Namespace SharpMiner

Classes

Component

Represent a principal component. A PC is a linear combination of the original variables that maximally explain the variance of all the variables

DataSetLoader

This class contains helper methods to read a dataset.

FactorResults

This class represents a summary of the factor analysis.

MatrixHelper

Helper class that contains various static methods

PrincipalComponentAnalysis

This class implements the Principal Component Analysis algortithm

Specs

This class is used to parameterize a Computation

Enums

DecompositionMethod

Specify which decomposition method to use to compute the principal components

FactorMethod

Class Component

Namespace: SharpMiner

Assembly: SharpMiner.dll

Represent a principal component. A PC is a linear combination of the original variables that maximally explain the variance of all the variables

```
public class Component
```

Inheritance

object ← Component

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Component(int, double, FactorResults, FactorResults)

Initialise a component object

```
public Component(int rank, double explainedVariance, FactorResults featureResults,
FactorResults rowResults)
```

Parameters

`rank` int

`explainedVariance` double

`featureResults` FactorResults

`rowResults` FactorResults

Properties

FeaturesResults

Represents statistics computed for the features of the dataset

```
public FactorResults FeaturesResults { get; }
```

Property Value

FactorResults

PercentExplainedVariance

The percentage of the total information of the dataset that this single component explains

```
public double PercentExplainedVariance { get; }
```

Property Value

double

Rank

The rank of the component by the amount of information explained.

```
public int Rank { get; }
```

Property Value

int

RowsResults

Represents statistics computed for the rows of the dataset

```
public FactorResults RowsResults { get; }
```

Property Value

FactorResults

Class DataSetLoader

Namespace: SharpMiner

Assembly: SharpMiner.dll

This class contains helper methods to read a dataset.

```
public static class DataSetLoader
```

Inheritance

object ← DataSetLoader

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

ConvertToMatrix(DataTable)

Converts a DataTable to a Matrix of doubles.

```
public static Matrix<double> ConvertToMatrix(this DataTable dataTable)
```

Parameters

`dataTable` DataTable

The DataTable to convert.

Returns

Matrix<double>

A Matrix of doubles representing the data in the DataTable.

LoadCsvFromRemoteFile(string, NumberFormatInfo, bool, string)

Creates a dataset instance from a remote CSV file.

```
public static Matrix<double> LoadCsvFromRemoteFile(string url, NumberFormatInfo
numberProvider = null, bool hasHeaders = true, string delimiter = ",")
```

Parameters

`url` string

The URL of the remote CSV file.

`numberProvider` NumberFormatInfo

The number format provider.

`hasHeaders` bool

Indicates if the CSV file has headers.

`delimiter` string

The delimiter used in the CSV file.

Returns

Matrix<double>

A Matrix containing the data from the remote CSV file.

LoadDataTableCsvFromRemoteFile(string, bool, string, NumberFormatInfo, string[], string[])

Loads a DataTable from a remote CSV file.

```
public static DataTable LoadDataTableCsvFromRemoteFile(string url, bool hasHeaders =
true, string delimiter = ",", NumberFormatInfo numberProvider = null, string[]
oneHotEncodedColumns = null, string[] labelEncodedColumns = null)
```

Parameters

`url` string

The URL of the remote CSV file.

`hasHeaders` bool

Indicates if the CSV file has headers.

`delimiter` string

The delimiter used in the CSV file.

`numberProvider` NumberFormatInfo

The number format provider.

`oneHotEncodedColumns` string[]

Columns that need to be one hot encoded.

`labelEncodedColumns` string[]

Columns that need to be label encoded.

Returns

DataTable

A DataTable containing the data from the remote CSV file.

LoadDataTableFromCsvFile(string, bool, string, NumberFormatInfo, string[], string[])

Loads a DataTable from a CSV file.

```
public static DataTable LoadDataTableFromCsvFile(string fileName, bool hasHeaders =
true, string delimiter = ",", NumberFormatInfo numberProvider = null, string[]
oneHotEncodedColumns = null, string[] labelEncodedColumns = null)
```

Parameters

`fileName` string

The path to the CSV file.

`hasHeaders` bool

Indicates if the CSV file has headers.

`delimiter` string

The delimiter used in the CSV file.

`numberProvider` NumberFormatInfo

The number format provider.

`oneHotEncodedColumns` string[]

Columns that need to be one hot encoded.

`labelEncodedColumns` string[]

Columns that need to be label encoded.

Returns

DataTable

A DataTable containing the data from the remote CSV file.

