Technische Hochschule Rosenheim

Modul - Unsupervised and Reinforcement Learning (URL)

Bachelor Programme AAI

06 - Eigenfaces and t-SNE

Prof. Dr. Marcel Tilly

Faculty of Computer Science, Cloud Computing

Next week

Technische Hochschule Rosenheim

Invited Talk

- **John Thompson**, Information Professionals GmbH
- Title: **Data Science in Practice!**
- When and where: 11:45-13:15 (B0.14) and exercises 13:45-15:15 (B0.08a)

Abstract

Mr. Thompson will talk about data science in general and about clustering in particular. He will explain how to cluster data from real world projects. This talk will present data science in practice beyond academic examples.

What to bring

- Python (> version 3.6)
- Ideally Anaconda with packages: numpy, pandas, matplotlib, seaborn, scikit-learn
- Running "Jupyter Notebook"

X-Mas Robocode Tournament

Technische Hochschule Rosenheim

'X-Mas Lecture': 22.12 - Robocode Tournament, Glühwein and Pizza!

Register in LC: https://learning-campus.th-rosenheim.de/course/view.php?id=3135

Plan

- 09:00 -- Introduction to Robocode and strategy
- 09:45 -- Introduction to setup and time plan
- 10:15 -- Start Coding and Testing
- 11:00 -- Test round - this round is a test to check whether basics are working!
- 12:00 -16:00 -- Tournament round 1-5 (every full hour!)
- 16:30 -- Award Ceremony

Agenda

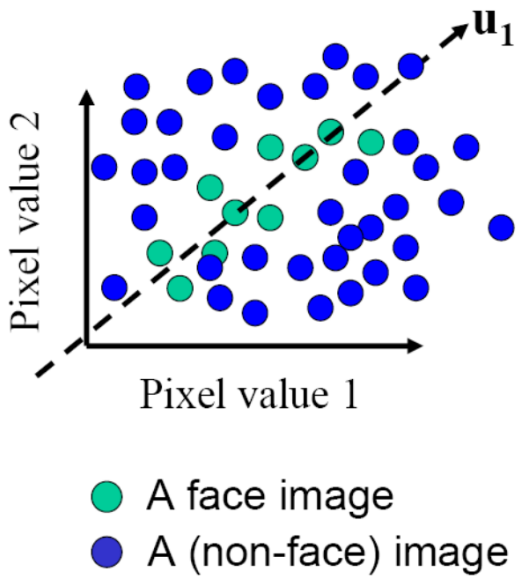
Technische Hochschule Rosenheim

> Code can be found in
>
> - Eigenfaces.ipynb
> - t-SNE.ipynb
> - or on GitHub or hosted on myBinder

- Applications of PCA algorithm
  - MNIST and PCA
  - Eigenfaces and PCA
- Stochastic Nearest Neighbor Embeddings (SNE)
  - t-SNE

