Foundations and Core Concepts

Generative Artificial Intelligence

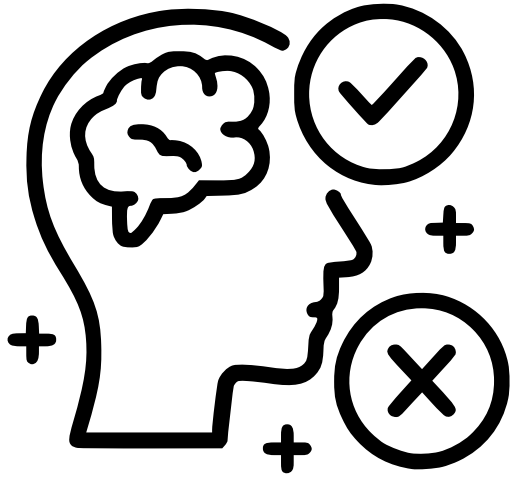
M. Sc. Angewandte Informatik

Prof. Dr. Marc Aubreville

Hochschule
Flensburg
University of
Applied Sciences

Generative vs. Discriminative Models

- Machine Learning models can be broadly subdivided into two categories:



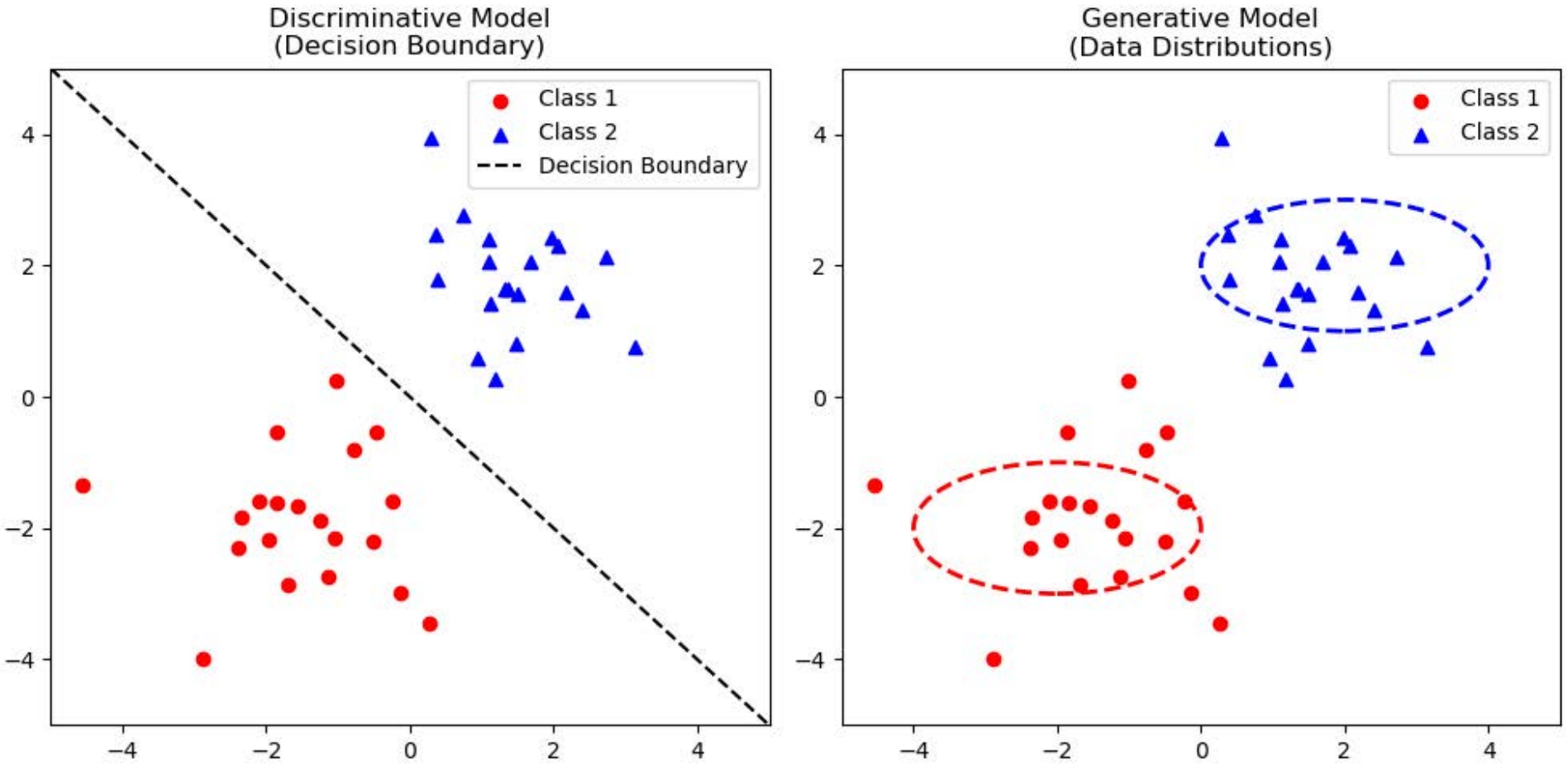
The get a sample and learn to discriminate it against others

(learn boundaries between classes)

They generate instances of the data distribution

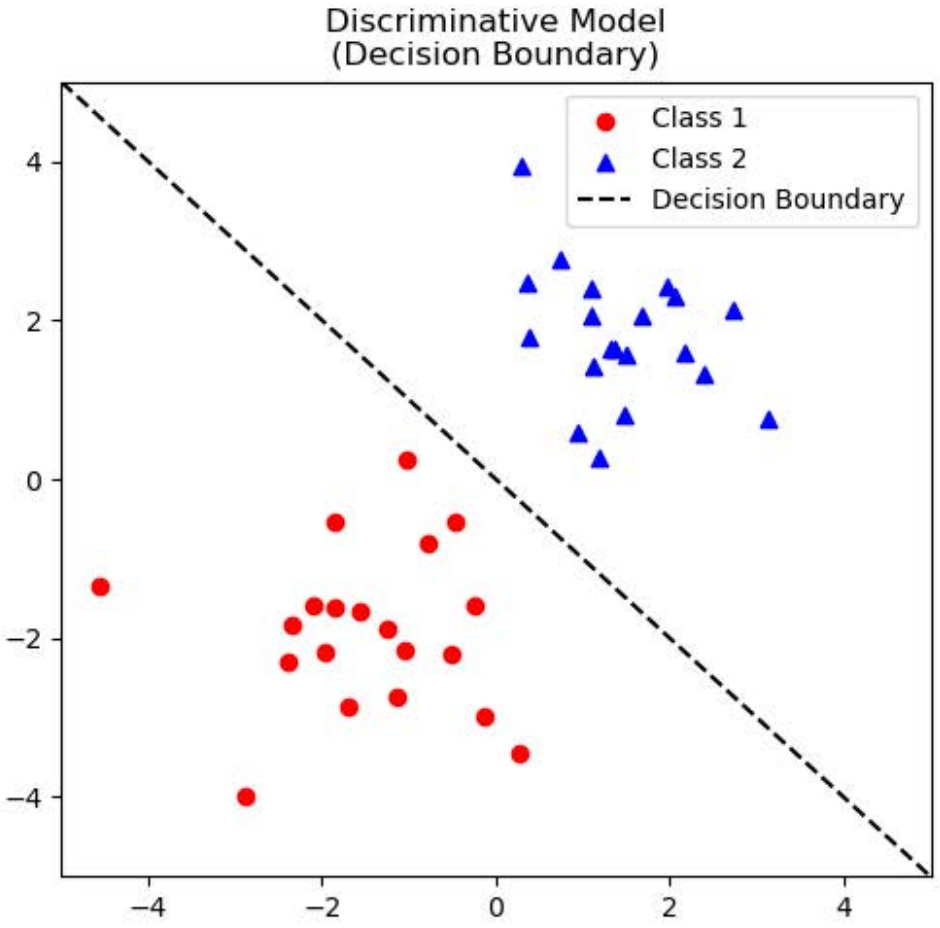
(learn data distributions)

Generative vs. Discriminative Models



Hochschule Flensburg
University of Applied Sciences

Discriminative Models



Discriminative Models model a decision boundary.

They try to estimate the conditional probability:

$P(y \mid x)$

y : class

x :data sample

Examples: Logistic regression, support vector machines, Deep Learning-based classifiers, random forests, ...

Generative Models

Hochschule Flensburg
University of Applied Sciences

Generative Models model a data distribution.

They try to model the joint probability:

$P(x, y)$

y : class

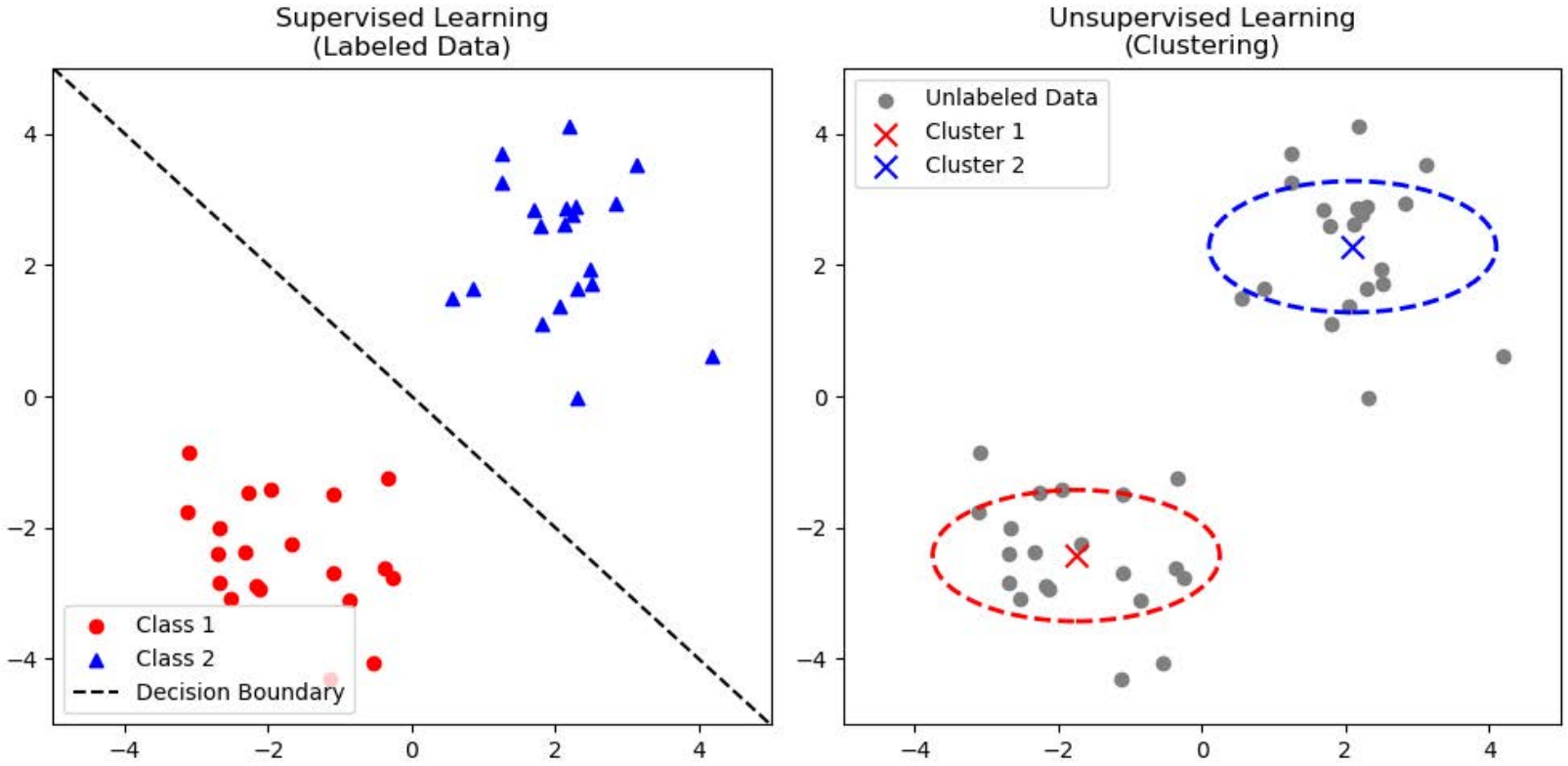
x :data sample

These models generate or simulate new data, based on old data they have seen during training.

Examples: Naive Bayes, Adversarial Networks, VAEs, Diffusion Models, GPT, ...

Supervised and Unsupervised Learning

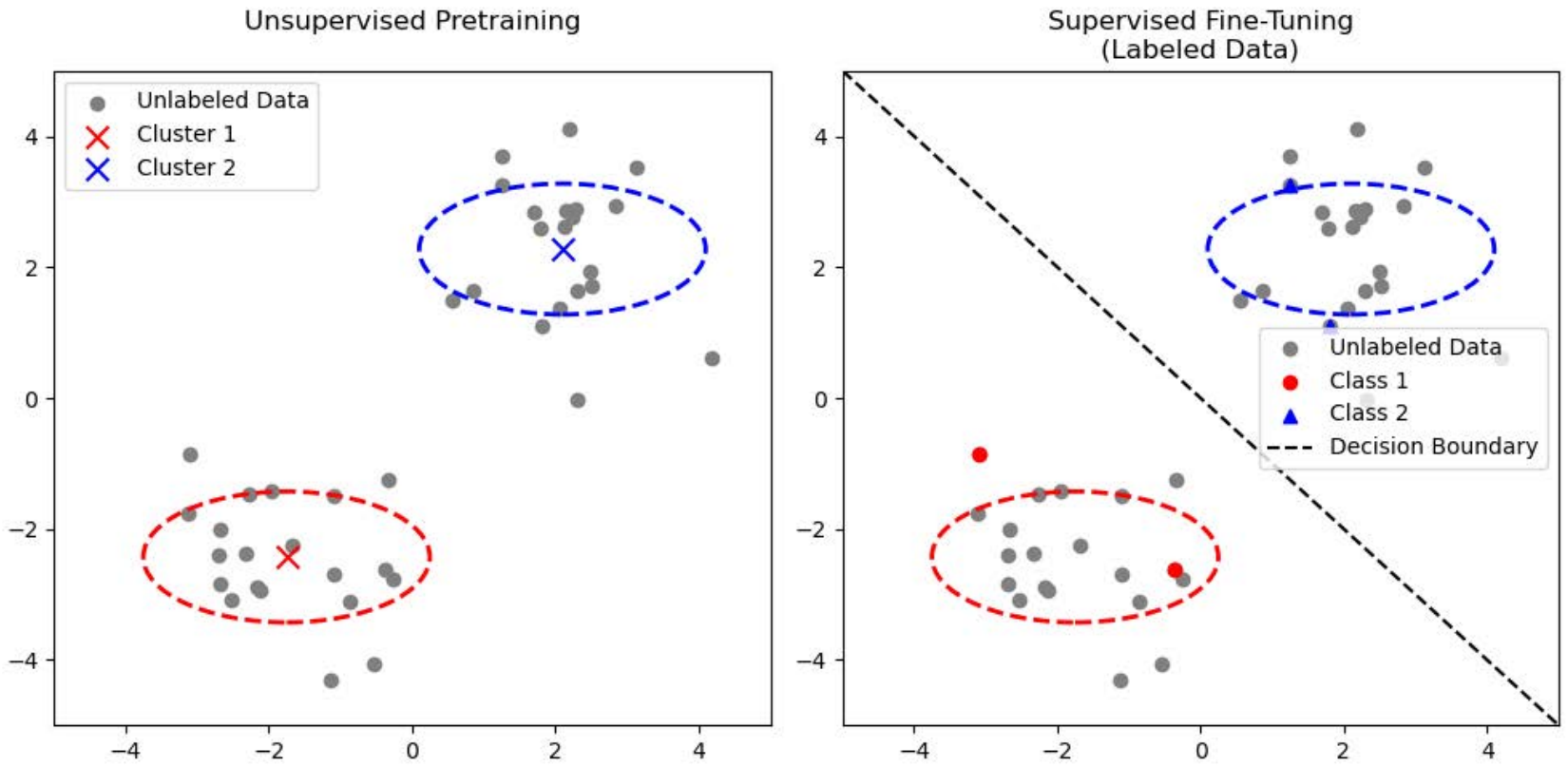
- In most machine learning scenarios today, we do have labeled training data.



- But especially in generative AI, unsupervised scenarios play a major role.

Combining Unsupervised and Supervised

- Many recent advances in AI have been made possible by a combination of unsupervised model pretraining and supervised model fine-tuning.



Self-Supervised Learning (more to that later)

Hochschule Flensburg
University of Applied Sciences

- Self-supervised learning is a form of unsupervised learning that tries to make use of the inherent structure in data.
- We can use these techniques to make use of large-scale unlabeled datasets.
- Self-supervised learning is key step in the training of most large models (large language models, foundation models, ...)
- The key idea is that a model has to fulfill a task that can be generated from data alone, e.g.:
  - Find out which word is missing in a gap text
  - Find out if two images are cropped from the same image
- By solving this task, the model has to understand the structure of the data.
- Goals:
  - Learn a latent space that captures semantic or structural features
  - Enable transfer learning on downstream tasks with few or no labels

Representation Learning

- Representation learning is about learning useful, compressed descriptions of raw data.
- Aim: Transform high-dimensional raw inputs into compact, meaningful, and task-relevant features.
- Good representations capture structure in the data
- They separate relevant from irrelevant variations
- They enable:
  - Better generalization
  - Transfer to new tasks (few-shot / zero-shot)
  - More efficient learning
- Self-supervised learning is one technique for representation learning.

Hochschule
Flensburg
University of
Applied Sciences

Let's go into latent space.

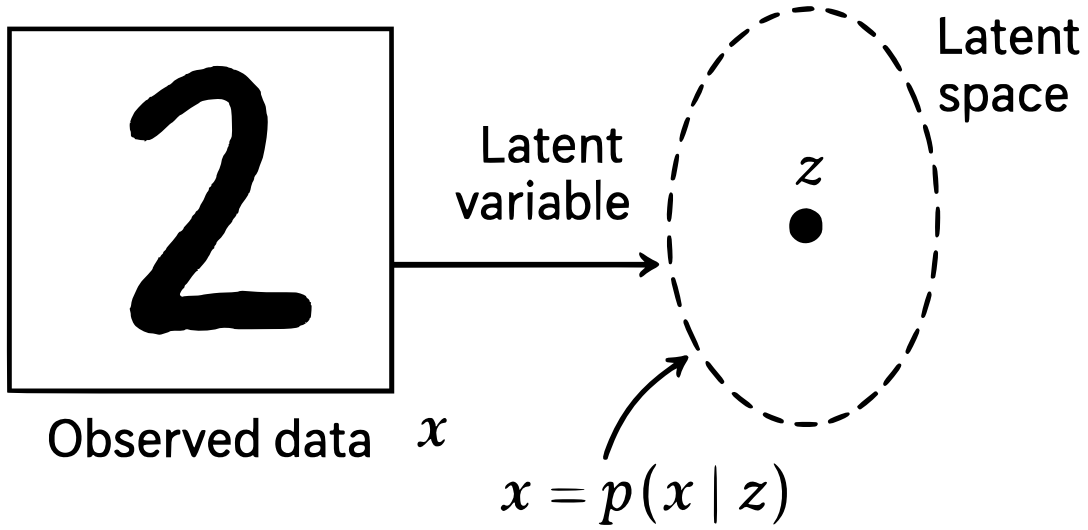
What is latent space?

Hochschule Flensburg
University of Applied Sciences

- Latent space is an
  - abstract, commonly low-dimensional vector space, which
  - extracts the important features of data (for instance: images, texts, audio).

- A deep learning model transforms input data into the latent space to extract important information.

- This can be thought of as compression, where we throw away redundant information and keep all that's needed to perform the desired task
  Example: To recognize the number 2 the model can completely ignore the background color.



