Lecture 13: Data representation - Preserving local geometry

Rebecka Jörnsten, Mathematical Sciences

MSA220/MVE441 Statistical Learning for Big Data

9th May 2022

CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Dimension reduction while preserving distances

Preserving distance

When creating a map, the goal is to project the three-dimensional earth onto a two-dimensional paper. It is desirable to retain certain properties of the projected areas, e.g. size of countries or length of coast lines. Keeping all properties is not possible and the field of **cartography** (and **differential geometry**) deals with the possible options.

Preserving distance

When creating a map, the goal is to project the three-dimensional earth onto a two-dimensional paper. It is desirable to retain certain properties of the projected areas, e.g. size of countries or length of coast lines. Keeping all properties is not possible and the field of **cartography** (and **differential geometry**) deals with the possible options.

Like in cartography, the goal of **dimension reduction** can be subject to different criteria, e.g. PCA preserves the directions of largest variance.

Preserving distance

When creating a map, the goal is to project the three-dimensional earth onto a two-dimensional paper. It is desirable to retain certain properties of the projected areas, e.g. size of countries or length of coast lines. Keeping all properties is not possible and the field of **cartography** (and **differential geometry**) deals with the possible options.

Like in cartography, the goal of **dimension reduction** can be subject to different criteria, e.g. PCA preserves the directions of largest variance.

What if we want to approximately preserve the **relative positions** of feature vectors while reducing the dimension?

For given vectors $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^p$ we want to find $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^q$ where $q < p$ such that

$$\|\mathbf{x}_i - \mathbf{x}_l\|_2 \approx \|\mathbf{y}_i - \mathbf{y}_l\|_2$$

Distance matrices and the linear kernel

Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, note that

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

is the **Gram matrix** $\mathbf{K}$ of the linear kernel.

Distance matrices and the linear kernel

Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, note that

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

is the **Gram matrix** $\mathbf{K}$ of the linear kernel.

Let $\mathbf{D}^{(l,m)} = \|\mathbf{x}_l - \mathbf{x}_m\|_2$ be the distance matrix in the Euclidean norm. Note that

$$\|\mathbf{x}_l - \mathbf{x}_m\|_2^2 = \mathbf{x}_l^\top \mathbf{x}_l - 2\mathbf{x}_l^\top \mathbf{x}_m + \mathbf{x}_m^\top \mathbf{x}_m$$

and (with element-wise exponentiation)

$$-\frac{1}{2}\mathbf{D}^2 = \mathbf{X}\mathbf{X}^\top - \frac{1}{2}\mathbf{1}\,\mathrm{diag}(\mathbf{X}\mathbf{X}^\top)^\top - \frac{1}{2}\,\mathrm{diag}(\mathbf{X}\mathbf{X}^\top)\mathbf{1}^\top.$$

Distance matrices and the linear kernel

Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, note that

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

is the **Gram matrix** $\mathbf{K}$ of the linear kernel.

Let $\mathbf{D}^{(l,m)} = \|\mathbf{x}_l - \mathbf{x}_m\|_2$ be the distance matrix in the Euclidean norm. Note that

$$\|\mathbf{x}_l - \mathbf{x}_m\|_2^2 = \mathbf{x}_l^\top \mathbf{x}_l - 2\mathbf{x}_l^\top \mathbf{x}_m + \mathbf{x}_m^\top \mathbf{x}_m$$

and (with element-wise exponentiation)

$$-\frac{1}{2}\mathbf{D}^2 = \mathbf{X}\mathbf{X}^\top - \frac{1}{2}\mathbf{1}\,\mathrm{diag}(\mathbf{X}\mathbf{X}^\top)^\top - \frac{1}{2}\,\mathrm{diag}(\mathbf{X}\mathbf{X}^\top)\mathbf{1}^\top.$$

Through calculation it can be shown that with $\mathbf{J} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$

