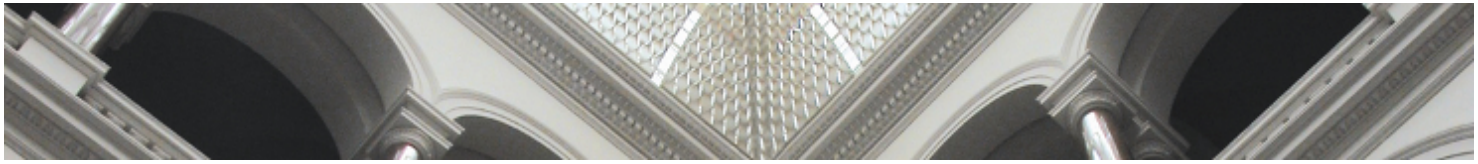
Technische Universität Berlin

Unsupervised Learning

Malte Esders, Christopher Anders, Wojciech Samek
Technische Universität Berlin - Machine Learning Group



Technische Universität Berlin

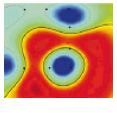
This presentation

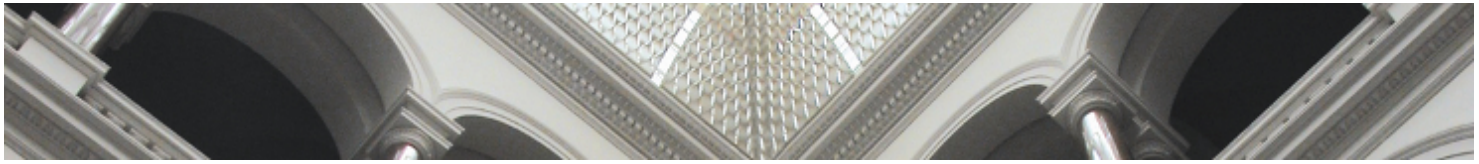
Presentation repository:

`https://github.com/Maltimore/unsupervised_learning_lecture`

Contains code to generate plots

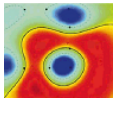
Parts not otherwise marked: BSD licensed

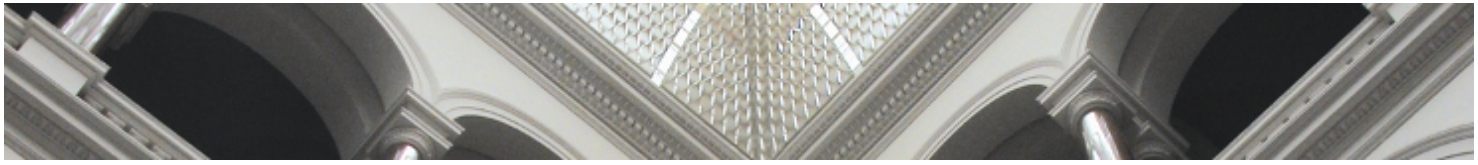




Technische Universität Berlin

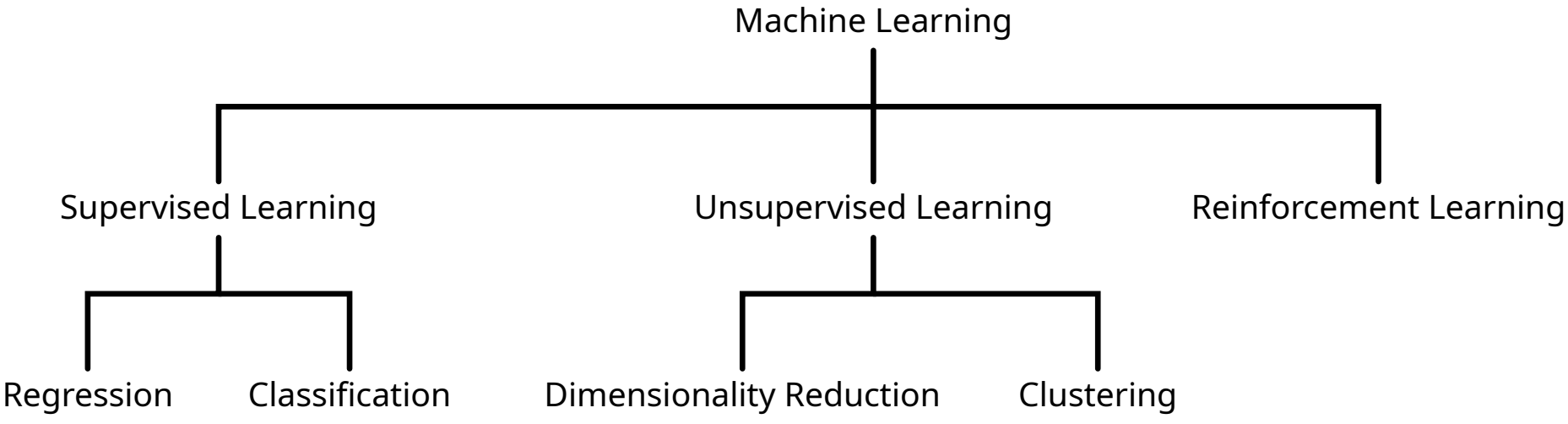
Agenda

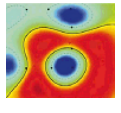


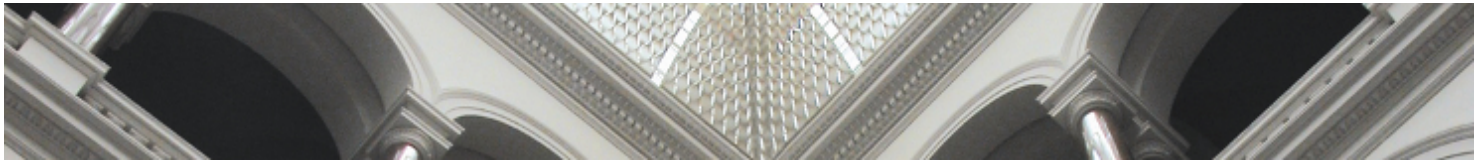


Technische Universität Berlin

Unsupervised Learning





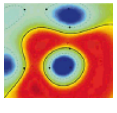


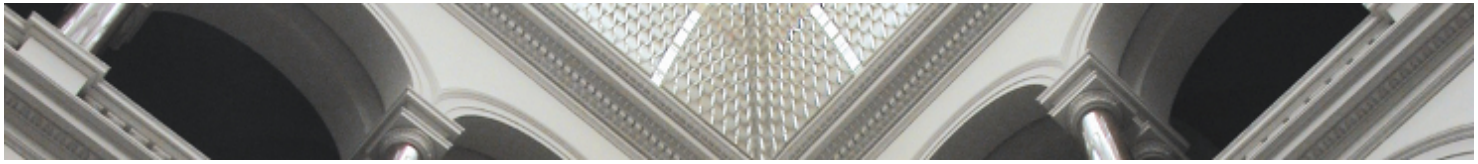
Technische Universität Berlin

Unsupervised Learning Motivation

> *When we're learning to see, nobody's telling us what the right answers are - we just look. Every so often, your mother says 'that's a dog', but that's very little information. You'd be lucky if you got a few bits of information - even one bit per second - that way. The brain's visual system has* 10^{14} *neural connections. And you only live for* 10^{9} *seconds. So it's no use learning one bit per second. You need more like* 10^{5} *bits per second. And there's only one place you can get that much information: from the input itself.*
>
> Geoffrey Hinton, 1996

Genome: $3 * 10^{9}$ base-pairs (A, T, C, G), with 2 bits per location has $6 * 10^{9}$ bits.





Technische Universität Berlin

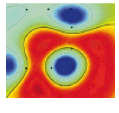
Unsupervised Learning Motivation

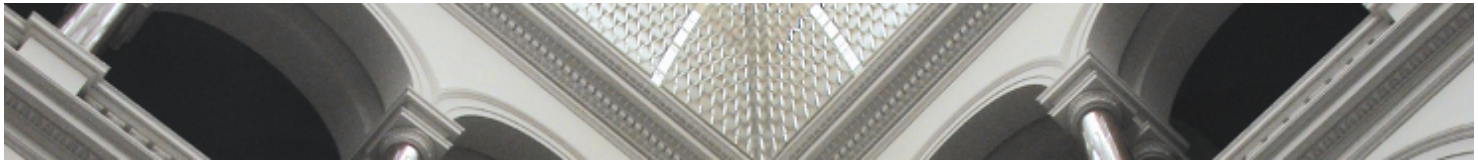
> *When we're learning to see, nobody's telling us what the right answers are - we just look. Every so often, your mother says 'that's a dog', but that's very little information. You'd be lucky if you got a few bits of information - even one bit per second - that way. The brain's visual system has* 10^{14} *neural connections. And you only live for* 10^{9} *seconds. So it's no use learning one bit per second. You need more like* 10^{5} *bits per second. And there's only one place you can get that much information: from the input itself.*
>
> Geoffrey Hinton, 1996

Genome: $3 * 10^9$ base-pairs (A, T, C, G), with 2 bits per location has $6 * 10^9$ bits.

Yann LeCun's cake analogy (2016):

- Unsupervised learning is the cake itself
- Supervised learning is the icing
- Reinforcement Learning is the cherry on top





Technische Universität Berlin

Unsupervised Learning Motivation

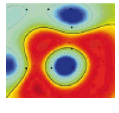
> *When we're learning to see, nobody's telling us what the right answers are - we just look. Every so often, your mother says 'that's a dog', but that's very little information. You'd be lucky if you got a few bits of information - even one bit per second - that way. The brain's visual system has* 10^{14} *neural connections. And you only live for* 10^{9} *seconds. So it's no use learning one bit per second. You need more like* 10^{5} *bits per second. And there's only one place you can get that much information: from the input itself.*
>
> Geoffrey Hinton, 1996

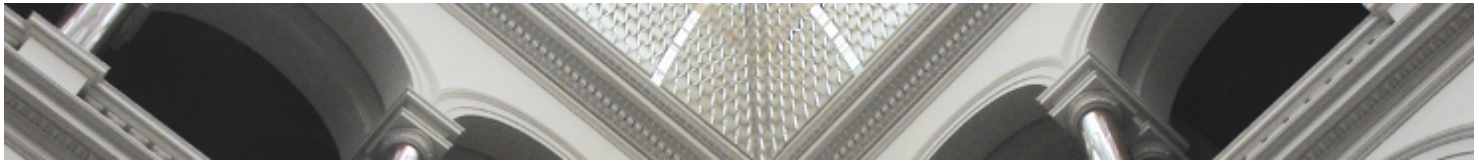
Genome: $3 * 10^9$ base-pairs (A, T, C, G), with 2 bits per location has $6 * 10^9$ bits.

Yann LeCun's cake analogy (2016):

- Unsupervised learning is the cake itself
- Supervised learning is the icing
- Reinforcement Learning is the cherry on top

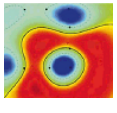
ChatGPT is unsupervised learning!

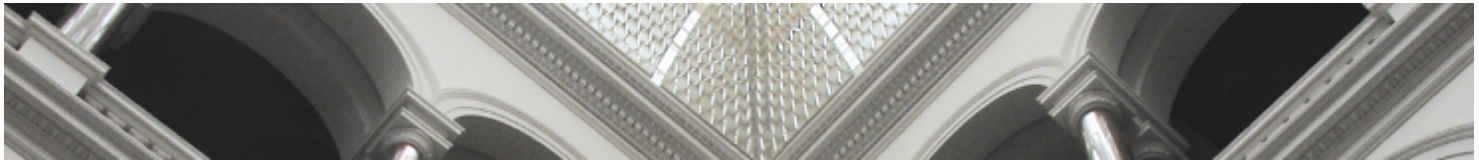




Technische Universität Berlin

Agenda

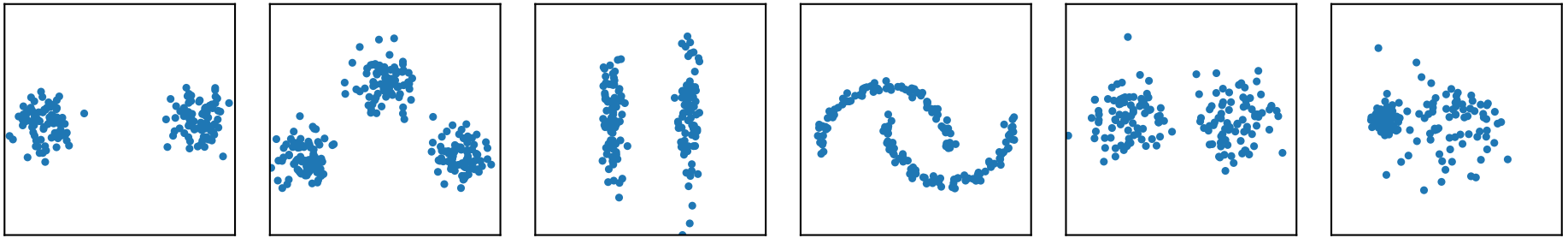


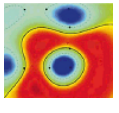


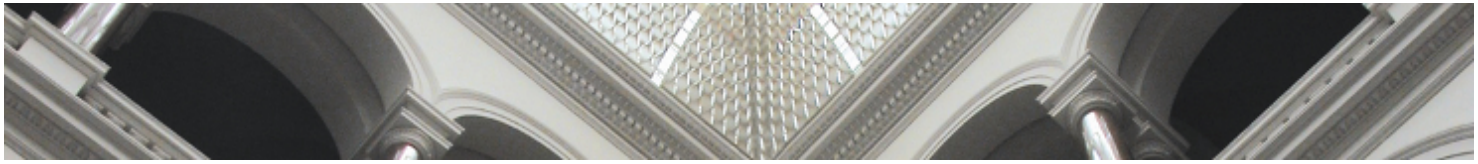
Technische Universität Berlin

Clusters - Intuition

Without labels:



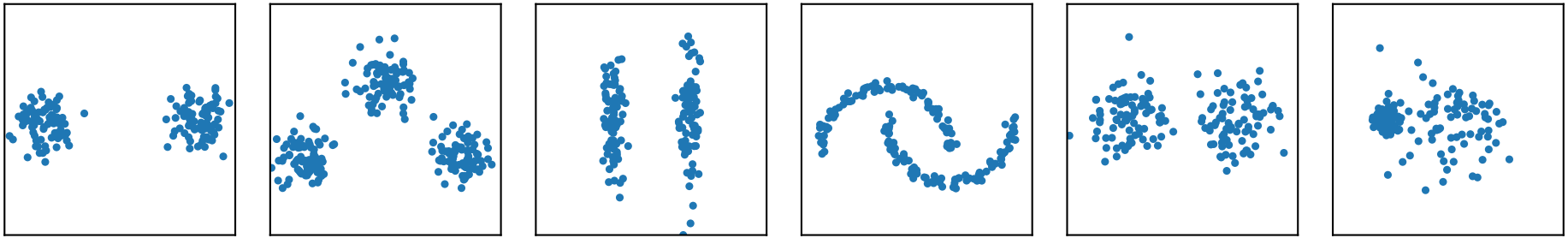




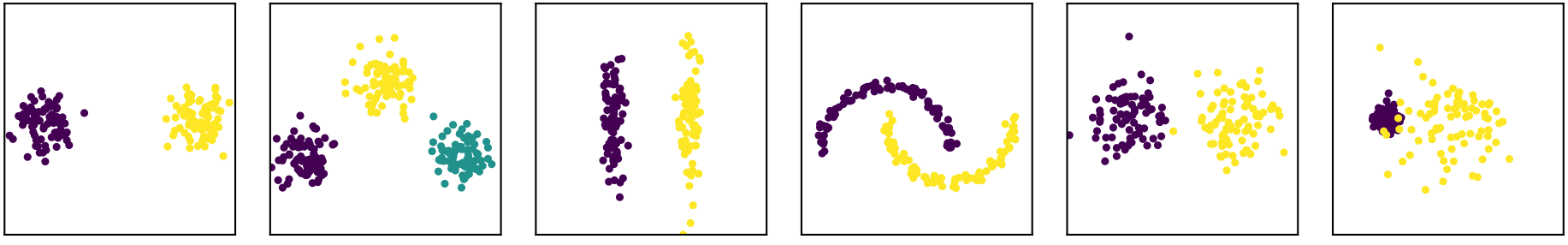
Technische Universität Berlin

Clusters - Intuition

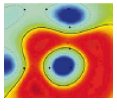
Without labels:

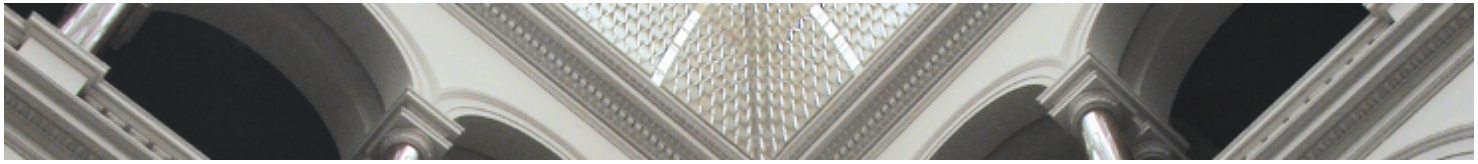


"True" clusters:



We do not have the "true" clusters with real data!





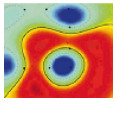
Technische Universität Berlin

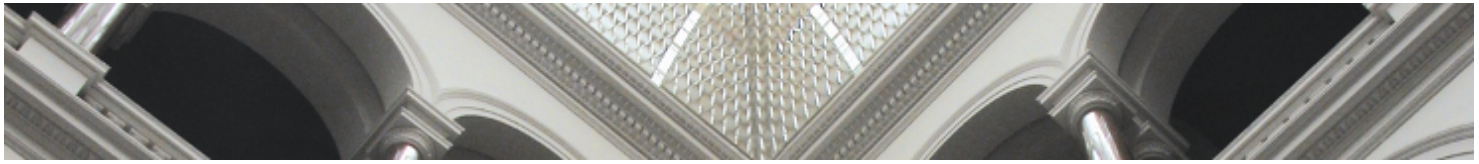
Clusters - Intuition

Clustering is an ill-posed problem: no definition of a cluster.

Some approaches:

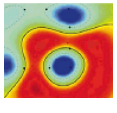
- samples within a cluster should be “similar”, samples between clusters “different”
- clusters are regions of densely populated feature space
- samples in the same cluster come from the same underlying process

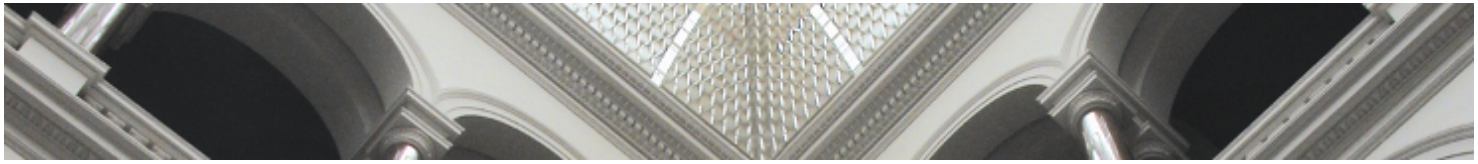




Technische Universität Berlin

Agenda





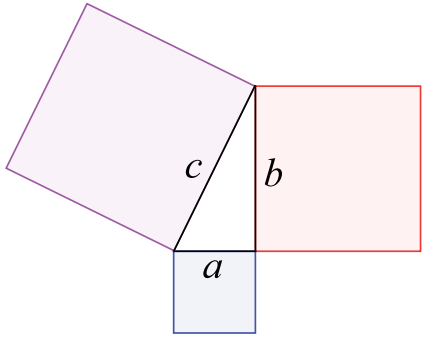
Technische Universität Berlin

Euclidean distance

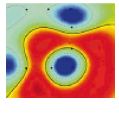
Pythagoras formula:

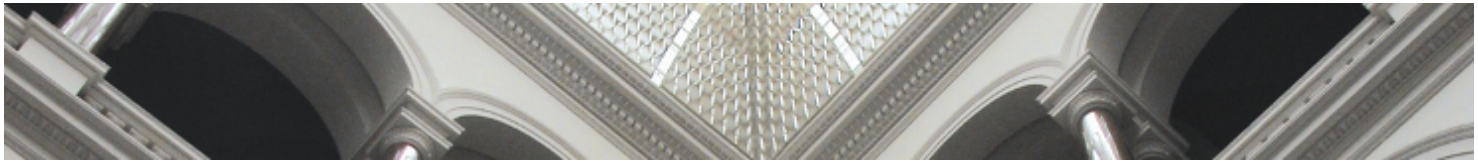
$$c^2 = a^2 + b^2$$

$$c = \sqrt{a^2 + b^2}$$



Wikipedia Pythagorean theorem



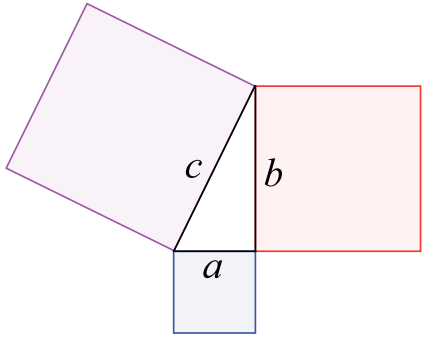


Euclidean distance

Pythagoras formula:

$$c^2 = a^2 + b^2$$

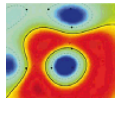
$$c = \sqrt{a^2 + b^2}$$

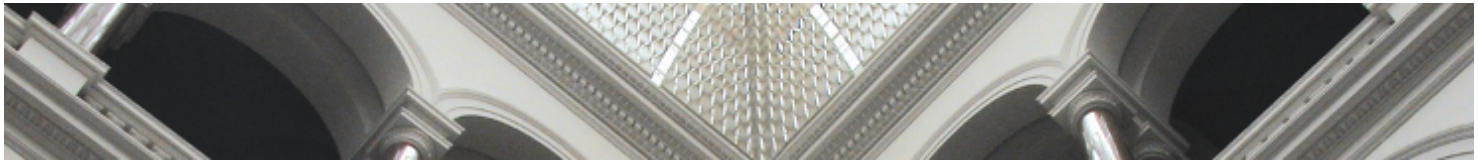


Wikipedia Pythagorean theorem

Euclidean distance notation:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\Delta x_1^2 + \Delta x_2^2}$$





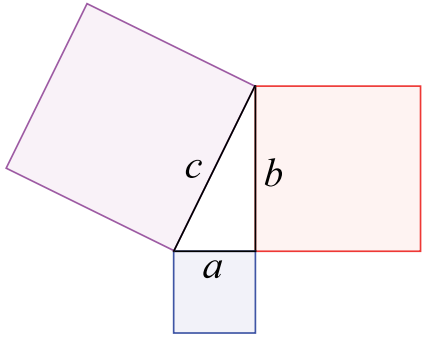
Technische Universität Berlin

Euclidean distance

Pythagoras formula:

$$c^2 = a^2 + b^2$$

$$c = \sqrt{a^2 + b^2}$$



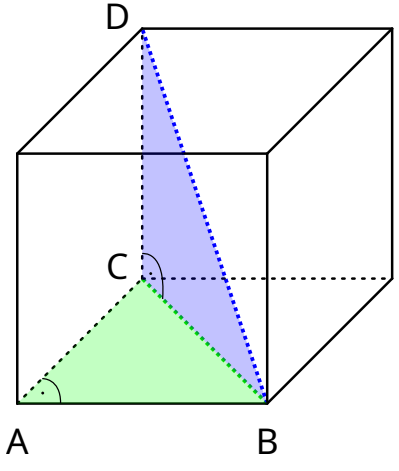
Wikipedia Pythagorean theorem

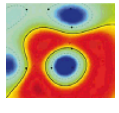
Euclidean distance notation:

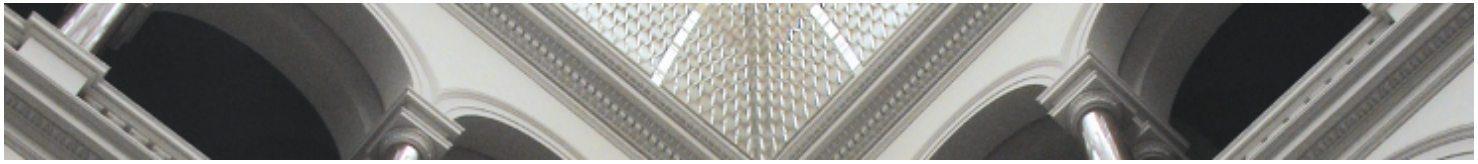
$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\Delta x_1^2 + \Delta x_2^2}$$

Euclidean distance 3-D:

$$\overline{BC} = \sqrt{\overline{AC}^2 + \overline{AB}^2}$$





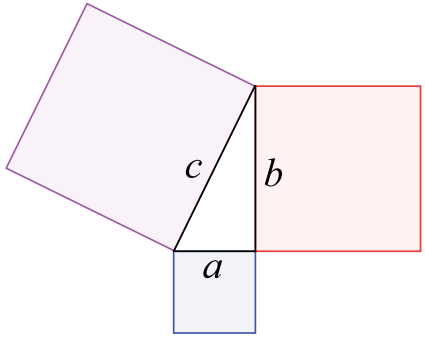


Euclidean distance

Pythagoras formula:

$$c^2 = a^2 + b^2$$

$$c = \sqrt{a^2 + b^2}$$



Wikipedia Pythagorean theorem

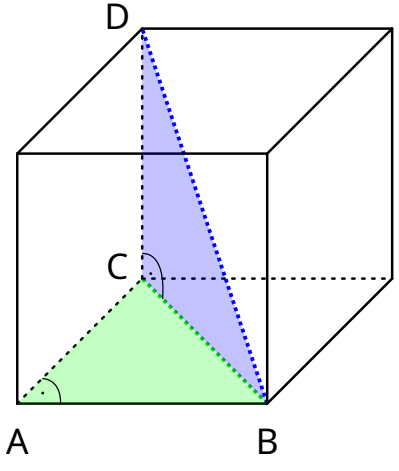
Euclidean distance notation:

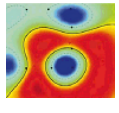
$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\Delta x_1^2 + \Delta x_2^2}$$

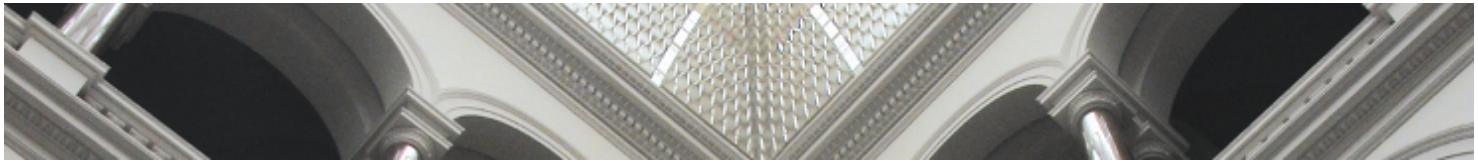
Euclidean distance 3-D:

$$\overline{BC} = \sqrt{\overline{AC}^2 + \overline{AB}^2}$$

$$\overline{BD} = \sqrt{\overline{BC}^2 + \overline{CD}^2}$$





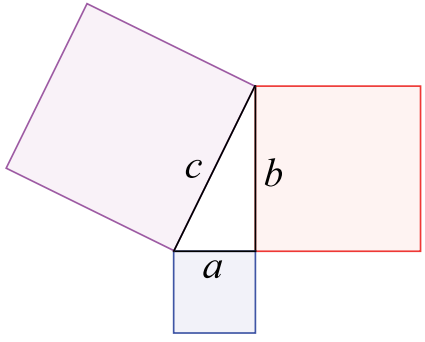


Euclidean distance

Pythagoras formula:

$$c^2 = a^2 + b^2$$

$$c = \sqrt{a^2 + b^2}$$



Wikipedia Pythagorean theorem

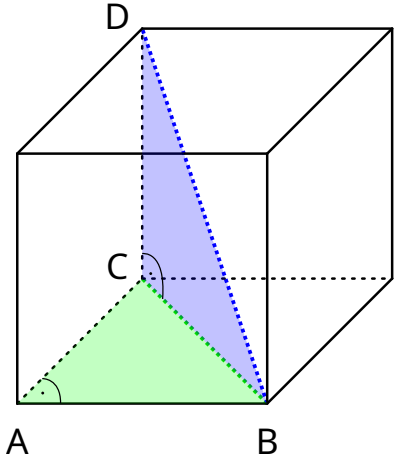
Euclidean distance notation:

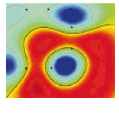
$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\Delta x_1^2 + \Delta x_2^2}$$

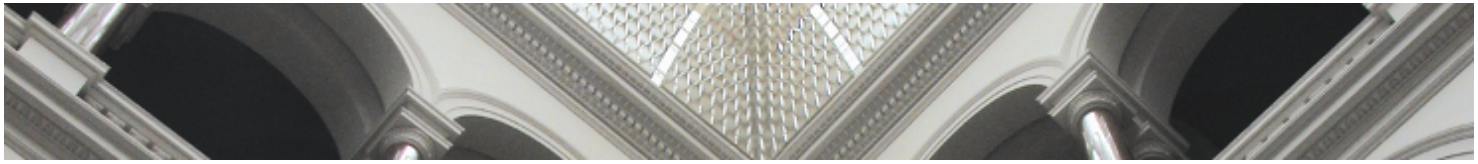
Euclidean distance 3-D:

$$\overline{BC} = \sqrt{\overline{AC}^2 + \overline{AB}^2}$$

$$\overline{BD} = \sqrt{\overline{AC}^2 + \overline{AB}^2 + \overline{CD}^2}$$







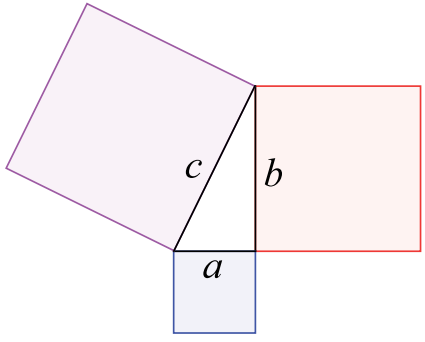
Technische Universität Berlin

Euclidean distance

Pythagoras formula:

$$c^2 = a^2 + b^2$$

$$c = \sqrt{a^2 + b^2}$$



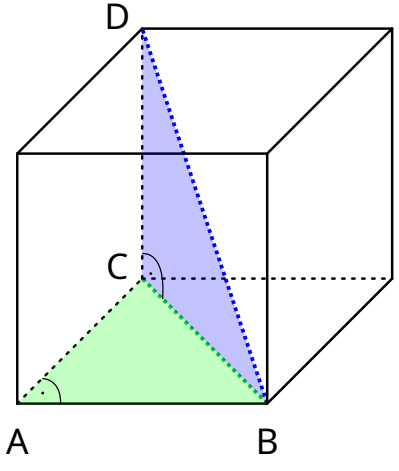
Wikipedia Pythagorean theorem

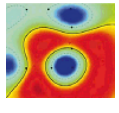
Euclidean distance notation:

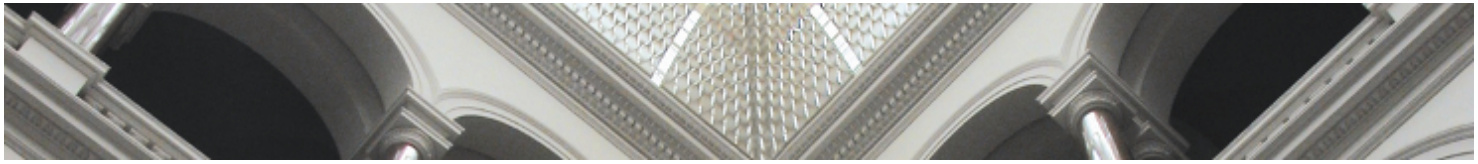
$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\Delta x_1^2 + \Delta x_2^2}$$

Euclidean distance d-D:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{d} \Delta x_i^2}$$







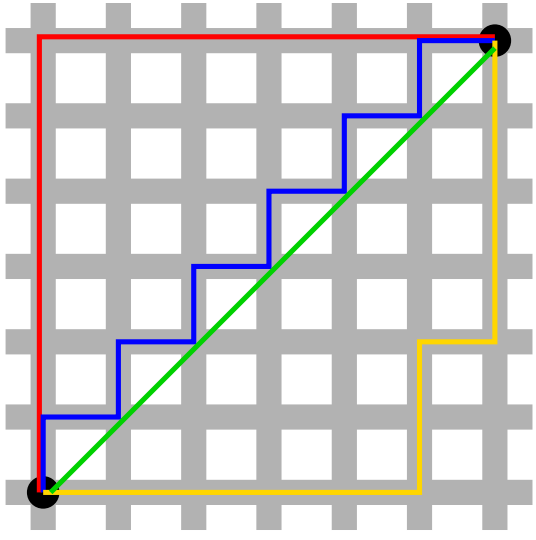
Technische Universität Berlin

Manhattan distance

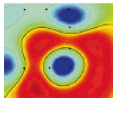
"Manhattan" / "Taxicab" distance

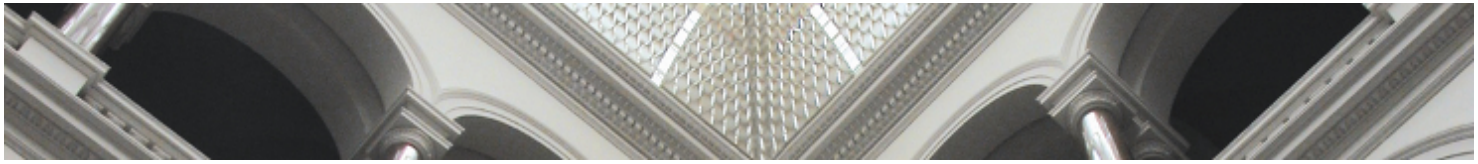
$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} |\Delta x_i|$$

Usage: LASSO regularization



Wikipedia Manhattan distance



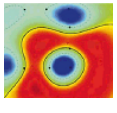


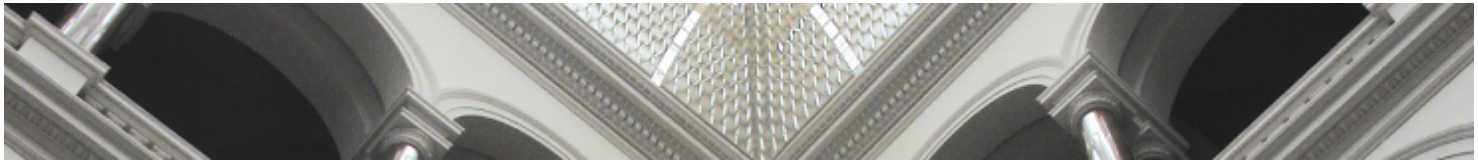
Technische Universität Berlin

Minkowski distance

Minkowski distance generalizes Euclidean and Manhattan distance:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} |\Delta x_i|^p \right)^{1/p}$$





Technische Universität Berlin

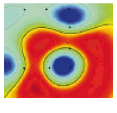
Minkowski distance

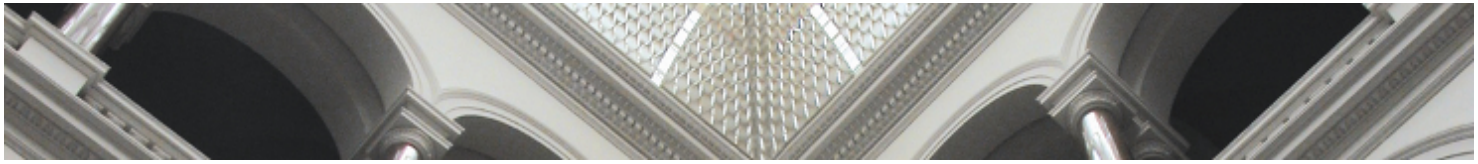
Minkowski distance generalizes Euclidean and Manhattan distance:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} |\Delta x_i|^p \right)^{1/p}$$

$p = 1$: Manhattan distance

$p = 2$: Euclidean distance





Technische Universität Berlin

Minkowski distance

Minkowski distance generalizes Euclidean and Manhattan distance:

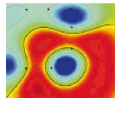
$$d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} |\Delta x_i|^p \right)^{1/p}$$

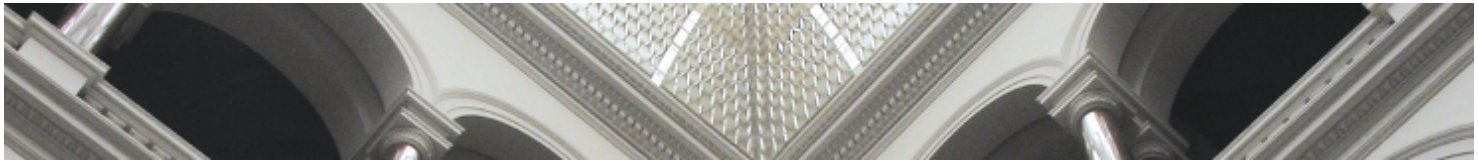
$p = 1$: Manhattan distance

$p = 2$: Euclidean distance

$p = \infty$:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \max_i |\Delta x_i|$$





Technische Universität Berlin

Minkowski distance

Minkowski distance generalizes Euclidean and Manhattan distance:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} |\Delta x_i|^p \right)^{1/p}$$

$p = 1$: Manhattan distance

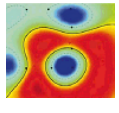
$p = 2$: Euclidean distance

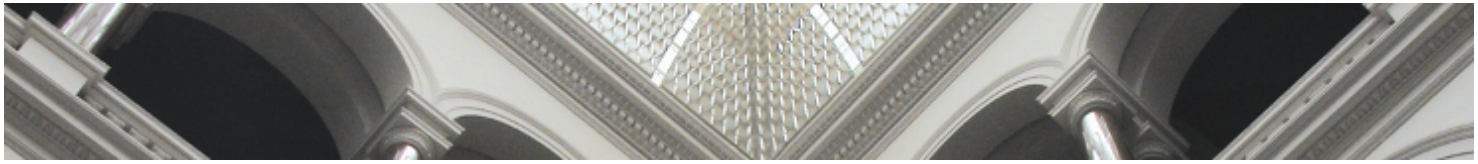
$p = \infty$:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \max_i |\Delta x_i|$$

$p \to 0$:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \min_i |\Delta x_i|$$





Technische Universität Berlin

Minkowski distance

Minkowski distance generalizes Euclidean and Manhattan distance:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} |\Delta x_i|^p \right)^{1/p}$$

$p = 1$: Manhattan distance

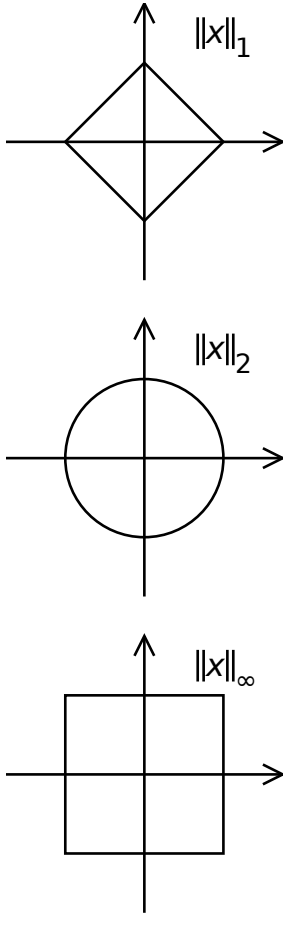
$p = 2$: Euclidean distance

$p = \infty$:

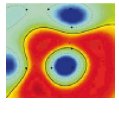
$$d(\boldsymbol{x}, \boldsymbol{y}) = \max_i |\Delta x_i|$$

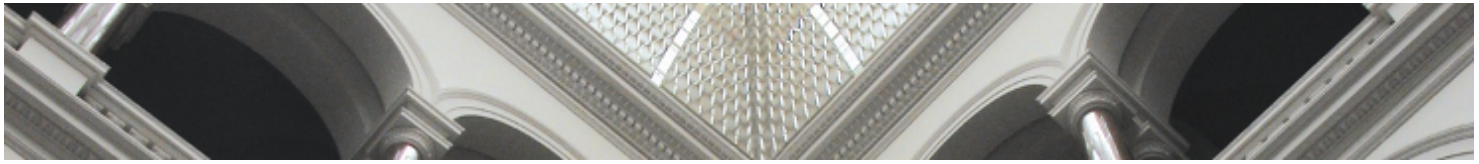
$p \to 0$:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \min_i |\Delta x_i|$$



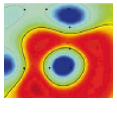
Wikipedia Norm (mathematics)

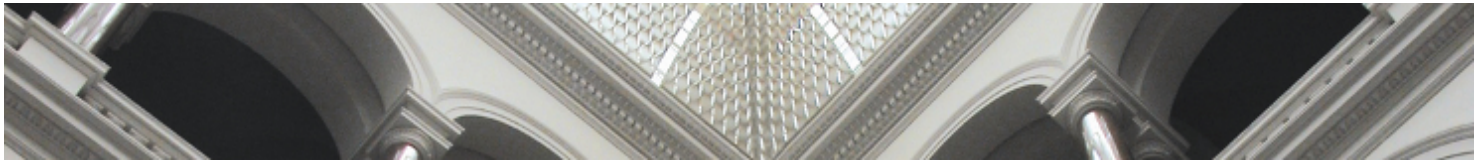




Technische Universität Berlin

Agenda



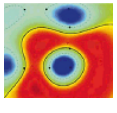


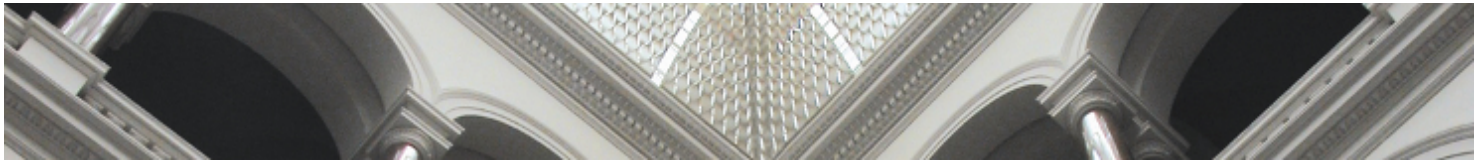
Technische Universität Berlin

Vector norm

The distance between vectors is the vector-norm of their difference:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} |\Delta x_i|^p \right)^{1/p} = ||\boldsymbol{x} - \boldsymbol{y}||_p$$





Technische Universität Berlin

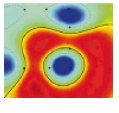
Vector norm

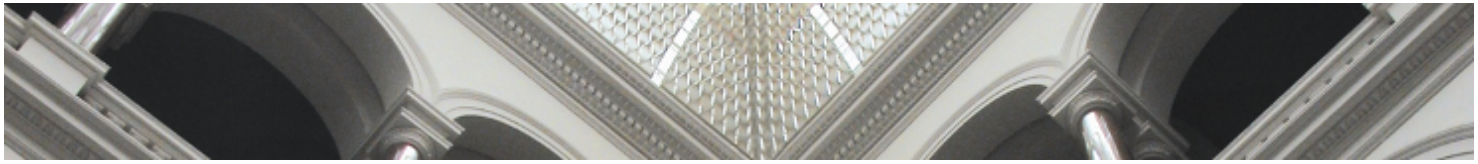
The distance between vectors is the vector-norm of their difference:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} |\Delta x_i|^p \right)^{1/p} = ||\boldsymbol{x} - \boldsymbol{y}||_p$$

For $p = 2$, the shorthand is:

$$||\boldsymbol{x} - \boldsymbol{y}||_2 = ||\boldsymbol{x} - \boldsymbol{y}||$$



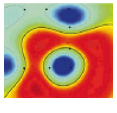


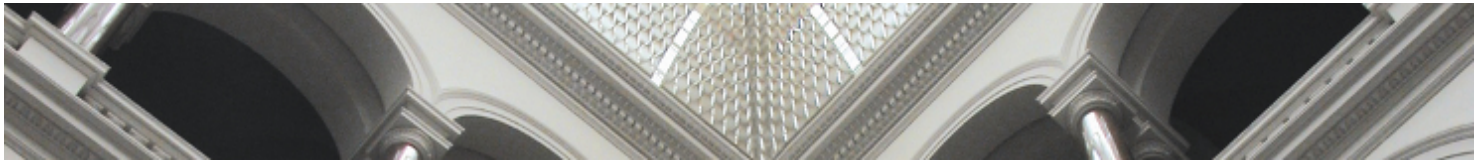
Technische Universität Berlin

Squared euclidean distance

Squared euclidean distance:

$$||\boldsymbol{x} - \boldsymbol{y}||_2^2 = \sum_{i=1}^{d} |\Delta x_i|^2$$





Technische Universität Berlin

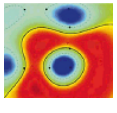
Squared euclidean distance

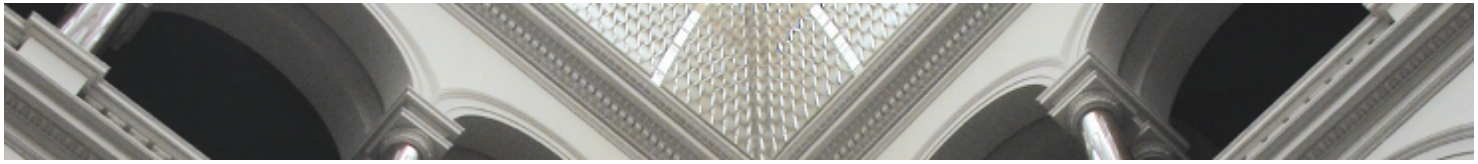
Squared euclidean distance:

$$||\boldsymbol{x} - \boldsymbol{y}||_2^2 = \sum_{i=1}^{d} |\Delta x_i|^2$$

Do not confuse

$$||\boldsymbol{x} - \boldsymbol{y}|| = ||\boldsymbol{x} - \boldsymbol{y}||_2 \neq ||\boldsymbol{x} - \boldsymbol{y}||^2 = ||\boldsymbol{x} - \boldsymbol{y}||_2^2$$





Technische Universität Berlin

Squared euclidean distance

Squared euclidean distance:

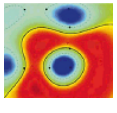
$$||\boldsymbol{x} - \boldsymbol{y}||_2^2 = \sum_{i=1}^{d} |\Delta x_i|^2$$

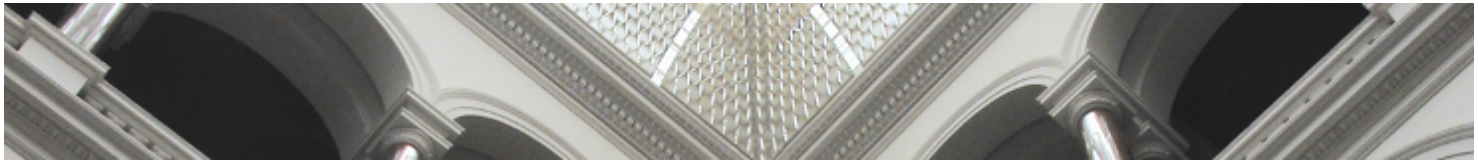
Do not confuse

$$||\boldsymbol{x} - \boldsymbol{y}|| = ||\boldsymbol{x} - \boldsymbol{y}||_2 \neq ||\boldsymbol{x} - \boldsymbol{y}||^2 = ||\boldsymbol{x} - \boldsymbol{y}||_2^2$$

Squared Euclidean distance:

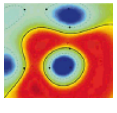
- not a true metric (no triangle inequality)
- overemphasizes big distances (and vice versa)
- makes some optimization problems converge better

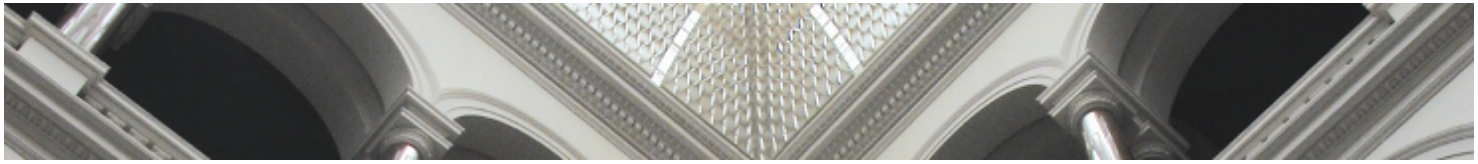




Technische Universität Berlin

Agenda

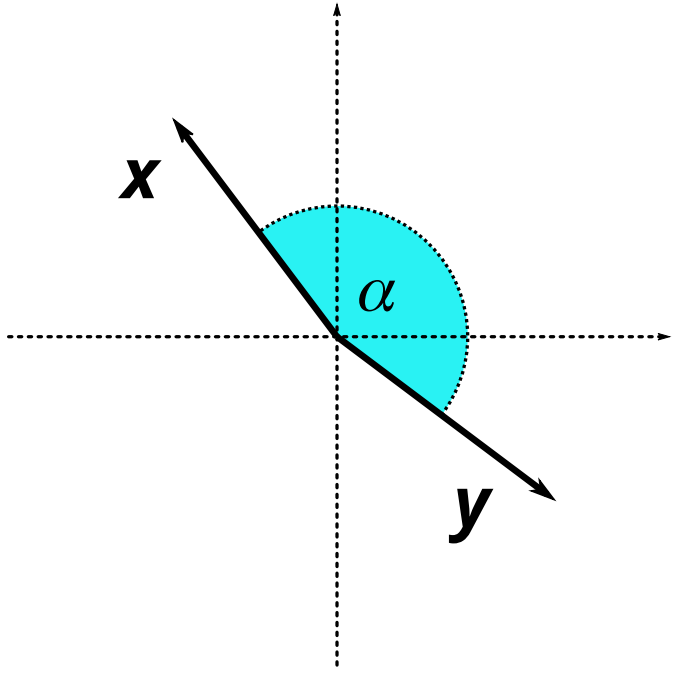


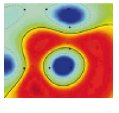


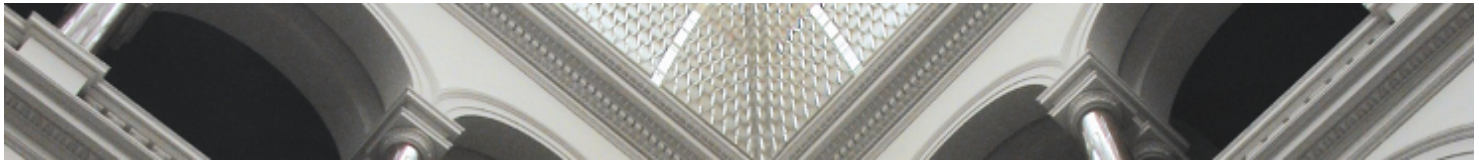
Technische Universität Berlin

Cosine distance

Cosine distance only compares angles between vectors.







