Lecture 5 - Preserving local geometry

Rebecka Jörnsten, Mathematical Sciences

MSA220/MVE441 Statistical Learning for Big Data

31th April 2025

Upcoming lectures - where are we going?

- Last lecture you learnt about CART and Random Forests (RF). RF was an example of an *ensemble method* - that is, our final classifier builds on a set of models. In RF, these were CART models built through randomization of observations and model building steps.
- In RF, the ensemble was built through *bagging*. The idea is that high-variance/low-bias models are combined in order to reduce the estimation variance. Thus, each member of the ensemble might suffer from overfitting/too flexible but the aggregate cancels out the overfitting and thus improves on generalization.

Upcoming lectures - where are we going?

- In the upcoming lecture, we will discuss how to build ensembles from low-variance/high-bias methods, so-called *weak learners*.
- Boosting weak learners has resulted in methods that are generally very high performing yet flexible.
- To introduce boosting we will look at
  1. Gradient Descent; in regression and functional gradient descent
  2. Kernel learning; kRR, kPCA
  3. The kernel trick - Support Vector Machines
  4. Boosting and Gradient Boosting, XGBOOST

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 $\mathbf{J} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$

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

Preview of next lecture: Examples of kernels

- **Linear kernel** $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$
- **Polynomial kernel** $k(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x}^\top \mathbf{y} + r)^m$
- **Radial basis function (RBF) kernel** $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{y}\|_2^2\right)$
- **Laplacian kernel** $k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|_1)$
- **Sigmoid kernel** $k(\mathbf{x}, \mathbf{y}) = \tanh(\alpha \mathbf{x}^\top \mathbf{y} + c)$

Preview 2: Advantages of using kernels

Summary

Using a positive definite kernel to measure the similarity between m-dimensional feature vectors is equivalent to

1. Using a (potentially non-linear) mapping to transform the feature vectors $\mathbf{x}$ to a q-dimensional vector $\boldsymbol{\phi}(\mathbf{x})$
2. Using the Euclidean scalar product to measure similarity between transformed feature vectors $\boldsymbol{\phi}(\mathbf{x})$

Preview 2: Advantages of using kernels

Summary

Using a positive definite kernel to measure the similarity between m-dimensional feature vectors is equivalent to

1. Using a (potentially non-linear) mapping to transform the feature vectors $\mathbf{x}$ to a q-dimensional vector $\boldsymbol{\phi}(\mathbf{x})$
2. Using the Euclidean scalar product to measure similarity between transformed feature vectors $\boldsymbol{\phi}(\mathbf{x})$

Problem: $\boldsymbol{\phi}(\mathbf{x})$ might be hard to compute.

Preview 2: Advantages of using kernels

Summary

Using a positive definite kernel to measure the similarity between m-dimensional feature vectors is equivalent to

1. Using a (potentially non-linear) mapping to transform the feature vectors $\mathbf{x}$ to a q-dimensional vector $\boldsymbol{\phi}(\mathbf{x})$
2. Using the Euclidean scalar product to measure similarity between transformed feature vectors $\boldsymbol{\phi}(\mathbf{x})$

Problem: $\boldsymbol{\phi}(\mathbf{x})$ might be hard to compute.

The **kernel-trick** is to replace scalar products with kernel evaluations. Computations are then done implicitly in the higher-dimensional space of the $\boldsymbol{\phi}(\mathbf{x})$, but all we need to do is evalute the kernel.

Preview 3: PCA and k(ernel)-PCA (I)

Idea: Use the **kernel-trick** to define augmentations implicitly and keep computations manageable.

Preview 3: PCA and k(ernel)-PCA (I)

Idea: Use the **kernel-trick** to define augmentations implicitly and keep computations manageable.

Given a positive definite kernel $k(\mathbf{x}, \mathbf{y})$, how can we perform PCA in the high-dimensional space of $\boldsymbol{\phi}(\mathbf{x})$?

Preview 3: PCA and k(ernel)-PCA (I)

Idea: Use the **kernel-trick** to define augmentations implicitly and keep computations manageable.

