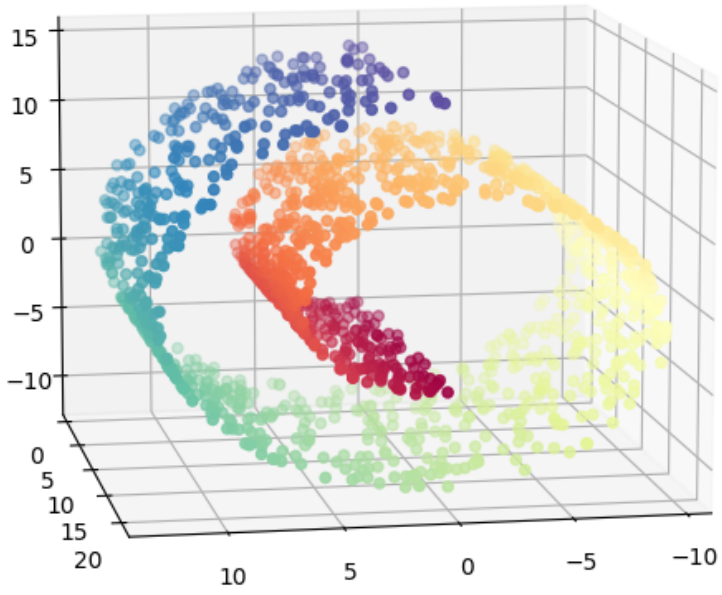
Dimensionality Reduction Algorithms

Dimensionality Reduction

- Sometimes the input data available have a large number of features, which makes the application of Machine Learning algorithms slower
- It is reasonable to reduce the number of features while preserving some of the properties of the data
- The main motivations behind dimensionality reduction algorithms are:
  - Compressing the original data as a pre-processing step makes their analysis computationally faster;
  - If the elements are mapped into a 2- or 3-dimensional space, it becomes possible to visualize them for exploratory analysis
  - Generate a smaller set of features that are more effective or more explanatory for the analysis to be performed

Dimensionality Reduction

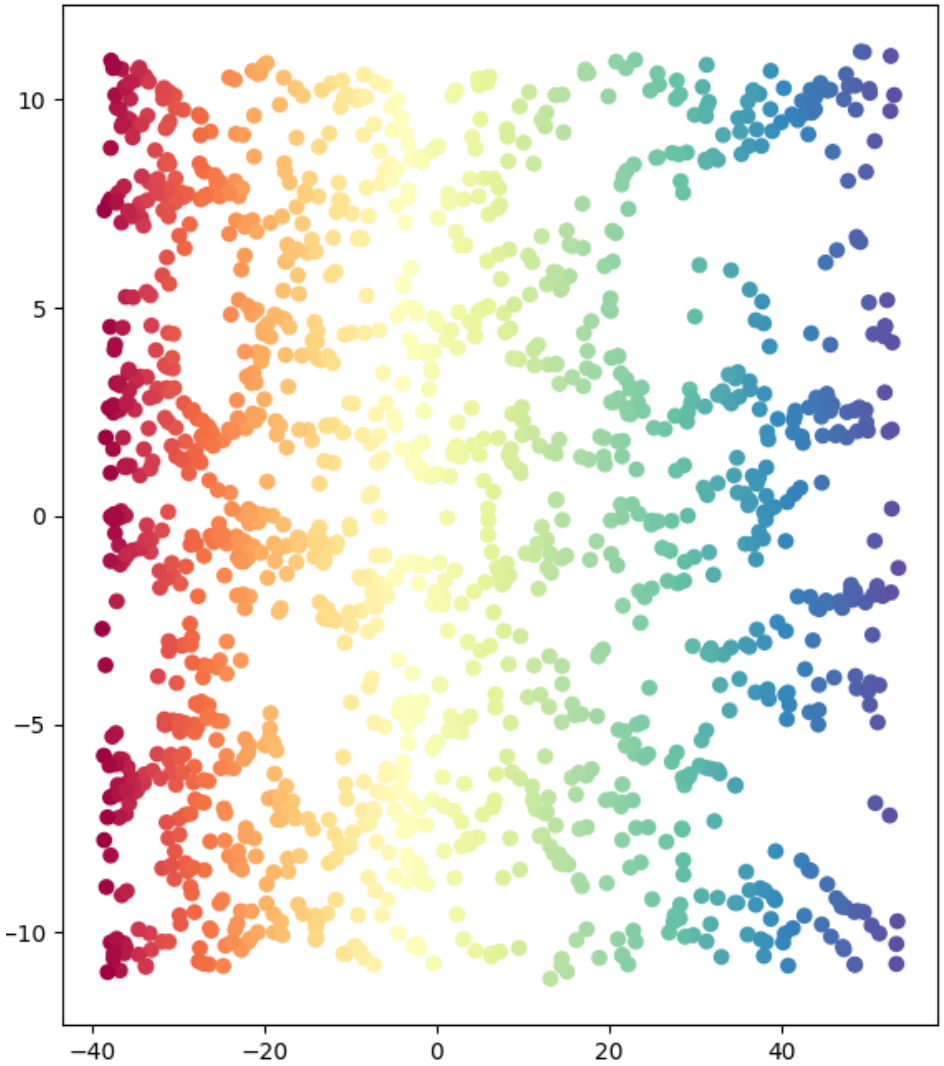
- Consider the following example:



- The elements in $\mathbb{R}^3$ are distributed along a curved surface
- If we want to apply a machine learning algorithm based on Euclidean distance to these data, the representation in $\mathbb{R}^3$ may be misleading
- Since a surface is defined by two parameters, we can project the data into a two-dimensional space without losing the information we are interested in