$$\mathbf{K} = \mathbf{J}\left(-\frac{1}{2}\mathbf{D}^2\right)\mathbf{J}$$

Finding an embedding from a distance matrix

1. Let $\mathbf{D} \in \mathbb{R}_+^{n\times n}$ be a given distance matrix in the Euclidean norm
2. Compute $\mathbf{K} = \mathbf{J}\left(-\frac{1}{2}\mathbf{D}^2\right)\mathbf{J} = \mathbf{X}\mathbf{X}^\top$. Then there exists an **exact embedding** in $q = \mathrm{rank}(\mathbf{K}) \leq \mathrm{rank}(\mathbf{X}) \leq \min(n, p)$ dimensions.
   2.1 Perform PCA on $\mathbf{K} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$
   2.2 If $q = \mathrm{rank}(\mathbf{K})$, set
   $$\mathbf{Y} = \mathbf{U}_q\boldsymbol{\Lambda}_q^{1/2} = (\sqrt{\lambda_1}\mathbf{u}_1, \dots, \sqrt{\lambda_q}\mathbf{u}_q) \in \mathbb{R}^{n\times q}.$$
   All other $\lambda_{q+1}, \dots, \lambda_{\min(n,p)}$ are equal to zero.
   2.3 The rows of $\mathbf{Y}$ are the sought-after embedding, i.e. for $\mathbf{y}_l = \mathbf{Y}^{(l,:)}$ it holds that
   $$\mathbf{Y}\mathbf{Y}^\top = \mathbf{U}_q\boldsymbol{\Lambda}_q^{1/2}\boldsymbol{\Lambda}_q^{1/2}\mathbf{U}_q^\top = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top = \mathbf{K} = \mathbf{X}\mathbf{X}^\top$$
   which implies
   $$\begin{aligned}\|\mathbf{x}_l - \mathbf{x}_m\|_2^2 &= \mathbf{x}_l^\top\mathbf{x}_l - 2\mathbf{x}_l^\top\mathbf{x}_m + \mathbf{x}_m^\top\mathbf{x}_m \\ &= \mathbf{y}_l^\top\mathbf{y}_l - 2\mathbf{y}_l^\top\mathbf{y}_m + \mathbf{y}_m^\top\mathbf{y}_m \\ &= \|\mathbf{y}_l - \mathbf{y}_m\|_2^2.\end{aligned}$$

Multi-dimensional scaling

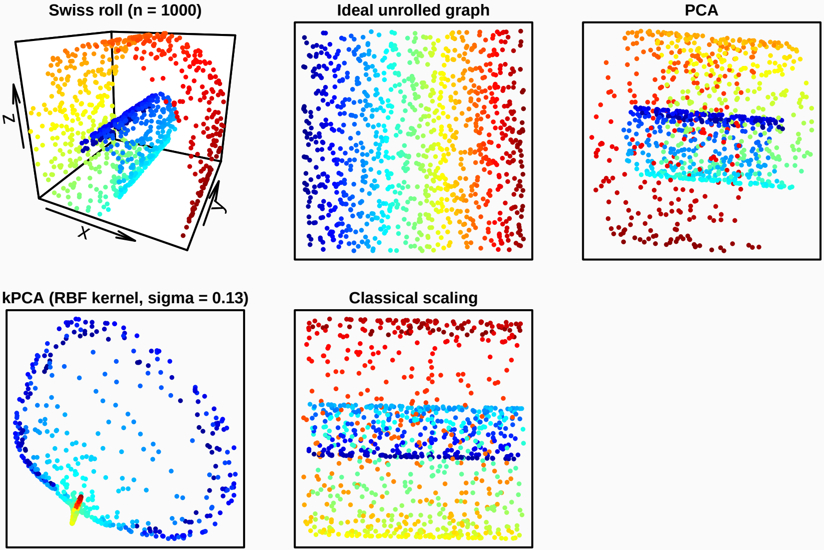
- This procedure is not guaranteed to lead to dimension reduction, i.e. $q = p$ possible. However, usually the internal structure of the data is lower-dimensional and $q < p$.
- Keeping only the first $m < q$ components of $\mathbf{y}_l$ is known as **classical scaling** or **multi-dimensional scaling (MDS)** and minimizes the so-called **stress** or **strain**

$$d(\mathbf{D}, \mathbf{Y}) = \left(\sum_{i \neq j} \left(\mathbf{D}^{(i,j)} - \|\mathbf{y}_i - \mathbf{y}_j\|_2 \right)^2 \right)^{1/2}$$

- Results also hold for general distance matrices $\mathbf{D}$ as long as $\lambda_1, \dots, \lambda_q > 0$ for $q = \mathrm{rank}(\mathbf{K})$. This is called **metric MDS**.

Lower-dimensional data in a high-dimensional space

A problematic geometry



What is the problem here?

- The data has an **intrinsic structure** that is quite simple (2D) in itself, but much more complex in the three-dimensional space

What is the problem here?

- The data has an **intrinsic structure** that is quite simple (2D) in itself, but much more complex in the three-dimensional space
- To understand this data set properly we need to learn about the **local structure** of the data

What is the problem here?

- The data has an **intrinsic structure** that is quite simple (2D) in itself, but much more complex in the three-dimensional space
- To understand this data set properly we need to learn about the **local structure** of the data
- PCA is a **global method** and will always look at all data

What is the problem here?

- The data has an **intrinsic structure** that is quite simple (2D) in itself, but much more complex in the three-dimensional space
- To understand this data set properly we need to learn about the **local structure** of the data
- PCA is a **global method** and will always look at all data
- kernel PCA introduces a different distance measure but the chosen Gaussian kernel does not represent the structure of the data well

What is the problem here?

