Distance Matrices and Multidimensional Scaling

Allyson Hahn

Northern Illinois University

Outline

Introduction

scipy.spatial.distance

Background

Definition

Let X be a set. A function $d : X \times X \to [0, \infty)$ is called a **metric** on X if it satisfies the following:

(a) $d(x, y) = 0$ if and only if $x = y$ in X

(b) $d(x, y) = d(y, x)$ for all $x, y \in X$

(c) $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in X$.

We refer to the pair (X, d) as a metric space.

Examples: Euclidean and Minkowski

Euclidean Metric

The **Euclidean metric**, denoted d, on $\mathbb{R}^n$ is defined as follows:

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{\frac{1}{2}}$$

where $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$.

To calculate the Euclidean metric we can use

`scipy.spacial.distance.euclidean(`**x**,**y**`)`

Example

The **Euclidean metric**, denoted d, on $\mathbb{R}^n$ is defined as follows:

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{\frac{1}{2}}$$

where $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$.

Example: Let $x = (1, 2, 3)$ and $y = (4, 5, 8)$. Compute $d(x, y)$.

The City Block Metric

The **City Block** metric denoted d, on $\mathbb{R}^n$ is defined as follows:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|$$

where $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$.

To calculate the City Block metric we can use

`scipy.spacial.distance.cityblock`(**x**,**y**)

Minkowski Metric

The **Minkowski** metric on $\mathbb{R}^n$ is the metric induced by the *p*-norm, so

$$d_p(x, y) = \|\mathbf{x} - \mathbf{y}\|_p = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

where $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$.

To calculate the Minkowski metric we can use

`scipy.spacial.distance.minkowski(`**x**,**y**, p)

What are distance matrices?

Distance Matrix

A **distance matrix** is a square, symmetric matrix with zeros along the diagonal containing the distances, taken pairwise, between the elements of a set. They are sometimes called dissimilarity matrices.

To compute a distance matrix given a matrix X we can use

```
scipy.spacial.distance.cdist(X,X, metric)
```

Let's look at an example.

Example

Example: Using Euclidean distance, compute the distance matrix given

$$X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}.$$

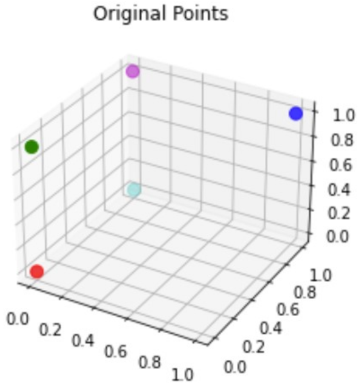
What is Multidimensional Scaling?

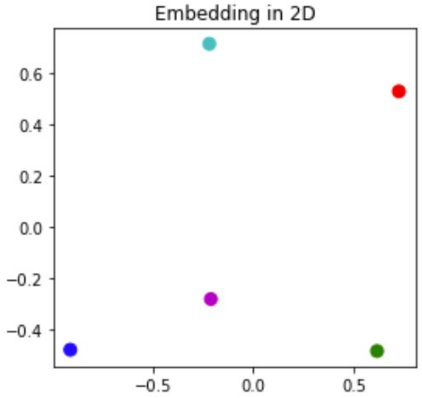
Definition

The process of taking a distance matrix containing the distances between each pair of objects in a set, and placing each object into a lower-dimensional representation such that the between-object distances are preserved as well as possible is know as **Multidimensional Scaling (MDS)**.

Our Goal: “Squeeze” a high-dimensional set of points into a smaller number of dimensions, typically 2 or 3.

What is MDS? (cont.)





Types of MDS

There are three types of multidimensional scaling:

- Classical MDS
- Metric MDS
- Non-Metric MDS

Classical MDS

Definition

The process of **Classical Multidimensional Scaling** take as an input a matrix containing dissimilarities of pairs of objects and outputs a coordinate matrix whose configuration minimizes a loss function called **strain**.

Classical MDS Process

Let D be an $n \times n$ Euclidean distance matrix and let $k \in \mathbb{N}$ be the dimension we are reducing to. This process is forming an $n \times n$ Gram matrix B corresponding to the squared distance matrix.

1. Let D_2 be the squared distance matrix, where we square component-wise.
2. Set $B = -\frac{1}{2} C D_2 C$, where $C = I_n - \frac{1}{n} J J^T$ and J is an $n \times 1$ vector of all ones.
3. Take the top k eigenvalues of B to be $\lambda_1, \ldots, \lambda_k$ and the corresponding eigenvectors $e_1, \ldots, e_k$.

Classical MDS Process (cont.)

4. Set the following:

$$L = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_k \end{bmatrix}$$

   and

$$V = \begin{bmatrix} e_1 & e_2 & \cdots & e_k \end{bmatrix}.$$

5. Then, the classical MDS solution is

$$X = VL^{1/2}.$$

The rows of X are the coordinate points in $\mathbb{R}^k$.

Frobenius Norm

Definition

Let A be an $m \times n$ matrix. Then, the **Frobenius** norm is defined as

$$\|A\|_F = \sqrt{\sum_{j=1}^{n} \sum_{i=1}^{m} |a_{ij}|^2}.$$

Strain

The Gram matrix of X is XX^T. We want to find a way to minimize the distance between B and XX^T. To do so we use the Frobenius norm. This motivates the need for the Strain(X) function.