Dimensionality Reduction

- Consider the following example:



- Using this representation with fewer features, we can apply machine learning algorithms based on Euclidean distance while losing little information

Dimensionality Reduction

- We can formalize dimensionality reduction as follows:
- Given a sample of elements $\mathcal{X} = \{x_1, \ldots x_m\} \subseteq \mathbb{R}^n$, we can consider the i-th element either as the vector x_i or as the i-th column of the matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$
- Dimensionality reduction techniques seek a k-dimensional representation of the data, with $k << n$, namely a matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{k \times m}$ whose columns are in some sense faithful to the original n-dimensional representation
- We will consider some of the main dimensionality reduction techniques:
  - Principal Component Analysis (PCA)
  - Manifold learning algorithms
  - Algorithms based on random projections (RP)

Principal Component Analysis

- Let us fix $k \in [n]$ and consider $\mathbf{X}$ a data matrix with zero mean, i.e.

$$\sum_{i=1}^{m} x_i = 0$$

- Let $\mathcal{P}_k$ be the set of n-dimensional orthogonal projection matrices of rank k
- The goal of PCA is to find the element $\mathbf{P}^* \in \mathcal{P}_k$ that minimizes the reconstruction error:

$$\mathbf{P}^* = \arg\min_{\mathbf{P} \in \mathcal{P}_k} \|\mathbf{PX} - \mathbf{X}\|_F^2$$

- The following result shows that the PCA solution coincides with the projection of each element onto the first k singular vectors of the covariance matrix

$$\mathbf{C} = \frac{1}{m}\mathbf{X}\mathbf{X}^\top$$

Principal Component Analysis

Theorem

Let $\mathbf{P}^*$ *be the PCA solution. Then*

$$\mathbf{P}^* = \mathbf{U}_k \mathbf{U}_k^\top,$$

where $\mathbf{U}_k \in \mathbb{R}^{n \times k}$ *is the matrix formed by the first* k *singular vectors of the covariance matrix*

$$\mathbf{C} = \frac{1}{m} \mathbf{X}\mathbf{X}^\top.$$

Moreover, the k*-dimensional representation of* $\mathbf{X}$ *is given by*

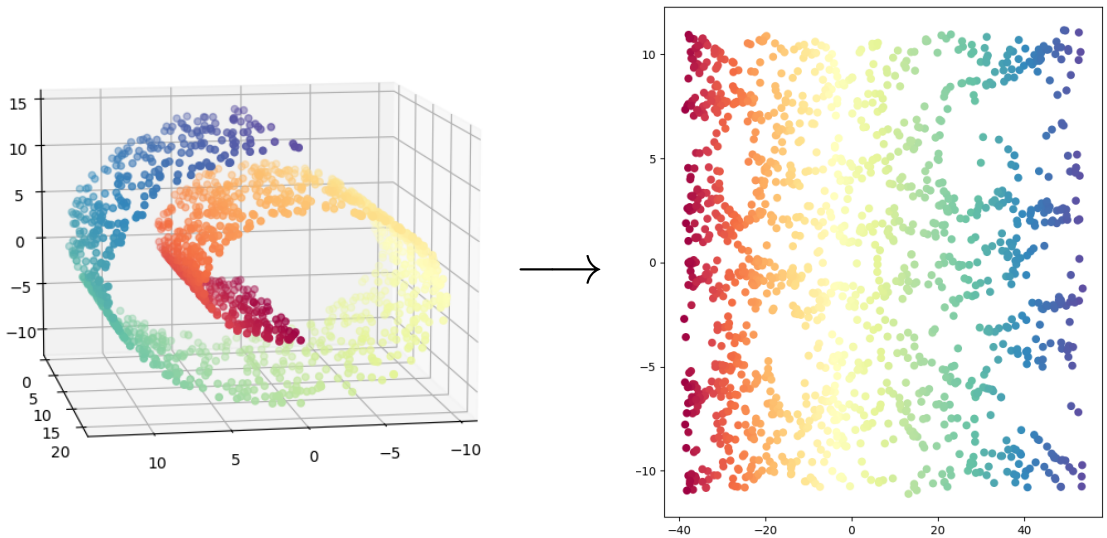
$$\tilde{\mathbf{X}} = \mathbf{U}_k^\top \mathbf{X}.$$

Principal Component Analysis

- By definition of the covariance matrix, the largest singular vectors of $\mathbf{C}$ are the directions along which the variance of the data is greatest, and the corresponding singular values are equal to those variances
- Therefore, we can interpret PCA as the projection onto the subspace with the greatest variance in the data
- Using this interpretation, the first principal component is given by the projection onto the direction of maximum variance
- Similarly, the i-th principal component is given by the projection onto the i-th direction of maximum variance under the condition that it is orthogonal to the previous $i-1$ directions of maximum variance