Given a positive definite kernel $k(\mathbf{x}, \mathbf{y})$, how can we perform PCA in the high-dimensional space of $\boldsymbol{\phi}(\mathbf{x})$?

Assume we have access to $\boldsymbol{\phi}(\mathbf{x}_l)$ for $l = 1, \dots, n$ and these transformed vectors are centred. Then we can perform PCA on

$$\widehat{\boldsymbol{\Sigma}}^{\boldsymbol{\phi}} = \frac{1}{n} \sum_{l=1}^{n} \boldsymbol{\phi}(\mathbf{x}_l) \boldsymbol{\phi}(\mathbf{x}_l)^\top = \mathbf{V}\mathbf{D}\mathbf{V}^\top$$

where $\mathbf{v}_i$ are the principal component axes and d_i the corresponding variances. The **kernel-trick** is to replace scalar products with kernel evaluations. Computations are then done implicitly in the higher-dimensional space of the $\boldsymbol{\phi}(\mathbf{x})$, but all we need to do is evalute the kernel.

Preview 3: General algorithm for kPCA

1. Choose a kernel $k(\cdot, \cdot)$ and possible hyper-parameters
2. Compute the Gram matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ for the data $\mathbf{x}_1, \dots, \mathbf{x}_n$
3. Centre $\mathbf{K}$ using $\mathbf{J} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ to get

$$\mathbf{K}' = \mathbf{J}\mathbf{K}\mathbf{J}$$

4. Perform a normal linear PCA on $\mathbf{K}' = \mathbf{A}\mathbf{\Lambda}\mathbf{A}^\top$.
5. The columns of $\mathbf{A}$ are the vectors $\mathbf{a}_i$ and set $d_i = \lambda_i / n$.
6. The projection of the l-th observation onto the i-th principal component axis is computed as

$$\boldsymbol{\eta}_l^{(i)} = \mathbf{K}'^{(l,:)} \mathbf{a}_i \in \mathbb{R}$$

Back to regular program: 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 $q = \operatorname{rank}(\mathbf{K}) \leq \operatorname{rank}(\mathbf{X}) \leq \min(n, p)$ dimensions.
   - 2.1 Perform PCA on $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$
   - 2.2 If $q = \operatorname{rank}(\mathbf{K})$, set
   $$\mathbf{Y} = \mathbf{U}_q\mathbf{\Lambda}_q^{1/2} = (\sqrt{\lambda_1}\mathbf{u}_1, \dots, \sqrt{\lambda_q}\mathbf{u}_q) \in \mathbb{R}^{n \times q}.$$
   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
   $$\mathbf{Y}\mathbf{Y}^\top = \mathbf{U}_q\mathbf{\Lambda}_q^{1/2}\mathbf{\Lambda}_q^{1/2}\mathbf{U}_q^\top = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top = \mathbf{K} = \mathbf{X}\mathbf{X}^\top$$
   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**

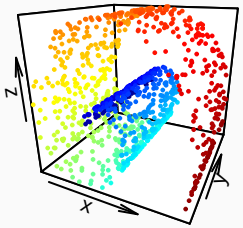
$$d(\mathbf{D}, \mathbf{Y}) = \left(\sum_{i \neq j} \left(\mathbf{D}^{(i,j)} - \|\mathbf{y}_i - \mathbf{y}_j\|_2 \right)^2 \right)^{1/2}$$

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

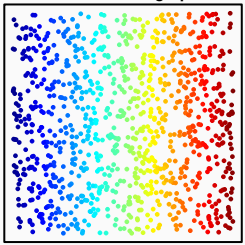
Lower-dimensional data in a high-dimensional space

A problematic geometry

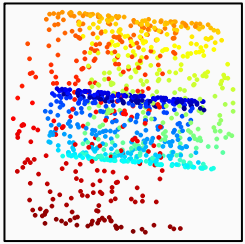
Swiss roll (n = 1000)



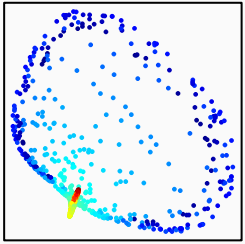
Ideal unrolled graph



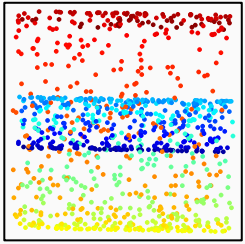
PCA



kPCA (RBF kernel, sigma = 0.13)



Classical scaling



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?

