

Algorithms

Dimensionality reduction

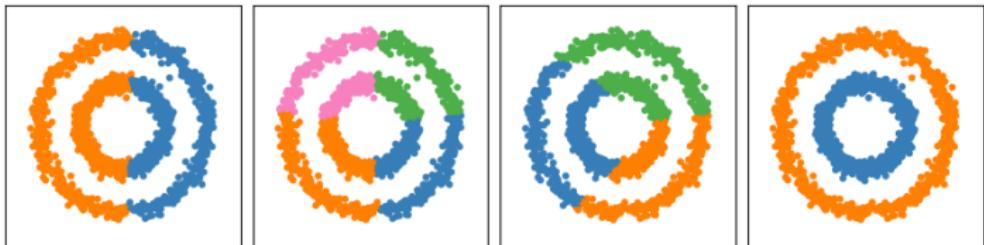
Emanuele Rodolà
rodola@di.uniroma1.it



Clustering

Input: data points $\{\mathbf{x}_i\}$ for $i = 1 \dots n$.

Output: labeled data points $\{\mathbf{x}_{i,\ell}\}$ with $\ell = 1 \dots k$.

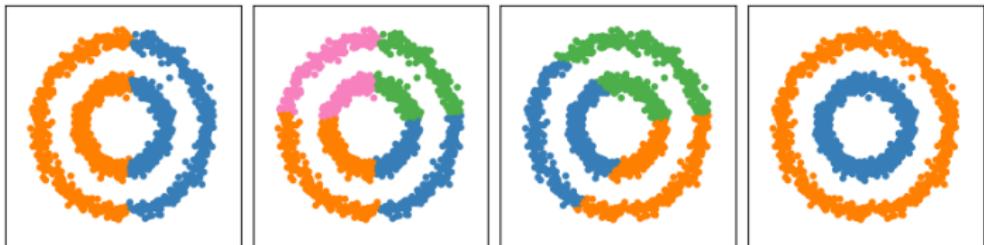


Clustering algorithms compute **distances** among the given points.

Clustering

Input: data points $\{\mathbf{x}_i\}$ for $i = 1 \dots n$.

Output: labeled data points $\{\mathbf{x}_{i,\ell}\}$ with $\ell = 1 \dots k$.

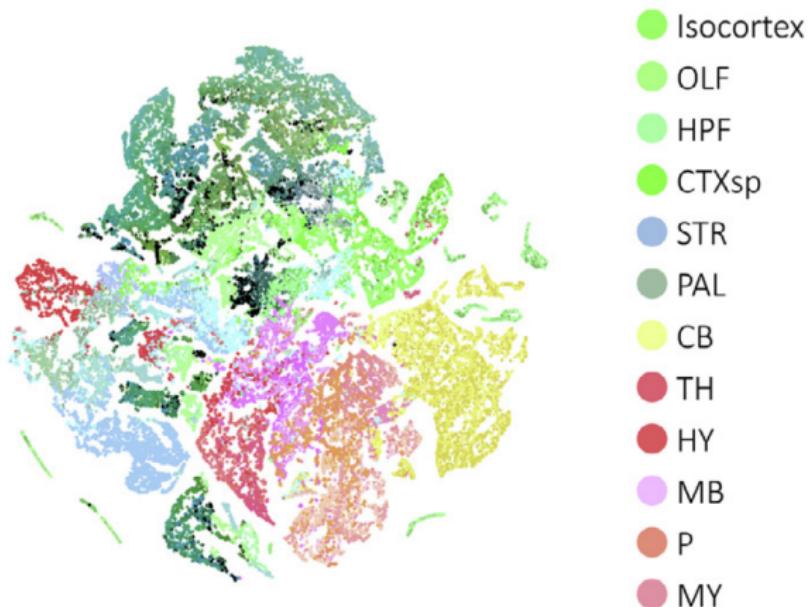


Clustering algorithms compute **distances** among the given points.

