \lambda v$.

$$A = (I - R)^T (I - R)$$

Solving the Generalized Eigenvalue Problem for NPPE

- Define a generalized eigenvalue problem as $Av = \lambda Bv$.

$$\phi(D - W)\phi^T v_i = \lambda \phi D \phi^T v_i, \quad {v_i}^T \phi D \phi^T v_j = \delta_{ij} \tag{3}$$

- Define $\phi = [\phi(x_1) \quad \phi(x_2) \quad \cdots \quad \phi(x_N)]$ to be a $(pn) \times N$ matrix.
- The optimal solutions are found by obtaining the eigenvectors $v_i, i = 1, 2, ..., m$, corresponding to the m smallest eigenvalues.

NPPE Algorithm - Step 3

LLE Eigenvalue Problem

- Define an eigenvalue problem as $Av = \lambda v$.

$$A = (I - R)^T (I - R)$$

Solving the Generalized Eigenvalue Problem for NPPE

- Define a generalized eigenvalue problem as $Av = \lambda Bv$.

$$\phi(D - W)\phi^T v_i = \lambda \phi D \phi^T v_i, \quad v_i^T \phi D \phi^T v_j = \delta_{ij} \tag{3}$$

- Define $\phi = [\phi(x_1) \quad \phi(x_2) \quad \cdots \quad \phi(x_N)]$ to be a $(pn) \times N$ matrix.
- The optimal solutions are found by obtaining the eigenvectors $v_i, i = 1, 2, ..., m$, corresponding to the m smallest eigenvalues.

NPPE Algorithm - Step 4

Polynomial Mapping Assumption

$$y_i^{(k)} = v_k^T \phi(x_i), \text{ where } v_k \in \mathbf{R}^{pn}$$

Mapping to the Low Dimensional Space

For a data sample $x_i \in \mathcal{X}$, its low dimensional representation $y_i \in \mathcal{Y}$ is given by

$$y_i = \begin{bmatrix} v_1^T \phi(x_i) & v_2^T \phi(x_i) & \cdots & v_m^T \phi(x_i) \end{bmatrix}^T \tag{4}$$

Importantly, the mapping function above holds true for a new data sample $x_t \notin \mathcal{X}$, allowing for its low dimensional representation y_t to be computed efficiently.

NPPE Algorithm - Step 4

Polynomial Mapping Assumption

$$y_i^{(k)} = v_k^T \phi(x_i), \text{ where } v_k \in \mathbf{R}^{pn}$$

Mapping to the Low Dimensional Space

For a data sample $x_i \in \mathcal{X}$, its low dimensional representation $y_i \in \mathcal{Y}$ is given by

$$y_i = \begin{bmatrix} v_1^T \phi(x_i) & v_2^T \phi(x_i) & \cdots & v_m^T \phi(x_i) \end{bmatrix}^T \tag{4}$$

Importantly, the mapping function above holds true for a new data sample $x_t \notin \mathcal{X}$, allowing for its low dimensional representation y_t to be computed efficiently.

Outline

1 Introduction and Motivation

2 Mathematical Intuition

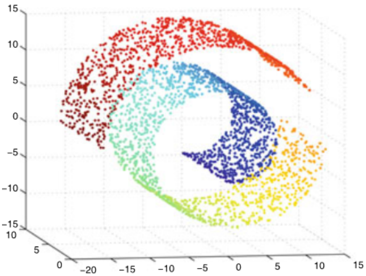
3 Developing an Algorithm

4 NPPE Algorithm

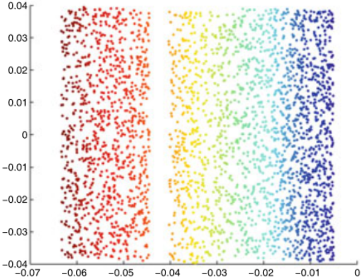
5 Results and Conclusion

Results on Swissroll Dataset

The parameter of k-nearest neighbors is set to be 1% of the training samples N and the polynomial degree $p = 2$.