Applications of PCA

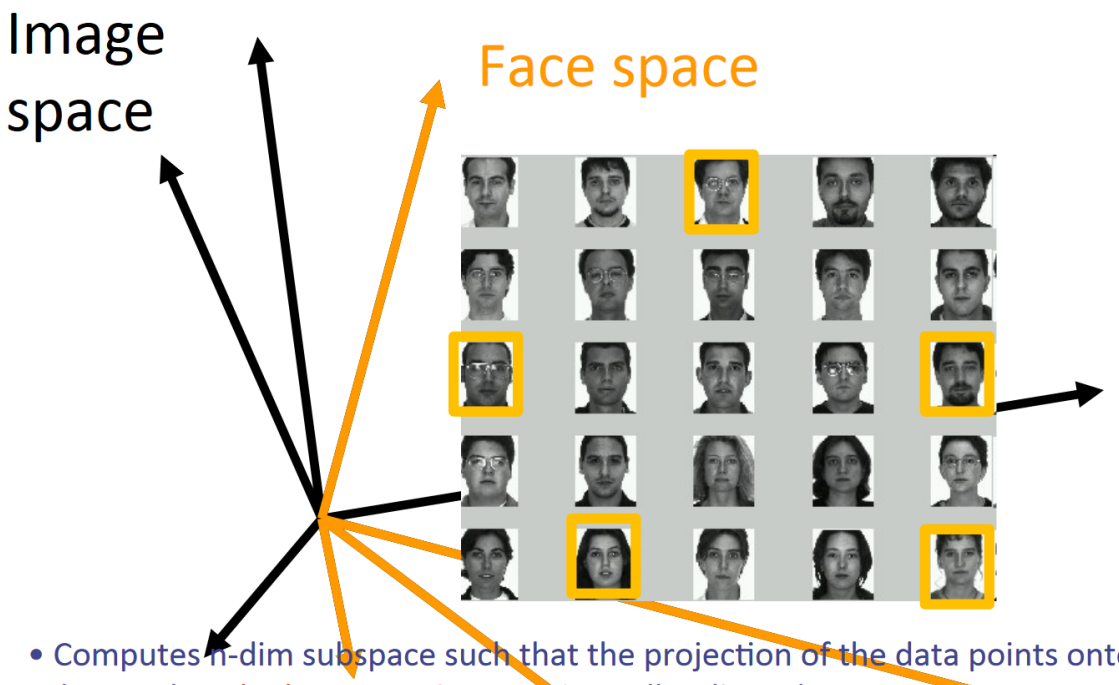
Ths space of faces



- An image is a point in a high dimensional space
  - If represented in grayscale intensity, an N x M image is a point in R^{NM}
  - E.g. 100x100 image = 10,000 dim
- However, relatively few high dimensional vectors correspond to valid face images
- We want to effectively model the subspace of face images

Slide credit: Chuck Dyer, Steve Seitz, Nishino

Ths space of faces



- Computes n-dim subspace such that the projection of the data points onto the subspace has the largest variance among all n-dim subspaces.
- Maximize the scatter of the training images in face space

The key idea

Technische Hochschule Rosenheim

- So, compress them to a low--dimensional subspace that captures key appearance characteristics of the visual images.
- Use **PCA** for estimating the sub-space (dimensionality reduction)
- Compare two faces by projecting the images into the subspace and measuring the EUCLIDEAN distance between them.
- Assume that most face images lie on a low-dimensional subspace determined by the first k (k<<d) directions of maximum variance
- Use PCA to determine the vectors or “eigenfaces” that span that subspace
- Represent all face images in the dataset as linear combinations of eigenfaces

Eigenface Algorithm

Technische Hochschule Rosenheim

1. Align training images $x_1, x_2, \ldots, x_i$ (Note: that each image is formulated into a long vector)
2. Compute average face (mean)
3. Compute the difference image (the centered data matrix)
4. Compute the covariance matrix
5. Compute the eigenvectors of the covariance matrix Σ
6. Compute each training image xi ‘s projections
7. Visualize the estimated training face xi

Stochastic Nearest Neighbor Embeddings

Stochastic Nearest Neighbor Embeddings

Technische Hochschule Rosenheim

- Stochastic Nearest Neighbor Embeddings is an alternative dimensionality reduction algorithm
  - t-SNE is the most popular one
- PCA tries to find a global structure
  - Low dimensional subspace
  - Can lead to local inconsistencies
  - Far away point can become nearest neighbors
- t-SNE tries to preserve local structure
  - Low dimensional neighborhood should be the same as original neighborhood.
- Unlike PCA almost only used for visualization

Embeddings of high dimensional data

Technische Hochschule Rosenheim

- High dimensional data is hard to visualize and work with.
- Embeddings to low dimensional spaces help us visualize data and improve the performance of data analysis algorithms.
  - PCA (linear dimension reduction)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)

> The key is to embed the data while still preserving important structures.

Stochastic Nearest Neighbor Embeddings (SNE)

Technische Hochschule Rosenheim

Given high-dimensional datapoints $x_1, \ldots, x_n \in \mathbb{R}^m$ represent intrinsic structure of the data in 1D, 2D, or 3D (for visualization)

How?

- Model neighboring datapoint pairs based on the distance of those points in the high-dimensional space
- Find a **probability distribution** of the pairwise distances in the low dimensional space that is as close as possible as the original probability distribution

> Main Idea: Map near points of high-dimensional space to a near position in low-dimensional space.