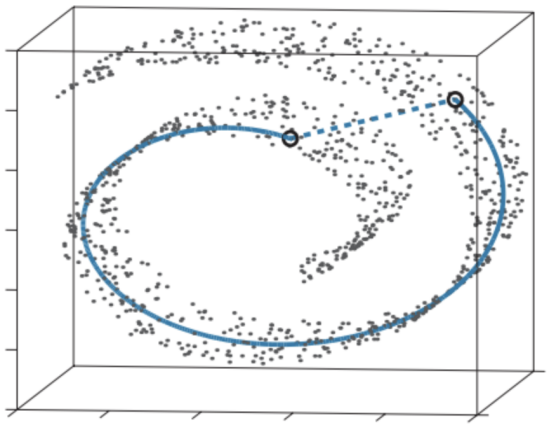
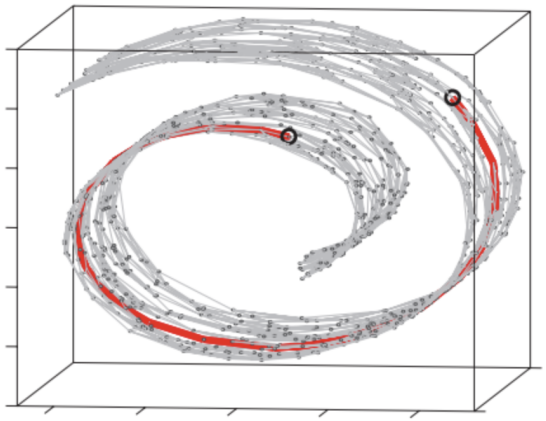
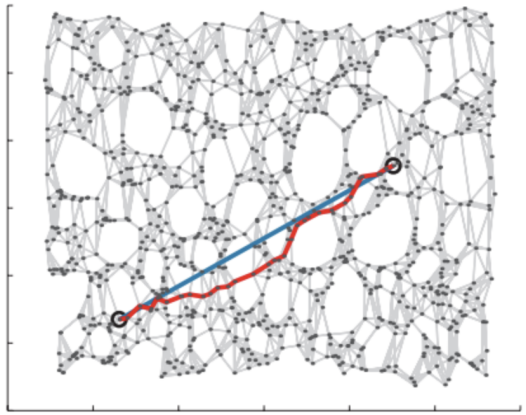
Manifold Learning

- Manifold Learning refers to those dimensionality reduction algorithms that assume the input data lie on a submanifold $\mathcal{M}$ embedded in the ambient space $\mathbb{R}^n$
- These are non-linear dimensionality reduction algorithms that attempt to understand the non-linear relationships between the components of an input element in order to find a set of features representing the same elements



Isomap

- The Isomap algorithm attempts to extract a low-dimensional representation of the data that best preserves the pairwise distances between the input points measured using the geodesic distance along the submanifold on which the points lie

Isomap

- We can summarize Isomap as follows:
  - For each dataset element x_i, we find the t nearest points (T-nn) $x_1^i, \ldots, x_t^i$ according to the Euclidean distance, and we construct a weighted graph $\mathcal{G}$ whose vertices are the input elements and where an edge is generated only between x_i and x_j^i
  - To approximate the geodesic distance on the graph $\mathcal{G}$, we initialize

$$d_{\mathcal{G}}(x_i, x_j) = d_X(x_i, x_j)$$

    if there exists an edge between x_i and x_j, and

$$d_{\mathcal{G}}(x_i, x_j) = \infty$$

    otherwise
  - For every $s = 1, \ldots, m$, we replace $d_{\mathcal{G}}(x_i, x_j)$ with

$$\min\{d_{\mathcal{G}}(x_i, x_j), d_{\mathcal{G}}(x_i, x_s) + d_{\mathcal{G}}(x_s, x_j)\}$$

  - We define the matrix

$$\Delta_{ij} = d_{\mathcal{G}}(x_i, x_j)$$

    which contains the shortest paths between the point x_i and the point x_j on the graph

Isomap

- We define the matrix
$$\mathbf{K} = -\frac{1}{2}\mathbf{H}\Delta\mathbf{H},$$
where
$$\mathbf{H} = \mathrm{Id}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$$
- The optimal k-dimensional representation $\tilde{\mathbf{X}}$ satisfying
$$\tilde{\mathbf{X}} = \arg\min \sum_{i,j} (\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|^2 - \Delta_{ij}^2)$$
is given by
$$\tilde{\mathbf{X}} = \mathbf{\Sigma}_k^{\frac{1}{2}} \mathbf{U}_k^\top$$
where $\mathbf{\Sigma}_k$ and $\mathbf{U}_k$ are the components of the truncated SVD of $\mathbf{K}$

Laplacian Eigenmaps

- Laplacian Eigenmaps seek a representation of the points that preserves the relationships between nearby points using a weight matrix $\mathbf{W}$
  - For each input element x, we find the t nearest points (T-nn)
  - We construct a sparse matrix $\mathbf{W} \in \mathbb{R}^{m \times m}$ where

$$\mathbf{W}_{ij} = \begin{cases} \exp\left(\frac{-\|x_i - x_j\|^2}{\sigma^2}\right) & \text{if } x_i \text{ and } x_j \text{ are neighbors} \\ 0 \quad \text{otherwise} \end{cases}$$

  - We define the diagonal matrix $\mathbf{D}$ with

$$\mathbf{D}_{ii} = \sum_j W_{ij}$$

  - The k-dimensional representation is obtained by minimizing the weighted distance between neighboring points, namely

$$\tilde{\mathbf{X}} = \arg\min \sum_{i,j} \mathbf{W}_{ij} \|\tilde{x}_i - \tilde{x}_j\|^2$$

  - The objective function penalizes neighboring points that are mapped too far away from each other, where neighborhood relationships are measured by the matrix $\mathbf{W}$
  - If $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian associated with the data and $\mathbf{U}_k^\top$ are the last k singular vectors of $\mathbf{L}$, then the representation is given by

$$\tilde{\mathbf{X}} = \mathbf{U}_k^\top$$

Random Projections

- By random projection we mean a projection matrix $\mathbf{T} \in \mathbb{R}^{k \times n}$ whose entries are distributed according to a certain probability distribution
- The use of random projections for dimensionality reduction is justified by the following result

Lemma (Johnson-Lindenstrauss Lemma)

Given $\varepsilon > 0$ and $\mathbf{X} \in \mathbb{R}^{n \times m}$ representing the m input points in $\mathbb{R}^n$, there exist $k \in O(\varepsilon^{-2} \ln(m))$ and a random matrix $T \in \mathbb{R}^{k \times n}$ such that

$$\mathbb{P}\left[(1-\varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \|T\mathbf{X}_i - T\mathbf{X}_k\|_2 \leq (1+\varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2\right] \geq 1 - 2e^{-\mathcal{C}\varepsilon^2 k}$$

where $\mathcal{C}$ is a constant.