Constructing the latent space

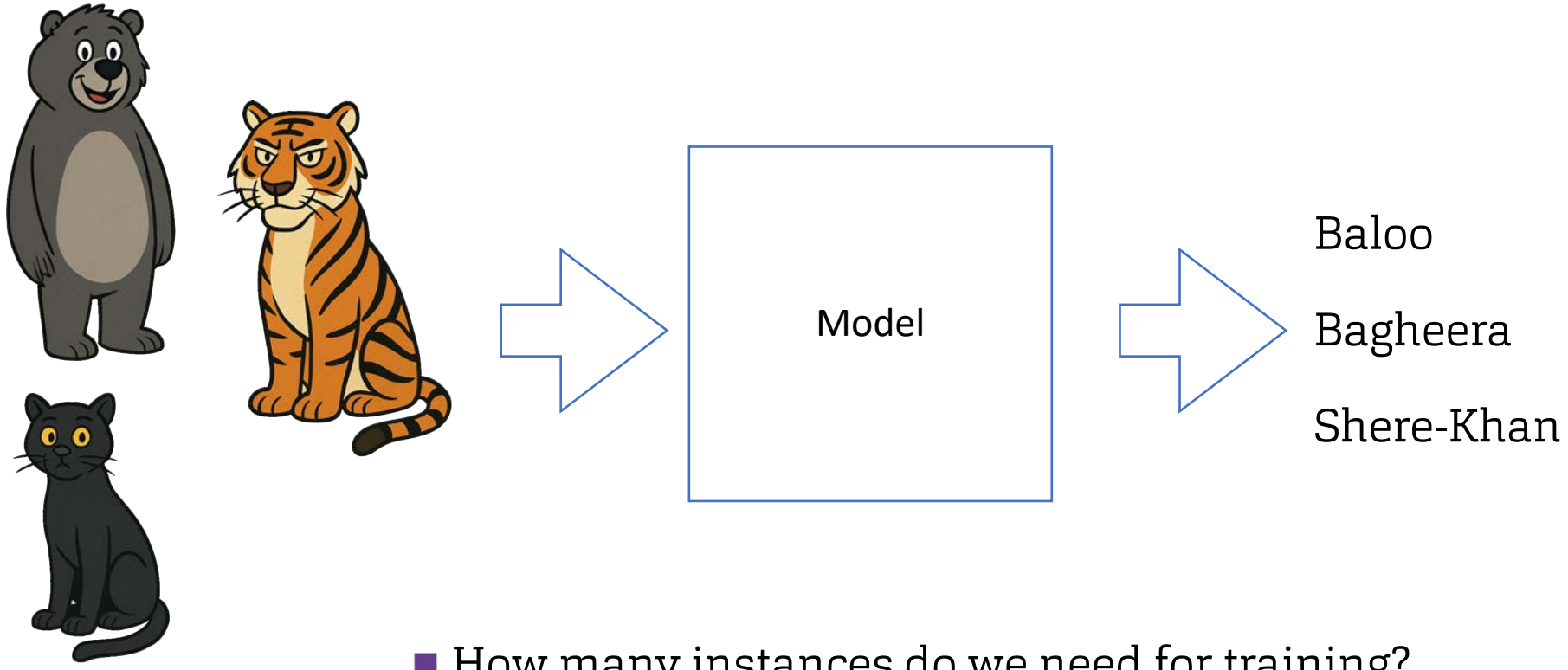
Hochschule Flensburg
University of Applied Sciences

- How does a model learn to construct the latent space?
  - It does because it needs to.
  - We define the model architecture, and a loss function, and at the end the model has to perform a prediction that achieves a low loss.

- The organization of the latent space is thus always a **consequence** of model training.
- Ideally, we want our latent space to be:
  - Free of redundancies
  - Represent all important features of the data
  - Not overfit to the training data

Why we need this

- Let's assume we want to classify a bunch of images of (comic) animals:

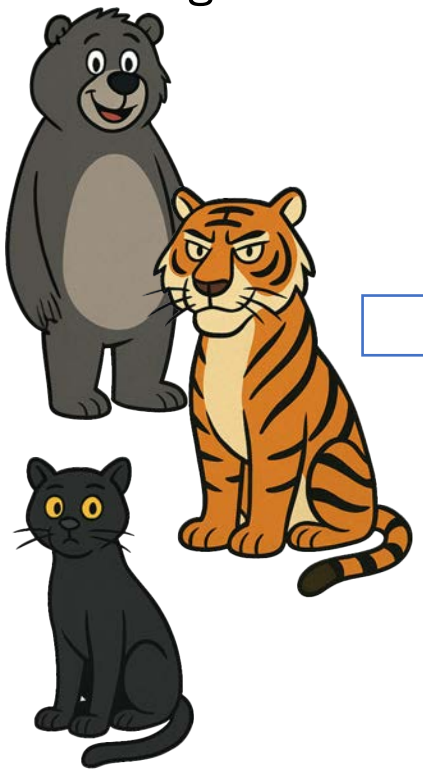


- How many instances do we need for training?

A significant amount.

What to do with a well-organized latent space

- Let's assume we have a model that generated a bunch of latent variables (features) from images:



Model

fur color

fur brightness

(very small) Model

Baloo

Bagheera

Shere-Khan

Let's revisit that

- In the example, we have a latent space that is highly discriminatory of our problem.
- It extracts essential features.
- We can build a very small model using only a small amount of data instances to find a well-generalizing solution.
- So we are able to
  - Find a well-performing solution.
  - Avoid overfitting.

Hochschule Flensburg
University of Applied Sciences

So: How to do that?

Hochschule Flensburg
University of Applied Sciences

A naive idea: Learning the latent space supervised

Hochschule
Flensburg
University of
Applied Sciences

- We could get a bunch of labels

fur brightness:	0.3	1	0	0.8
fur color:	0.1	0.2	0.1	0.5

- And then train a model to predict this "latent space" as a regression task.
- But then we have even more effort and need even more samples to learn from.

Learning a latent space

- How we typically learn a latent space is different though:
  - We try to achieve an organization of the latent space by setting a loss function that either:
    - Directly acts in the latent space (e.g., Triplet Loss, Contrastive Loss, InfoNCE)
      This enforces relationships between the embeddings
    - Act indirectly via a supervised reconstruction or prediction task
      This forces the model to establish the latent space in a way so that it can solve the reconstruction or prediction task (i.e., that it contains the relevant information for this).
  - We can also combine both approaches.
- We will learn a lot about these methods later in the lecture.

Hochschule Flensburg
University of Applied Sciences

Hochschule
Flensburg
University of
Applied Sciences

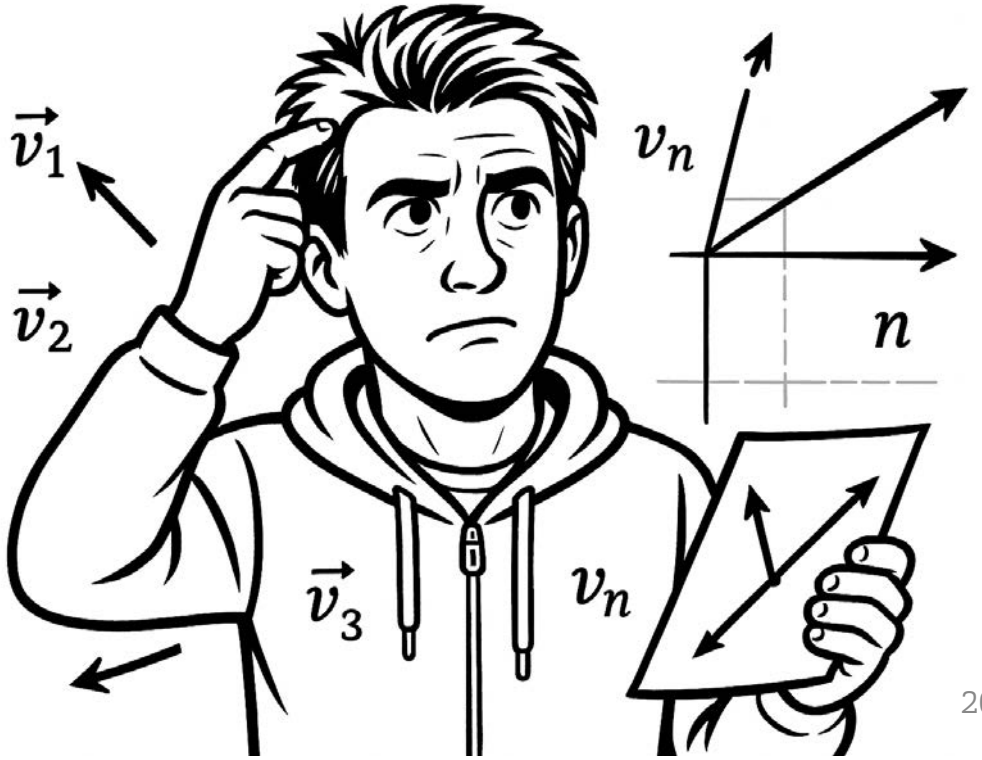
Interpretation of latent variables

(Dimensionality reduction)

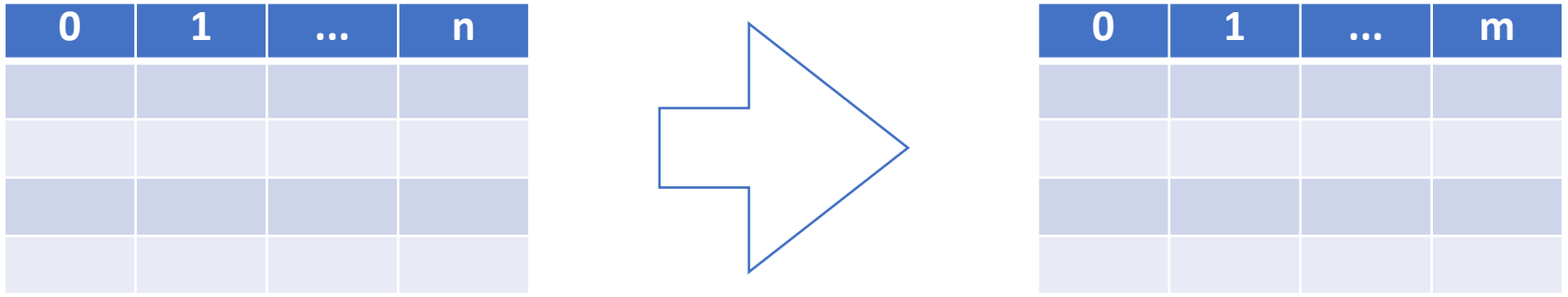
Making sense of n-dimensional vectors

Hochschule Flensburg
University of Applied Sciences

- Typically, our models are built so they construct n-dimensional embeddings / latents from input data.
- This makes them really hard to visualize.
  - We typically can only visualize in 2 or maybe 3 dimensions.
  - To make sense of higher-dimensional latent spaces, we rely on **dimensionality reduction** techniques.
  - These will reduce a n-dimensional vector into some smaller dimensionality that we can then analyze / visualize.



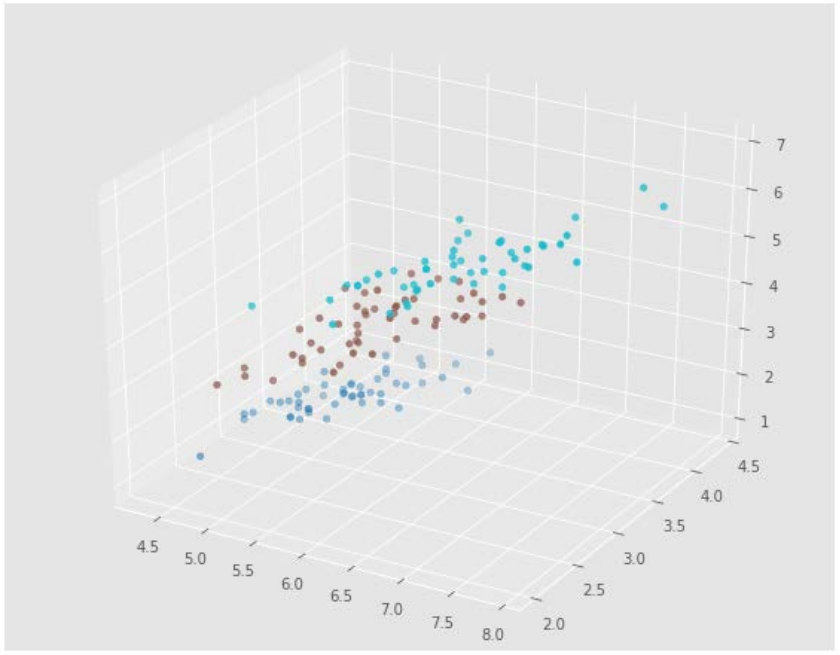
What are dimensionality reduction methods?

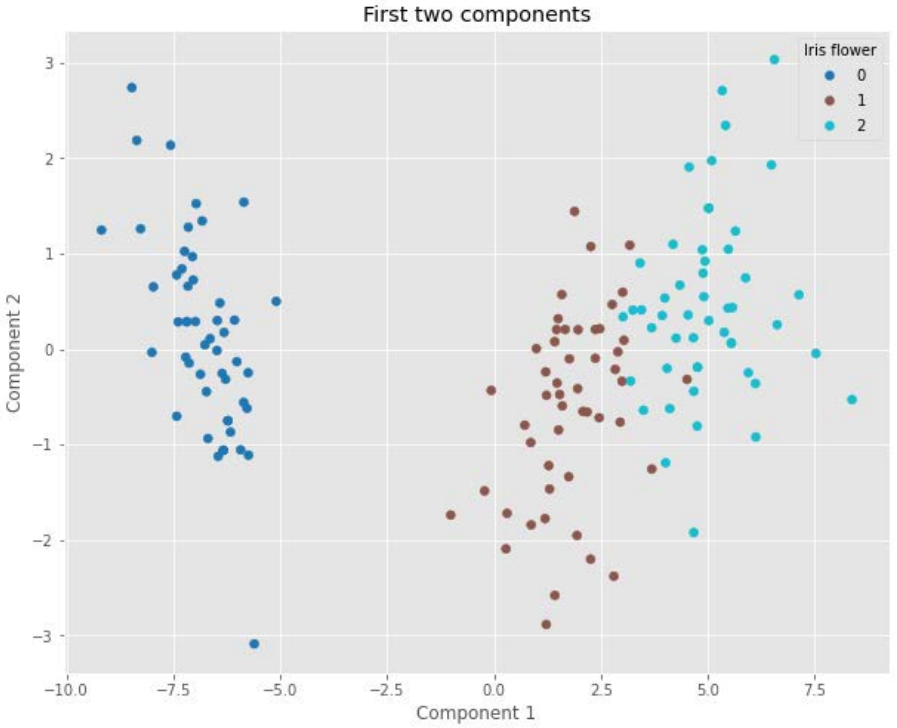


- The core idea is to reduce the dimensionality of our samples from n to m (with $m \ll n$).
- Side conditions:
  - We do not want to loose any (important) information
  - The new (reduced) vector should describe the data as good as the original one.

Example: Reduction 3 -> 2 dimensions

Hochschule Flensburg
University of Applied Sciences





When to apply dimensionality reduction

Hochschule Flensburg
University of Applied Sciences

- We typically apply dimensionality reduction techniques:
  - To visualize high dimensional data.
  - To reduce the data size (in disk storage or RAM).
  - To escape the "curse of dimensionality" for high-dimensional machine learning problems.
  - To reduce the likelihood of overfitting.

Recap: Curse of dimensionality

Hochschule Flensburg
University of Applied Sciences

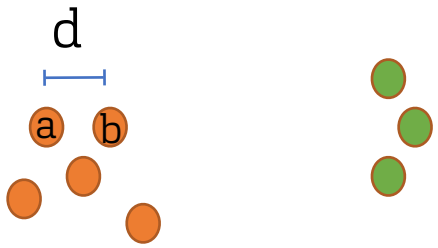
- In machine learning, the amount of training data required increases exponentially with the number of features or dimensions in the data set.

- Background:
  - With each additional dimension in the data set, the number of possible combinations of features increases exponentially.
  - This makes it more difficult to collect enough training data to train a model that is capable of making accurate predictions.

- Additionally:
  - The distance in feature space between two data points increases the higher the dimension of the feature space (see next slide).

Recap: Curse of dimensionality

- The distance between to samples is commonly a measure of their similarity.



- Let's assume a simple distance metric such as L2 for now:

$$d(a,b) = \sqrt{\sum_i b_i^2 - a_i^2} \quad (\mathbf{a} = [a_0, a_1, \dots], \mathbf{b} = [b_0, b_1, \dots])$$

- Now, if we assume to add another (in this case: third) dimension with random values, how will the distance change?

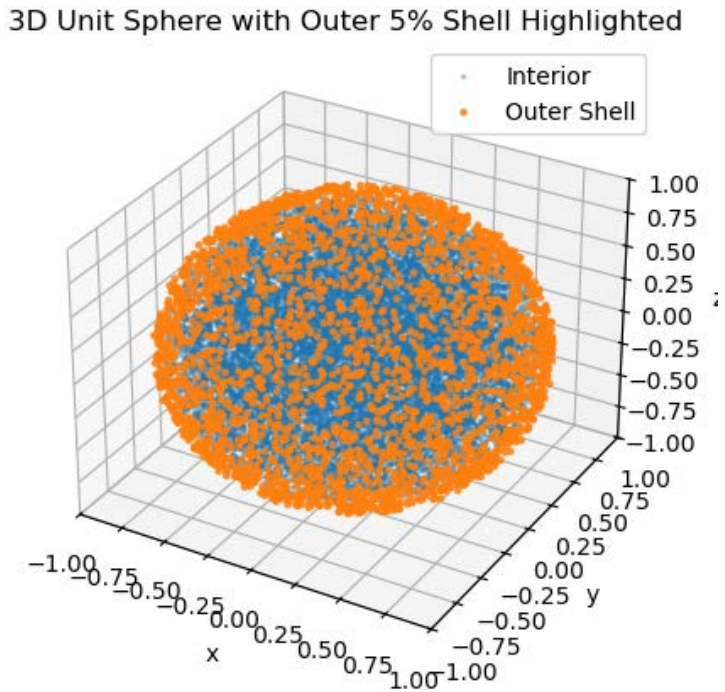
Recap: Curse of dimensionality

Hochschule Flensburg
University of Applied Sciences

- If we increase the number of dimensions to N (with non-zero values in the new dimensions), the distance will go up for all samples:

$$d(a,b) = \sqrt{\sum_{i}^{N} b_i^2 - a_i^2}$$

- This means that once we add more dimensions, the distance metric becomes less discriminatory.
- For high dimensional spaces, the distances between all points become equal.
- We can think about this like a (n-dimensional) Sphere, where all points are distributed equally.



Overview of dimensionality reduction techniques

Hochschule Flensburg
University of Applied Sciences

- There are a couple of unsupervised techniques that are commonly used for this:
  - Feature selection: Several dimensions are selected, others are discarded
  - PCA (Principal Component Analysis): Projects the latent space onto the directions of highest variance.
  - t-SNE (t-distributed Stochastic Neighbor Embedding): Focuses on preserving local neighborhoods when mapping high-dimensional points into 2D/3D.
  - UMAP (Uniform Manifold Approximation and Projection):
    A more recent method that balances local and global structure better than t-SNE, often used in modern embedding visualization.
  - Autoencoder:
    A method that uses a model that can reconstruct input data and has an information bottleneck in-between.