SNE - Idea

Technische Hochschule Rosenheim

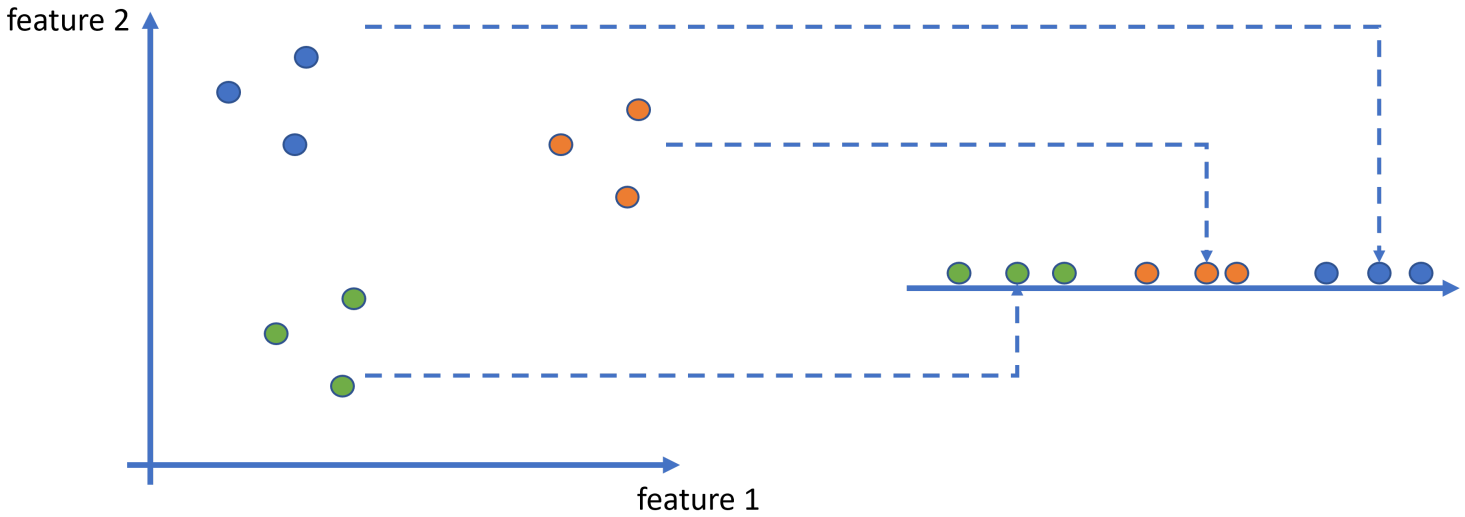
> Main Idea: Map near points of high-dimensional space to a near position in low-dimensional space.

1. Measure euclidean distance in high dim and convert to probability of picking a point as a neighbor (similarity is proportional to probability)
    - use Gaussian distribution for density of each point
2. In low-dimension do same as under 1. in low dimensionality (for t-SNE use t-distribution - has heavier tails)
3. Minimize the difference of the conditional probabilities (KL-divergence)

- **Input space**: building P using probability distributions so that picking two similar points is much more probable than picking two points which are well far apart.
- **Output space**: building Q using probability distributions to judge distances between points

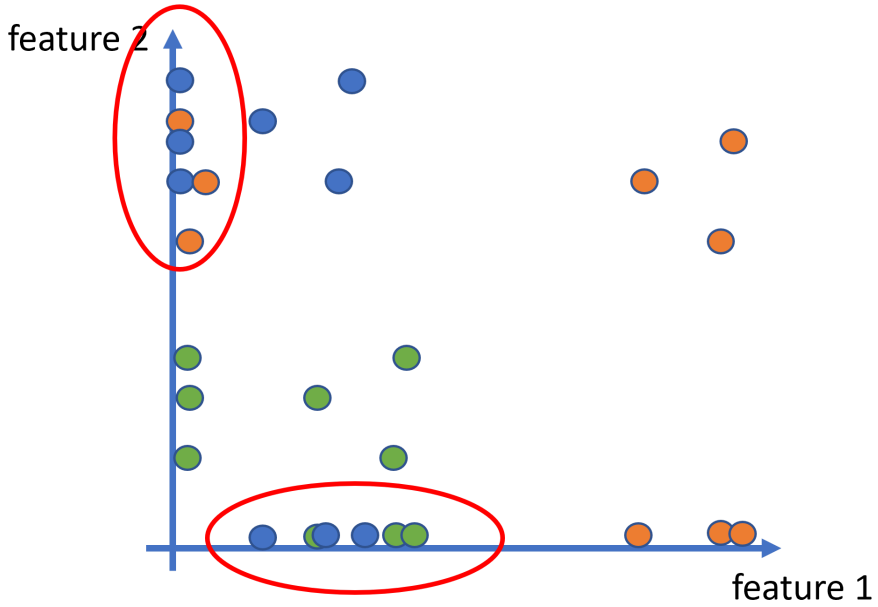
What t-SNE does?