- The data has an **intrinsic structure** that is quite simple (2D) in itself, but much more complex in the three-dimensional space
- To understand this data set properly we need to learn about the **local structure** of the data
- PCA is a **global method** and will always look at all data
- kernel PCA introduces a different distance measure but the chosen Gaussian kernel does not represent the structure of the data well
- Classical scaling performs (and works) roughly like PCA

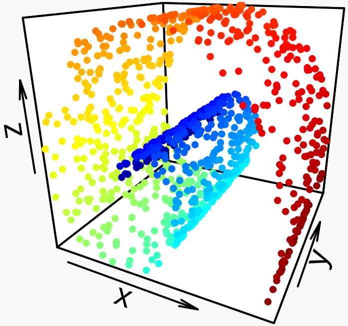
What is the problem here?

- The data has an **intrinsic structure** that is quite simple (2D) in itself, but much more complex in the three-dimensional space
- To understand this data set properly we need to learn about the **local structure** of the data
- PCA is a **global method** and will always look at all data
- kernel PCA introduces a different distance measure but the chosen Gaussian kernel does not represent the structure of the data well
- Classical scaling performs (and works) roughly like PCA
- **What is the issue?** All approaches measure distances in the Euclidean norm of the surrounding three-dimensional space.

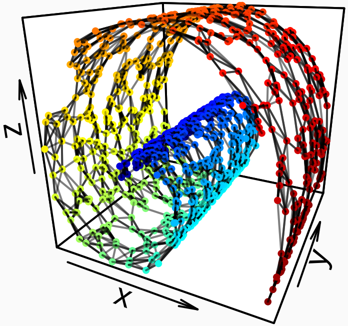
Data-driven distance measure (I)

We can create a local, data-driven distance measure by looking at the k nearest neighbours of a data point.

Swiss roll (n = 1000)



Nearest neighbours (k = 6)



Data-driven distance measure (II)

Computation

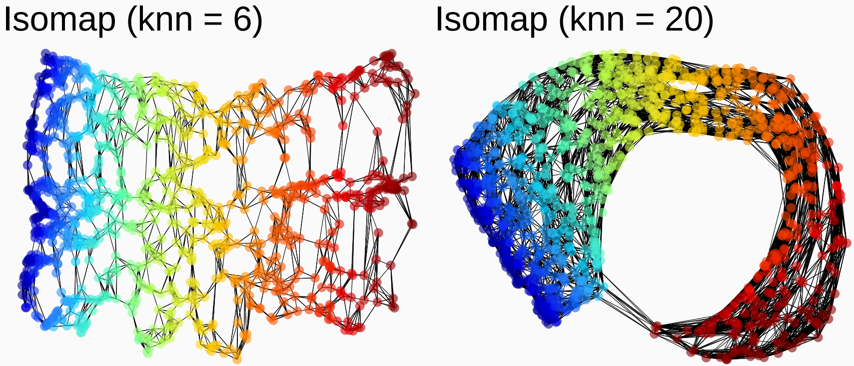
1. For a data point $\mathbf{x}_l$ find the k nearest neighbours
2. Construct a graph between data points and their k nearest neighbours, weighting each edge by the Euclidean distance
3. To measure distance between data points measure their **geodesic distance**, i.e. find the shortest path in the weighted graph and sum up the weights

This creates a distance matrix $\mathbf{D}_G$ between data points that is more adapted to the actual geometry.

To **embed the geometry** in a lower-dimensional space, MDS can be applied to $\mathbf{D}_G$, the resulting method is called **Isomap**.

Isomap

Isomap can work well but is sensitive to the number of nearest neighbours.



Caveats of Isomap

- The graph that is formed in the first stage of Isomap can have **multiple unconnected components**. This leads to **infinite** geodesic distances between some data points (because they are unreachable from each other)

Caveats of Isomap

- The graph that is formed in the first stage of Isomap can have **multiple unconnected components**. This leads to **infinite** geodesic distances between some data points (because they are unreachable from each other)
- Implementations typically return a different embedding for each component of the graph

Caveats of Isomap

- The graph that is formed in the first stage of Isomap can have **multiple unconnected components**. This leads to **infinite** geodesic distances between some data points (because they are unreachable from each other)
- Implementations typically return a different embedding for each component of the graph
- Isomap also has problems with datasets that have **varying density**

Caveats of Isomap

- The graph that is formed in the first stage of Isomap can have **multiple unconnected components**. This leads to **infinite** geodesic distances between some data points (because they are unreachable from each other)
- Implementations typically return a different embedding for each component of the graph
- Isomap also has problems with datasets that have **varying density**
- Number of nearest neighbours has to be carefully tuned