Hochschule
Flensburg
University of
Applied Sciences

Principal Component Analysis (PCA)

PCA: Use cases

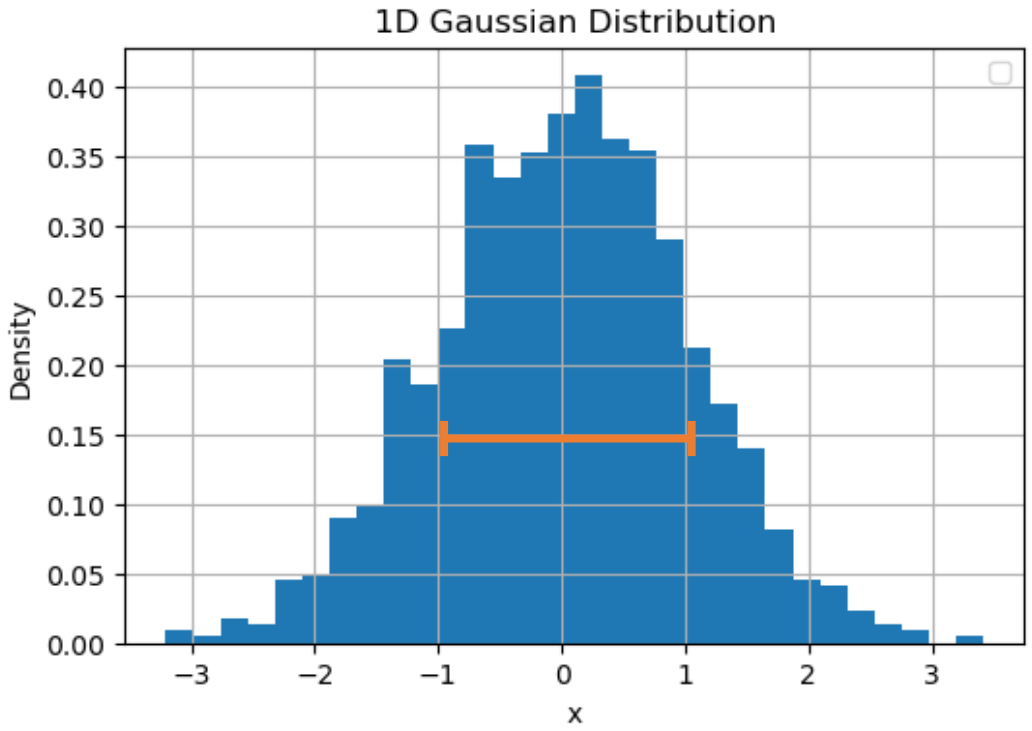
- Dimensionality reduction – simplify datasets with many variables while preserving most of the information (what we are here for)
- Noise reduction / denoising – filter out small, irrelevant variations in data
- Visualization – project high-dimensional data into 2D or 3D for easier interpretation
- Feature extraction – generate uncorrelated features (principal components) for downstream models
- Compression – store or transmit data more efficiently by keeping only top components
- Preprocessing for machine learning – remove redundancy and multicollinearity between variables

Hochschule Flensburg
University of Applied Sciences

Let's look into some data properties first

Hochschule
Flensburg
University of
Applied Sciences

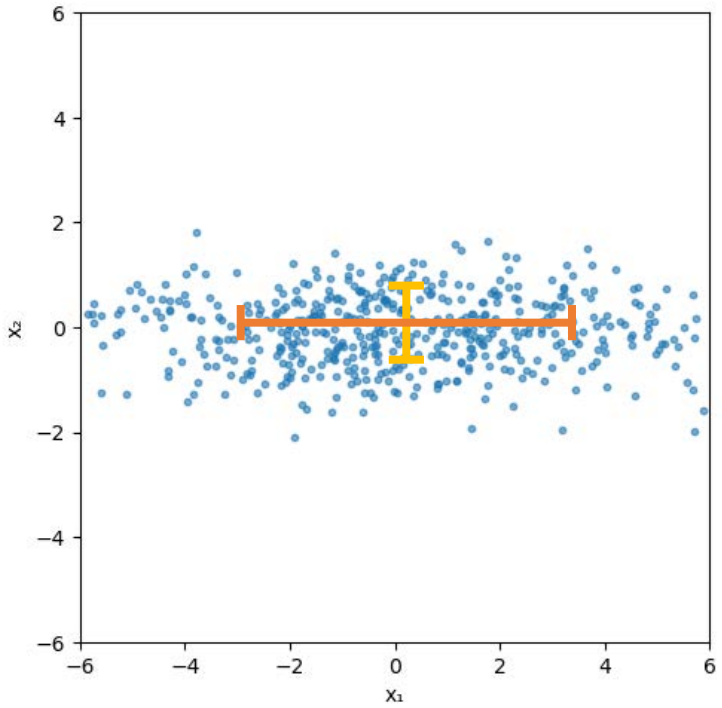
- Variance



- The spread of the data, i.e. how much it differs from its mean.
- Sample variance is defined as: $Var(x) = \frac{1}{n-1}\sum_{k=1}^{n}\left(x^{(k)} - \bar{x}\right)^2$

Sample variance in more dimensions

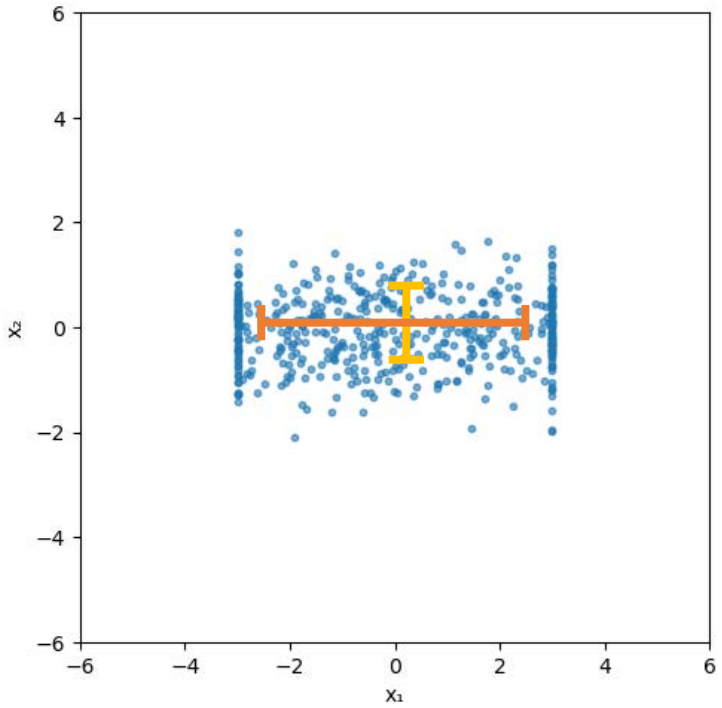
- We can determine data variance across every dimension of a signal



- $Var(x_i) = \frac{1}{n-1} \sum_{k=1}^{n} \left(x_i^{(k)} - \bar{x}_i \right)^2$

Variance is indicative of information.

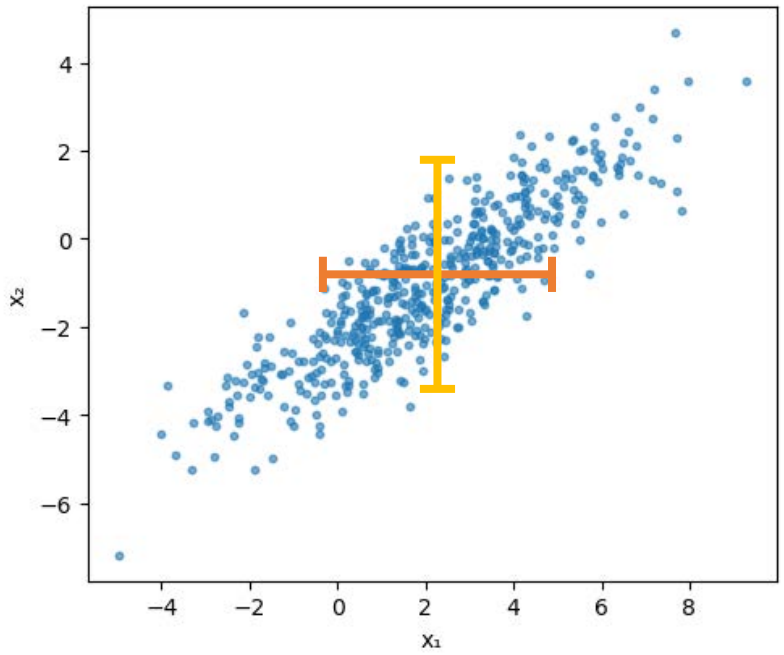
- If we block values from reaching higher values, we restrict their variance.

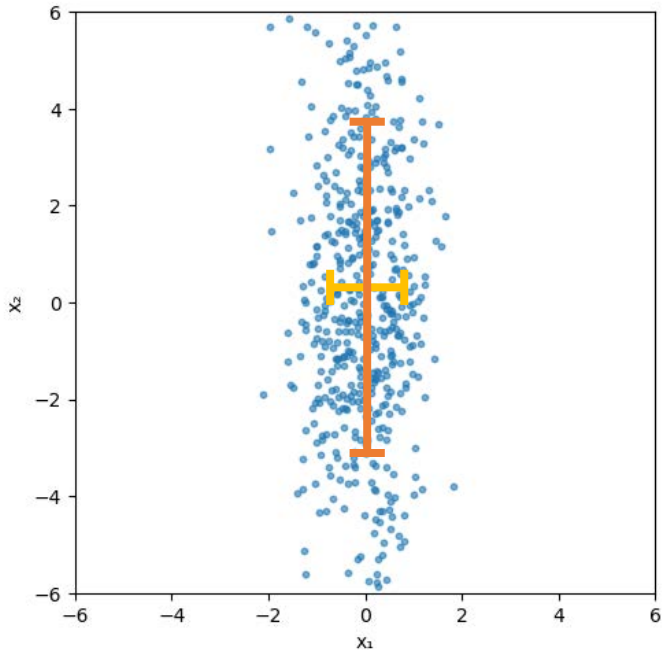


- Also: We lose information. Variance is indicative of information.

Changes in variance

- What do you think happens to the variance if we rotate the coordinate system?





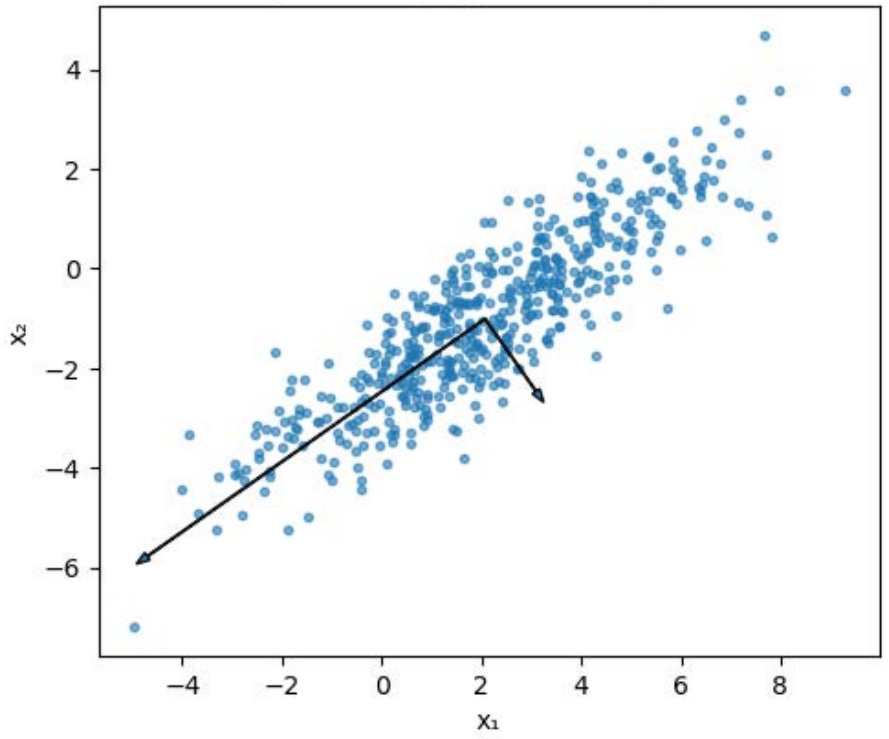
- The variance of every dimension changes accordingly.

Total sum of variance

- If we project one coordinate system into another, the total variance is constant.
- This means that by projecting it, no information get's lost.
- The translation is 100% invertible.

The goal of PCA

- We want to be able to transform the samples into a new coordinate system
- In this new coordinate system, the first axis points towards the direction of greatest variance.
- The second axis then points towards an orthogonal direction of second-greatest variance.

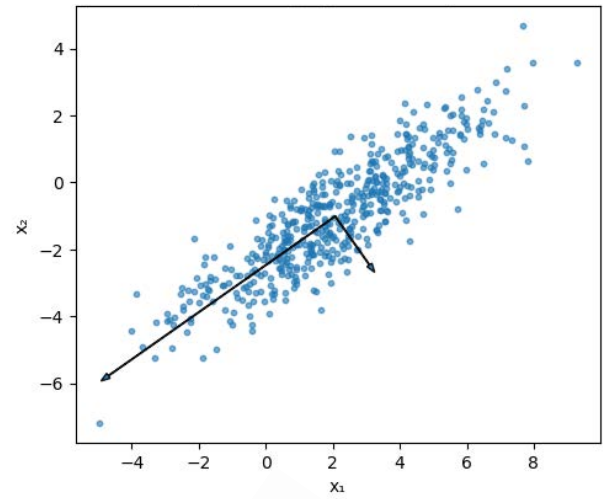


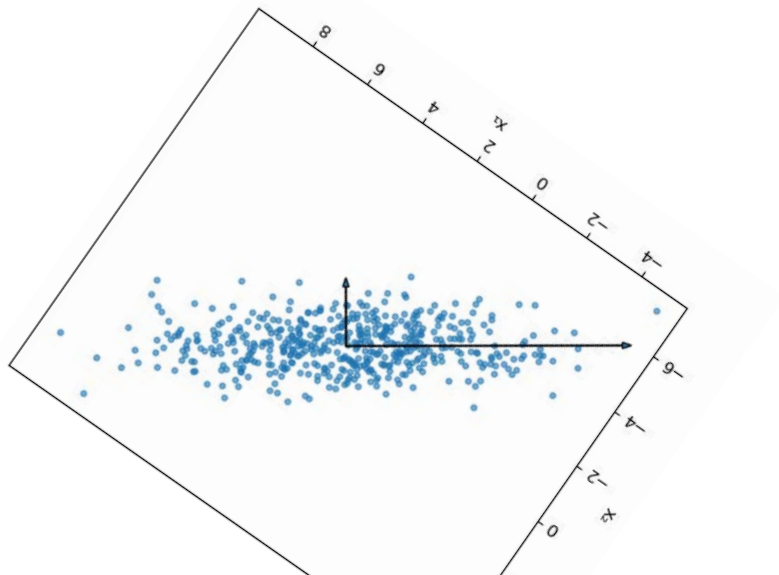
- Why should we do this?
  - We want to go from a high-dimensional problem into a lower-dimensional problem.
  - We want to loose as little information as possible by this.

(of course, later it's not 2 to 1, but with much larger numbers)

How we go about a PCA

- Goal: At the end, we want to have a coordinate system consisting of less dimensions (e.g., 2 or 3) that we can still visualize (or at least analyze).
- We want to find out a new coordinate system that that has basis vectors ordered by importance.
- The most important dimension is the one pointing into the direction of greatest variance.
- We thus want to find an orthogonal basis vector system that allows us to later reduce some of the components.

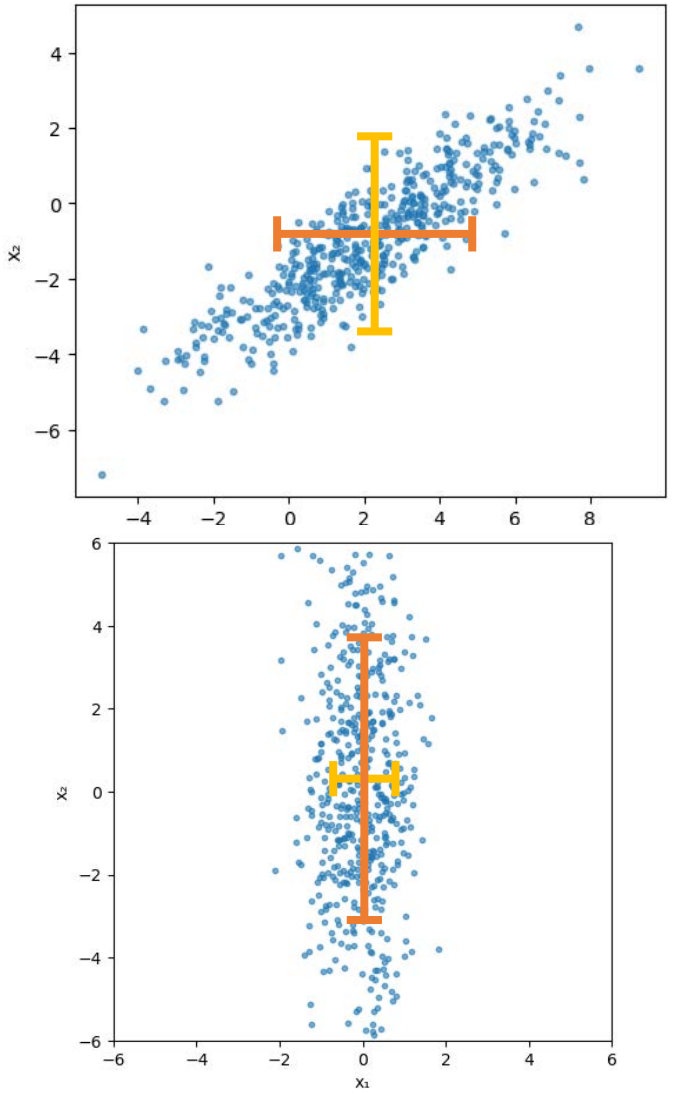




Removing components later on

- We want to have a smaller amount of dimensions, but lose only minor in variance (information!).
- This depiction tells us why:
  - If we remove the yellow axis in the top image, we loose a lot of variance (information)
  - If we do this on the bottom image, we loose far less.

Hence: It is sensible to first align the data according to the direction of highest variance, before we start removing dimensions.



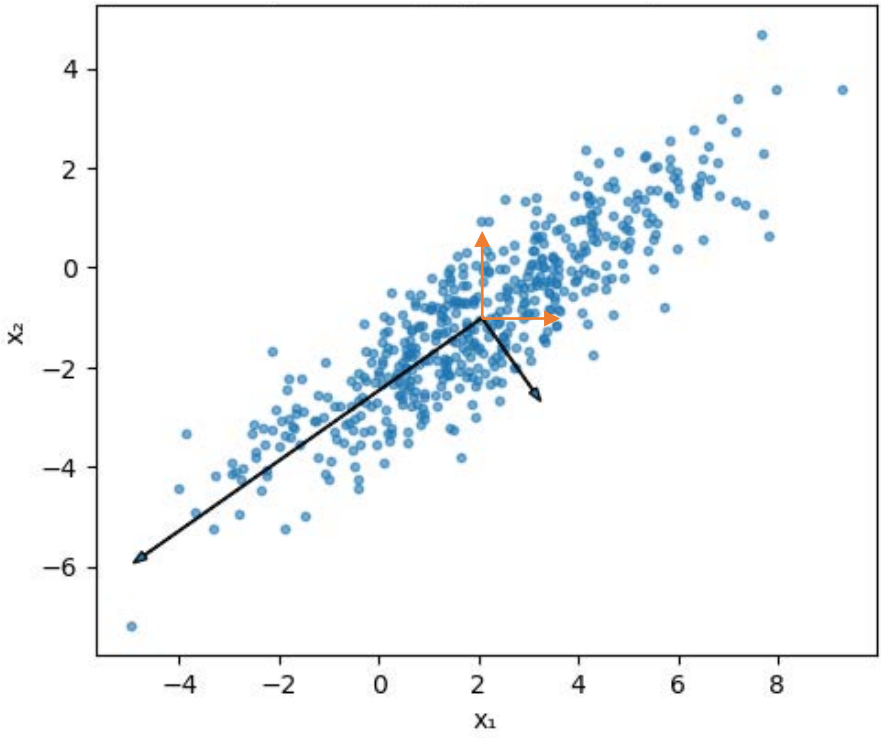
Hochschule Flensburg
University of Applied Sciences

A projection

- What we are doing here is nothing but a projection into a new coordinate system.