Find a way to project data into a low dimensional space (e.g. 1-D) so that clustering in the high dimensional space (e.g. 2-D) is preserved!



Problem

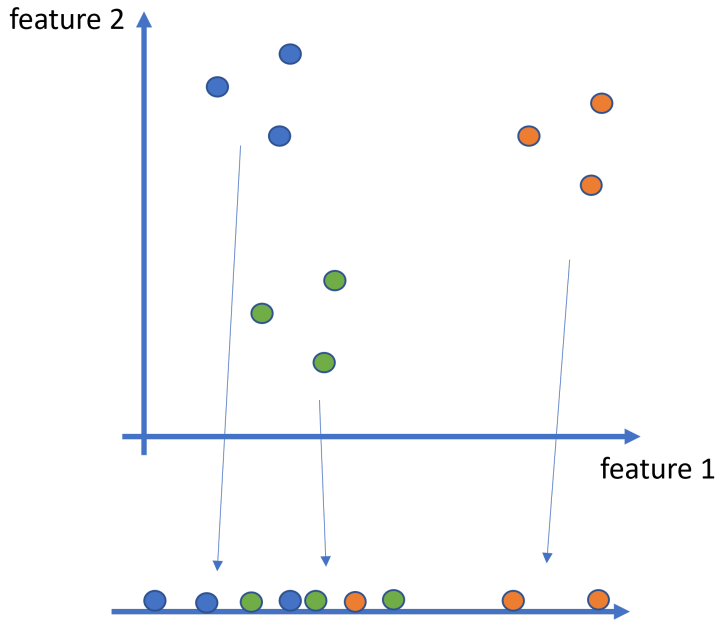
- In case data is simply projected one axes, it does not preserve clustering!



t-SNE Algorithm

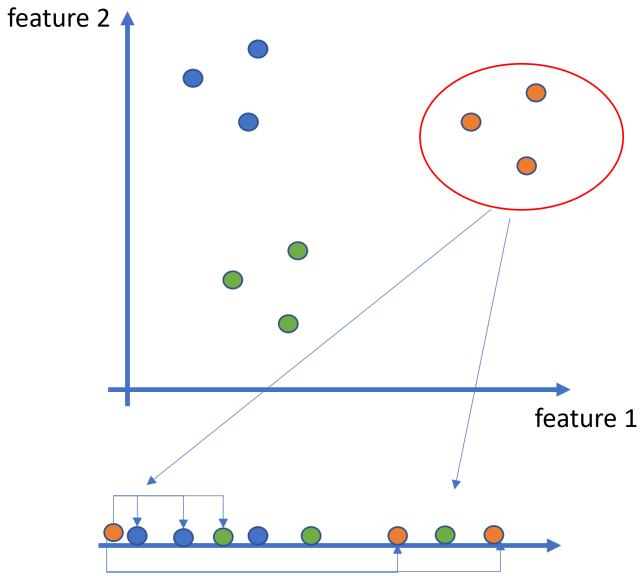
Technische Hochschule Rosenheim

- Start with original scattered data and map data points to lower dimension in random order.
- t-SNE moves data points until they are clustered correctly.



t-SNE Algorithm

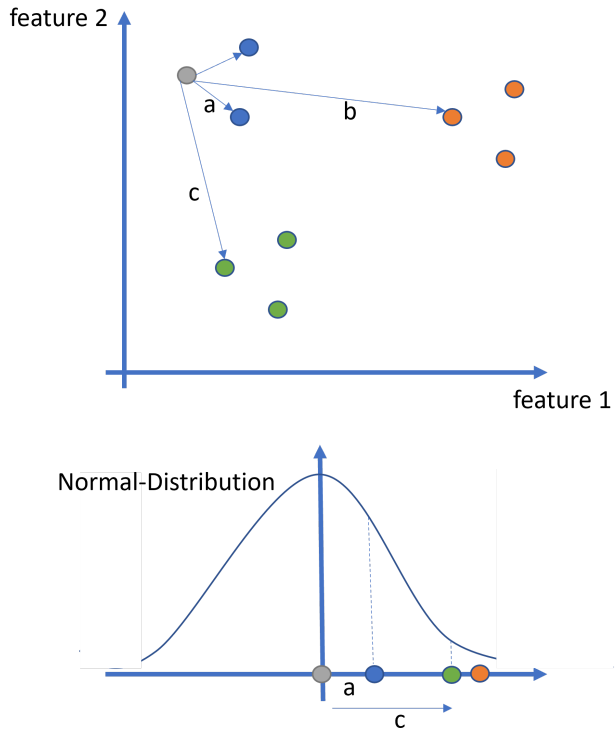
- Move a point closer to its cluster points: These points *attract*!
- Other points who are not part of the cluster push pack: These points *repel*!



