ManifoldLearning Documentation

Release 0.1.0

Art Wild

Oct 04, 2018

Contents

ManifoldLearning.jl is a Julia package for manifold learning and non-linear dimensionality reduction. It proides set of nonlinear dimensionality reduction methods, such as *Isomap*, *LLE*, *LTSA*, etc.

Methods:

CHAPTER 1

Isomap

Isomap is a method for low-dimensional embedding. Isomap is used for computing a quasi-isometric, low-dimensional embedding of a set of high-dimensional data points[1].

This package defines a `Isomap` type to represent a Isomap results, and provides a set of methods to access its properties.

1.1 Properties

Let `M` be an instance of `Isomap`, `n` be the number of observations, and `d` be the output dimension.

outdim(*M*)
: Get the output dimension `d`, *i.e* the dimension of the subspace.

projection(*M*)
: Get the projection matrix (of size `(d, n)`). Each column of the projection matrix corresponds to an observation in projected subspace.

neighbors(*M*)
: The number of nearest neighbors used for approximating local coordinate structure.

ccomponent(*M*)
: The observations index array of the largest connected component of the distance matrix.

1.2 Data Transformation

One can use the `transform` method to perform Isomap over a given dataset.

transform(*Isomap*, *X; ...*)
: Perform Isomap over the data given in a matrix `X`. Each column of `X` is an observation.

[1] Tenenbaum, J. B., de Silva, V. and Langford, J. C. "A Global Geometric Framework for Nonlinear Dimensionality Reduction". Science 290 (5500): 2319-2323, 22 December 2000. http://isomap.stanford.edu/

This method returns an instance of `Isomap`.

Keyword arguments:

name	description	default
k	The number of nearest neighbors for determining local coordinate structure.	`12`
d	Output dimension.	`2`

Example:

```
using ManifoldLearning

# suppose X is a data matrix, with each observation in a column
# apply Isomap transformation to the dataset
Y = transform(Isomap, X; k = 12, d = 2)
```

References

CHAPTER 2

Diffusion maps

Diffusion maps leverages the relationship between heat diffusion and a random walk; an analogy is drawn between the diffusion operator on a manifold and a Markov transition matrix operating on functions defined on the graph whose nodes were sampled from the manifold[1].

This package defines a `DiffMap` type to represent a Hessian LLE results, and provides a set of methods to access its properties.

2.1 Properties

Let `M` be an instance of `DiffMap`, `n` be the number of observations, and `d` be the output dimension.

outdim(*M*)
: Get the output dimension `d`, *i.e* the dimension of the subspace.

projection(*M*)
: Get the projection matrix (of size `(d, n)`). Each column of the projection matrix corresponds to an observation in projected subspace.

kernel(*M*)
: The kernel matrix.

2.2 Data Transformation

One can use the `transform` method to perform DiffMap over a given dataset.

transform(*DiffMap, X; ...*)
: Perform DiffMap over the data given in a matrix `X`. Each column of `X` is an observation.

  This method returns an instance of `DiffMap`.

[1] Coifman, R. & Lafon, S. "Diffusion maps". Applied and Computational Harmonic Analysis, Elsevier, 2006, 21, 5-30. DOI:10.1073/pnas.0500334102

Keyword arguments:

name	description	default
d	Output dimension.	`2`
t	Number of time steps.	`1`
	The scale parameter.	`1.0`

Example:

```
using ManifoldLearning

# suppose X is a data matrix, with each observation in a column
# apply DiffMap transformation to the dataset
Y = transform(DiffMap, X; d=2, t=1, =1.0)
```

References

CHAPTER 3

Laplacian Eigenmaps

Laplacian Eigenmaps (LEM) method uses spectral techniques to perform dimensionality reduction. This technique relies on the basic assumption that the data lies in a low-dimensional manifold in a high-dimensional space. The algorithm provides a computationally efficient approach to non-linear dimnsionality reduction that has locally preserving properties[1].

This package defines a `LEM` type to represent a Laplacian Eigenmaps results, and provides a set of methods to access its properties.

3.1 Properties

Let `M` be an instance of `LEM`, `n` be the number of observations, and `d` be the output dimension.

outdim(*M*)
: Get the output dimension `d`, *i.e* the dimension of the subspace.

projection(*M*)
: Get the projection matrix (of size `(d, n)`). Each column of the projection matrix corresponds to an observation in projected subspace.

neighbors(*M*)
: The number of nearest neighbors used for approximating local coordinate structure.

eigvals(*M*)
: The eigenvalues of alignment matrix.

3.2 Data Transformation

One can use the `transform` method to perform LEM over a given dataset.

[1] Belkin, M. and Niyogi, P. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation". Neural Computation, June 2003; 15 (6):1373-1396. DOI:10.1162/089976603321780317

transform(*LEM*, *X; ...*)

Perform LEM over the data given in a matrix `X`. Each column of `X` is an observation.

This method returns an instance of `LEM`.

Keyword arguments:

name	description	default
k	The number of nearest neighbors for determining local coordinate structure.	`12`
d	Output dimension.	`2`
t	The temperature parameters of the heat kernel.	`1.0`

Example:

```
using ManifoldLearning

# suppose X is a data matrix, with each observation in a column
# apply Laplacian Eigenmaps transformation to the dataset
Y = transform(LEM, X; k = 12, d = 2, t = 1.0)
```

References

CHAPTER 4

Locally Linear Embedding

Locally Linear Embedding (LLE) technique builds a single global coordinate system of lower dimensionality. By exploiting the local symmetries of linear reconstructions, LLE is able to learn the global structure of nonlinear manifolds[1].