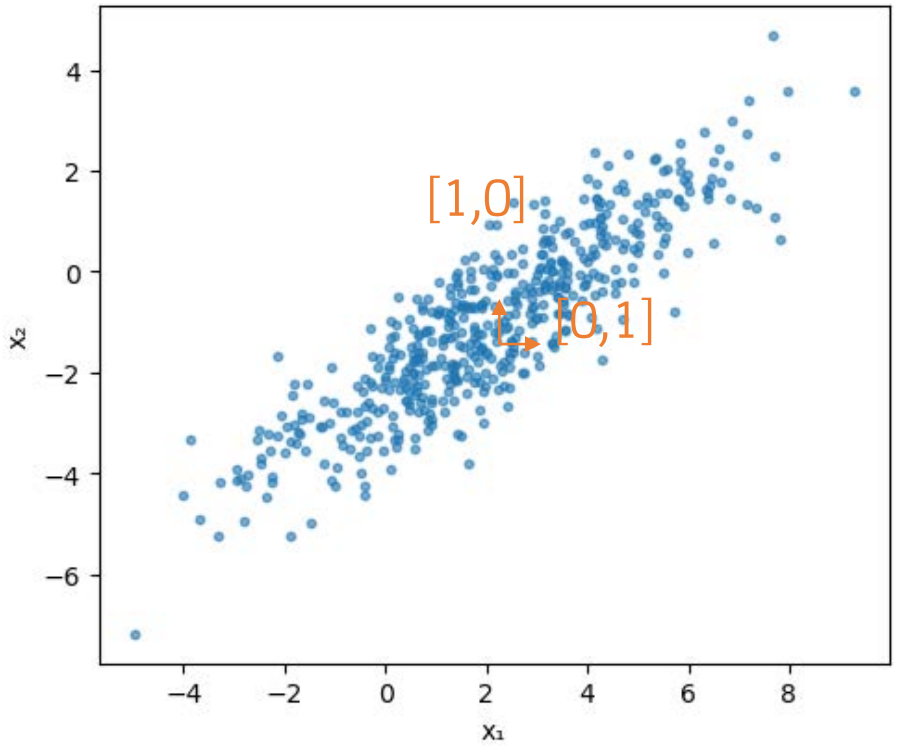
old coordinate system

- This means, that all of our samples change their values, as they now correspond to a new coordinate system.

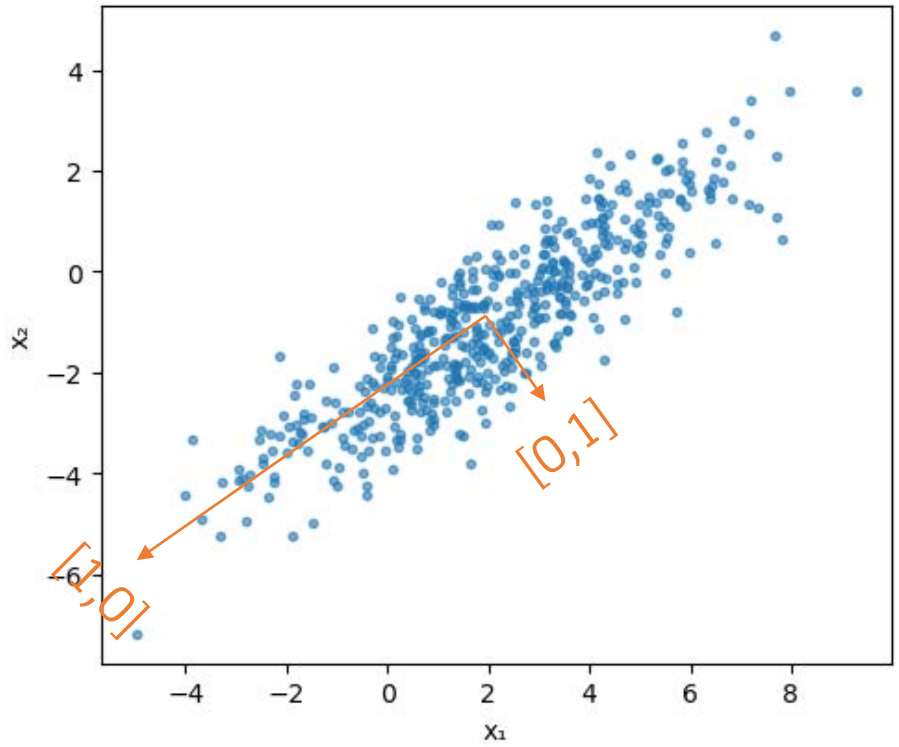


A projection

- Comparing coordinates in old vs. new coordinate system



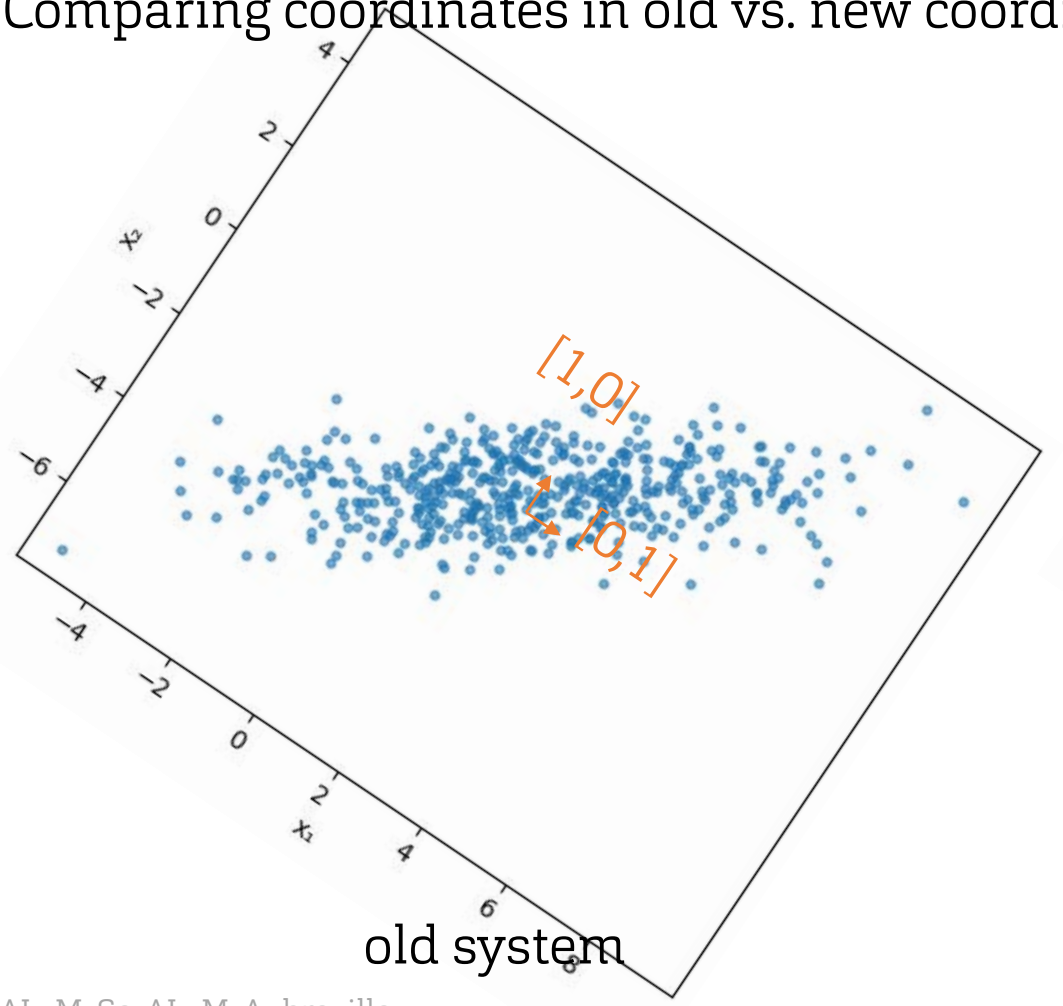
old system



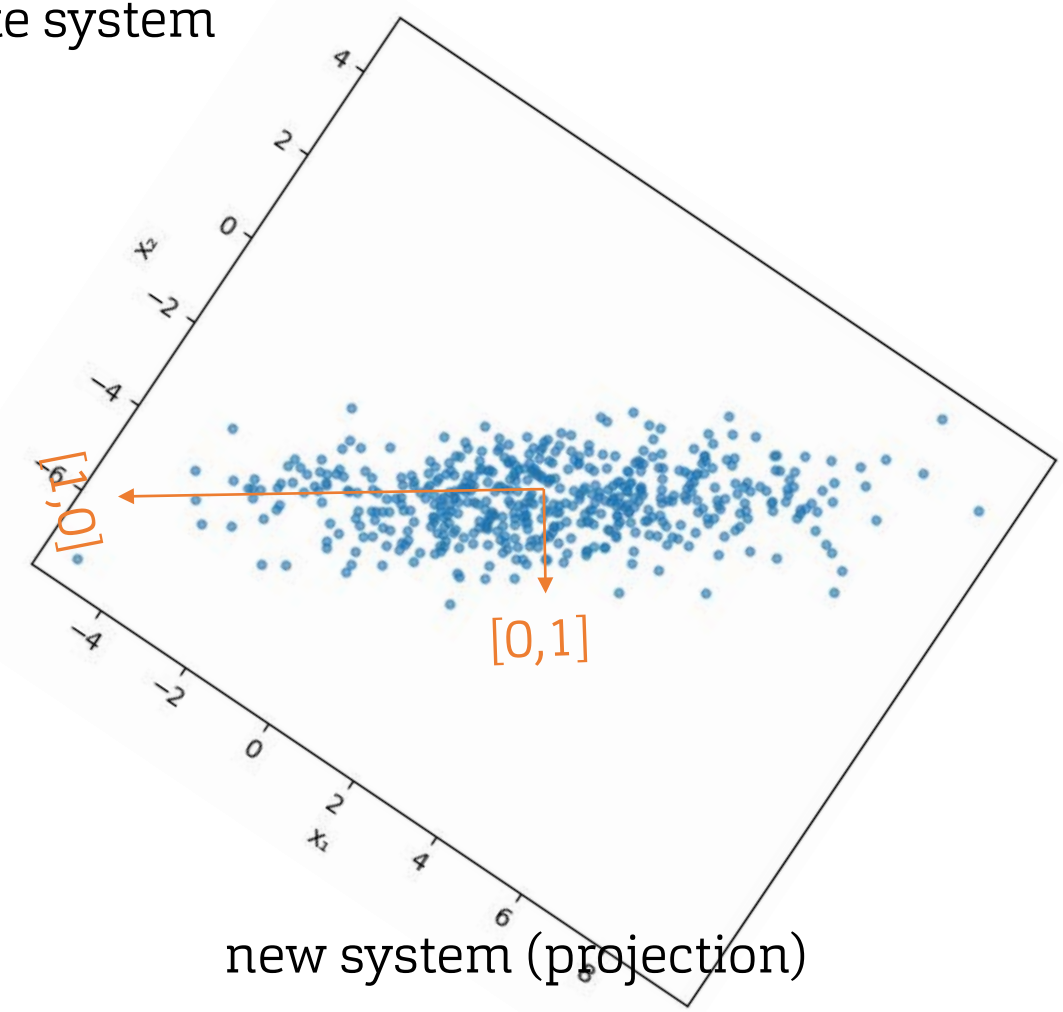
new system (projection)

A projection

- Comparing coordinates in old vs. new coordinate system



old system



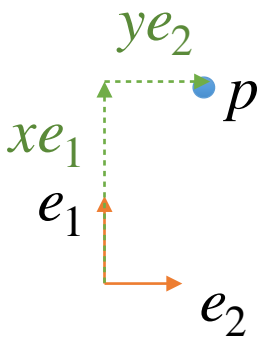
new system (projection)

Basis vectors

- Each coordinate system is defined by it's basis vectors.
- In Euclidean coordinate systems, the basis vectors are usually chosen to be orthogonal and of unit length, forming what is called an orthonormal basis.
- In a standard 2D cartesian coordinate system, the basis vectors are:

  $\mathbf{e}_1 = (1,0)$ and $\mathbf{e}_2 = (0,1)$
- We can reach any point $\mathbf{p} = (x, y)$ in the 2D space by using a linear combination of basis vectors:

  $\mathbf{p} = x\mathbf{e}_1 + y\mathbf{e}_2$



Hochschule Flensburg
University of Applied Sciences

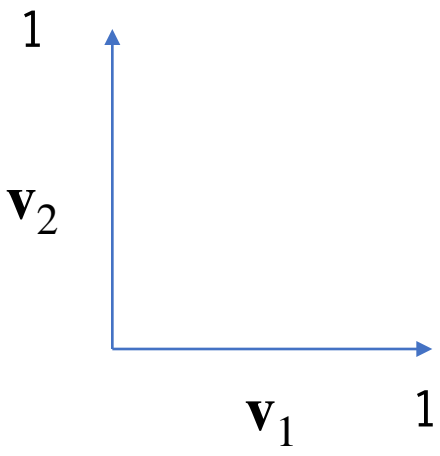
A little bit of math: Orthogonality

Hochschule Flensburg
University of Applied Sciences

- Let V be an Euclidean vector space.
- A set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ from V is called orthogonal, if

$$\langle v_i, v_j \rangle = 0 \quad \text{for all } i \neq j, i, j \in \{1, \ldots, k\}$$

($\langle \cdot, \cdot \rangle$ is a scalar product / dot product)

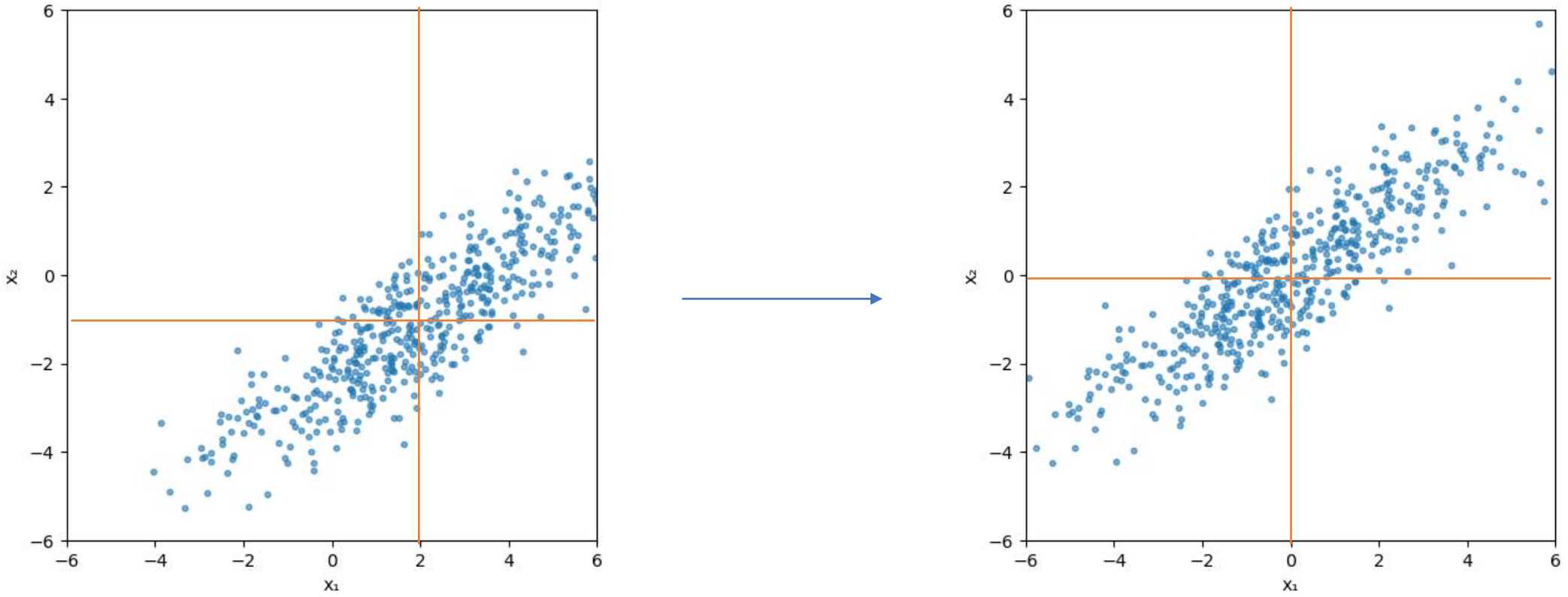


An example for an orthogonal vector space of dimension 2 with basis vectors:

$$\mathbf{v}_2 = [0,1], \mathbf{v}_1 = [1,0]$$

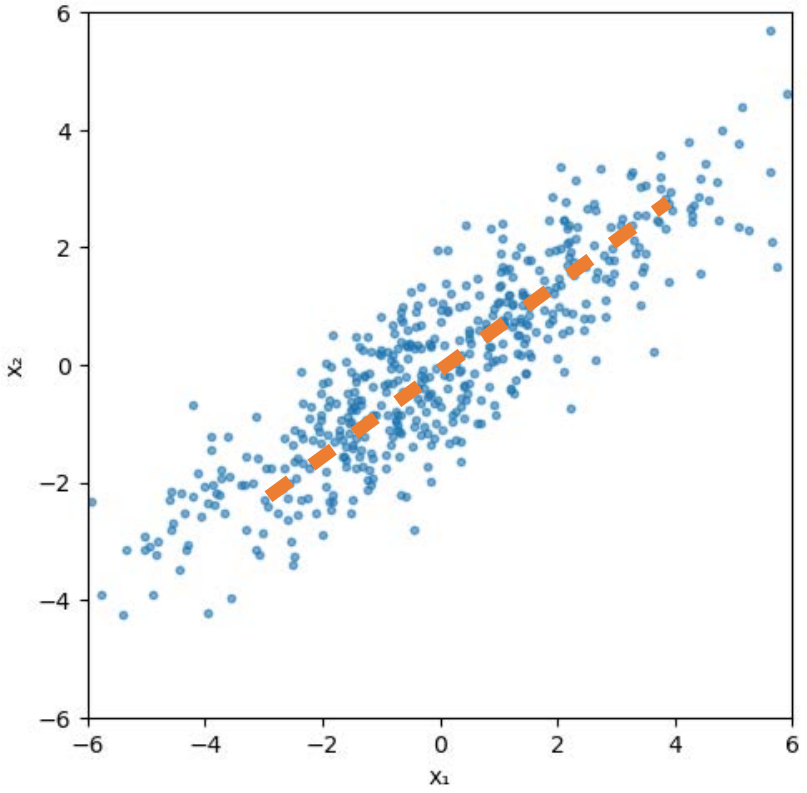
PCA: Preprocessing

- If we want to rotate everything around it's center, it's clever to move the distribution first by removing their sample mean values, so it is centered around a mean value of 0:



Dependency in variables

- In our original coordinate system, we can observe that the variables can be related to each other.
- As seen on the right side:
  - If we increase the x value, the y values seems to increase with it.
  - This type of relationship is captured mathematically by the covariance.
- In our rotated coordinate system, we want this to be no longer the case. This thus becomes one of our goals.

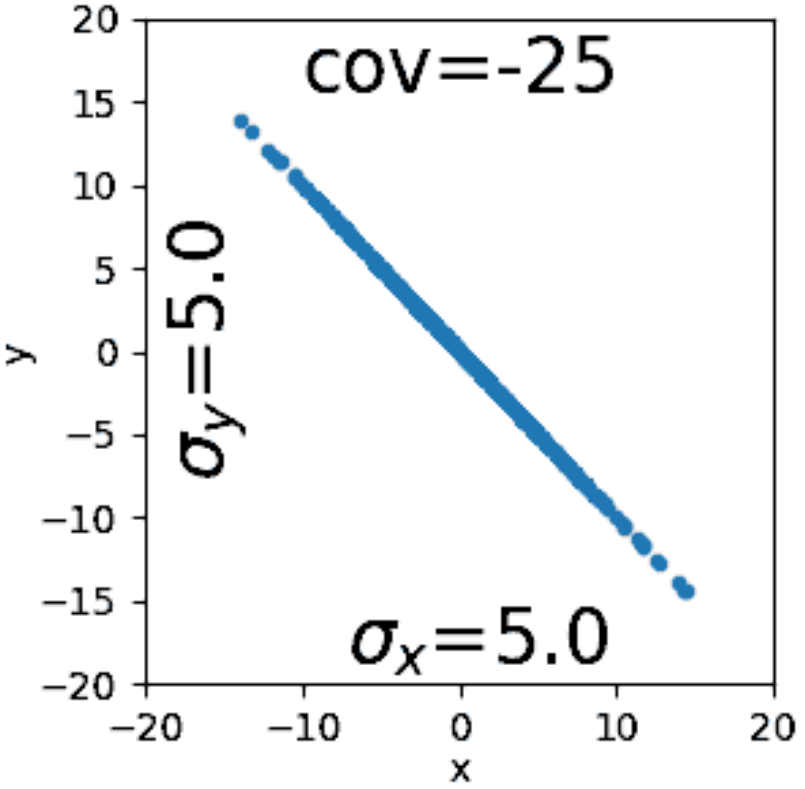


A Brief Excursus: What Is Covariance Again?