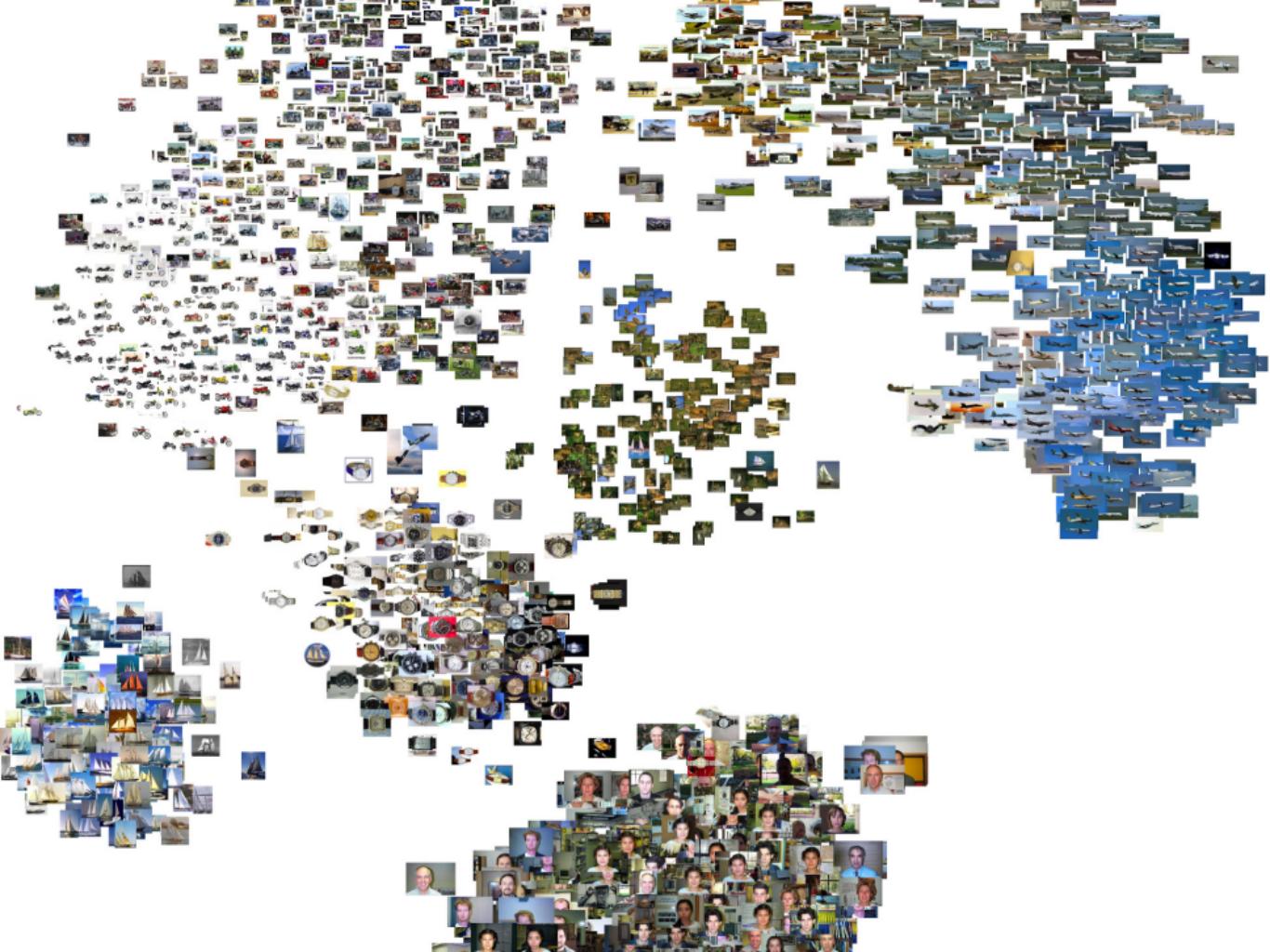
What if we only have the distances, and not the points?

