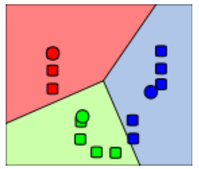
From (chemical) data to information

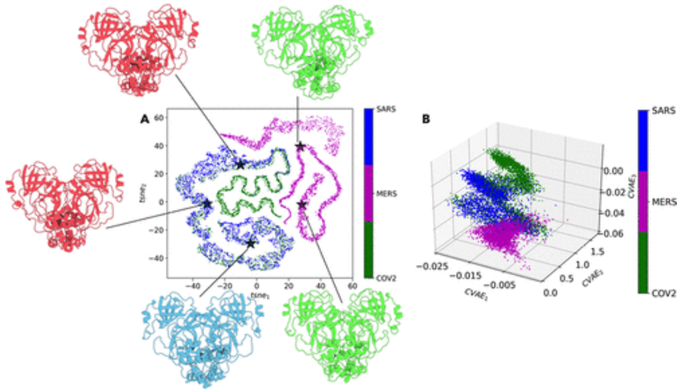
Introduction to Machine Learning Lecture 2

Dr Antonia (Toni) Mey

antonia.mey@ed.ac.uk

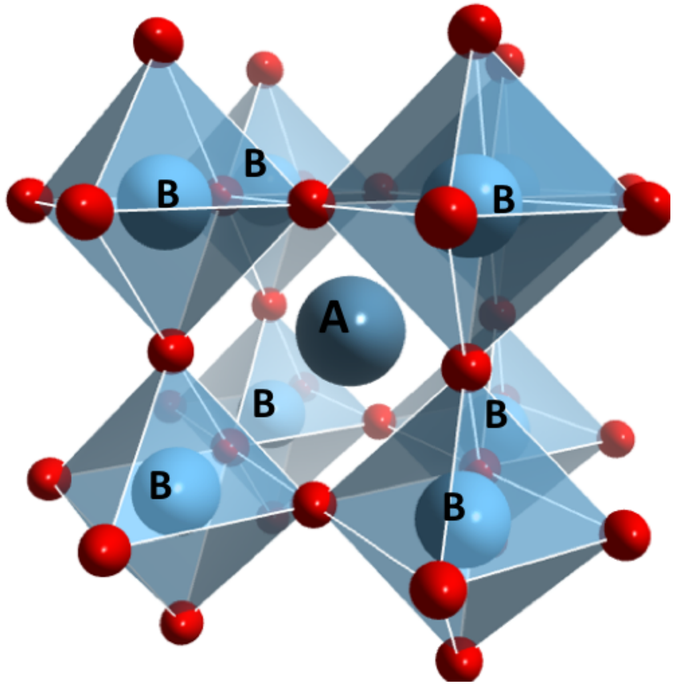
Room B23 JBB





Nature is not 2-dimensional

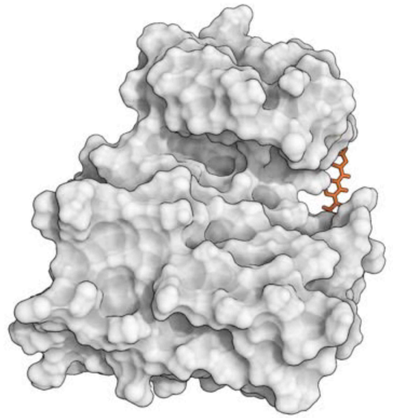
Inorganic Chemistry



Perovskites

https://chemicalstructure.net/portfolio/perovskite/

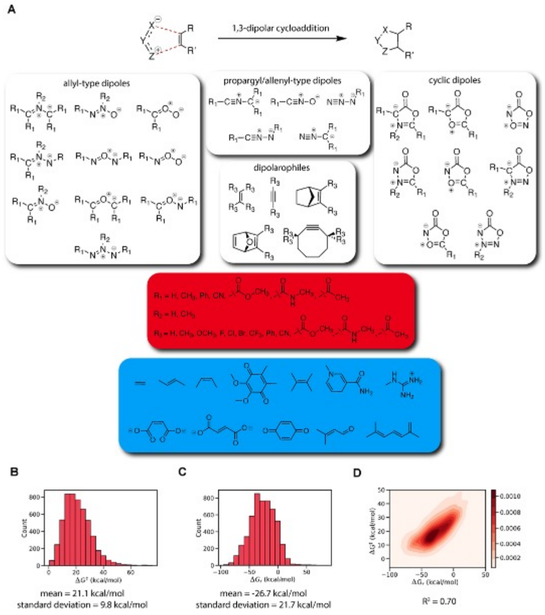
Thermodynamics



ΔG

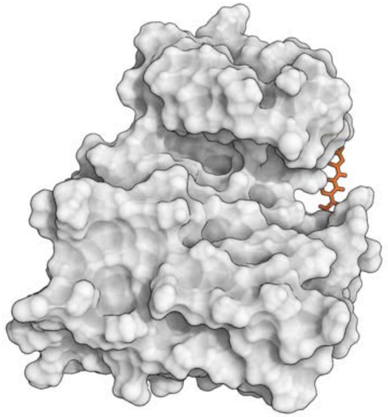
Shan, et al. *JACS* **133,** 9181 (2011)