Observe that the dimension k of the projected space does not depend on the original dimension n, but only on the number of considered points.

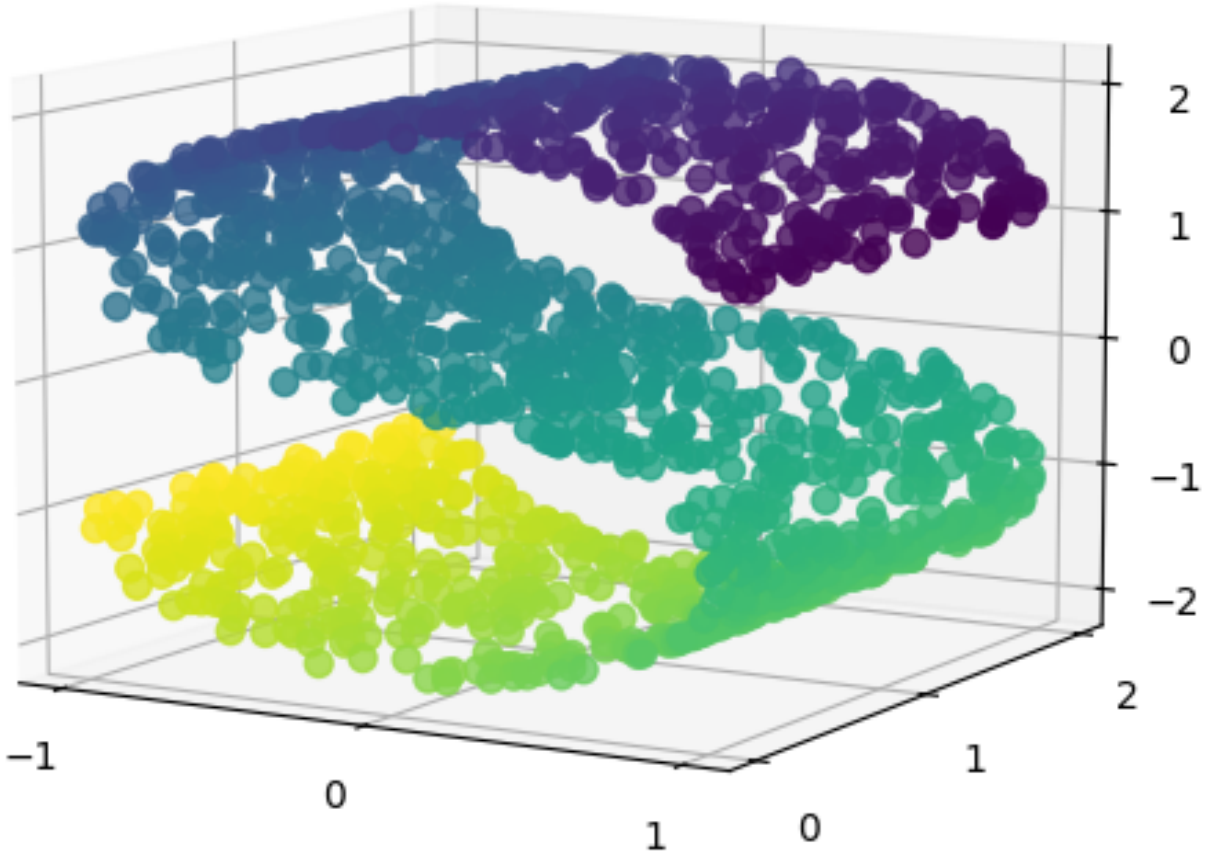
Random Projections

- The Johnson-Lindenstrauss Lemma only provides the theoretical existence of a random projection preserving the distances between points
- It has been shown that the following random matrices satisfy the Lemma:
  - The matrix T is generated row by row, where the first row is a unit vector sampled uniformly from $\mathbb{S}^{k-1}$, the second row is a unit vector sampled uniformly from the subspace orthogonal to the first row, and so on until the matrix T is completed;
  - A simpler random matrix to construct is the matrix T whose entries are generated according to the following rule