How can we measure the strengt for *attract* and *repel*?

t-SNE Algorithm step 1

- Determine the "similarity" of all points in the scatter plot.
- Measure the distance of one point to all other points
- Calculate for given distance the normal distribution value
- Using normal distribution means that similar points have **high value** and others have **low value**



t-SNE Algorithm step 2

- Normalize scores (sum to 1) to make clusters comparable

=> make scores of relatively tight clusters same as for relatively loose clusters!

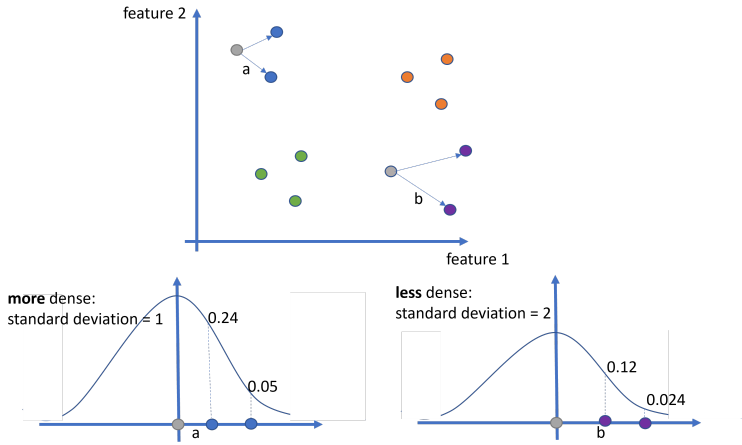
$$\frac{Score}{\sum scores} = scaled_score$$

blue:

$$\frac{0.24}{0.24 + 0.5} = 0.82$$

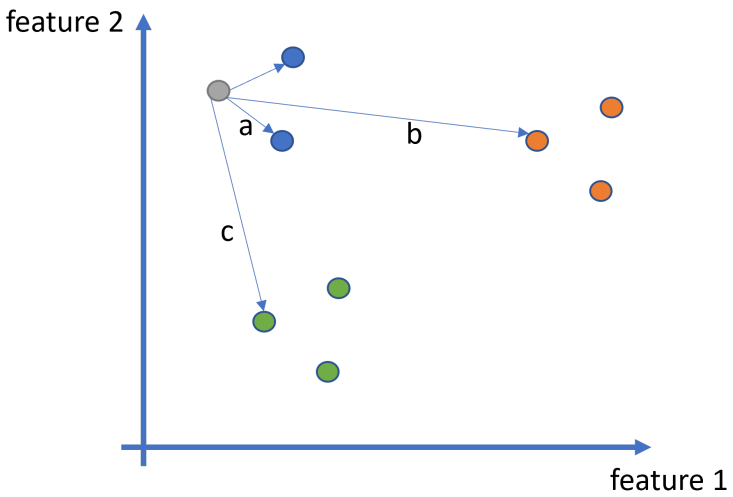
violett:

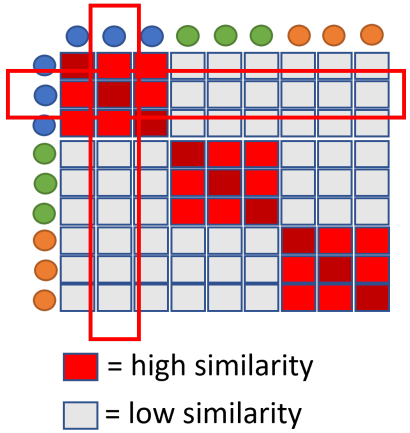
$$\frac{0.12}{0.12 + 0.024} = 0.82$$



t-SNE Algorithm step 3

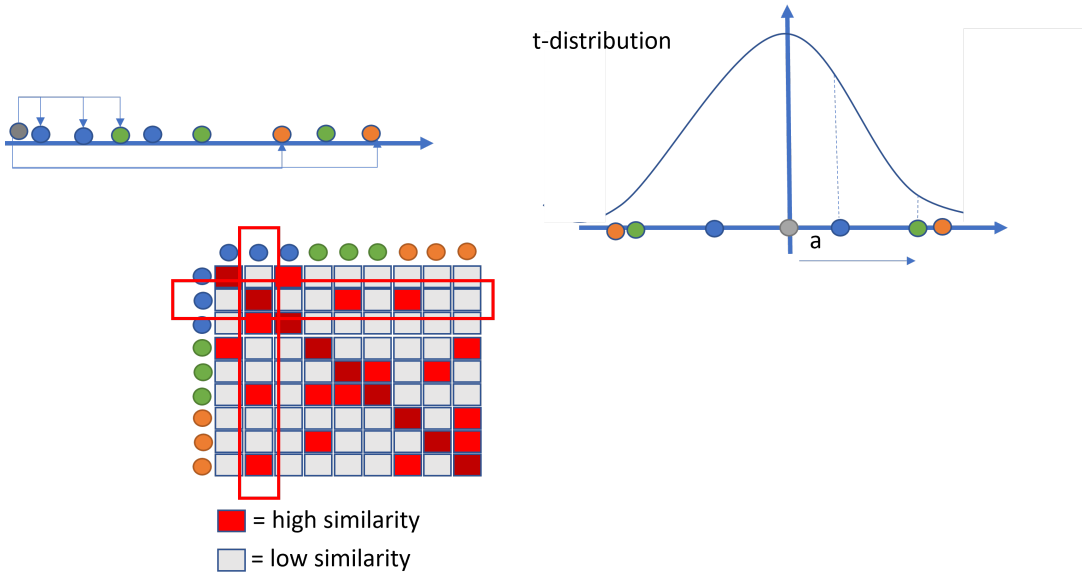
- Similarity matrix of all data points
- t-SNE sets value for same points to 0!





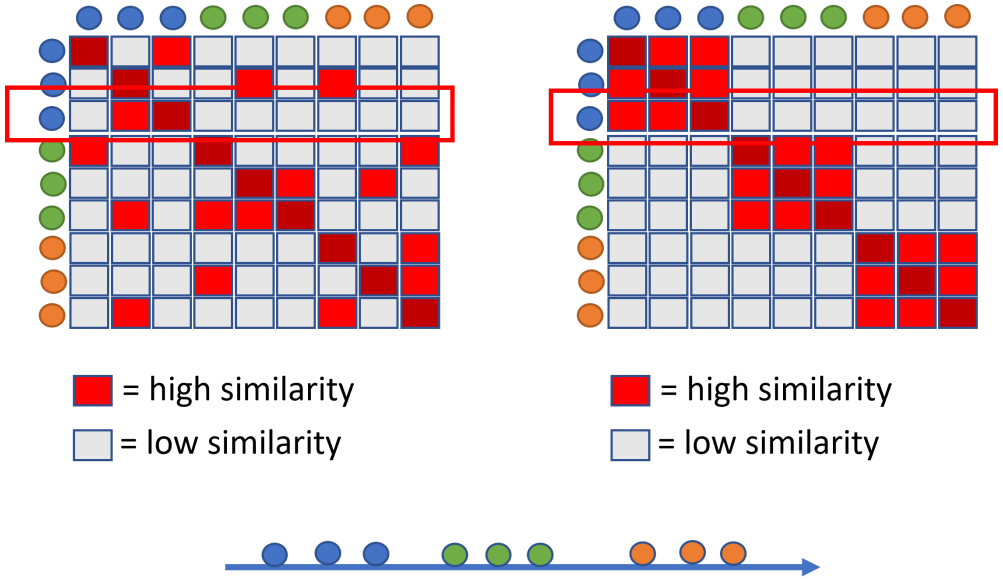
t-SNE Algorithm step 4

- Repeat for low-dimension (here:1-D)
  - Calculate scores of each point now base on t-distribution
  - Create matrix of similarity score … BUT …



t-SNE Algorithm step 5

- t-SNE tries to create a matrix of scores which looks like the original by moving each point a bit at a time
- Each step t-SNE chooses a direction that makes the matrix on the left more like the matrix on the right