- Covariance tells us how strongly two variables X and Y are linearly related.

- Covariance is positive if the relationship between X and Y has the same trend.
  - High values of X often occur in conjunction with
  - high values of Y

- If the covariance is negative, the relationship has an opposite trend.
  - High values of X often occur in conjunction with
  - low values of Y



https://de.wikipedia.org/wiki/Kovarianz_(Stochastik)#/media/Datei:Varianz.gif

A brief excursus: What Is the link between covariance and correlation?

Hochschule Flensburg
University of Applied Sciences

- The covariance is proportional to the strength of the relationship and the amplitudes of both variances. The correlation is only proportional to the strength of the relationship.
- So: Covariance does increase with the amplitude of the variables, while correlation is normalized.

- The correlation (Pearson's correlation coefficient) is calculated from the covariance by normalization:

$$\rho(x,y) = \frac{Cov(x,y)}{\sqrt{Var(x)Var(y)}}$$

- This means that, regardless of the output of variables x and y, the degree of linear dependence is projected into the value range $[-1,1]$.
  - [-1..0) corresponds to an inverse relationship
  - (0..+1] corresponds to a positive relationship
  - 0 corresponds to uncorrelated variables

Covariance matrix

Hochschule Flensburg
University of Applied Sciences

The covariance matrix Σ_x of the variables $x_1, \ldots, x_m$ is defined as:

$$\Sigma_x = \begin{pmatrix} Var(x_1) & \cdots & Cov(x_1, x_m) \\ \vdots & \ddots & \vdots \\ Cov(x_m, x_1) & \cdots & Var(x_m) \end{pmatrix}$$

whereas

$$Cov(x_i, x_j) = \frac{1}{n-1} \sum_{k=1}^{n} \left(x_i^{(k)} - \bar{x}_i \right) \left(x_j^{(k)} - \bar{x}_j \right) = \frac{1}{n-1} \sum_{k=1}^{n} x_i^{(k)} x_j^{(k)}$$

and sample covariance

↑ for mean-free variables

$$Var(x_i) = \frac{1}{n-1} \sum_{k=1}^{n} \left(x_i^{(k)} - \bar{x}_i \right)^2 = \frac{1}{n-1} \sum_{k=1}^{n} x^{(k)} x^{(k)}$$

sample variance

↑ for mean-free variables

Example: Covariance matrix

Hochschule Flensburg
University of Applied Sciences

Sample data:

k	x_1	x_2	x_3
1	-0.5	-0.5	-0.75
2	-0.5	0.5	0.25
3	0.5	0.5	0.25
4	0.5	-0.5	0.25

x_1 and x_2 are uncorrelated.

x_1 and x_3 are correlated ($\rho = 0.577$)

$$Var(x_1) = \frac{1}{3}\sum_{k=1}^{4} x_1^{(k)} x_1^{(k)} = 0.33 \quad Var(x_2) = \frac{1}{3}\sum_{k=1}^{4} x_2^{(k)} x_2^{(k)} = 0.33 \quad Var(x_3) = \frac{1}{3}\sum_{k=1}^{4} x_3^{(k)} x_3^{(k)} = 0.25$$

$$Cov(x_1, x_2) = \frac{1}{3}\sum_{k=1}^{4} x_1^{(k)} x_2^{(k)} = 0 \quad Cov(x_1, x_3) = \frac{1}{3}\sum_{k=1}^{4} x_1^{(k)} x_3^{(k)} \approx 0.1667$$

Example of the covariance matrix

$$\Sigma_x = \frac{1}{n-1}\sum_{k=1}^{n} x^{(k)^T} x^{(k)}$$

k=1	0.25	0.25	0.375
	0.25	0.25	0.375
	0.375	0.375	0.5625
k=2	0.25	-0.25	-0.125
	-0.25	0.25	0.125
	-0.125	0.125	0.0625
k=3	0.25	0.25	0.125
	0.25	0.25	0.125
	0.125	0.125	0.0625
k=4	0.25	-0.25	0.125
	-0.25	0.25	-0.125
	0.125	-0.125	0.0625

k			
1	-0.5	-0.5	-0.75
2	-0.5	0.5	0.25
3	0.5	0.5	0.25
4	0.5	-0.5	0.25

Covariance matrix:

$\Sigma_x =$

0.33	0	0.1667
0	0.33	0.1667
0.1667	0.1667	0.25

$$\frac{1}{4-1}\left(\frac{1}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4}\right)=\frac{1}{3}$$

A brief recap of what we've learnt so far

Hochschule Flensburg
University of Applied Sciences

- For analyzing high-dimensional vector spaces, it makes sense to think about how to reduce the number of dimensions to a degree we can visualize (2D or 3D).
- One method for this is principal component analysis, where the target is to project the original data space into a new coordinate system, with the end goal of reducing the least important vector dimensions.
- Rotation of a coordinate system will not change it's total variance.
  - Individual variances alongside the individual basis vectors do change.
  - But the sum of all variances does not change.
- This is true for any kind of coordinate system projection (not only rotation).
- Covariance is expressing the tendency of data variables to co-vary with each other, i.e. change in a similar way.

Hochschule
Flensburg
University of
Applied Sciences

Principal Component Analysis

(now really)

PCA: How to calculate the PCs

- Let's start with stating what we know about the principal components:
  - We know that the PCs need to span a new coordinate system, where the first component points towards the direction of largest variance.
  - These vectors $\mathbf{u}_i$ act as the basis vectors of the new coordinate system. This means:
    - They need to be orthogonal, so:

      $$\mathbf{u}_i \cdot \mathbf{u}_j = 0 \qquad \forall i \neq j$$

    - and for orthonormality we want them to be of length 1:

      $$\|\mathbf{u}_i\| = \mathbf{u}_i^T \mathbf{u}_i = 1 \qquad \forall i$$

This is important, since we do not have any "fake" variance just by scaling it.

PCA: How to calculate the PCs

- Let's now continue with what we want:
  - We have an original coordinate system consisting of the variables $x_0, x_1, x_2, \ldots, x_n$
  - We want to express these in a new coordinate system.
  - We can thus state that for every variable x_i we want to be able to express it as linear combination of the new basis vectors:

    $\tilde{x}_0 = u_0 x_0 + u_1 x_1 + \ldots u_m x_m$
  - We also want these new variables to have no covariance:

    $Cov(\tilde{x}_i, \tilde{x}_j) = 0 \forall i \neq j$

$$\Sigma_{\tilde{x}} = \begin{pmatrix} Var(\tilde{x}_1) & & \\ & \ddots & \\ & & Var(\tilde{x}_m) \end{pmatrix}$$

Consequence: The covariance matrix becomes a diagonal matrix:

Calculation of the principal components u_i

- In the following we want to calculate the variance $Var(\tilde{x}_i)$
- We initially calculate the variance for the first principal component:

$$Var(\tilde{x}_1) = \frac{1}{n-1}\sum_{k=1}^{n} \tilde{x}_1^{(k)}\tilde{x}_1^{(k)}$$

substituting $\tilde{x}_1^{(k)} = x_1^{(k)} \cdot u_1$ yields:

$$Var(\tilde{x}_1) = \frac{1}{n-1}\sum_{k=1}^{n} \left(x_1^{(k)} \cdot u_1\right)^T \left(x_1^{(k)} \cdot u_1\right)$$

Since $(ab)^T = b^T a^T$, we can rearrange this to:

$$Var(\tilde{x}_1) = \frac{1}{n-1}\sum_{k=1}^{n} \left(u_1^T \cdot x_1^{(k)^T}\right) \left(x_1^{(k)} \cdot u_1\right) = u_1^T \frac{1}{n-1}\sum_{k=1}^{n} \left(x_1^{(k)^T} x_1^{(k)}\right) \cdot u_1 \quad = u_1^T \Sigma_x \cdot u_1$$

Maximizing the variance of the first PC

- We want to maximize the variance of the first PC:
  $\max \lambda_1 = \max u_1^T \Sigma_x \cdot u_1$
- Under the side condition that $\|u_1\| = 1$.
- This can be done using a Lagrange function:
  - Combine the maximization of the original function λ_1 with the side condition:
    $\mathscr{L}(u, \lambda) = u_1^T \Sigma_x u_1 - \lambda \left(\|u_1\|^2 - 1 \right)$
  - This is like a loss function - we want to optimize both at the same time.
- We can calculate the derivative of this function w.r.t. u_1, since we want to maximize the Langrange function by changing u_1.

Maximizing the variance of the first PC (2)

Hochschule Flensburg
University of Applied Sciences

- Let's derive this:
  $\nabla_{u_1} \mathcal{L}(u, \lambda) = \nabla_{u_1} \left(u_1^T \Sigma_x u_1 - \lambda_1 \left(\|u_1\|^2 - 1 \right) \right)$
- This is two additive terms, which we can calculate independently:
  $\nabla_{u_1} \left(u_1^T \Sigma_x u_1 \right) = 2\Sigma_x u_1$ and
  $\nabla_{u_1} \left(-\lambda_1 \left(\|u_1\|^2 - 1 \right) \right) = \nabla_{u_1} \left(-\lambda_1 \left(u_1^T u_1 - 1 \right) \right) = -2\lambda_1 u_1$
- In sum this is:
  $2\Sigma_x u_1 - 2\lambda u_1$
- To maximize we set this to zero:
  $2\Sigma_x u_1 - 2\lambda_1 u_1 = 0 \quad \text{or} \quad \Sigma_x u_1 = \lambda_1 u_1$
- For this exact value of λ_1 and u_1 we reach a maximum of the variance.

An Eigenvalue problem

- Our long, complicated derivation resulted in this equation for the first principal component:

  $\Sigma_x u_1 = \lambda_1 u_1$

- Here:
  - u_1 is the (so far unknown) principal component, and hence pointing towards the direction of greatest variance
  - λ_1 is the (so far unknown) spread of the data in this direction (and hence the variance of the data in the direction of the 1st PC)
  - Σ_x is the known covariance matrix.
- So now we need to "only" find a vector u_1 for which the equation holds.
- This is an Eigenvalue problem.

Let's revisit this

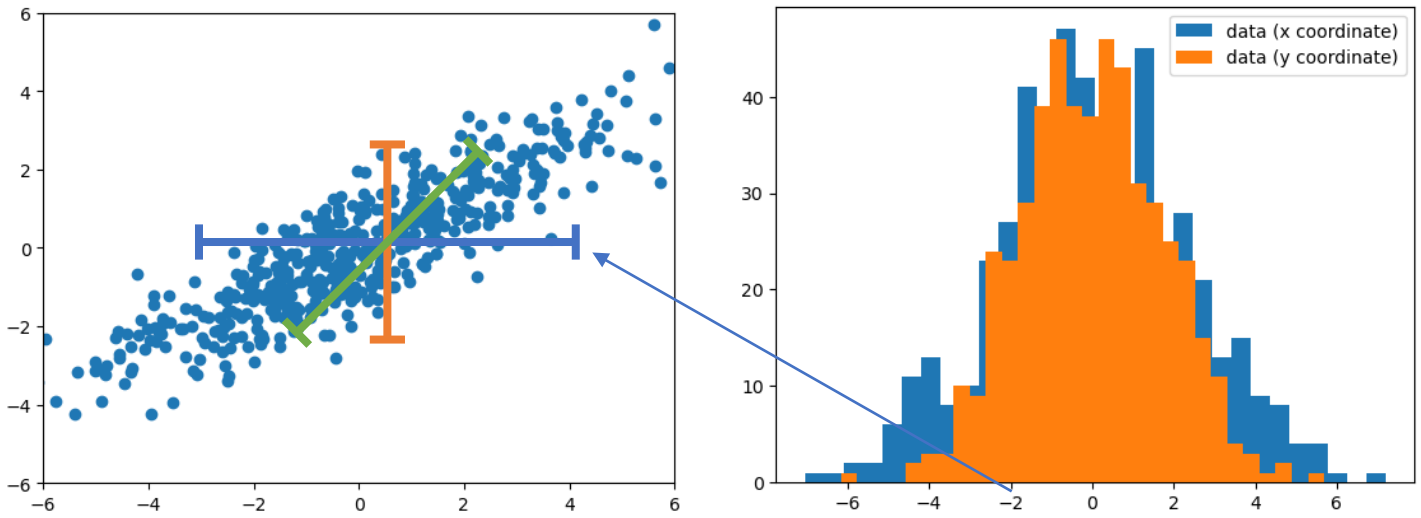
- Let's revisit our equation
  $\Sigma_x u_1 = \lambda_1 u_1$
- On the left side, we have the covariance matrix Σ_x, which tells us how the (original) data variables vary with the other.
- We multiply this by the new basis vector (the PC1) and get a projection of the spread into the direction of the new basis vector (PC1).
- This is also what we see on the right-hand side of the equation: We have the new basis vector, and it is scaled by a value - it's variance.
- The scaling factor we found here is thus just the variance of the data in the direction of PC1 - and this is what we wanted to maximize.

On eigenvalue problems

- Problems of the form
  $\mathbf{Ab} = \gamma\mathbf{b}$
  (where $\mathbf{A}$, $\mathbf{b}$ are vectors, γ is a scalar)
  are called eigenvalue problems.
- We say that $\mathbf{b}$ is an eigenvektor of the Matrix $\mathbf{A}$, and γ is its eigenvalue.
- We can solve eigenvalue problems using a number of well-established methods:
  - We can use a numeric solver (e.g., np.linalg.eigh)
  - We can use a singular value decomposition (SVD).

Hochschule Flensburg
University of Applied Sciences

Let's have a graphical look at this



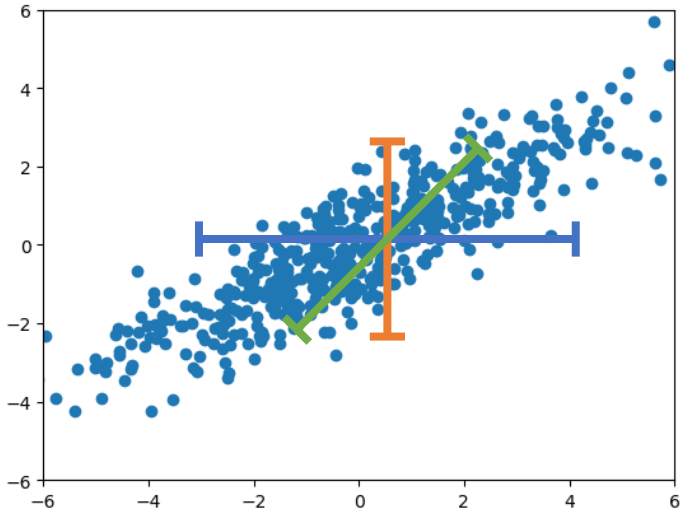
Covariance matrix:

5.5871361	3.603604
3.603604	3.000672

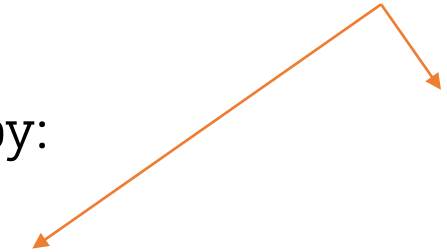
- This is the situation before the PCA.
- We have a covariance between the x and y axis.

Let's have a graphical look at this

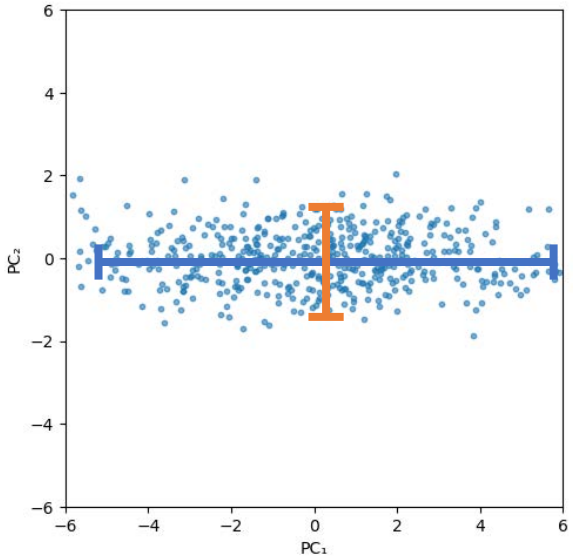
$$\Sigma_x u_1 = \lambda_1 u_1$$



multiply by:

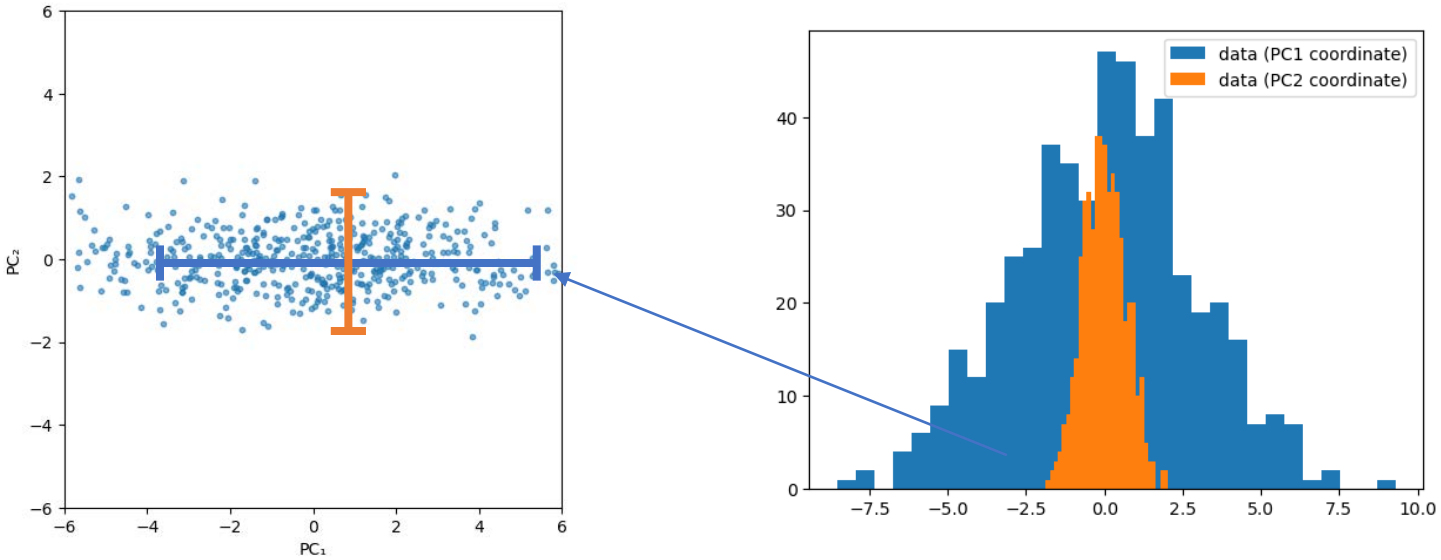


=



- Projecting the variances along the principal components effectively removes covariances.
- Projection onto the principal components u_i yields removes all components that are of the data that are orthogonal to u_i from it (and yields only the part pointing into the same direction).