Definition

Given B and X as previously defined, the **strain** of X is given by

$$Strain(X) = \sqrt{\frac{\|B - XX^T\|}{\|B\|}} = \left(\frac{\sum_{i,j}(b_{ij} - x_i x_j^T)^2}{\sum_{i,j} b_{ij}^2} \right)^{1/2}.$$

It follows that $\|B - XX^T\| = \lambda_{k+1}$, so we could substitute to simplify.

The most accurate X is found by minimizing the strain function, or simply minimizing $\|B - XX^T\|$.

Classical MDS Process (cont.)

Theorem

A distance matrix D is a Euclidean distance matrix if and only if the corresponding centered matrix B is positive semi-definite.

Important: Classical MDS assumes the input matrix is a Euclidean distance matrix.

Question: What do we do if D is a non-Euclidean distance matrix?

Metric MDS

An alternative to minimizing strain is minimizing what is known as stress.

Definition

A superset of classical MDS which generalizes the optimization procedure to a variety of input matrices of known distances is known as **Metric Multidimensional Scaling**. The process takes as an input a matrix containing dissimilarities of pairs of objects and outputs a coordinate matrix whose configuration minimizes a loss function called **stress** using the process of **stress majorization**.

Stress

Metric MDS minimizes the loss function stress rather than strain.

Definition

Let D be an $n \times n$ distance matrix and $D(X)$ an $n \times n$ distance matrix of a $n \times k$ matrix X. Then **stress** is defined as

$$Stress_D(X) = \|D - D(X)\|.$$

The **normalized stress** is

$$NormStress_D(X) = \frac{\|D - D(X)\|}{\|D\|}.$$

SMACOF

- To minimize stress we use a procedure called stress majorization.
- Alternatively we could define stress as

$$\sigma(X) = \sum_{i<j\leq n} w_{ij}(d_{ij}(X) - \delta_{ij})^2,$$

  where $w_{ij} \geq 0$ is a weight measurement, $d_{ij}(X)$ is the euclidean distance between i and j and δ_{ij} is the ideal distance between the points.
- We can use the process of **Scaling by Majorizing a Complicated Function (SMACOF)**.
- SMACOF is an iterative majorization method which at each step minimizes a simple convex function which bounds σ from above and touches the surface of σ.

Necessary Tools

- To perform MDS we will need to use `sklearn.manifold.MDS`.
- The parameters that we will be concerned with for this class are the follwoing:
  - `n_components`: integer
  - `metric`: boolean
  - `n_init`: integer, Number of times the SMACOF algorithm will be run with different initializations. The final results will be the best output of the runs, determined by the run with the smallest final stress.
  - `max_iter`: integer, Maximum number of iterations of the SMACOF algorithm for a single run.
  - `dissimilarity`: {'euclidean','precomputed'}
- For example

```
 sklearn.manifold.MDS(n_components=2, metric=True,
n_init=4, max_init=300, dissimilarity=‘euclidean’)
```

- In the class we will use the following method:

```
fit_transform(X)
```

This takes in a matrix and returns a matrix of the lower-representations for the points.

Necessary Tools (cont.)

- This class does have an attribute which is supposed to return the value of the stress. However, there is an issue.
- The package minimizes $\frac{1}{2} Stress(X)^2$.
- We seek a meaningful value.
- Resolution: `np.sqrt(2*stress)/np.linalg.norm(D)`

Necessary Tools (cont.)

- The `sklearn.manifold` module also has the following function

```
sklearn.manifold.smacof(dissimilarities, metric=True,
  n_components=2, init=None, n_init=8, max_iter=300)
```

  which also computes MDS using SMACOF.
- In short, this function follows these steps:
  1. Set an initial start configuration, randomly or not
  2. Compute the stress
  3. Compute the Guttman Transform
  4. Iterate 2 and 3 until convergence.

Example 1

Example 2–Improving Accuracy

Example 3: Application (Maps)

Example 4: Tetrahedron

Bibliography

Wikimedia Foundation. (2022, October 19). Multidimensional scaling. Wikipedia. Retrieved October 26, 2022, from https://en.wikipedia.org/wiki/Multidimensional_scaling

Jung, S. (2013). Department of Statistics — University of Pittsburgh. Retrieved October 26, 2022, from https://www.stat.pitt.edu/sungkyu/course/2221Fall13/lec8mds_combined.pdf

Chapter 435 Multidimensional Scaling - NCSS. NCSS. (n.d.). Retrieved October 26, 2022, from https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Multidimensional_Scaling.pdf

Multidimensional scaling. (n.d.). Retrieved October 26, 2022, from http://www.analytictech.com/borgatti/mds.htm

Bibliography

API reference. scikit. (n.d.). Retrieved October 26, 2022, from https://scikit-learn.org/stable/modules/classes.htmlmodule-sklearn.manifold

Source code for pyseer.cmdscale. pyseer.cmdscale - pyseer 1.3.10 documentation. (n.d.). Retrieved October 26, 2022, from https://pyseer.readthedocs.io/en/master/_modules/pyseer/cmdscale.html