Example

Based on **similarities** between gene expression levels (rows of the expression matrix):



Mahfouz et al, "Visualizing the spatial gene expression organization in the brain through non-linear similarity embeddings", Methods 2015



Distances

A **distance** function (or **metric**) must satisfy the following properties for every pair of points x, y :

- $d(x, y) \geq 0$

Distances

A **distance** function (or **metric**) must satisfy the following properties for every pair of points x, y :

- $d(x, y) \geq 0$
- $d(x, y) = 0 \Leftrightarrow x = y$ (identity of indiscernibles)

Distances

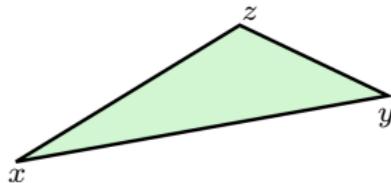
A **distance** function (or **metric**) must satisfy the following properties for every pair of points x, y :

- $d(x, y) \geq 0$
- $d(x, y) = 0 \Leftrightarrow x = y$ (identity of indiscernibles)
- $d(x, y) = d(y, x)$ (symmetry)

Distances

A **distance** function (or **metric**) must satisfy the following properties for every pair of points x, y :

- $d(x, y) \geq 0$
- $d(x, y) = 0 \Leftrightarrow x = y$ (identity of indiscernibles)
- $d(x, y) = d(y, x)$ (symmetry)
- $d(x, y) \leq d(x, z) + d(z, y)$ for any x, y, z (triangle inequality)



A cartographer's problem

Consider the following:



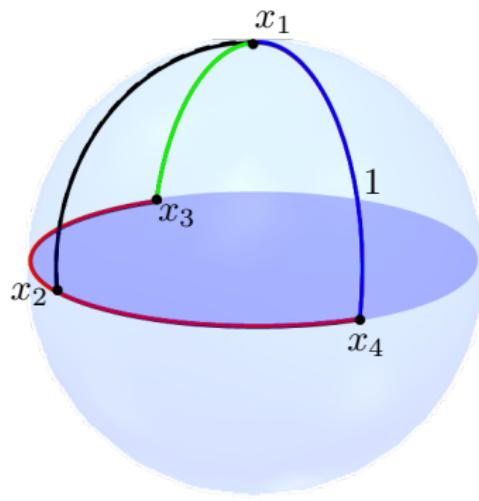
A cartographer's problem

Consider the following:



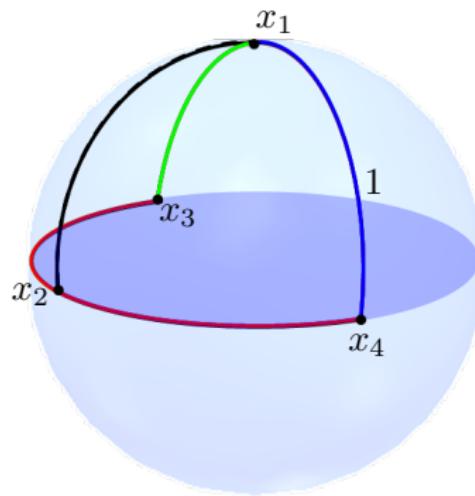
The set of points on the plane is called an [embedding](#) of the sphere in \mathbb{R}^2 .
Any approximate solution introduces [metric distortion](#).

Non-embeddability of the sphere



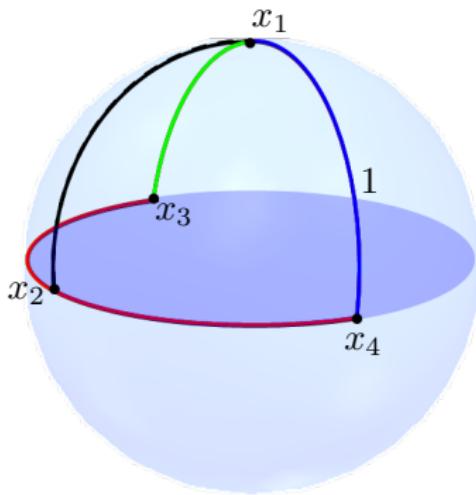
- Consider the triangle $\Delta(x_1, x_3, x_4)$ on the plane