Organic Chemistry

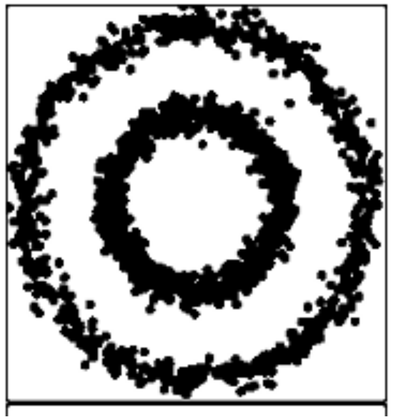


Stuyver, T., Coley, C. W., *Chem. Eur. J.* 2023, 29, e202300387

The many dimensions of chemistry

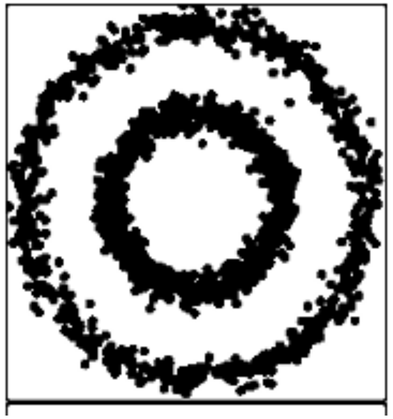


≠

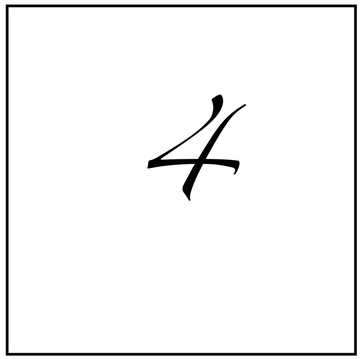


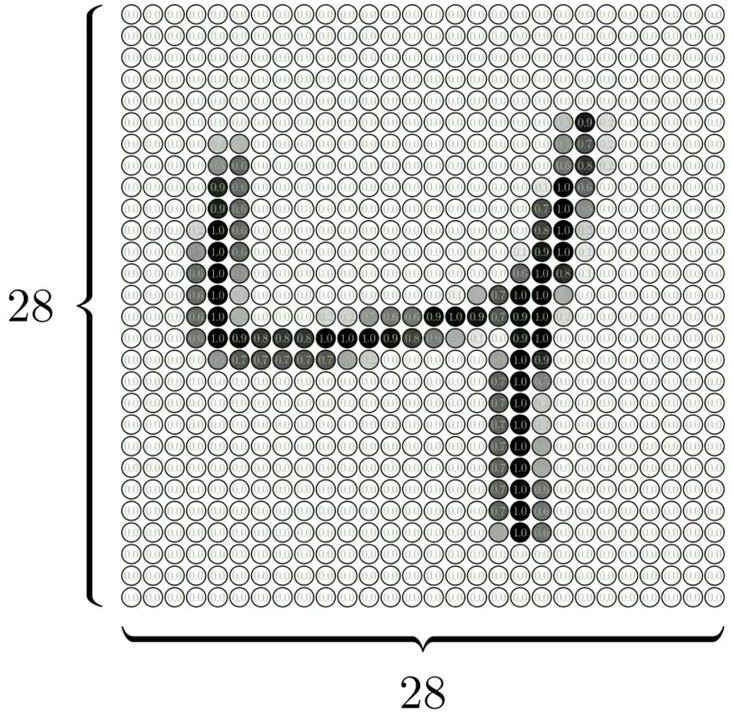
Looking at dimensions of an image

2D coordinates



N-D vector

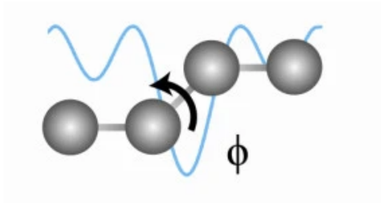


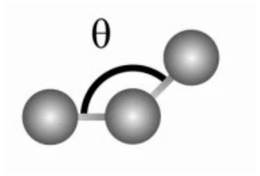


Features are possible representation of data as input

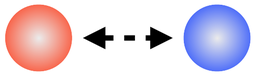
Features from coordinates

Charge





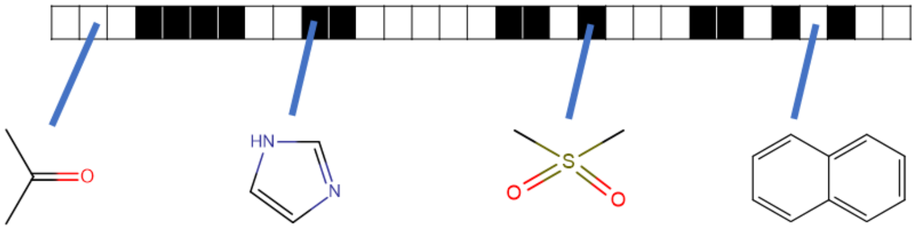
Angle

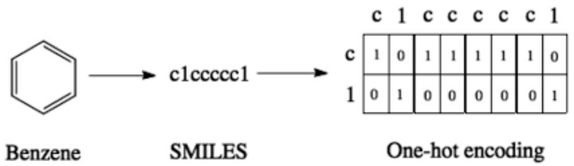


Distances

Bonds

Other features



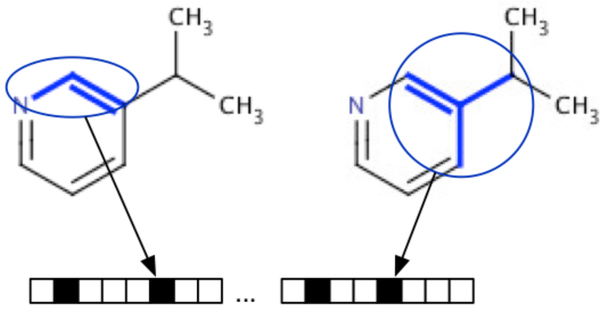


SMILES string

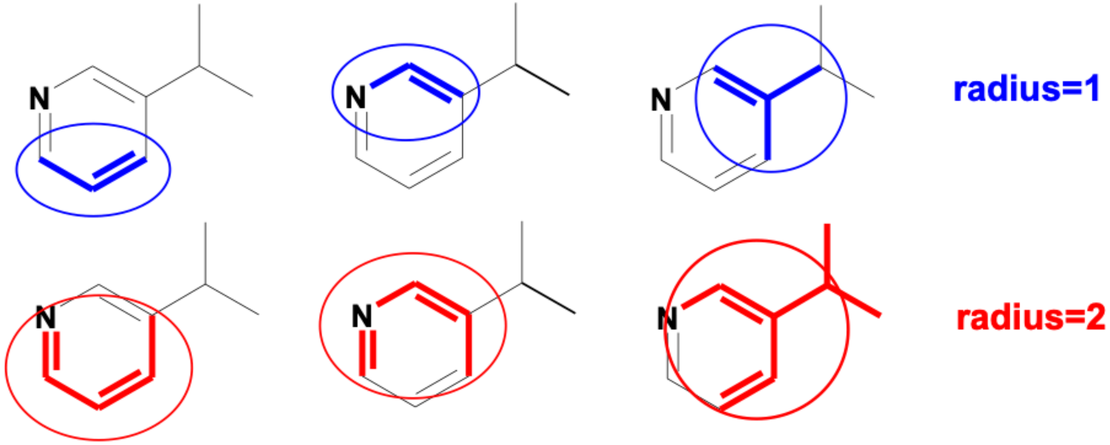
Feature vectors

```
['ATOM:ACE 1 CH3 1 x',
 'ATOM:ACE 1 CH3 1 y',
 'ATOM:ACE 1 CH3 1 z',
 'ATOM:ACE 1 C 4 x',
 'ATOM:ACE 1 C 4 y',
 'ATOM:ACE 1 C 4 z',
 'ATOM:ACE 1 O 5 x',
 'ATOM:ACE 1 O 5 y',
 'ATOM:ACE 1 O 5 z',
 'ATOM:ALA 2 N 6 x',
 'ATOM:ALA 2 N 6 y',
 'ATOM:ALA 2 N 6 z',
 'ATOM:ALA 2 CA 8 x',
 'ATOM:ALA 2 CA 8 y',
 'ATOM:ALA 2 CA 8 z',
 'ATOM:ALA 2 CB 10 x',
 'ATOM:ALA 2 CB 10 y',
 'ATOM:ALA 2 CB 10 z',
 'ATOM:ALA 2 C 14 x',
 'ATOM:ALA 2 C 14 y',
 'ATOM:ALA 2 C 14 z',
 'ATOM:ALA 2 O 15 x',
 'ATOM:ALA 2 O 15 y',
 'ATOM:ALA 2 O 15 z',
 'ATOM:NME 3 N 16 x',
 'ATOM:NME 3 N 16 y',
 'ATOM:NME 3 N 16 z',
 'ATOM:NME 3 C 18 x',
 'ATOM:NME 3 C 18 y',
 'ATOM:NME 3 C 18 z']
```

Molecular fingerprints encode molecule information

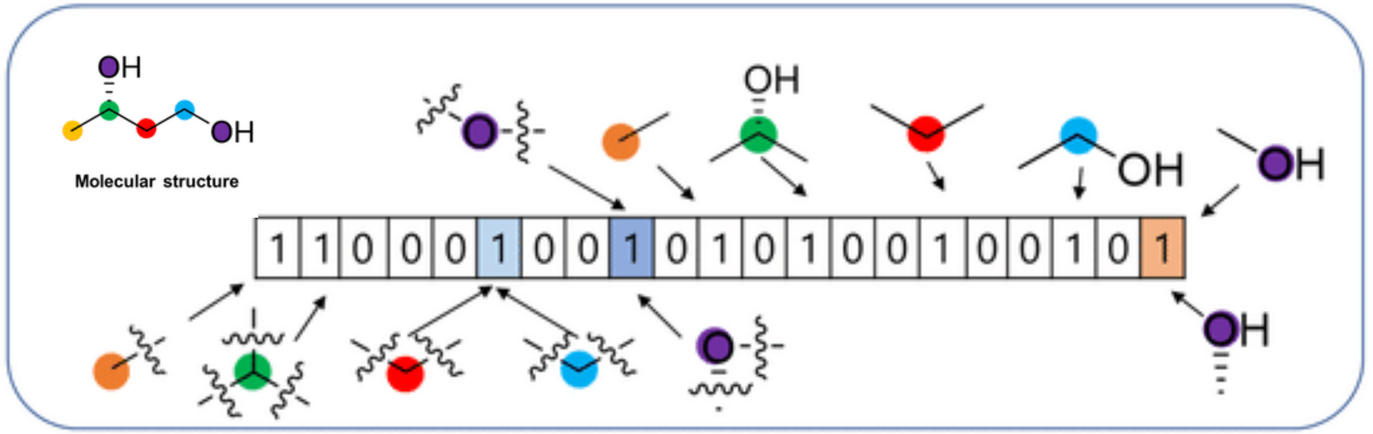


- Similar fingerprints mean similar molecules
- There are many different ways of defining a fingerprint



Atom types :

- Connectivity: (Element, #heavy neighbors, #Hs, charge, isotope, inRing)
- Chemical features: Donor, Acceptor, Aromatic, Halogen, Basic, Acidic
- Fingerprint takes into account the neighborhood of each atom (typical radii 0-3 bonds)

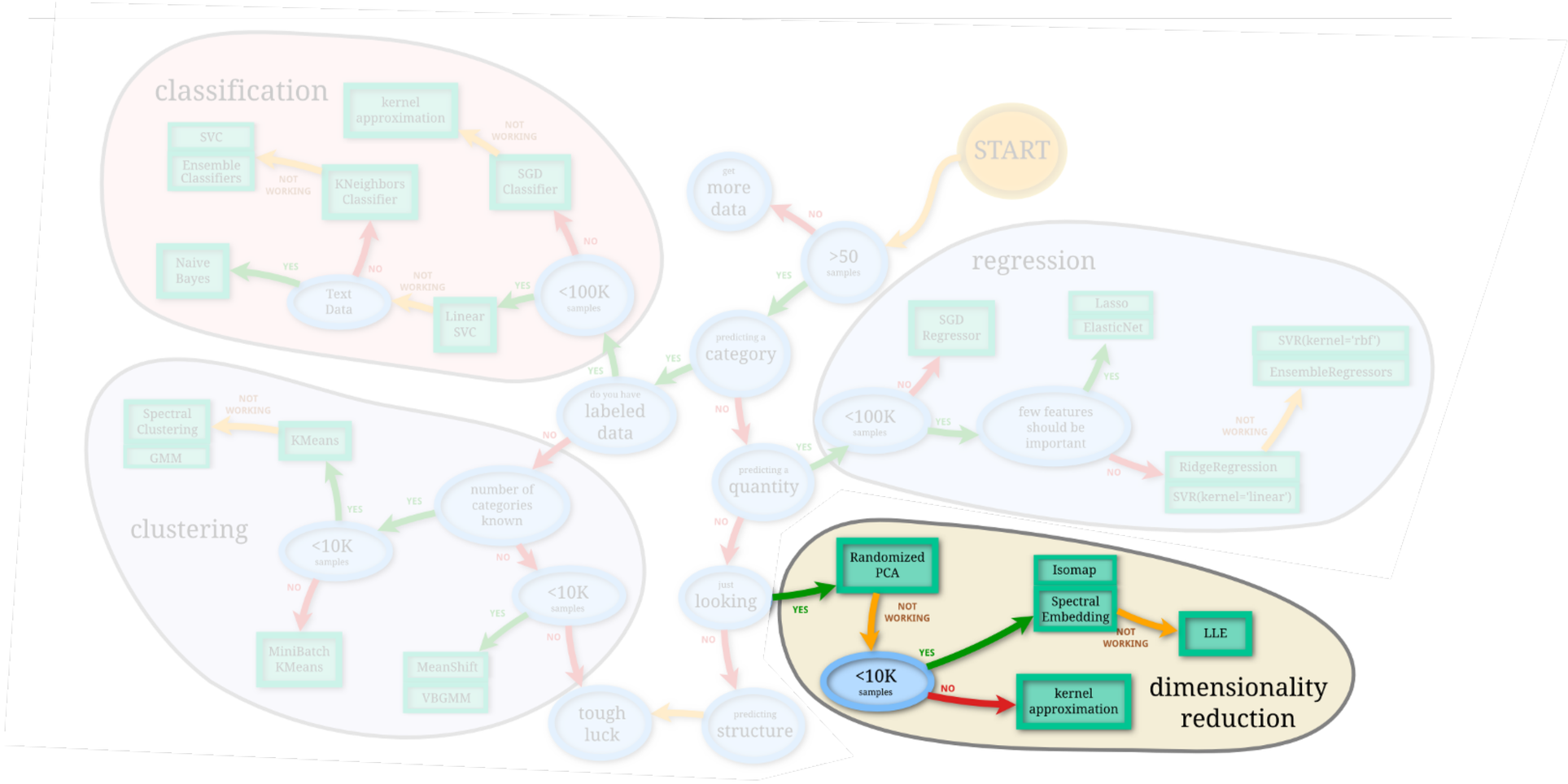


Original Morgan fingerprints

RDKit

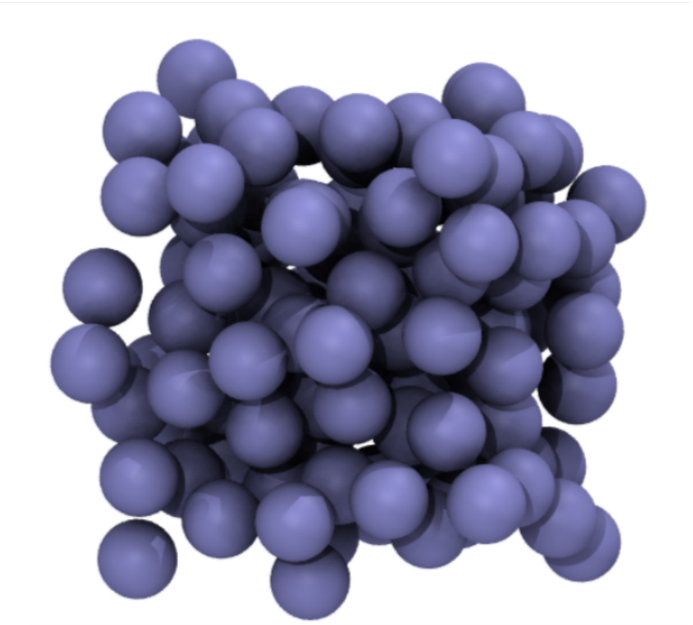
Open-Source Cheminformatics and Machine Learning

The Data Mining World — Dimensionality Reduction

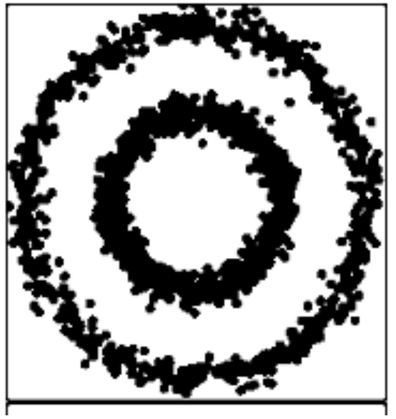


From **scikit-learn.org**

What are the ‘useful’ dimensions?