LoadFromCsvFile(string, NumberFormatInfo, bool, string)

Creates a dataset instance from a CSV file.

```
public static Matrix<double> LoadFromCsvFile(string fileName, NumberFormatInfo
numberProvider = null, bool hasHeaders = true, string delimiter = ",")
```

Parameters

`fileName` string

The path to the CSV file.

`numberProvider` NumberFormatInfo

The number format provider.

`hasHeaders` bool

Indicates if the CSV file has headers.

`delimiter` string

The delimiter used in the CSV file.

Returns

Matrix<double>

A Matrix containing the data from the CSV file.

Enum DecompositionMethod

Namespace: SharpMiner

Assembly: SharpMiner.dll

Specify which decomposition method to use to compute the principal components

```
public enum DecompositionMethod
```

Fields

`Svd = 0`

Singular Value Decomposition

Enum FactorMethod

Namespace: SharpMiner

Assembly: SharpMiner.dll

```
public enum FactorMethod
```

Fields

`MCA = 1`

Multi Correspondance Analysis

`PCA = 0`

Principal Component Analysis

Class FactorResults

Namespace: SharpMiner

Assembly: SharpMiner.dll

This class represents a summary of the factor analysis.

```
public class FactorResults
```

Inheritance

object ← FactorResults

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Contributions

Represents the contributions of observations or variables to the principal components in a Principal Component Analysis (PCA). This field quantifies how much each observation or variable contributes to the formation of each principal component, helping to identify which elements are most influential in defining the axes of the PCA space.

```
public Matrix<double> Contributions { get; }
```

Property Value

Matrix<double>

Remarks

The contributions are often expressed as a percentage and organized in a matrix where rows correspond to observations or variables, and columns represent the principal components. Key points to note:

- A high contribution value for an observation or variable in a particular principal component suggests that it plays a significant role in defining that component.
- Low contribution values indicate that the observation or variable has a lesser influence on the component's definition.

Contribution values can help in understanding the structure of the PCA and identifying which variables or observations drive the main patterns and directions in the data.

Coordinates

Represents the coordinates of a Principal Component Analysis (PCA). This field can hold either the scores or loadings of the PCA, depending on the context of its use.

```
public Matrix<double> Coordinates { get; }
```

Property Value

Matrix<double>

CumulativeExplainedVariance

Get the cumulative sum of explained variance

```
public Vector<double> CumulativeExplainedVariance { get; }
```

Property Value

Vector<double>

ExplainedVariance

Get the percentage of explained variance by each components

```
public Vector<double> ExplainedVariance { get; }
```

Property Value

Vector<double>

SquaredCosinus

Represents the squared cosines (SquaredCosinus) of the observations or variables in a Principal Component Analysis (PCA). This field provides a measure of the quality of the representation of each observation or variable in the reduced PCA space. Each value indicates the proportion of the total variance of an observation or variable that is explained by the selected principal components, effectively quantifying how well it is represented in the PCA dimensions.

```
public Matrix<double> SquaredCosinus { get; }
```

Property Value

Matrix<double>

Class MatrixHelper

Namespace: SharpMiner

Assembly: SharpMiner.dll

Helper class that contains various static methods

```
public static class MatrixHelper
```

Inheritance

object ← MatrixHelper

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

CenterAndScale(Matrix<double>)

Centers and scales the columns of the input matrix by subtracting the mean and dividing by the standard deviation of each column.

```
public static Matrix<double> CenterAndScale(Matrix<double> data)
```

Parameters

`data` Matrix<double>

The matrix containing the data to be centered and scaled, where each column represents a variable.

Returns

Matrix<double>

A new matrix where each column has been centered (mean subtracted) and scaled (divided by standard deviation). If a column's standard deviation is zero, the data is only centered, without scaling.

Remarks

This method normalizes the input matrix by making each column have a mean of 0 and a standard deviation of 1, which is often useful for machine learning algorithms or statistical analysis.

ComputeCorrelationMatrix(Matrix<double>)

Computes the correlation matrix from the given data matrix. Each column of the data matrix represents a variable, and each row represents an observation.

```
public static Matrix<double> ComputeCorrelationMatrix(Matrix<double> data)
```

Parameters

`data` Matrix<double>

A matrix of size (n x m) where 'n' is the number of observations and 'm' is the number of variables.

Returns

Matrix<double>

A correlation matrix of size (m x m) where the element at (i, j) represents the correlation coefficient between variables i and j.