Technische Universität Berlin

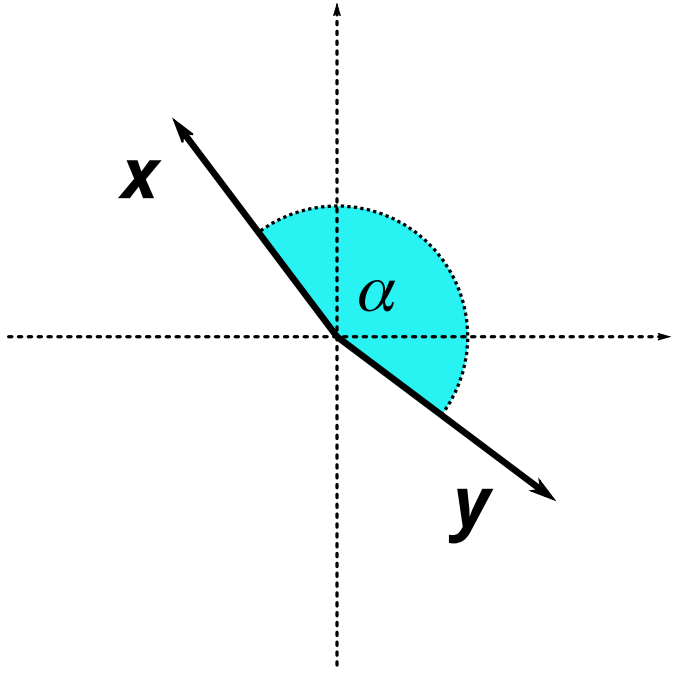
Cosine distance

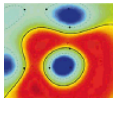
Cosine distance only compares angles between vectors.

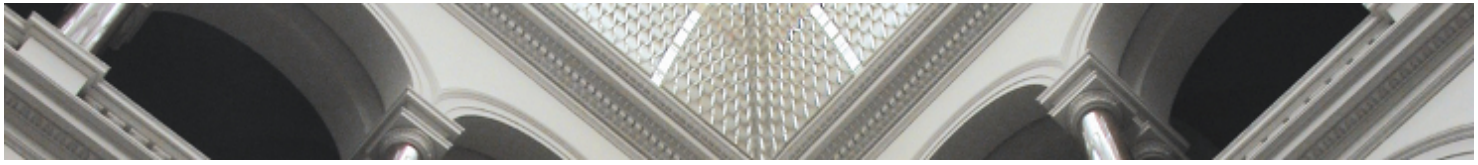
Cosine definition:

$$\boldsymbol{x}^\mathsf{T}\boldsymbol{y} = |\boldsymbol{x}||\boldsymbol{y}|cos(\alpha)$$

$$cos(\alpha) = \frac{\boldsymbol{x}^\mathsf{T}\boldsymbol{y}}{|\boldsymbol{x}||\boldsymbol{y}|}$$







Technische Universität Berlin

Cosine distance

Cosine distance only compares angles between vectors.

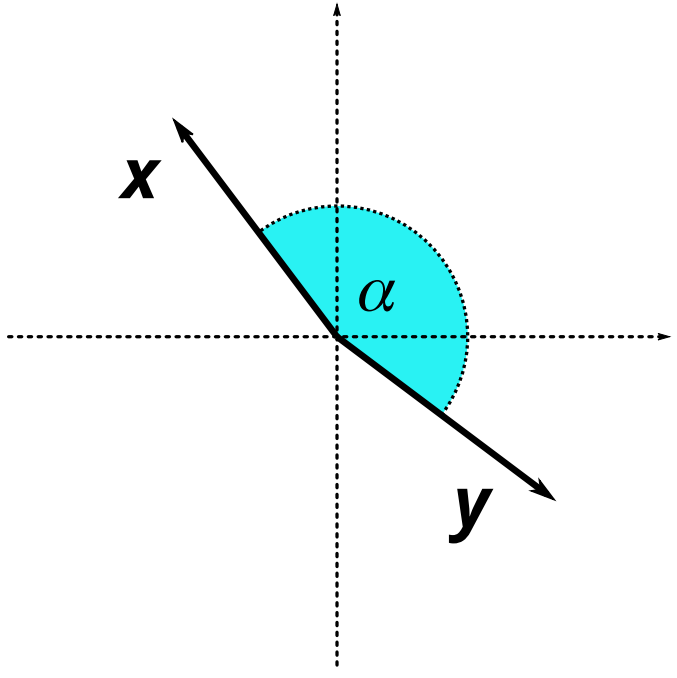
Cosine definition:

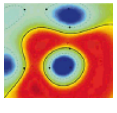
$$\boldsymbol{x}^\mathsf{T}\boldsymbol{y} = |\boldsymbol{x}||\boldsymbol{y}|cos(\alpha)$$

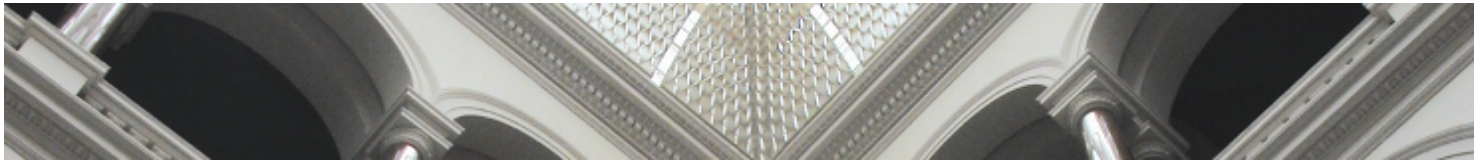
$$cos(\alpha) = \frac{\boldsymbol{x}^\mathsf{T}\boldsymbol{y}}{|\boldsymbol{x}||\boldsymbol{y}|}$$

Cosine distance:

$$d(\boldsymbol{x}, \boldsymbol{y}) = 1 - \frac{\boldsymbol{x}^\mathsf{T}\boldsymbol{y}}{|\boldsymbol{x}||\boldsymbol{y}|}$$

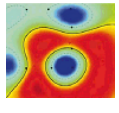


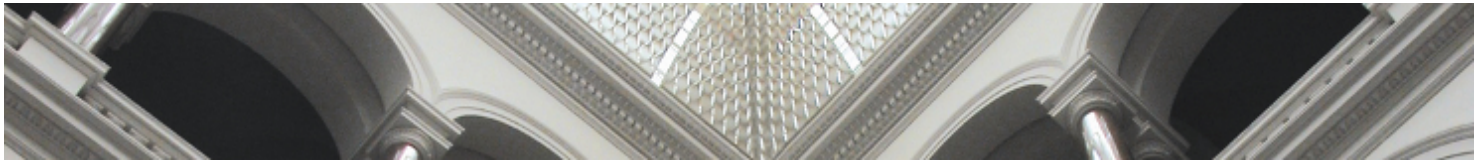




Technische Universität Berlin

Agenda





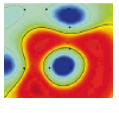
Technische Universität Berlin

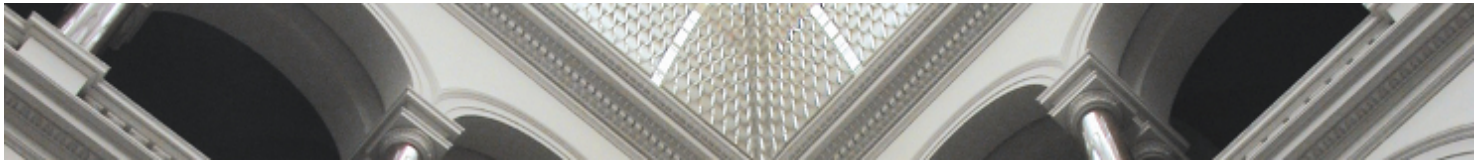
Different scales

Measurements can be at different scales (for instance cm, kg etc.)

We can normalize:

$$z = \frac{x - \mu}{\sigma}$$





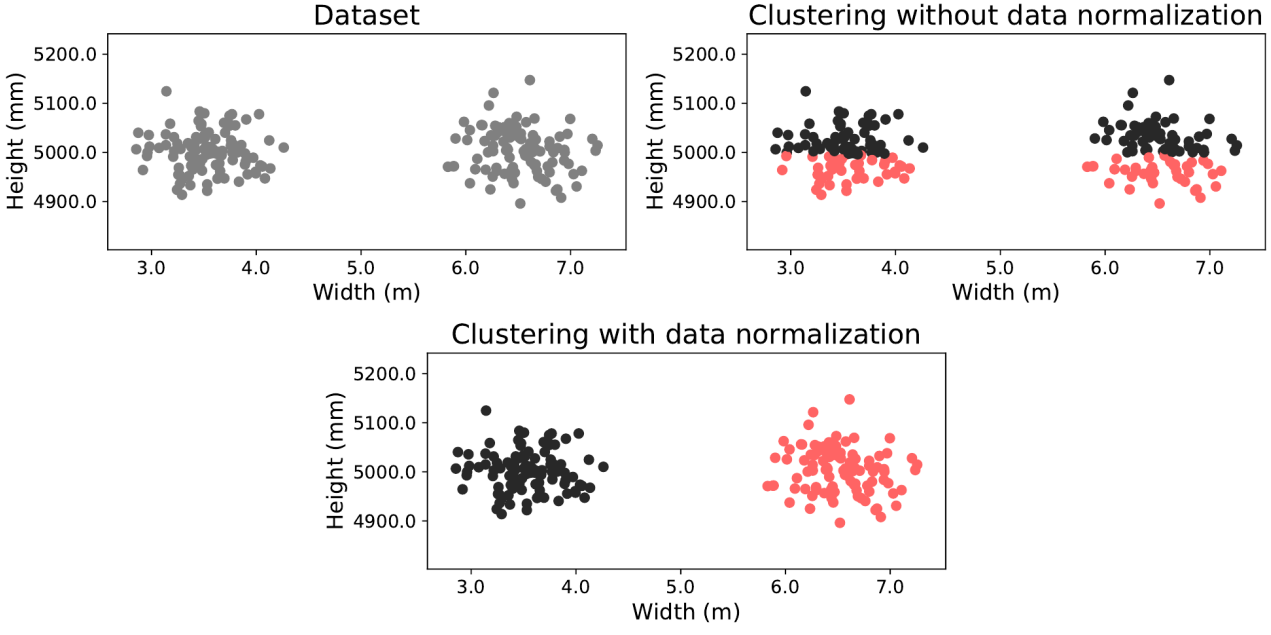
Technische Universität Berlin

Different scales

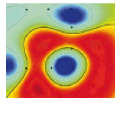
Measurements can be at different scales (for instance cm, kg etc.)

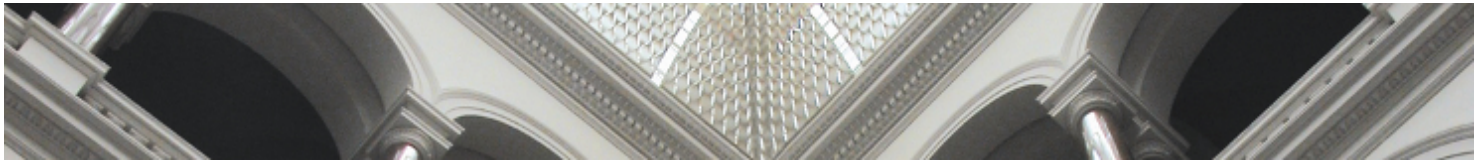
We can normalize:

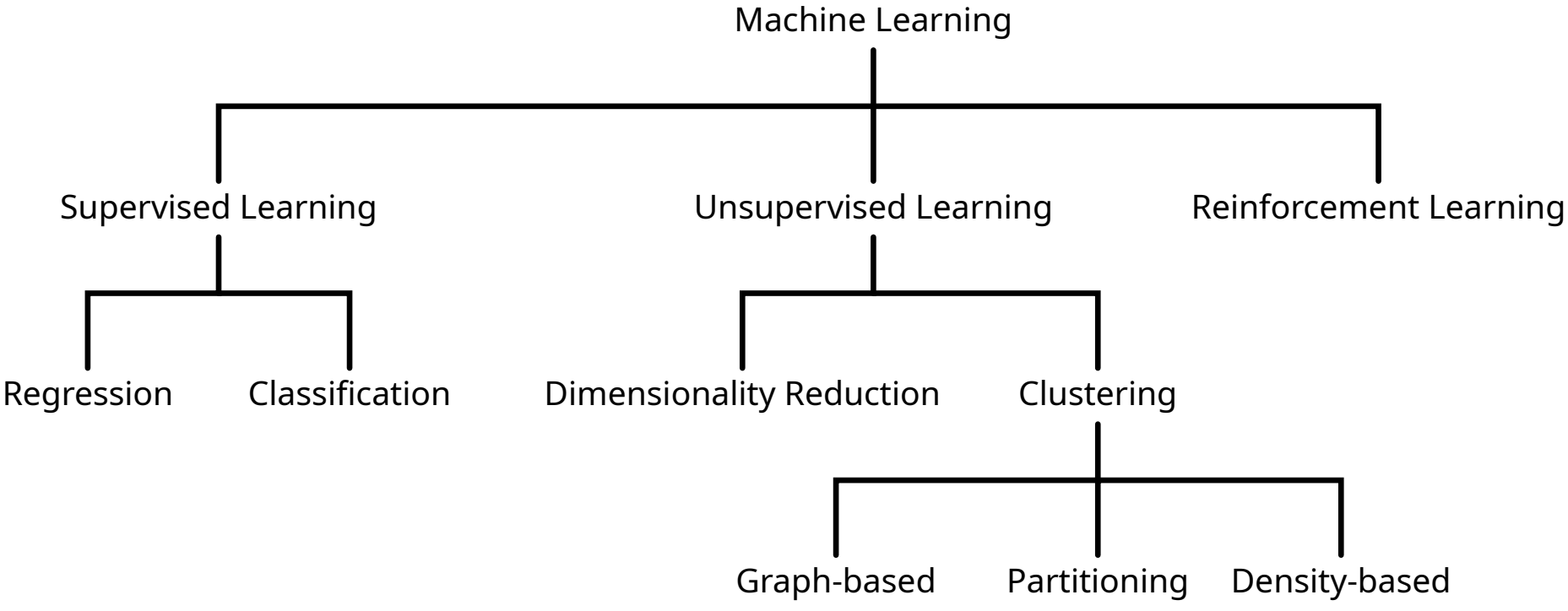
$$z = \frac{x - \mu}{\sigma}$$

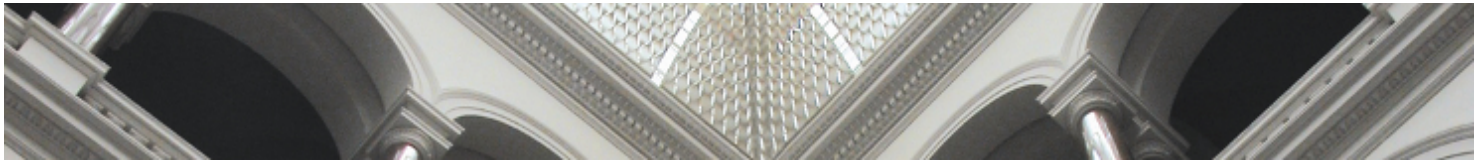


Guérin et al. 2017



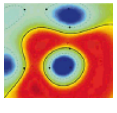


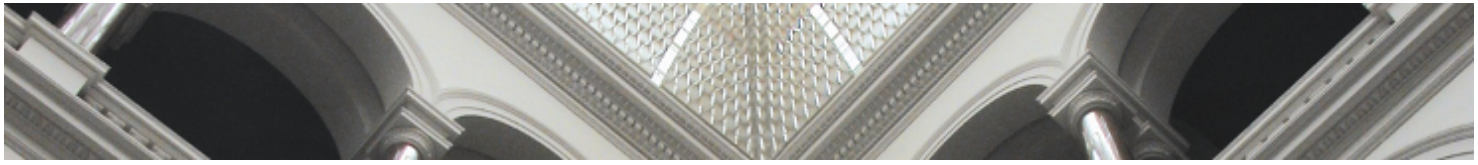

Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Regression
Classification
Dimensionality Reduction
Clustering
Graph-based
Partitioning
Density-based



Technische Universität Berlin

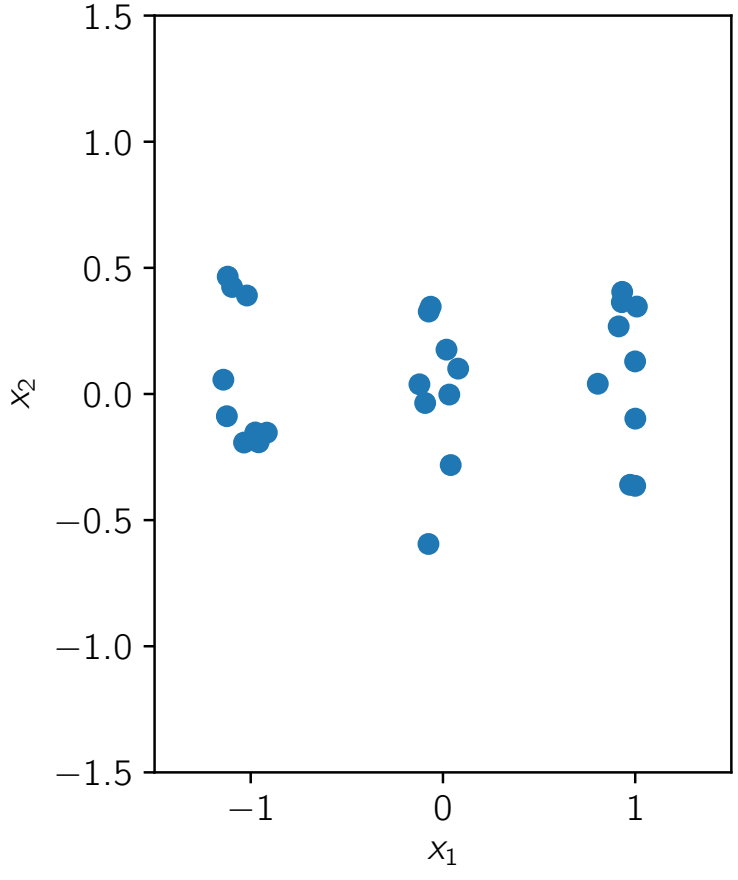
Agenda

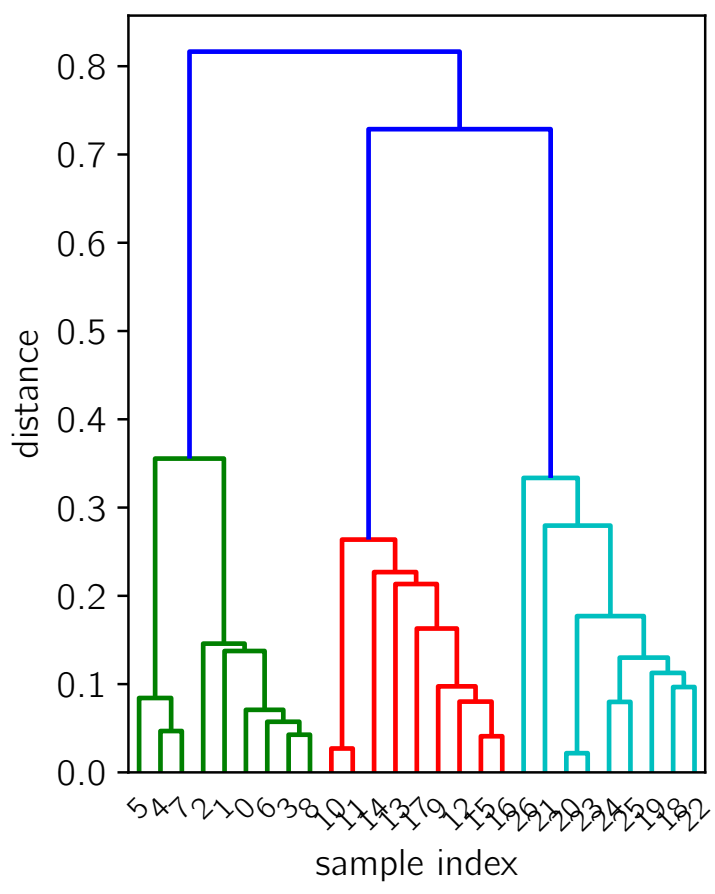


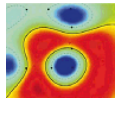


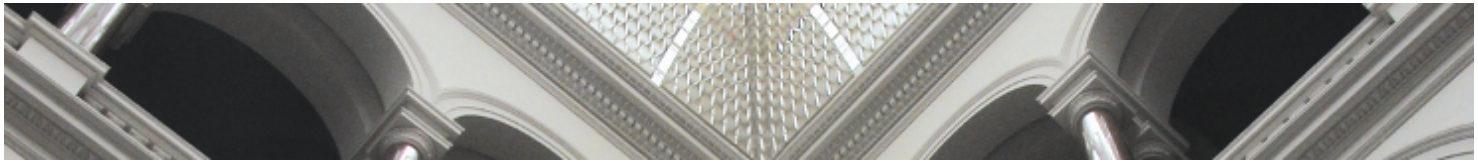
Technische Universität Berlin

Dendrogram with Euclidean distance





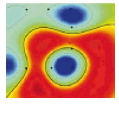


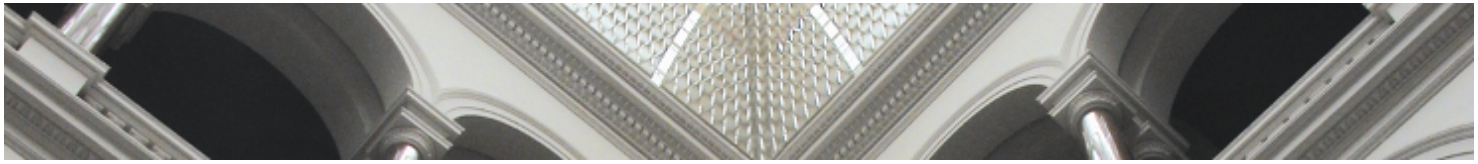


Technische Universität Berlin

Hierarchical Clustering

– graph-based (no embedding needed!)