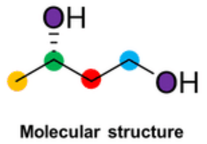
$\neq$



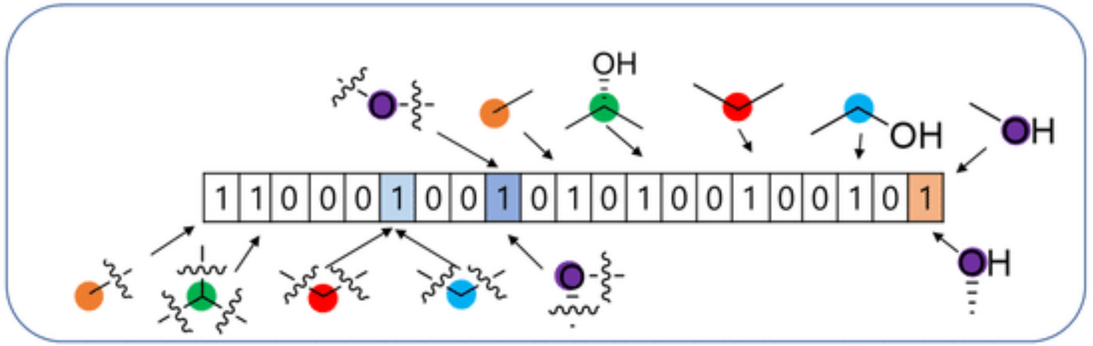
Dimensionality selection or reduction

Feature vectors

```
['ATOM:ACE 1 CH3 1 x',
 'ATOM:ACE 1 CH3 1 y',
 'ATOM:ACE 1 CH3 1 z',
 'ATOM:ACE 1 C 4 x',
 'ATOM:ACE 1 C 4 y',
 'ATOM:ACE 1 C 4 z',
 'ATOM:ACE 1 O 5 x',
 'ATOM:ACE 1 O 5 y',
 'ATOM:ACE 1 O 5 z',
 'ATOM:ALA 2 N 6 x',
 'ATOM:ALA 2 N 6 y',
 'ATOM:ALA 2 N 6 z',
 'ATOM:ALA 2 CA 8 x',
 'ATOM:ALA 2 CA 8 y',
 'ATOM:ALA 2 CA 8 z',
 'ATOM:ALA 2 CB 10 x',
 'ATOM:ALA 2 CB 10 y',
 'ATOM:ALA 2 CB 10 z',
 'ATOM:ALA 2 C 14 x',
 'ATOM:ALA 2 C 14 y',
 'ATOM:ALA 2 C 14 z',
 'ATOM:ALA 2 O 15 x',
 'ATOM:ALA 2 O 15 y',
 'ATOM:ALA 2 O 15 z',
 'ATOM:NME 3 N 16 x',
 'ATOM:NME 3 N 16 y',
 'ATOM:NME 3 N 16 z',
 'ATOM:NME 3 C 18 x',
 'ATOM:NME 3 C 18 y',
 'ATOM:NME 3 C 18 z']
```



Molecular structure



Original Morgan fingerprints

Learning low-dimensional representations / Dimensionality reduction

Aim: Capture as much information as possible with fewer dimensions.

Methods

- Linear: PCA
- Non-linear: *t*-SNE, UMAPs

- Visualization for exploring/analyzing data
- Removes noise/irrelevant features in data
- Resource efficiency

Not all features are useful

Task: predict the weather in Edinburgh using historical data

data = {X, Y, Z} → sun, rain, snow

{Temperature (C),
~~Temperature (K)~~,
Humidity (g/m^3)}

2 decorrelated features

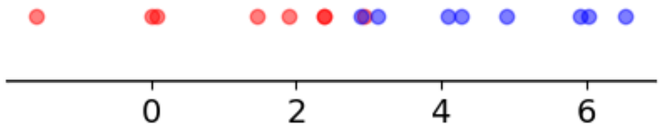
{Temperature (C),
~~Swiss cheese export (£)~~,
Humidity (g/m^3)}

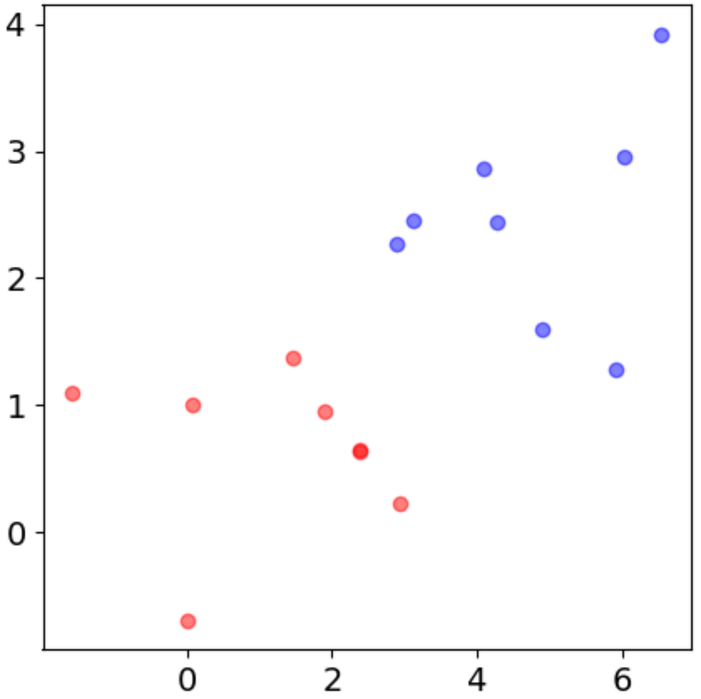
2 relevant features

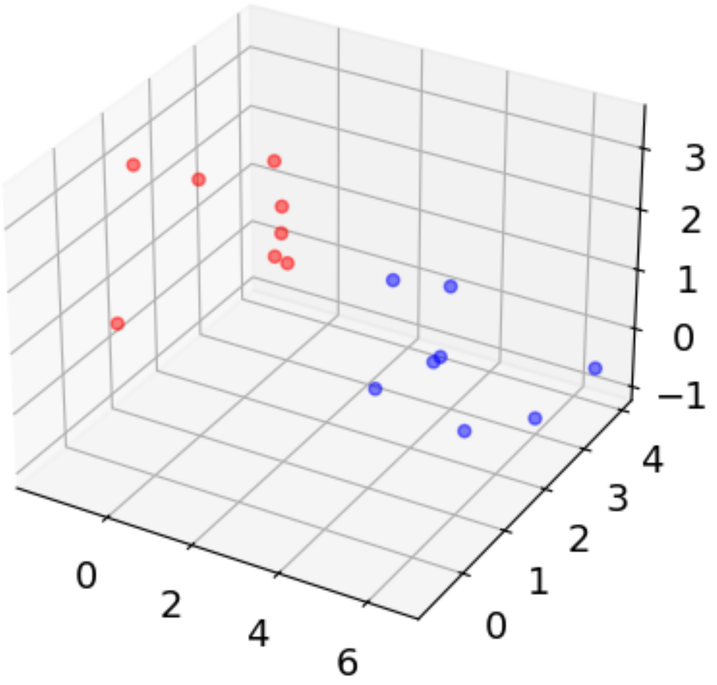
{Avg. gas expenditure (£),
Heat strokes (#),
Slipping accidents (#),
Sunscreen sold (£)}

4 features connected to another quantity temperature

Curse of dimensionality







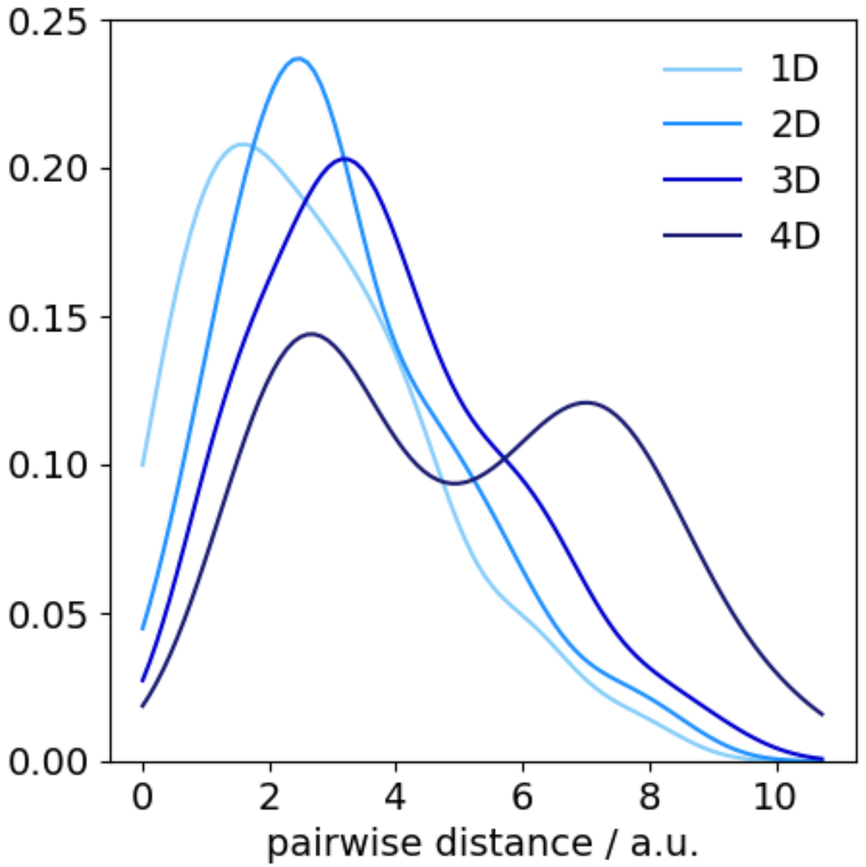
$$\|x\|_2 = \sqrt{\sum_{i=1}^{N} x_i^2}$$

Distances between *M* data points $\boldsymbol{x} \in \mathbb{R}^N$increase, when *N* increases

Problem: less data density increases uncertainty on underlying data structure

Curse of dimensionality

$$\|x\|_2 = \sqrt{\sum_{i=1}^{N} x_i^2}$$



Distribution of pairwise distances between points shown in previous slide

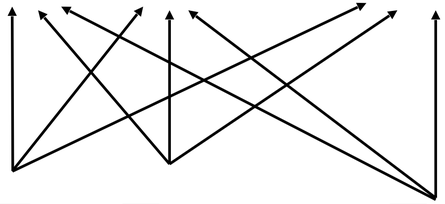
Distances between *M* data points $\boldsymbol{x} \in \mathbb{R}^N$increase, when *N* increases

Problem: less data density increases uncertainty on underlying data structure

Reducing features increases data density

- Chose appropriate features [expert user]
- Remove features
- Find a lower dimensional representation *E* of features *X*

$$\{ X_1 , X_2 , \ldots , X_N \}$$

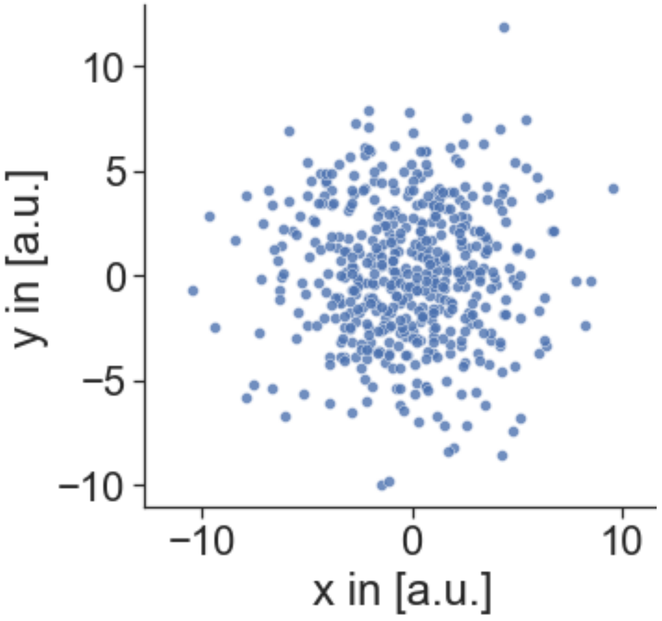


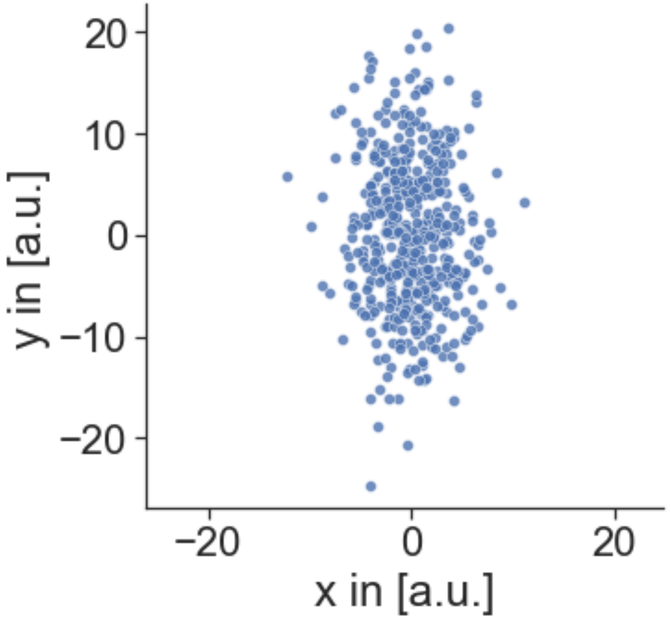
$$\{ E_1 , E_2 , \ldots , E_K \} \quad E_i = f(\{X\})$$