Non-embeddability of the sphere



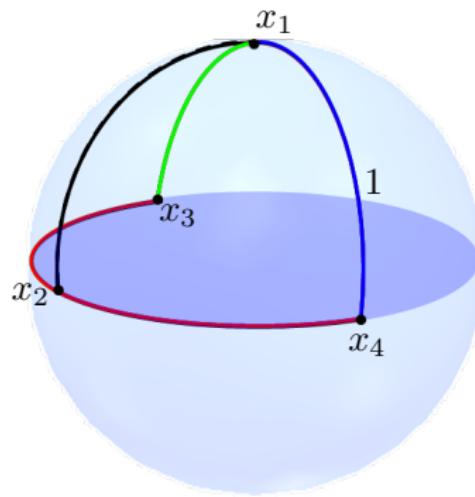
- Consider the triangle $\Delta(x_1, x_3, x_4)$ on the plane \Rightarrow collinear!

Non-embeddability of the sphere



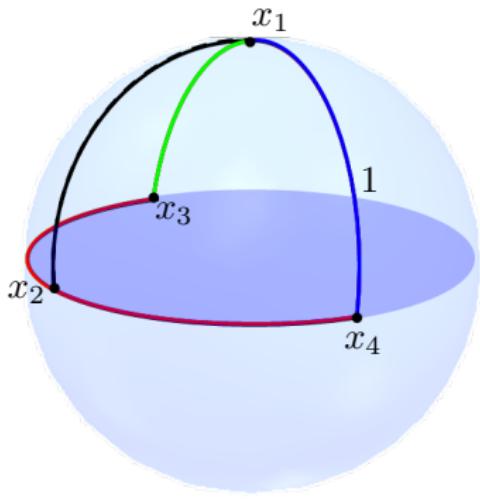
- Consider the triangle $\Delta(x_1, x_3, x_4)$ on the plane \Rightarrow collinear!
- Consider the triangle $\Delta(x_2, x_3, x_4)$ on the plane

Non-embeddability of the sphere



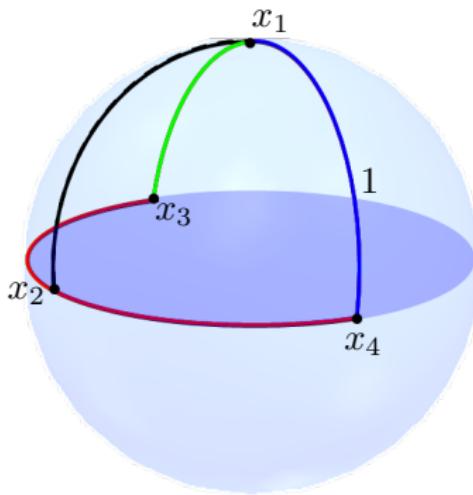
- Consider the triangle $\Delta(x_1, x_3, x_4)$ on the plane \Rightarrow collinear!
- Consider the triangle $\Delta(x_2, x_3, x_4)$ on the plane \Rightarrow collinear!

Non-embeddability of the sphere



- Consider the triangle $\Delta(x_1, x_3, x_4)$ on the plane \Rightarrow collinear!
- Consider the triangle $\Delta(x_2, x_3, x_4)$ on the plane \Rightarrow collinear!
- Then $x_1 = x_2$ on the plane contradicts $d(x_1, x_2) = 1$ on the sphere.