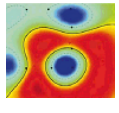


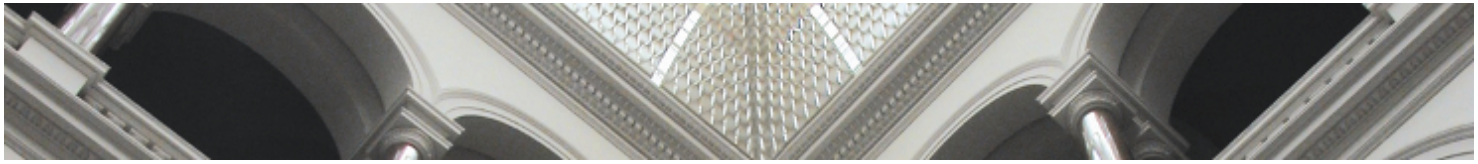


Technische Universität Berlin

Hierarchical Clustering

- graph-based (no embedding needed!)
- builds a hierarchy of clusters, either top down (“divisive”) or bottom up (“agglomerative”)

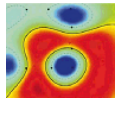


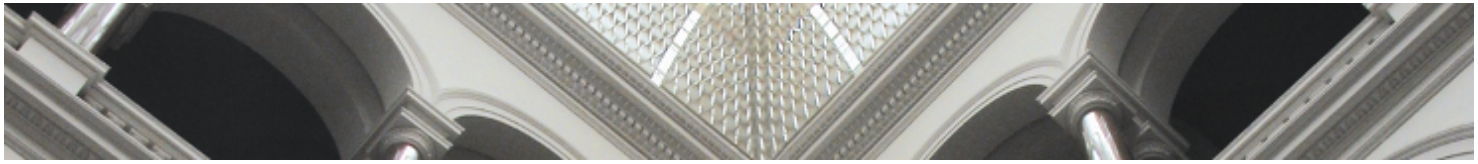


Technische Universität Berlin

Hierarchical Clustering

- graph-based (no embedding needed!)
- builds a hierarchy of clusters, either top down (“divisive”) or bottom up (“agglomerative”)
- intuitive

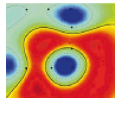


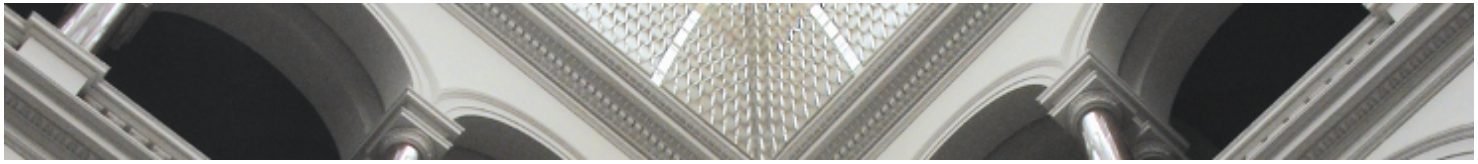


Technische Universität Berlin

Hierarchical Clustering

- graph-based (no embedding needed!)
- builds a hierarchy of clusters, either top down (“divisive”) or bottom up (“agglomerative”)
- intuitive
- suggests numbers of clusters

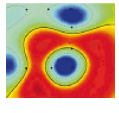


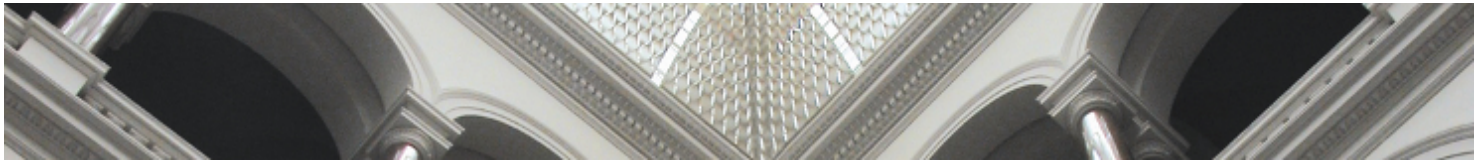


Technische Universität Berlin

Hierarchical Clustering

- graph-based (no embedding needed!)
- builds a hierarchy of clusters, either top down (“divisive”) or bottom up (“agglomerative”)
- intuitive
- suggests numbers of clusters
- arbitrary cluster shapes

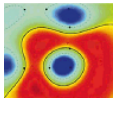


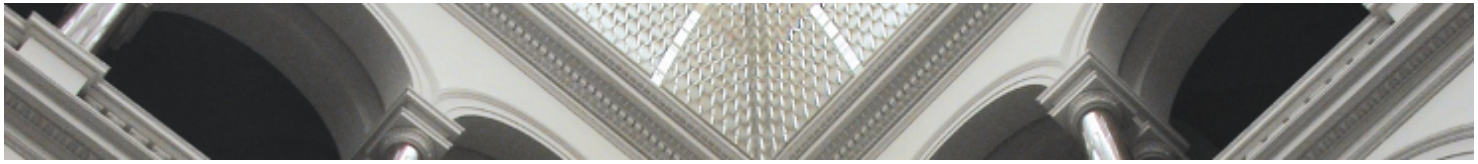


Technische Universität Berlin

Hierarchical Clustering

- graph-based (no embedding needed!)
- builds a hierarchy of clusters, either top down (“divisive”) or bottom up (“agglomerative”)
- intuitive
- suggests numbers of clusters
- arbitrary cluster shapes
- sensitive to noise





Agglomerative clustering algorithm:

Let $\mathcal{D}_i$ be the sets containing the samples in cluster i.

choose n_{final} clusters

initialize $\mathcal{D}_i$ as singleton clusters

initialize $n_{clusters} = n_{samples}$

while $n_{clusters} > n_{final}$ **do**

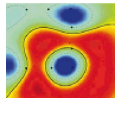
  find clusters i and j with smallest distance

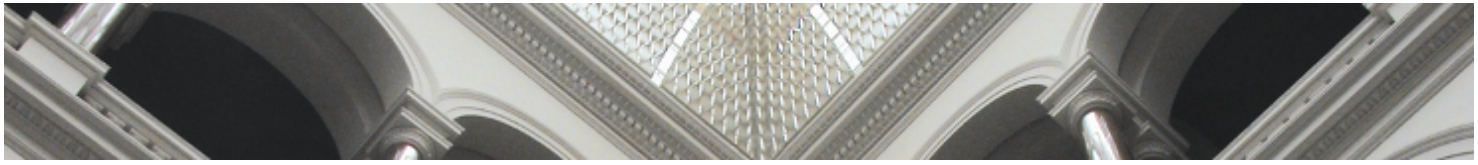
  merge $\mathcal{D}_i$ and $\mathcal{D}_j$

  $n_{clusters} \leftarrow n_{clusters} - 1$

end

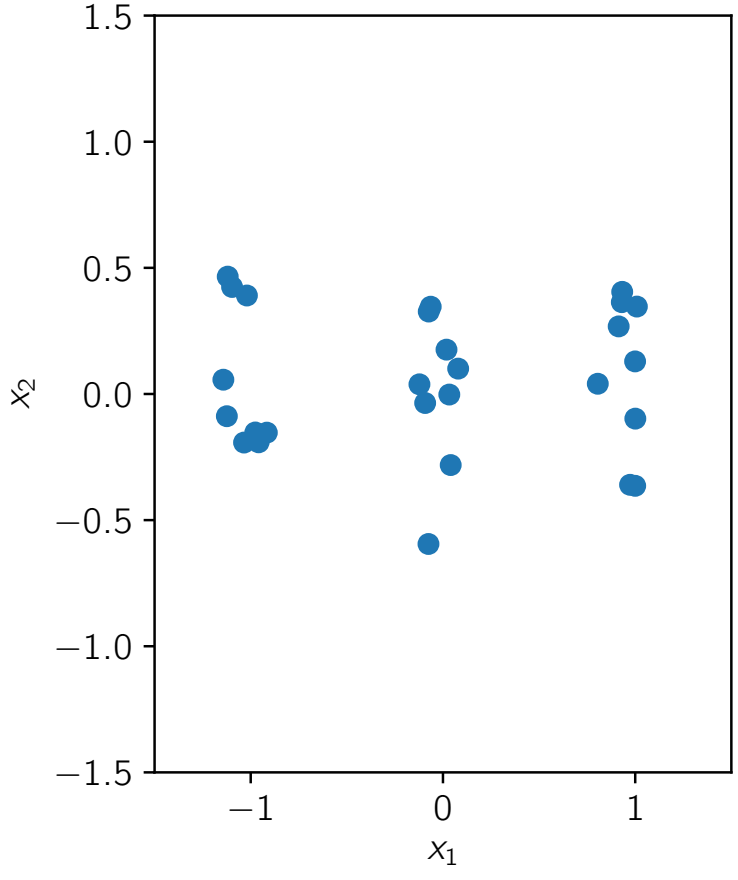
return clusters

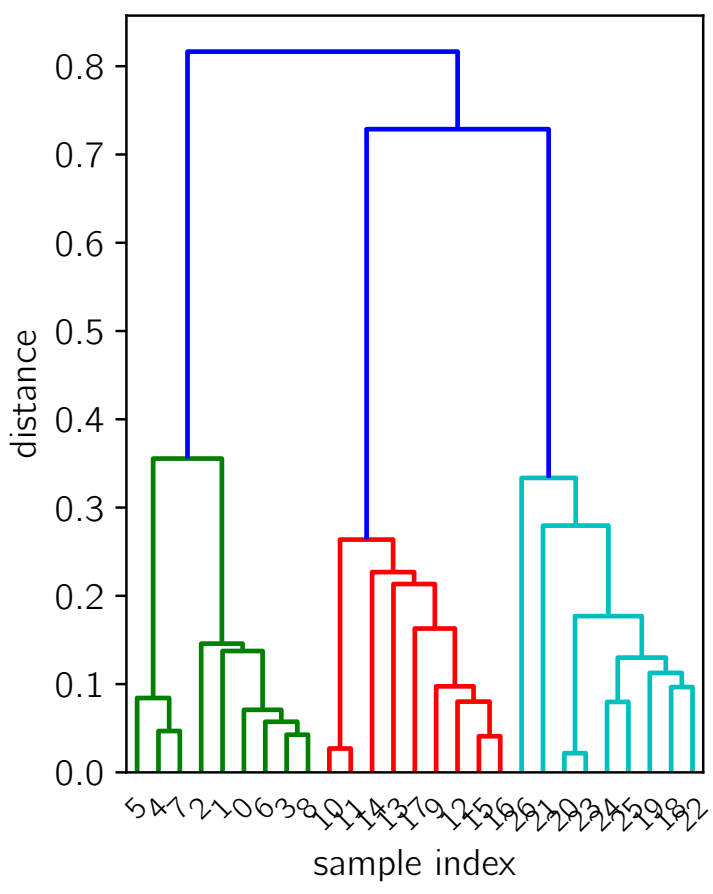


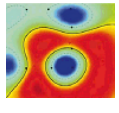


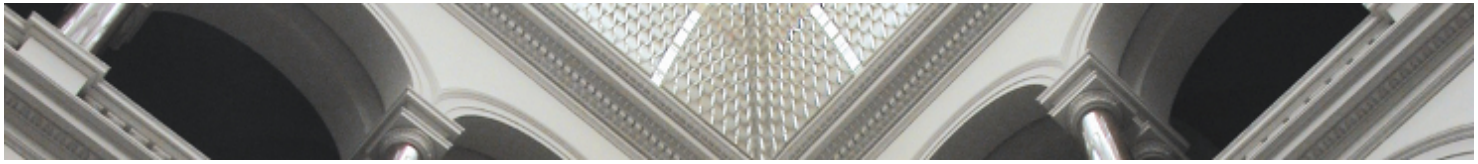
Technische Universität Berlin

Dendrogram with Euclidean distance









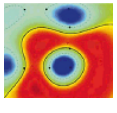
Technische Universität Berlin

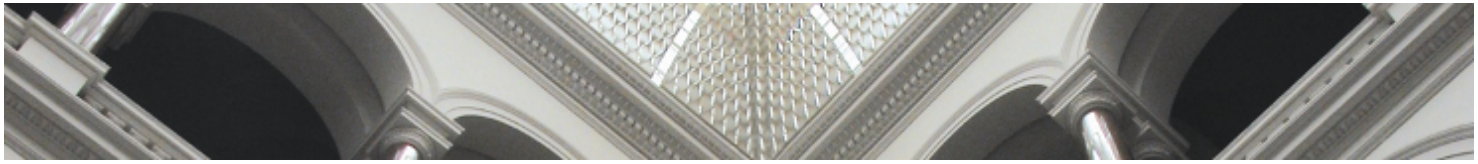
Cluster distance measures

Let $\mathcal{D}_i$, $\mathcal{D}_j$ be the sets containing the samples in clusters *i* and *j*.

Minimum linkage:

$$d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\boldsymbol{x} \in \mathcal{D}_i, \boldsymbol{x}' \in \mathcal{D}_j} ||\boldsymbol{x} - \boldsymbol{x}'||$$





Cluster distance measures

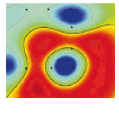
Let $\mathcal{D}_i$, $\mathcal{D}_j$ be the sets containing the samples in clusters i and j.

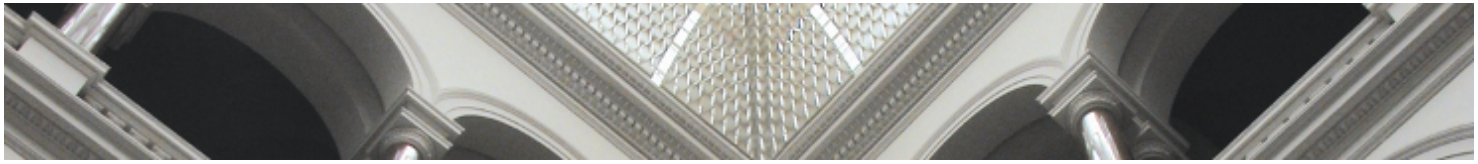
Minimum linkage:

$$d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\boldsymbol{x} \in \mathcal{D}_i, \boldsymbol{x}' \in \mathcal{D}_j} ||\boldsymbol{x} - \boldsymbol{x}'||$$

Maximum or “complete” linkage:

$$d_{max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\boldsymbol{x} \in \mathcal{D}_i, \boldsymbol{x}' \in \mathcal{D}_j} ||\boldsymbol{x} - \boldsymbol{x}'||$$





Technische Universität Berlin

Cluster distance measures

Let $\mathcal{D}_i$, $\mathcal{D}_j$ be the sets containing the samples in clusters *i* and *j*.

Minimum linkage:

$$d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\boldsymbol{x} \in \mathcal{D}_i, \boldsymbol{x}' \in \mathcal{D}_j} ||\boldsymbol{x} - \boldsymbol{x}'||$$

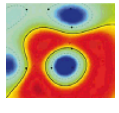
Maximum or “complete” linkage:

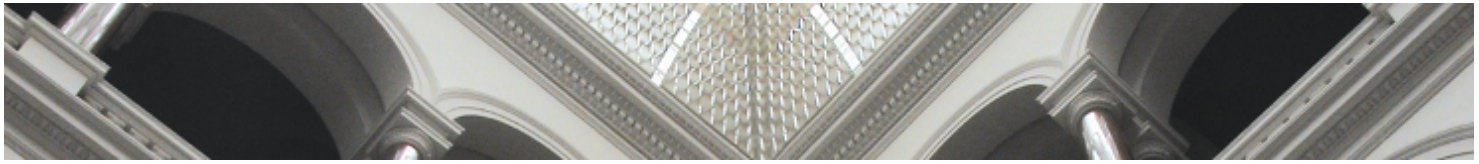
$$d_{max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\boldsymbol{x} \in \mathcal{D}_i, \boldsymbol{x}' \in \mathcal{D}_j} ||\boldsymbol{x} - \boldsymbol{x}'||$$

Average linkage:

$$d_{avg}(\mathcal{D}_i, \mathcal{D}_j) = \frac{1}{n_i n_j} \sum_{\boldsymbol{x} \in \mathcal{D}_i} \sum_{\boldsymbol{x}' \in \mathcal{D}_j} ||\boldsymbol{x} - \boldsymbol{x}'||$$

Notation from Duda, Hart, and Stork 2012

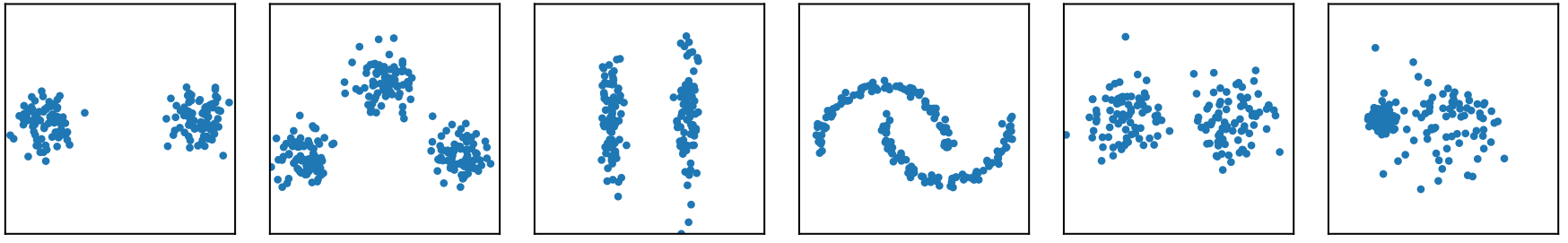


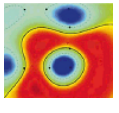


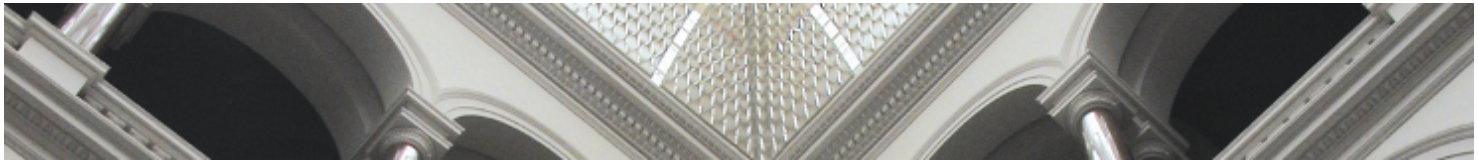
Technische Universität Berlin

Original data

After clustering, assignments can seem “natural”, good to compare original data:



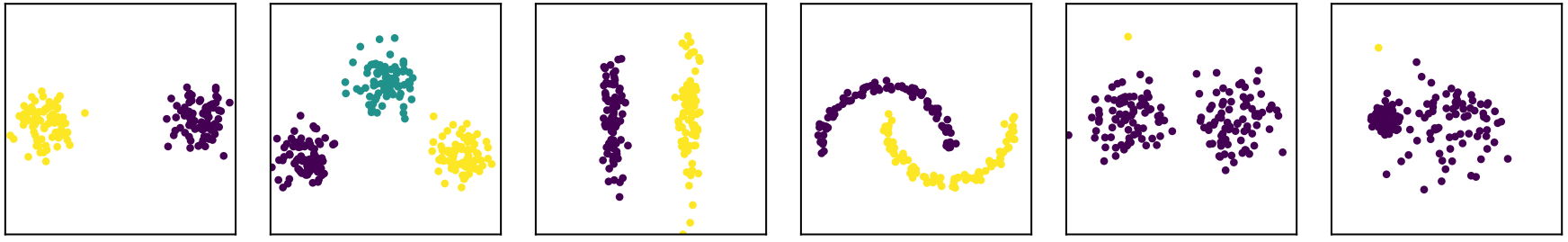


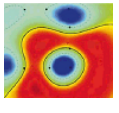


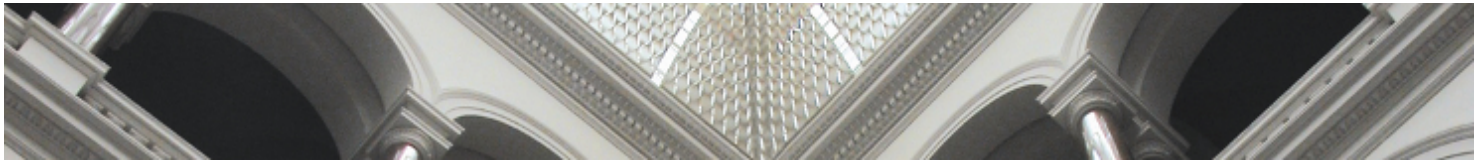
Technische Universität Berlin

Agglomerative clustering results

Minimum

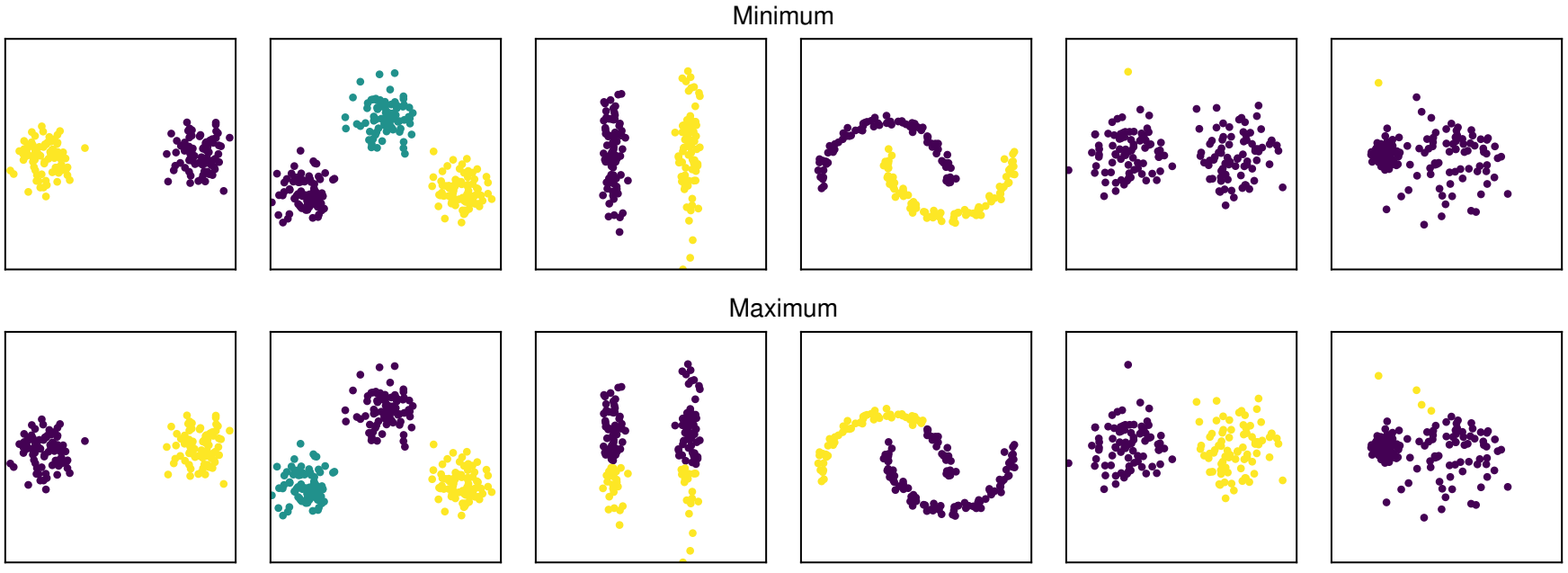


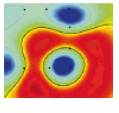


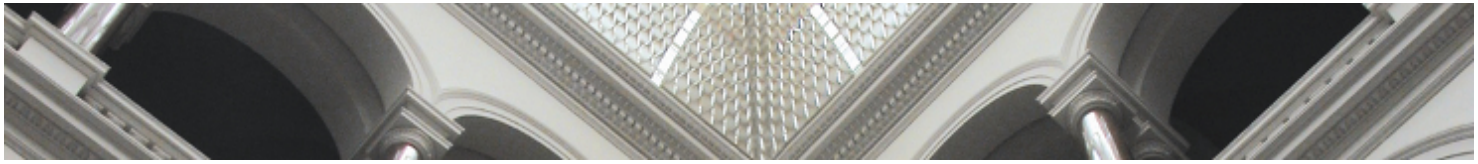


Technische Universität Berlin

Agglomerative clustering results

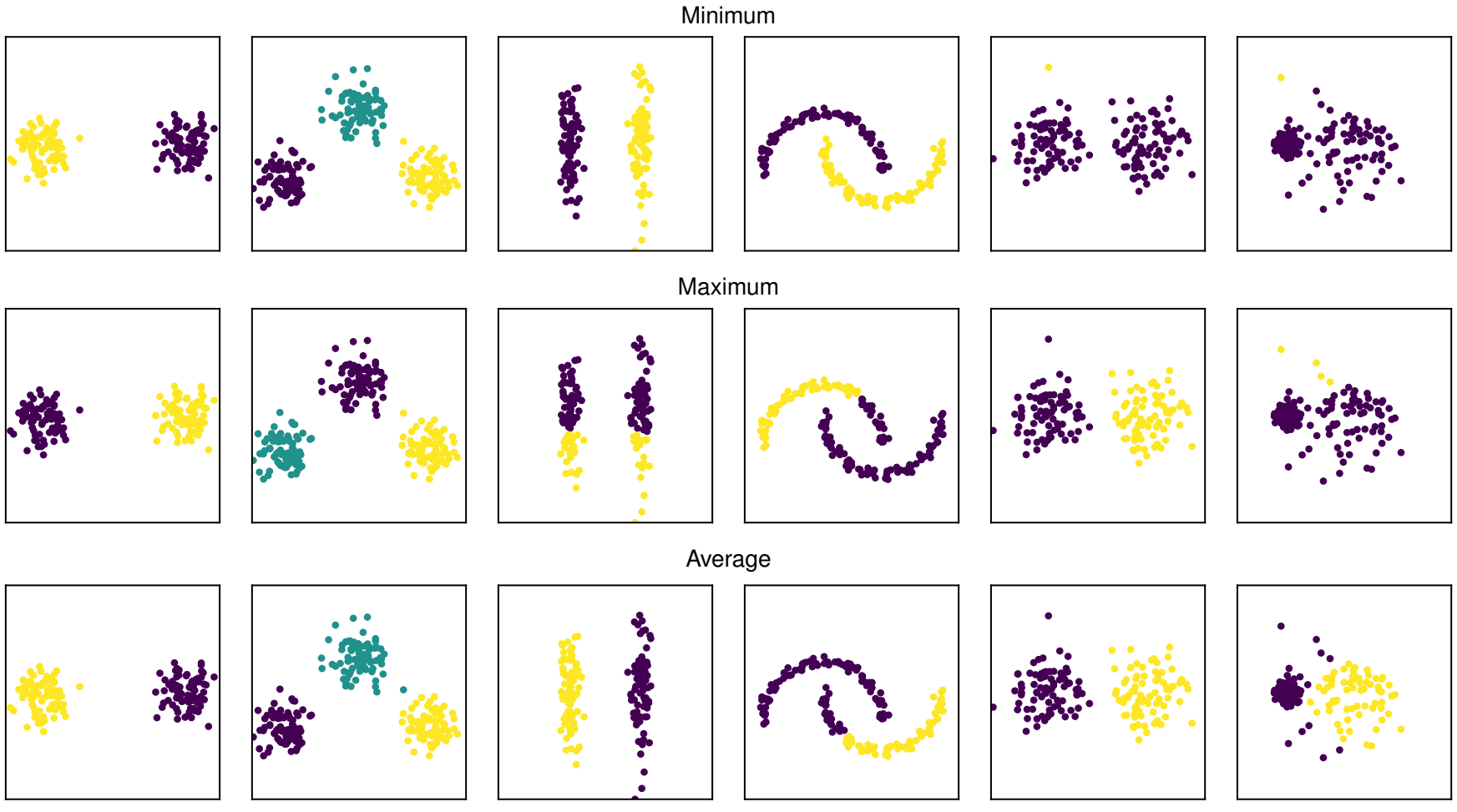


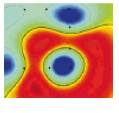


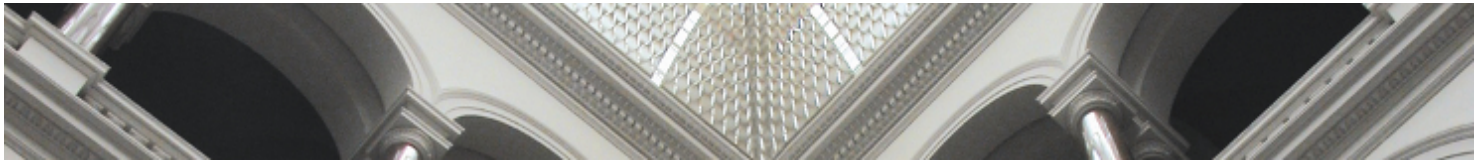


Technische Universität Berlin

Agglomerative clustering results







Technische Universität Berlin

Tips and tricks

class `sklearn.cluster.` **`AgglomerativeClustering`** (*n_clusters=2, affinity='euclidean', memory=None, connectivity=None, compute_full_tree='auto', linkage='ward', pooling_func=<function mean>*) [source]

Agglomerative Clustering

Recursively merges the pair of clusters that minimally increases a given linkage distance.