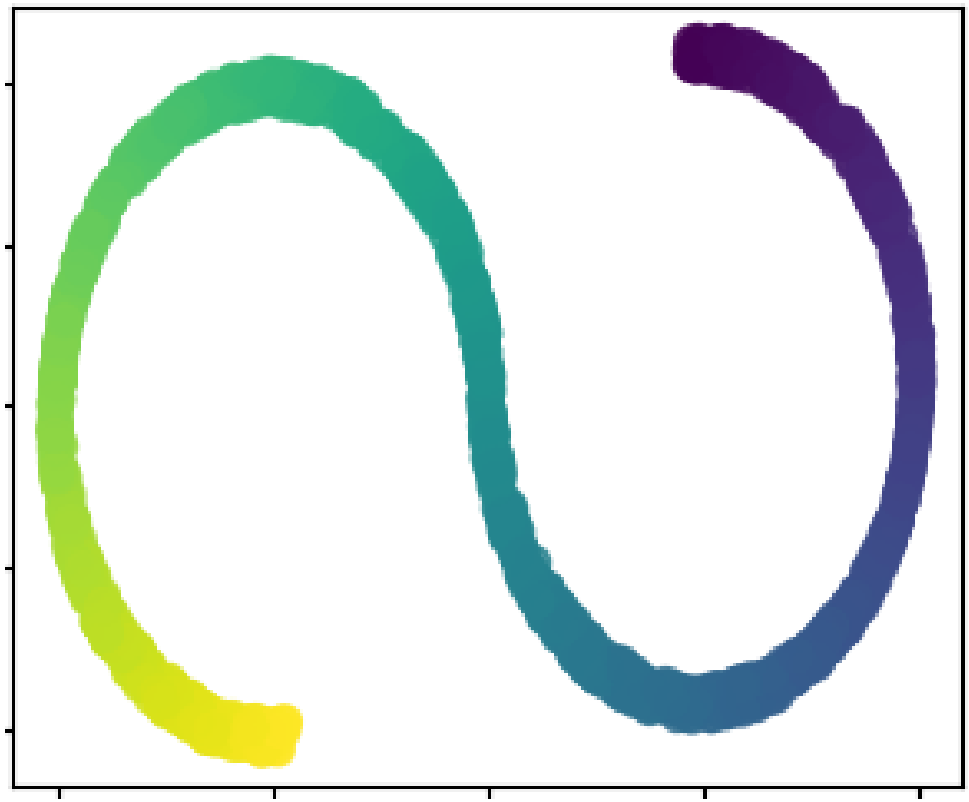
$$T_{ij} = \begin{cases} \sqrt{3} & \text{with probability } \frac{1}{6} \\ 0 & \text{with probability } \frac{2}{3} \\ -\sqrt{3} & \text{with probability } \frac{1}{6} \end{cases}$$