A different approach to local dimension reduction

Probability as a measure of similarity

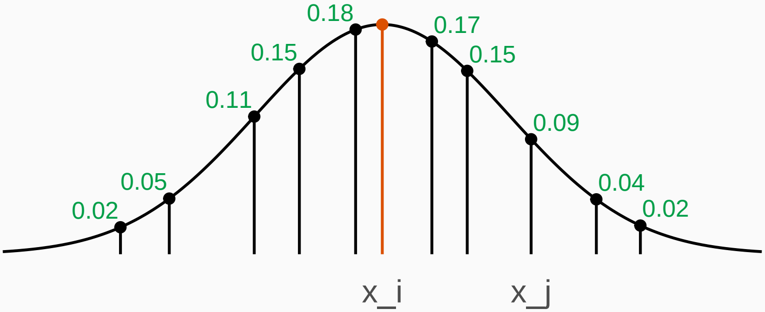
Given a set of feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ a measure of **relative similarity** between the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is their **pairwise Euclidean distance** $\|\mathbf{x}_i - \mathbf{x}_j\|_2$.

Probability as a measure of similarity

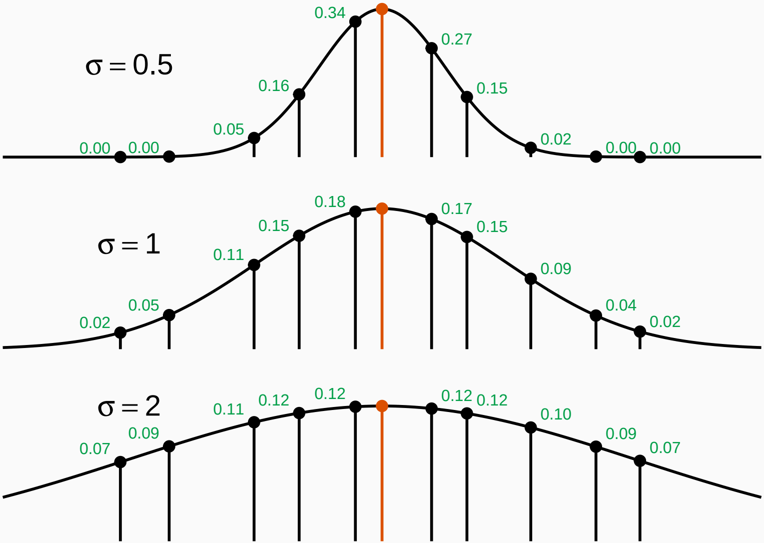
Given a set of feature vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ a measure of **relative similarity** between the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is their **pairwise Euclidean distance** $\|\mathbf{x}_i - \mathbf{x}_j\|_2$.

This measure can be **localized** around a vector $\mathbf{x}_i$ for a $\sigma > 0$

$$p_{j|i} = \frac{\exp\left(-\|\mathbf{x}_j - \mathbf{x}_i\|_2^2/(2\sigma^2)\right)}{\sum_{l\neq i} \exp\left(-\|\mathbf{x}_l - \mathbf{x}_i\|_2^2/(2\sigma^2)\right)} \quad j = 1, \dots, n,\ j \neq i \quad \text{and} \quad p_{i|i} = 0$$



σ determines the size of the local neighbourhood



Connection between σ and entropy

Denote the discrete probability distribution $P_i = (p_{j|i})_j$.

The **entropy** of P_i is a measure for how much information we gain by observing a random variable $X \sim P_i$ (i.e. by observing one of the $\mathbf{x}_j$'s in the neighbourhood). It is defined as

$$H(X) = -\sum_{j \neq i} p_{j|i} \log_2 p_{j|i}.$$

with $p_{j|i} \log_2 p_{j|i} := 0$ if $p_{j|i} = 0$.

Connection between σ and entropy

Denote the discrete probability distribution $P_i = (p_{j|i})_j$.

The **entropy** of P_i is a measure for how much information we gain by observing a random variable $X \sim P_i$ (i.e. by observing one of the $\mathbf{x}_j$'s in the neighbourhood). It is defined as

$$H(X) = -\sum_{j \neq i} p_{j|i} \log_2 p_{j|i}.$$

with $p_{j|i} \log_2 p_{j|i} := 0$ if $p_{j|i} = 0$.

Observations:

- Small values of σ lead to small values for the entropy (e.g. $\sigma = 0.5$, $H(X) = 2.44$)
- When σ increases, then the entropy increases as well (e.g. $\sigma = 2$, $H(X) = 4.22$)

Connection between σ and perplexity

The **Perplexity** of $X \sim P_i$ is defined as

$$\mathrm{Perp}(X) = 2^{H(X)}$$

and is interpreted as the average number of neighbours in the local neighbourhood around $\mathbf{x}_i$.

Connection between σ and perplexity

The **Perplexity** of $X \sim P_i$ is defined as

$$\mathrm{Perp}(X) = 2^{H(X)}$$

and is interpreted as the average number of neighbours in the local neighbourhood around $\mathbf{x}_i$.