Read more in the User Guide.

Parameters: **n_clusters** : int, default=2

The number of clusters to find.

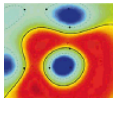
affinity : string or callable, default: "euclidean"

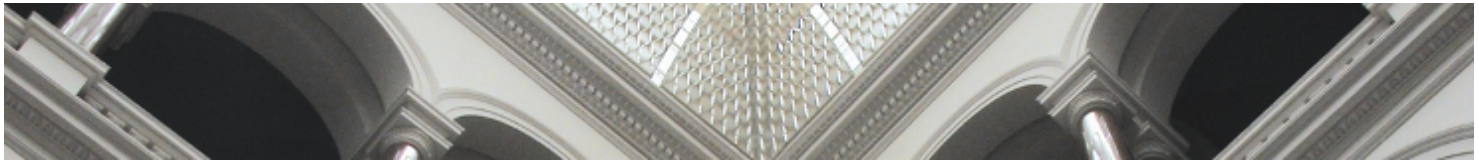
Metric used to compute the linkage. Can be "euclidean", "l1", "l2", "manhattan", "cosine", or 'precomputed'. If linkage is "ward", only "euclidean" is accepted.

Sometimes, documentation can be confusing:

What is the difference between "l2" and "euclidean"?

Or "l1" and "manhattan"?





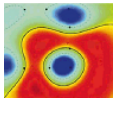
Technische Universität Berlin

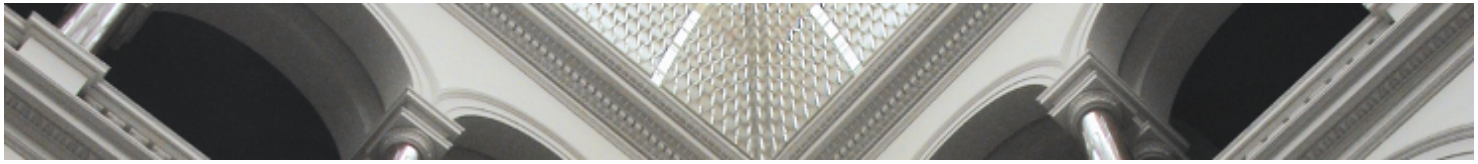
Tips and tricks

Looking into the code:

```
408             elif affinity == 'l2':
409                 # Translate to something understood by scipy
410                 affinity = 'euclidean'
411             elif affinity in ('l1', 'manhattan'):
412                 affinity = 'cityblock'
```

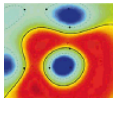
In the background it does exactly the same.

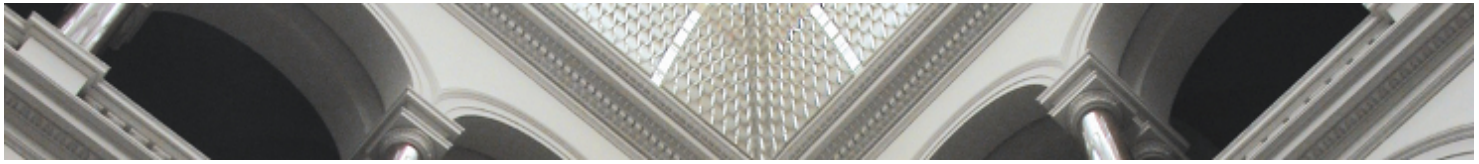




Technische Universität Berlin

Agenda





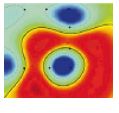
Technische Universität Berlin

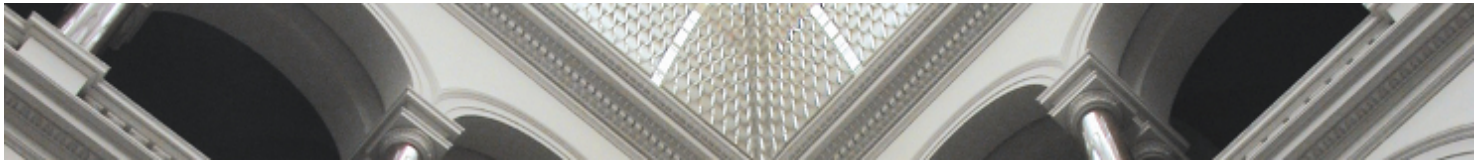
K-Means

Idea: prototype for every cluster

Prototype is mean of all samples in cluster c, $\mathcal{D}_c$:

$$\mu_c = \frac{1}{n_c} \sum_{i \in \mathcal{D}_c} x_i$$





Technische Universität Berlin

K-Means

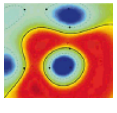
Idea: prototype for every cluster

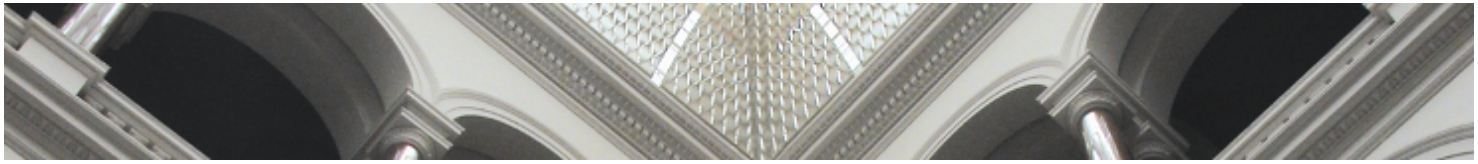
Prototype is mean of all samples in cluster c, $\mathcal{D}_c$:

$$\mu_c = \frac{1}{n_c} \sum_{i \in \mathcal{D}_c} x_i$$

Minimize distance from prototype to each member in the cluster:

$$\mathcal{L} = \sum_c \sum_{i \in \mathcal{D}_c} d(\boldsymbol{x}_i, \mu_c)$$





Technische Universität Berlin

K-Means

Idea: prototype for every cluster

Prototype is mean of all samples in cluster c, $\mathcal{D}_c$:

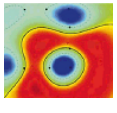
$$\mu_c = \frac{1}{n_c} \sum_{i \in \mathcal{D}_c} x_i$$

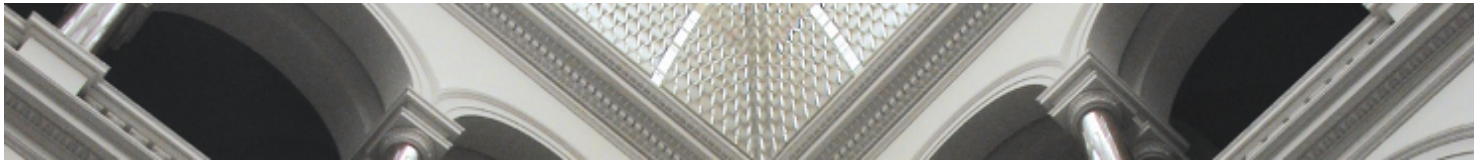
Minimize distance from prototype to each member in the cluster:

$$\mathcal{L} = \sum_c \sum_{i \in \mathcal{D}_c} d(\boldsymbol{x}_i, \mu_c)$$

Finding a solution is NP-hard, but in practice straightforward

K-means cannot work with pairwise data such as user ratings





K-Means algorithm

initialize μ_c to random samples

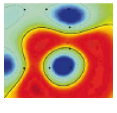
while clusters change do

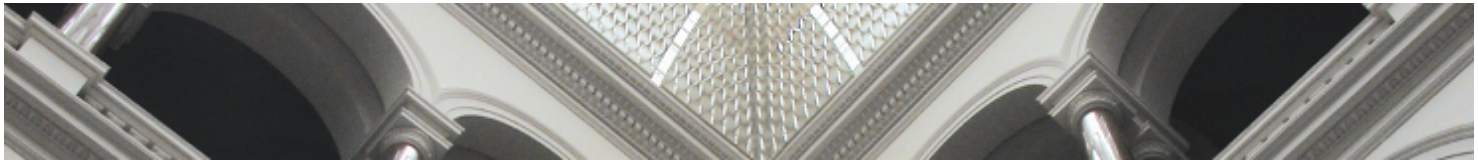
    assign μ_c to cluster means

    assign samples to clusters

end

return clusters

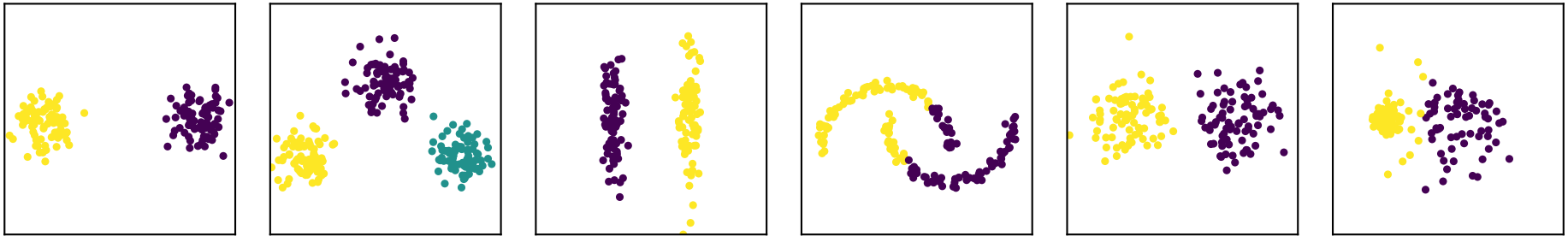


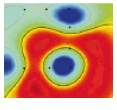


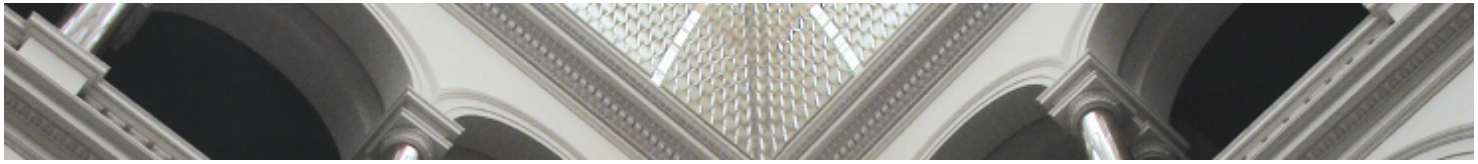
Technische Universität Berlin

K-Means clusters

K-Means assignment:



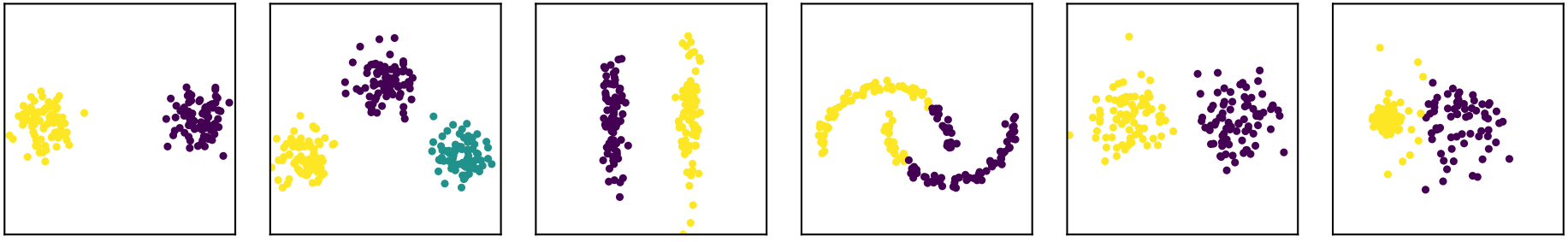




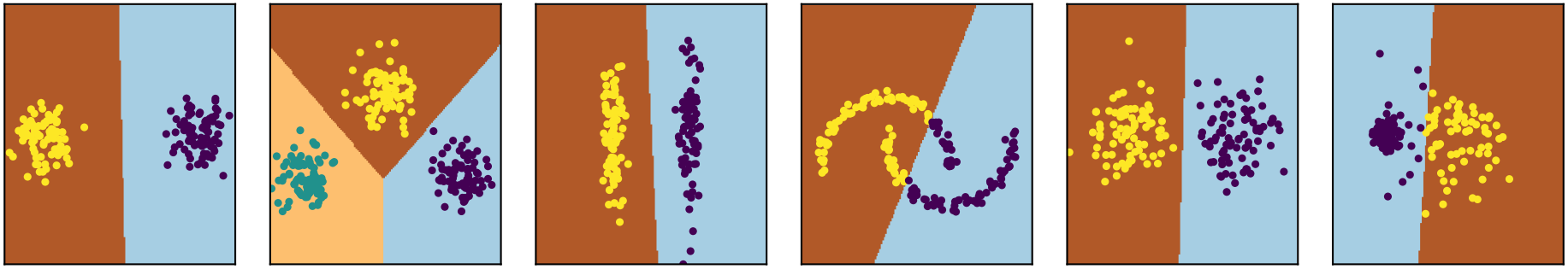
Technische Universität Berlin

K-Means clusters

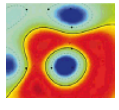
K-Means assignment:

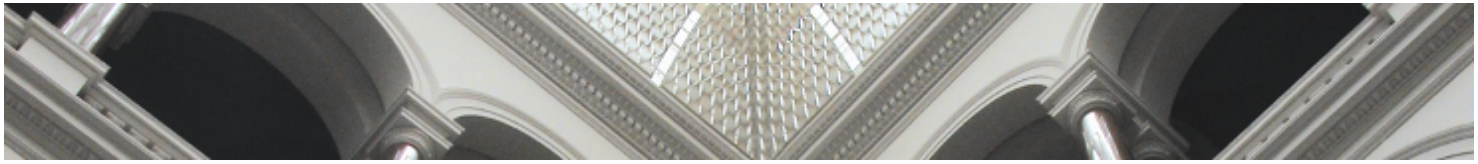


Voronoi tesselation:



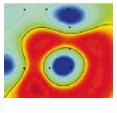
Voronoi diagram not possible with hierarchical clustering!

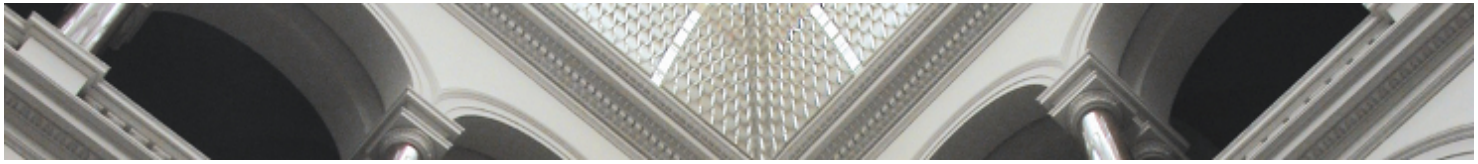




Technische Universität Berlin

Agenda

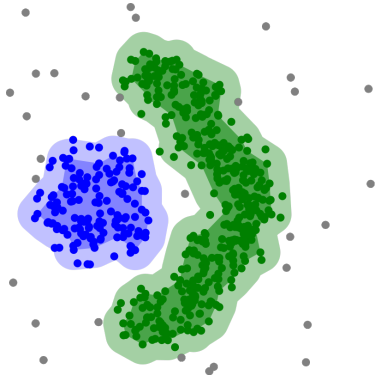




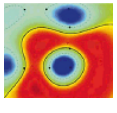
Technische Universität Berlin

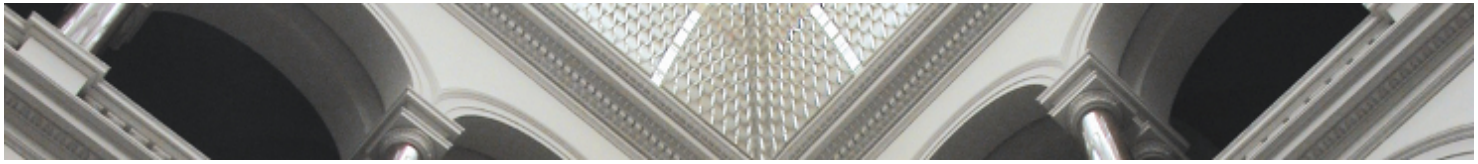
DBSCAN

- clusters are regions of high density separated by regions of low density
- arbitrary cluster shapes



(Wikipedia ”DBSCAN”)

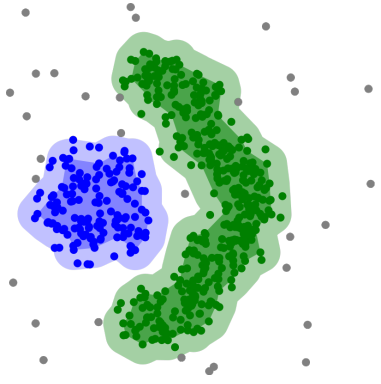




Technische Universität Berlin

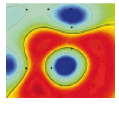
DBSCAN

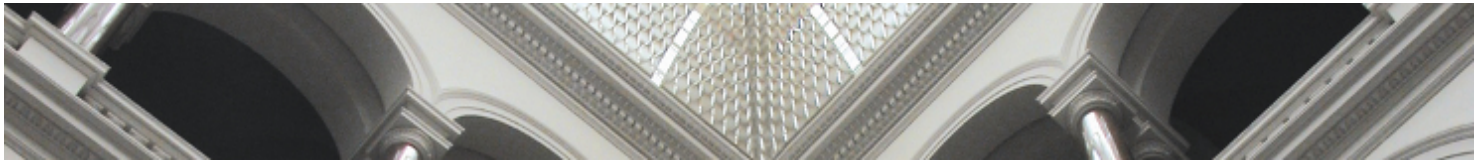
- clusters are regions of high density separated by regions of low density
- arbitrary cluster shapes



(Wikipedia ”DBSCAN”)

- very simple, but effective (2014 SIGKDD test-of-time award (Schubert et al. 2017))
- explicit noise point modeling



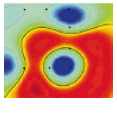


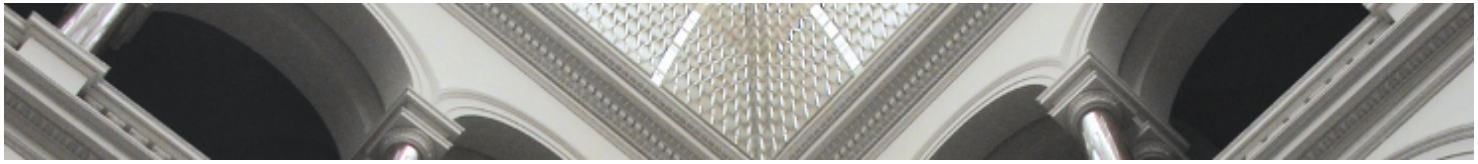
Technische Universität Berlin

DBSCAN

Two parameters:

- ϵ: distance, defines a neighborhood
- MinPts: amount of neighbors to be a core point



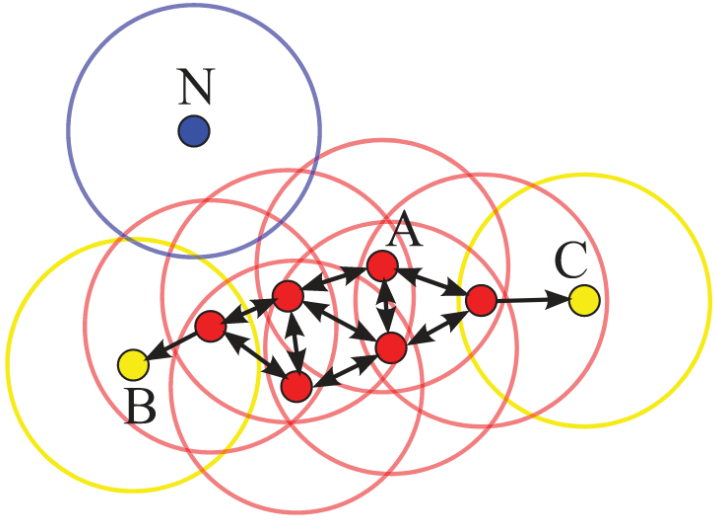


Technische Universität Berlin

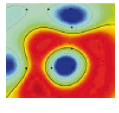
DBSCAN

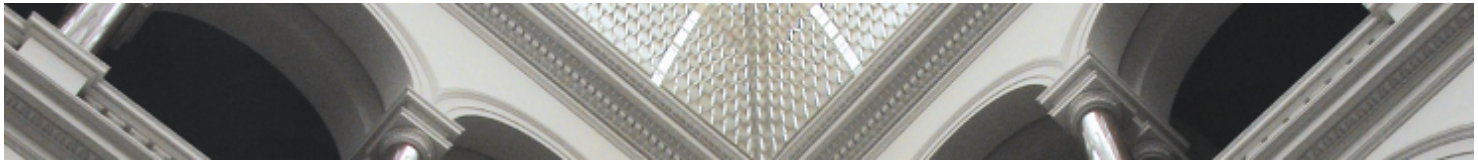
Two parameters:

- ϵ: distance, defines a neighborhood
- MinPts: amount of neighbors to be a core point



(Wikipedia ”DBSCAN”)





Technische Universität Berlin

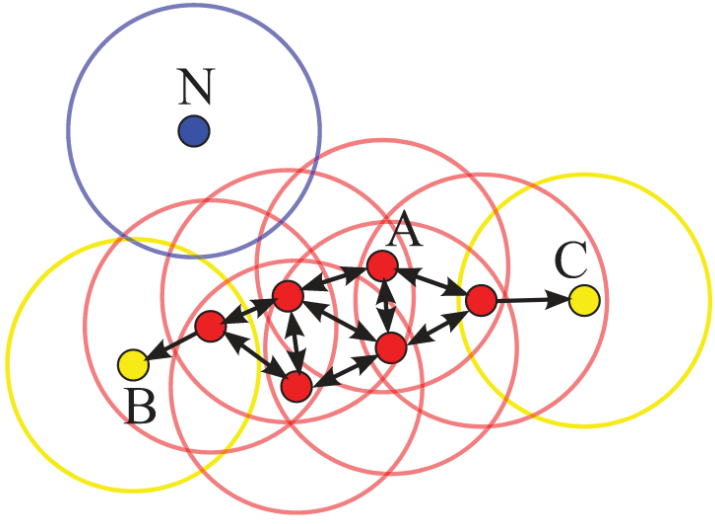
DBSCAN

Two parameters:

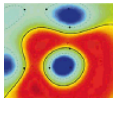
- ϵ: distance, defines a neighborhood
- MinPts: amount of neighbors to be a core point

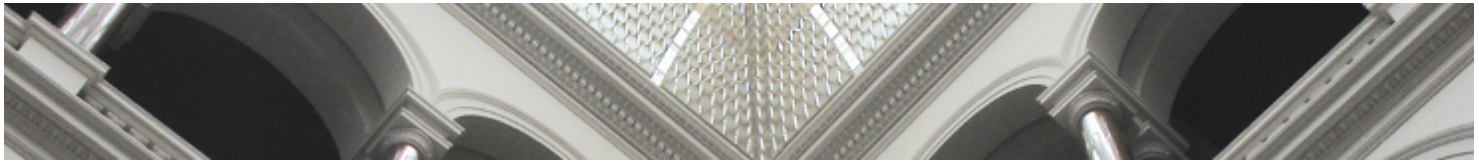
Three types of points:

- core: has MinPts points in its epsilon-neighborhood
- border: within epsilon-neighborhood of core point
- noise: all other points



(Wikipedia ”DBSCAN”)