Non-embeddability of the sphere

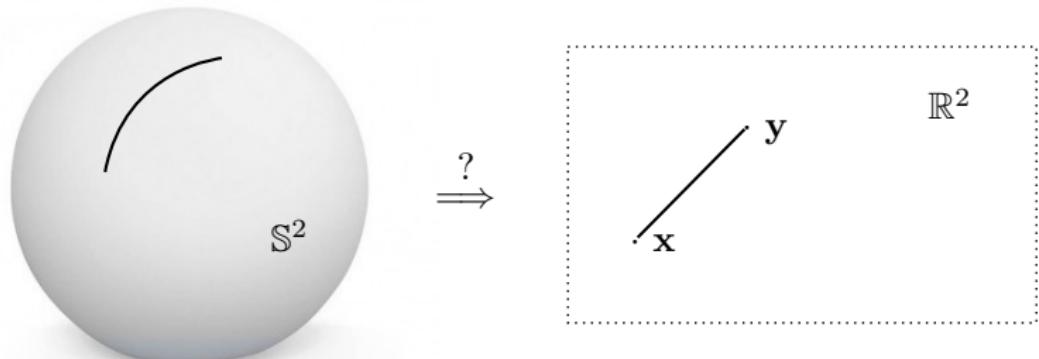


- Consider the triangle $\Delta(x_1, x_3, x_4)$ on the plane \Rightarrow collinear!
- Consider the triangle $\Delta(x_2, x_3, x_4)$ on the plane \Rightarrow collinear!
- Then $x_1 = x_2$ on the plane contradicts $d(x_1, x_2) = 1$ on the sphere.
 \Rightarrow The sphere cannot be embedded into \mathbb{R}^k for any k .

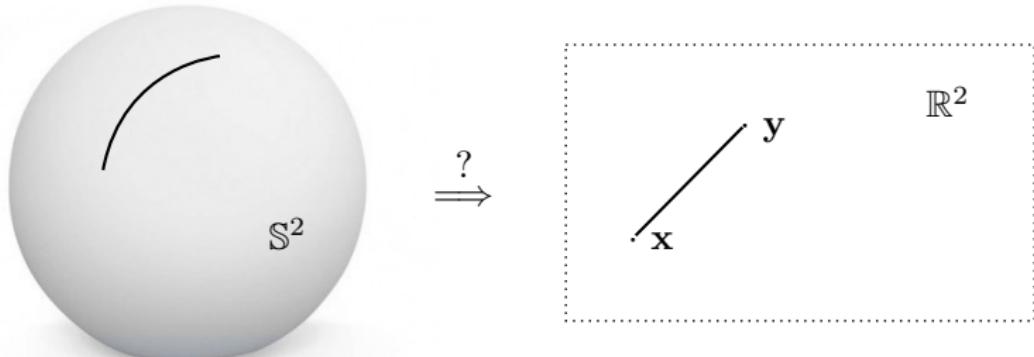
A cartographer's solution



Minimum-distortion embeddings



Minimum-distortion embeddings



Given $d(x, y)$, find $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$ such that :

$$d(x, y) \approx \|\mathbf{x} - \mathbf{y}\|_2 \text{ for all } x, y \in \mathbb{S}^2$$

Stress minimization

This can be cast as an **optimization** problem:

$$\min_{\{\mathbf{x}\}} \sum_i \sum_j |\mathbf{D}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|$$

Stress minimization

This can be cast as an optimization problem:

$$\min_{\{\mathbf{x}\}} \sum_i \sum_j |\mathbf{D}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|$$

For any given set of points $\{\mathbf{x}\}$, the stress measures how well those points match the given distances. We look for the set of minimum stress.

Stress minimization

This can be cast as an optimization problem:

$$\min_{\{\mathbf{x}\}} \sum_i \sum_j |\mathbf{D}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|$$

For any given set of points $\{\mathbf{x}\}$, the stress measures how well those points match the given distances. We look for the set of minimum stress.

The dimensionality d of the points $\mathbf{x} \in \mathbb{R}^d$ is a parameter chosen by us.

Stress minimization

This can be cast as an optimization problem:

$$\min_{\{\mathbf{x}\}} \sum_i \sum_j |\mathbf{D}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|$$

For any given set of points $\{\mathbf{x}\}$, the stress measures how well those points match the given distances. We look for the set of minimum stress.

Can be solved by differentiable programming techniques. Consider:

$$\min_{\mathbf{x} \in \mathbb{R}^k} f(\mathbf{x})$$

Stress minimization

This can be cast as an optimization problem:

$$\min_{\{\mathbf{x}\}} \sum_i \sum_j |\mathbf{D}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|$$

For any given set of points $\{\mathbf{x}\}$, the stress measures how well those points match the given distances. We look for the set of minimum stress.

Can be solved by differentiable programming techniques. Consider:

$$\min_{\mathbf{x} \in \mathbb{R}^k} f(\mathbf{x})$$

Gradient descent algorithm:

- Choose a starting point $\mathbf{x}^{(0)}$ (initialization)

Stress minimization

This can be cast as an optimization problem:

$$\min_{\{\mathbf{x}\}} \sum_i \sum_j |\mathbf{D}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|$$

For any given set of points $\{\mathbf{x}\}$, the stress measures how well those points match the given distances. We look for the set of minimum stress.