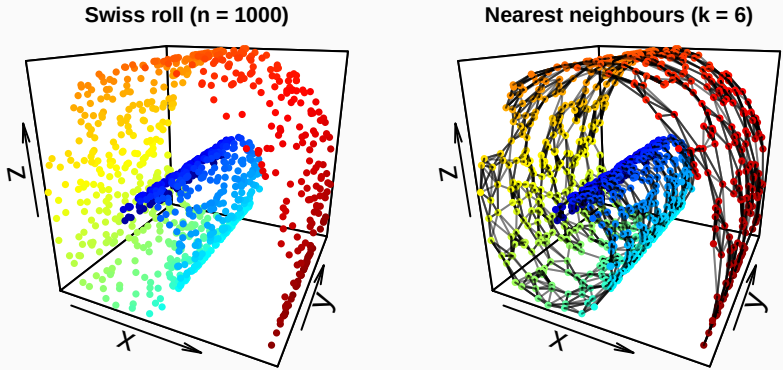
- The data has an **intrinsic structure** that is quite simple (2D) in itself, but much more complex in the three-dimensional space
- To understand this data set properly we need to learn about the **local structure** of the data
- PCA is a **global method** and will always look at all data
- kernel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



Data-driven distance measure (II)

Computation

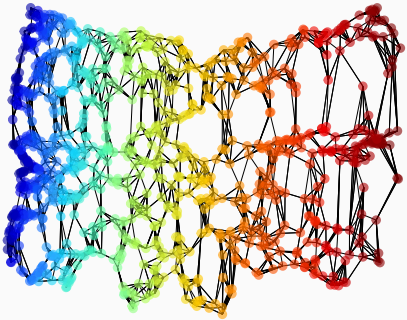
1. For a data point $\mathbf{x}_l$ find the k nearest neighbours
2. Construct a graph between data points and their k nearest neighbours, weighting each edge by the Euclidean distance
3. To measure distance between data points measure their **geodesic distance**, i.e. find the shortest path in the weighted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

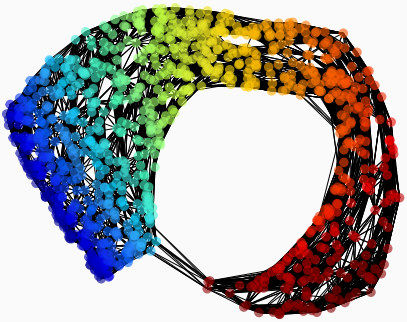
To **embed the geometry** in a lower-dimensional space, MDS can be applied to $\mathbf{D}_G$, the resulting method is called **Isomap**.

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

Isomap (knn = 6)



Isomap (knn = 20)



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

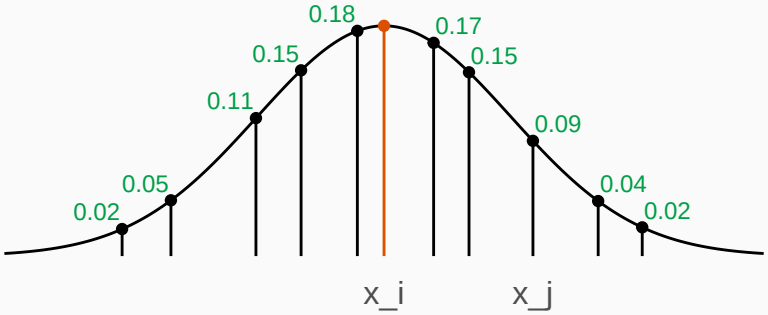
Given a set of feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ a measure of **relative similarity** between the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is their **pairwise Euclidean distance** $\|\mathbf{x}_i - \mathbf{x}_j\|_2$.