Example I

- Consider the following set of points in $\mathbf{R}^3$

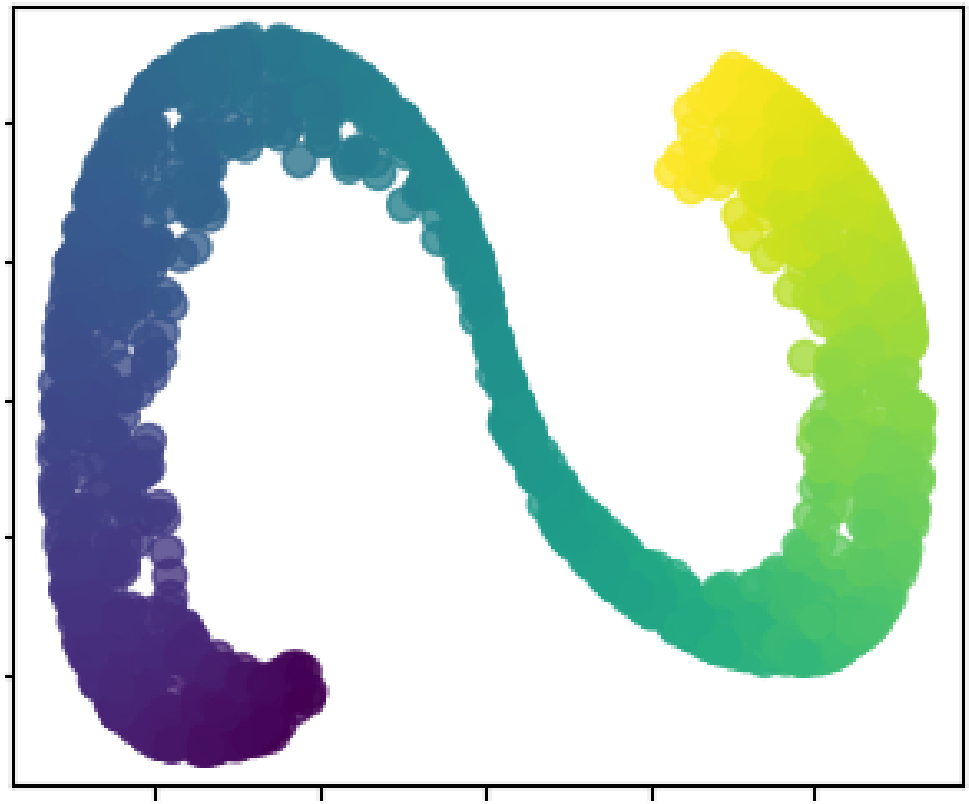


Example II



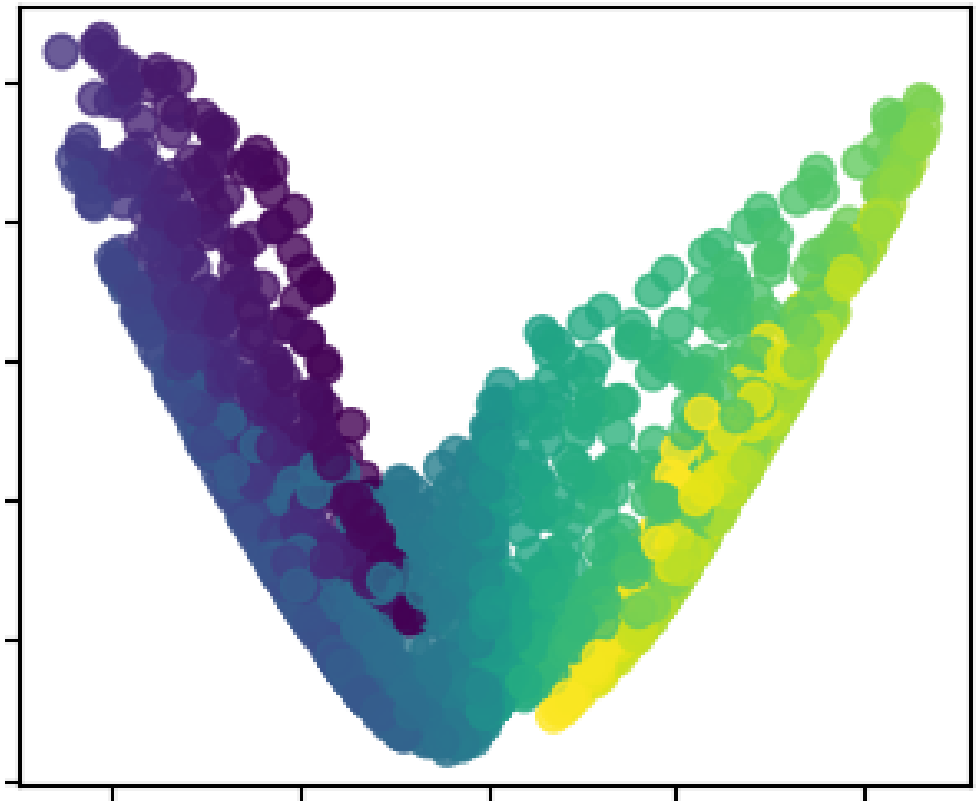
PCA lineare

Example III



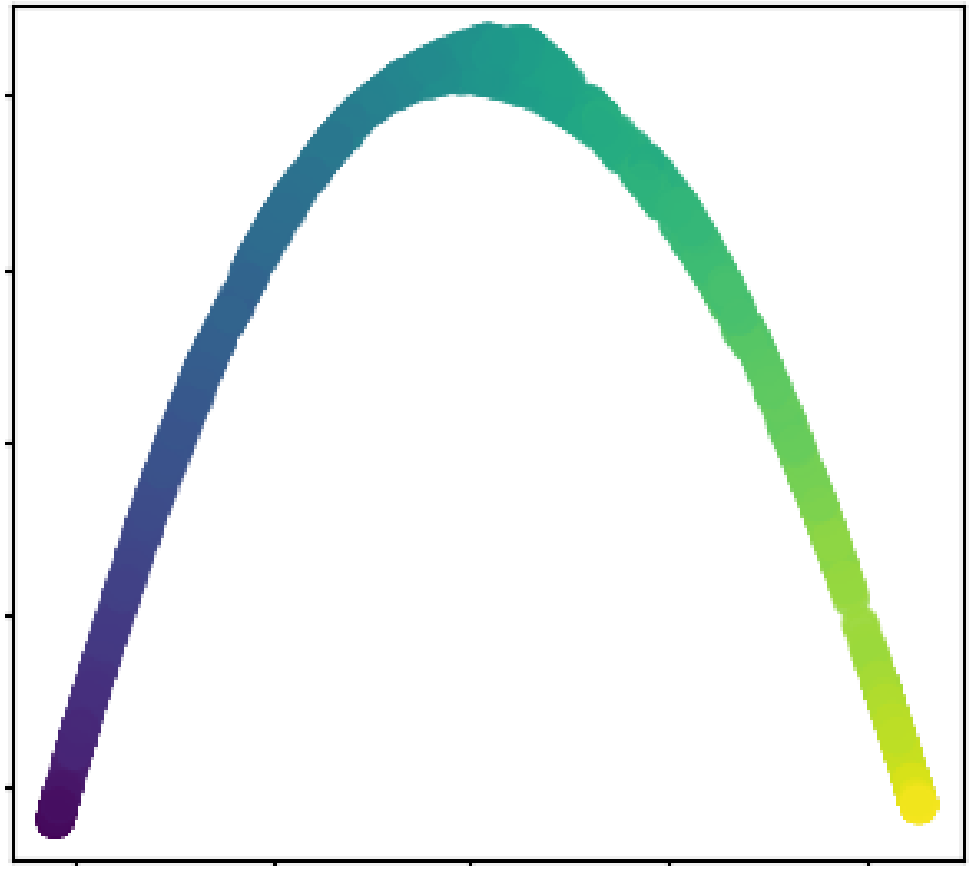