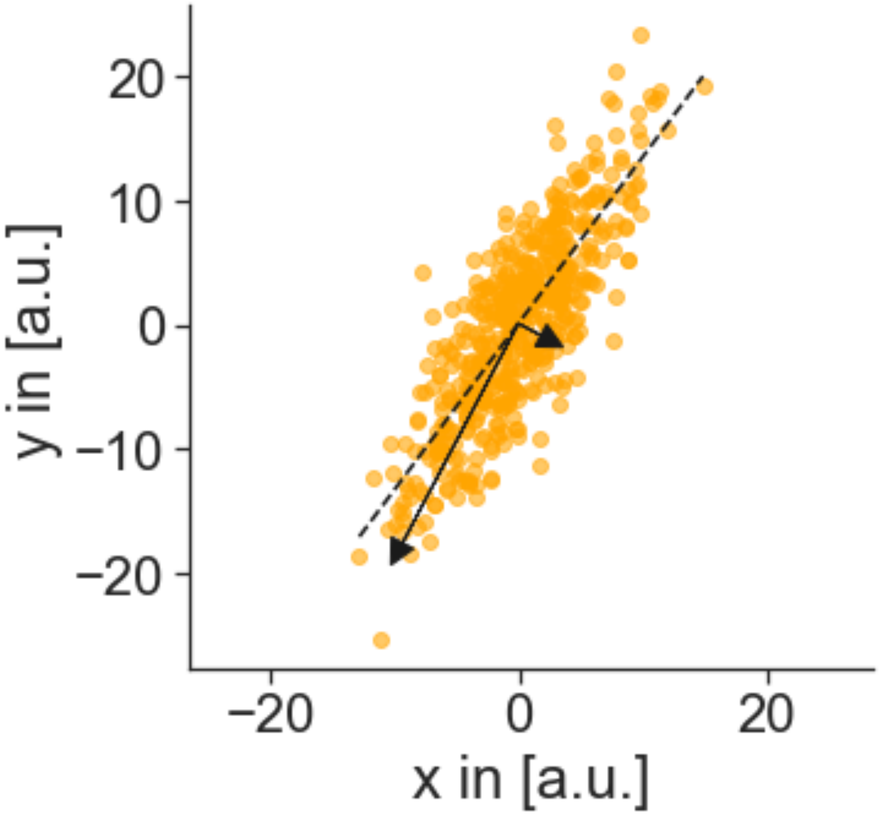
where $K < N$

Principal component analysis (PCA)

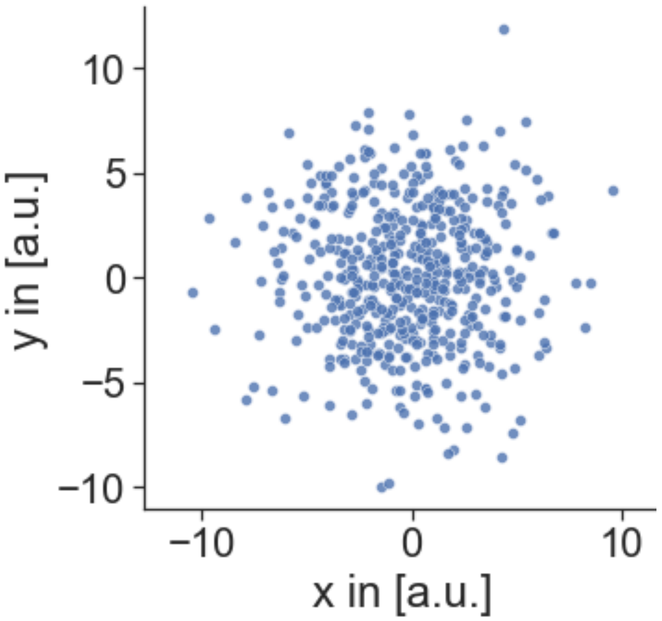
- Principle component analysis finds the **dominant component** in the feature space
- A regression finds the **correlation** of the data in feature space

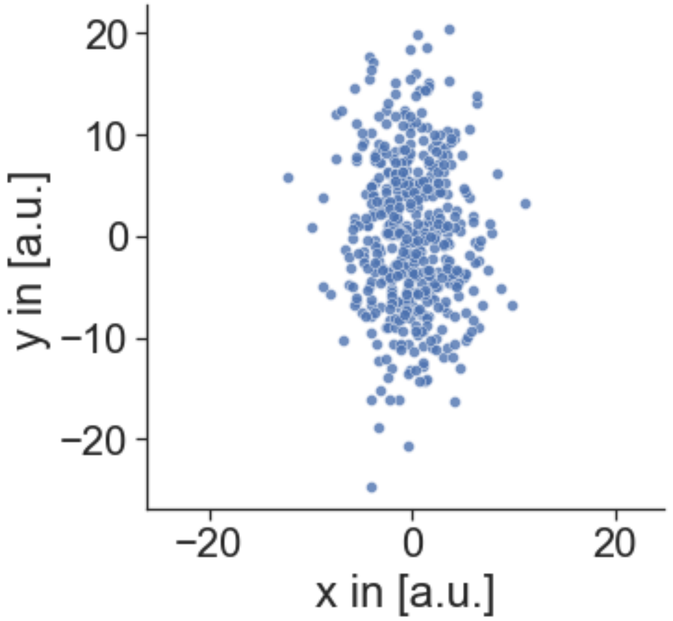


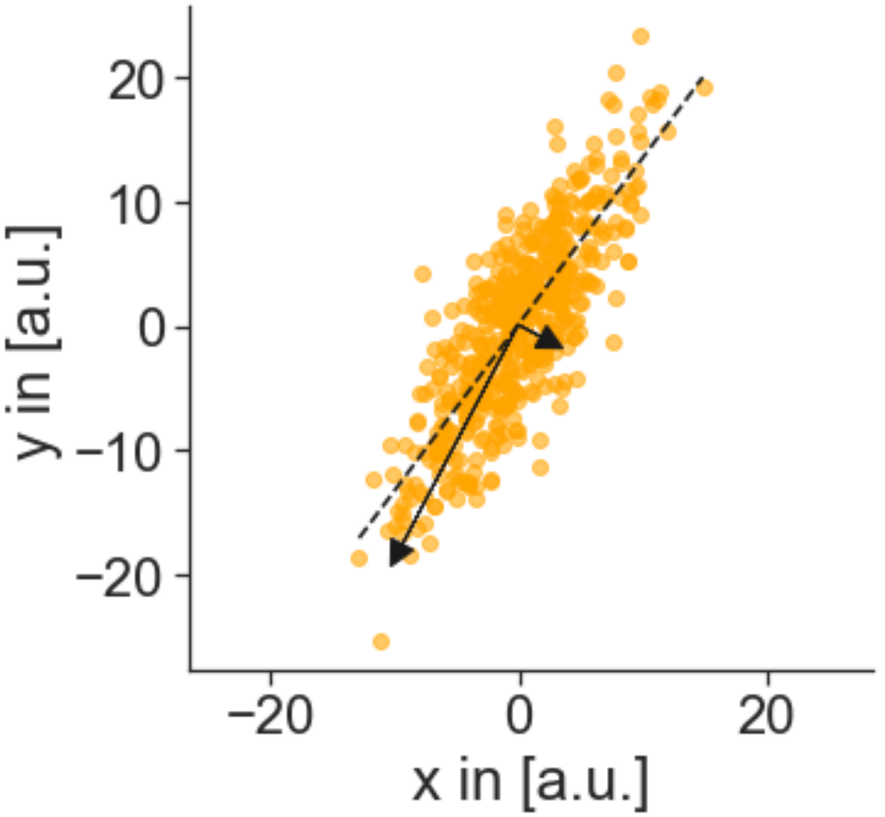




Principal component analysis (PCA)



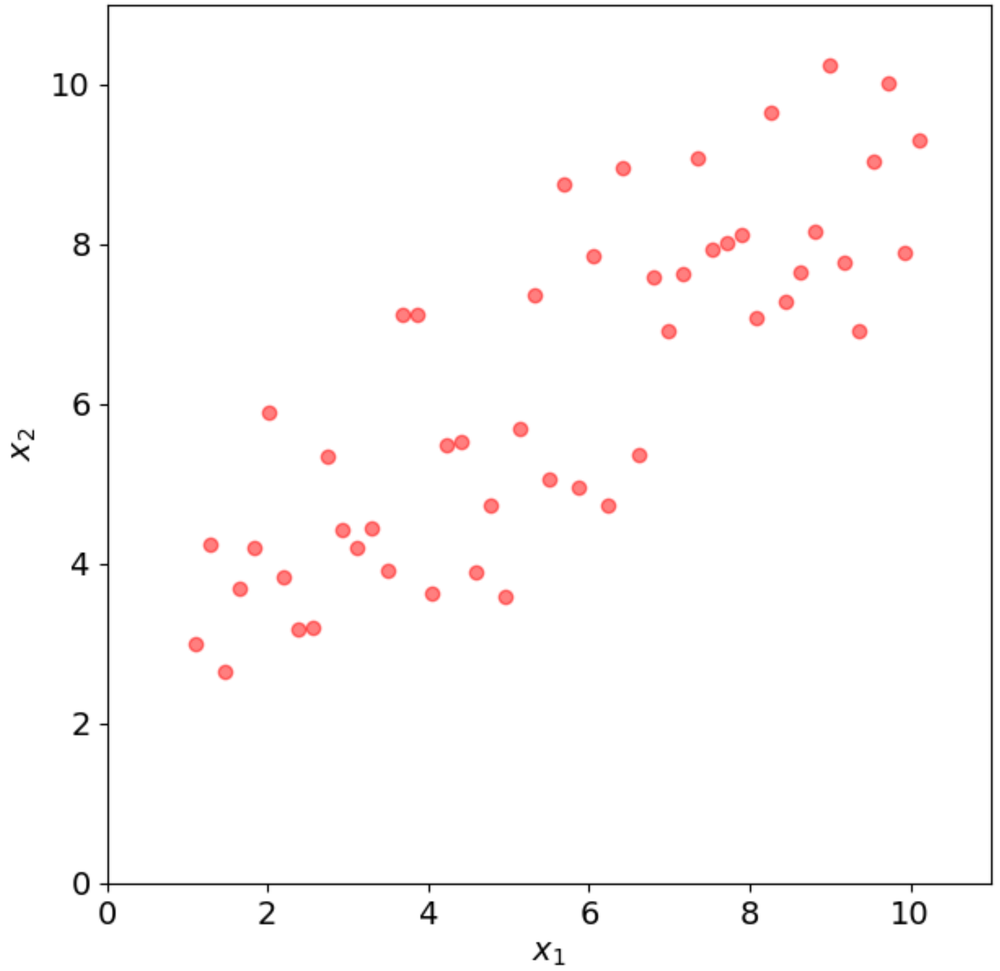




- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**

In PCA you solve an eigenvalue problem

Let ***X*** a dataset of *M* datapoints in *N* dimensions (here, *M*=50 and *N*=2)