ComputeSignFlip(Svd<double>, Matrix<double>, out Matrix<double>, out Matrix<double>)

The `ComputeSignFlip` function adjusts the signs of the singular vectors (both left and right) from the Singular Value Decomposition (SVD) of a matrix in a consistent way. This is necessary because the SVD of a matrix is not unique: the signs of the singular vectors can be arbitrarily flipped (positive or negative) without changing the mathematical correctness of the decomposition. This function resolves these ambiguities by ensuring consistent signs across both the left and right singular vectors.

```
public static void ComputeSignFlip(Svd<double> svd, Matrix<double> X, out
Matrix<double> U_prime, out Matrix<double> V_prime)
```

Parameters

`svd` Svd<double>

The MathNet.Numerics.LinearAlgebra.Factorization.Svd<T> object that represents the singular value decomposition of a matrix. This object contains three components:

- U: Matrix of left singular vectors (orthonormal).
- V: Matrix of right singular vectors (orthonormal), typically transposed as VT in MathNet.
- S: Diagonal matrix of singular values (non-negative real numbers).

`X` Matrix<double>

The original matrix for which the SVD was computed. This matrix is used in the sign-flipping algorithm to help determine the appropriate signs of the singular vectors.

`U_prime` Matrix<double>

Output matrix of left singular vectors (`U_prime`) with adjusted signs. This matrix is a copy of `U`, where the signs of its columns are flipped as necessary to ensure consistency between the left and right singular vectors.

`V_prime` Matrix<double>

Output matrix of right singular vectors (`V_prime`) with adjusted signs. Similar to `U_prime`, this matrix is derived from `V`, but with adjusted signs to maintain consistency with the left singular vectors.

Remarks

In the context of the Singular Value Decomposition (SVD), given a matrix `X`, it can be factored into: `X = U * S * V^T`, where:

- `U` is a matrix of left singular vectors (orthonormal columns).
- `S` is a diagonal matrix of singular values (non-negative, real numbers).
- `V^T` is the transpose of the matrix of right singular vectors (orthonormal rows).

However, the SVD is not unique because for any singular vector `u_k` and `v_k` (the k-th left and right singular vectors), flipping the sign of both (i.e., replacing `u_k` with `-u_k` and `v_k` with `-v_k`) leaves the product `X = U * S * V^T` unchanged. This means that without further constraints, there can be ambiguity in the signs of the singular vectors.

The `ComputeSignFlip` function eliminates this ambiguity by ensuring that the signs of the left (`U`) and right (`V`) singular vectors are consistent. The function evaluates the sign of each singular vector by analyzing its relationship with the original data matrix `X`. It adjusts the signs of the singular vectors to minimize inconsistencies, leading to a more interpretable and consistent SVD.

This process is useful when using SVD for dimensionality reduction, principal component analysis (PCA), and other applications where the interpretability and consistency of the singular vectors is important.

GetWeighedMatrix(Matrix<double>, Vector<double>, Vector<double>)

Applies row and column weights to the input matrix by adjusting each element based on the square roots of the corresponding row and column weights. This transformation is often used in dimensionality reduction techniques such as Principal Component Analysis (PCA) to balance the influence of each row and column while preserving the overall structure of the data.

```
public static Matrix<double> GetWeighedMatrix(Matrix<double> X, Vector<double>
rowWeights = null, Vector<double> colWeights = null)
```

Parameters

`X` Matrix<double>

The input matrix where rows represent observations and columns represent features.

`rowWeights` Vector<double>

A vector of weights corresponding to each row in the matrix. If not provided, uniform weights are assumed, meaning each row has an equal contribution. The weights are normalized to sum to 1.

`colWeights` Vector<double>

A vector of weights corresponding to each column in the matrix. If not provided, each column is assumed to have an equal weight of 1, meaning each feature has an equal contribution.

Returns

Matrix<double>

A new matrix where each element is scaled by the square root of its respective row and column weights.

Remarks

The square root of the weights is used to ensure that the overall weighting remains balanced. Multiplying by the square root of both row and column weights means that the scaling for each matrix element is proportional to the contribution of both its row and column without overly inflating or diminishing the influence of any single weight. This allows the matrix to be transformed in such a way that maintains relative distances between data points, which is crucial for algorithms like PCA, where geometric relationships (such as variance) are important.