PCA con kernel gaussiano

Example IV



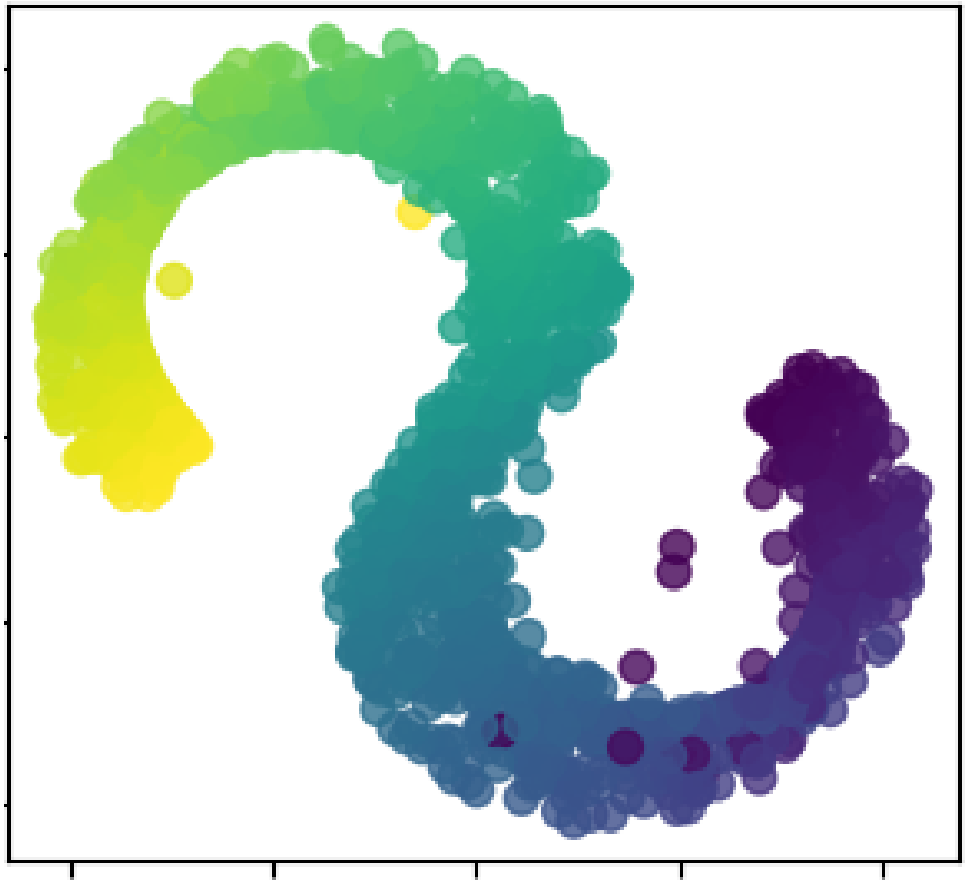
PCA con kernel polinomiale

Example V



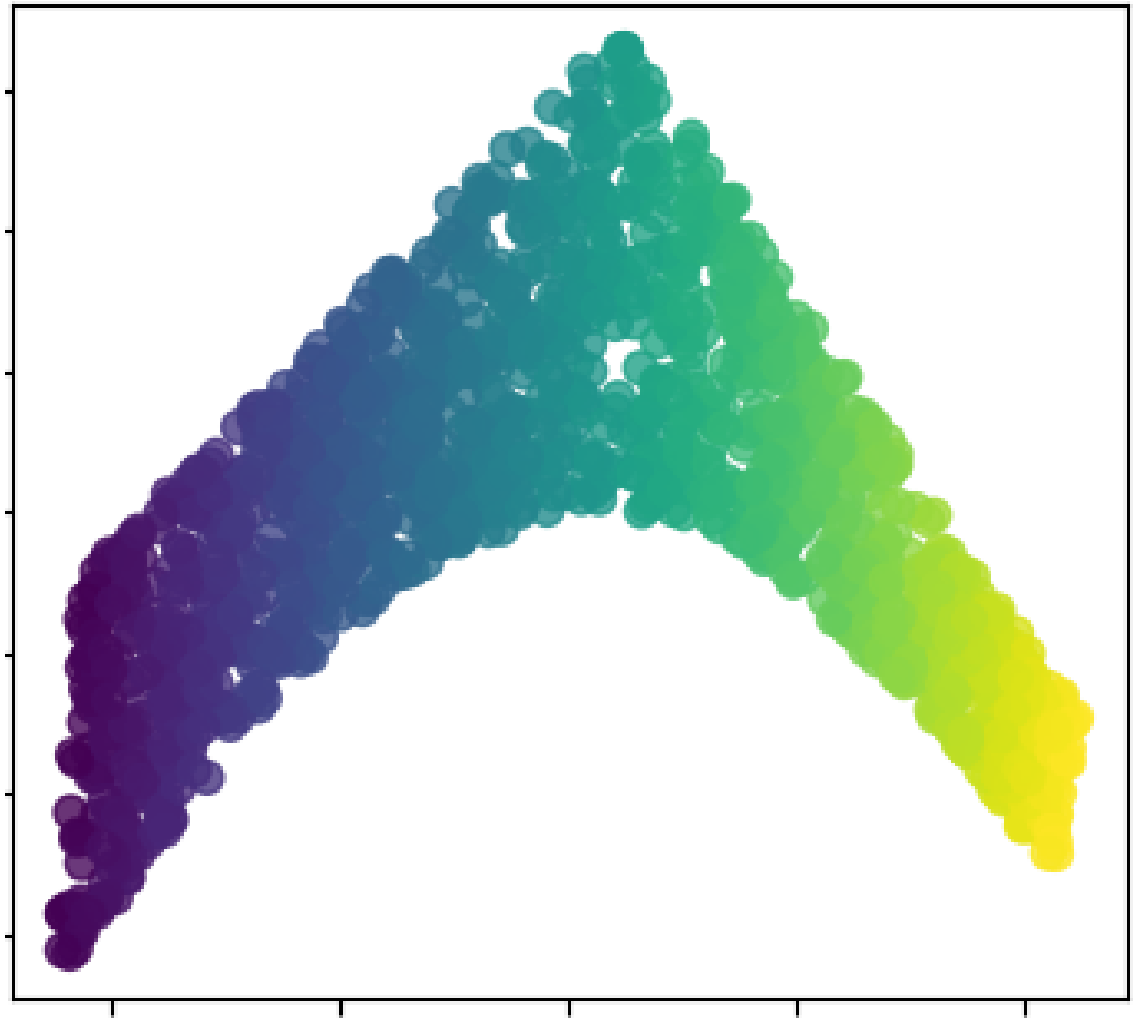
Riduzione dimensionale spettrale

Example VI