Input Space

Technische Hochschule Rosenheim

- Stochastic Neighbor Embedding (SNE) starts by converting the high dimensional Euclidean distances between data-points into conditional probabilities that represent similarities.
- Given a set of high-dimensional data-points $x_1, \ldots, x_n$
- The similarity of data-point x_j to x_i is the conditional probability p, that x_i would pick x_j as its neighbor.
- We assume that neighbors are picked in proportion to their probability density under a Gaussian centered at x_i with variance σ_i:

$$p_j^i = \frac{exp(-||x_i - x_j||^2 / 2\sigma_i^2)}{\sum exp(-||x_i - x_k||^2 / 2\sigma_i^2)}$$

with $p_{i|i} = 0$

Output Space

Technische Hochschule Rosenheim

- SNE expects an output space distribution:

$$q_j^i = \frac{exp(-||y_i - y_j||^2)}{\sum exp(-||y_i - y_k||^2)}$$

- t-SNE expects an output space t-distribution:

$$q_j^i = \frac{1 + (-||y_i - y_j||^2)}{\sum -||y_i - y_k||^2}$$

with $q_{i|i} = 0$

> Goal: Optimize q to be close to p

Crowding Problem

Technische Hochschule Rosenheim

- In high dimension we have more room, points can have a lot of different neighbors
- In 2D a point can have a few neighbors at distance one all far from each other - what happens when we embed in 1D?
- This is the ”crowding problem” - we don’t have enough room to accommodate all neighbors.
- This is one of the biggest problems with SNE.
- **t-SNE solution**: Change the Gaussian in Q to a heavy tailed distribution.
    - if Q changes slower, we have more ”wiggle room” to place points at (we will cover this in the exercise!)

Kullback-Leibler divergence

Technische Hochschule Rosenheim

Definition Kullback-Leibler (KL) divergence

Given two discrete probability distributions P and Q the KL divergence from Q to P is defined to be

$$KL(P||Q) = -\sum_i P_i log \frac{P_i}{Q_i}$$

- We assume P_i and Q_i are both greater than zero for all i.
- The KL divergence is in fact the expectation of the logarithmic difference between probabilities P and Q, where the expectation is taken using the probabilities P.

Cost Function

Technische Hochschule Rosenheim

SNE minimizes the sum of KL divergences over all data-points using a gradient descent method.

The cost function C is given by

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p^i_j log \frac{p^i_j}{q^i_j}$$

where P_i represents the conditional probability distribution over all given data-points given x_i, and Q_i represents the conditional probability distribution over all map points given y_i.

- The SNE cost function focuses on retaining the local structure of the data in the map.

Perplexity

Technische Hochschule Rosenheim

- SNE selects the variance σ_i of the Gaussian that is centered over each x_i.
- The value of σ_i induces a probability distribution P_i over all the other data-points.
- SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity, specified by the user.

The perplexity is defined as

$$Perp(P_i) = 2^{H(P_i)}$$

where H is the *Shannon entropy of* P_i

$$H(P_i) = -\sum_j p^i_j log_2 p^i_j$$

Stochastic Gradient Descent

Technische Hochschule Rosenheim

The cost function C is given by

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_j^i log \frac{p_j^i}{q_j^i}$$

The minimization is performed using a gradient descent method. We optimize the function by taking steps proportional to the negative of the function gradient. The gradient of C w.r.t. yi,

$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_j^i - q_j^i + p_i^j - q_i^j)(y_i - y_j)$$

(We will validate this in the exercise!)

Since gradient descent does not guarantee convergence to the optimal solution (it might get stuck at a local optimum), it is common to run the optimization several times in order to find the appropriate parameters.

Summary

Technische Hochschule Rosenheim

- Today we talked about applications of PCA
  - MNIST and Eigenfaces
- and about t-SNE

> Code can be found in
>
> - Eigenfaces.ipynb

- t-SNE.ipynb
- or on GitHub or hosted on myBinder

Exercise

- On GitLab: https://inf-git.fh-rosenheim.de/aai-url/06_exercise
- Applications on PCA

Application of t-SNE