Observations

- If $H(X) = 0$, i.e. we learn on average nothing by observing X, then $\mathrm{Perp}(X) = 1$ and the neighbourhood contains only the centre data point itself.
  - Small σ leads on average to smaller neighbourhood
- If $H(X)$ grows larger, i.e. we learn on average more about X by observing it, then $\mathrm{Perp}(X)$ grows as well and the neighbourhood around $\mathbf{x}_i$ gets larger.
  - For growing σ the average size of neighbourhoods grows as well

Fixating perplexity and symmetrising the distribution

As a slight generalisation, let each $\mathbf{x}_i$ have its own $\sigma_i > 0$, i.e.

$$p_{j|i} = \frac{\exp\left(-\|\mathbf{x}_j - \mathbf{x}_i\|_2^2/(2\sigma_i^2)\right)}{\sum_{l\neq i}\exp\left(-\|\mathbf{x}_l - \mathbf{x}_i\|_2^2/(2\sigma_i^2)\right)} \quad j = 1, \dots, n,\ j \neq i \quad \text{and} \quad p_{i|i} = 0.$$

By setting perplexity to a fixed value $\gamma > 0$ we **control** the **average size of neighbourhoods** around all $\mathbf{x}_i$. For a fixed γ the individual σ_i can be calculated. Depending on the data these can vary with i.

Note: This is similar but more flexible than building nearest neighbour graphs for Isomap.

The values $p_{j|i}$ and $p_{i|j}$ are **asymmetric** similarity measures, due to the different parameters σ_i and σ_j. Define the **symmetrized** probabilities

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2} \quad \text{and} \quad p_{ii} = 0.$$

Connection to lower-dimensional embeddings

Our goal is to embed the potentially high-dimensional vectors $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^p$ into a lower dimensional space $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^q$ with $q \ll p$.

Given a perplexity parameter $\gamma > 0$, we found a way to measure local similarity in $\mathbb{R}^p$ using the probabilities p_{ij}.

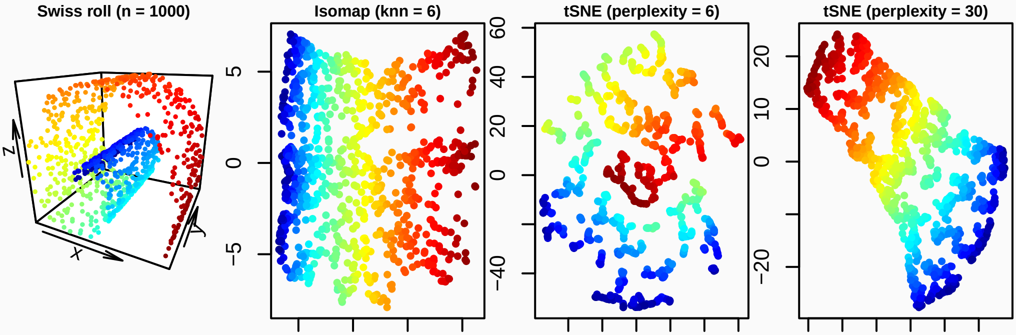
A technique called **t-distributed stochastic neighbour embedding (tSNE)** uses the **t-distribution with one degree of freedom** (or **Cauchy distribution**) to measure similarity in $\mathbb{R}^q$ with

$$q_{ij} = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2\right)^{-1}}{\sum_{l \neq r} \left(1 + \|\mathbf{y}_l - \mathbf{y}_r\|_2^2\right)^{-1}} \quad \text{and} \quad q_{ii} = 0.$$

To determine the $\mathbf{y}_l$ the Kullback-Leibler divergence between the distributions $P = (p_{ij})_{ij}$ and $Q = (q_{ij})_{ij}$ is minimized with gradient descent (+ numerical tricks)

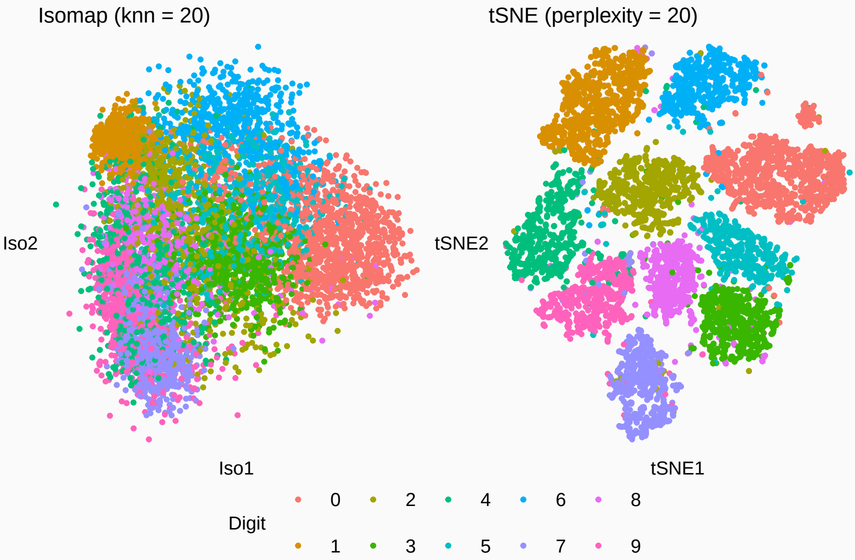
$$\mathrm{KL}(P||Q) = \sum_{i \neq l} p_{il} \log \frac{p_{il}}{q_{il}}$$