- Center data:

$$\boldsymbol{X}' = \boldsymbol{X} - \mu$$

- Compute data covariance matrix ***C***:

$$c_{i,j} = \frac{1}{M} \sum_{k=1}^{M} \boldsymbol{x}'_i \boldsymbol{x}'_j$$

- Calculate eigenvalue decomposition:

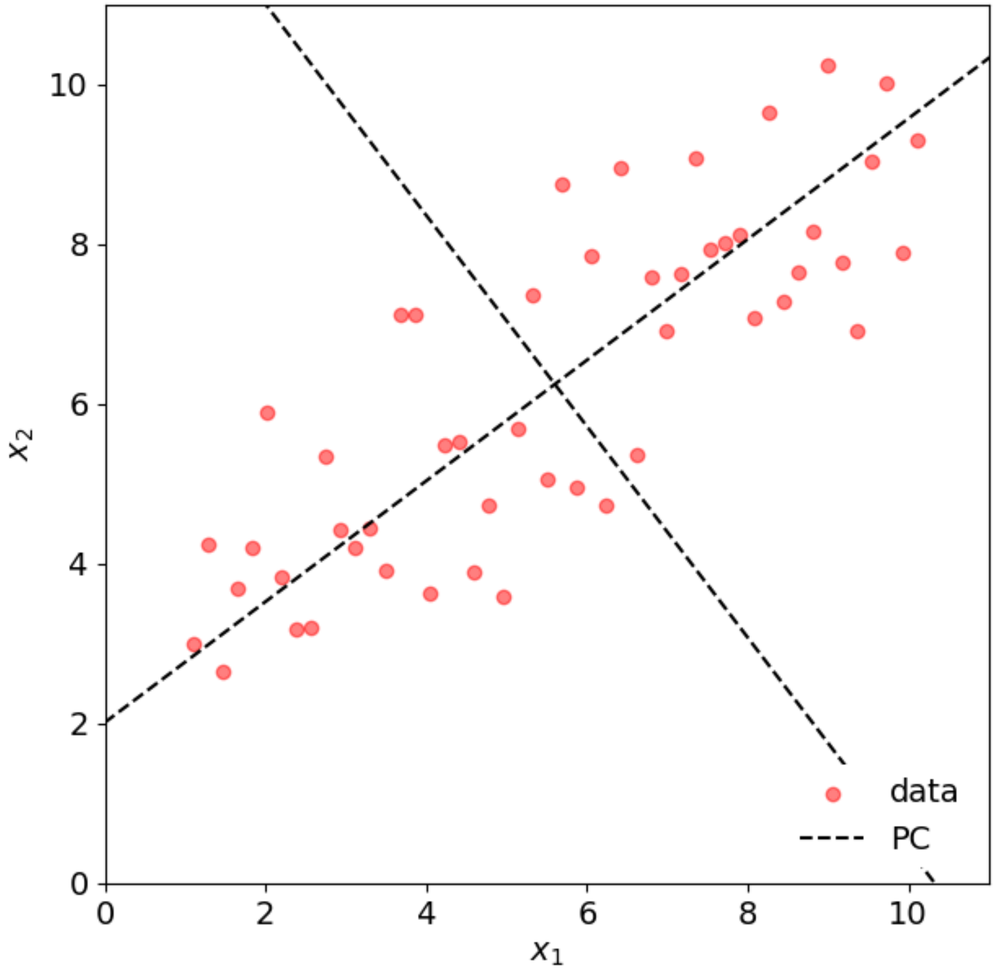
$$\mathbf{C} = \mathbf{W} \lambda \mathbf{W}^{-1}$$

*N*x*N* matrix of ***eigenvectors***

*N*x*N* diagonal matrix of ***eigenvalues***

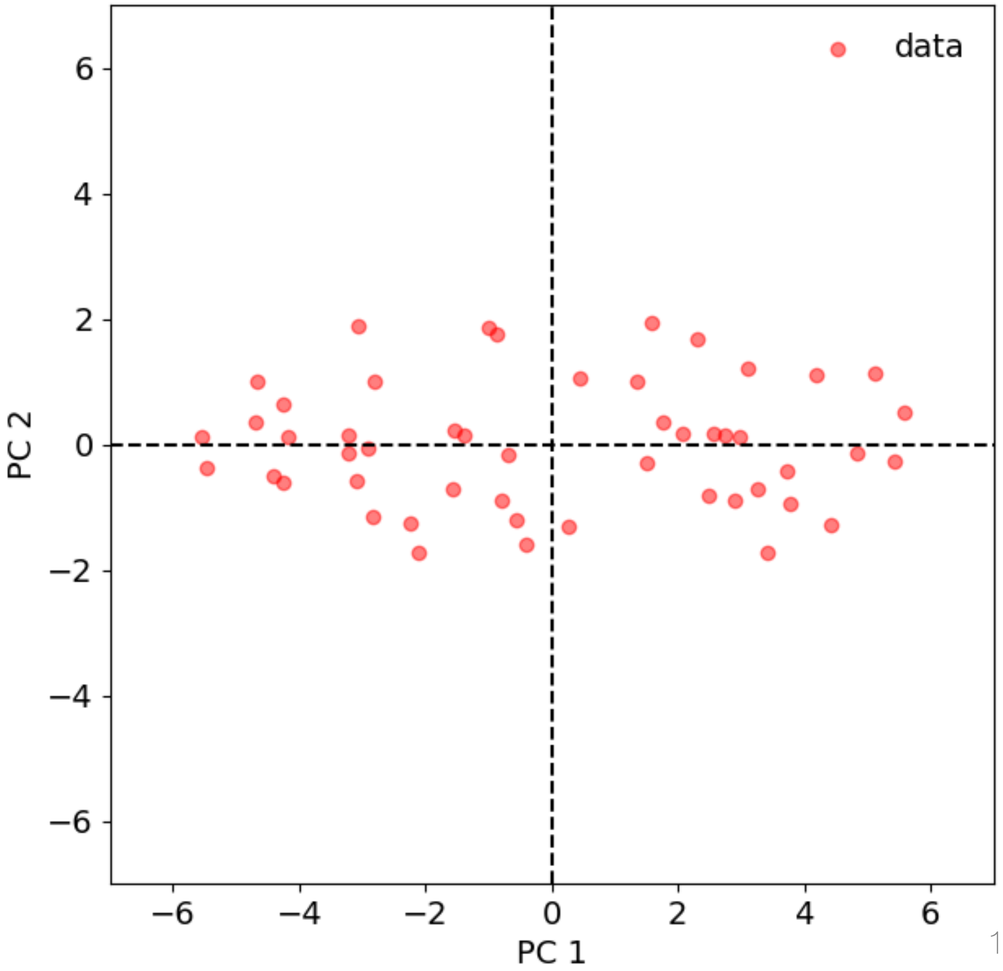
Projection into the eigenspace

Let **X** a dataset of *M* datapoints in *N* dimensions (here, *M*=50 and *N*=2)



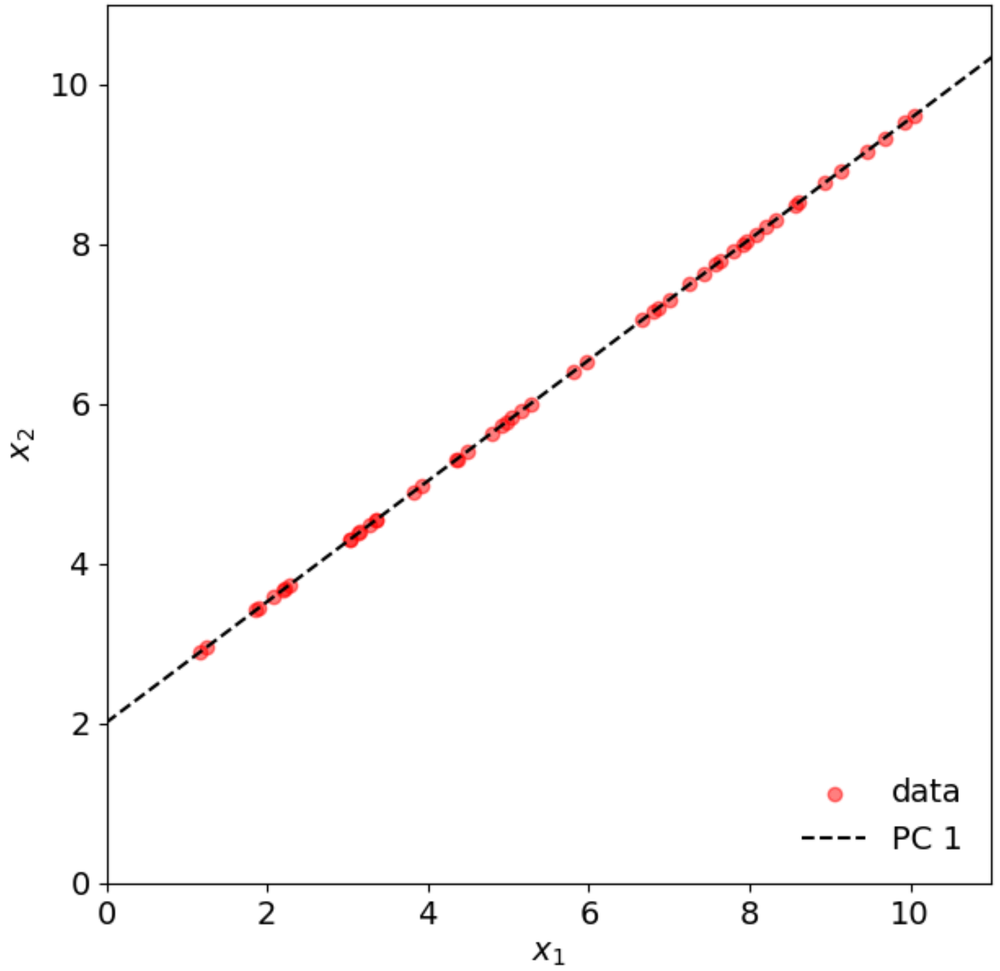
$$p = \mathbf{W}^{\mathrm{T}}(\mathbf{x} - \mu)$$