Let's have a graphical look at this



Covariance matrix:

8.12253	0.000000
0.000000	0.465273

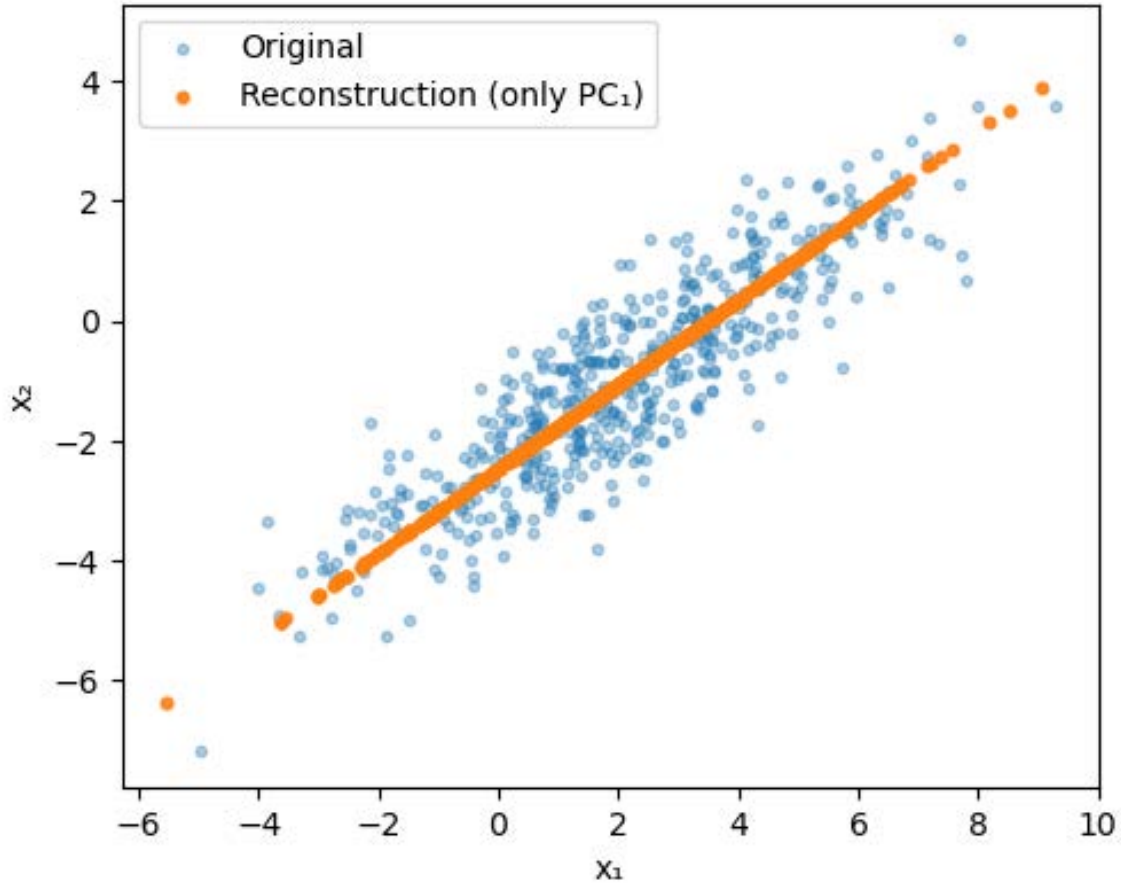
- This is the situation after the PCA.
- We have **no** covariance between the PC1 and PC2 axis.

The result

- If we remove only the second PC from the data and project it back into the cartesian (x,y) coordinate system, we see the effect of this.
- Of the original variance, most was kept, as we can see from the covariance matrix:

8.12253	0.000000
0.000000	~~0.465273~~

- Eigenvalues:
  0.94582165
  0.05417835



- Almost 95% of variance can be attributed to PC1. Removing PC2 only kills 5% of the information.

Summary: Principal Component Analysis

Hochschule Flensburg
University of Applied Sciences

- Principal component analysis (PCA) is a machine learning method for reducing the dimensions of data.
- The goal is to find the most compact representation of the data by reducing the variance of the data to those dimensions that contain the most information.
- PCA finds a new basis by identifying the directions with the highest variance in the data, which are called principal components.
- The principal components are sorted in order of their variance.
- The principal components represent an orthogonal (Cartesian) coordinate system.
- The number of principal components used to project the data can vary depending on the application. One measure of this is the proportional explanation of the total variance in the data by the selected principal components.

Applying PCA to latent spaces

Hochschule Flensburg
University of Applied Sciences

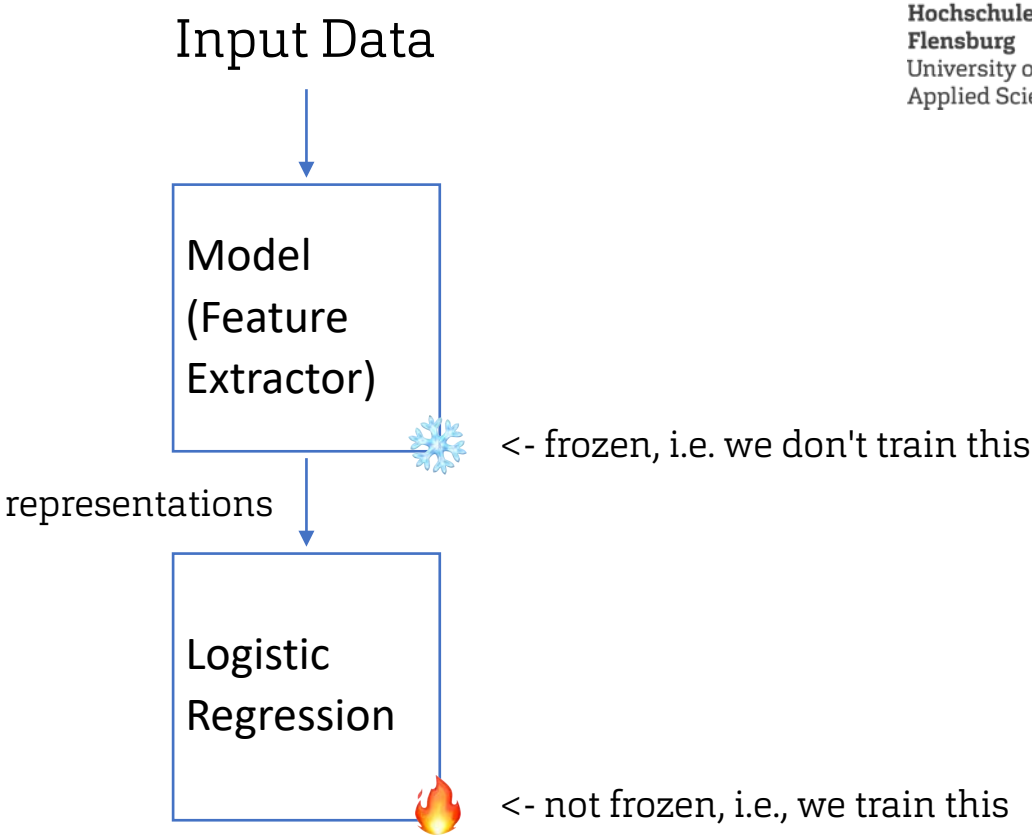
- We can run PCA on high-dimensional latent vectors (embeddings/activations) from a model layer (AE/VAE/encoder/CLS-token, etc.)
  - Visualize dominant data patterns
  - Dimensionality reduction for downstream tasks
  - Detect data structure and issues (Outliers & anomalies, domain shift, batch effects)
  - Interpretation (Inspect loadings: which original features contribute most to each PC
  - Use in pipelines
    - Preprocessing step before nonlinear methods (t-SNE, UMAP)
    - Latent space whitening (remove correlations)
    - Dimensionality reduction for visualization dashboards

Hochschule
Flensburg
University of
Applied Sciences

Linear Probing

What this does

- After (pre)training a model, we often want to know, if the information about a certain class of a classification problem is present in the model's representations.
- Linear probing is nothing but taking a model's representations and learning a logistic regression on top of them.
- This follows the idea:
  "Can a simple model solve this task using only the latent space?"



Interpreting Linear Probing Results

Outcome	Interpretation
High accuracy	Features are **linearly separable** → well-structured latent space
Low accuracy	Latent space lacks structure or doesn't capture task-relevant info
Gap to fine-tuning	Large gap suggests **non-linear transformations** are still needed

- We typically use linear probing for:
  - Model comparison
  - Layer-wise analysis
  - Representation diagnostics

Example for using linear probing

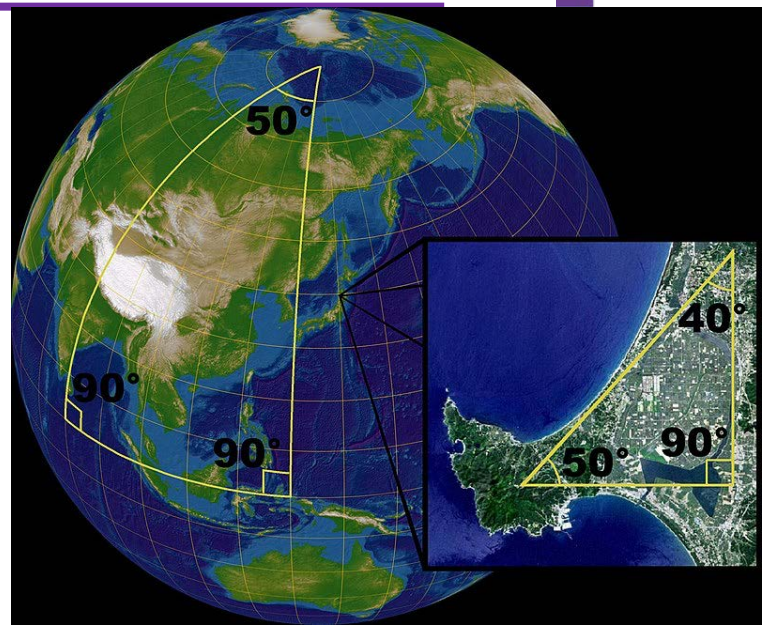
- Goal: Evaluate a self-supervised vision encoder (e.g. DINO)
- Steps:
  - Train encoder on unlabeled data
  - Freeze encoder
  - Extract representations for labeled dataset
  - Train logistic regression on top
  - Evaluate on downstream classification task
- Result: You learn how well the encoder "understood" the task without being trained for it

Hochschule Flensburg
University of Applied Sciences

Hochschule
Flensburg
University of
Applied Sciences

Manifold approximation

What's a manifold?



- A manifold is a mathematical object that has local properties of a flat plane but can be globally curved.
- An example of a manifold is the Earth's surface, which looks flat locally but is curved when viewed globally.
- The projection of the Earth's surface only exhibits the properties of a flat plane locally; when approaching the edges, distances appear distorted.
- In data analysis, manifolds can be used to model the intrinsic structures of data and visualize them in a lower dimension.

Linear and nonlinear methods

- The PCA we just discussed produces a linear projection from the original data space (e.g., the latent space) into a target space that can be used for interpretation.
- However, for further analysis (e.g., clustering), linear methods can sometimes be too restrictive, since they can only capture linear relationships.
- That's why we now turn to two methods from the family of nonlinear dimensionality reduction techniques:
  - t-SNE (t-Distributed Stochastic Neighbor Embedding)
  - UMAP (Uniform Manifold Approximation and Projection)
- Both methods go beyond PCA by modeling data as lying on a potentially nonlinear manifold, not just a rotated linear subspace.

Intro: Manifold Approximation

- Manifold approximation is a method for visualizing data in a lower dimension based on the concept of a manifold, a mathematical object that is locally flat but globally curved.
- UMAP (Uniform Manifold Approximation and Projection) and t-SNE (t-Distributed Stochastic Neighbor Embedding) are two of the best-known methods for manifold approximation.
- UMAP is a fast and scalable method that has improved clustering and classification performance and is less susceptible to local minima than other methods such as t-SNE.
- t-SNE, on the other hand, has the advantage of being highly sensitive to local structures in the data, enabling it to reveal complex patterns in the data.

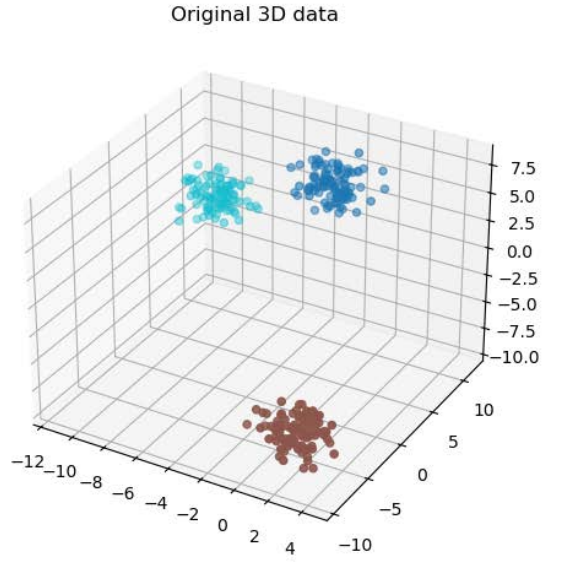
Core idea behind manifold approximation

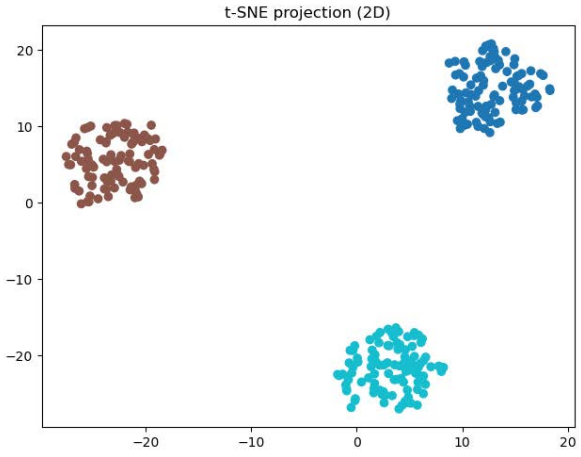
- We have a high dimensional space, and we want to map that to a low-dimensional space.
- However, we generally can't assume a linear relationship (else, we can use PCA for that)
- We know a metric that works in high- and low-dimensional spaces, but only locally
  - This means that we can determine, which samples are similar, because they are close to each other, but not, how dissimilar others are.

- So: How would we determine an overall structure that we can use for clustering or visualization?

t-stochastic Neighbor Embedding (t-SNE, Maaten und Hinton, 2008)

- t-SNE is characterized by high sensitivity to local structures in the data and can therefore reveal complex patterns in the data.
- It is a nonlinear transformation used to approximate the manifold and learn an embedding function.
- Disadvantages:
  - Poor scalability to large data sets.
  - Inability to model global structures in the data.





How t-stochastic neighbor embedding (t-SNE) works

Hochschule Flensburg
University of Applied Sciences

- The basic idea behind t-SNE is:
  If points are neighbors in the high dimensional space, they should be neighbors in the low-dimensional space.

- But: How do we determine "being neighbors"?

How t-stochastic neighbor embedding (t-SNE) works

Hochschule Flensburg
University of Applied Sciences

- First step:
  We define the probability of being a neighbor in the high dimensional space as a Gaussian distribution:

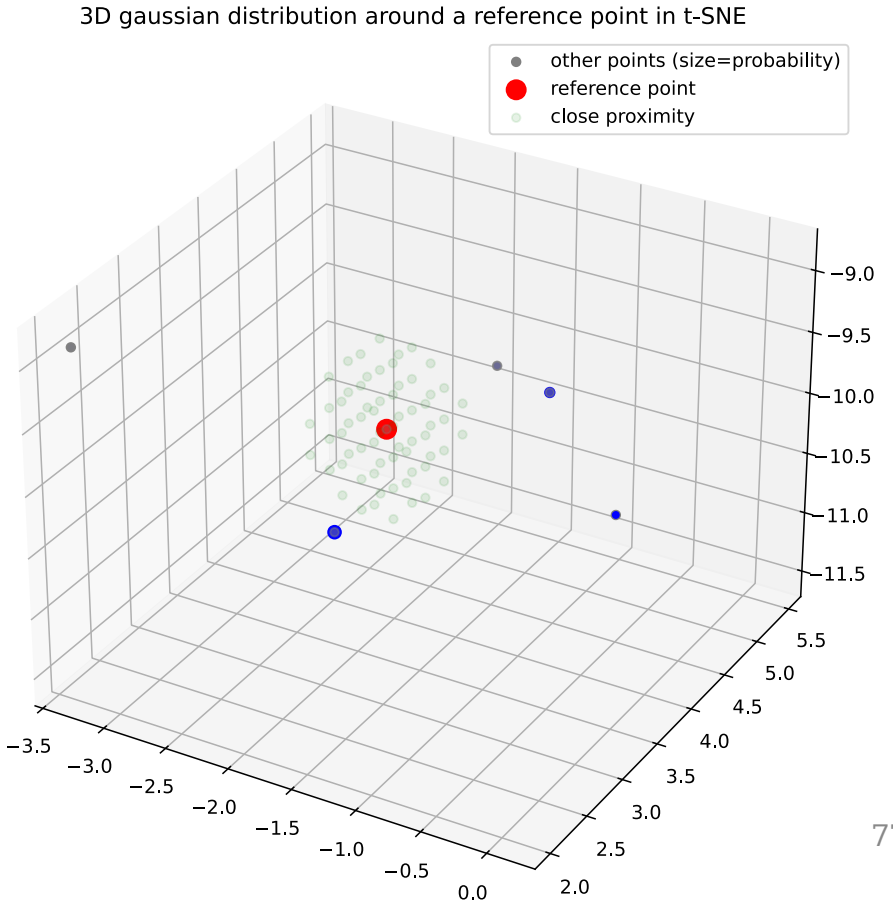
$$p_{j|i} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)}$$

whereas:

$p_{j|i}$: Probability of points j and i are being neighbors

x_i : Coordinates of point i

σ_i: Adaptive standard deviation

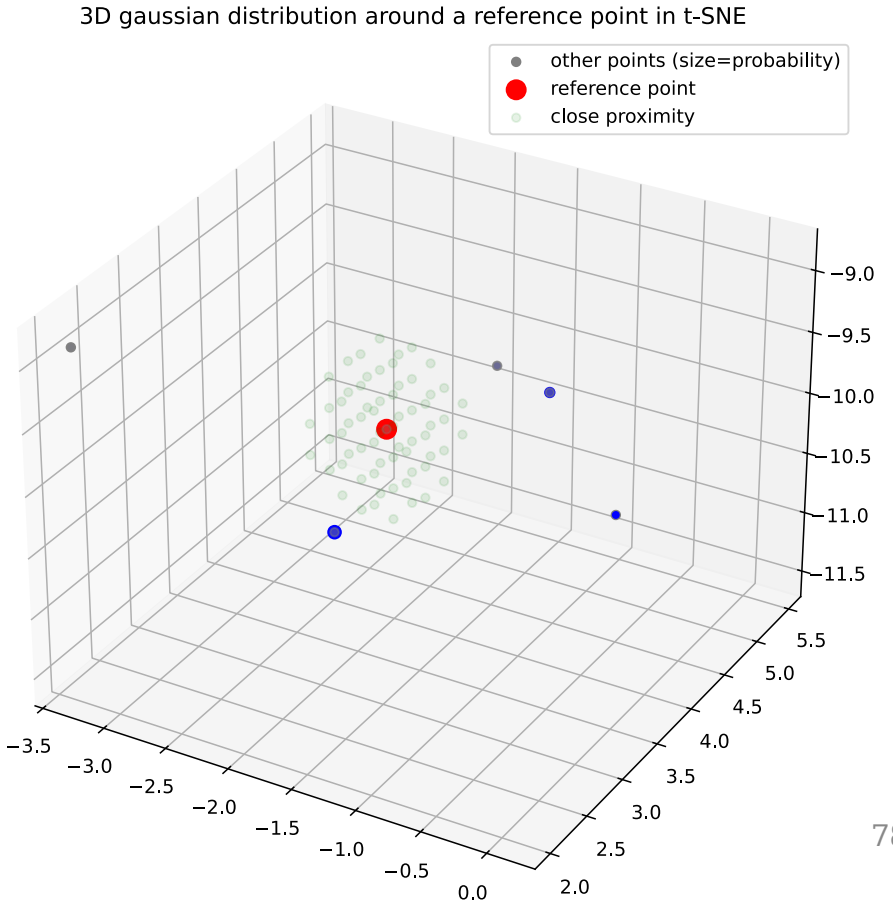


How t-stochastic neighbor embedding (t-SNE) works

Hochschule Flensburg
University of Applied Sciences

$$p_{j|i} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)}$$

- The probability of being a neighbor is dependent on the Euclidean distance and the standard deviation σ.
- So: How wide shall such a distribution be? (how large is σ_i?)
- In t-SNE, the value of σ_i is adaptively set, so that a certain perplexity (see next slide) is reached.
- The perplexity thus becomes the hyperparameter of t-SNE