Can be solved by differentiable programming techniques. Consider:

$$\min_{\mathbf{x} \in \mathbb{R}^k} f(\mathbf{x})$$

Gradient descent algorithm:

- Choose a starting point $\mathbf{x}^{(0)}$ (initialization)
- Iterate: $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$

Stress minimization

This can be cast as an optimization problem:

$$\min_{\{\mathbf{x}\}} \sum_i \sum_j |\mathbf{D}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|$$

For any given set of points $\{\mathbf{x}\}$, the stress measures how well those points match the given distances. We look for the set of minimum stress.

Can be solved by differentiable programming techniques. Consider:

$$\min_{\mathbf{x} \in \mathbb{R}^k} f(\mathbf{x})$$

Gradient descent algorithm:

- Choose a starting point $\mathbf{x}^{(0)}$ (initialization)
- Iterate: $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$
- Stop when some convergence criterion is satisfied

Stress minimization

This can be cast as an optimization problem:

$$\min_{\{\mathbf{x}\}} \sum_i \sum_j |\mathbf{D}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|$$

For any given set of points $\{\mathbf{x}\}$, the stress measures how well those points match the given distances. We look for the set of minimum stress.

Can be solved by differentiable programming techniques. Consider:

$$\min_{\mathbf{x} \in \mathbb{R}^k} f(\mathbf{x})$$

Gradient descent algorithm:

- Choose a starting point $\mathbf{x}^{(0)}$ (initialization)
- Iterate: $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$
- Stop when some convergence criterion is satisfied

$$f(\mathbf{x}^{(0)}) \geq f(\mathbf{x}^{(1)}) \geq f(\mathbf{x}^{(2)}) \dots$$

Multidimensional scaling

Why “stress”?

Problems of this sort started appearing in psychology several decades ago, and are also called **multidimensional scaling (MDS)** problems.

Empirical procedures of several diverse kinds have this in common: they start with a fixed set of entities and determine, for every pair of these, a number reflecting how closely the two entities are related psychologically. The nature of the psychological relation depends upon the nature of the entities. If the entities are all stimuli or all responses, we are inclined to think of the relation as one of similarity. A somewhat more objective (though less intuitive) characterization of such a relation, perhaps, is that of substitutability. The statement that stimulus A is more similar to B than to C , for example, could be interpreted to say that the psychological (or behavioral) consequences are greater when C , rather than B , is substituted for A . From this standpoint a natural procedure for determining similarities of stimuli or responses is by recording substitution errors during identification learning [2, 7, 12, 14, 17, 18]. In addition, though, disjunctive reaction time and sorting time have also been proposed as measures of psychological similarity [20]. Finally, of course, individuals have sometimes been instructed simply to rate each pair of stimuli, directly, on a scale of apparent similarity [1, 6]. The notion of similarity is not necessarily restricted to stimuli or responses (in the narrow sense of these words), however. Serviceable measures of similarity may also be found for concepts, attitudes, personality structures, or even social institutions, political systems, and the like.

Shepard, "The analysis of proximities: Multidimensional scaling with an unknown distance function". Psychometrika 27(2), 1962

Stress minimization

$$\min_{\{\mathbf{x}\}} \sum_i \sum_j |\mathbf{D}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|$$

Stress minimization

$$\min_{\{\mathbf{x}\}} \sum_i \sum_j |\mathbf{D}_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2|$$

In Python available through the [scikit-learn](#) library, together with other algorithms such as k -means, random forests, etc.