transform



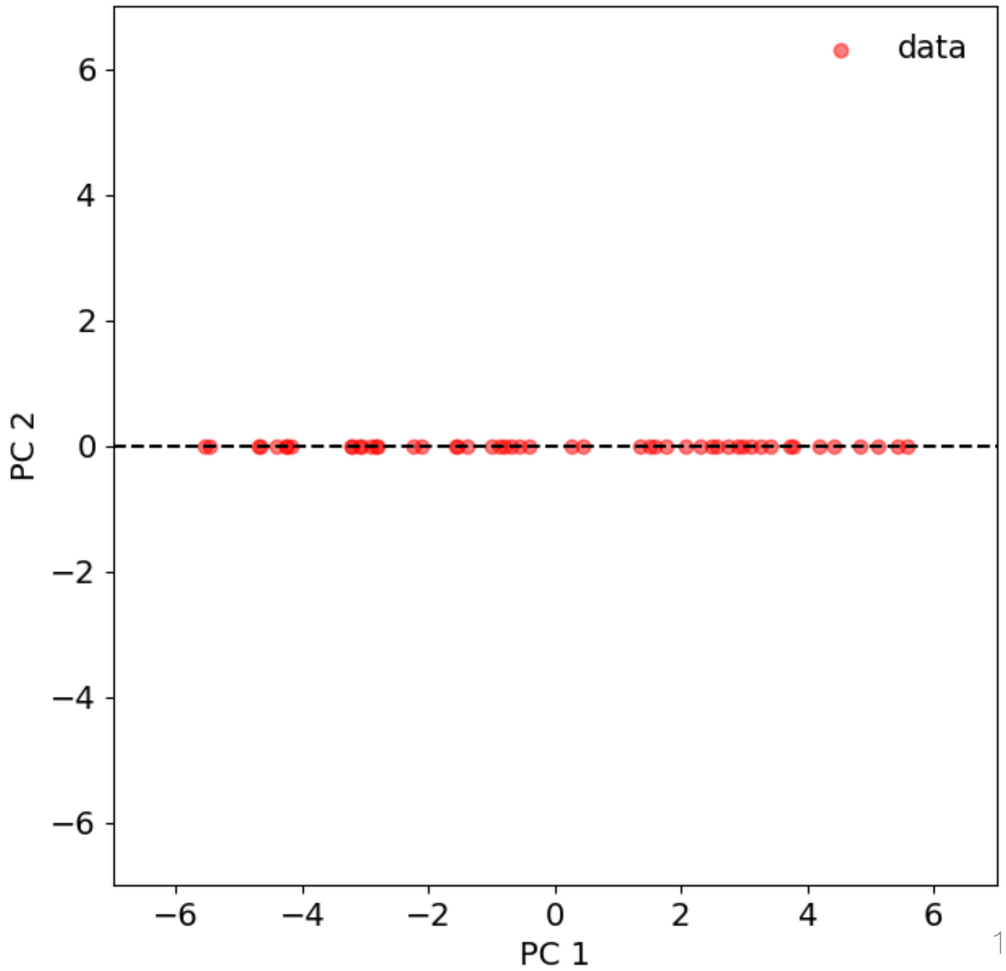
Dimensionality reduction

Remove dimensions that least contribute to data variance

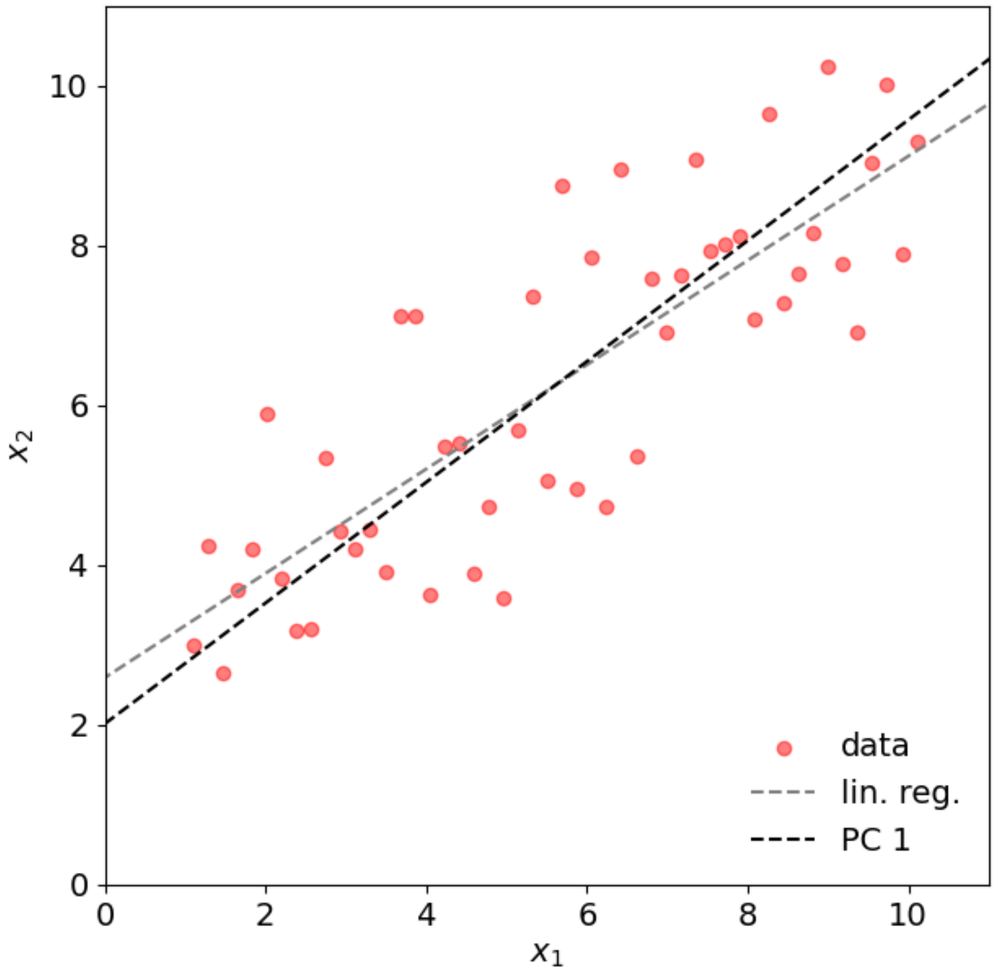


$$p = \mathbf{W}^{\mathrm{T}}(\mathbf{x} - \mu)$$

transform



First eigenvectors ≠ linear regression

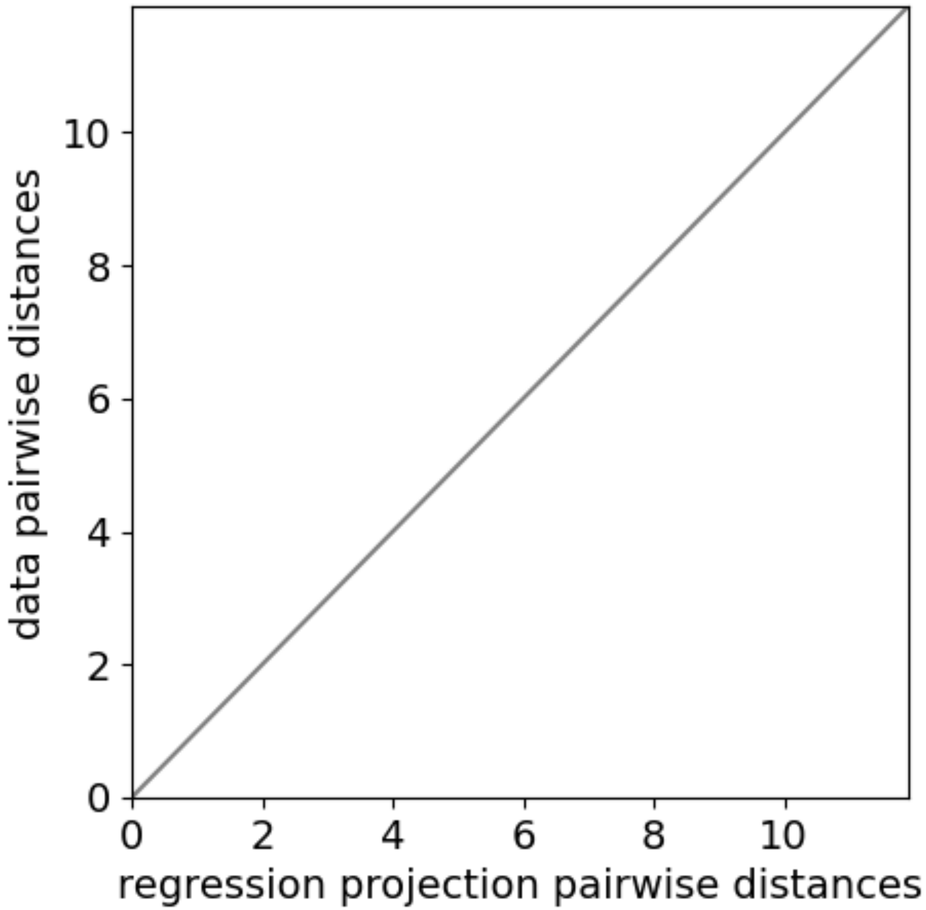


- Linear regression minimizes average distance of data points to trend line

- First eigenvector maximally preserves relative distance between data points

First eigenvectors ≠ linear regression

Projecting on principal components preserves data *structure*



Principal component analysis (PCA)

- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**

$\mathbf{C}(0)$ Is the covariance matrix of the data **X**

$$\begin{array}{c} \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{bmatrix} \mathbf{2.0} & \mathbf{0.8} \\ \mathbf{0.8} & \mathbf{0.6} \end{bmatrix} \end{array}$$

$$\mathrm{var}(a) = \frac{1}{n}\sum_{i=1}^{n} x_{ia}^2$$

$$\mathrm{cov}(a,b) = \frac{1}{n}\sum_{i=1}^{n} x_{ia}x_{ib}$$

Solving the generalised eigenvalue problem

$$\mathbf{C}(0)\mathbf{W} = \mathbf{W}\lambda$$

Gives an eigenvector matrix **W** that will allow the transform of the original data **X** onto a new basis **T** that maximises the variance.

$$\mathbf{T} = \mathbf{X}\mathbf{W}$$

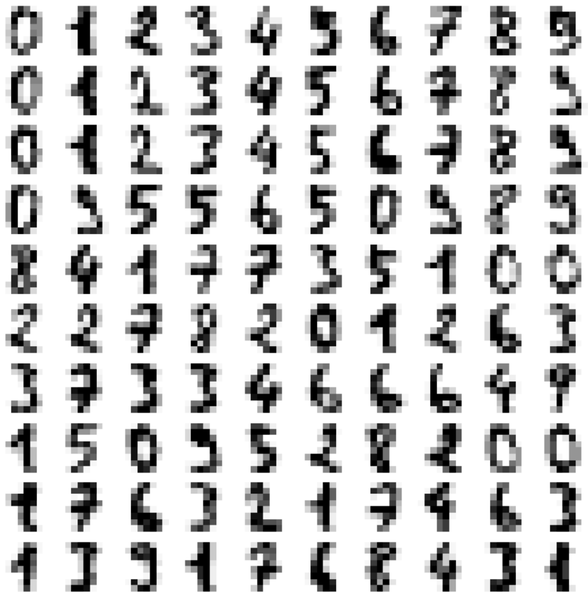
It is possible to choose m eigenvectors to project onto by only using the first m-columns of **W**.

These are the 'dominant' features!

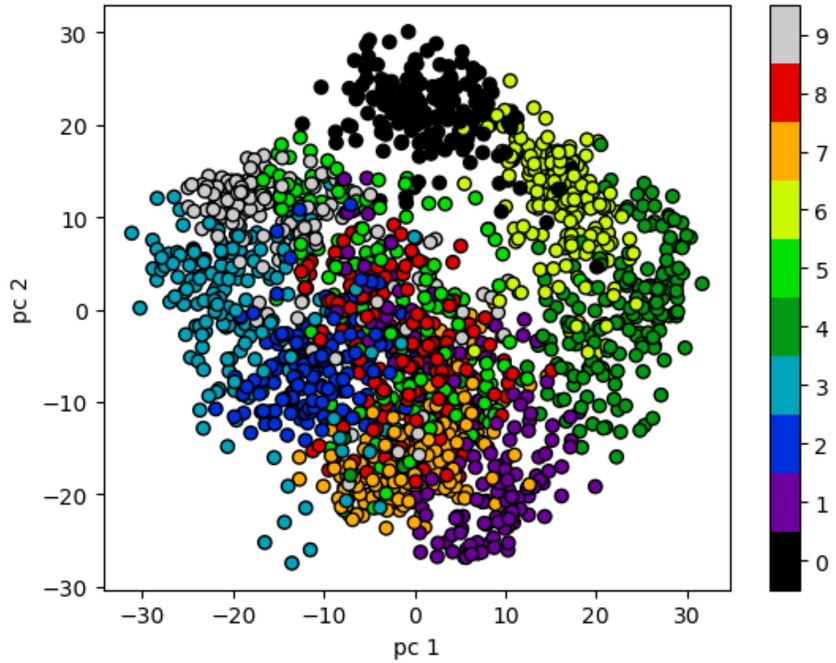
PCA example

- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**

MNIST database of written digits



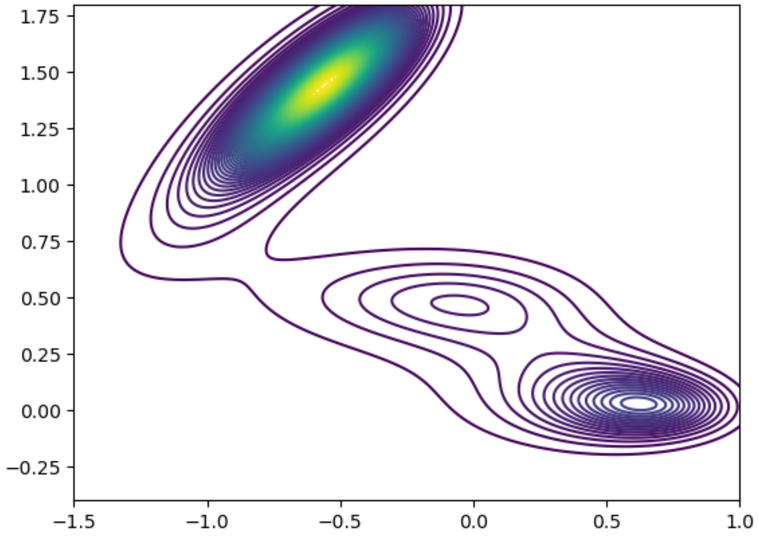
Input is a vector of 64 dimensions 8x8 pixel digits