Perplexity

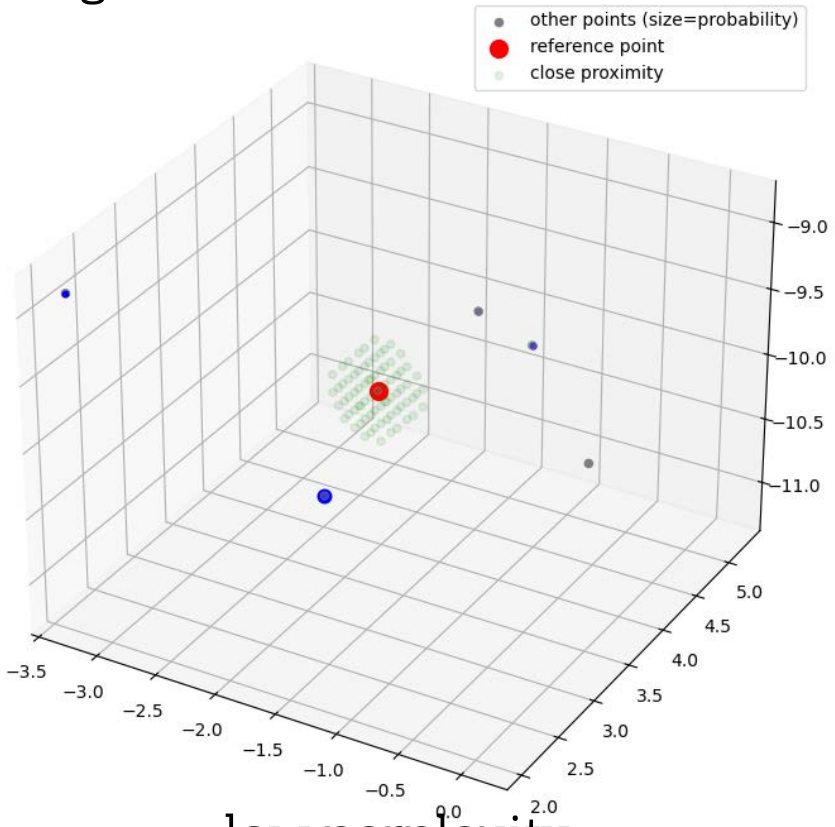
- Perplexity is a measure of uncertainty in machine learning.
- The perplexity in t-SNE controls how many "effective neighbors" each data point considers.
- For a data point x_i with conditional probabilities $p_{j|i}$:

$$\text{Perplexity}(P_i) = 2^{H(P_i)} \quad \text{where} \quad H(P_i) = -\sum_j p_{j|i} \log_2(p_{j|i})$$

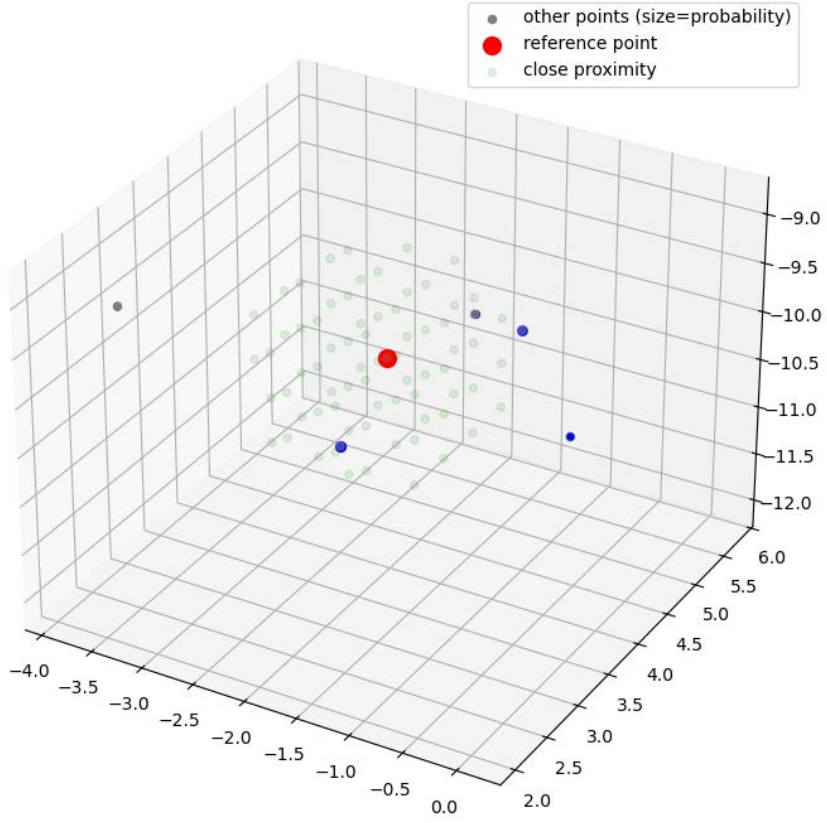
- The entropy $H(P_i)$ measures the uncertainty of the neighborhood distribution.
  - Low perplexity → very local neighborhood (tight Gaussian)
  - High perplexity → broader neighborhood (more global structure)

How t-stochastic neighbor embedding (t-SNE) works

- So by setting the perplexity, we set an overall value for how uncertain the probability distribution may be, i.e., how wide the gaussians are, and thus which points should be considered neighbors.



low perplexity



high perplexity

How t-stochastic neighbor embedding (t-SNE) works

Hochschule Flensburg
University of Applied Sciences

- Ok, so now we know how likely it is that two points in the high dimensional space are neighbors - but what do we make out of this?
- The step that is missing is:
  We need to find a 2D representation that has the same properties, meaning:
  - If point are highly likely to be neighbors in the high dimensional space
  - They need also be highly likely to be neighbors in the low dimensional space

- Note that we don't talk about points that are far apart from each other. We know **nothing** about their direct pairwise relationship.

Finding a low-dimensional representation

Hochschule Flensburg
University of Applied Sciences

- We want to find a low-dimensional representation of our data such that the **pairwise similarities** between points are preserved.
- Hence, we want to achieve that the conditional probabilities

  $p_{j|i}$ (in high-dimensional space) $\approx$ $q_{j|i}$ (in low-dimensional space)

  for all pairs (i, j).
- To measure how similar the high- and low-dimensional distributions are, we use the Kullback–Leibler (KL) divergence:

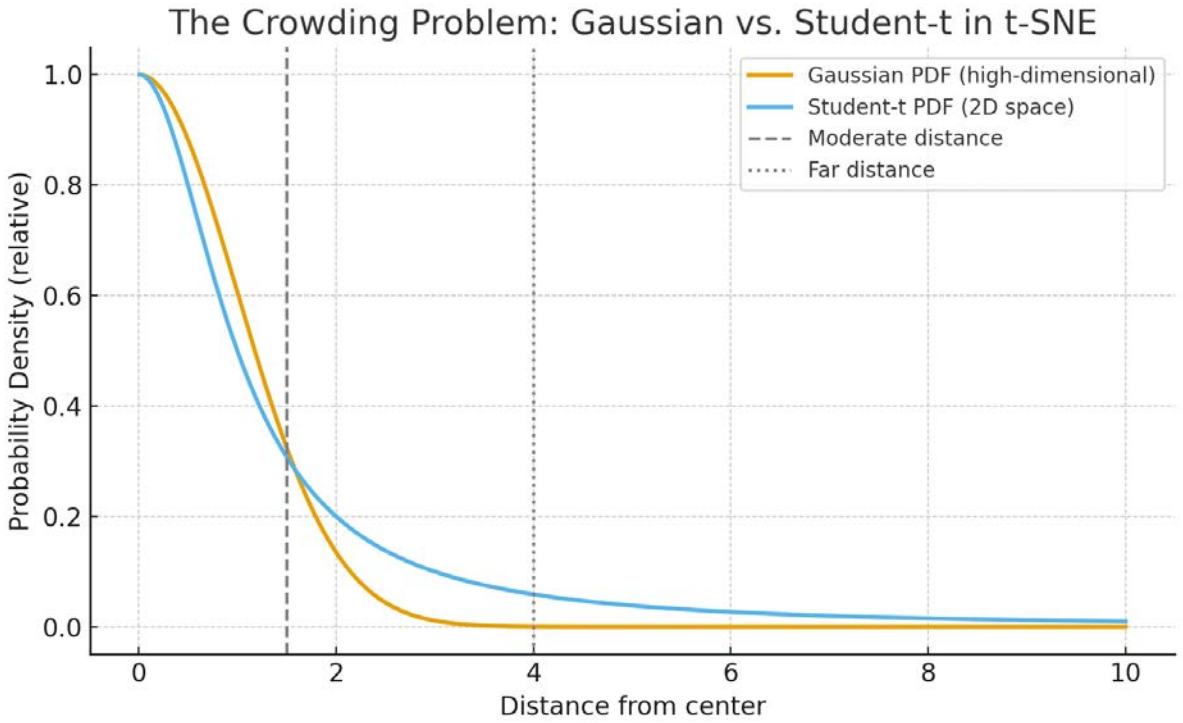
$$\mathrm{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

- The KL Divergence is 0 if the distributions are equal. We can thus minimize this.
- t-SNE uses stochastic gradient descent to minimize the loss function, and in this optimizes the coordinates of the points in the low-dimensional space.

Finding a low-dimensional representation

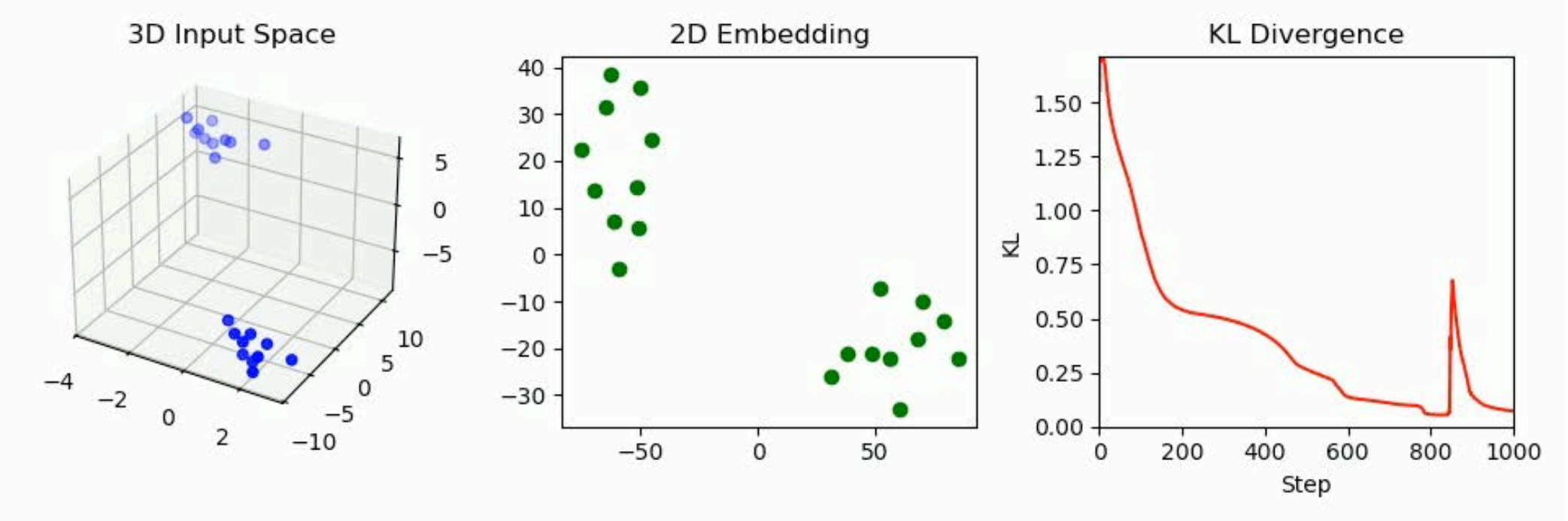
Hochschule Flensburg
University of Applied Sciences

- We now could go ahead and use the same Gaussian approach in the low dimensional.
- This is also what Hinton and Roweis originally proposed (Hinton and Roweis, Stochastic Neighbor Embedding, NIPS 2002).
- However, Gaussians fall off pretty quickly.
- A fast-falling Gaussian in 2D forces many weakly-related points too close together, distorting the structure;
- This is why a t-Distribution is chosen instead.
- The heavy tails of the Student-t fix this by allowing distant points to still have low, but meaningful similarity.



t-SNE: Demo (3D to 2D)

- As you can see, t-SNE is a stochastic iterative process.



- Besides being relatively slow, this also means that it does not have deterministic outputs.

t-SNE: Wrapping it up

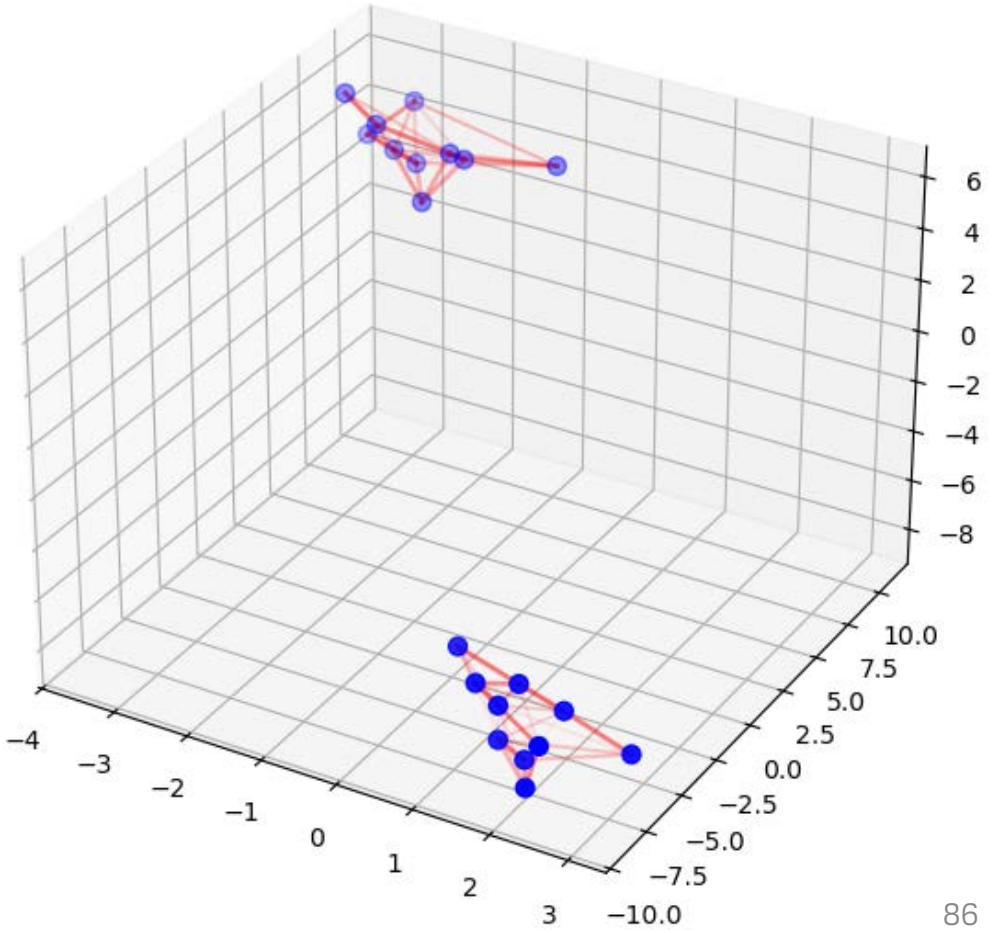
- t-stochastic neighbor embedding is a method to generate a low dimensional representation from a high dimensional distribution.
  - It uses a Gaussian distribution to model neighborhood in the high dimensional space
  - It uses a student t distribution to model neighborhood in the low dimensional space
- We can use this to understand the structure of the high dimensional data.
- If we have data labels, we can use t-SNE to get a better understanding of the data in high-dimensional latent vectors.
- But: t-SNE is a iterative, nondeterministic and relatively slow method for this.

Hochschule Flensburg
University of Applied Sciences

Uniform Manifold Approximation and Projection (UMAP, McInnes et al., 2018)

Hochschule Flensburg
University of Applied Sciences

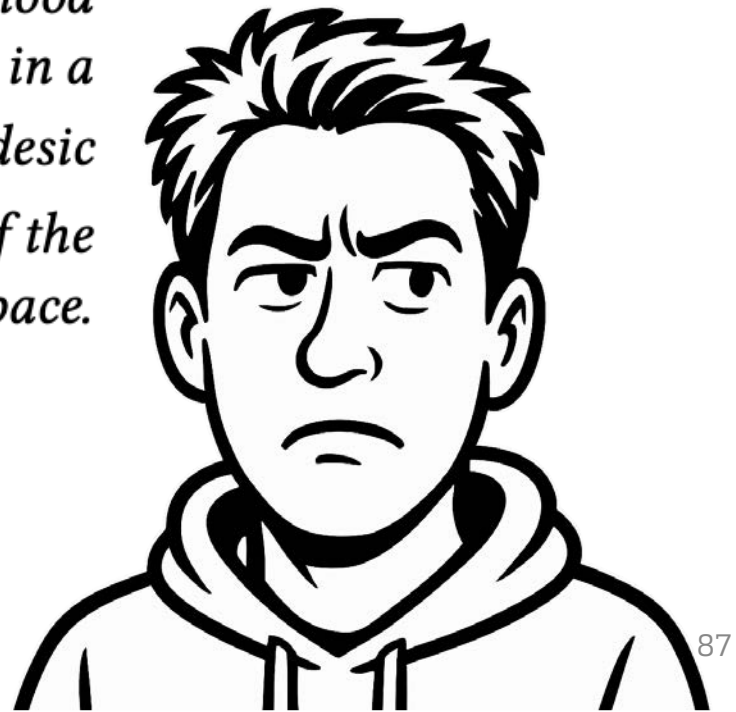
- Another and recently very popular method for creating low-dimensional representations from high-dimensional spaces is UMAP.
- While it appears similar to t-SNE at first sight, it is working very differently under the hood.
- Fundamentally, it does not try to create similar probability distributions like t-SNE, but works by using fuzzy graphs.
- In this graph, we have nodes and edges.
  - The nodes represent the points in the high-dimensional (or low-dimensional) space.
  - The edges represent their relationships. These edges have weights, which stand for how related the points are.



UMAP: Understanding the math behind it

- Unfortunately, the UMAP paper was written by a mathematician.
- While these are very smart people, they often express themselves in way we can't understand.