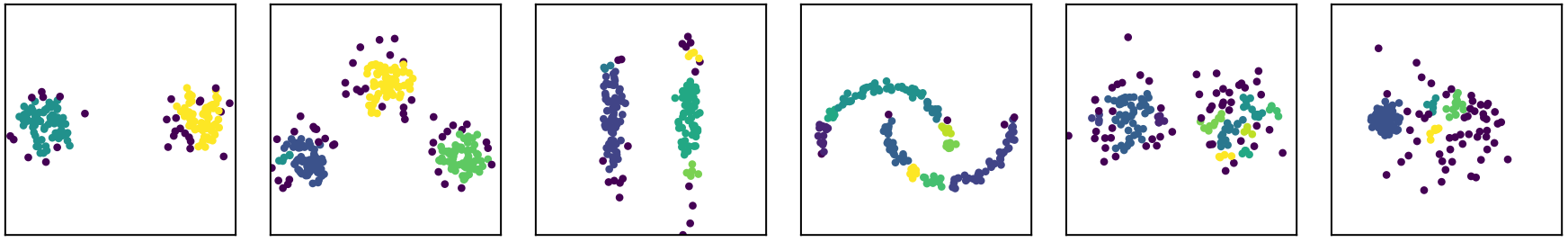


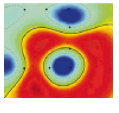


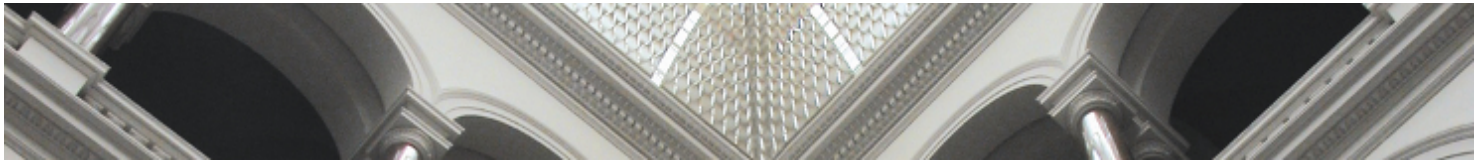
Technische Universität Berlin

DBSCAN clustering results

$\epsilon = 0.1$, MinPts $= 4$



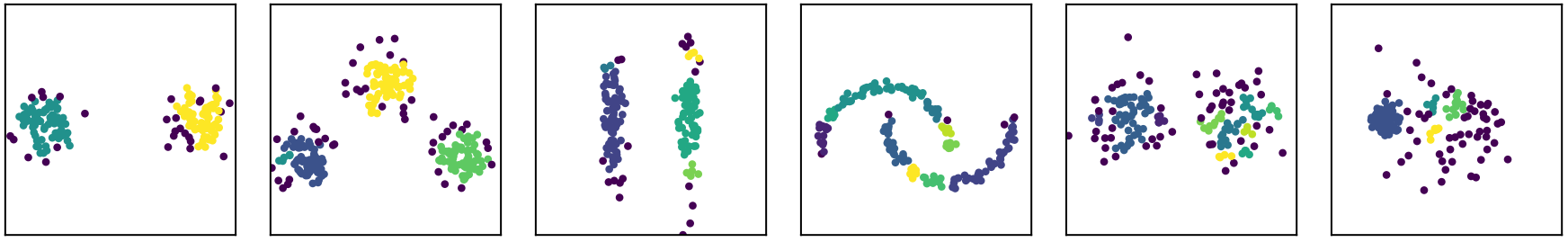




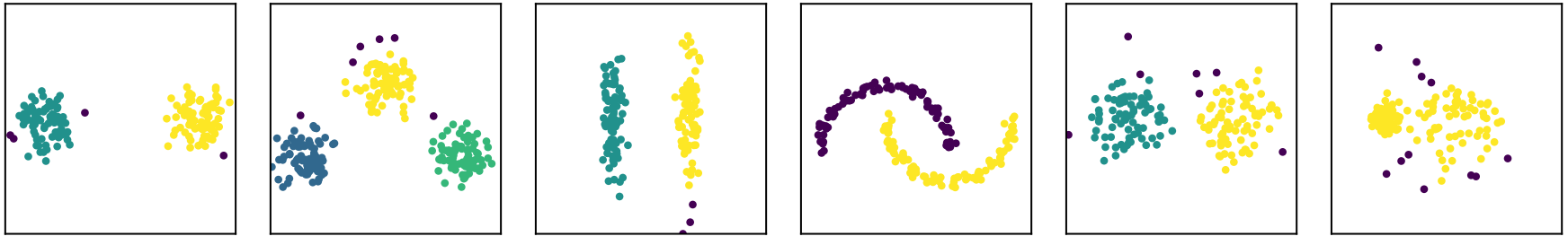
Technische Universität Berlin

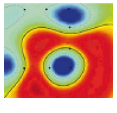
DBSCAN clustering results

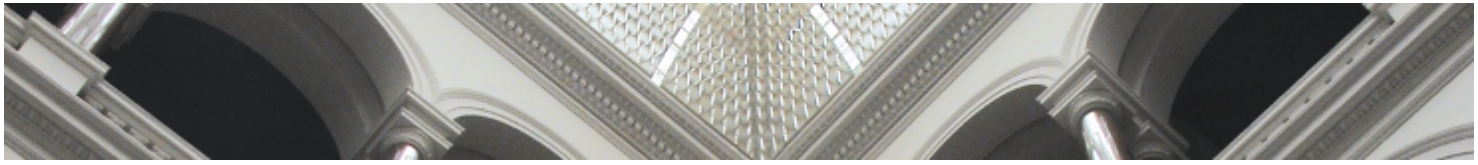
$\epsilon = 0.1$, MinPts $= 4$



$\epsilon = 0.2$, MinPts $= 4$



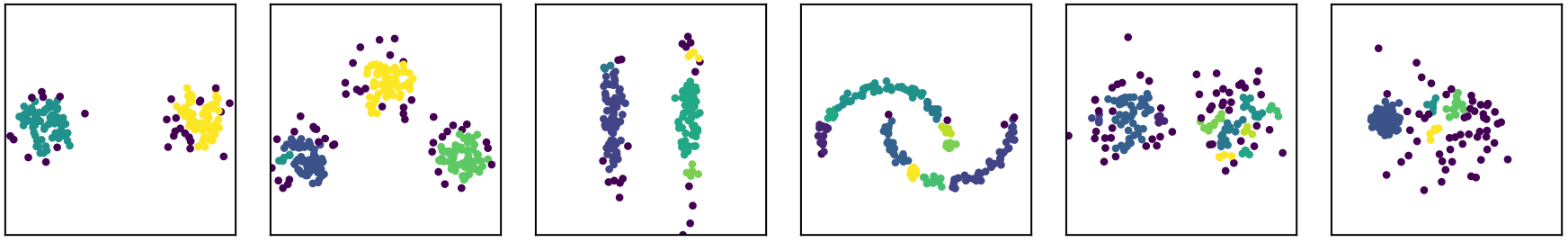




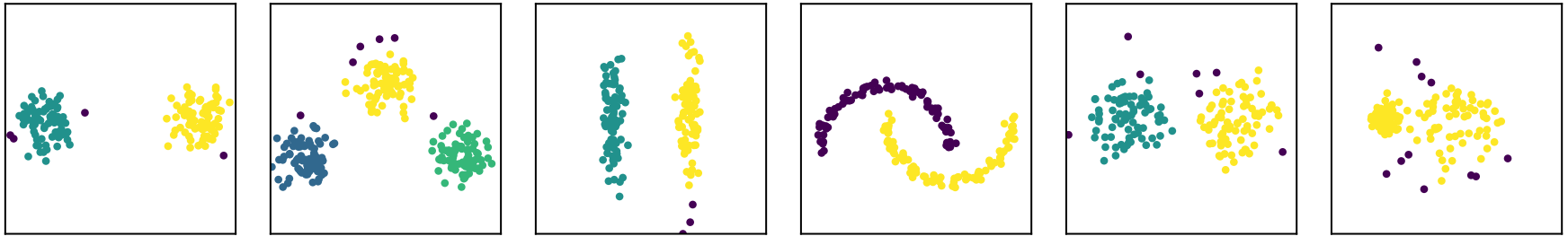
Technische Universität Berlin

DBSCAN clustering results

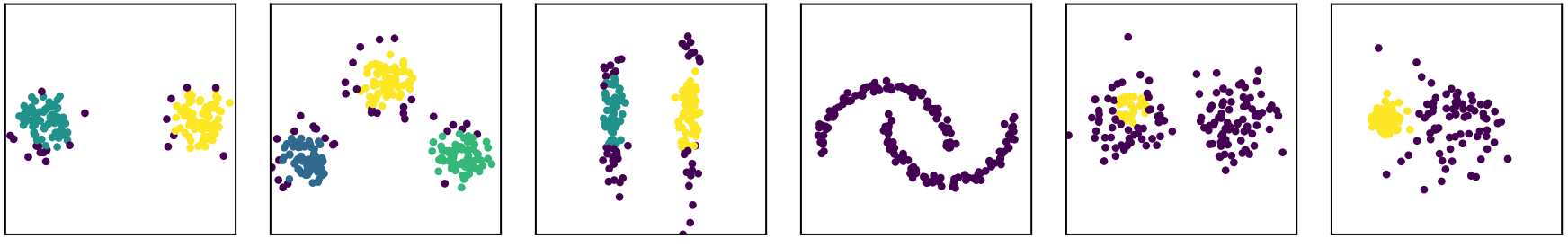
$\epsilon = 0.1$, MinPts $= 4$

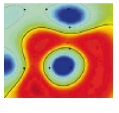


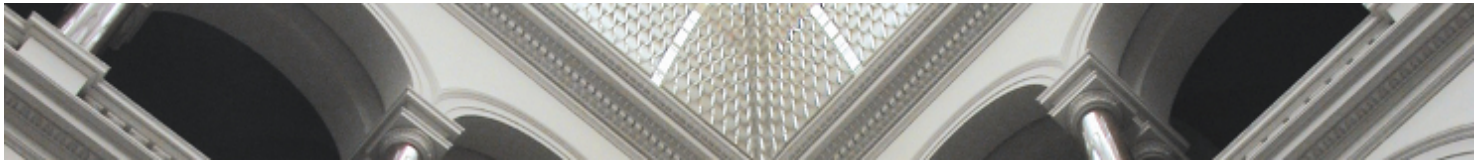
$\epsilon = 0.2$, MinPts $= 4$



$\epsilon = 0.2$, MinPts $= 20$

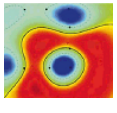


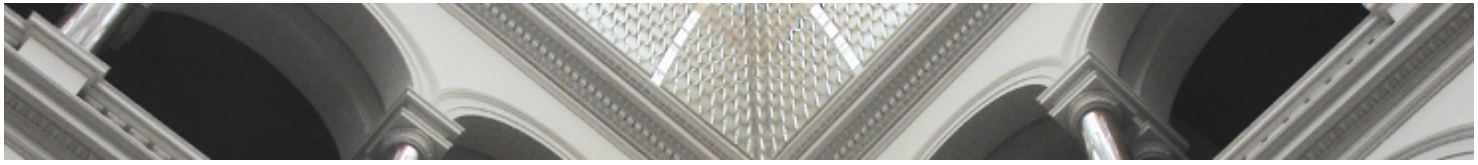




Technische Universität Berlin

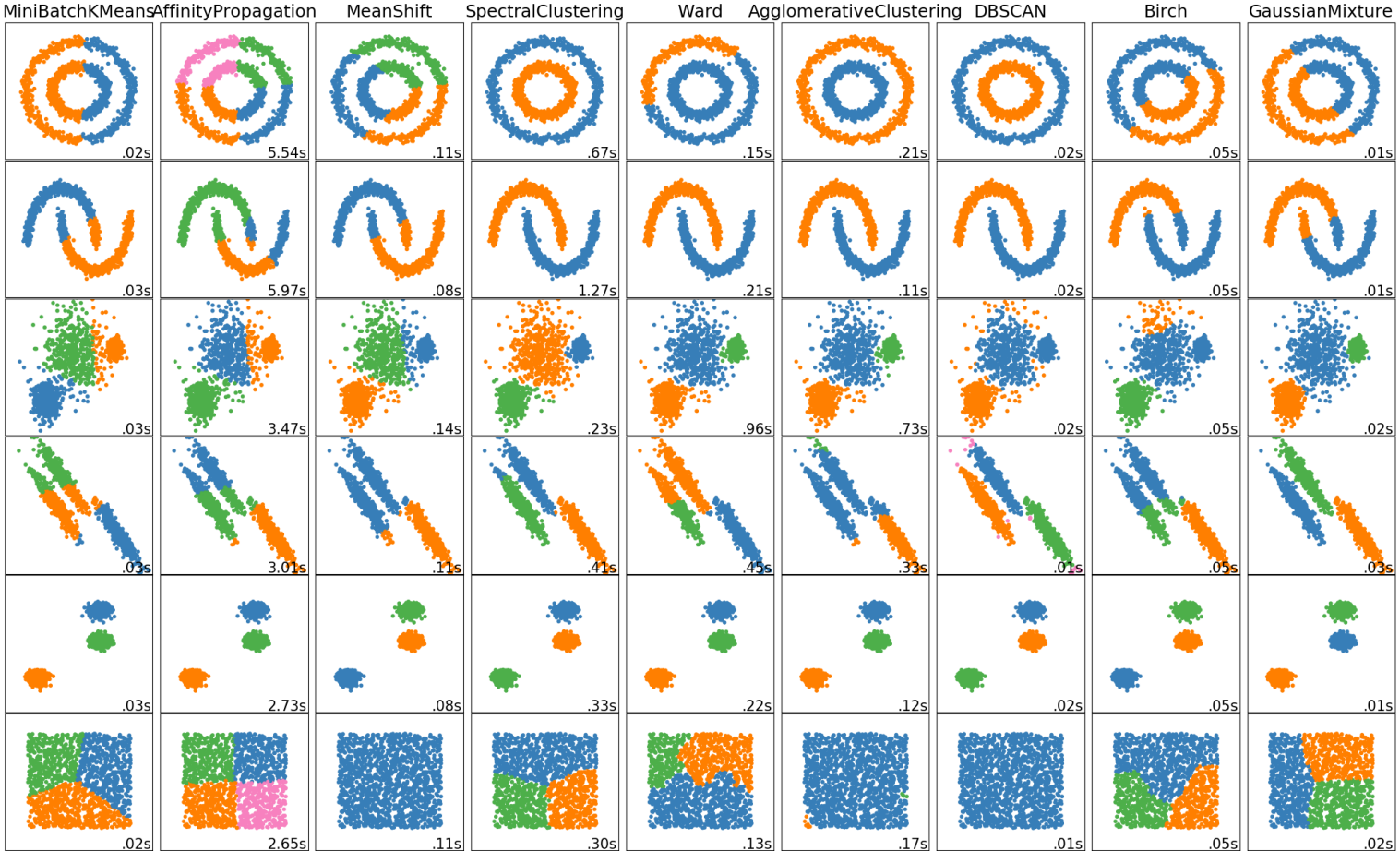
Agenda



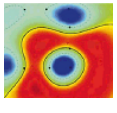


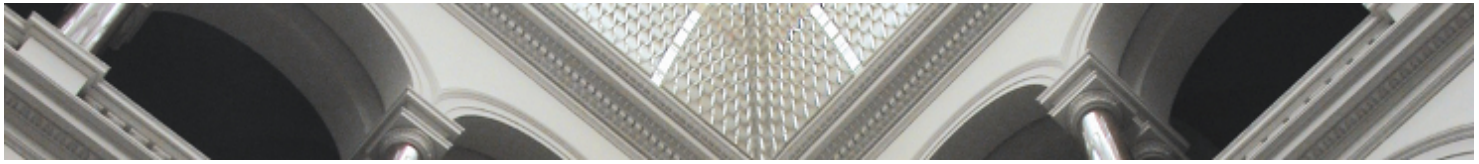
Technische Universität Berlin

Overview of clustering algorithms



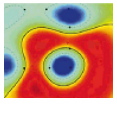
http://scikit-learn.org/stable/modules/clustering.html Pedregosa et al. 2011

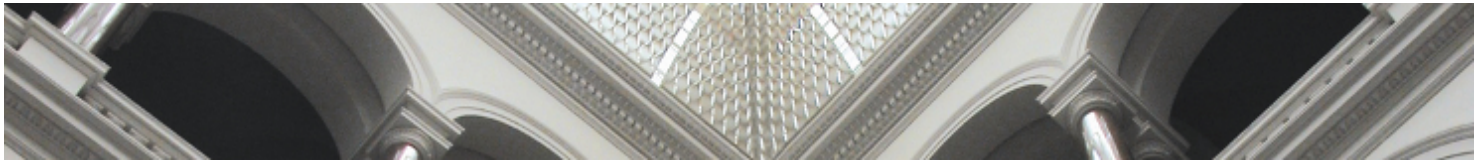




Technische Universität Berlin

Agenda





Technische Universität Berlin

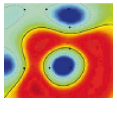
Dimensionalty Reduction - Motivation

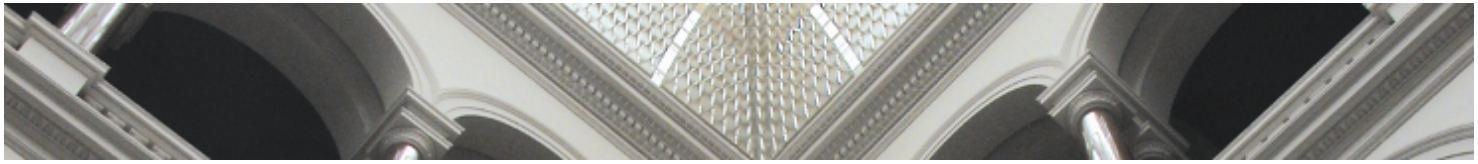
In many applications, we have

- high-dimensional data
- reason to believe they lie close to a lower dimensional subspace

$\rightarrow$ Fewer parameters needed to account for the data properties

hidden causes or *latent variables*





Technische Universität Berlin

Dimensionalty Reduction - Motivation

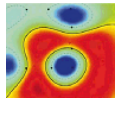
In many applications, we have

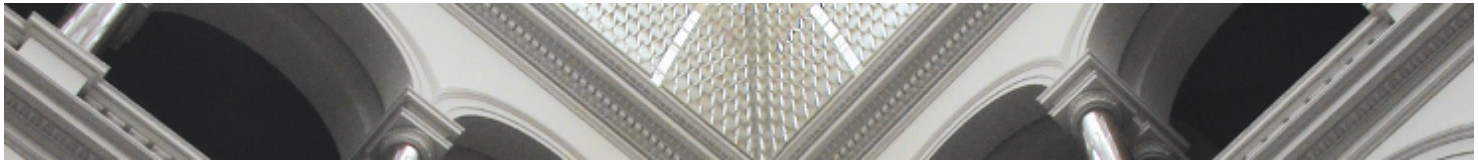
- high-dimensional data
- reason to believe they lie close to a lower dimensional subspace

$\rightarrow$ Fewer parameters needed to account for the data properties
hidden causes or *latent variables*

Examples:

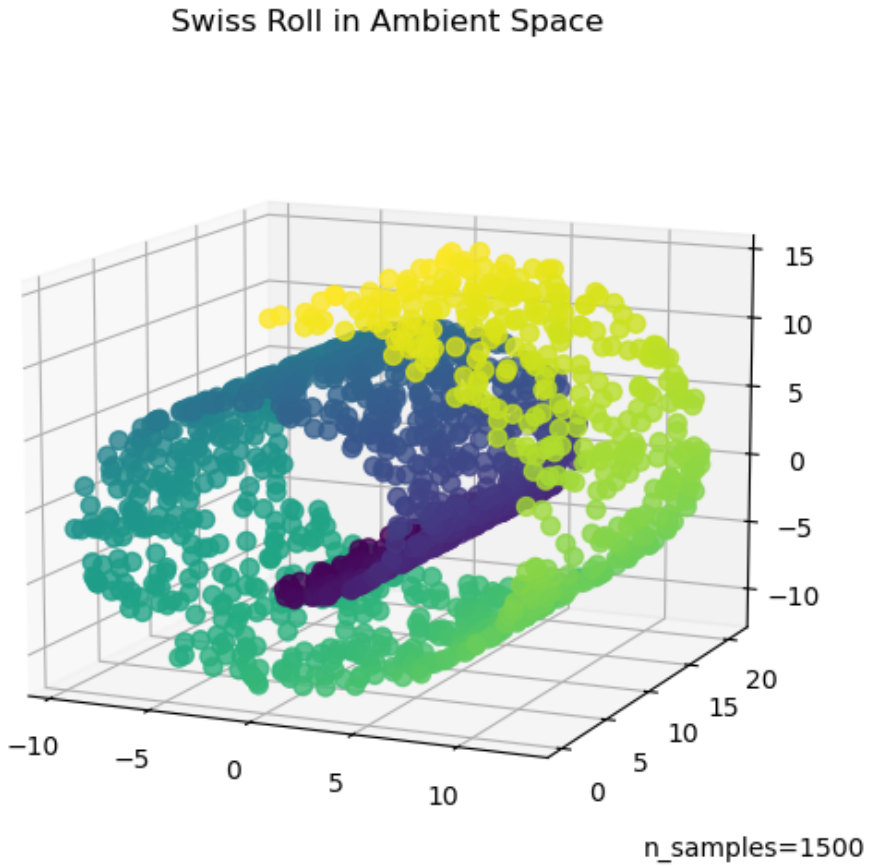
- classify high resolution images
- make a predictive model based on hundreds of customer attributes
- analyse high dimensional neural data
- detect trends in news data



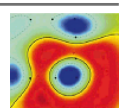


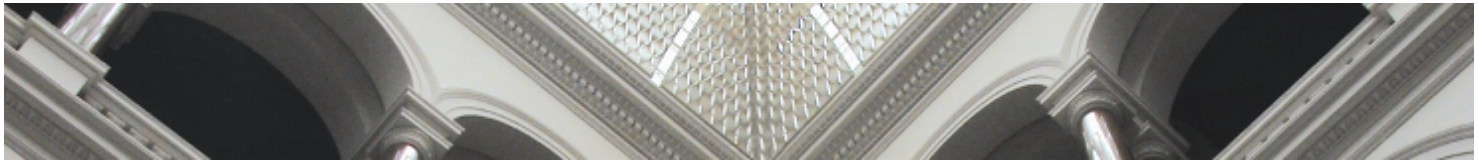
Technische Universität Berlin

Swiss roll data



Scikit-Learn swiss roll dataset

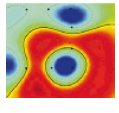


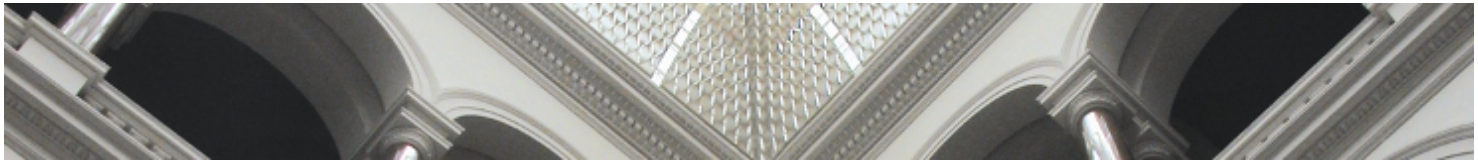


Technische Universität Berlin

Why Dimensionalty Reduction

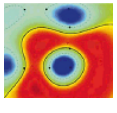
- **Visualization**:
  Insights into high-dimensional structures in the data
- **Better Generalization**:
  Fewer dimensions $\rightarrow$ less chances of overfitting
- **Speeding up** learning algorithms:
  Most algorithms scale badly with increasing data dimensionality
- **Data compression**:
  Less storage requirements

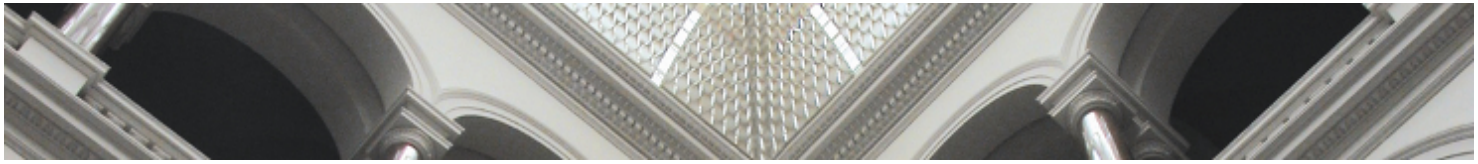




Technische Universität Berlin

Agenda





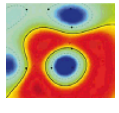
Technische Universität Berlin

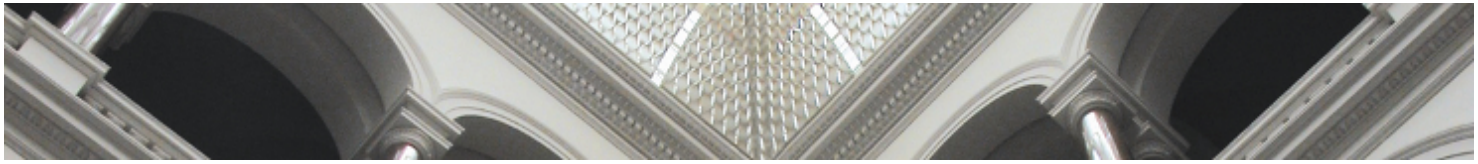
The mathematical model for dimensionalty reduction

We have

- high-dimensional data $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$
- reason to believe they lie close to a lower dimensional subspace

$\rightarrow$ $k < D$ parameters needed to account for the data properties
hidden causes or *latent variables* $H \in \mathbb{R}^{k \times N}$





Technische Universität Berlin

The mathematical model for dimensionalty reduction

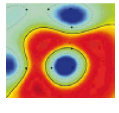
We have

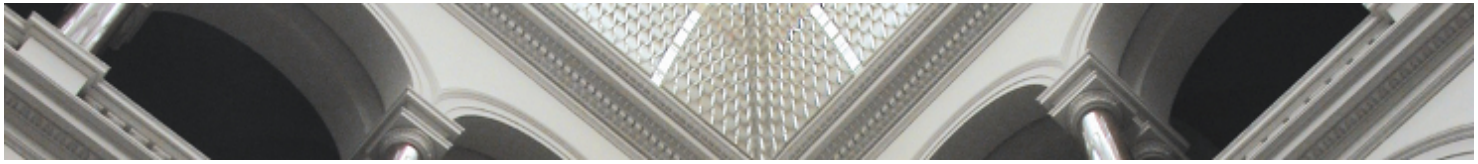
- high-dimensional data $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$
- reason to believe they lie close to a lower dimensional subspace