Probability as a measure of similarity

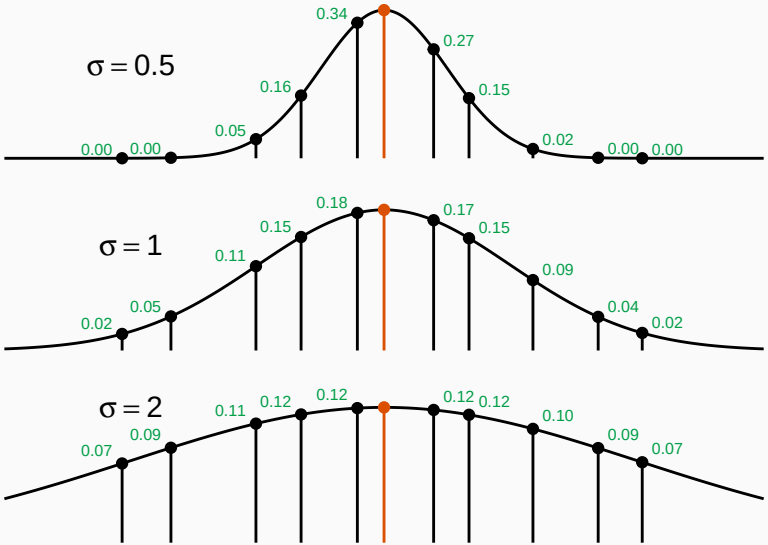
Given a set of feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ a measure of **relative similarity** between the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is their **pairwise Euclidean distance** $\|\mathbf{x}_i - \mathbf{x}_j\|_2$.

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

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



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}.

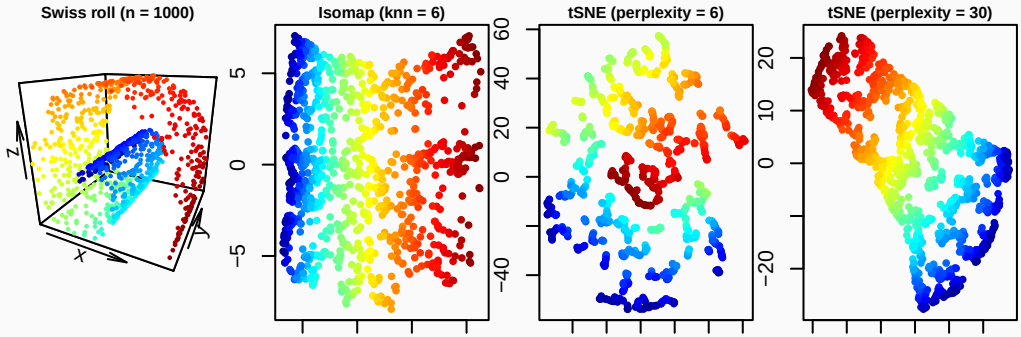
A technique called **t-distributed stochastic neighbour embedding (tSNE)** uses the **t-distribution with one degree of freedom** (or **Cauchy distribution**) to measure similarity in $\mathbb{R}^q$ with

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

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)

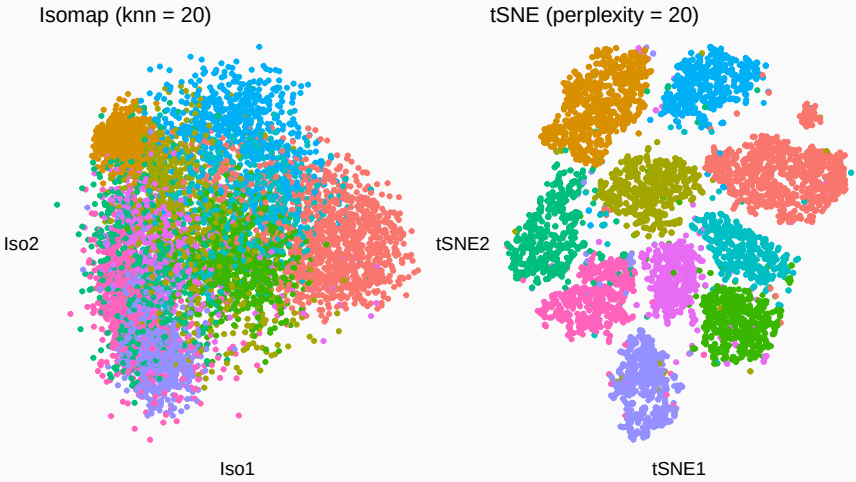
$$\mathrm{KL}(P||Q) = \sum_{i \neq l} p_{il} \log \frac{p_{il}}{q_{il}}$$