Lemma 1. *Let $(\mathcal{M}, g)$ be a Riemannian manifold in an ambient $\mathbb{R}^n$, and let $p \in M$ be a point. If g is locally constant about p in an open neighbourhood U such that g is a constant diagonal matrix in ambient coordinates, then in a ball $B \subseteq U$ centered at p with volume $\frac{\pi^{n/2}}{\Gamma(n/2+1)}$ with respect to g, the geodesic distance from p to any point $q \in B$ is $\frac{1}{r} d_{\mathbb{R}^n}(p, q)$, where r is the radius of the ball in the ambient space and $d_{\mathbb{R}^n}$ is the existing metric on the ambient space.*



UMAP: A computer/data science perspective

- Luckily, the authors knew that wrapping everything in proper math does not help us to understand it, so they provide a more comprehensible section as well:

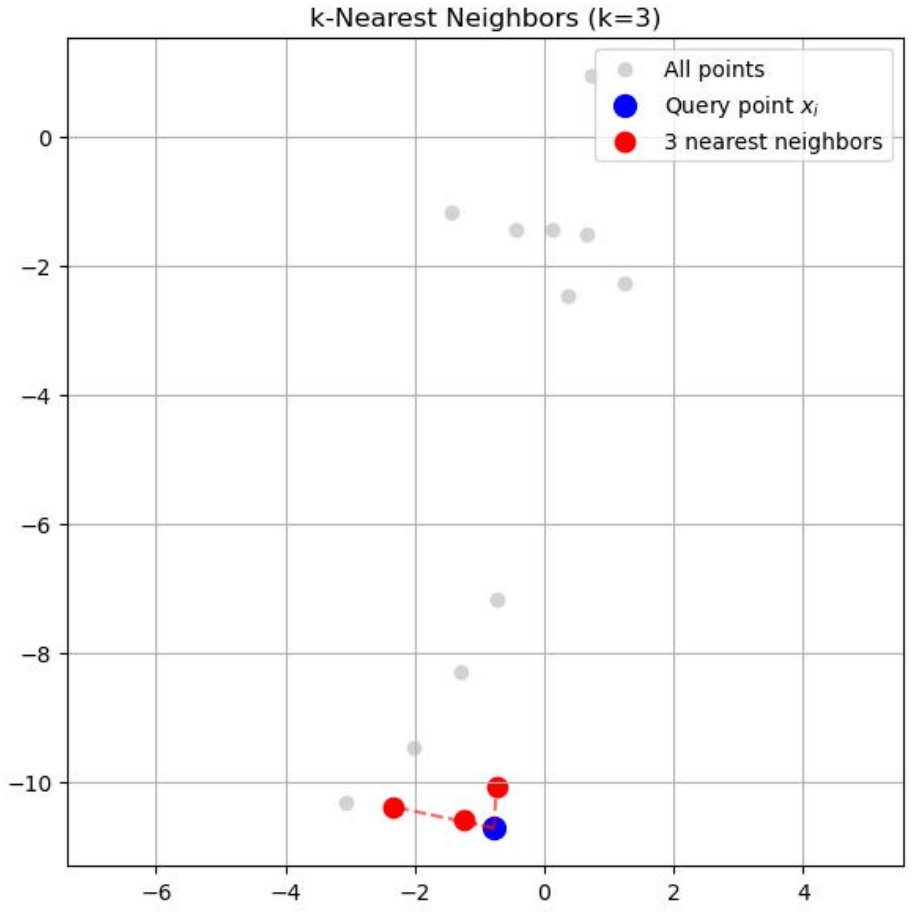
3 A Computational View of UMAP

To understand what computations the UMAP algorithm is actually making from a practical point of view, a less theoretical and more computational description may be helpful for the reader. This description of the algorithm lacks the motivation for a number of the choices made. For that motivation please see Section 2.

- Let's dig into this now.

k nearest Neighbors (kNN)

- kNN is an machine learning algorithm that is used in clustering as well as in supervised learning.
- The core idea is simple:
  - If we have a distance metric, we can find the k closest points to that point.
  - For classification, we can then look for the majority class of the k samples.
  - For regression problems, we can take the mean value of the k samples.
  - But we can also use it just to define the $(k)-$neighborhood.



Hochschule Flensburg
University of Applied Sciences

Two steps of UMAP

Hochschule Flensburg
University of Applied Sciences

- UMAP constructs two graphs:
  - In the first phase a particular weighted k-neighbour graph is constructed.
  - In the second phase a low dimensional layout of this graph is computed.

- The edges contain weights that reflect local distance. In particular, this is modeled by an exponential decay:

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\| - p_i}{\sigma_i}\right)$$

(where $\|x_i - x_j\|$ is the Euclidian distance between points (i, j), p_i is the distance to the closest neighbor , σ_i is chosen as a smoothed local normalization factor)

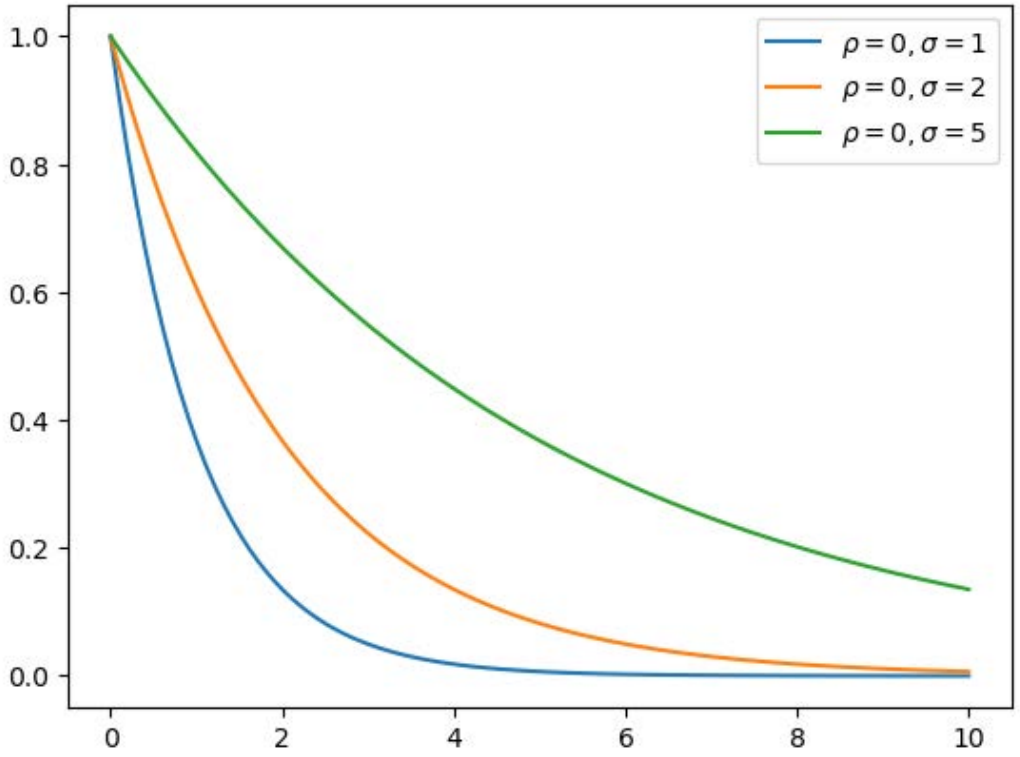
- This formula looks a bit complex, so let's dismantle it:

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\| - p_i}{\sigma_i}\right)$$

- This is just an exponential decay, with the input shifted and scaled.

* Actually, they are using a formula that looks a bit more complicated but is just ensuring numerical stability:

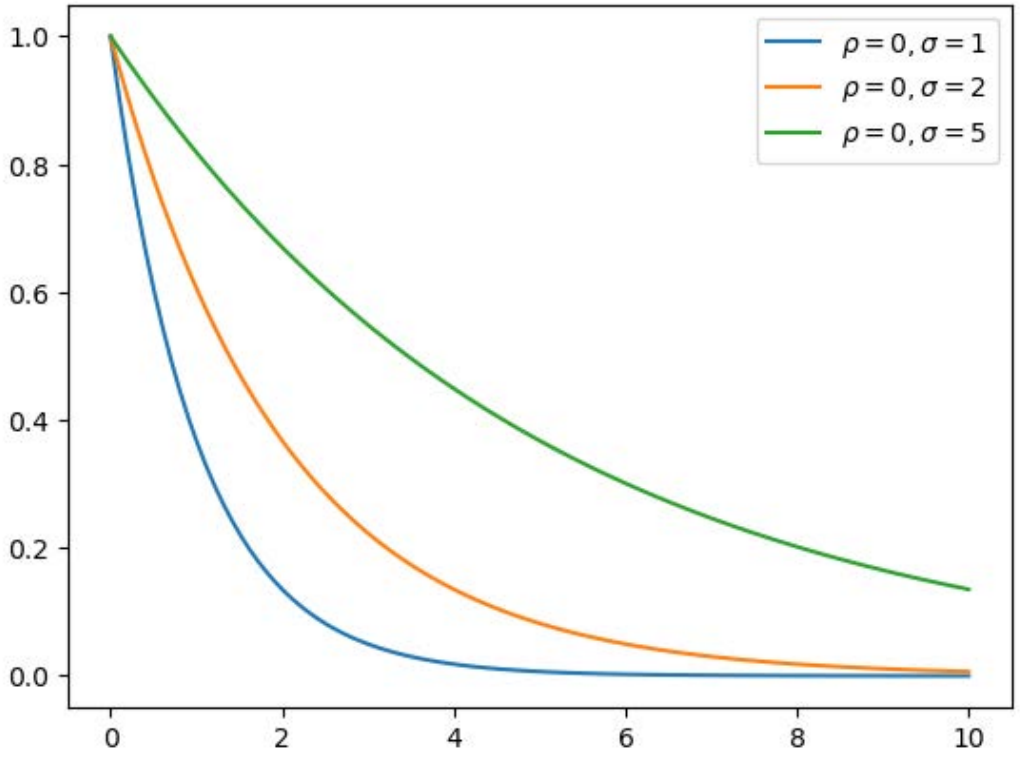
$$\exp\left(-\frac{\max(0, \|x_i - x_j\| - \rho_i)}{\sigma_i}\right)$$



How strongly should this decay?

- By setting the parameter σ_i, we define how strongly the distribution decays, i.e., how much the weight in the graph (which defines which point is close to another point) goes down with distance.

- If this parameter is set to a low value, have a strong decay, i.e., only the points in really close proximity are considered neighbors (have a high neighborhood-weight).



Hochschule Flensburg
University of Applied Sciences

How strongly do we want it to decay?

Hochschule Flensburg
University of Applied Sciences

- Let's now look at this question from another angle: The stronger the exponential decay goes down, the less points will be considered neighbors.
- Let's recap that we have a k nearest neighbor graph that we are working on.
  => For each point, we look at its k closest neighbors, but instead of treating them as binary connections, we assign each one a soft weight — how "strongly" it belongs to the neighborhood.
- If we want to look for a lot of neighbors
  - We need the weights to decay more slowly.
  - That means using a wider exponential kernel — i.e., larger σ_i.
  - This allows more of the k neighbors to receive non-negligible weight.

UMAP: The choice of σ_i

- In UMAP, the parameter σ_i is set for each point i such that

distance to point j

distance to closest point

$$\sum_{j=1}^{k} \exp\left(-\frac{\|x_i - x_j\| - \rho_i}{\sigma_i}\right) = \log_2 k$$

number of neighbors

- For the closest point, the difference in the numerator is 0. This results in a weight of 1.
- In dense regions, this leads to high weights, and in sparse regions to low weights.

UMAP: Construction of a low-dimensional graph

Hochschule Flensburg
University of Applied Sciences

- Now that we understand how this works in high dimensionality, we can construct a graph using the same methodology in low-dimensionality.
- We want the weights to be similar in the end (and can optimize the coordinates to match this)
- In low-dimensional space, UMAP uses another distribution, however (similar to t-SNE):
  - $w_{ij}^{(\text{low})} = \left(1 + a \cdot \|y_i - y_j\|^{2b}\right)^{-1}$ where a, b are hyperparameters
  - The hyperparameters a, b are being fitted from another hyperparameter: min_dist
  - This function also has heavy tails (much like the t distribution in t-SNE)
  - The idea behind it is very similar.

UMAP: Optimization

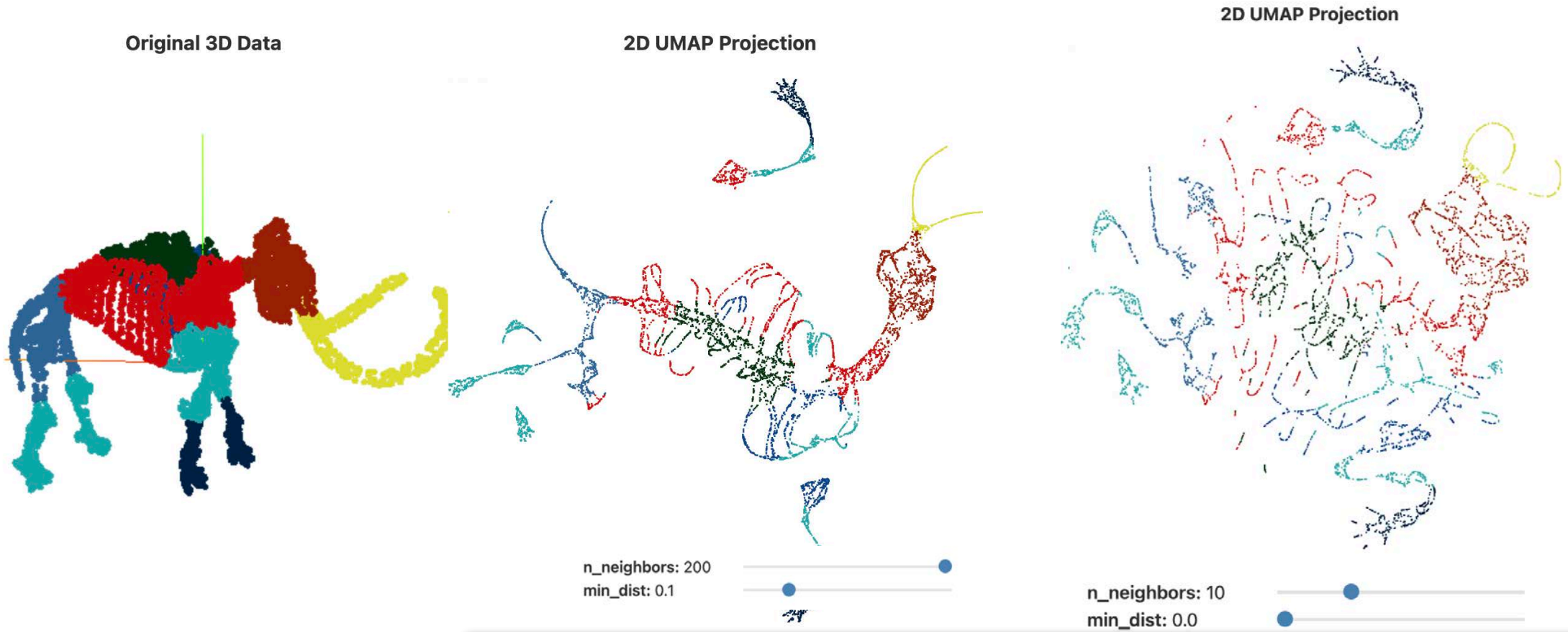
- Finally, we want to optimize the coordinates of the points (i, j) in the low-dimensional space such that the weights of the corresponding graphs are similar.

- This is done (again) using stochastic gradient descent, by optimizing this loss:

$$\mathscr{L} = \sum_{(i,j)} w_{ij} \cdot \log\left(\frac{w_{ij}}{w_{ij}^{(\text{low})} + \epsilon}\right) + (1 - w_{ij}) \cdot \log\left(\frac{1 - w_{ij}}{1 - w_{ij}^{(\text{low})} + \epsilon}\right)$$

- This means, however, that UMAP, much like t-SNE is not deterministic.
- It is, however, much faster, and can be parallelized better as it has fewer interdependencies.

Examples / hyperparameters



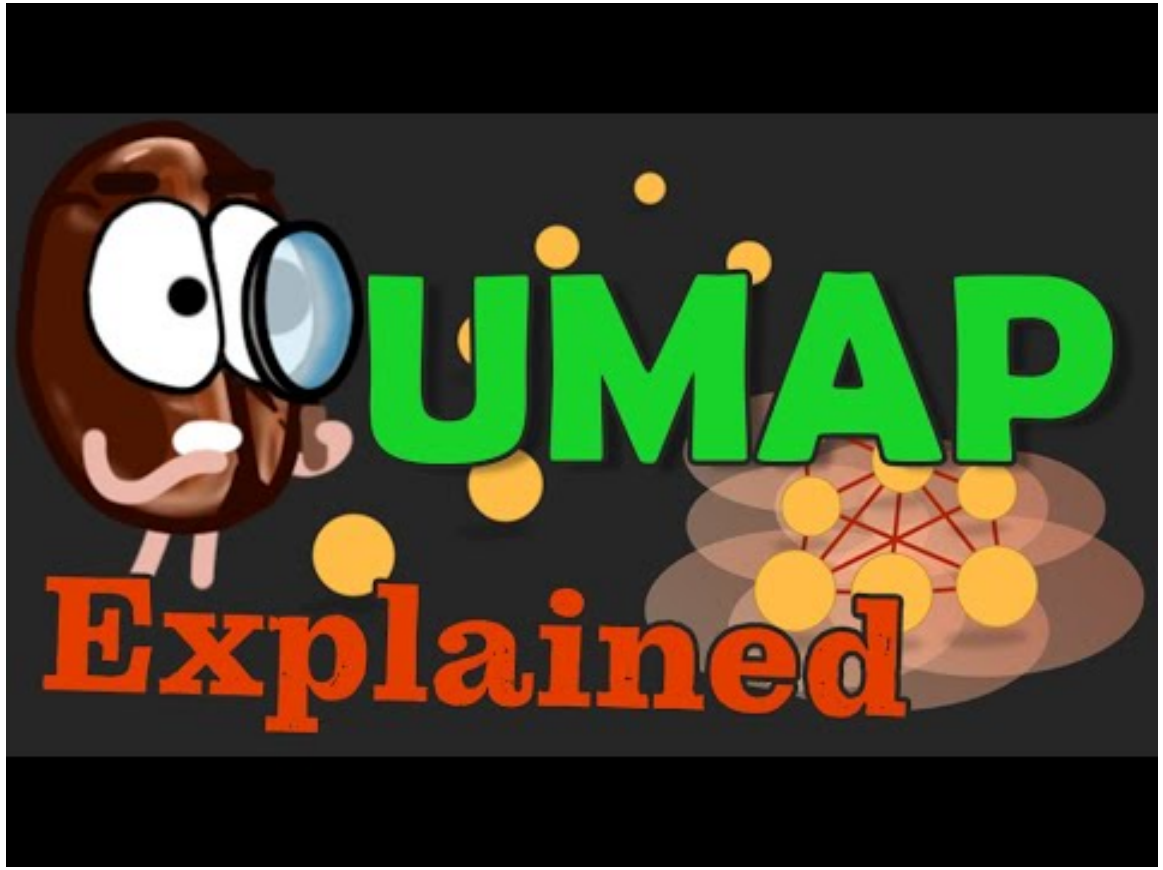
n_neighbors decides between better local and global structural coherence

Source: https://pair-code.github.io/understanding-umap/

How to read UMAP wrong

- UMAP is not a universal tool and requires careful interpretation of results.
- Choosing good hyperparameters is very important and depends on the data and objectives.
- Cluster sizes and distances between clusters in a UMAP diagram may be meaningless, as local distances are used to construct the graph.
- At low values of n_neighbors, incorrect clustering may be observed.
- UMAP is stochastic:
  - Different runs with the same hyperparameters may produce different results.
  - It may be useful to run the projection several times with different hyperparameters.

Recommendation



https://www.youtube.com/watch?v=6BPl81wGGP8

Wrapup: UMAP

- UMAP is a parallelizable (and thus relatively fast) method for manifold projection
- In contrast to t-SNE, UMAP works by using topological graphs.
- The "matching" between low dimensional and high dimensional spaces is done by matching the weights of the graphs (t-SNE: KL Divergence between probability distributions)
- It achieves a nonlinear projection, where local structure and global structure can be incorporated (depending on the hyperparameter).
- Due to its flexibility, versatility and speed, it is one of the most popular nonlinear projection methods today.

Hochschule
Flensburg
University of
Applied Sciences

Summary

Let's summarize this

Hochschule Flensburg
University of Applied Sciences

- In this unit, we've had a look at various techniques for machine learning.
- Of particular interest for generative AI are self-supervised learning techniques, as they allow to utilize huge unlabeled datasets.
- In this scenario in particular, we want to understand the organization of our model's latent space.
- We've had a look at four techniques for this:
  - Principal Component Analysis (PCA)
  - Linear Probing
  - t-stochastic neighbor embedding (t-SNE)
  - uniform manifold approximation and projection (UMAP)

Fill the gaps

Hochschule Flensburg
University of Applied Sciences

- Self-supervised learning creates labels from the _____ itself.
- The goal of SSL is to learn useful _____.
- A latent space is the output of a model layer before the final _____.
- Good latent representations should group _____ inputs together.
- If a linear classifier performs well, the latent space likely contains _____ features.
- UMAP constructs a graph using k-nearest neighbors and transforms it into a _____ graph.
- UMAP's low-dimensional layout is optimized using _____ on a cross-entropy loss between high- and low-dimensional graphs.
- In t-SNE, the _____ between the probabilities in both spaces is minimized.