Revisiting the Swiss roll with tSNE



- Results are similar to Isomap
- Strong dependence on perplexity and no literal relationship between k-nearest neighbours and perplexity parameter
- Slightly more condensed, but manages the main goal to unroll data

A more impressive example of tSNE

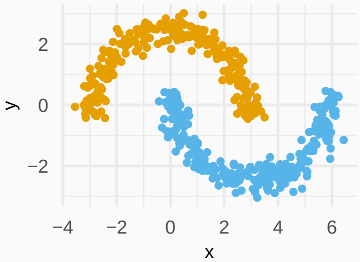


Caveats of tSNE

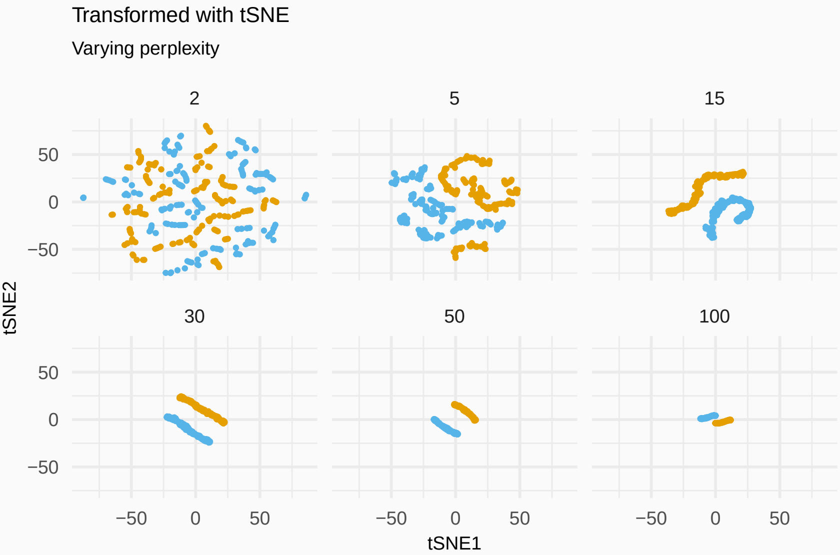
tSNE is a powerful method but comes with some difficulties as well

- Convergence to local minimum (i.e. repeated runs can give different results)
- Perplexity is hard to tune (as with any tuning parameter)

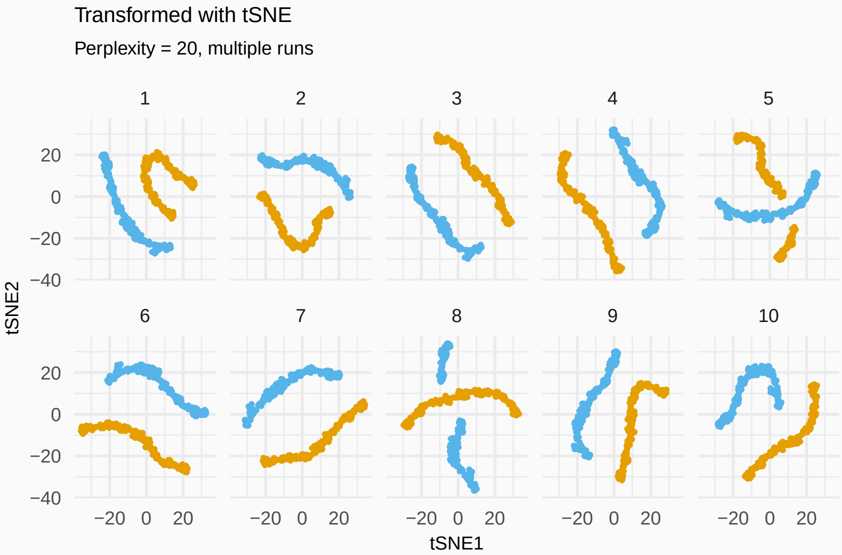
Let's see what tSNE does to our old friend, the moons dataset.