$\rightarrow$ $k < D$ parameters needed to account for the data properties
hidden causes or *latent variables* $H \in \mathbb{R}^{k \times N}$

Goal: Find $k < D$ hidden causes $H \in \mathbb{R}^{k \times N}$, that explain the observed data via a mixing $W \in \mathbb{R}^{D \times k}$:

$$X \approx WH$$



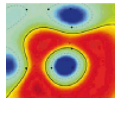


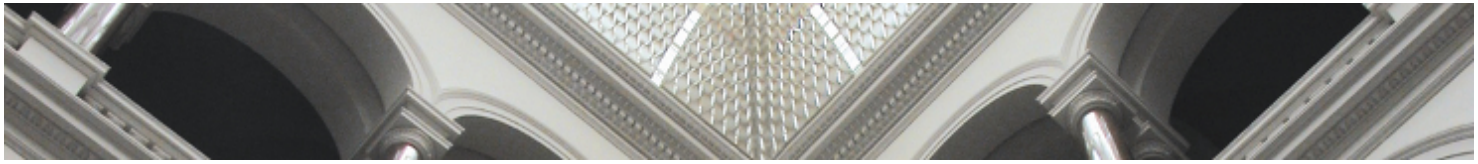
Technische Universität Berlin

Principal Component Analysis

We obtained some data $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$

PCA finds a direction $\mathbf{w} \in \mathbb{R}^D$ such that the variance of the projected data $\mathbf{w}^\top X$ is maximal



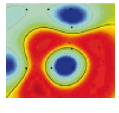


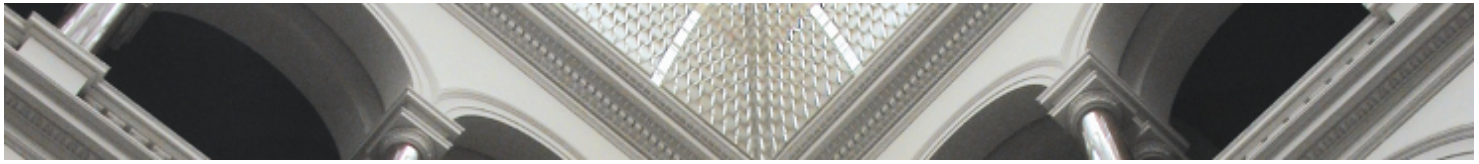
Technische Universität Berlin

Principal Component Analysis

We obtained some data $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$
PCA finds a direction $\mathbf{w} \in \mathbb{R}^D$ such that the variance of the projected data $\mathbf{w}^\top X$ is maximal

$$\mathrm{Var}(\mathbf{w}^T X) \quad = \quad \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^\top \mathbf{x}_n - \mathrm{E}(\mathbf{w}^\top \mathbf{x}))^2$$



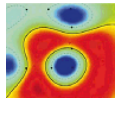


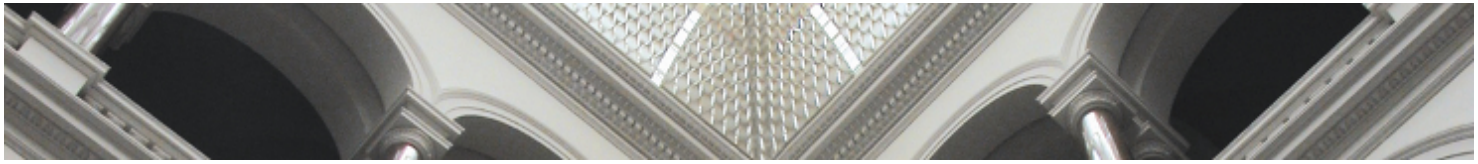
Technische Universität Berlin

Principal Component Analysis

We obtained some data $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$
PCA finds a direction $\mathbf{w} \in \mathbb{R}^D$ such that the variance of the projected data $\mathbf{w}^\top X$ is maximal

$$\begin{aligned} \mathrm{Var}(\mathbf{w}^T X) &= \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^\top \mathbf{x}_n - \mathrm{E}(\mathbf{w}^\top \mathbf{x}))^2 \\ &= \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^\top \mathbf{x}_n - \mathbf{w}^\top \mathrm{E}(\mathbf{x}))^2 = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^\top (\mathbf{x}_n - \mathrm{E}(\mathbf{x})))^2 \end{aligned}$$



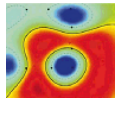


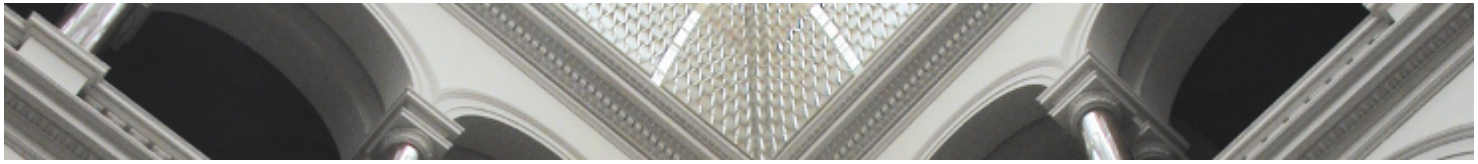
Technische Universität Berlin

Principal Component Analysis

We obtained some data $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$
PCA finds a direction $\mathbf{w} \in \mathbb{R}^D$ such that the variance of the projected data $\mathbf{w}^\top X$ is maximal

$$
\begin{aligned}
\mathrm{Var}(\mathbf{w}^T X) &= \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^\top \mathbf{x}_n - \mathrm{E}(\mathbf{w}^\top \mathbf{x}))^2 \\
&= \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^\top \mathbf{x}_n - \mathbf{w}^\top \mathrm{E}(\mathbf{x}))^2 = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^\top (\mathbf{x}_n - \mathrm{E}(\mathbf{x})))^2 \\
&= \frac{1}{N} \sum_{n=1}^{N} \mathbf{w}^\top (\mathbf{x}_n - \mathrm{E}(\mathbf{x})) \cdot (\mathbf{x}_n - \mathrm{E}(\mathbf{x}))^\top \mathbf{w}
\end{aligned}
$$



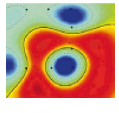


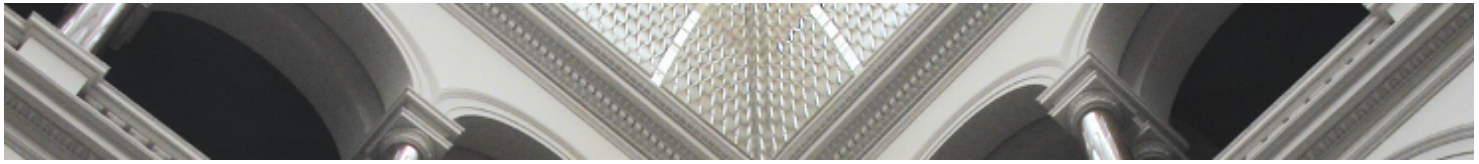
Technische Universität Berlin

Principal Component Analysis

We obtained some data $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$
PCA finds a direction $\mathbf{w} \in \mathbb{R}^D$ such that the variance of the projected data $\mathbf{w}^\top X$ is maximal

$$
\begin{aligned}
\mathrm{Var}(\mathbf{w}^T X) &= \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^\top \mathbf{x}_n - \mathrm{E}(\mathbf{w}^\top \mathbf{x}))^2 \\
&= \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^\top \mathbf{x}_n - \mathbf{w}^\top \mathrm{E}(\mathbf{x}))^2 = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^\top (\mathbf{x}_n - \mathrm{E}(\mathbf{x})))^2 \\
&= \frac{1}{N} \sum_{n=1}^{N} \mathbf{w}^\top (\mathbf{x}_n - \mathrm{E}(\mathbf{x})) \cdot (\mathbf{x}_n - \mathrm{E}(\mathbf{x}))^\top \mathbf{w} \\
&= \mathbf{w}^\top \underbrace{\left(\frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \mathrm{E}(\mathbf{x})) \cdot (\mathbf{x}_n - \mathrm{E}(\mathbf{x}))^\top \right)}_{\text{Covariance matrix S}} \mathbf{w}
\end{aligned}
$$





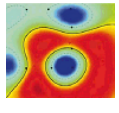
Technische Universität Berlin

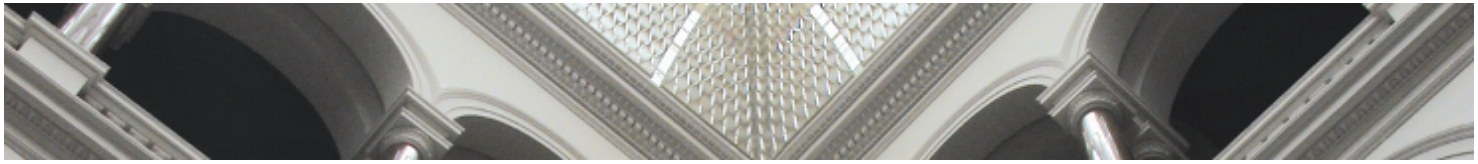
Principal Component Analysis

We obtained some data $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$

PCA finds a direction $\mathbf{w} \in \mathbb{R}^D$ such that the variance of the projected data $\mathbf{w}^\top X$ is maximal

$$\mathrm{Var}(\mathbf{w}^T X) = \mathbf{w}^\top \underbrace{\left(\frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \mathrm{E}(\mathbf{x})) \cdot (\mathbf{x}_n - \mathrm{E}(\mathbf{x}))^\top \right)}_{\text{Covariance matrix S}} \mathbf{w}$$





Technische Universität Berlin

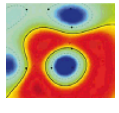
Principal Component Analysis

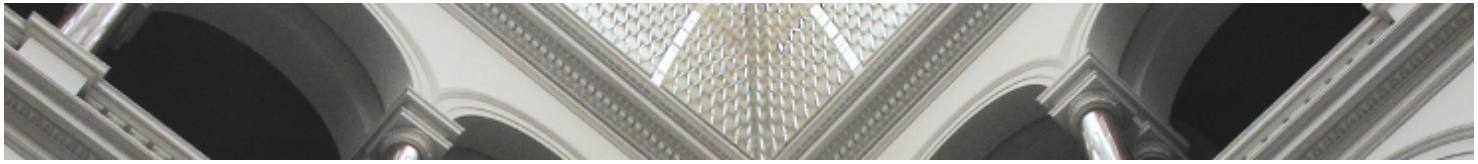
We obtained some data $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$

PCA finds a direction $\mathbf{w} \in \mathbb{R}^D$ such that the variance of the projected data $\mathbf{w}^\top X$ is maximal

$$\begin{aligned} \mathrm{Var}(\mathbf{w}^T X) &= \mathbf{w}^\top \underbrace{\left(\frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \mathrm{E}(\mathbf{x})) \cdot (\mathbf{x}_n - \mathrm{E}(\mathbf{x}))^\top \right)}_{\text{Covariance matrix S}} \mathbf{w} \\ &= \mathbf{w}^\top X X^\top \mathbf{w} \end{aligned}$$

where we assume centered data





Technische Universität Berlin

Principal Component Analysis

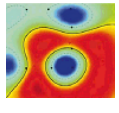
We obtained some data $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$

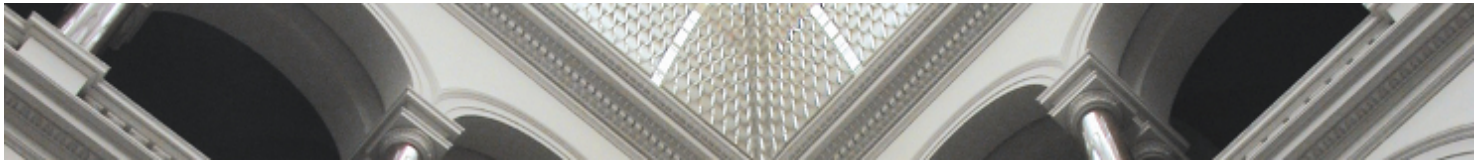
PCA finds a direction $\mathbf{w} \in \mathbb{R}^D$ such that the variance of the projected data $\mathbf{w}^\top X$ is maximal

$$\begin{aligned} \mathrm{Var}(\mathbf{w}^T X) &= \mathbf{w}^\top \underbrace{\left(\frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \mathrm{E}(\mathbf{x})) \cdot (\mathbf{x}_n - \mathrm{E}(\mathbf{x}))^\top \right)}_{\text{Covariance matrix S}} \mathbf{w} \\ &= \mathbf{w}^\top X X^\top \mathbf{w} \end{aligned}$$

where we assume centered data

And: we need to constrain $\mathbf{w}$





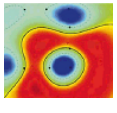
Technische Universität Berlin

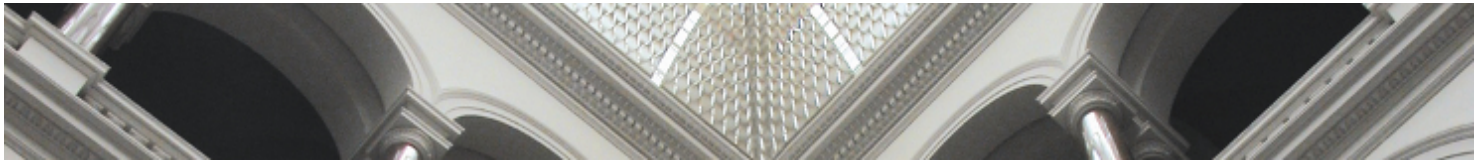
Principal Component Analysis

PCA finds a direction $\mathbf{w} \in \mathbb{R}^D$ such that the variance of the projected data $\mathbf{w}^T X$ is maximal

$$\arg\max_{w} \frac{\mathbf{w}^\top S \mathbf{w}}{\mathbf{w}^\top \mathbf{w}}$$

This objective function is independent of the scaling of $\mathbf{w}$.





Technische Universität Berlin

Principal Component Analysis

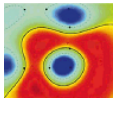
PCA finds a direction $\mathbf{w} \in \mathbb{R}^D$ such that the variance of the projected data $\mathbf{w}^T X$ is maximal

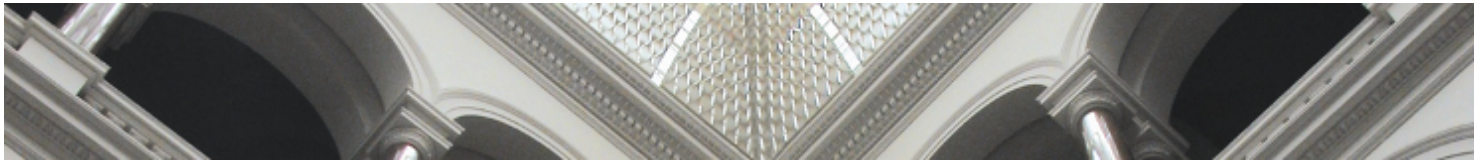
$$\arg\max_{w} \frac{\mathbf{w}^\top S \mathbf{w}}{\mathbf{w}^\top \mathbf{w}}$$

This objective function is independent of the scaling of $\mathbf{w}$.

Note the similarity to the objective of Linear Discriminant Analysis!

$\rightarrow$ Different covariance matrices, different problem, but: same maths solve it





Technische Universität Berlin

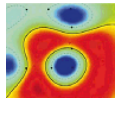
Principal Component Analysis

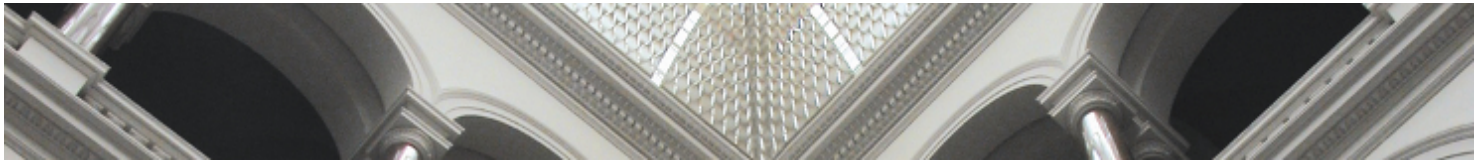
$$\arg\max_{\mathbf{w}} \frac{\mathbf{w}^\top S\mathbf{w}}{\mathbf{w}^\top \mathbf{w}}$$

Set the derivative w.r.t $\mathbf{w}$ to zero:

$$\begin{aligned}
\frac{(\mathbf{w}^\top \mathbf{w})2S\mathbf{w} - (\mathbf{w}^\top S\mathbf{w})2\mathbf{w}}{(\mathbf{w}^\top \mathbf{w})^2} &= 0 \\
(\mathbf{w}^\top S\mathbf{w})\mathbf{w} &= (\mathbf{w}^\top \mathbf{w})S\mathbf{w} \\
\mathbf{w} &= S\mathbf{w} \cdot \underbrace{\frac{\mathbf{w}^\top \mathbf{w}}{\mathbf{w}^\top S\mathbf{w}}}_{\textit{scalar}} \\
S\mathbf{w} &= \lambda \mathbf{w}
\end{aligned}$$

This is the standard eigenvalue problem.





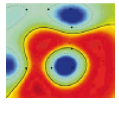
Technische Universität Berlin

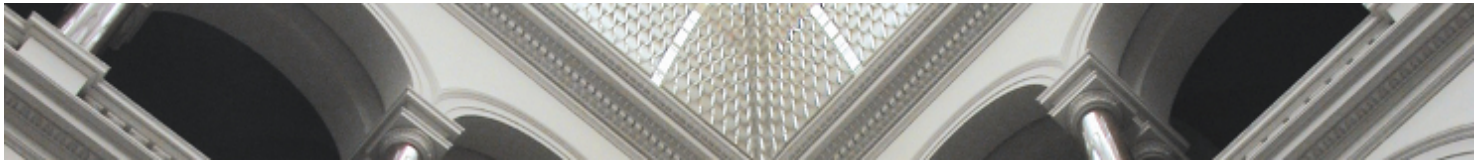
Principal Component Analysis

For $S\mathbf{w} = \lambda\mathbf{w}$, we see that the variance in direction $\mathbf{w}$ is given by:

$$\arg\max_{\mathbf{w}} \frac{\mathbf{w}^\top S\mathbf{w}}{\mathbf{w}^\top\mathbf{w}} = \frac{\mathbf{w}^\top \lambda\mathbf{w}}{\mathbf{w}^\top\mathbf{w}} = \lambda$$

The variance of the projected data in an eigendirection $\mathbf{w}$ is given by the corresponding eigenvalue!





Technische Universität Berlin

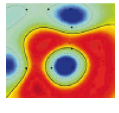
Principal Component Analysis

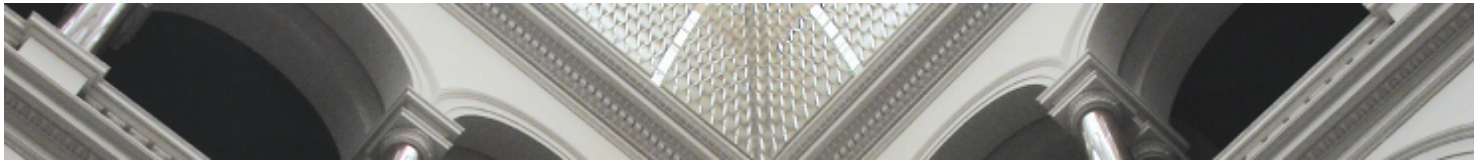
For $S\mathbf{w} = \lambda\mathbf{w}$, we see that the variance in direction $\mathbf{w}$ is given by:

$$\arg\max_{\mathbf{w}} \frac{\mathbf{w}^\top S\mathbf{w}}{\mathbf{w}^\top\mathbf{w}} = \frac{\mathbf{w}^\top \lambda\mathbf{w}}{\mathbf{w}^\top\mathbf{w}} = \lambda$$

The variance of the projected data in an eigendirection $\mathbf{w}$ is given by the corresponding eigenvalue!

The direction of maximal variance in the data is equal to the eigenvector having the largest eigenvalue.





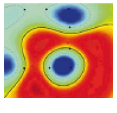
Technische Universität Berlin

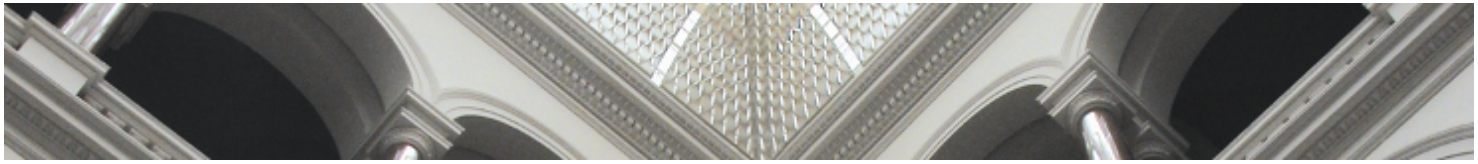
Principal Component Analysis

Require: data $x_1, \ldots, x_N \in \mathbb{R}^d$, number of principal components k

1: # Center Data
2: $X = X - 1/N \sum_i x_i$
3: # Compute Covariance Matrix
4: $C = 1/N \, XX^\top$
5: # Compute eigenvectors corresponding to the k largest eigenvalues
6: $W = \text{eig}(C)$
7: # Project data onto W
8: $H = W^\top X$
9: **return** W, H

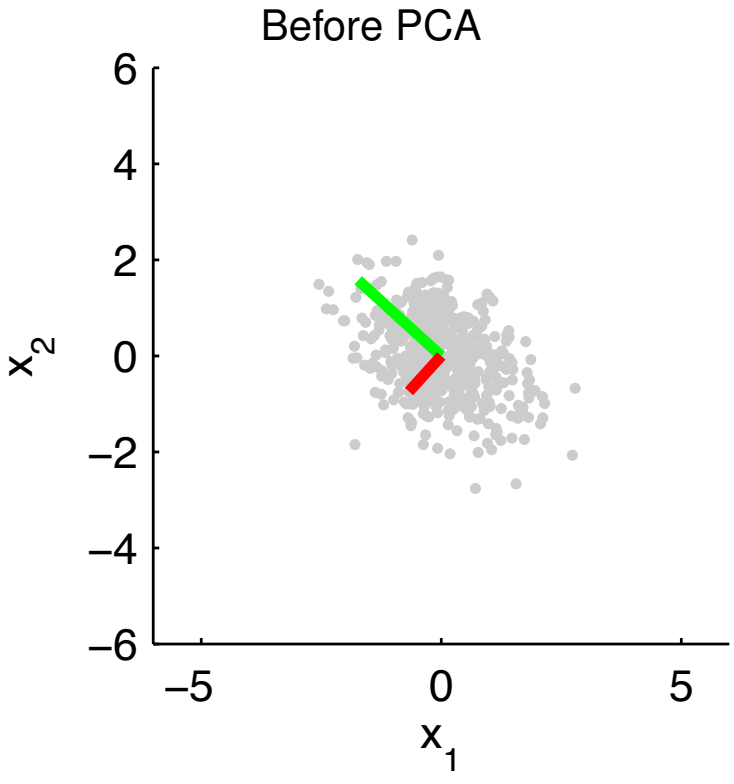
Algorithm 1: Principal Component Analysis