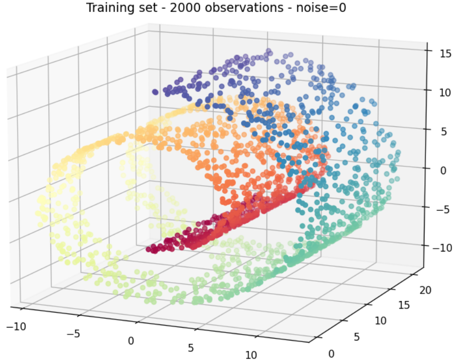


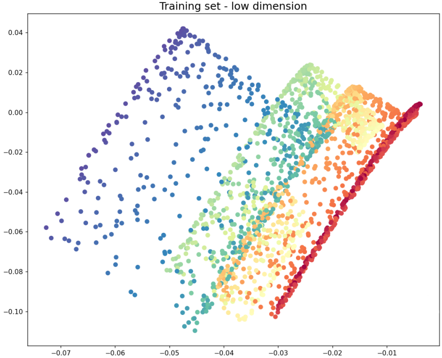
SwissRoll dataset in $\mathbf{R}^3$



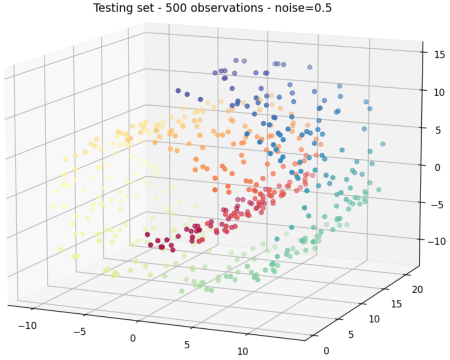
SwissRoll representation in $\mathbf{R}^2$

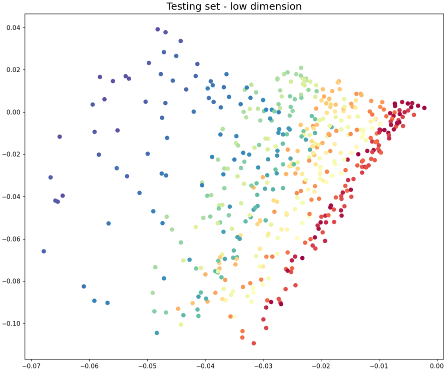
My Current Results





Performance on New Data





Future Work

- Use the algorithm as a pre-processing step to improve the performance of machine learning algorithms.
- Improving the algorithm to use less memory / GPU Compute.
- Research methods for estimating the intrinsic dimension.
- Applications in image/data compression.

Future Work

- Use the algorithm as a pre-processing step to improve the performance of machine learning algorithms.
- Improving the algorithm to use less memory / GPU Compute.
- Research methods for estimating the intrinsic dimension.
- Applications in image/data compression.

Future Work

- Use the algorithm as a pre-processing step to improve the performance of machine learning algorithms.
- Improving the algorithm to use less memory / GPU Compute.
- Research methods for estimating the intrinsic dimension.
- Applications in image/data compression.

Future Work

- Use the algorithm as a pre-processing step to improve the performance of machine learning algorithms.
- Improving the algorithm to use less memory / GPU Compute.
- Research methods for estimating the intrinsic dimension.
- Applications in image/data compression.

References I

Benyamin Ghojogh, Fakhri Karray, and Mark Crowley.

Eigenvalue and generalized eigenvalue problems: Tutorial, 03 2019.

Marina Meilă and Hanyu Zhang.

Manifold learning: What, how, and why.

Annual Review of Statistics and Its Application, 11(Volume 11, 2024):393–417, 2024.

Hong Qiao, Chao Ma, and Rui Li.

The "Hand-eye-brain" System of Intelligent Robot: From Interdisciplinary Perspective of Information Science and Neuroscience, chapter Explicit Nonlinear Mapping for Manifold Learning with Neighborhood Preserving Polynomial Embedding, pages 81–106.

Springer Singapore, 2022.

Hong Qiao, Peng Zhang, Di Wang, and Bo Zhang.

An Explicit Nonlinear Mapping for Manifold Learning.

IEEE Transactions on Cybernetics, 43(1):51–63, 2013.

Lawrence Saul and Sam Roweis.

An Introduction to Locally Linear Embedding.

Journal of Machine Learning Research, 7, 2001.

References II

Cosma Rohilla Shalizi.
Lecture 14: Data mining.
Online, 2009.
Accessed: 2024-05-02.

Laurens van der Maaten, Eric Postma, and H. Herik.
Dimensionality Reduction: A Comparative Review.
Journal of Machine Learning Research - JMLR, 10, 01 2007.

Jianzhong Wang.
Geometric Structure of High-Dimensional Data and Dimensionality Reduction, chapter Locally Linear Embedding, pages 203–220.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

Acknowledgements

Research Advisor: Jonathan Duncan, PhD

Eigen 3.4.0.
https://gitlab.com/libeigen/eigen.

Keras 2.15.0.
https://github.com/fchollet/keras.

Scikit-learn 1.3.2.
https://github.com/scikit-learn/scikit-learn.

Questions?