For stress minimization see:

<https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html>

You are encouraged to experiment with the library!

scikit-learn

Install User Guide API Examples More ▾

scikit-learn

Machine Learning in Python

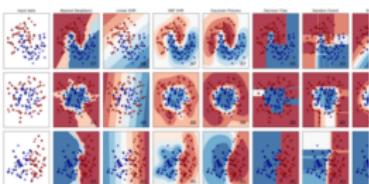
Getting Started What's New in 0.22.2 GitHub

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



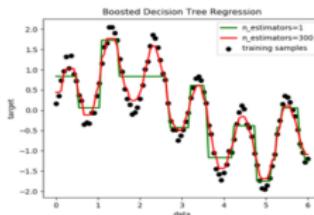
Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



Examples

Similarity

In practice, often we are not given distances but we are given **similarities**.

For example:

$$d(x, y) \Rightarrow e^{-d^2(x, y)}$$

Similarity

In practice, often we are not given distances but we are given **similarities**.

For example:

$$d(x, y) \Rightarrow e^{-d^2(x, y)}$$

Similarities do **not** satisfy the metric properties.

Similarity

In practice, often we are not given distances but we are given **similarities**.

For example:

$$d(x, y) \Rightarrow e^{-d^2(x, y)}$$

Similarities do **not** satisfy the metric properties.

Multidimensional scaling is just one of many other Euclidean embedding techniques: T-SNE, eigenmaps, locally linear embeddings, etc.

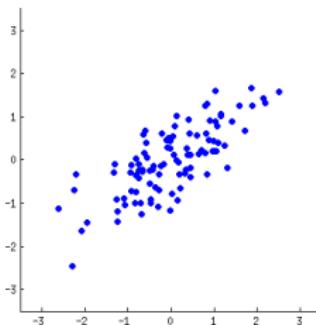
You can experiment with others in scikit-learn:

<https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE>

Dimensionality reduction: another perspective

Let us be given n data points stored in matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$:

$$\mathbf{X}^\top = \begin{pmatrix} & \mathbf{x}_1^\top & \\ & \vdots & \\ & \mathbf{x}_n^\top & \end{pmatrix}$$

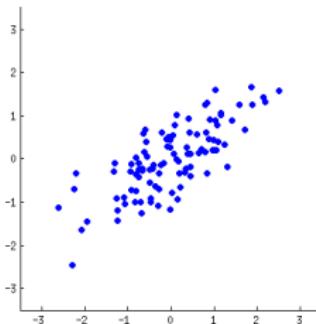


Dimensionality reduction: another perspective

Let us be given n data points stored in matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$:

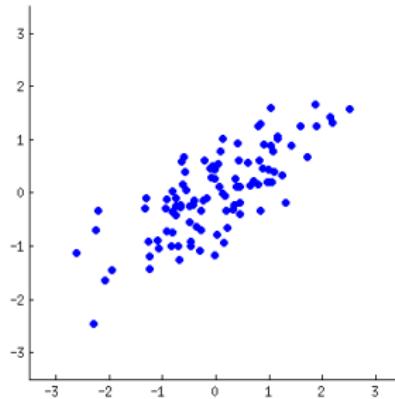
$$\mathbf{X}^\top = \begin{pmatrix} - & \mathbf{x}_1^\top & - \\ \vdots & & \vdots \\ - & \mathbf{x}_n^\top & - \end{pmatrix} \approx \begin{pmatrix} - & \tilde{\mathbf{x}}_1^\top & - \\ \vdots & & \vdots \\ - & \tilde{\mathbf{x}}_n^\top & - \end{pmatrix} = \tilde{\mathbf{X}}^\top$$

We want to replace them with a **lower-dimensional** approximation $\tilde{\mathbf{X}} \in \mathbb{R}^{k \times n}$, with $k \ll d$.



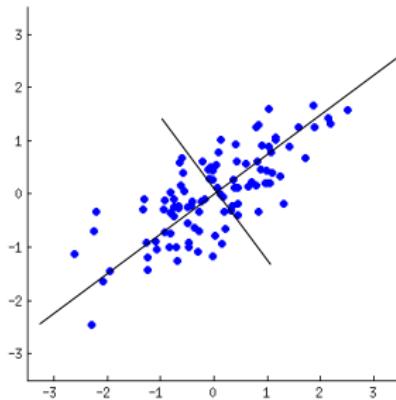
Principal component analysis (PCA)

Regard our data as n points in \mathbb{R}^d :



