<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Isomap</td>
<td>3</td>
</tr>
<tr>
<td>2 Diffusion maps</td>
<td>5</td>
</tr>
<tr>
<td>3 Laplacian Eigenmaps</td>
<td>7</td>
</tr>
<tr>
<td>4 Locally Linear Embedding</td>
<td>9</td>
</tr>
<tr>
<td>5 Hessian Eigenmaps</td>
<td>11</td>
</tr>
<tr>
<td>6 Local Tangent Space Alignment</td>
<td>13</td>
</tr>
</tbody>
</table>
ManifoldLearning.jl is a Julia package for manifold learning and non-linear dimensionality reduction. It provides set of nonlinear dimensionality reduction methods, such as Isomap, LLE, LTSA, etc.

Methods:
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 \texttt{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 \texttt{Isomap}, \( n \) be the number of observations, and \( d \) be the output dimension.

- \texttt{outdim}(\( M \))
  - Get the output dimension \( d \), i.e the dimension of the subspace.

- \texttt{projection}(\( M \))
  - Get the projection matrix (of size \((d, n)\)). Each column of the projection matrix corresponds to an observation in projected subspace.

- \texttt{neighbors}(\( M \))
  - The number of nearest neighbors used for approximating local coordinate structure.

- \texttt{ccomponent}(\( M \))
  - The observations index array of the largest connected component of the distance matrix.

### 1.2 Data Transformation

One can use the \texttt{transform} method to perform Isomap over a given dataset.

- \texttt{transform(\texttt{Isomap}, X; ...)}
  - Perform Isomap over the data given in a matrix \( X \). Each column of \( X \) is an observation.

---

This method returns an instance of Isomap.

**Keyword arguments:**

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>The number of nearest neighbors for determining local coordinate structure.</td>
<td>12</td>
</tr>
<tr>
<td>d</td>
<td>Output dimension.</td>
<td>2</td>
</tr>
</tbody>
</table>

**Example:**

```plaintext
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**
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 \texttt{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 \texttt{DiffMap}, \(n\) be the number of observations, and \(d\) be the output dimension.

\[\text{outdim}(M)\]

Get the output dimension \(d\), \textit{i.e.} the dimension of the subspace.

\[\text{projection}(M)\]

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

\[\text{kernel}(M)\]

The kernel matrix.

### 2.2 Data Transformation

One can use the \texttt{transform} method to perform DiffMap over a given dataset.

\[\text{transform}(\text{DiffMap}, X; ...)\]

Perform DiffMap over the data given in a matrix \(X\). Each column of \(X\) is an observation.

This method returns an instance of \texttt{DiffMap}.

Keyword arguments:

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>Output dimension.</td>
<td>2</td>
</tr>
<tr>
<td>t</td>
<td>Number of time steps.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>The scale parameter.</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Example:

```julia
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
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 dimensionality 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.

---

**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:**

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>The number of nearest neighbors for determining local coordinate structure.</td>
<td>12</td>
</tr>
<tr>
<td>d</td>
<td>Output dimension.</td>
<td>2</td>
</tr>
<tr>
<td>t</td>
<td>The temperature parameters of the heat kernel.</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**Example:**

```plaintext
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**

8 Chapter 3. Laplacian Eigenmaps
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.

---

This method returns an instance of LLE.

**Keyword arguments:**

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>The number of nearest neighbors for determining local coordinate structure.</td>
<td>12</td>
</tr>
<tr>
<td>d</td>
<td>Output dimension.</td>
<td>2</td>
</tr>
</tbody>
</table>

**Example:**

```julia
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**
The Hessian Eigenmaps (Hessian LLE, HLLE) method adapts the weights in \textit{LLE} to minimize the Hessian operator. Like \textit{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\textsuperscript{1}.

This package defines a \texttt{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 \texttt{HLLE}, $n$ be the number of observations, and $d$ be the output dimension.

- \texttt{outdim}(M)
  - Get the output dimension $d$, \textit{i.e.} the dimension of the subspace.

- \texttt{projection}(M)
  - Get the projection matrix (of size $(d, n)$). Each column of the projection matrix corresponds to an observation in projected subspace.

- \texttt{neighbors}(M)
  - The number of nearest neighbors used for approximating local coordinate structure.

- \texttt{eigvals}(M)
  - The eigenvalues of alignment matrix.

### 5.2 Data Transformation

One can use the \texttt{transform} method to perform HLLE over a given dataset.

transform(\texttt{HLLE}, X; \ldots)

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

This method returns an instance of \texttt{HLLE}.

Keyword arguments:

\begin{tabular}{|c|p{8cm}|c|}
\hline
name & description & default \\
\hline
\texttt{k} & The number of nearest neighbors for determining local coordinate structure. & 12 \\
\texttt{d} & Output dimension. & 2 \\
\hline
\end{tabular}

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
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.

\texttt{outdim}(M)
Get the output dimension \( d \), i.e. the dimension of the subspace.

\texttt{projection}(M)
Get the projection matrix (of size \((d, n)\)). Each column of the projection matrix corresponds to an observation in projected subspace.

\texttt{neighbors}(M)
The number of nearest neighbors used for approximating local coordinate structure.

\texttt{eigvals}(M)
The eigenvalues of alignment matrix.

6.2 Data Transformation

One can use the \texttt{transform} method to perform LTSA over a given dataset.

\texttt{transform}(LTSA, X; ...)
Perform LTSA over the data given in a matrix \( X \). Each column of \( X \) is an observation.

This method returns an instance of LTSA.

**Keyword arguments:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>The number of nearest neighbors for determining local coordinate structure.</td>
<td>12</td>
</tr>
<tr>
<td>d</td>
<td>Output dimension.</td>
<td>2</td>
</tr>
</tbody>
</table>

**Example:**

```julia
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

C
ccomponent() (built-in function), 3

E
eigvals() (built-in function), 7, 9, 11, 13

K
kernel() (built-in function), 5

N
neighbors() (built-in function), 3, 7, 9, 11, 13

O
outdim() (built-in function), 3, 5, 7, 9, 11, 13

P
projection() (built-in function), 3, 5, 7, 9, 11, 13

T
transform() (built-in function), 3, 5, 7, 9, 11, 13