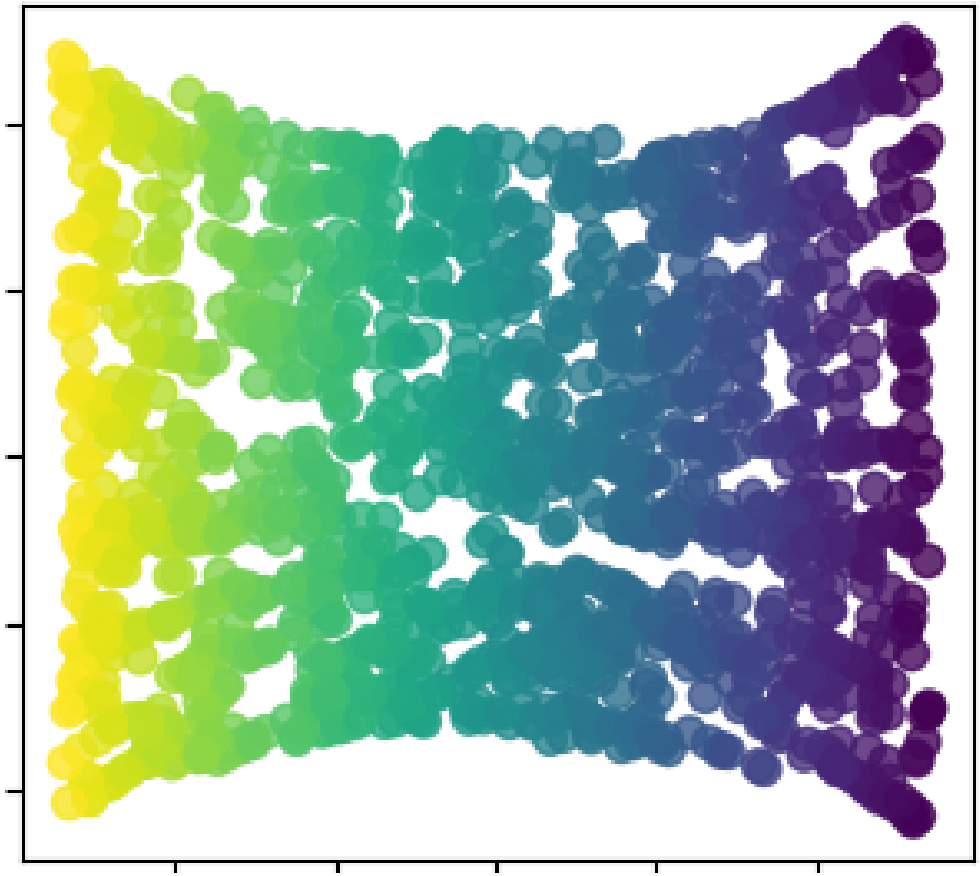
Multidimensional scaling

Example VII



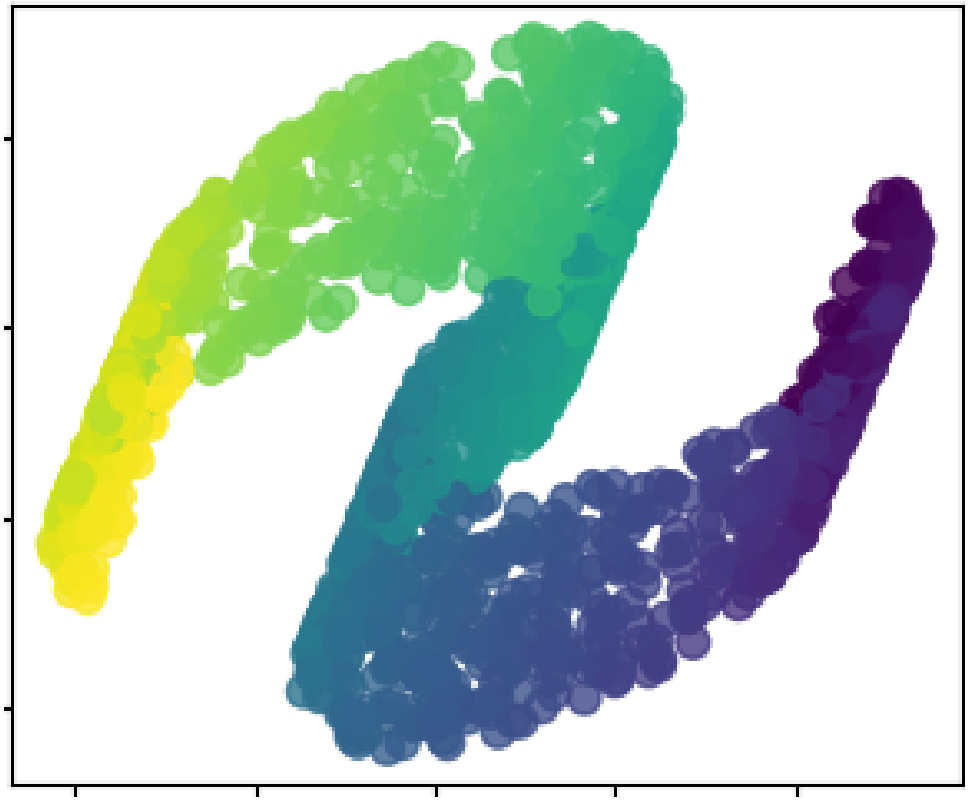
Embedding localmente lineari

Example VIII



Isomap

Example IX



Proiezione casuale Gaussiana