Principal component analysis (PCA)

Regard our data as n points in \mathbb{R}^d :

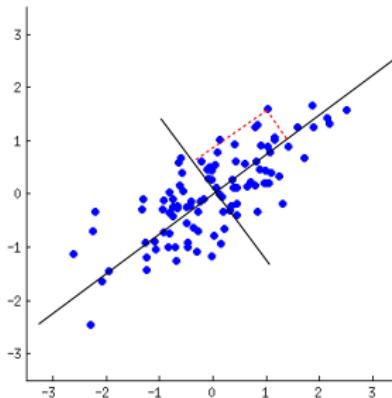


Overall idea:

- Find $k \leq d$ **orthogonal directions** with the most variance.
These span a k -dimensional **subspace** of the data.

Principal component analysis (PCA)

Regard our data as n points in \mathbb{R}^d :



Overall idea:

- Find $k \leq d$ orthogonal directions with the most variance.
These span a k -dimensional subspace of the data.
- Project all the data points onto these directions.
This is lossy, but can be done with the smallest possible error.

Principal component analysis (PCA)

We seek the **direction w** that:

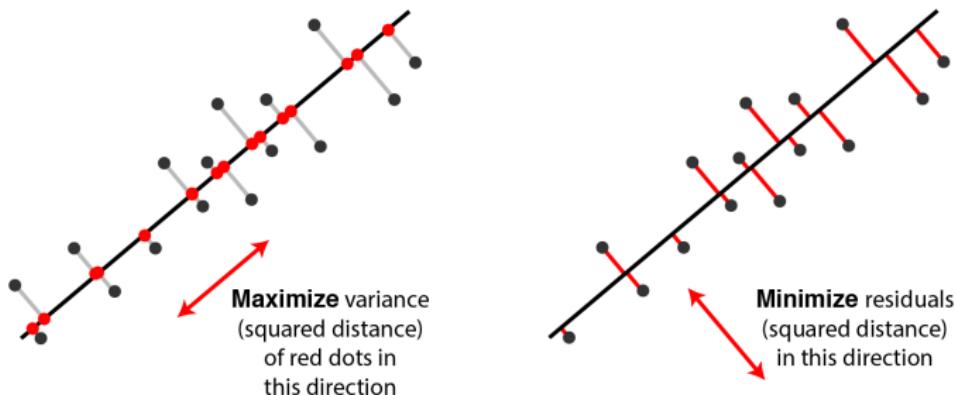
- Minimizes the **projection/reconstruction error**.
- Maximizes the **variance** of the projected data.

Principal component analysis (PCA)

We seek the **direction w** that:

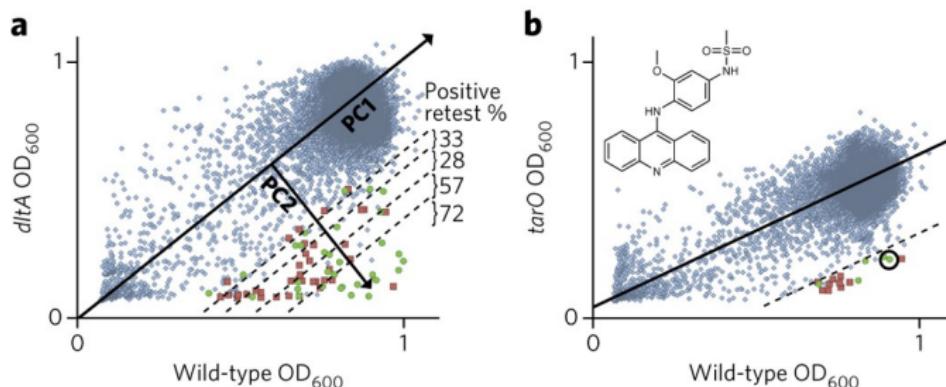
- Minimizes the **projection/reconstruction error**.
- Maximizes the **variance** of the projected data.

Principal component analysis (PCA)



Examples

PCA is widely used in computational biology and bioinformatics.



Suggested short reading:

Ringnér, “What is principal component analysis?”, Nature Biotech. 2008
[\(click here\)](#)