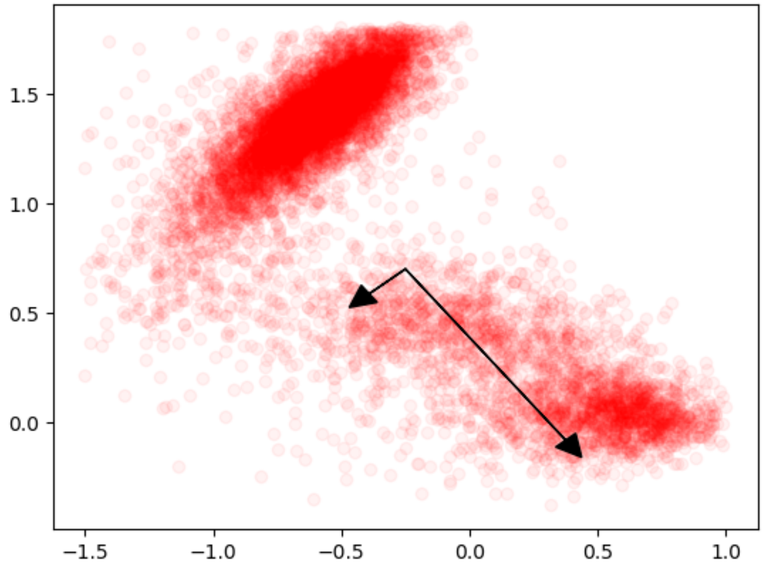


2 principal components manage to project a few of the digits in a similar area of space!

PCA example

- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**





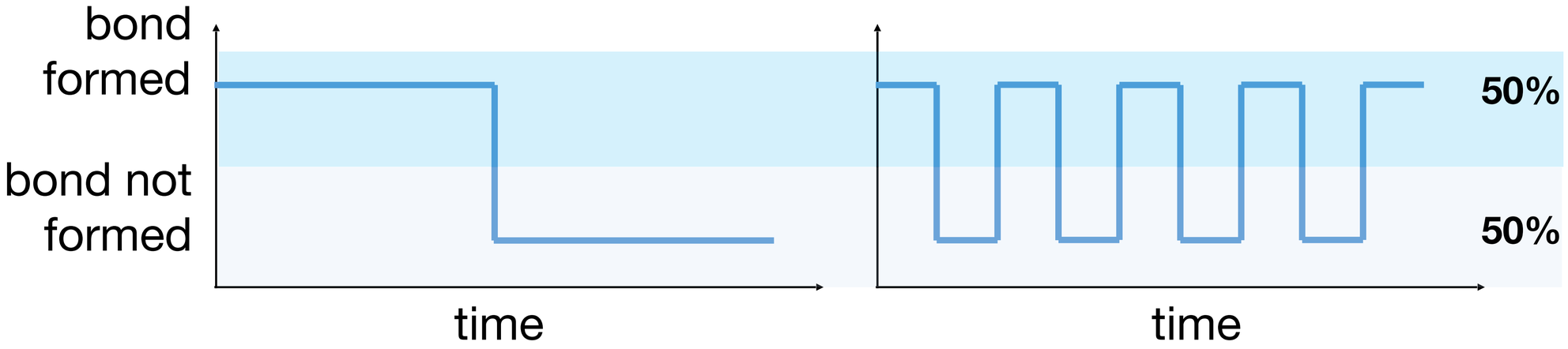
Finding the dominant reaction coordinate of a potential energy surface of a reaction!

Time-lagged independent component analysis (tICA)

tICA is a **linear transform** similar to PCA

The transform is chosen such that, amongst all linear transforms, tICA **maximizes the autocorrelation** of transformed coordinates.

Thought experiment: typically hydrogen bond is considered established if donor-acceptor distance <2.5 Å, and donor-acceptor-hydrogen angle <20°.



reporting % time a bond is established in simulation can be misleading!

Time-lagged independent component analysis (tICA)

tICA is a **linear transform** similar to PCA

The transform is chosen such that, amongst all linear transforms, tICA **maximizes the autocorrelation** of transformed coordinates.

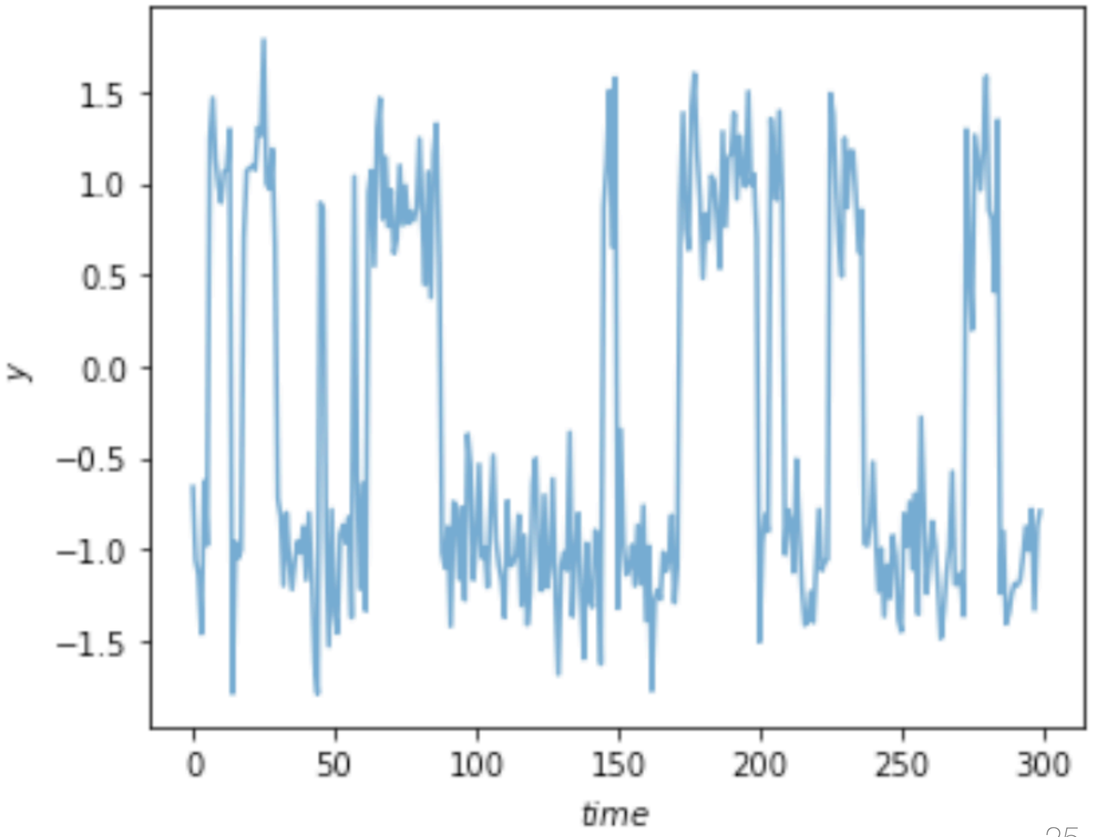
$\mathbf{r}(t) = (r_i(t))_{i=1,\ldots,D}$

D-dimensional input data vector that is mean free, i.e., $\mathbf{r}(t) = \mathbf{r}(t) - \langle \mathbf{r}(t) \rangle_t$

Computing the covariance of the data at $t = 0$ and $t = \tau$ which is the lag-time chosen

$c_{ij}(\tau) = \langle r_i(t) r_j(t+\tau) \rangle_t$

enables computing two covariance matrices: $\mathbf{C}(0)$ and $\mathbf{C}(\tau)$



F. Nüske, […], A. S. J. S. Mey and F. Noé: *JCTC*, 2014

G. Perez-Hernandez, et al., ***JCTC***, 2013

C. R. Schwantes et al., ***JCTC***, 2013

Time-lagged independent component analysis (tICA)

tICA is a **linear transform** similar to PCA

The transform is chosen such that, amongst all linear transforms, tICA **maximizes the autocorrelation** of transformed coordinates.

Entries of the covariance matrix can be computed as:

$$c_{ij}(\tau) = \frac{1}{N-\tau-1} \sum_{t=1}^{N-\tau} r_i(t) r_j(t+\tau)$$

$\mathbf{C}(0)$ will be a symmetric matrix. The symmetry of $\mathbf{C}(\tau)$ will need to be enforced with:

$$\mathbf{C}(\tau) = \frac{1}{2}(\mathbf{C}_d(\tau) + \mathbf{C}_d^{\mathsf{T}}(\tau))$$

We can now solve the generalised eigenvalue problem:

$$\mathbf{C}(\tau)\mathbf{U} = \mathbf{C}(0)\mathbf{U}\mathbf{\Lambda}$$

Eigenvector matrix containing ICs

Diagonal matrix with eigenvalues

$$\mathbf{z}^{\mathsf{T}}(t) = \mathbf{r}^{\mathsf{T}}(t)\mathbf{U}$$

M columns of full rank U for DR

F. Nüske, […], A. S. J. S. Mey and F. Noé: ***J*CTC**, 2014

G. Perez-Hernandez, et al., ***JCTC***, 2013

C. R. Schwantes et al., ***JCTC***, 2013

Non-linear methods for dimensionality reduction

MATTERS ARISING
https://doi.org/10.1038/s41587-020-00809-z
nature biotechnology
Check for updates

Initialization is critical for preserving global data structure in both *t*-SNE and UMAP

Dmitry Kobak[1] and George C. Linderman[2]

ARISING FROM Becht, E. et al. *Nature Biotechnology* https://doi.org/10.1038/nbt.4314 (2019)

One of the most ubiquitous analysis tools in single-cell transcriptomics and cytometry is *t*-distributed stochastic neighbor embedding (*t*-SNE)[1], which is used to visualize individual cells as points on a two-dimensional scatterplot such that similar cells are positioned close together[2]. A related algorithm, called uniform manifold approximation and projection (UMAP)[3], has attracted substantial attention in the single-cell community[4]. In *Nature Biotechnology*, Becht et al.[4] argued that UMAP is preferable to *t*-SNE because it better preserves the global structure of the data and is more consistent across runs. Here we show that this alleged superiority of UMAP can be entirely attributed to different choices of initialization in the implementations used by Becht et al.: the *t*-SNE implementations by default used random initialization, while the UMAP implementation used a technique called Laplacian eigenmaps (LE)[5] to initialize the embedding. We show that UMAP with random initialization preserves global structure as poorly as *t*-SNE with random initialization, while *t*-SNE with informative initialization performs as well as UMAP with informative initialization. On the basis of these observations, we argue that there is currently no evidence that the UMAP algorithm per se has any advantage over *t*-SNE in terms of preserving global structure. We also contend that these algorithms should always use informative initialization by default.