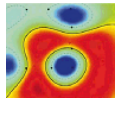


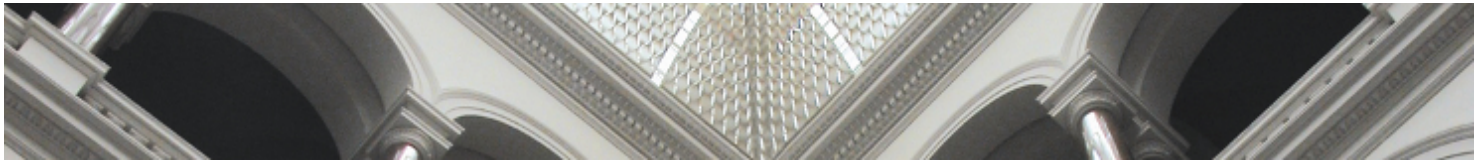


Technische Universität Berlin

Principal Component Analysis

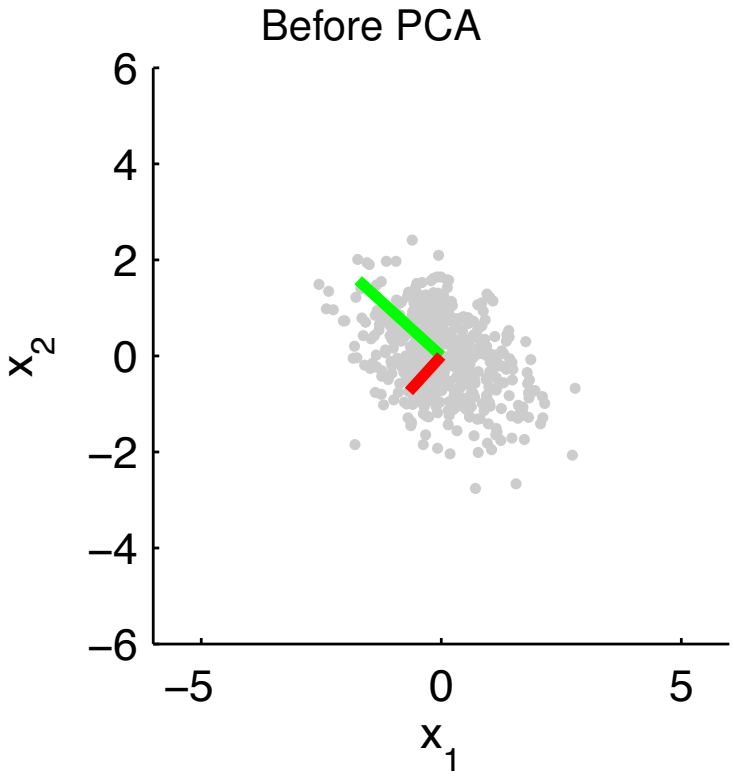


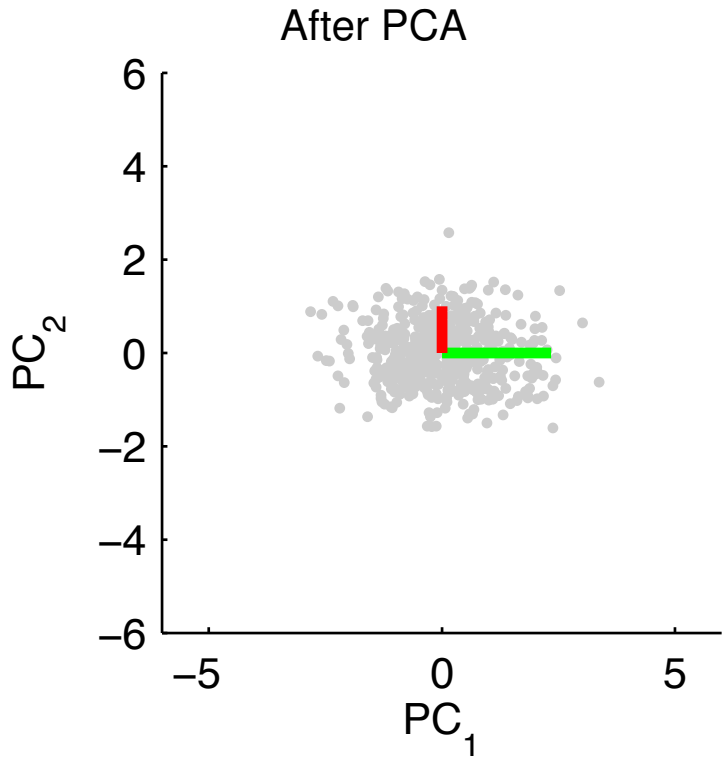


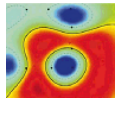


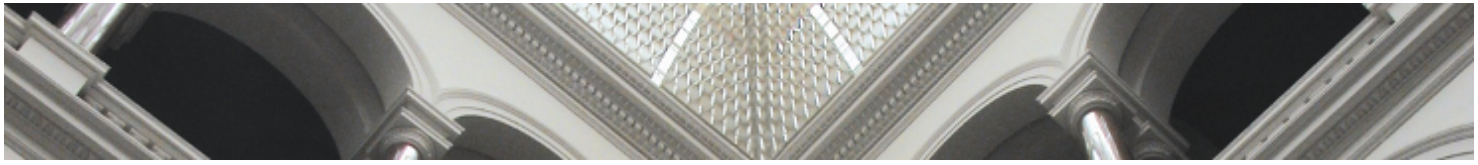
Technische Universität Berlin

Principal Component Analysis



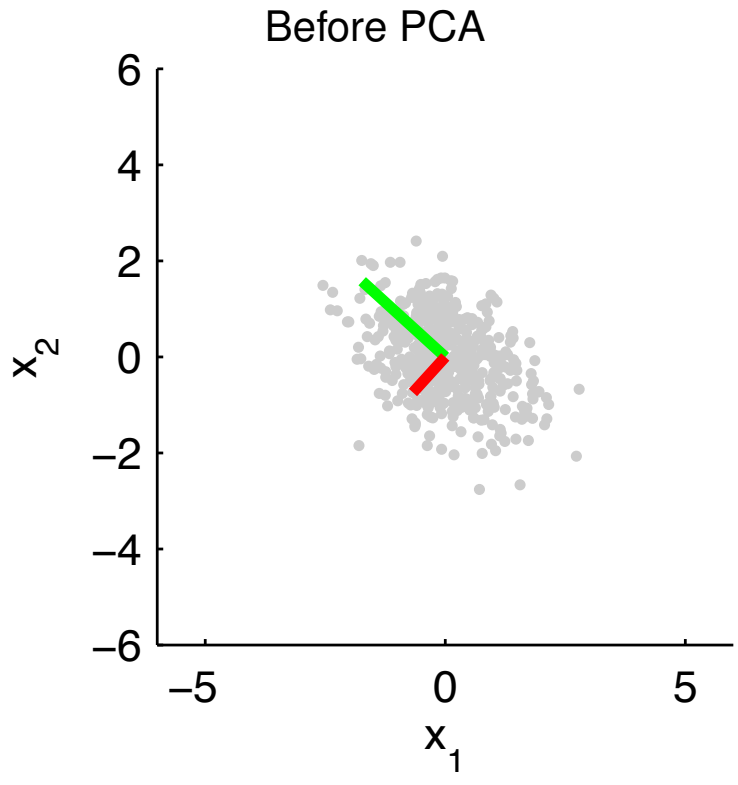


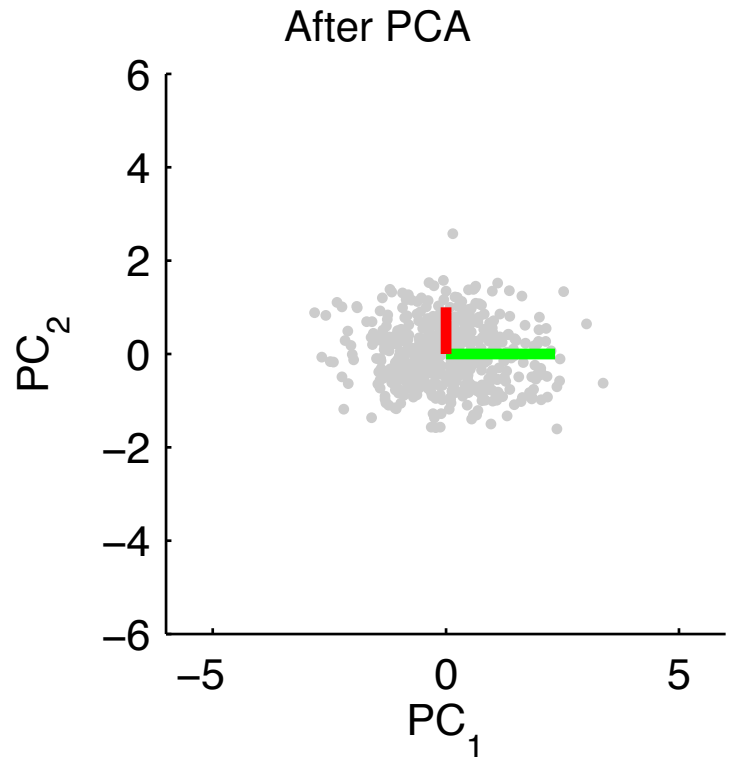




Technische Universität Berlin

Principal Component Analysis

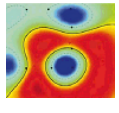


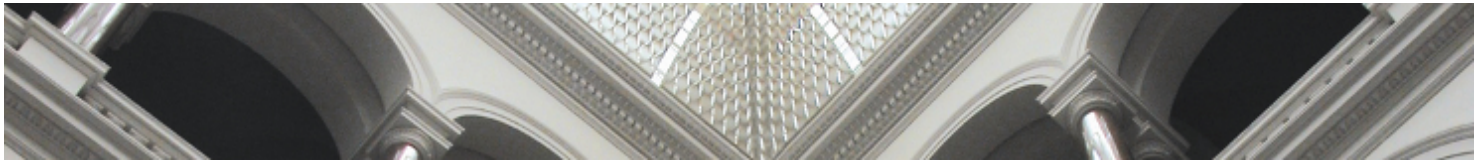


PCA aligns maximum variance directions with standard basis

$\rightarrow$ Variance along each dimension is **uncorrelated**

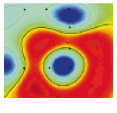
$\rightarrow$ Now we can remove each dimension separately

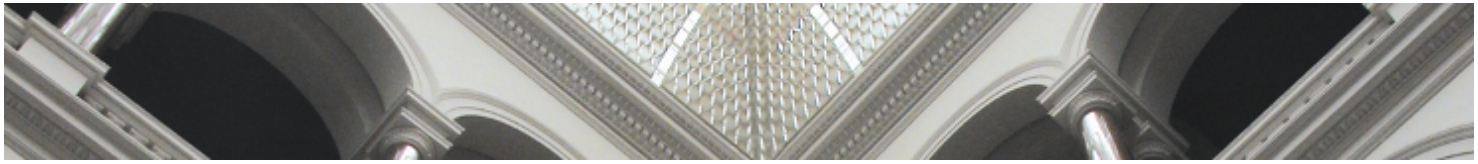




Technische Universität Berlin

Agenda

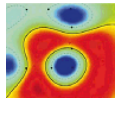


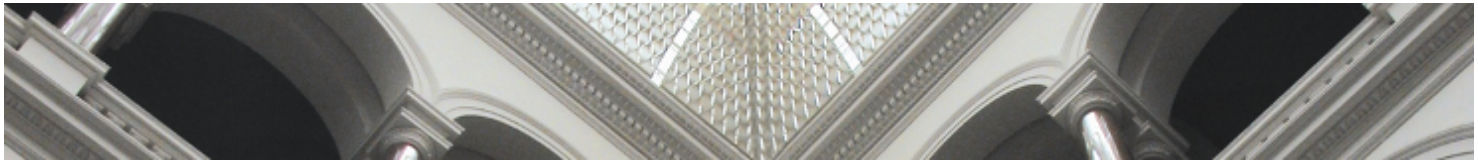


Technische Universität Berlin

Non-Negative Matrix Factorization

- For some data PCA does not make sense
- Example: Non-negative data
  - Principal directions will have negative entries
  - This can be hard to interpret
- Many data sets are strictly positive
  - Text data
  - Image data
  - Probabilistic data
- Very easy to implement

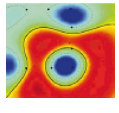


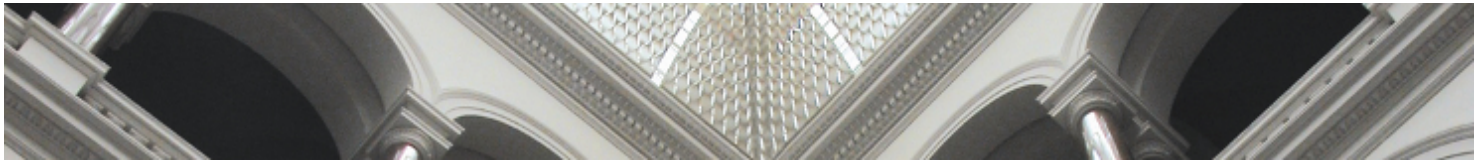


Non-Negative Matrix Factorization

Given non-negative data $X \in \mathbb{R}_+^{D \times N}$ we want to find $W \in \mathbb{R}_+^{D \times C}$, $H \in \mathbb{R}_+^{C \times N}$ such that

$$\arg\min_{W,H} ||X - WH||_{\text{Fro}}^2 = \arg\min_{W,H} \sum_{d=1}^{D} \sum_{n=1}^{N} (X_{dn} - (WH)_{dn})^2$$

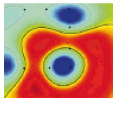


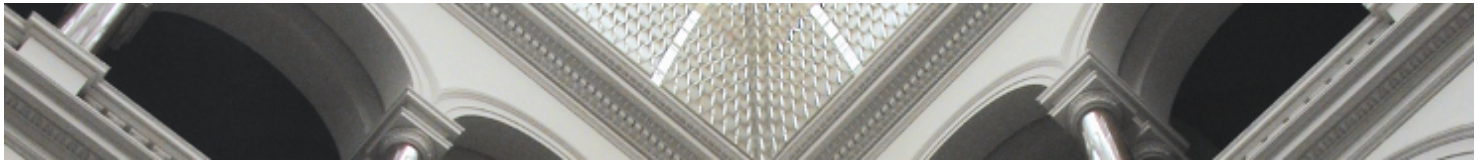


Technische Universität Berlin

Non-Negative Matrix Factorization

– Local minima





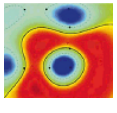
Technische Universität Berlin

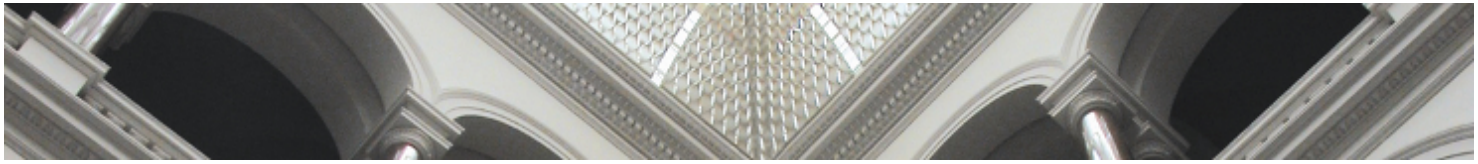
Non-Negative Matrix Factorization

- Local minima
- Gradient descent finds an optimal solution by iterating

$$H \leftarrow H + \eta \left(W^\top WH - X^\top W \right)$$

$$W \leftarrow W + \eta \left(WHH^\top - XH^\top \right)$$





Technische Universität Berlin

Non-Negative Matrix Factorization

- Local minima
- Gradient descent finds an optimal solution by iterating

$$H \leftarrow H + \eta \left(W^\top WH - X^\top W \right)$$

$$W \leftarrow W + \eta \left(WHH^\top - XH^\top \right)$$

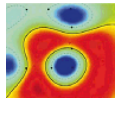
- By choosing η wisely one can transform the additive updates into multiplicative ones:

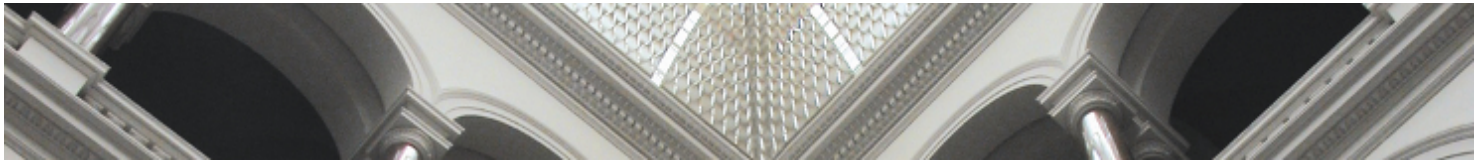
$$H = H \odot W^\top X \oslash W^\top WH$$

$$W = W \odot XH^\top \oslash WHH^\top$$

where

- $\odot$ is *element-wise* multiplication (in python '*')
- $\oslash$ is *element-wise* division (in python '/')



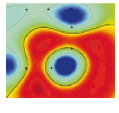


NMF Algorithm

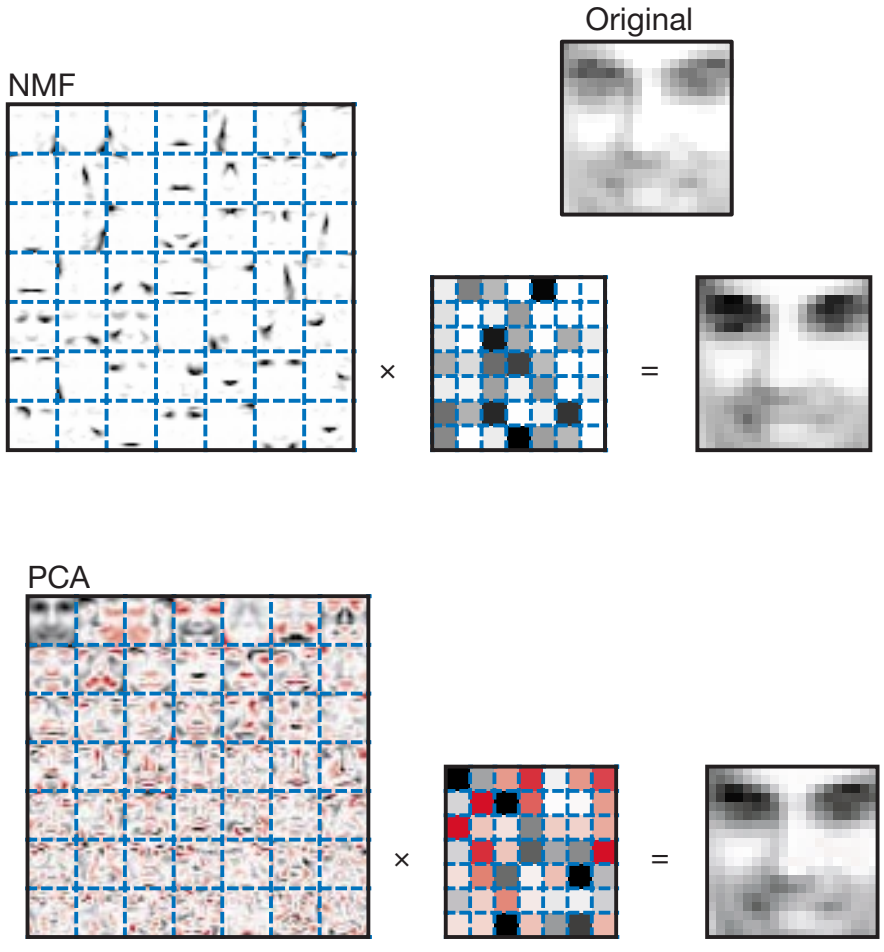
Require: data $X = [x_1, \ldots, x_N] \in \mathbb{R}_+^{D \times N}$, number of factors k

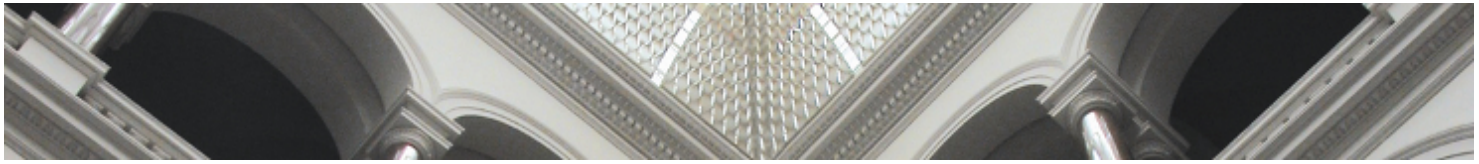
1: # Initialize $W \in \mathbb{R}_+^{D \times k}$, $H \in \mathbb{R}_+^{k \times N}$ randomly
2: # Add a small constant $\epsilon = 10^{-19}$ to X to avoid zero-divisions
3: **for** it $\leq$ Iterations **do**
4: $\quad H = H \odot W^\top X \oslash W^\top WH$
5: $\quad W = W \odot XH^\top \oslash WHH^\top$
6: **end for**
7: **return** W, H

Algorithm 2: Non-negative Matrix Factorization



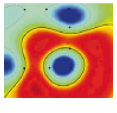
NMF Applications - Face Parts

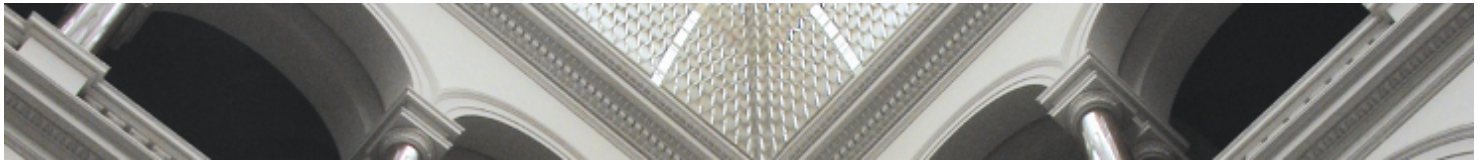




Technische Universität Berlin

Agenda





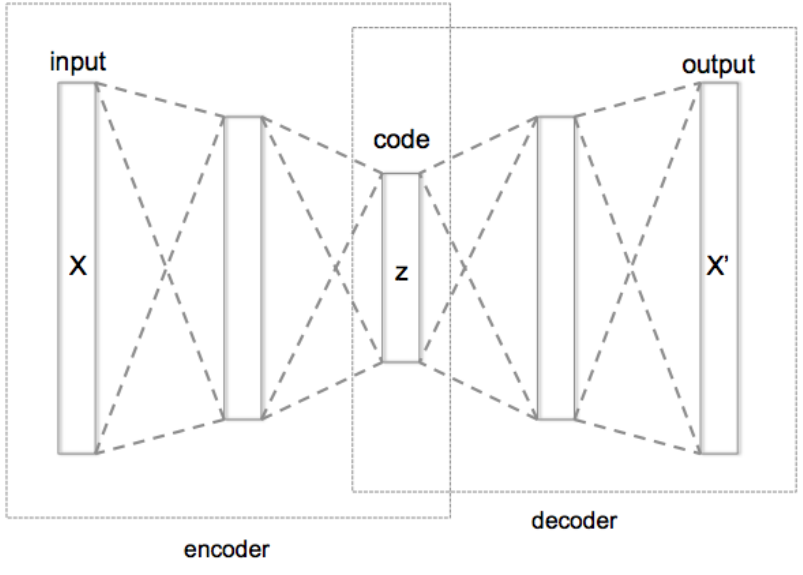
Technische Universität Berlin

Autoencoder

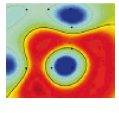
Two parts:

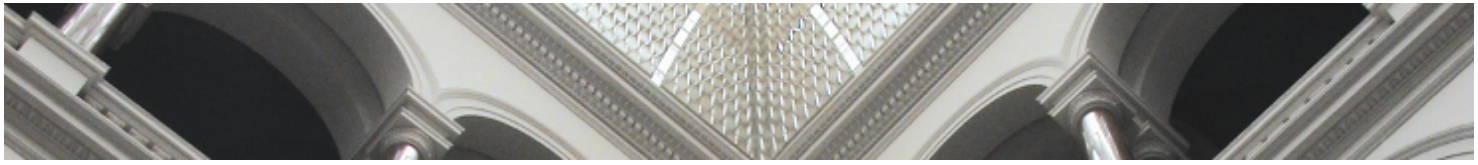
Encoder: A neural network that forces the input into a low-dimensional ”code”

Decoder: A neural network that tries to reconstruct the input from the ”code”



Wikipedia autoencoder



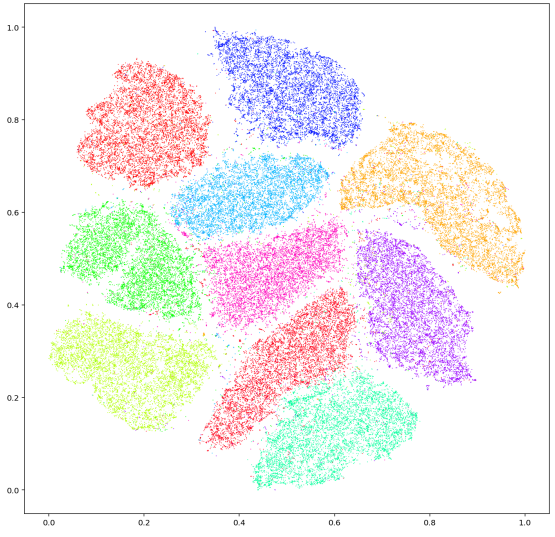


Technische Universität Berlin

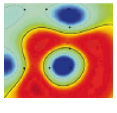
t-SNE (t-distributed Stochastic Neighbor Embedding)

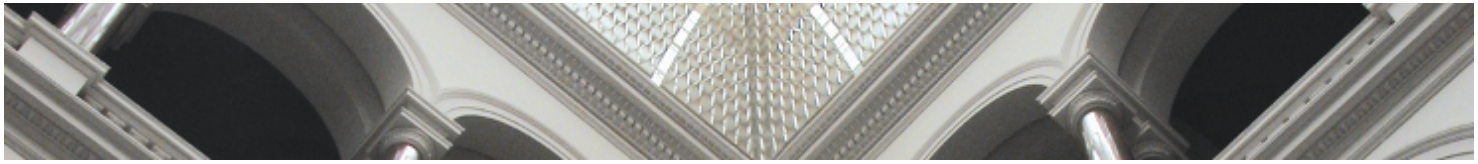
Unlike Autoencoder, non-parametric, i.e. can not easily embed new points.

Local similarities in the data preserved, higher distances not preserved.



Wikipedia t-SNE. A visualization of MNIST in two t-SNE dimensions.





Technische Universität Berlin

References

Duda, Richard O, Peter E Hart, and David G Stork (2012). **Pattern classification**. John Wiley & Sons.

Guérin, Joris et al. (Mar. 2017). "Clustering for Different Scales of Measurement - the Gap-Ratio Weighted K-means Algorithm". en. In: **arXiv:1703.07625 [cs, stat]**. arXiv: 1703.07625. URL: http://arxiv.org/abs/1703.07625 (visited on 08/29/2018).

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine learning in Python". In: **Journal of machine learning research** 12.Oct, pp. 2825–2830.

Schubert, Erich et al. (2017). "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN". In: **ACM Transactions on Database Systems (TODS)** 42.3, p. 19.