Influence of perplexity on tSNE



tSNE multiple runs

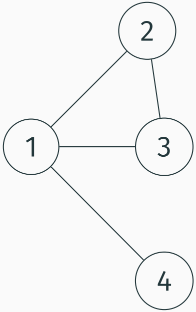


Spectral clustering

Starting point

- Many clustering methods focus on **global behaviour** of the data (e.g. GMM, k-means, hierarchical clustering with complete linkage)
- To adapt to **local behaviour** hierarchical clustering with single linkage and the group of **density-based** algorithms (e.g. DBSCAN) showed some success
- In dimension reduction **building a graph of the data** based on k nearest neighbours helped to capture **local behaviour** (e.g. Isomap)
- **Idea:** Build a graph representing local behaviour in the data and find good cut points to partition the graph into K clusters.

Graphs and adjacency matrices

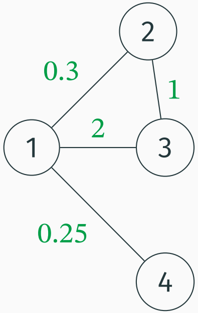


An **adjacency matrix** $\mathbf{A} \in \{0,1\}^{n\times n}$ describes edges between n nodes such that $\mathbf{A}^{(i,j)} = 1$ when there is an edge between nodes i and j and zero otherwise.

For the graph on the left

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Graphs and adjacency matrices



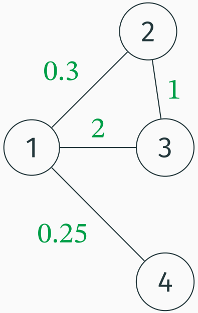
An **adjacency matrix** $\mathbf{A} \in \{0,1\}^{n \times n}$ describes edges between n nodes such that $\mathbf{A}^{(i,j)} = 1$ when there is an edge between nodes i and j and zero otherwise.

In addition, **weights** can be added to the edges, leading to a **weighted adjacency matrix** $\mathbf{W} \in [0,\infty)^{n \times n}$.

For the graph on the left

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{W} = \begin{pmatrix} 0 & 0.3 & 2 & 0.25 \\ 0.3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 0.25 & 0 & 0 & 0 \end{pmatrix}$$

Graphs and adjacency matrices



An **adjacency matrix** $\mathbf{A} \in \{0,1\}^{n\times n}$ describes edges between n nodes such that $\mathbf{A}^{(i,j)} = 1$ when there is an edge between nodes i and j and zero otherwise.

In addition, **weights** can be added to the edges, leading to a **weighted adjacency matrix** $\mathbf{W} \in [0,\infty)^{n\times n}$.

For the graph on the left

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{W} = \begin{pmatrix} 0 & 0.3 & 2 & 0.25 \\ 0.3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 0.25 & 0 & 0 & 0 \end{pmatrix}$$

Note: Undirected graphs have symmetric adjacency matrices. Directed graphs can be described by unsymmetric adjacency matrices.

A graph from data

Recall: In the first step of Isomap, a **weighted undirected graph** was built based on the k nearest neighbours of a data point.

A weighted undirected graph can be constructed from a **weighted adjacency matrix** $\mathbf{W}$.

1. For a data point $\mathbf{x}_l$, find the k nearest neighbours.
2. Set $\mathbf{W}^{(l,l_i)} = g(\|\mathbf{x}_l - \mathbf{x}_{l_i}\|_2)$ where $g : [0, \infty) \to [0, \infty)$ is a monotone function and $\mathbf{x}_{l_i}$, $i = 1, \dots, k$ are the nearest neighbours of $\mathbf{x}_l$. In addition, set all $\mathbf{W}^{(l,m)} = 0$ for $m \notin \{l_1, \dots, l_k\}$ (in particular $\mathbf{W}^{(l,l)} = 0$).
3. Construct a graph where each node represents a data point $\mathbf{x}_l$ and there is a weighted edge between $\mathbf{x}_l$ and $\mathbf{x}_m$ if $\mathbf{W}^{(l,m)} > 0$.

In Isomap $g(z) = z$, but in the following $g_c(z) = \exp(-z^2/c)$ for $c > 0$.

Node degree and the graph Laplacian

Given the weighted adjacency matrix $\mathbf{W}$, the **degree of node l** describes how well-connected a node is

$$d_l = \sum_{m=1}^{n} \mathbf{W}^{(l,m)}$$

and the **degree matrix** is $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)$.

Define now the **graph Laplacian**, a measure of **information flow**, as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

Interpretation: If heat were to be distributed from node to node with flow speeds described by $\mathbf{W}$, then $\mathbf{L}$ takes the role of the discretised **Laplacian operator** ∇^2 in the **heat equation** for the heat distribution $\boldsymbol{\phi}$

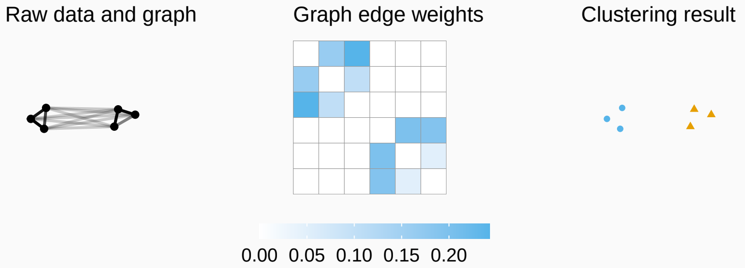
$$\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} + k\mathbf{L}\boldsymbol{\phi} = \mathbf{0}$$