Class PrincipalComponentAnalysis

Namespace: SharpMiner

Assembly: SharpMiner.dll

This class implements the Principal Component Analysis algortithm

```
public class PrincipalComponentAnalysis
```

Inheritance

object ← PrincipalComponentAnalysis

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PrincipalComponentAnalysis(Specs)

Initialize and compute PCA using SVD

```
public PrincipalComponentAnalysis(Specs specs)
```

Parameters

specs Specs

Properties

ColumnsResults

Get the features results

```
public FactorResults ColumnsResults { get; }
```

Property Value

FactorResults

Components

Get the list of computed components organised by rows and columns results

```
public Component[] Components { get; }
```

Property Value

Component[]

CumulativeExplainedVariance

Get the cumulative sum of explained variance

```
public double[] CumulativeExplainedVariance { get; }
```

Property Value

double[]

ExplainedVariance

Get the percentage of inertia explained by each component

```
public double[] ExplainedVariance { get; }
```

Property Value

double[]

RowsResults

Get the individuals results

```
public FactorResults RowsResults { get; }
```

Property Value

FactorResults

Svd

Get the result of the computed singular value decomposition

```
public Svd<double> Svd { get; }
```

Property Value

Svd<double>

Class Specs

Namespace: SharpMiner

Assembly: SharpMiner.dll

This class is used to parameterize a Computation

```
public class Specs
```

Inheritance

object ← Specs

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Specs(FactorMethod, int, Matrix<double>, double[], double[], bool)

Initialize a specification for computation

```
public Specs(FactorMethod factorMethod, int numberOfComponents, Matrix<double>
dataSet, double[] rowsWeights = null, double[] columnWeights = null, bool
centeredAndScale = true)
```

Parameters

`factorMethod` FactorMethod

`numberOfComponents` int

`dataSet` Matrix<double>

`rowsWeights` double[]

`columnWeights` double[]

`centeredAndScale` bool

Properties

CenteredAndScaledData

Compute centered and scaled data for computation

```
public Matrix<double> CenteredAndScaledData { get; }
```

Property Value

Matrix<double>

ColumnsWeights

Specify the weights attributed to each column in the dataset

```
public double[] ColumnsWeights { get; set; }
```

Property Value

double[]

DataSet

Specify the dataset on which the factor analysis method is to be computed

```
public Matrix<double> DataSet { get; set; }
```

Property Value

Matrix<double>

FactorMethod

Specify which factor analysis method to use

```
public FactorMethod FactorMethod { get; set; }
```

Property Value

FactorMethod

IsCenteredAndScaled

Specify if the dataset must be centered and scaled before computation

```
public bool IsCenteredAndScaled { get; set; }
```

Property Value

bool

NumberOfComponents

Number of components to be kept after the analysis

```
public int NumberOfComponents { get; set; }
```

Property Value

int

RowsWeights

Specify the weights attributed to each row in the dataset

```
public double[] RowsWeights { get; set; }
```

Property Value

double[]

WeighedMatrix

Represents the weighted matrix in a Principal Component Analysis (PCA), which is a scaled and reduced version of the original data matrix. This field holds the matrix after applying scaling and weighting transformations, used to standardize or normalize variables, making them comparable within the PCA framework.

```
public Matrix<double> WeighedMatrix { get; }
```

Property Value

Matrix<double>

Namespace SharpMiner.Graphics

Classes

Visualisations

This class contains methods to draw the results of the factor analysis on various types of charts

Class Visualisations

Namespace: SharpMiner.Graphics

Assembly: SharpMiner.dll

This class contains methods to draw the results of the factor analysis on various types of charts

```
public static class Visualisations
```

Inheritance

object ← Visualisations

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

CorrelationCircle((double, double)[], (sbyte, sbyte)?)

Represents the correlation circle in a Principal Component Analysis (PCA), which visualizes the correlations between the original variables and the principal components. This field contains data that helps illustrate how each variable is associated with the components, revealing patterns and relationships within the data in the context of the reduced PCA dimensions.

```
public static Plot CorrelationCircle((double, double)[] correlations, (sbyte,
sbyte)? dimensions = null)
```

Parameters

`correlations` (double, double)[]

`dimensions` (sbyte, sbyte)?

Returns

Plot

Remarks

The correlation circle is typically a graphical representation where:

- Each variable is displayed as a vector, with the coordinates on each axis representing its correlation with the respective principal component.
- The length and direction of each vector indicate both the strength and nature of the correlation, where longer vectors imply stronger correlations.
- Variables with vectors pointing in similar directions are positively correlated, while those in opposite directions are negatively correlated. Vectors closer to the origin suggest weaker correlations.

The data in this field is usually represented in matrix form, where rows correspond to variables and columns to the principal components. This visualization is valuable for interpreting the roles of variables in the principal component space and understanding underlying data structures.

ScreePlot(double[])

```
public static Plot ScreePlot(double[] explainedVariances)
```

Parameters

`explainedVariances` double[]

Returns

Plot