At the core of both *t*-SNE and UMAP are loss functions that make similar points attract each other and push dissimilar points away from each other. Both algorithms minimize their loss functions by using gradient descent. Gradient descent begins with some initial configuration of points, and with each iteration the points are moved to decrease the loss function. The specific implementations of these algorithms used by Becht et al. differed in how the initial configuration of points was chosen: the *t*-SNE implementations placed the points randomly, whereas the UMAP implementation used LE[5], an algorithm that can often achieve globally accurate embedding on its own[6]. The effect of this difference on how well the two algorithms preserve global structure was not discussed or investigated by Becht et al.

We can illustrate the importance of initialization for both algorithms using a simple toy dataset (Fig. 1). We sampled $n = 7{,}000$ points from a circle with some added Gaussian noise and used UMAP and *t*-SNE to construct an embedding. We kept all parameters for both algorithms at their default values and only changed the initialization. The *t*-SNE algorithm with random initialization produced a knot; UMAP with random initialization produced a tangled web with multiple tears. In both cases, the global structure was evidently not preserved; indeed, gradient descent in *t*-SNE and UMAP only pulls close neighbors together and is not much influenced by the global arrangement of points. At the same time, LE recovers the original circle for this toy dataset and, when used for initialization, strongly improves the UMAP result. A method often recommended for initialization in *t*-SNE is principal component analysis (PCA)[2]. Here PCA also recovers the original circle and strongly improves the *t*-SNE result. In both cases, only with informative initialization can UMAP and *t*-SNE produce a faithful representation of the closed one-dimensional manifold—the circle.

Using the code published by Becht et al., we analyzed the separate effects of initialization and algorithm on their results by adding UMAP with random initialization and *t*-SNE (using FIt-SNE[7]) with PCA initialization to the benchmarking comparison. Apart from the initialization, both algorithms were run with the same parameters as in Becht et al. We used all three datasets analyzed in the original publication (sample sizes from 320,000 to 820,000 cells)[8–10]. To quantify preservation of global structure, Becht et al. computed Pearson correlation between pairwise Euclidean distances in high-dimensional space and in the embedding. To quantify the reproducibility of the embedding, the authors embedded random subsamples of the data and measured the correlation of the coordinates of subsample embeddings with the coordinates of the full-dataset embeddings (up to symmetries around the coordinate axes). Our results show that *t*-SNE and UMAP with random initialization perform similarly poorly with regard to both metrics, whereas *t*-SNE and UMAP with PCA and LE initialization, respectively, perform similarly well (Table 1). See Extended Data Figs. 1–6 for the exact analogs of the original figures from Becht et al.

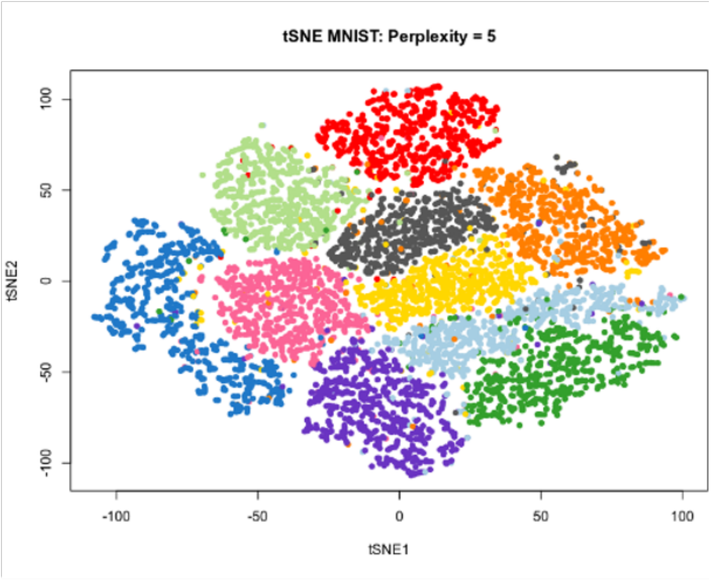
Becht et al. wrote that their findings were "consistent with the idea" that UMAP performs "optimizations that are sensitive to global features of the data, thus reaching similar arrangements more consistently." Our results show that this conclusion can be misleading: their findings were in fact not due to UMAP optimizations but rather to its initialization.

We have recently argued that, for single-cell transcriptomic data, *t*-SNE with PCA initialization produces more meaningful embeddings than *t*-SNE with random initialization[2]. The findings of Becht et al., when interpreted correctly, also underscore the importance of using informative initialization and suggest that it should be used as the default option in *t*-SNE and UMAP implementations. PCA initialization has always been the default in openTSNE[11], a Python reimplementation of FIt-SNE, and FIt-SNE v.1.2 now also uses it by default. OpenTSNE v.0.4 now also supports LE initialization. Importantly, *t*-SNE with non-random initialization should not be considered a new algorithm or even an extension of the original *t*-SNE; it is exactly the same algorithm with the same loss function, and almost any existing implementation trivially allows the use of any given initialization, including the PCA-based one.

[1]Institute for Ophthalmic Research, University of Tübingen, Tübingen, Germany. [2]Applied Mathematics Program, Yale University, New Haven, CT, USA.

- Useful for visualisation, project high-dimensional data in 2 or 3 dimensions.
- Controlled by one main parameters: “perplexity”
- Relative distance between points not quantitatively meaningful

MNIST: database of written digits



https://distill.pub/2016/misread-tsne/

T-distributed Stochastic Neighbour Embedding (t-SNE)

$$X = x_{i\,i=1}^{n}, \quad x_i \in \mathbb{R}^n \longrightarrow Y = y_{i\,i=1}^{n}, \quad y_i \in \mathbb{R}^m$$

1. Modelling high-dimensional space

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \qquad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

Perplexity: User-defined parameter that influences determination of σ.

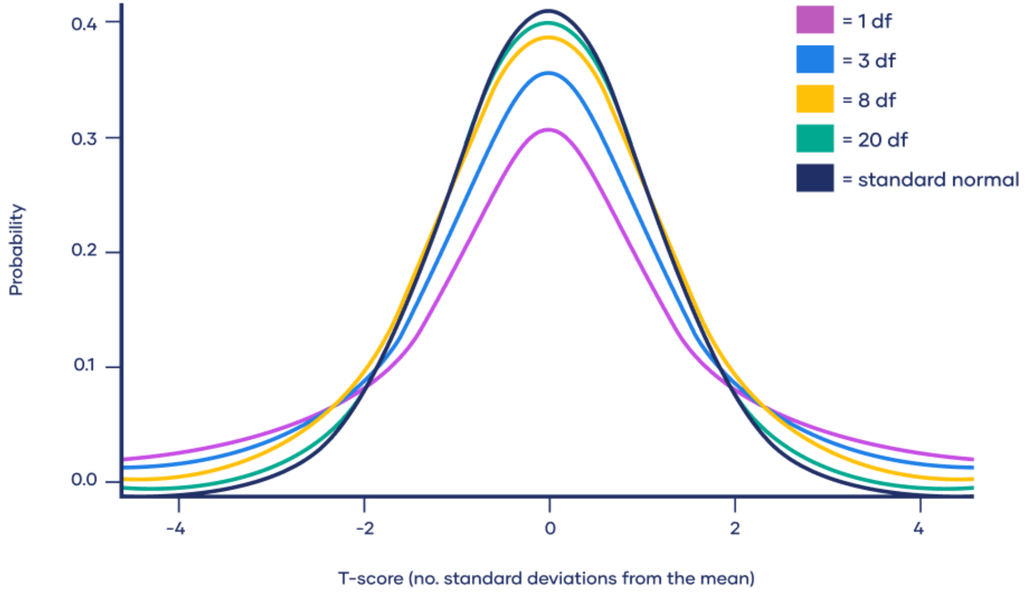
- Low Perplexity (σ_i^2 small): Focuses on local structures
- High Perplexity (σ_i^2 large): Incorporates more distant points

2. Modelling low-dimensional space

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

3. Minimize KL-divergence between P and Q

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$



Uniform manifold approximation and projection (UMAP)

UMAP constructs a high-dimensional graph representation of the data and then optimizes a low-dimensional graph to be as structurally similar as possible.

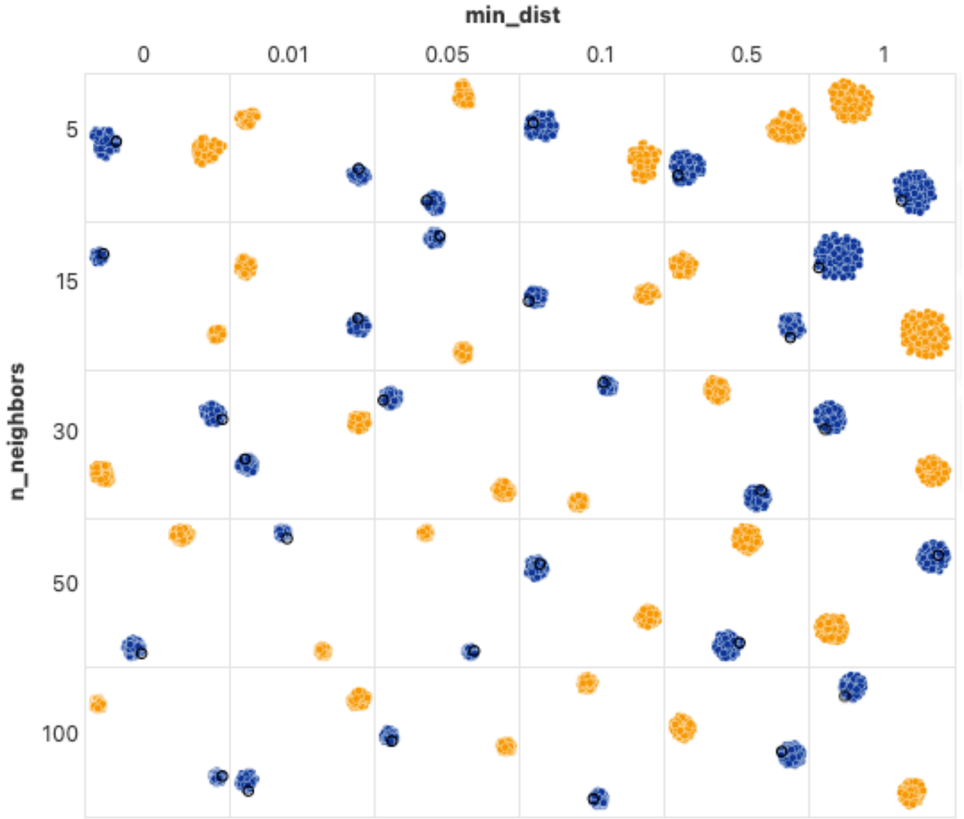
1. Construct a weighted graph in high-dimensional space

$$w_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma}\right) & \text{if } x_j \in \text{kNN}(x_i) \\ 0 & \text{otherwise} \end{cases}$$

2. Optimize to approximate the topological structure of the high-dimensional graph

$$C = \sum_{i,j} \left[w_{ij} \log\left(\frac{w_{ij}}{v_{ij}}\right) + (1 - w_{ij}) \log\left(\frac{1 - w_{ij}}{1 - v_{ij}}\right) \right]$$

- *n_neighbors:* local vs. global structure
- *min_distance*: how close points are allowed
- *metric*: quantify the similarity



Two clusters with equal numbers of points.
Points Per Cluster: 100; **Dimensions:** 50