This package defines a `LLE` type to represent a LLE results, and provides a set of methods to access its properties.

4.1 Properties

Let `M` be an instance of `LLE`, `n` be the number of observations, and `d` be the output dimension.

`outdim`(*M*)
Get the output dimension `d`, *i.e* the dimension of the subspace.

`projection`(*M*)
Get the projection matrix (of size `(d, n)`). Each column of the projection matrix corresponds to an observation in projected subspace.

`neighbors`(*M*)
The number of nearest neighbors used for approximating local coordinate structure.

`eigvals`(*M*)
The eigenvalues of alignment matrix.

4.2 Data Transformation

One can use the `transform` method to perform HLLE over a given dataset.

`transform`(*LLE, X; ...*)
Perform LLE over the data given in a matrix `X`. Each column of `X` is an observation.

[1] Roweis, S. & Saul, L. "Nonlinear dimensionality reduction by locally linear embedding", Science 290:2323 (2000). DOI:10.1126/science.290.5500.2323

This method returns an instance of `LLE`.

Keyword arguments:

name	description	default
k	The number of nearest neighbors for determining local coordinate structure.	`12`
d	Output dimension.	`2`

Example:

```
using ManifoldLearning

# suppose X is a data matrix, with each observation in a column
# apply LLE transformation to the dataset
Y = transform(LLE, X; k = 12, d = 2)
```

References

CHAPTER 5

Hessian Eigenmaps

The Hessian Eigenmaps (Hessian LLE, HLLE) method adapts the weights in *LLE* to minimize the Hessian operator. Like *LLE*, it requires careful setting of the nearest neighbor parameter. The main advantage of Hessian LLE is the only method designed for non-convex data sets[1].

This package defines a `HLLE` type to represent a Hessian LLE results, and provides a set of methods to access its properties.

5.1 Properties

Let `M` be an instance of `HLLE`, `n` be the number of observations, and `d` be the output dimension.

outdim(*M*)
Get the output dimension `d`, *i.e* the dimension of the subspace.

projection(*M*)
Get the projection matrix (of size `(d, n)`). Each column of the projection matrix corresponds to an observation in projected subspace.

neighbors(*M*)
The number of nearest neighbors used for approximating local coordinate structure.

eigvals(*M*)
The eigenvalues of alignment matrix.

5.2 Data Transformation

One can use the `transform` method to perform HLLE over a given dataset.

[1] Donoho, D. and Grimes, C. "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data", Proc. Natl. Acad. Sci. USA. 2003 May 13; 100(10): 5591–5596. DOI:10.1073/pnas.1031596100

transform(*HLLE, X; ...*)

Perform HLLE over the data given in a matrix `X`. Each column of `X` is an observation.

This method returns an instance of `HLLE`.

Keyword arguments:

name	description	default
k	The number of nearest neighbors for determining local coordinate structure.	12
d	Output dimension.	2

Example:

```
using ManifoldLearning

# suppose X is a data matrix, with each observation in a column
# apply HLLE transformation to the dataset
Y = transform(HLLE, X; k = 12, d = 2)
```

References

CHAPTER 6

Local Tangent Space Alignment

Local tangent space alignment (LTSA) is a method for manifold learning, which can efficiently learn a nonlinear embedding into low-dimensional coordinates from high-dimensional data, and can also reconstruct high-dimensional coordinates from embedding coordinates[1].

This package defines a `LTSA` type to represent a LTSA results, and provides a set of methods to access its properties.

6.1 Properties

Let `M` be an instance of `LTSA`, `n` be the number of observations, and `d` be the output dimension.

outdim(*M*)
: Get the output dimension `d`, *i.e* the dimension of the subspace.

projection(*M*)
: Get the projection matrix (of size `(d, n)`). Each column of the projection matrix corresponds to an observation in projected subspace.

neighbors(*M*)
: The number of nearest neighbors used for approximating local coordinate structure.

eigvals(*M*)
: The eigenvalues of alignment matrix.

6.2 Data Transformation

One can use the `transform` method to perform LTSA over a given dataset.

transform(*LSTA, X; ...*)
: Perform LTSA over the data given in a matrix `X`. Each column of `X` is an observation.

[1] Zhang, Zhenyue; Hongyuan Zha. "Principal Manifolds and Nonlinear Dimension Reduction via Local Tangent Space Alignment". SIAM Journal on Scientific Computing 26 (1): 313–338, 2004. DOI:10.1137/s1064827502419154

This method returns an instance of `LTSA`.

Keyword arguments:

name	description	default
k	The number of nearest neighbors for determining local coordinate structure.	`12`
d	Output dimension.	`2`

Example:

```
using ManifoldLearning

# suppose X is a data matrix, with each observation in a column
# apply LTSA transformation to the dataset
Y = transform(LTSA, X; k = 12, d = 2)
```

References

Notes:

All methods implemented in this package adopt the column-major convention of JuliaStats: in a data matrix, each column corresponds to a sample/observation, while each row corresponds to a feature (variable or attribute).

Index