Graph cutting

A **good separation of the graph** into two parts A and B is one where flow between the parts is minimized and neither is chosen too small, i.e.

$$\min_{A,B} \left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)} \right) \sum_{l \in A, m \in B} \mathbf{W}^{(l,m)}$$

$$\text{where} \quad \mathrm{vol}(A) = \sum_{l \in A} \sum_{m=1}^{n} \mathbf{W}^{(l,m)} = \sum_{l \in A} d_l$$



Finding good cut points

Finding the best cut point would require to check all possible cuts and is an **NP-hard combinatorial problem**.

Observations and theorem

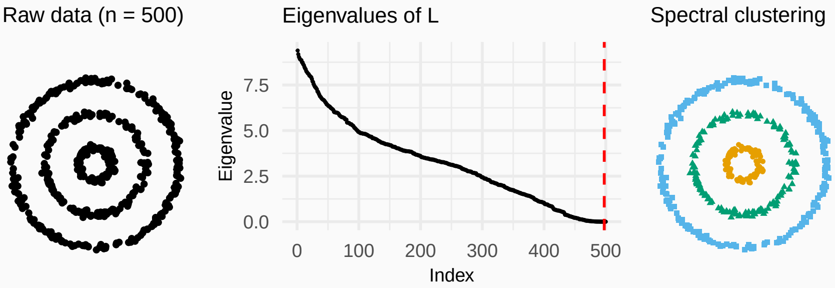
1. The graph Laplacian is symmetric and positive semi-definite, since $\mathbf{y}^\top \mathbf{L}\mathbf{y} = \sum_{i,j=1}^{n} \mathbf{W}^{(i,j)}(\mathbf{y}^{(i)} - \mathbf{y}^{(j)})^2 \geq 0$ for all $\mathbf{y} \in \mathbb{R}^n$.
2. If there are K connected components of the graph, then the set of eigenvectors of $\mathbf{L}$ with eigenvalue 0 is spanned by $\mathbf{1}_{A_k}$ for $k = 1, \dots, K$, where $\mathbf{1}_{A_k}^{(i)} = 1$ if $i \in A_k$ and zero otherwise.

In practice

- There will not be K separate connected components
- However, if K clusters exist, the smallest K eigenvalues will be near zero and the and corresponding eigenvectors close to indicator vectors.

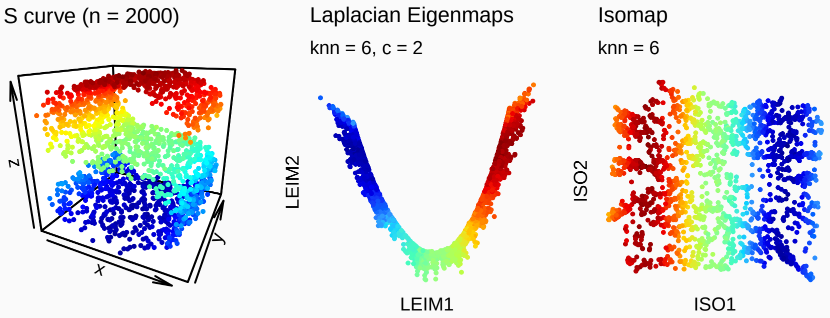
Spectral Clustering

1. Determine the weighted adjacency matrix $\mathbf{W}$ and the graph Laplacian $\mathbf{L}$
2. Find the K smallest eigenvalues of $\mathbf{L}$ that are near zero and well separated from the others
3. Find the corresponding eigenvectors $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_K) \in \mathbb{R}^{n \times K}$ and use k-means on the rows of $\mathbf{U}$ to determine cluster membership



Laplacian Eigenmaps for dimension reduction

- In addition to clustering, the eigenvectors of the Laplacian can also be used for **dimension reduction**.
- For each component, use the q eigenvectors corresponding to the q smallest non-zero eigenvalues as an embedding of the original data.
- Laplacian Eigenmaps can be shown to **optimally preserve the local behaviour on average**, but not necessarily global behaviour.



Take-home message

- Dimension reduction can aim to preserve global and local structure
- Data can have structure at multiple scales (e.g. locally flat but globally spirally as the in swiss roll example)
- Isomap and tSNE are powerful dimension reduction techniques that can help in explorative data analysis to uncover hidden structure. However, be careful not to use them blindly
- Spectral clustering can also be used for flexible clustering. There is a lot of theoretical work underpinning this technique - see e.g. Statistics and Machine Learning in High-dimensions, EEN100 given by Giuseppe Durisi and myself in LP1.