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

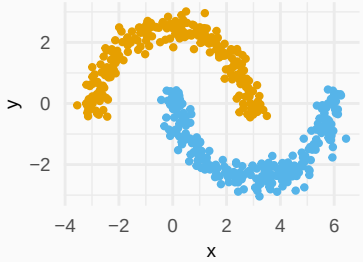


Caveats of tSNE

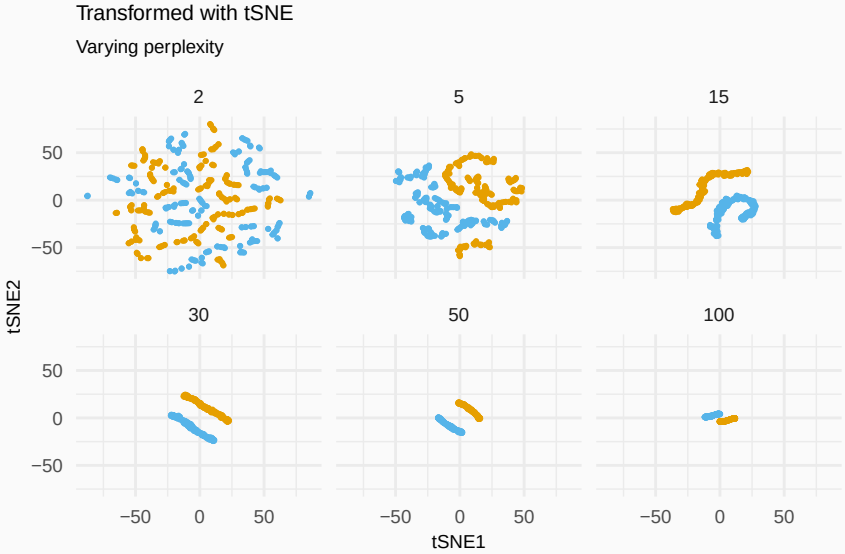
tSNE is a powerful method but comes with some difficulties as well

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

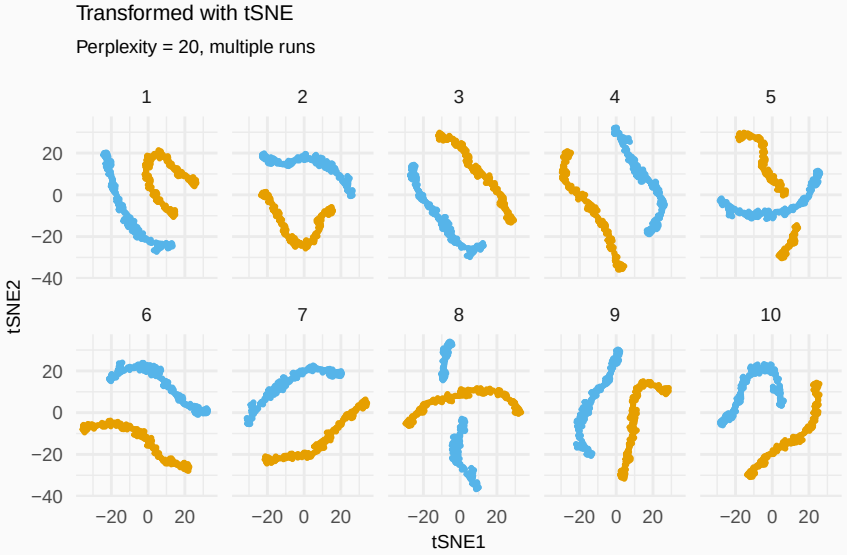
Let's see what tSNE does to our old friend, the moons dataset.



Influence of perplexity on tSNE



tSNE multiple runs



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
- Kernel PCA - a first look at the use of kernels for learning.